

Ahmedabad Institute of Technology

CE & IT Department

Hand Book

WEB Programming(3160713)

Year: 2020-2021

Prepared By: Dr. Ajay N. Upadhyaya, HOD CE & IT Dept., AIT &
Prof. Heena Panjwani, Asst. Prof. CE & IT Dept, AIT

Index

Sr. No.	Content	Page No.
1	Introduction to WEB	
	1.1 Basics of WWW	04
	1.2 HTTP Protocol	14
	1.3 Client Server Architecture	24
2	Concepts of effective web design	
	2.1 Web design issues including Browser, Bandwidth and Cache, Display resolution, Look and Feel of the Website	28
	2.2 Page Layout and linking, User centric design, Sitemap, Planning and publishing website,	30
	2.3 Designing effective navigation	32
	2.4 Web Browser Architecture	35
	2.5 N-Tier Architecture	38
3	Basics of HTML	42
4	CSS	84
5	Bootstrap	137

6	Client Side Scripting using JavaScript	158

1: Introduction to WEB

1.1 Basics of WWW

World Wide Web, which is also known as a Web, is a collection of websites or web pages stored in web servers and connected to local computers through the internet. These websites contain text pages, digital images, audios, videos, etc. Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, cell phones, etc. The WWW, along with the internet, enables the retrieval and display of text and media to your device.



The building blocks of the Web are web pages which are formatted in HTML and connected by links called "hypertext" or hyperlinks and accessed by HTTP. These links are electronic connections that link related pieces of information so that users can access the desired information quickly. Hypertext offers the advantage to select a word or phrase from text and thus to access other pages that provide additional information related to that word or phrase.

Uniform Resource Locator (URL)

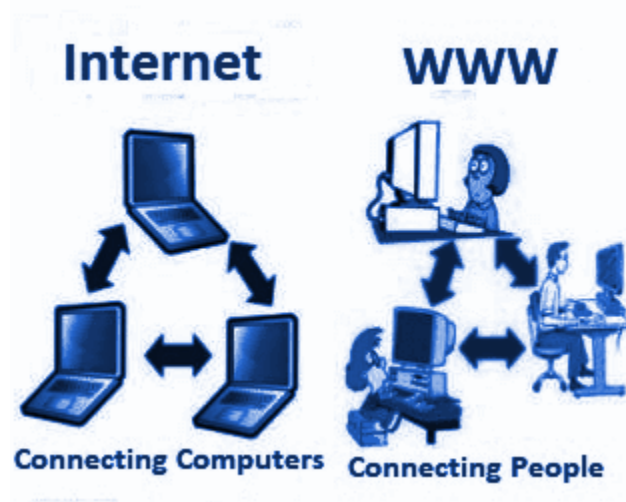
A web page is given an online address called a Uniform Resource Locator (URL). A particular collection of web pages that belong to a specific URL is called a website, e.g., www.facebook.com, www.google.com, etc. So, the World Wide Web is like a huge electronic book whose pages are stored on multiple servers across the world.

Small websites store all of their WebPages on a single server, but big websites or organizations place their WebPages on different servers in different countries so that when users of a country search their site they could get the information quickly from the nearest server.

So, the web provides a communication platform for users to retrieve and exchange information over the internet. Unlike a book, where we move from one page to another in a sequence, on the World Wide Web we follow a web of hypertext links to visit a web page and from that web page to move to other web pages. You need a browser, which is installed on your computer, to access the Web.

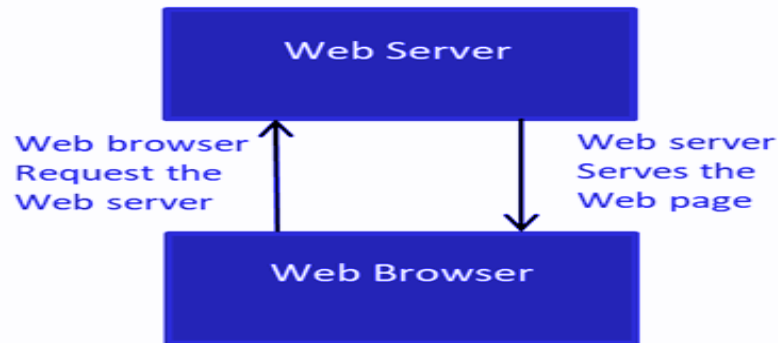
Difference between World Wide Web and Internet:

Some people use the terms 'internet' and 'World Wide Web' interchangeably. They think they are the same thing, but it is not so. The Internet is entirely different from WWW. It is a worldwide network of devices like computers, laptops, tablets, etc. It enables users to send emails to other users and chat with them online. For example, when you send an email or chat with someone online, you are using the internet.

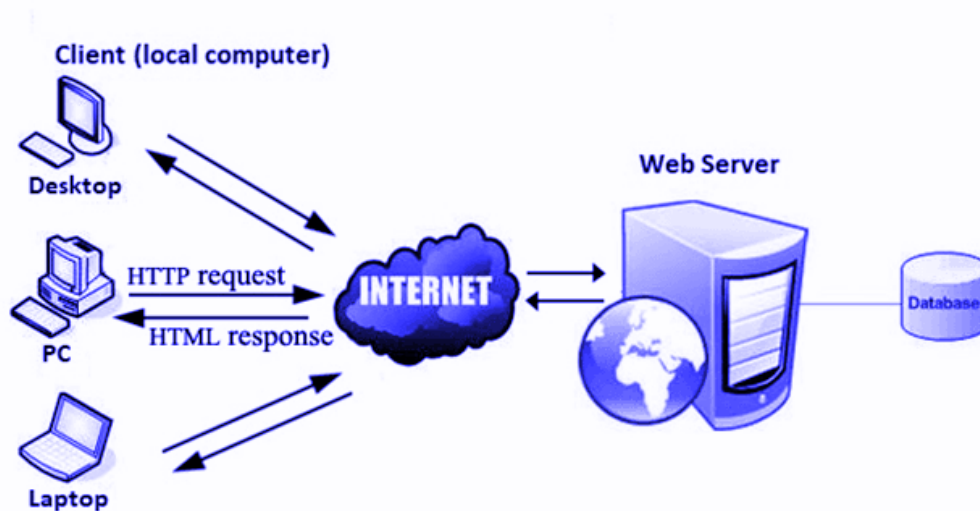


How the World Wide Web Works?

Now, we have understood that WWW is a collection of websites connected to the internet so that people can search and share information. Now, let us understand how it works!



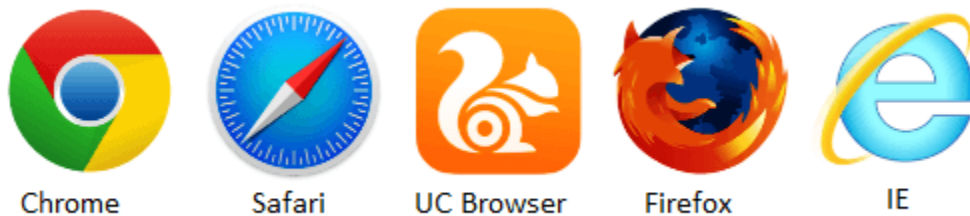
The Web works as per the internet's basic client-server format as shown in the following image. The servers store and transfer web pages or information to user's computers on the network when requested by the users. A web server is a software program which serves the web pages requested by web users using a browser. The computer of a user who requests documents from a server is known as a client. Browser, which is installed on the user's computer, allows users to view the retrieved documents.



All the websites are stored in web servers. Just as someone lives on rent in a house, a website occupies a space in a server and remains stored in it. The server hosts the website whenever a user requests its WebPages, and the website owner has to pay the hosting price for the same.

The moment you open the browser and type a URL in the address bar or search something on Google, the WWW starts working. There are three main technologies involved in transferring information (web pages) from servers to clients (computers of users). These technologies include Hypertext Markup Language (HTML), Hypertext Transfer Protocol (HTTP) and Web browsers.

Web Browser:



A web browser, which is commonly known as a browser, is a program that displays text, data, pictures, videos, animation, and more. It provides a software interface that allows you to click hyperlink resources on the World Wide Web. When you double click the Browser icon installed on your computer to launch it, you get connected to the World Wide Web and can search Google or type a URL into the address bar.

In the beginning, browsers were used only for browsing due to their limited potential. Today, they are more advanced; along with browsing you can use them for e-mailing, transferring multimedia files, using social media sites, and participating in online discussion groups and more. Some of the commonly used browsers include Google Chrome, Mozilla Firefox, Internet Explorer, Safari, and more.

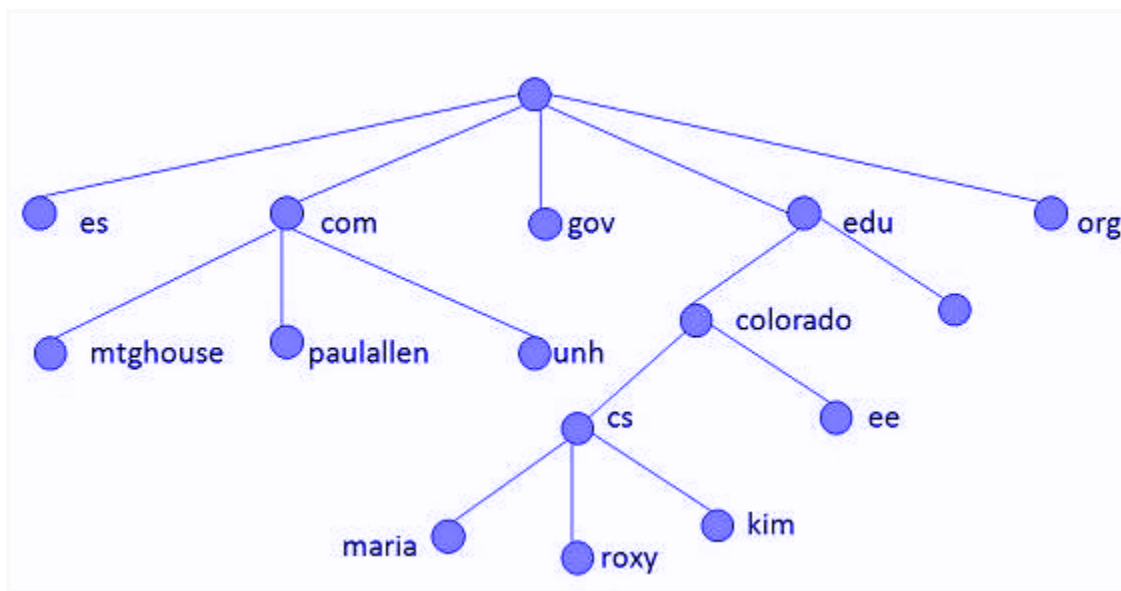
WWW Overview

WWW stands for World Wide Web. A technical definition of the World Wide Web is : all the resources and users on the Internet that are using the Hypertext Transfer Protocol (HTTP).

A broader definition comes from the organization that Web inventor Tim Berners-Lee helped found, the World Wide Web Consortium (W3C).The World Wide Web is the universe of network-accessible information, an embodiment of human knowledge.

In simple terms, The World Wide Web is a way of exchanging information between computers on the Internet, tying them together into a vast collection of interactive multimedia resources.

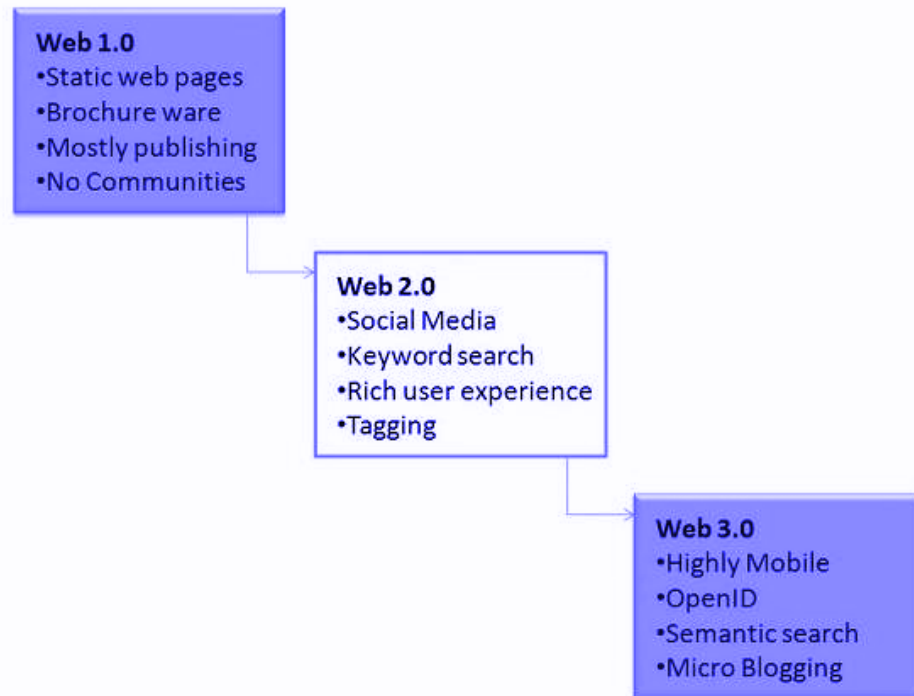
The Internet and Web are not the same thing: Web uses the internet to pass over the information.



Evolution

World Wide Web was created by Timothy Berners Lee in 1989 at CERN in Geneva. The World Wide Web came into existence as a proposal by him, to allow researchers to work together effectively and efficiently at CERN. Eventually it became the World Wide Web.

The following diagram briefly defines evolution of World Wide Web:



Web 1.0

It is the “readable” phrase of the World Wide Web with flat data. In Web 1.0, there is only limited interaction between sites and web users. Web 1.0 is simply an information portal where users passively receive information without being given the opportunity to post reviews, comments, and feedback.

Web 2.0

It is the “writable” phrase of the World Wide Web with interactive data. Unlike Web 1.0, Web 2.0 facilitates interaction between web users and sites, so it allows users to interact more freely with each other. Web 2.0 encourages participation, collaboration, and

information sharing. Examples of Web 2.0 applications are Youtube, Wiki, Flickr, Facebook, and so on.

Web 3.0

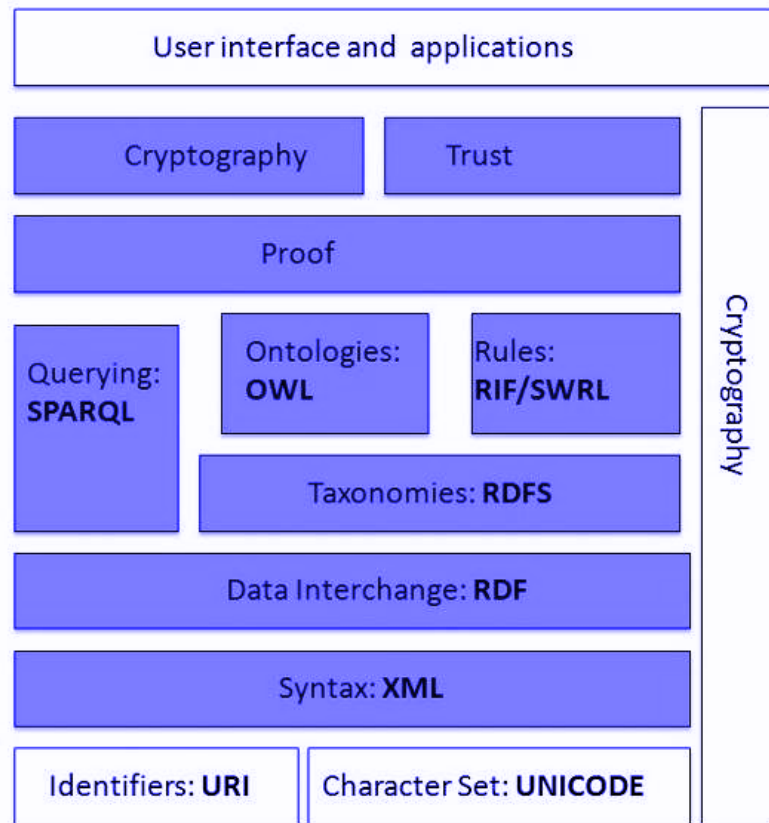
It is the “executable” phrase of Word Wide Web with dynamic applications, interactive services, and “machine-to-machine” interaction. Web 3.0 is a semantic web which refers to the future. In Web 3.0, computers can interpret information like humans and intelligently generate and distribute useful content tailored to the needs of users.

Comparison:

<u>Web 1.0</u>	<u>Web 2.0</u>	<u>Web 3.0</u>
Mostly Read-Only	Wildly Read-Write	Portable and Personal
Company Focus	Community Focus	Individual Focus
Home Pages	Blogs / Wikis	Live-streams / Waves
Owning Content	Sharing Content	Consolidating Content
Web Forms	Web Applications	Smart Applications
Directories	Tagging	User Behavior
Page Views	Cost Per Click	User Engagement
Banner Advertising	Interactive Advertising	Behavioral Advertising
Britannica Online	Wikipedia	The Semantic Web
HTML/Portals	XML / RSS	RDF / RDFS / OWL

WWW Architecture

WWW architecture is divided into several layers as shown in the following diagram:



Identifiers and Character Set

Uniform Resource Identifier (URI) is used to uniquely identify resources on the web and UNICODE makes it possible to build web pages that can be read and written in human languages.

Syntax

XML (Extensible Markup Language) helps to define common syntax in semantic web.

Data Interchange

The Resource Description Framework (RDF)

The Resource Description Framework (RDF) framework helps in defining core representation of data for the web. RDF represents data about resources in graph form.

Taxonomies

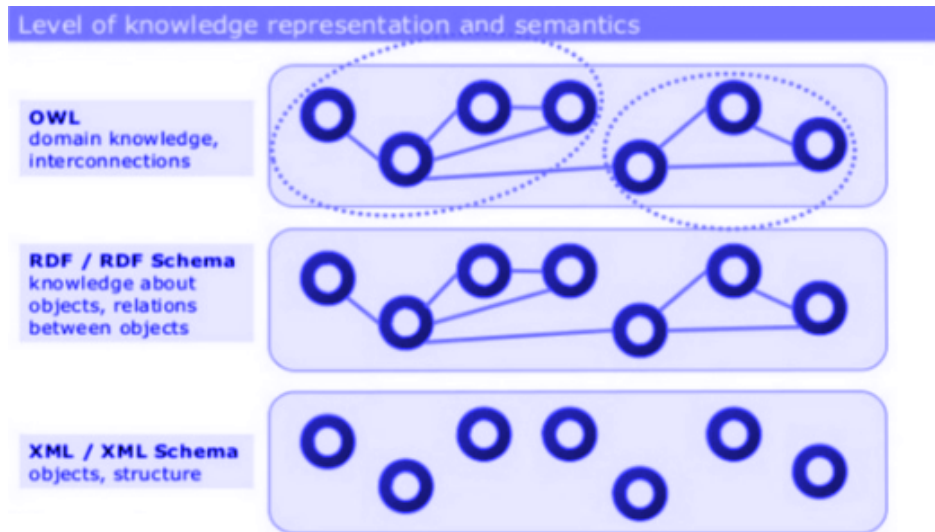
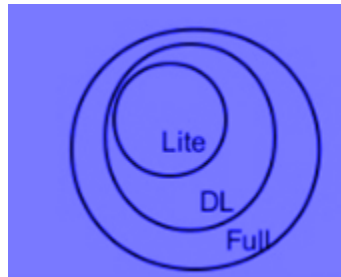
RDF Schema (RDFS)

RDF Schema (RDFS) allows more standardized description of taxonomies and other ontological constructs.

Ontologies

Ontology is about the exact description of things and their relationship. Web Ontology Language (OWL) offers more constructs over RDFS. It comes in following three versions:

- OWL Lite for taxonomies and simple constraints.
- OWL DL for full description logic support.
- OWL for more syntactic freedom of RDF



Rules

RIF and SWRL offer rules beyond the constructs that are available from RDFs and OWL. Simple Protocol and RDF Query Language (SPARQL) is SQL like language used for querying RDF data and OWL Ontologies.

Proof

All semantic and rules that are executed at layers below Proof and their result will be used to prove deductions.

Cryptography

Cryptography means such as digital signature for verification of the origin of sources is used.

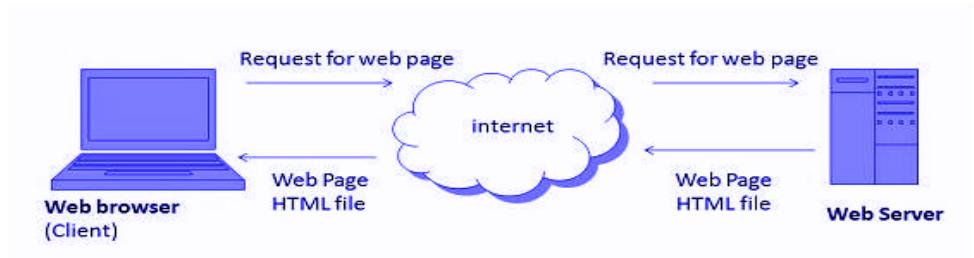
User Interface and Applications

On the top of layer User interface and Applications layer is built for user interaction.

WWW Operation

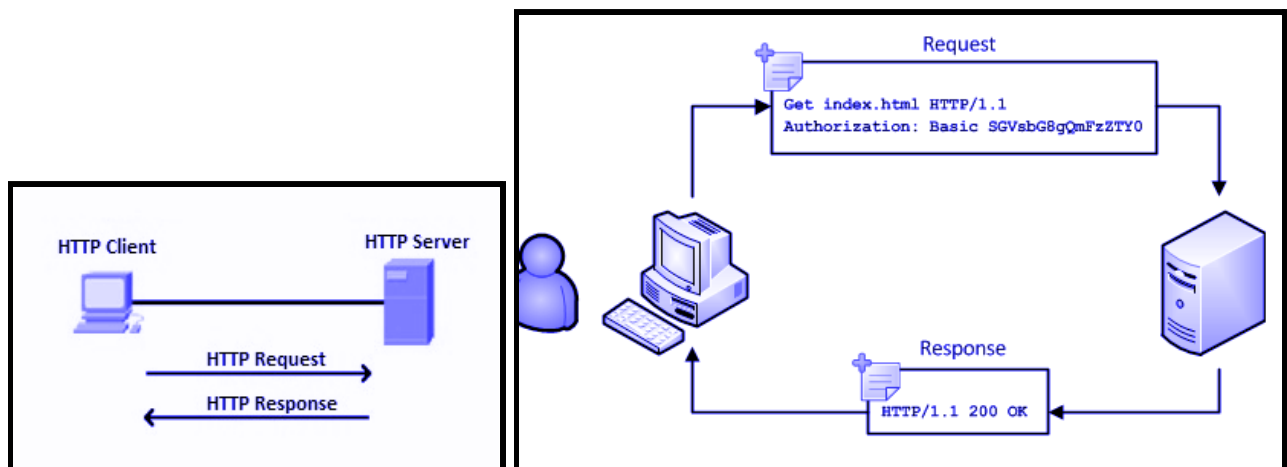
WWW works on a client- server approach. Following steps explains how the web works:

1. User enters the URL (say, <http://www.tutorialspoint.com>) of the web page in the address bar of the web browser.
2. Then the browser requests the Domain Name Server for the IP address corresponding to www.tutorialspoint.com.
3. After receiving the IP address, the browser sends the request for the web page to the web server using HTTP protocol which specifies the way the browser and web server communicates.
4. Then the web server receives a request using HTTP protocol and checks its search for the requested web page. If found it returns it back to the web browser and closes the HTTP connection.
5. Now the web browser receives the web page, It interprets it and displays the contents of the web page in the web browser's window.



1.2 Hypertext Transfer Protocol (HTTP):

HTTP (Hypertext Transfer Protocol) is a set of rules that runs on top of the TCP/IP suite of protocols and defines how files are to be transferred between clients and servers on the World Wide Web. HyperText Transfer Protocol (HTTP) is an application layer protocol which enables WWW to work smoothly and effectively. It is based on a client-server model. The client is a web browser which communicates with the web server which hosts the website. This protocol defines how messages are formatted and transmitted and what actions the Web Server and browser should take in response to different commands. When you enter a URL in the browser, an HTTP command is sent to the Web server, and it transmits the requested Web Page.

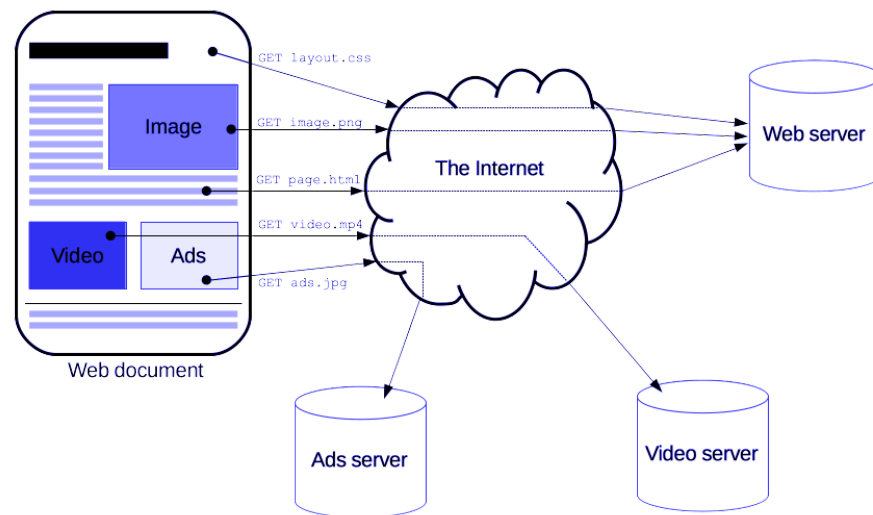


When we open a website using a browser, a connection to the web server is opened, and the browser communicates with the server through HTTP and sends a request. HTTP is

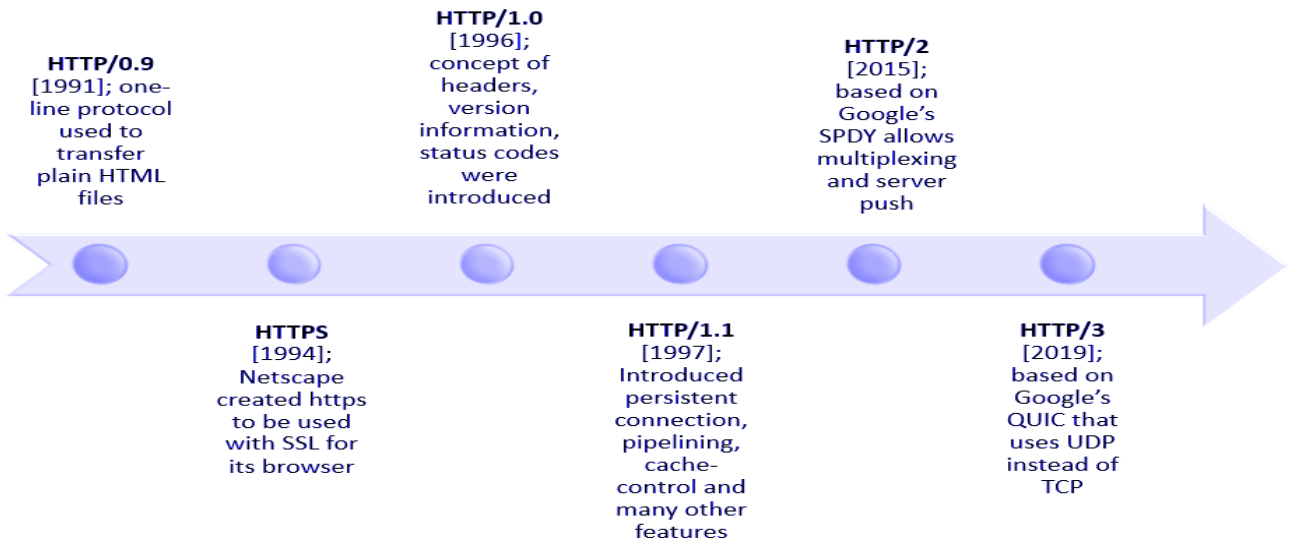
carried over TCP/IP to communicate with the server. The server processes the browser's request and sends a response, and then the connection is closed. Thus, the browser retrieves content from the server for the user.

Basics of HTTP Protocol

- HTTP is client-server network protocol.
- It has been used by the world wide web since 1990.
- It is based on a request response paradigm.
- **HTTP** is a protocol which allows the fetching of resources, such as HTML documents.
- It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser.
- A complete document is reconstructed from the different sub-documents fetched, for instance text, layout description, images, videos, scripts, and more.



Different Version of HTTP



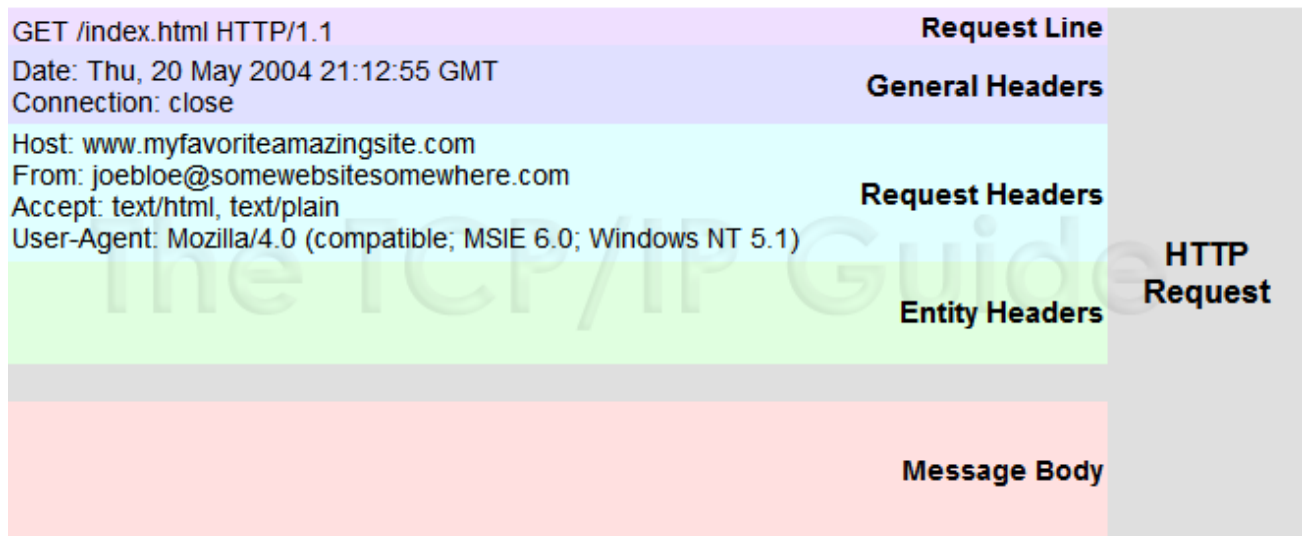
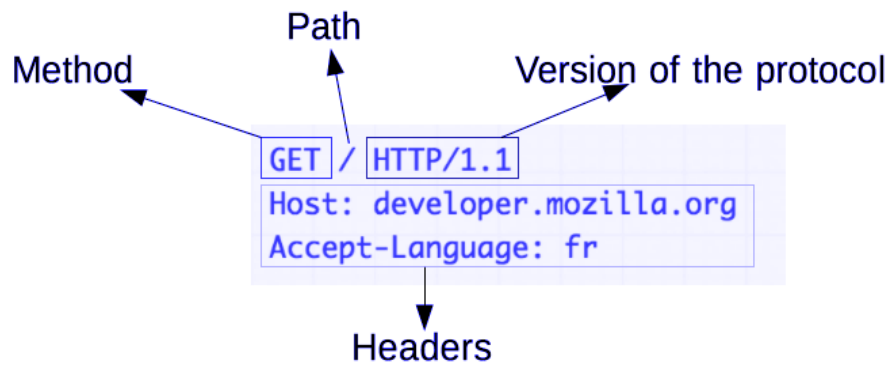
Differentiator	HTTP/1.0	HTTP/1.1	HTTP/2
Year	1991	1997	2015
Key Features	<p>For every TCP connection there is only one request and one response.</p> <p>HTTP/1.0</p>	<p>It supports connection reuse i.e. for every TCP connection there could be multiple requests and responses, and pipelining where the client can request several resources from the server at once. However, pipelining was hard to implement due to issues such as head-of-line blocking and was not a feasible solution.</p> <p>HTTP/1.1</p>	<p>Uses multiplexing, where over a single TCP connection resources to be delivered are interleaved and arrive at the client almost at the same time. It is done using streams which can be prioritized, can have dependencies and individual flow control. It also provides a feature called server push that allows the server to send data that the client will need but has not yet requested.</p> <p>HTTP/2</p>

Ref: <https://cheapsslsecurity.com/p/http2-vs-http1/>

HTTP Request Message

Structure

- <start line>
- <Header field>
- <blank Line>
- <Message Body>



HTTP Request Message

Structure

A) Start Line

Consist of three parts.

- 1) Request Method
- 2) Request URI
- 3) HTTP Version

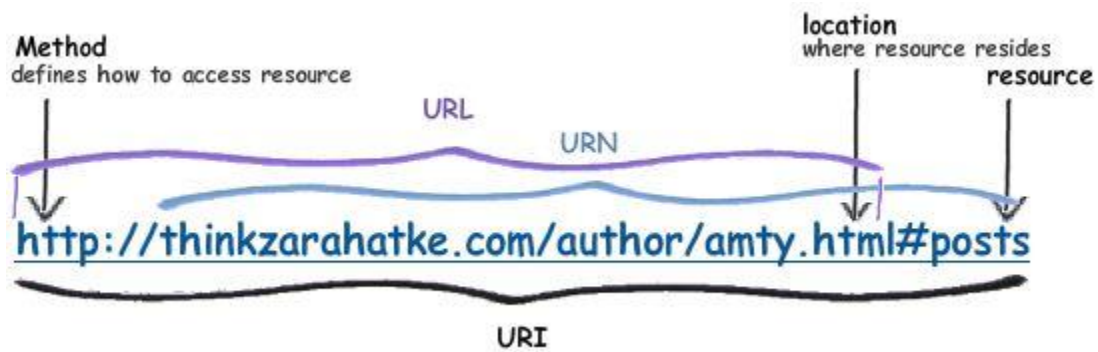
1) Request Method:

- It is always written in upper case letters.

- Primary method in HTTP is GET.
- The GET method is used when-
 - ❑ You type the URL in the address bar.
 - ❑ When you click the hyperlink which is present in the document.
 - ❑ When browsers download images for display within an HTML document.
- Another commonly used method is POST.
- POST is used to send information collected from a user form.
- Various methods used by HTTP is given below:
 - ❑ **HEAD** Same as GET, but transfers the status line and header section only.
 - ❑ **OPTIONS** Describes the communication options for the target resource.
 - ❑ **PUT** Replaces all current representations of the target resource with the uploaded content.
 - ❑ **DELETE** This method is useful in deleting the specific resources.
 - ❑ **TRACE** when request is made using a trace method the server echoes back the received request so that a client can see what intermediate servers are adding or changing in the request.

2) Request URI

- The uniform resource identifier (URI) is a string to identify the names or resources on the internet.
- URI is a combination of URL and URN.
- URL= uniform resource locator
- URN= uniform resource name

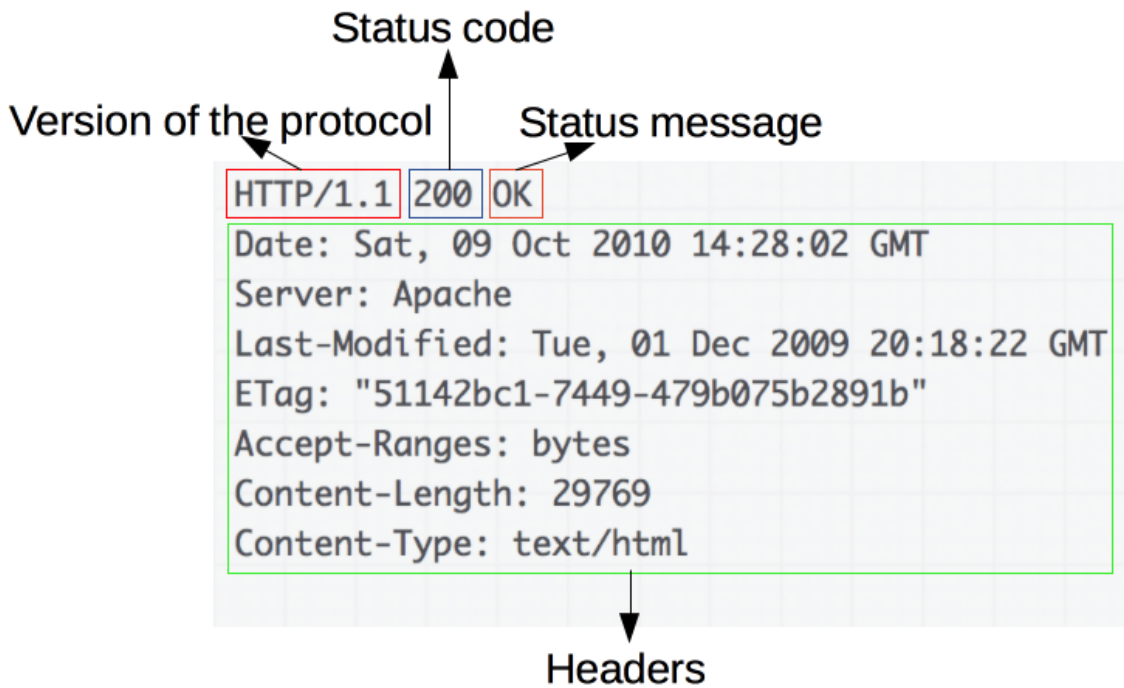


3) HTTP Version:

- The first version of HTTP was HTTP/0.9 but the official version of HTTP was HTTP/1.1

Response Message Structure

- <status line>
- <header field>
- <blank line>
- <message body>



HTTP/1.1 200 OK	Status Line	HTTP Response
Date: Thu, 20 May 2004 21:12:58 GMT	General Headers	
Connection: close		
Server: Apache/1.3.27	Response Headers	
Accept-Ranges: bytes		
Content-Type: text/html	Entity Headers	
Content-Length: 170		
Last-Modified: Tue, 18 May 2004 10:14:49 GMT		
<html> <head> <title>Welcome to the Amazing Site!</title> </head> <body> <p>This site is under construction. Please come back later. Sorry!</p> </body> </html>	Message Body	

Status Line

- Status line consists of three fields.
 - ❑ HTTP Version

- ❑ Status code
- ❑ Reason phrase

HTTP /1.1 200 OK

Status Code:

1xx: Informational - It means the request has been received and the process is continuing.

2xx: Success - It means the action was successfully received, understood, and accepted.

3xx: Redirection- It means further action must be taken in order to complete the request.

4xx: Client Error-It means the request contains incorrect syntax or cannot be fulfilled.

5xx: Server Error -It means the server failed to fulfill an apparently valid request.

Message	Description
100 Continue	Only a part of the request has been received by the server, but as long as it has not been rejected, the client should continue with the request.
101 Switching Protocols	The server switches protocol.
200 OK	The request is OK.
201 Created	The request is complete, and a new resource is created .
202 Accepted	The request is accepted for processing, but the processing is not complete.
300 Multiple Choices	A link list. The user can select a link and go to that location. Maximum five addresses .

301 Moved Permanently	The requested page has moved to a new url .
400 Bad Request	The server did not understand the request.
401 Unauthorized	The requested page needs a username and a password.
404 Not Found	The server can not find the requested page.
500 Internal Server Error	The request was not completed. The server met an unexpected condition.
505 HTTP Version Not Supported	The server does not support the "http protocol" version.

Features of HTTP Protocol

- **HTTP is connectionless:** The HTTP client, i.e., a browser initiates an HTTP request and after a request is made, the client disconnects from the server and waits for a response. The server processes the request and re-establishes the connection with the client to send a response back.
- **HTTP is media independent:** It means, any type of data can be sent by HTTP.
- **HTTP is stateless:** that means HTTP protocol can not remember the previous information of the user nor it remembers the number of times the user has visited a particular website.

Limitation of HTTP Protocol

- HTTP is not secured one.
- It is stateless protocol and therefore can not remember the previous communication.
- HTTP is a text based protocol. Hence the communication is readable to anybody

HTTP Example:

Ex-1 HTTP request & response using GET Method to fetch hello.htm page

//Request

GET /hello.htm HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.google.com

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

//Response

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

ETag: "34aa387-d-1568eb00"

Vary: Authorization,Accept

Accept-Ranges: bytes

Content-Length: 88

Content-Type: text/html

Connection: Closed

<html>

<body>

<h1>Hello, World!</h1>

</body>

</html>

Ex-2 HTTP request & response using Head Method to fetch hello.htm page

//Request

HEAD /hello.htm HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.google.com

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

//Response

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT


```
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Vary: Authorization,Accept
Accept-Ranges: bytes
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

Ex-3 HTTP request & response using POST Method.

```
//Request
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.Google.com
Content-Type: text/xml; charset=utf-8
Content-Length: 88
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://clearforest.com/">string</string>
```

```
//Response
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Vary: Authorization,Accept
Accept-Ranges: bytes
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

```
<html>
<body>
<h1>Request Processed Successfully</h1>
</body>
</html>
```

Web Page

a web page is a document available on the world wide web. Web Pages are stored on a web server and can be viewed using a web browser.

A web page can contain huge information including text, graphics, audio, video and hyperlinks. These hyperlinks are the link to other web pages.

Collection of linked web pages on a web server is known as a website. A unique Uniform Resource Locator (URL) is associated with each web page.

Static Web page

Static web pages are also known as flat or stationary web pages. They are loaded on the client's browser as exactly they are stored on the web server. Such web pages contain only static information. Users can only read the information but can't do any modification or interact with the information.

Static web pages are created using only HTML. Static web pages are only used when the information is no more required to be modified.



Dynamic Web page

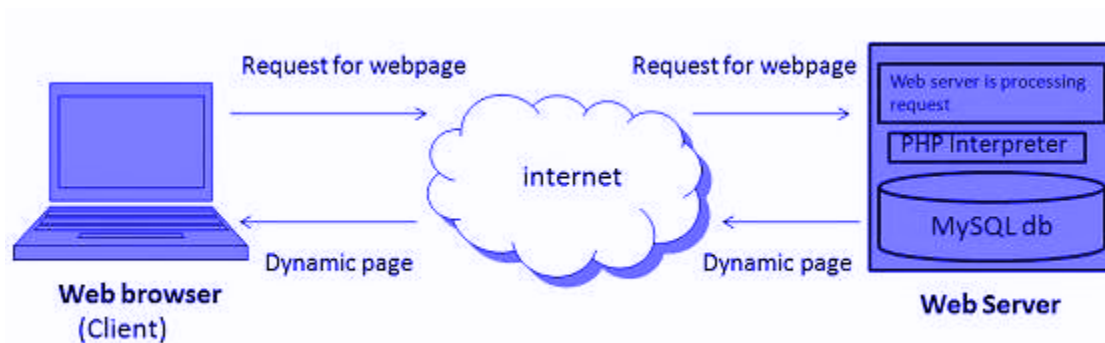
Dynamic web page shows different information at different points of time. It is possible to change a portion of a web page without loading the entire web page. It has been made possible using Ajax technology.

Server-side dynamic web page

It is created by using server-side scripting. There are server-side scripting parameters that determine how to assemble a new web page which also include setting up of more client-side processing.

Client-side dynamic web page

It is processed using client side scripting such as JavaScript. And then passed in to Document Object Model (DOM).



Scripting Languages

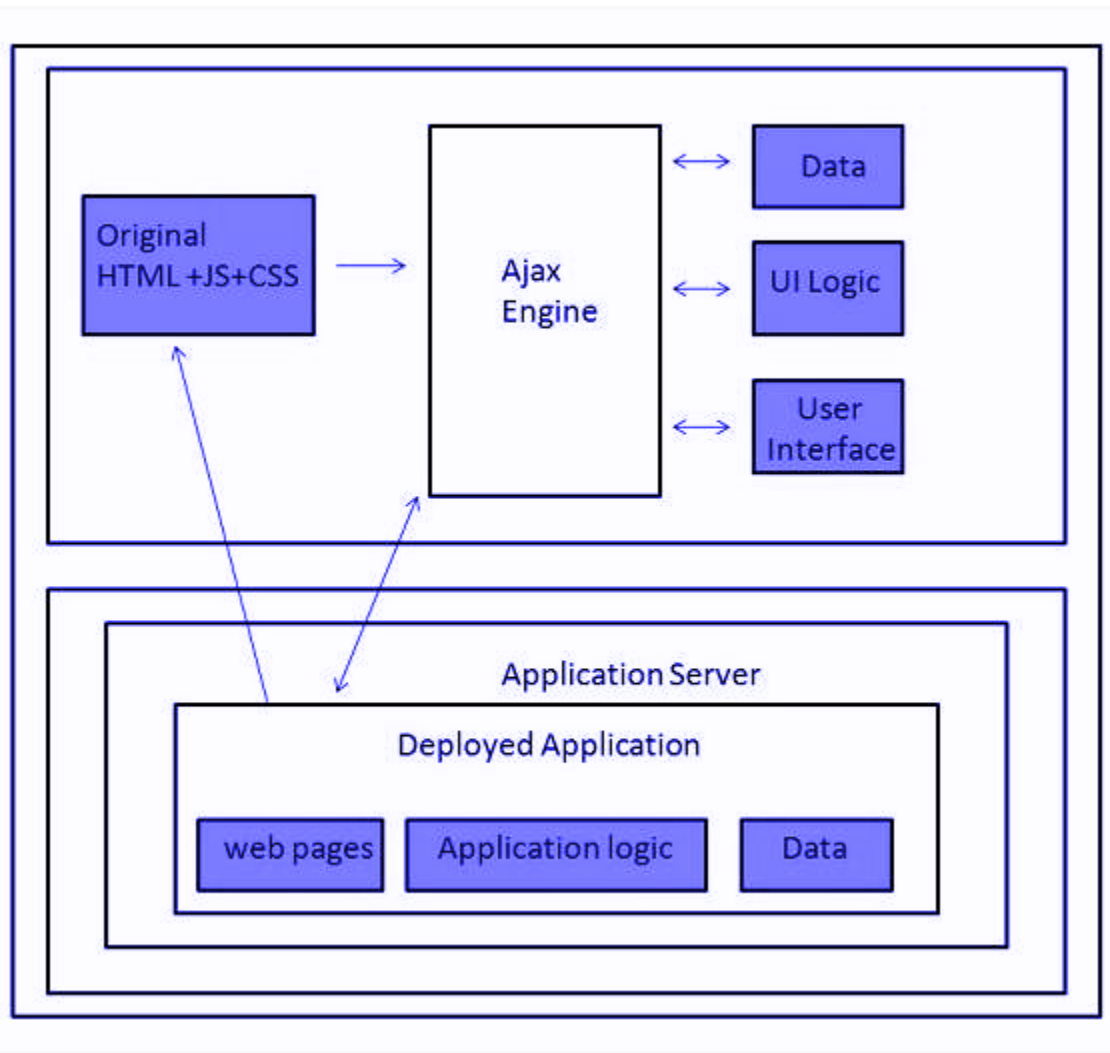
Scripting languages are like programming languages that allow us to write programs in the form of script. These scripts are interpreted not compiled and executed line by line.

Scripting language is used to create dynamic web pages.

Client-side Scripting

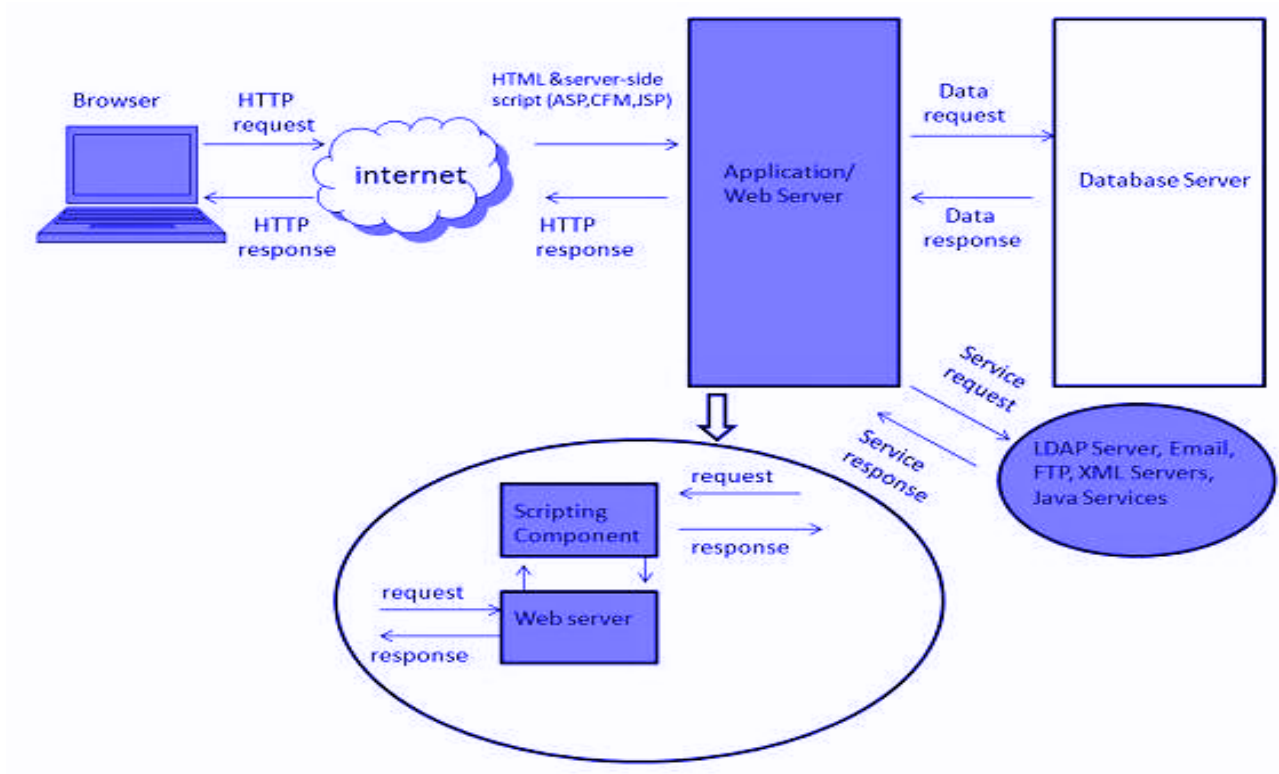
Client-side scripting refers to the programs that are executed on client-side. Client-side scripts contain the instruction for the browser to be executed in response to certain user's action.

Client-side scripting programs can be embedded into HTML files or also can be kept as separate files.



Server-side Scripting

Server-side scripting acts as an interface for the client and also limits the user's access to the resources on the web server. It can also collect the user's characteristics in order to customize response.



2.0 Concepts of Effective Web Design

2.1 Web Design Issues

Browser & Operating Systems

- Web pages are written using different HTML tags and viewed in the browser window.
- The different browsers and their versions greatly affect the way a page is rendered, as different browsers sometimes interpret the same HTML tag in a different way.
- Different versions of HTML also support different sets of tags.
- The support for different tags also varies across the different browsers and their versions.
- Same browser may work slightly differently on different operating systems and hardware platforms.
- To make a web page portable, test it on different browsers on different operating systems.

Bandwidth and Cache

Users have different connection speed, i.e. bandwidth, to access the Web sites.

Connection speed plays an important role in designing web pages, if user has low bandwidth connection and a web page contains too many images, it takes more time to download.

Generally, users have no patience to wait for longer than 10-15 seconds and move to another site without looking at the contents of your web page.

Browsers provide temporary memory called cache to store the graphics.

When a user gives the URL of the web page for the first time, the HTML file together with all the graphics files referred to in a page is downloaded and displayed.

Display Resolution

- Display resolution is another important factor affecting the Web page design, as we do not have any control on display resolution of the monitors on which the user views our pages.
- Display or screen resolution is measured in terms of pixels and common resolutions are 800 X 600 and 1024 X 786.
- We have three choices for Web page design.
- Design a web page with fixed resolution.
- Make a flexible design using an HTML table to fit into different resolutions.
- If the page is displayed on a monitor with a higher resolution, the page is displayed on the left-hand side and some part on the right-hand side remains blank. We can use centered design to display the page properly.
- (Not For Exam) Ideally we should use some frameworks for designing like
- Bootstrap/Material design.

Look & Feel

Look and feel of the website decides the overall appearance of the website.

It includes all the design aspects such as

- Web site theme
- Web typography
- Graphics
- Visual structure
- Navigation etc...

2.2 Page Layout and Linking

Website contains individual web pages that are linked together using various navigational links.

Page layout defines the visual structure of the page and divides the page area into different parts to present the information of varying importance.

Page layout allows the designer to distribute the contents on a page such that visitors can view it easily and find necessary details.

Locating Information

A Web Page is viewed on a computer screen and the screen can be divided into five major areas such as center, top, right, bottom and left in this particular order.

The first major area of importance in terms of users viewing pattern is the center, then top, right, bottom and left in this particular order.

Making Design user-Centric

It is very difficult for any Web designer to predict the exact behavior of the Web site users.

However, the idea of general behavior of the common user helps in making design of the Web site user-centric.

Users either scan the information on the web page to find the section of their interest or read the information to get details.

Sitemap

Many times Web sites are too complex as there are a large number of sections and each section contains many pages.

It becomes difficult for visitors to quickly move from one part to another.

Once the user selects a particular section and pages in that section, the user gets confused about where he/she is and where to go from there.

To make it simple, keep your hierarchy of information to a few levels or provide the navigation bar on each page to jump directly to a particular section.

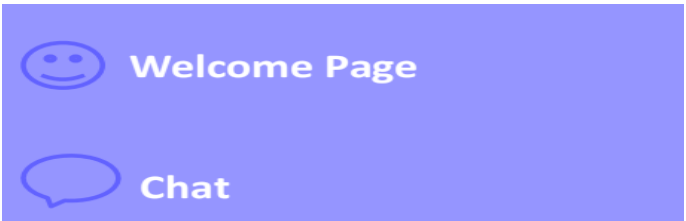
Representation of Site Map:

- Textual Site Map
- Graphical Site Map
- Textual Site Map
- This is a simplest form of site map represents all links as basic HTML text with varying size, colors, that show the important of pages within a map.



Graphical Site Map

- Graphical site maps make use of graphical icons.
- They are used to provide more obvious clue for the depth and links to the web pages



2.3 Designing effective navigation

Tips for Effective Navigation.

- Navigation links are either text based, i.e. a word or a phrase is used as a link, or graphical, i.e. a
- image, i.e. an icon or a logo is used as a link.
- Navigation links should be clear and meaningful.
- It should be consistent.
- Link should be understandable.
- Organize the links such that contents are grouped logically.
- Provide a search link, if necessary, usually on top of the page. Use common links such as 'about us' or 'Contact us'.
- Provide the way to return to the first page.
- Provide the user with information regarding location
- Horizontal navigation bar can be provided on each page to directly jump to any section

Navigation System

A complex web site often includes several types of navigation systems. To design a successful site, it is essential to understand the types of systems and how they work together to provide flexibility and context.

Types of Navigation System:

- 1) Hierarchical Navigation Systems
- 2) Global navigation
- 3) Local navigation
- 4) Contextual navigation or Ad Hoc Navigation

1) Hierarchical Navigation Systems

From the main page to the destination pages that house the actual content, the main options on each page are taken directly from the hierarchy

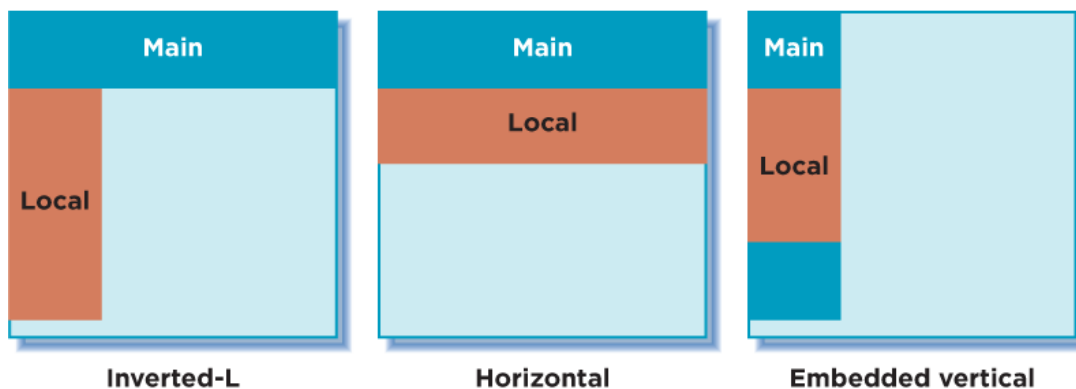
2) Global Navigation Systems:

Global navigation system is presented on every page throughout an interface, it is what we commonly called the 'site-wide navigation bar'. It typically sits at the top of the page, consisting a set of links that allow users to access the top-level pages of a site, and a link to the homepage (presented as the logo).

3) Local Navigation System:

Local navigation systems help users to explore what is nearby. It is also known as sub-navigation or page navigation. We can view it as 'an extension' of the global navigation.

There are 3 common ways to arrange the global and the local navigation — inverted-L, horizontal and embedded vertical.



a) Inverted-L:

Local navigation is presented as a vertical link list and is placed below the global navigation, aligned along the left, forming an inverted-L shape.



What are you looking for?

APPEALS

Home > What We Do > High Court (Family Division) > Appeals

High Court (Family Division) ^

▸ Appeals

Family Courts ▾

Youth Courts ▾

Mediation & Counselling ▾

Others ▾

Overview

The Family Division of the High Court exercises original jurisdiction and hears appeals against the decision of the Family Courts and the Youth Courts in family proceedings.

This section provides information on the appeals process to the High Court (Family Division). Subjected to certain exceptions, appeals to the High Court (Family Division) can be filed against:

- a. a decision of a Judge (District Judge / Magistrate) in the Family Courts;
- b. a decision of a Judge (District Judge / Magistrate) in the Youth Courts; or
- c. a decision of the Assistant Registrar / Deputy Registrar in the High Court (Family Division).

The information provided below is general in nature and is not intended as legal advice. The Family Justice Courts (FJC) cannot provide legal advice or assist with drafting the contents of any document.

b) Horizontal:

Local navigation is placed directly below the horizontal global navigation and is presented as a row, providing more options.



c) Embedded vertical

When the main navigation is presented in a vertical menu, it's common to see the local navigation being embedded between the main navigation options.

The screenshot shows the Philips website interface. At the top left is the Philips logo. To its right is a language dropdown menu set to 'Netherlands / Dutch', a search bar with the text 'Zoeken', and a button for 'Contact en ondersteuning'. Below this is a horizontal navigation bar with tabs for 'Over Philips', 'Consumentenproducten', 'Licht', and 'Medische systemen'. A banner image of a baby's face is visible on the left side of the page. The main content area features a breadcrumb trail: 'U bevindt zich hier: Home > Over Philips'. A red-bordered navigation menu is open on the left, listing: 'Bedrijfsprofiel', 'Ondernemingen', 'Ons merk', 'Design', 'Research', 'Duurzaamheid', 'Nieuwscentrum', 'Financiële informatie', and 'Carrière'. The main content area is titled 'Over Philips' and includes a large image of a modern building. Below the image is a 'Bedrijfsprofiel' section with text: 'Philips heeft wereldwijd circa 125.500 werknemers in dienst, verspreid over meer dan 60 landen, en heeft zijn hoofdkantoor in Amsterdam. Met een omzet van EUR 30,4 miljard in 2005 is de onderneming marktleider op het gebied...'. To the right of the image are two sidebars: 'Duurzaamheid' with text 'Bij Philips geloven we dat duurzame ontwikkeling niet alleen noodzakelijk, maar ook wenselijk is.' and '+ Meer', and 'Nieuwscentrum' with text 'Ontvang het laatste nieuws en de meest recente toespraken van Philips, download foto's, video's en logo's en abonneer u'.

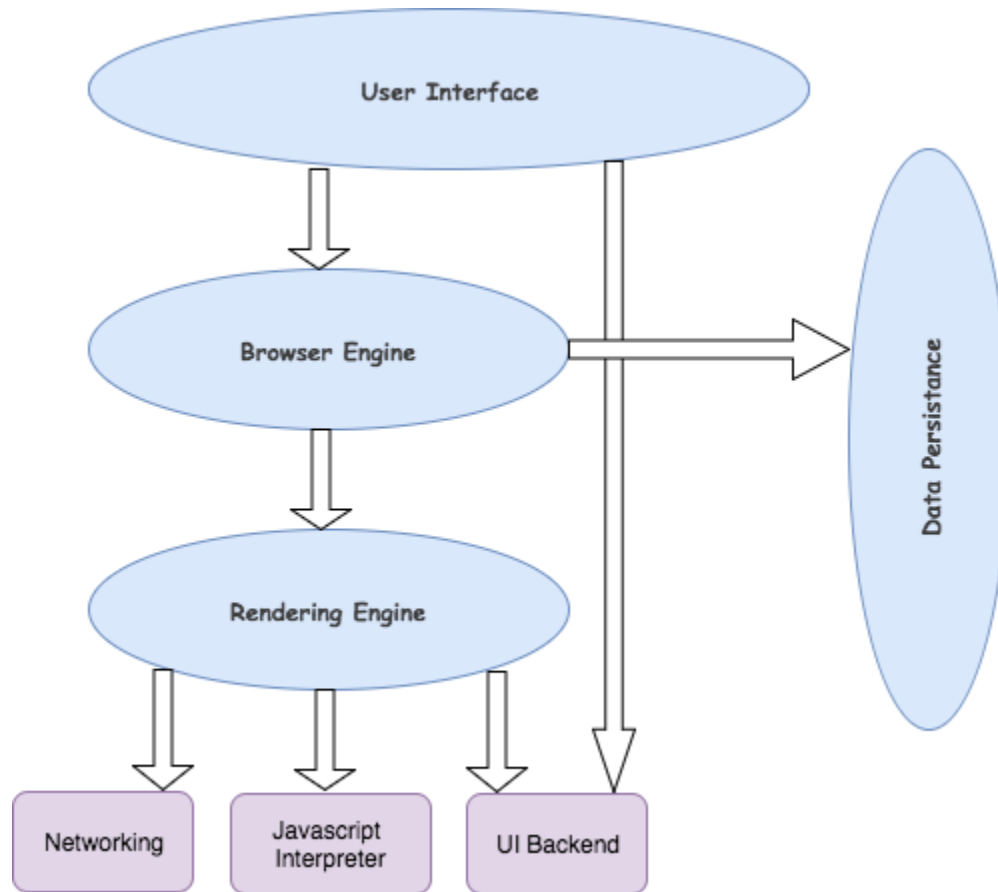
4) Contextual navigation or Ad Hoc Navigation

When the information can't be fitted neatly into the structure of global and local navigation, we can make use of contextual navigation to point users to related pages, supporting associative learning. On an e-commerce website, this could be "related products" or "you may also like" links, typically use for cross-selling. Typically an editor or content specialist will determine appropriate places for these types of links once the content has been placed into the architectural framework of the web site.

2.4 Web browser Architecture

A browser is a software application used to locate, retrieve and display content on the World Wide Web, including Web pages, images, video and other files. As a client/server model, the browser is the client run on a computer that contacts the Web server and requests information. The Web server sends the information back to the Web browser which displays the results on the computer or other Internet-enabled device that supports a browser.

Today's browsers are fully-functional software suites that can interpret and display HTML Web pages, applications, JavaScript, AJAX and other content hosted on Web servers. Many browsers offer plug-ins which extend the capabilities of the software so it can display multimedia information (including sound and video), or the browser can be used to perform tasks such as videoconferencing, to design web pages or add anti-phishing filters and other security features to the browser.



1. **The User Interface:** The user interface is the space where User interacts with the browser. It includes the address bar, back and next buttons, home button, refresh and stop, bookmark option, etc. Every other part, except the window where requested web page is displayed, comes under it.

2. **The Browser Engine:** The browser engine works as a bridge between the User interface and the rendering engine. According to the inputs from various user interfaces, it queries and manipulates the rendering engine.
3. **The Rendering Engine:** The rendering engine, as the name suggests is responsible for rendering the requested web page on the browser screen. The rendering engine interprets the HTML, XML documents and images that are formatted using CSS and generates the layout that is displayed in the User Interface. However, using plugins or extensions, it can display other types data also. Different browsers user different rendering engines:
 - Internet Explorer: Trident
 - Firefox & other Mozilla browsers: Gecko
 - Chrome & Opera 15+: Blink
 - Chrome (iPhone) & Safari: Webkit
4. **Networking:** Component of the browser which retrieves the URLs using the common internet protocols of HTTP or FTP. The networking component handles all aspects of Internet communication and security. The network component may implement a cache of retrieved documents in order to reduce network traffic.
5. **JavaScript Interpreter:** It is the component of the browser which interprets and executes the javascript code embedded in a website. The interpreted results are sent to the rendering engine for display. If the script is external then first the resource is fetched from the network. Parser keeps on hold until the script is executed.
6. **UI Backend:** UI backend is used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform specific. It underneath uses operating system user interface methods.
7. **Data Persistence/Storage:** This is a persistence layer. Browsers support storage mechanisms such as localStorage, IndexedDB, WebSQL and FileSystem. It is a small

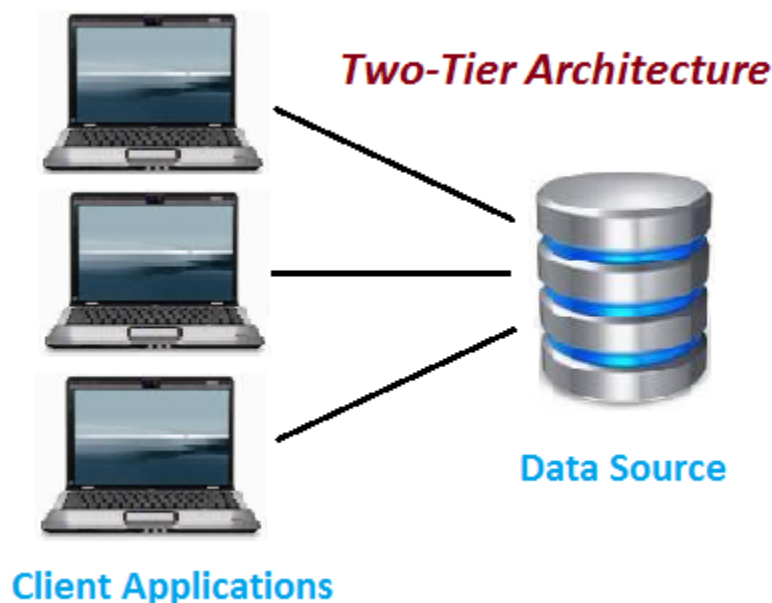
database created on the local drive of the computer where the browser is installed. It manages user data such as cache, cookies, bookmarks and preferences.

2.5 N-Tier Architecture

There are different types of N-Tier Architectures, like 3-tier Architecture, 2-Tier Architecture and 1- Tier Architecture. Generally all projects are broadly divided into two types of applications 2 tier and 3 tier architecture. Basically high level we can say that *2-tier architecture* is Client server application and *3-tier architecture* is Web based application. Below I am concentrating on the difference between Two-Tier and Three-Tier Architecture, what all advantages, disadvantages and practical examples.

Two-Tier Architecture:

The two-tier is based on Client Server architecture. The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server. Because of tight coupling a 2 tiered application will run faster.



The above figure shows the architecture of two-tier. Here the direct communication between client and server, there is no intermediate between client and server.

The Two-tier architecture is divided into two parts:

1) Client Application (Client Tier)

2) Database (Data Tier)

On client application side the code is written for saving the data in the SQL server database. Client sends the request to server and it process the request & send back with data. The main problem of two tier architecture is the server cannot respond multiple request same time, as a result it cause a data integrity issue.

Advantages:

1. Easy to maintain and modification is bit easy
2. Communication is faster

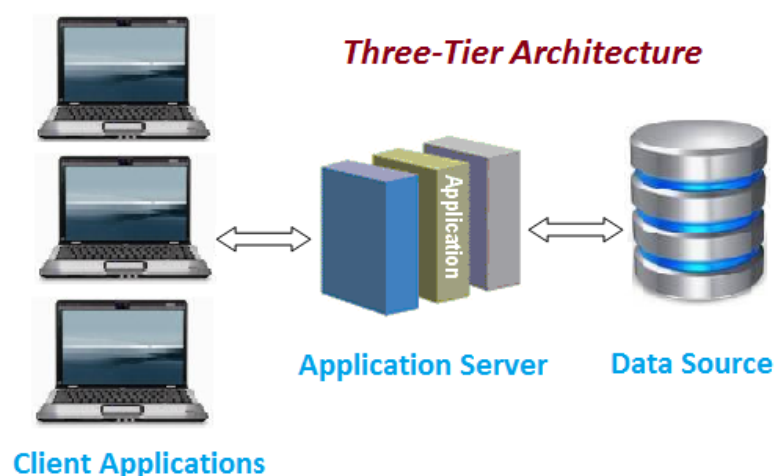
Disadvantages:

1. In two tier architecture application performance will be degrade upon increasing the users.
2. Cost-ineffective

Three-Tier Architecture:

Three-tier architecture typically comprise a presentation tier, a business or data access tier, and a data tier. Three layers in the three tier architecture are as follows:

- 1) Client layer
- 2) Business layer
- 3) Data layer



1) Client layer:

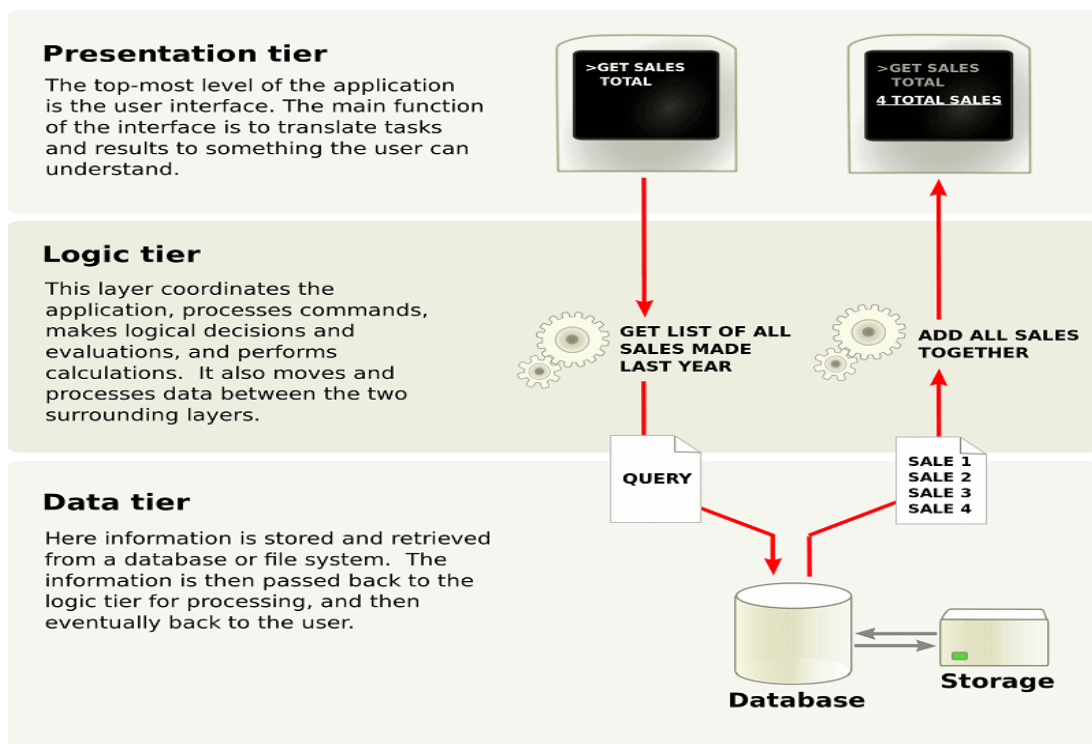
It is also called as *Presentation layer* which contains UI part of our application. This layer is used for the design purpose where data is presented to the user or input is taken from the user. For example designing registration form which contains text box, label, button etc.

2) Business layer:

In this layer all business logic written like validation of data, calculations, data insertion etc. This acts as a interface between Client layer and Data Access Layer. This layer is also called the intermediary layer helps to make communication faster between client and data layer.

3) Data layer:

In this layer actual database is comes in the picture. Data Access Layer contains methods to connect with database and to perform insert, update, delete, get data from database based on our input data.



Advantages

1. High performance, lightweight persistent objects
2. Scalability – Each tier can scale horizontally
3. Performance – Because the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers.
4. High degree of flexibility in deployment platform and configuration
5. Better Re-use
6. Improve Data Integrity
7. Improved Security – Client is not direct access to database.
8. Easy to maintain and modification is bit easy, won't affect other modules
9. In three tier architecture application performance is good.

Disadvantages

1. Increase Complexity/Effort

Single Tier or 1-Tier Architecture:

It is the simplest one as it is equivalent to running the application on the personal computer. All of the required components for an application to run are on a single application or server. Presentation layer, Business logic layer, and data layer are all located on a single machine.

3.0 Basics of HTML

HTML

Introduction

HTML stands for Hyper Text Markup Language. It is a formatting language used to define the appearance and contents of a web page. It allows us to organize text, graphics, audio, and video on a web page.

Key Points:

- The word Hypertext refers to the text which acts as a link.
- The word markup refers to the symbols that are used to define the structure of the text. The markup symbols tell the browser how to display the text and are often called tags.
- The word Language refers to the syntax that is similar to any other language.

HTML was created by Tim Berners-Lee at CERN.

HTML Versions

The following table shows the various versions of HTML:

Version	Year
HTML 1.0	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.0	1999
XHTML	2000

HTML5	2012
-------	------

HTML Tags

Tag is a command that tells the web browser how to display the text, audio, graphics or video on a web page.

Key Points:

- Tags are indicated with a pair of angle brackets.
- They start with a less than (<) character and end with a greater than (>) character.
- The tag name is specified between the angle brackets.
- Most of the tags usually occur in pairs: the start tag and the closing tag.
- The start tag is simply the tag name enclosed in an angle bracket whereas the closing tag is specified including a forward slash (/).
- Some tags are the empty i.e. they don't have the closing tag.
- Tags are not case sensitive.
- The starting and closing tag name must be the same. For example hello </i> is invalid as both are different.
- If you don't specify the angle brackets (<>) for a tag, the browser will treat the tag name as a simple text.
- The tag can also have attributes to provide additional information about the tag to the browser.

Basic tags

The following table shows the Basic HTML tags that define the basic web page:

Tag	Description
<html> </html>	Specifies the document as a web page.
<head> </head>	Specifies the descriptive information about the web documents.
<title> </title>	Specifies the title of the web page.
<body> </body>	Specifies the body of a web document.

The following code shows how to use basic tags.

```
<html>
  <head> Heading goes here...</head>
  <title> Title goes here...</title>
  <body> Body goes here...</body>
</html>
```

Formatting Tags

The following table shows the HTML tags used for formatting the text:

Tag	Description
<code> </code>	Specifies the text as bold. Eg. this is bold text
<code> </code>	It is a phrase text. It specifies the emphasized text. Eg. <i>Emphasized text</i>
<code> </code>	It is a phrase tag. It specifies an important text. Eg. this is strong text
<code><i> </i></code>	The content of the italic tag is displayed in italic. Eg. <i>Italic text</i>
<code><sub> </sub></code>	Specifies the subscripted text. Eg. X ₁
<code><sup> </sup></code>	Defines the superscripted text. Eg. X ²
<code><ins> </ins></code>	Specifies the inserted text. Eg. The price of the pen is now 2015.
<code> </code>	Specifies the deleted text. Eg. The price of the pen is now 2015.
<code><mark> </mark></code>	Specifies the marked text. Eg. It is raining

Table Tags

Following table describe the commonly used table tags:

Tag	Description
-----	-------------

<table> </table>	Specifies a table.
<tr> </tr>	Specifies a row in the table.
<th> </th>	Specifies header cell in the table.
<td> </td>	Specifies the data in a cell of the table.
<caption> </caption>	Specifies the table caption.
<colgroup> </colgroup>	Specifies a group of columns in a table for formatting.

List tags

Following table describe the commonly used list tags:

Tag	Description
 	Specifies an unordered list.
 	Specifies an ordered list.
 	Specifies a list item.
<dl> </dl>	Specifies a description list.
<dt> </dt>	Specifies the term in a description list.
<dd> </dd>	Specifies description of term in a description list.

HTML Anchor

The **HTML anchor tag** defines a hyperlink that links one page to another page. It can create hyperlinks to other web pages as well as files, location, or any URL. The "href" attribute is the most important attribute of the HTML a tag. and which links to the destination page or URL.href attribute of HTML anchor tag

The href attribute is used to define the address of the file to be linked. In other words, it points out the destination page.

The syntax of HTML anchor tag is given below.

```
<a href = "....."> Link Text </a>
```

Let's see an example of an HTML anchor tag.

1. `Click for Second Page`

Specify a location for Link using target attribute

If we want to open that link to another page then we can use the target attribute of <a> tag. With the help of this link will be open on the next page.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
</head>
<body>
<p>Click on <a href="`https://www.google.com/" target="`_blank'">
this-link </a> to go on the homepage of Google.</p>
</body>
</html>
```

Output:

Click on [this-link](https://www.google.com/) to go on the homepage of Google.

Note:

- The **target** attribute can only be used with the href attribute in the anchor tag.
- If we will not use the target attribute then the link will open on the same page.

Appearance of HTML anchor tag

An **unvisited link** is displayed underlined and blue.

A **visited link** displayed underlined and purple.

An **active link** is underlined and red.

HTML Image

HTML img tag is used to display images on the web page. HTML img tag is an empty tag that contains attributes only, closing tags are not used in an HTML image element.

Let's see an example of an HTML image.

```
<h2>HTML Image Example</h2>
```

```

```

Output:



Attributes of HTML img tag

The src and alt are important attributes of HTML img tag. All attributes of HTML image tag are given below.

1) src

It is a necessary attribute that describes the source or path of the image. It instructs the browser where to look for the image on the server.

The location of the image may be on the same directory or another server.

2) alt

The alt attribute defines an alternate text for the image, if it can't be displayed. The value of the alt attribute describes the image in words. The alt attribute is considered good for SEO prospective.

3) width

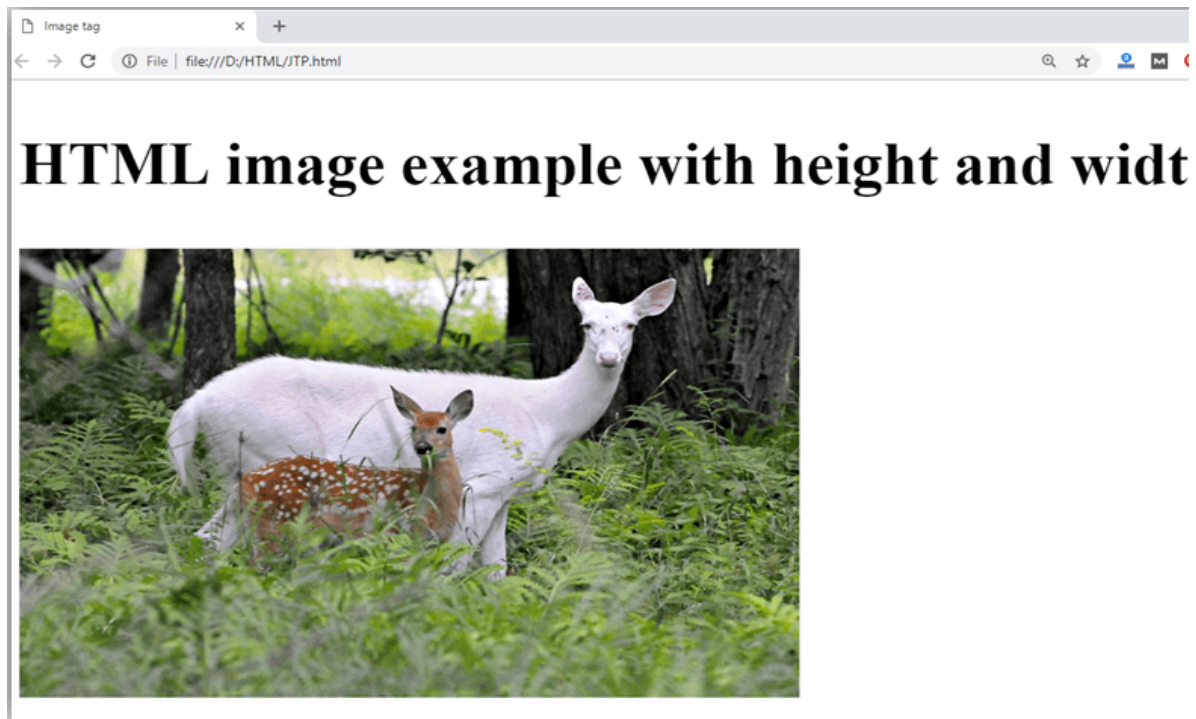
It is an optional attribute which is used to specify the width to display the image. It is not recommended now. You should apply CSS in place of width attribute.

4) height

It h3 the height of the image. The HTML height attribute also supports iframe, image and object elements. It is not recommended now. You should apply CSS in place of height attribute.

Example:

1. ``



Use of alt attribute

We can use alt attributes with `img` tags. It will display an alternative text in case the image cannot be displayed on the browser. Following is the example for alt attribute:

1. ``



HTML Table

HTML table tag is used to display data in tabular form (row * column). There can be many columns in a row.

We can create a table to display data in tabular form, using the <table> element, with the help of <tr> , <td>, and <th> elements.

In Each table, table row is defined by <tr> tag, table header is defined by <th>, and table data is defined by <td> tags.

HTML tables are used to manage the layout of the page e.g. header section, navigation bar, body content, footer section etc. But it is recommended to use a div tag over table to manage the layout of the page .

HTML Table Tags

Tag	Description
<table>	It defines a table.
<tr>	It defines a row in a table.
<th>	It defines a header cell in a table.
<td>	It defines a cell in a table.
<caption>	It defines the table caption.
<colgroup>	It specifies a group of one or more columns in a table for formatting.

<code><col></code>	It is used with the <code><colgroup></code> element to specify column properties for each column.
<code><tbody></code>	It is used to group the body content in a table.
<code><thead></code>	It is used to group the header content in a table.
<code><tfooter></code>	It is used to group the footer content in a table.

HTML Table Example

Let's see the example of the HTML table tag. Its output is shown below.

```
<table>
<tr><th>First_Name</th><th>Last_Name</th><th>Marks</th></tr>
<tr><td>Sonoo</td><td>Jaiswal</td><td>60</td></tr>
<tr><td>James</td><td>William</td><td>80</td></tr>
<tr><td>Swati</td><td>Sironi</td><td>82</td></tr>
<tr><td>Chetna</td><td>Singh</td><td>72</td></tr>
</table>
```

Output:

First_Name	Last_Name	Marks
Sonoo	Jaiswal	60
James	William	80
Swati	Sironi	82
Chetna	Singh	72

In the above html table, there are 5 rows and 3 columns = 5 * 3 = 15 values.

HTML Table with Border

There are two ways to specify borders for HTML tables.

1. By border attribute of table in HTML
2. By border property in CSS

1) HTML Border attribute

You can use the border attribute of table tag in HTML to specify border. But it is not recommended now.

```
<table border="1">
<tr><th>First_Name</th><th>Last_Name</th><th>Marks</th></tr>
<tr><td>Sonoo</td><td>Jaiswal</td><td>60</td></tr>
<tr><td>James</td><td>William</td><td>80</td></tr>
<tr><td>Swati</td><td>Sironi</td><td>82</td></tr>
<tr><td>Chetna</td><td>Singh</td><td>72</td></tr>
</table>
```

Output:

First_Name	Last_Name	Marks
Sonoo	Jaiswal	60
James	William	80
Swati	Sironi	82
Chetna	Singh	72

2) CSS Border property

It is now recommended to use the border property of CSS to specify the border in the table.

```
<style>
table, th, td {
  border: 1px solid black;
}
</style>
```

You can collapse all the borders in one border by border-collapse property. It will collapse the border into one.

```
<style>
table, th, td {
  border: 2px solid black;
  border-collapse: collapse;
}
</style>
```

Output:

Name	Last Name	Marks
Sonoo	Jaiswal	60
James	William	80
Swati	Sironi	82
Chetna	Singh	72

HTML Table with cell padding

You can specify padding for table header and table data by two ways:

1. By cellpadding attribute of table in HTML

2. By padding property in CSS

The cellpadding attribute of the HTML table tag is obsolete now. It is recommended to use CSS. So let's see the code of CSS.

```
<style>
table, th, td {
  border: 1px solid pink;
  border-collapse: collapse;
}
th, td {
  padding: 10px;
}
</style>
```

HTML Table width:

We can specify the HTML table width using the CSS **width** property. It can be specified in pixels or percentage.

We can adjust our table width as per our requirement. Following is the example to display the table with width.

```
table{
    width: 100%    }
```

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>table</title>
  <style>
    table{
      border-collapse: collapse;
      width: 100%;
    }
    th,td{
      border: 2px solid green;
      padding: 15px;
    }
  </style>
</head>
<body>
```

```

<table>
  <tr>
    <th>1 header</th>
    <th>1 header</th>
    <th>1 header</th>
  </tr>
  <tr>
    <td>1data</td>
    <td>1data</td>
    <td>1data</td>
  </tr>
  <tr>
    <td>2 data</td>
    <td>2 data</td>
    <td>2 data</td>
  </tr>
  <tr>
    <td>3 data</td>
    <td>3 data</td>
    <td>3 data</td>
  </tr>
</table>
</body>
</html>

```

Output:

1 header	1 header	1 header
1data	1data	1data
2 data	2 data	2 data
3 data	3 data	3 data

HTML Table with colspan

If you want to make a cell span more than one column, you can use the colspan attribute.

It will divide one cell/row into multiple columns, and the number of columns depends on the value of the colspan attribute.

Let's see the example that spans two columns.

CSS code:

```
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
th, td {
  padding: 5px;
}
</style>
```

HTML code:

```
<table style="width:100%">
  <tr>
    <th>Name</th>
    <th colspan="2">Mobile No.</th>
  </tr>
  <tr>
    <td>Heena Panjwani</td>
    <td>9125678593</td>
    <td>7995825454</td>
  </tr>
</table>
```

Output:

Name	Mobile No.	
Heena Panjwani	9125678593	7995825454

HTML Table with rowspan

If you want to make a cell span more than one row, you can use the rowspan attribute.

It will divide a cell into multiple rows. The number of divided rows will depend on rowspan values.

Let's see the example that spans two rows.

CSS code:

```
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
th, td {
  padding: 10px;
}
</style>
```

HTML code:

```
<table>
<tr><th>Name</th><td>Heena Panjwani</td></tr>
<tr><th rowspan="2">Mobile No.</th><td>9125678593</td></tr>
<tr><td>7995825454</td></tr>
</table>
```

Output:

Name	Heena Panjwani
Mobile No.	9125678593
	7995825454

HTML table with caption

HTML caption is displayed above the table. It must be used after the table tag only.

```
<table>
<caption>Student Records</caption>
<tr><th>First_Name</th><th>Last_Name</th><th>Marks</th></tr>
<tr><td>Vimal</td><td>Jaiswal</td><td>70</td></tr>
<tr><td>Mike</td><td>Warn</td><td>60</td></tr>
<tr><td>Shane</td><td>Warn</td><td>42</td></tr>
<tr><td>Jai</td><td>Malhotra</td><td>62</td></tr>
</table>
```

Frames

Frames help us to divide the browser's window into multiple rectangular regions. Each region contains a separate html web page and each of them work independently.

A set of frames in the entire browser is known as frameset. It tells the browser how to divide the browser window into frames and the web pages that each has to load.

The following table describes the various tags used for creating frames:

Tag	Description
<code><frameset></code> <code></frameset></code>	It is a replacement of the <code><body></code> tag. It doesn't contain the tags that are normally used in <code><body></code> element; instead it contains the <code><frame></code> element used to add each frame.
<code><frame></code> <code></frame></code>	Specifies the content of different frames in a web page.
<code><base></code> <code></base></code>	It is used to set the default target frame in any page that contains links whose contents are displayed in another frame.

HTML `<meta>` tag

HTML `<meta>` tag is used to represent the metadata about the HTML document. It specifies page description, keywords, copyright, language, author of the documents, etc.

The metadata does not display on the webpage, but it is used by search engines, browsers and other web services which scan the site or webpage to know about the webpage.

With the help of meta tag, you can experiment and preview how your webpage will render on the browser.

The `<meta>` tag is placed within the `<head>` tag, and it can be used more than one time in a document.

Syntax:

1. **<meta charset="utf-8">**

Following are some specifications about the HTML **<meta>** tag

Display	None
Start tag/End tag	Empty Tag(Only Start tag)
Usage	Document Structural

Following are some specific syntaxes of meta tag which shows the different uses of meta Tag.

1. **<meta charset="utf-8">**

It defines the character encoding. The value of charset is "utf-8" which means it will support displaying any language.

2. **<meta name="keywords" content="HTML, CSS, JavaScript, Tutorials">**

It specifies the list of keywords which is used by search engines.

3. **<meta name="description" content="Free Online tutorials">**

It defines the website description which is useful to provide relevant search performed by search engines.

4. **<meta name="author" content="this author">**

It specifies the author of the page. It is useful to extract author information by Content management system automatically.

5. **<meta name="refresh" content="50">**

It specifies to provide instruction to the browser to automatically refresh the content after every 50sec (or any given time).

6. **<meta http-equiv="refresh" content="5; url=https://www.google.com/html-tags-list">**

In the above example we have set a URL with content so it will automatically redirect to the given page after the provided time.

7. **<meta name="viewport" content="width=device-width, initial-scale=1.0">**

It specifies the viewport to control the page dimension and scaling so that our website looks good on all devices. If this tag is present, it indicates that this page is mobile device supported.

Example

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="keywords" content="HTML, CSS, JavaScript,
Tutorials">
  <meta name="description" content="Free Online tutorials">
  <meta name="author" content="thisauthor">
  <meta http-equiv="refresh" content="5;
url=https://www.google.com/html-tags-list">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
</head>
<body>
<h2>Example of Meta tag</h2>
<p>This example shows the use of meta tag within an HTML
document</p>
</body>
```

`</html>`

HTML iframes

HTML Iframe is used to display a nested webpage (a webpage within a webpage). The HTML `<iframe>` tag defines an inline frame, hence it is also called an Inline frame.

An HTML iframe embeds another document within the current HTML document in the rectangular region.

The webpage content and iframe contents can interact with each other using JavaScript.

Iframe Syntax

An HTML iframe is defined with the `<iframe>` tag:

1. `<iframe src="URL"></iframe>`

Here, "src" attribute specifies the web address (URL) of the inline frame page.

Set Width and Height of iframe

You can set the width and height of an iframe by using "width" and "height" attributes. By default, the attributes values are specified in pixels but you can also set them in percent. i.e. 50%, 60% etc.

Example: (Pixels)

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Iframes example</h2>
<p>Use the height and width attributes to specify the size of the
iframe:</p>
```



```
<iframe src="https://www.google.com/" height="300"
width="400"></iframe>
</body>
</html>
```

Iframe Target for a link

You can set a target frame for a link by using iframe. Your specified target attribute of the link must refer to the name attribute of the iframe.

Example:

```
<!DOCTYPE html>
<html>
<body>

<h2>Iframe - Target for a Link</h2>
<iframe height="300px" width="100%" src="new.html"
name="iframe_a"></iframe>
<p><a href="https://www.google.com"
target="iframe_a">google.com</a></p>
<p>The name of the iframe and link target must have the same value
else the link will not open as a frame. </p>
</body>
</html>
```

Forms

Forms are used to input the values. These values are sent to the server for processing. Forms use input elements such as text fields, check boxes, radio buttons, lists, submit buttons etc. to enter the data into it.

The following table describes the commonly used tags while creating a form:

Tag	Description
<form> </form>	It is used to create HTML forms.

<code><input> </input></code>	Specifies the input field.
<code><textarea> </textarea></code>	Specifies a text area control that allows users to enter multi-line text.
<code><label> </label></code>	Specifies the label for an input element.

What is an HTML Form? Discuss different form attributes.

Form is a data collection mechanism within HTML that allows the design of various styles of input to suit most types of information.

An input element can vary in many ways, depending on the type attribute. An input element can

be of type textfield, checkbox, password, radiobutton, submit button, and more.

Following are attributes of `<form>`.

Name:

The name attribute specifies the name of a form which is used to reference elements in JavaScript.

`<form action="URL">` Value : URL

Description : Where to send the form data.

2. Action:

The required action attribute specifies where to send the form-data when a form is submitted.

`<form action="URL">` Value : URL

Description : Where to send the form data.

3. Method :

The method attribute specifies how to send form-data (the form-data is sent to the page specified in the action attribute).

```
<form method="get|post">
```

Value : get

Description : Default. Appends the form-data to the URL in name/value pairs:

```
URL?name=value name=value
```

Value : post

Description : Sends the form-data as an HTTP post transaction.

4. Target

The target attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

```
<form target="_blank|_self|_parent|_top|framename">
```

HTML Audio Tag

HTML audio tag is used to define sounds such as music and other audio clips. Currently there are three supported file formats for HTML 5 audio tag.

1. mp3
2. wav
3. ogg

HTML5 supports <video> and <audio> controls. The Flash, Silverlight and similar technologies are used to play the multimedia items.

This table defines that which web browser supports which audio file format.

HTML Audio Tag Example

Let's see the code to play an mp3 file using the HTML audio tag.

```
<audio controls>
```

```
  <source src="koyal.mp3" type="audio/mpeg">
```

Your browser does not support the html audio tag.

```
</audio>
```

Attributes of HTML Audio Tag

There is a list of HTML audio tags.

Attribute	Description
controls	It defines the audio controls which are displayed with play/pause buttons.
autoplay	It specifies that the audio will start playing as soon as it is ready.
loop	It specifies that the audio file will start over again, every time when it is completed.
muted	It is used to mute the audio output.
preload	It specifies the author view to upload an audio file when the page loads.
src	It specifies the source URL of the audio file.

HTML Audio Tag Attribute Example

Here we are going to use controls, autoplay, loop and src attributes of HTML audio tag.

1. `<audio controls autoplay loop>`
2. `<source src="koyal.mp3" type="audio/mpeg"></audio>`

MIME Types for HTML Audio format

The available MIME type HTML audio tag is given below.

Audio Format	MIME Type
mp3	audio/mpeg
ogg	audio/ogg

wav	audio/wav
-----	-----------

HTML Video Tag

HTML 5 supports <video> tag also. The HTML video tag is used for streaming video files such as a movie clip, song clip on the web page.

Currently, there are three video formats supported for HTML video tag:

1. mp4
2. webM
3. ogg

Let's see the table that defines which web browser supports video file format.

HTML Video Tag Example

Let's see the code to play mp4 files using the HTML video tag.

```
<video controls>
  <source src="movie.mp4" type="video/mp4">
  Your browser does not support the html video tag.
</video>
```

Attributes of HTML Video Tag

Let's see the list of HTML 5 video tag attributes.

Attribute	Description
controls	It defines the video controls which are displayed with play/pause buttons.
height	It is used to set the height of the video player.
width	It is used to set the width of the video player.
poster	It specifies the image which is displayed on the screen when the video is not played.

autoplay	It specifies that the video will start playing as soon as it is ready.
loop	It specifies that the video file will start over again, every time when it is completed.
muted	It is used to mute the video output.
preload	It specifies the author view to upload a video file when the page loads.
src	It specifies the source URL of the video file.

HTML Video Tag Attribute Example

Let's see the example of a video tag in HTML where we are using height, width, autoplay, controls and loop attributes.

```
<video width="320" height="240" controls autoplay loop>
  <source src="movie.mp4" type="video/mp4">
  Your browser does not support the html video tag.
</video>
```

MIME Types for HTML Video format

The available MIME type HTML video tag is given below.

Video Format	MIME Type
mp4	video/mp4
ogg	video/ogg
webM	video/webM

HTML Drag and Drop

HTML Drag and Drop (DnD) is a feature of HTML5. It is a powerful user interface concept which is used to copy, reorder and delete items with the help of mouse. You can hold the mouse button down over an element and drag it to another location. If you want to drop the element there, just release the mouse button.

If you want to achieve the Drag and Drop functionality in traditional HTML4, you must either have to use complex JavaScript programming or other JavaScript frameworks like jQuery etc.

Events for Drag and Drop feature

Event	Description
Drag	It fires every time when the mouse is moved while the object is being dragged.
Dragstart	It is a very initial stage. It fires when the user starts dragging the object.
Dragenter	It fires when the user moves his/her mouse cursor over the target element.
Dragover	This event is fired when the mouse moves over an element.
Dragleave	This event is fired when the mouse leaves an element.
Drop	Drop It fires at the end of the drag operation.
Dragend	It fires when the user releases the mouse button to complete the drag operation.

Stages during Drag and Drop operations

1) Make an element draggable

If you want to make an element draggable, set the draggable attribute to "true" on the element. For example:

1. ``

2) What to drag:

Use `ondragstart` and `setData ()` methods.

Specify what should happen when the element is dragged.

3) Where to Drop:

Use an `ondragover` event.

4) Do the Drop:

Use `ondrop` events.

HTML5 Geolocation

The Geolocation is one of the best HTML5 API which is used to identify the user's geographic location for the web application.

This new feature of HTML5 allows you to navigate the latitude and longitude coordinates of the current website's visitor. These coordinates can be captured by JavaScript and send to the server which can show your current location on the website

Most of the geolocation services use Network routing addresses such as IP addresses, RFID, WIFI and MAC addresses or internal GPS devices to identify the user's location.

User privacy:

The user's location is a privacy concern, so the geolocation API protects the user's privacy by taking the user's permission before getting the location. Geolocation API sends a notification prompt box which the user can allow or deny, and if the user allows then only his location will be identified.

Geolocation object

The Geolocation API works with the navigation.geolocation object. Its read-only property returns a Geolocation object which identifies the location of the user and can generate a customized result based on user location.

Syntax:

1. geo=navigator. geolocation;

If this object is present, then you can get the geolocation services.

Geolocation Methods

The Geolocation API uses three methods of Geolocation interface which are given following:

Methods	Description
getCurrentPosition()	It identifies the device or the user's current location and returns a position object with data.
watchPosition()	Return a value whenever the device location changes.
clearWatch()	It cancels the previous watchPosition() call

Checking for browser support:

The geolocation property of navigator.geolocation object helps to determine the browser support for the Geolocation API.

```
<!DOCTYPE html>
<html>
<head>
  <title>Geolocation API</title>
</head>
<body>
  <h1>Find your Current location</h1>
  <button onclick="getlocation()">Click me</button>
  <div id="location"></div>
```

```

<script>
    var x= document.getElementById("location");
    function getlocation() {
        if(navigator.geolocation){
            alert("your browser is supporting Geolocation API")
        }
        else
        {
            alert("Sorry! your browser is not supporting")
        }
    }
</script>
</body>
</html>

```

Getting the User's current position:

To get the user's current location, `getCurrentPosition()` method of the `navigator.geolocation` object is used. This method accepts three parameters:

- **success:** A success callback function to get the location of the user
- **error:** An error callback function which takes "Position Error" object as input.
- **options:** It defines various options for getting the location.

The below example will return the longitude and latitude of the visitor's current location.

Example

```

<!DOCTYPE html>
<html>
<head>
<title>Geolocation API</title>
</head>
<body>
    <h1>Find your Current location</h1>
    <button onclick="getlocation()">Click me</button>
    <div id="location"></div>
<script>
    var x= document.getElementById("location");
    function getlocation() {

```

```

        if(navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(showPosition)
        }
        else
        {
            alert("Sorry! your browser is not supporting")
        } }
function showPosition(position) {
    var x = "Your current location is (" + "Latitude: " +
position.coords.latitude + ", " + "Longitude: " +
position.coords.longitude + ")";
        document.getElementById("location").innerHTML = x;
    }
</script>
</body>
</html>

```

HTML Web Storage

Web Storage is one of the great features of HTML5. With the Web Storage feature, web applications can locally store data within the browser on the client side. It stores data in the form of a key/value pair on the browser. Web Storage sometimes also known as DOM storage.

Storing data with the help of web storage is similar to cookies, but it is better and faster than cookie storage.

In compared to cookies Web Storage has Following Advantages:

- Web Storage can use storage space upto 5MB per domain. (The browser software may prompt the user if the space limit is reached).
- It will not send data to the server side, hence it is faster than cookie storage.
- The data stored by local Storage never expires, but cookie data expires after some time or session.
- Web Storage is more secure than cookies.

Types of Web Storage

There are two types of web storage with different scope and lifetime.

- **Local Storage:** Local Storages uses `Windows.localStorage` object which stores data and is available for every page. But data persist even if the browser is closed and reopened (Stores data with no Expiration).
- **Session Storage:** Session Storage uses `Windows.sessionStorage` object which stores data for one session and data will be lost if the window or browser tab will be closed.

```
<!DOCTYPE html>
<html>
<body>
  <div id="result"></div>
  <script>
    if(typeof(Storage)!=="undefined") {
      document.getElementById("result").innerHTML = "Hey, Your browser
supports the Web Storage.";
    }
    else{
document.getElementById("result").innerHTML = "Sorry, your browser
does not support Web Storage";
    }
  </script>
</body>
</html>
```

HTML5

HTML5 provides details of all 40+ HTML tags including audio, video, header, footer, data, datalist, article etc. This HTML tutorial is designed for beginners and professionals.

HTML5 is the next version of HTML. Here, you will get some brand new features which will make HTML much easier. These new introducing features make your website layout

clearer to both website designers and users. There are some elements like <header>, <footer>, <nav> and <article> that define the layout of a website.

Why use HTML5

It is enriched with advanced features which makes it easy and interactive for designer/developer and users.

- It allows you to play a video and audio file.
- It allows you to draw on a canvas.
- It facilitates you to design better forms and build web applications that work offline.
- It provides you advanced features for that you would normally have to write JavaScript to do.
- The most important reason to use HTML 5 is, we believe it is not going anywhere. It will be here to serve for a long time according to W3C recommendation.

New Features

HTML5 introduces a number of new elements and attributes that can help you in building modern websites. Here is a set of some of the most prominent features introduced in HTML5.

- New Semantic Elements – These are like <header>, <footer>, and <section>.
- Forms 2.0 – Improvements to HTML web forms where new attributes have been introduced for <input> tag.
- Persistent Local Storage – To achieve without resorting to third-party plugins.

- WebSocket – A next-generation bidirectional communication technology for web applications.
- Server-Sent Events – HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).
- Canvas – This supports a two-dimensional drawing surface that you can program with JavaScript.
- Audio & Video – You can embed audio or video on your webpages without resorting to third-party plugins.
- Geolocation – Now visitors can choose to share their physical location with your web application.
- Microdata – This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
- Drag and drop – Drag and drop the items from one location to another location on the same webpage.

Difference between HTML and HTML5

HTML	HTML5
It didn't support audio and video without the use of flash player support.	It supports audio and video controls with the use of <audio> and <video> tags.
It uses cookies to store temporary data.	It uses SQL databases and application cache to store offline data.

Does not allow JavaScript to run in the browser.	Allows JavaScript to run in the background. This is possible due to JS Web worker API in HTML5.
Vector graphics is possible in HTML with the help of various technologies such as VML, Silver-light, Flash, etc.	Vector graphics is additionally an integral part of HTML5 like SVG and canvas.
It does not allow drag and drop effects.	It allows drag and drop effects.
Not possible to draw shapes like circle, rectangle, triangle etc.	HTML5 allows you to draw shapes like circle, rectangle, triangle etc.
It works with all old browsers.	It is supported by all new browsers like Firefox, Mozilla, Chrome, Safari, etc.
Older versions of HTML are less mobile-friendly.	HTML5 language is more mobile-friendly.
Doctype declaration is too long and complicated.	Doctype declaration is quite simple and easy.
Elements like nav, header were not present.	New element for web structure like nav, header, footer etc.
Character encoding is long and complicated.	Character encoding is simple and easy.
It is almost impossible to get the true GeoLocation of a user with the help of a browser.	One can track the GeoLocation of a user easily by using JS GeoLocation API.

It can not handle inaccurate syntax.	It is capable of handling inaccurate syntax.
Attributes like charset, async and ping are absent in HTML.	Attributes of charset, async and ping are a part of HTML 5.

HTML 5 Example

```
<!DOCTYPE>
<html>
<head>
<title>Web page title</title>
</head>
<body>
<h1>Write Your First Heading</h1>
<p>Write Your First Paragraph.</p>
</body>
</html>
```

Write Your First Heading

Write Your First Paragraph.

HTML 5 Tags

There is a list of newly included tags in HTML 5. These HTML 5 tags (elements) provide a better document structure. This list shows all HTML 5 tags in alphabetical order with description.

List of HTML 5 Tags

Tag	Description
<article>	This element is used to define an independent piece of content in a document, that may be a blog, a magazine or a newspaper article.
<aside>	It specifies that the article is slightly related to the rest of the whole page.

<audio>	It is used to play audio files in HTML.
<bdi>	The bdi stands for bi-directional isolation. It isolates a part of text that is formatted in another direction from the outside text document.
<canvas>	It is used to draw canvas.
<data>	It provides a machine readable version of its data.
<datalist>	It provides an auto complete feature for textfield.
<details>	It specifies the additional information or controls required by the user.
<dialog>	It defines a window or a dialog box.
<figcaption>	It is used to define a caption for a <figure> element.
<figure>	It defines a self-contained content like photos, diagrams etc.
<footer>	It defines a footer for a section.
<header>	It defines a header for a section.
<main>	It defines the main content of a document.
<mark>	It specifies the marked or highlighted content.
<menuitem>	It defines a command that the user can invoke from a popup menu.
<meter>	It is used to measure the scalar value within a given range.
<nav>	It is used to define the navigation link in the document.
<progress>	It specifies the progress of the task.
<rp>	It defines what to show in browsers that don't support ruby annotation.
<rt>	It defines an explanation/pronunciation of characters.

<ruby>	It defines ruby annotation along with <rp> and <rt>.
<section>	It defines a section in the document.
<summary>	It specifies a visible heading for a <detailed> element.
<svg>	It is used to display shapes.
<time>	It is used to define a date/time.
<video>	It is used to play video files in HTML.
<wbr>	It defines a possible line break.

HTML 5 | <header> Tag

The <header> tag in HTML is used to define the header for a document or a section.

- The header tag contains information related to the title and heading of the related content.
- The <header> element is intended to usually contain the section's heading (an h1-h6 element or an <hgroup> element), but this is not required.
- The <header> element can also be used to wrap a section's table of contents, a search form, or any relevant logos.
- The <header> tag is a new tag in HTML5 and it requires a starting tag as well as an end tag.
- There can be several <header> elements in one document.
- A <header> tag cannot be placed within a <footer>, <address> or another <header> element.

Syntax:

<header> ...</header>

Below examples illustrate the <header> element in HTML:

Example 1:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Header Tag</title>
  </head>
  <body>
    <article>
      <header>
        <h1>This is the heading.</h1>
        <h4>This is the
sub-heading.</h4>
        <p>This is the metadata.</p>
      </header>
    </article>
  </body>
</html>
```

Output:

This is the heading.

This is the sub-heading.

This is the metadata.

HTML5 | <footer> Tag

The <footer> tag in HTML is used to define a footer of HTML documents. This section contains the footer information (author information, copyright information, carriers, etc). The footer tag is used within the body tag. The <footer> tag is new in the HTML5. The footer elements require a start tag as well as an end tag.

Syntax :

```
<footer> ... </footer>
```

A footer element typically contains authorship information, copyright information, contact information, sitemap, back to top links, related documents, etc.

Below examples illustrate the <footer> Tag in HTML elements:

```
<!DOCTYPE html>

<html>
  <head>
    <title>footer tag</title>
    <style>
      .column {
        float: left;
        width: 27%;
        height: 300px;
      }
      p {
        font-size:20px;
        font-weight:bold;
      }
    </style>
  </head>
  <body>
    <footer>
      <div class="column">
        <p>Company</p>
        <ul style="list-style-type:disc">
          <li>About Us</li>
          <li>Careers</li>
          <li>Privacy Policy</li>
          <li>Contact Us</li>
        </ul>
      </div>
    <div class="column">
      <p>Learn</p>
      <ul>
        <li>Algorithms</li>
        <li>Data Structures</li>
        <li>Languages</li>
        <li>CS Subjects</li>
        <li>Video Tutorials</li>
      </ul>
    </div>
  </body>
</html>
```

```
        </div>
<div class="column">
    <p>Practice</p>
    <ul>
```

HTML5 <article> Tag

The <article> tag is one of the new sectioning elements in HTML5. The HTML <article> tag is used to represent an article. More specifically, the content within the <article> tag is independent of the other content of the site (even though it can be related).

In other words, The article element represents a component of a page that consists of self-contained composition in a document, page, or site. For Ex. in syndication.

A potential source for Article Element are:

- A magazine/newspaper article
- A blog entry
- A forum post
- A user-submitted a comment

This tag is most often used in two contexts:

- On a page with a single piece of content, a single <article> element can be used to contain the main content and set it off from the rest of the page.
- On a page with multiple pieces of content (a blog index page, a search results page, a category page, news feed), multiple <article> elements can be used to contain each individual piece of content.

Either way, it is similar to the <div> element and displays the stylish work the same. However, using the <article> element instead of <div> provides more semantic information

to screen readers, search engines, and third-party applications.

Note: This tag does not render as anything special in a browser, you have to use CSS for that.

Default CSS setting: Most browsers will display the Article element with the following values.

HTML

```
<article> {  
    display:block;  
}
```

HTML5 – Section Tag Example

The HTML5 <section> tag defines sections in a document, such as chapters, headers, footers, or any other sections of the document. Previously (before HTML5), only a way to provide such separation was possible through only <div> tags.

section element is a semantic element. This means that it provides meaning to both user agents and humans about what the enclosed content is—specifically a section of the document. It is a generic semantic element so you should use it when none of the other semantic container elements (article, aside and nav) are appropriate.

Section Tag Syntax

When creating a <section> in HTML5, as when you used the <div> tag in HTML, you can use either the id or class attributes. Each id must be unique, as in HTML, and class can be used multiple times when necessary.

You should always have a header element (H1 through H6) as a part of the section. If you can't come up with a title for the section, then again the <div> element is probably more appropriate. And never ever use section tags for putting styles only.

Let's say you were creating a document about data processing. The following represents a typical use for the <section> elements.

```

<section id="preparation" class="imaginaryClass">
  <h2>Prepare Data</h2>
  <p>Random text Random text Random text...</p>
</section>
<section id="processing">
  <h2>Process Prepared Data</h2>
  <p>Some More Random text Some More Random text Some More
Random text ...</p>
</section>
<section id="display">
  <h2>Processed Data</h2>
  <p>Some More Random text Some More Random text Some More
Random text ...</p>
</section>

```

HTML5 | figure Tag

The <figure> tag in HTML is used to add self-contained content like illustrations, diagrams, photos or codes listing in a document. It is related to main flow but it can be used in any position of a document and the figure goes with the flow of the document and if you remove it then it should not affect the flow of the document. This tag is new in HTML5.

Syntax:

```
<figure> Image content... </figure>
```

Attributes: It contains mostly two tags which are listed below:

- **img src:** This tag is used to add image source in the document.
- **figcaption:** This tag is used to set the caption to the image.

HTML5 | figcaption Tag

he <figcaption> tag in HTML is used to set a caption to the figure element in a document. This tag is new in HTML5.

syntax:

```
<figcaption> Figure caption </figcaption>
```

4. CSS

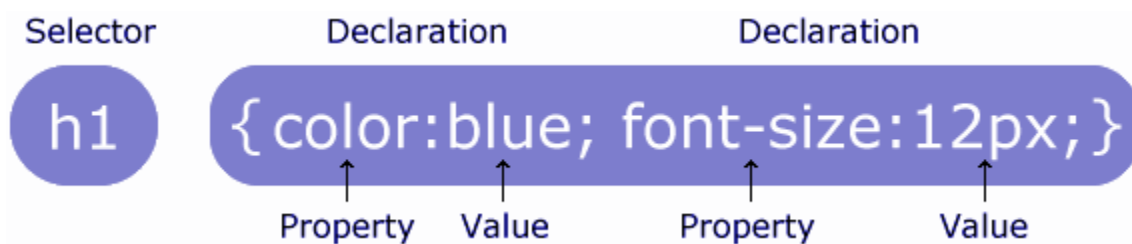
Introduction

What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red;
  text-align: center;
}
```



```
</style>
</head>
<body>
<p>Hello World!</p>
<p>These paragraphs are styled with CSS.</p>
</body>
</html>
```

Hello World!

These paragraphs are styled with CSS.

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
/* This is a single-line comment */
p {
  color: red;
}
</style>
</head>
<body>
<p>Hello World!</p>
<p>This paragraph is styled with CSS.</p>
<p>CSS comments are not shown in the output.</p>
</body>
</html>
```

Hello World!

This paragraph is styled with CSS.

CSS comments are not shown in the output.

css types:

You can use CSS in three ways in your HTML document –

1. **External Style Sheet** – Define style sheet rules in a separate .css file and then include that file in your HTML document using HTML <link> tag.
2. **Internal Style Sheet** – Define style sheet rules in header section of the HTML document using <style> tag.
3. **Inline Style Sheet** – Define style sheet rules directly along-with the HTML elements using **style** attribute.

External Style Sheet

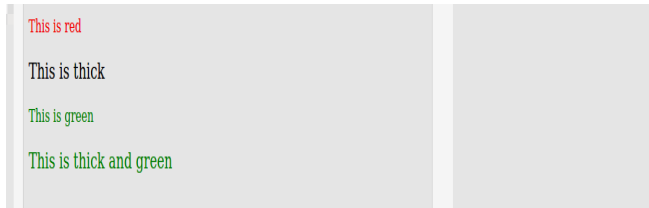
If you need to use your style sheet to various pages, then it's always recommended to define a common style sheet in a separate file. A cascading style sheet file will have extension as .css and it will be included in HTML files using <link> tag.

Example

```
.red {
    color: red;
}
.thick {
    font-size:20px;
}
.green {
    color:green;
}
<!DOCTYPE html>
<html>
    <head>
        <title>HTML External CSS</title>
        <link rel = "stylesheet" type = "text/css" href =
"/html/style.css">
    </head>
    <body>
        <p class = "red">This is red</p>
        <p class = "thick">This is thick</p>
        <p class = "green">This is green</p>
        <p class = "thick green">This is thick and green</p>
```

```
</body>  
</html>
```

This will produce the following result –



Internal Style Sheet

If you want to apply Style Sheet rules to a single document only, then you can include those rules in the header section of the HTML document using `<style>` tag.

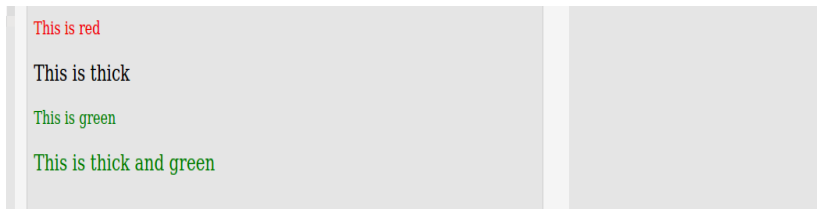
Rules defined in the internal style sheet overrides the rules defined in an external CSS file.

Example

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>HTML Internal CSS</title>  
  
    <style type = "text/css">  
      .red {  
        color: red;  
      }  
      .thick{  
        font-size:20px;  
      }  
      .green {  
        color:green;  
      }  
    </style>  
  </head>  
<body>  
  <p class = "red">This is red</p>  
  <p class = "thick">This is thick</p>  
  <p class = "green">This is green</p>
```

```
<p class = "thick green">This is thick and green</p>
</body>
</html>
```

This will produce the following result –



Inline Style Sheet

You can apply style sheet rules directly to any HTML element using the style attribute of the relevant tag. This should be done only when you are interested to make a particular change in any HTML element only.

Rules defined inline with the element overrides the rules defined in an external CSS file as well as the rules defined in <style> element.

What are the advantages and disadvantages of Inline Styles?

The advantages of Inline Styles are:

- It is especially useful for a small number of style definitions.
- It has the ability to override other style specification methods at the local level.

The disadvantages of Inline Styles are:

- It does not separate out the style information from content.
- The styles for many documents can not be controlled from one source.

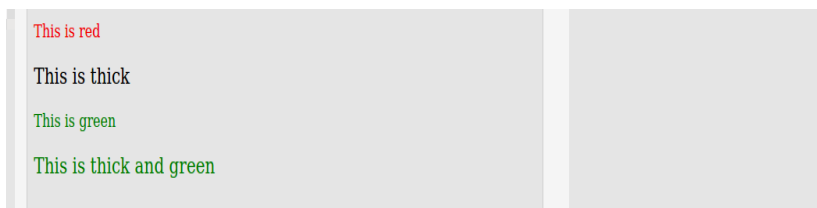
- Selector grouping methods can not be used to handle complex situations.

- Control classes can not be created to control multiple element types within the document.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Inline CSS</title>
  </head>
  <body>
    <p style = "color:red;">This is red</p>
    <p style = "font-size:20px;">This is thick</p>
    <p style = "color:green;">This is green</p>
    <p style = "color:green;font-size:20px;">This is thick and
green</p>
  </body>
</html>
```

This will produce the following result –



CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

The element selector selects HTML elements based on the element name.

-We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)

- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<p>Every paragraph will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Every paragraph will be affected by the style.

Me too!

And me!

The CSS id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
#para1 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
</body>
</html>
```

Hello World!

This paragraph is not affected by the style.

The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>
</body>
</html>
```

Red and center-aligned heading

Red and center-aligned paragraph.

The CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>
<h1>Hello world!</h1>
<p>Every element on the page will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Hello world!

Every element on the page will be affected by the style.

Me too!

And me!

The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
<!DOCTYPE html>
<html>
<head>
```



```
<style>
h1, h2, p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

Hello World!

Smaller heading!

This is a paragraph.

CSS Pseudo-classes

What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Syntax

The syntax of pseudo-classes:

```
selector:pseudo-class {
    property:value;
```

```
}
```

Anchor Pseudo-classes

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

/* mouse hover link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
</style>
</head>
<body>
<h2>CSS Links</h2>
<p><b><a href="default.asp" target="_blank">This is a
link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the
CSS definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS
definition in order to be effective.</p>
</body>
</html>
```

CSS Links

[This is a link](#)

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.

Note: a:active MUST come after a:hover in the CSS definition in order to be effective.

Hover on <div>

An example of using the :hover pseudo-class on a <div> element:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: green;
  color: white;
  padding: 25px;
  text-align: center;
}

div:hover {
  background-color: blue;
}
</style>
</head>
<body>
<p>Mouse over the div element below to change its background
color:</p>
<div>Mouseover Me</div>
</body>
</html>
```

Mouse over the div element below to change its background color:

Mouse Over Me

CSS background-image Property

```
<!DOCTYPE html>
<html>
```

```

<head>
<style>
body {
  background-image: url("paper.gif");
}
</style>
</head>
<body>
<h1>Hello World!</h1>

<p>This page has an image as the background!</p>
</body>
</html>

```



CSS color Property

```

<!DOCTYPE html>
<html>
<body>
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat
volutpat.
Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.

```

```
</p>
</body>
</html>
```

Hello World

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div.a {
    text-align: center;
}

div.b {
    text-align: left;
}

div.c {
    text-align: right;
}

div.d {
    text-align: justify;
}
</style>
</head>
<body>
<h1>The text-align Property</h1>
<div class="a">
<h2>text-align: center:</h2>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam
semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum
volutpat tellus diam, consequat gravida libero rhoncus ut.</p>
</div>
<div class="b">
<h2>text-align: left:</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam
semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum
volutpat tellus diam, consequat gravida libero rhoncus ut.</p>
</div>

<div class="c">
<h2>text-align: right:</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam
semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum
volutpat tellus diam, consequat gravida libero rhoncus ut.</p>
</div>
<div class="d">
<h2>text-align: justify:</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam
semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum
volutpat tellus diam, consequat gravida libero rhoncus ut.</p>
</div>
</body>
</html>
```

O/P:

The text-align Property

text-align: center:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align: left:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align: right:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align: justify:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

Property Values

Value	Description
left	Aligns the text to the left
right	Aligns the text to the right
center	Centers the text
justify	Stretches the lines so that each line has equal width (like in newspapers and magazines)

CSS text-align-last Property

```
div.a {  
  text-align: justify; /* For Edge */  
  text-align-last: right;  
}
```

```
div.b {  
  text-align: justify; /* For Edge */  
  text-align-last: center;  
}
```

```
div.c {
  text-align: justify; /* For Edge */
  text-align-last: justify;
}
```

O/P:

The text-align-last Property

text-align-last: right:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align-last: center:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align-last: justify:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

Definition and Usage

The text-align-last property specifies how to align the last line of a text.

Notice that the text-align-last property sets the alignment for all last lines within the selected element. So, if you have a <div> with three paragraphs in it, text-align-last will apply to the last line of EACH of the paragraphs.

Note: In Edge/Internet Explorer the text-align-last property only works on text that has "text-align: justify".

CSS text-decoration Property

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-decoration: overline;
}

h2 {
  text-decoration: line-through;
}

h3 {
  text-decoration: underline;
}

h4 {
  text-decoration: underline overline;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>

</body>
</html>
```

O/P:

This is heading 1

~~This is heading 2~~

This is heading 3

This is heading 4

Definition and Usage

The text-decoration property specifies the decoration added to text, and is a shorthand property for:

- text-decoration-line (required)
- text-decoration-color
- text-decoration-style

CSS text-decoration-color Property

```
p {  
  
    text-decoration: underline;  
  
    text-decoration-color: red;  
  
}
```

CSS Borders

CSS Border Style

The border-style property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right

border, bottom border, and the left border).

Example

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

The border-style Property

This property specifies what kind of border to display:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border.

A ridge border.

An inset border.

An outset border.

No border.

A hidden border.

A mixed border.

CSS Border Width

The border-width property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three predefined values: thin, medium, or thick.

The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

```
p.one {  
  border-style: solid;  
  border-width: 5px;  
}  
p.two {  
  border-style: solid;  
  border-width: medium;  
}
```

```
p.three {  
  border-style: solid;  
  border-width: 2px 10px 4px 20px;  
}
```

CSS Border Color

```
p.one {  
  border-style: solid;  
  border-color: red;  
}  
p.two {  
  border-style: solid;  
  border-color: green;  
}  
  
p.three {  
  border-style: solid;  
  border-color: red green blue yellow;  
}
```

CSS Margins

The CSS margin properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

Tip: Negative values are allowed.

- margin-top
- margin-right
- margin-bottom
- margin-left

All the margin properties can have the following values:

- auto - the browser calculates the margin
- length - specifies a margin in px, pt, cm, etc.
- % - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element

EXAMPLE:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
  background-color: lightblue;
}
</style>
</head>
<body>

<h2>Using individual margin properties</h2>

<div>This div element has a top margin of 100px, a right margin of
150px, a bottom margin of 100px, and a left margin of 80px.</div>

</body>
</html>
```

O/P:

Using individual margin properties

This div element has a top margin of 100px, a right margin of 150px, a bottom margin of 100px, and a left margin of 80px.

CSS Padding

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

- padding-top
- padding-right
- padding-bottom
- padding-left

All the padding properties can have the following values:

- length - specifies a padding in px, pt, cm, etc.
- % - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

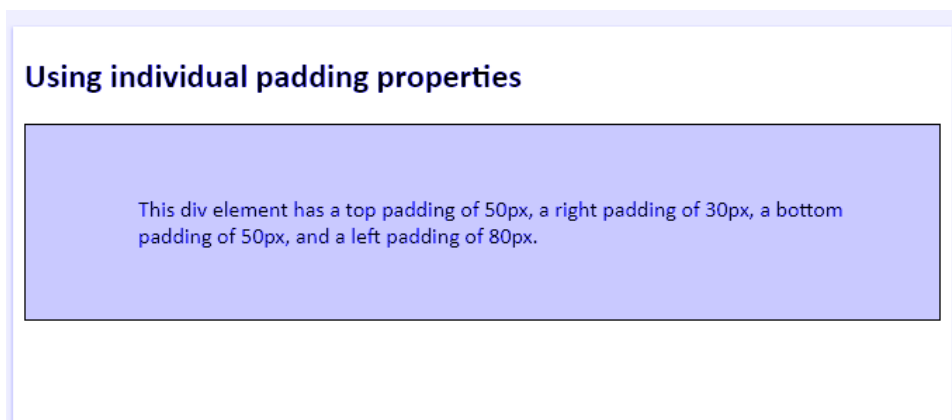
Note: Negative values are not allowed.

Example:

```
<!DOCTYPE html>
```

```
<html>
<head>
<style>
div {
  border: 1px solid black;
  background-color: lightblue;
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
</style>
</head>
<body>
<h2>Using individual padding properties</h2>
<div>This div element has a top padding of 50px, a right padding of
30px, a bottom padding of 50px, and a left padding of 80px.</div>
</body>
</html>
```

O/P:



CSS position Property

```
h2 {
  position: absolute;
  left: 100px;
  top: 150px;
```


}

The position Property

This is a heading with an absolute position

With absolute positioning, an element can be placed anywhere on a page. The heading below is placed 100px from the left of the page and 150px from the top of the page.

Property Values

Value	Description
static	Default value. Elements render in order, as they appear in the document flow
absolute	The element is positioned relative to its first positioned (not static) ancestor element
fixed	The element is positioned relative to the browser window
relative	The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position

CSS - Lists

Lists are very helpful in conveying a set of either numbered or bullet points. This chapter teaches you how to control list type, position, style, etc., using CSS.

We have the following five CSS properties, which can be used to control lists –

- The list-style-type allows you to control the shape or appearance of the marker.
- The list-style-position specifies whether a long point that wraps to a second line should align with the first line or start underneath the start of the marker.
- The list-style-image specifies an image for the marker rather than a bullet point or number.
- The list-style serves as shorthand for the preceding properties.
- The marker-offset specifies the distance between a marker and the text in the list.

Now, we will see how to use these properties with examples.

The list-style-type Property

The list-style-type property allows you to control the shape or style of bullet point (also known as a marker) in the case of unordered lists and the style of numbering characters in ordered lists

Here are the values which can be used for an unordered list –

Sr.No.	Value & Description
1	none NA
2	disc (default) A filled-in circle
3	circle An empty circle
4	square A filled-in square

Here is an example –

```

<html>
  <head>
  </head>

  <body>
    <ul style = "list-style-type:circle;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ul>

    <ul style = "list-style-type:square;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ul>

    <ol style = "list-style-type:decimal;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ol>

    <ol style = "list-style-type:lower-alpha;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ol>

    <ol style = "list-style-type:lower-roman;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ol>
  </body>
</html>

```

It will produce the following result –

- Maths
- Social Science
- Physics
- Maths
- Social Science

- **Physics**
 1. **Maths**
 2. **Social Science**
 3. **Physics**
 - a. **Maths**
 - b. **Social Science**
 - c. **Physics**
 - i. **Maths**
 - ii. **Social Science**
 - iii. **Physics**

CSS Gradients

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines two types of gradients:

- Linear Gradients (goes down/up/left/right/diagonally)
- Radial Gradients (defined by their center)

CSS Linear Gradients

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
```

Direction - Top to Bottom (this is default)

Example:

```
#grad {
```

```
background-image: linear-gradient(red, yellow);  
  
}
```

CSS Radial Gradients

A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

Syntax

```
background-image: radial-gradient(shape size at position, start-color, ..., last-color);
```

By default, shape is ellipse, size is farthest-corner, and position is center.

Radial Gradient - Evenly Spaced Color Stops (this is default)

Example:

```
#grad {  
  
background-image: radial-gradient(red, yellow, green);  
  
}
```

Radial Gradient - Differently Spaced Color Stops

CSS Animations

CSS allows animation of HTML elements without using JavaScript or Flash!

- @keyframes
- animation-name
- animation-duration
- animation-delay

- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

What are CSS Animations?

An animation lets an element gradually change from one style to another. You can change as many CSS properties you want, as many times you want. To use CSS animation, you must first specify some keyframes for the animation. Keyframes hold what styles the element will have at certain times.

The @keyframes Rule

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

```
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

Note: The animation-duration property defines how long an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
/* The animation code */
@keyframes example {
  0%    {background-color: red;}
  25%   {background-color: yellow;}
  50%   {background-color: blue;}
  100%  {background-color: green;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

The following example will change both the background-color and the position of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
/* The animation code */
@keyframes example {
  0%    {background-color:red; left:0px; top:0px;}
  25%   {background-color:yellow; left:200px; top:0px;}
  50%   {background-color:blue; left:200px; top:200px;}
  75%   {background-color:green; left:0px; top:200px;}
  100%  {background-color:red; left:0px; top:0px;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
```

```
position: relative;
background-color: red;
animation-name: example;
animation-duration: 4s;
}
```

CSS Tooltips

CSS Tooltips are a great way to display extra information about something when the user moves the mouse cursor over an element.

Basic Tooltip Example

Let's create a basic tooltip that appears when the cursor of the mouse moves over an element.

See this example:

```
<!DOCTYPE html>
<html>
<style>
.tooltip {
    position: relative;
    display: inline-block;
    border-bottom: 1px dotted black;
}
.tooltip .tooltiptext {
    visibility: hidden;
    width: 120px;
    background-color: red;
    color: #fff;
    text-align: center;
    border-radius: 6px;
    padding: 5px 0;
    position: absolute;
    z-index: 1;
}
.tooltip:hover .tooltiptext {
    visibility: visible;
}
</style>
<body style="text-align:center;">
<p>Move the mouse over the text below:</p>
<div class="tooltip">Hover over me
    <span class="tooltiptext">This is tooltip text</span>
</div>
</body>
</html>
```


By using tooltips, you can display the position of the tooltip information in four ways:

- Top of the element
- Left side of the element
- Right side of the element
- Bottom of the element

Top Tooltip

The top tooltip specifies that if you move your mouse cursor over the element, the tooltip information will be displayed on the top of the element.

See this example:

```
<!DOCTYPE html>
<html>
<style>
.tooltip {
    position: relative;
    display: inline-block;
    border-bottom: 1px dotted black;
}
.tooltip .tooltiptext {
    visibility: hidden;
    width: 120px;
    background-color: red;
    color: #fff;
    text-align: center;
    border-radius: 6px;
    padding: 5px 0;
    position: absolute;
    z-index: 1;
    bottom: 100%;
    left: 50%;
    margin-left: -60px;
}
.tooltip:hover .tooltiptext {
    visibility: visible;
}
</style>
<body style="text-align:center;">
<h2>Top Tooltip Example</h2>
<p>Move your mouse cursor over the below heading</p>
```

```

<div class="tooltip"><strong> Welcome to AIT</strong>
  <span class="tooltiptext">A solution of all technology.</span>
</div>
</body>
</html>

```

Bottom Tooltip

The bottom tooltip specifies that if you move your mouse cursor over the element, the tooltip information will be displayed on the bottom of the element.

See this example:

```

<!DOCTYPE html>
<html>
<style>
.tooltip {
  position: relative;
  display: inline-block;
  border-bottom: 1px dotted black;
}
.tooltip .tooltiptext {
  visibility: hidden;
  width: 120px;
  background-color: red;
  color: #fff;
  text-align: center;
  border-radius: 6px;
  padding: 5px 0;
  position: absolute;
  z-index: 1;
  top: 100%;
  left: 50%;
  margin-left: -60px;
}
.tooltip:hover .tooltiptext {
  visibility: visible;
}
</style>
<body style="text-align:center;">
<h2>Bottom Tooltip Example</h2>
<p>Move your mouse cursor over the below heading</p>
<div class="tooltip"><strong> Welcome to AIT</strong>
<span class="tooltiptext">A solution of all technology.</span>
</div>

```

```
</body>
</html>
```

Left Tooltip

The left tooltip specifies that if you move your mouse cursor over the element, the tooltip information will be displayed on the left side of the element.

See this example:

```
<!DOCTYPE html>
<html>
<style>
.tooltip {
    position: relative;
    display: inline-block;
    border-bottom: 1px dotted black;
}
.tooltip .tooltiptext {
    visibility: hidden;
    width: 120px;
    background-color: red;
    color: #fff;
    text-align: center;
    border-radius: 6px;
    padding: 5px 0;
    position: absolute;
    z-index: 1;
    top: -5px;
    right: 105%;
}
.tooltip:hover .tooltiptext {
    visibility: visible;
}
</style>
<body style="text-align:center;">
<h2>Left Tooltip Example</h2>
<p>Move your mouse cursor over the below heading</p>
<div class="tooltip"><strong> Welcome to AIT</strong>
<span class="tooltiptext">A solution of all technology.</span>
</div>
</body>
</html>
```

Right Tooltip

The right tooltip specifies that if you move your mouse cursor over the element, the tooltip information will be displayed on the right side of the element.

See this example:

```
<!DOCTYPE html>
<html>
<style>
.tooltip {
    position: relative;
    display: inline-block;
    border-bottom: 1px dotted black;
}
.tooltip .tooltiptext {
    visibility: hidden;
    width: 120px;
    background-color: red;
    color: #fff;
    text-align: center;
    border-radius: 6px;
    padding: 5px 0;
    position: absolute;
    z-index: 1;
    top: -5px;
    left: 105%;
}
.tooltip:hover .tooltiptext {
    visibility: visible;
}
</style>
<body style="text-align:center;">
<h2>Right Tooltip Example</h2>
<p>Move your mouse cursor over the below heading</p>
<div class="tooltip"><strong> Welcome to AIT</strong>
<span class="tooltiptext">A solution of all technology.</span>
</div>
</body>
</html>
```

CSS Styling Images

Rounded Images

Use the border-radius property to create rounded images:



Example

Rounded Image:

```
img {  
  border-radius: 8px;  
}
```

Thumbnail Images

Use the border property to create thumbnail images.

Thumbnail Image:



Example:

```
img {  
  border: 1px solid #ddd;  
  border-radius: 4px;  
  padding: 5px;  
  width: 150px;  
}
```

```

```

Responsive Images

Responsive images will automatically adjust to fit the size of the screen.

Resize the browser window to see the effect:



If you want an image to scale down if it has to, but never scale up to be larger than its original size, add the following:

Example:

```
img {  
  max-width: 100%;  
  height: auto;  
}
```

Transparent Image

The opacity property can take a value from 0.0 - 1.0. The lower value, the more transparent:



opacity 1

(default)

```
img {  
  
  opacity: 0.5;  
  
}
```

CSS Variables - The var() Function

CSS Variables

The var() function is used to insert the value of a CSS variable.

CSS variables have access to the DOM, which means that you can create variables with local or global scope, change the variables with JavaScript, and change the variables based on media queries.

A good way to use CSS variables is when it comes to the colors of your design. Instead of copy and pasting the same colors over and over again, you can place them in variables.

The Traditional Way

The following example shows the traditional way of defining some colors in a style sheet (by defining the colors to use, for each specific element):

Example:

```
body { background-color: #1e90ff; }
h2 { border-bottom: 2px solid #1e90ff; }
.container {
  color: #1e90ff;
  background-color: #ffffff;
  padding: 15px;
}
button {
  background-color: #ffffff;
  color: #1e90ff;
  border: 1px solid #1e90ff;
  padding: 5px;
}
```

Syntax of the var() Function

The var() function is used to insert the value of a CSS variable.

The syntax of the var() function is as follows:

`var(name, value)`

Value	Description
name	Required. The variable name (must start with two dashes)
value	Optional. The fallback value (used if the variable is not found)

Note: The variable name must begin with two dashes (--) and it is case sensitive!

How var() Works

First of all: CSS variables can have a global or local scope.

Global variables can be accessed/used through the entire document, while local variables can be used only inside the selector where it is declared.

To create a variable with global scope, declare it inside the `:root` selector. The `:root` selector matches the document's root element.

To create a variable with local scope, declare it inside the selector that is going to use it.

The following example is equal to the example above, but here we use the `var()` function.

First, we declare two global variables (`--blue` and `--white`). Then, we use the `var()` function to insert the value of the variables later in the style sheet:

Example:

```
:root {
  --blue: #1e90ff;
  --white: #ffffff;
}
body { background-color: var(--blue); }
h2 { border-bottom: 2px solid var(--blue); }
.container {
  color: var(--blue);
  background-color: var(--white);
  padding: 15px;
}
button {
  background-color: var(--white);
  color: var(--blue);
  border: 1px solid var(--blue);
  padding: 5px;
}
```

CSS Flexbox Layout Module

Before the Flexbox Layout module, there were four layout modes:

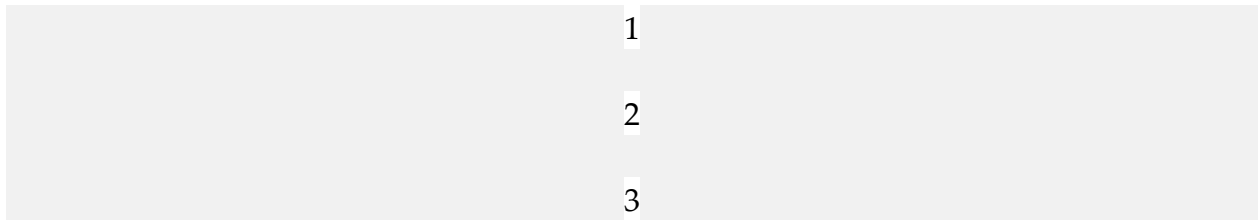
- Block, for sections in a webpage
- Inline, for text
- Table, for two-dimensional table data

- Positioned, for explicit position of an element

The Flexible Box Layout Module makes it easier to design flexible responsive layout structure without using float or positioning.

Flexbox Elements

To start using the Flexbox model, you need to first define a flex container.



The element above represents a flex container with three flex items.

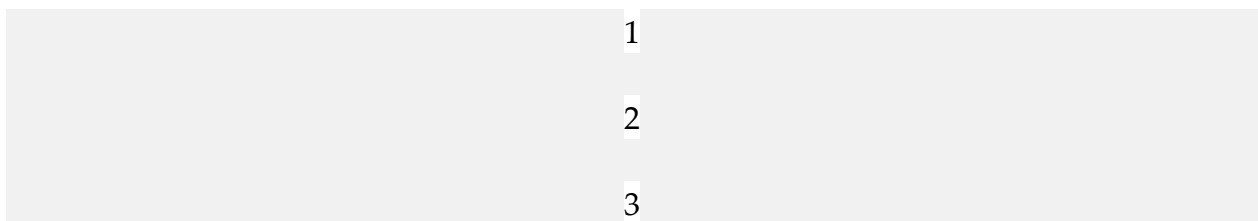
Example:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

CSS Flex Container

Parent Element (Container)

this is a flex container with three flex items:



The flex container becomes flexible by setting the display property to flex:

```
.flex-container {
```

```
display: flex;

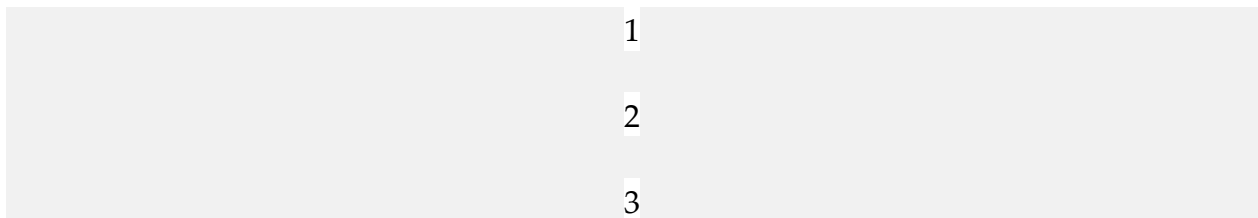
}
```

The flex container properties are:

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

The flex-direction Property

The flex-direction property defines in which direction the container wants to stack the flex items.



Example

The column value stacks the flex items vertically (from top to bottom):

```
.flex-container {
  display: flex;
  flex-direction: column;
}
```

Example

The column-reverse value stacks the flex items vertically (but from bottom to top):

```
.flex-container {  
  display: flex;  
  flex-direction: column-reverse;  
}
```

Example

The row value stacks the flex items horizontally (from left to right):

```
.flex-container {  
  display: flex;  
  flex-direction: row;  
}
```

Example

The row-reverse value stacks the flex items horizontally (but from right to left):

```
.flex-container {  
  display: flex;  
  flex-direction: row-reverse;  
}
```

The flex-wrap Property

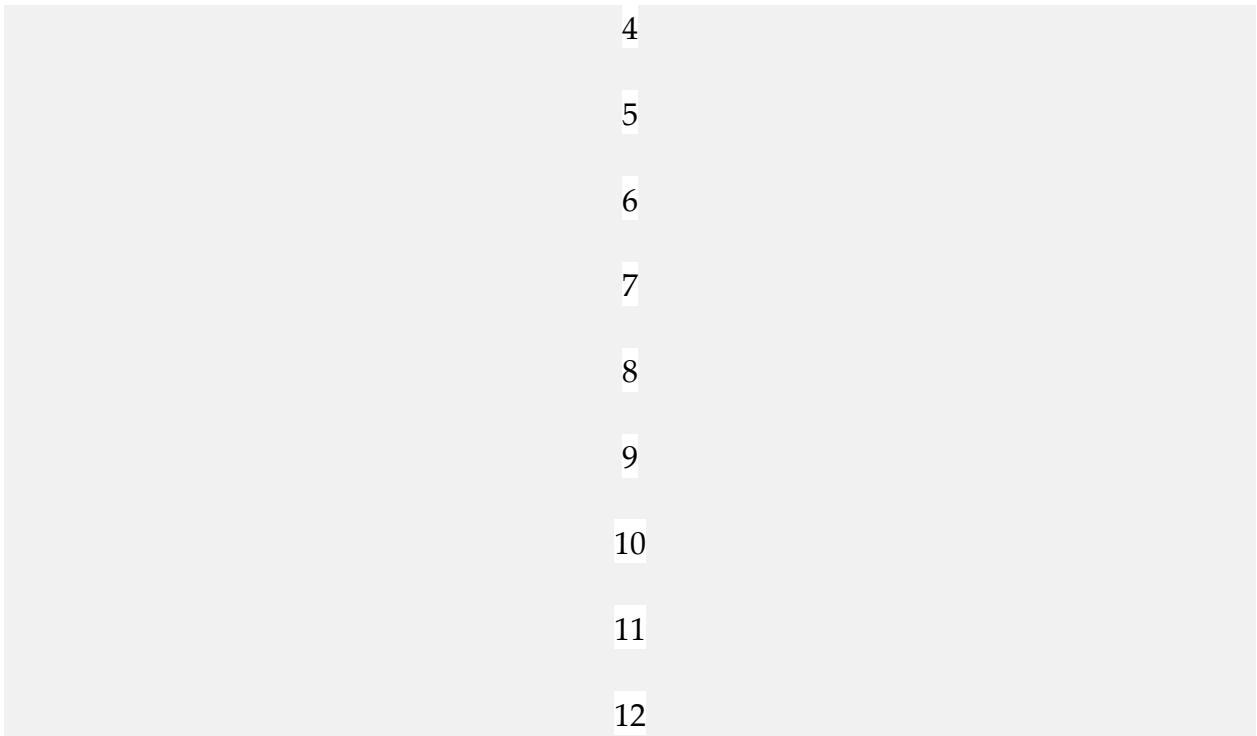
The flex-wrap property specifies whether the flex items should wrap or not.

The examples below have 12 flex items, to better demonstrate the flex-wrap property.

1

2

3



Example

The wrap value specifies that the flex items will wrap if necessary:

```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
}
```

Example

The nowrap value specifies that the flex items will not wrap (this is default):

```
.flex-container {  
  display: flex;  
  flex-wrap: nowrap;  
}
```

Example

The wrap-reverse value specifies that the flexible items will wrap if necessary, in reverse order:

```
.flex-container {  
  display: flex;  
  flex-wrap: wrap-reverse;  
}
```

The flex-flow Property

The flex-flow property is a shorthand property for setting both the flex-direction and flex-wrap properties.

Example

```
.flex-container {  
  display: flex;  
  flex-flow: row wrap;  
}
```

The CSS Flexbox Items Properties

The following table lists all the CSS Flexbox Items properties:

Property	Description
align-self	Specifies the alignment for a flex item (overrides the flex container's align-items property)
flex	A shorthand property for the flex-grow, flex-shrink, and the flex-basis properties
flex-basis	Specifies the initial length of a flex item
flex-grow	Specifies how much a flex item will grow relative to the rest of the flex items inside the same container

flex-shrink	Specifies how much a flex item will shrink relative to the rest of the flex items inside the same container
order	Specifies the order of the flex items inside the same container

CSS Media Queries

CSS2 Introduced Media Types

The @media rule, introduced in CSS2, made it possible to define different style rules for different media types.

Examples: You could have one set of style rules for computer screens, one for printers, one for handheld devices, one for television-type devices, and so on.

Unfortunately these media types never got a lot of support by devices, other than the print media type.

CSS3 Introduced Media Queries

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries is a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

Media Query Syntax

A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

```
@media not|only mediatype and (expressions) {
```


CSS-Code;

}

The result of the query is true if the specified media type matches the type of device the document is being displayed on and all expressions in the media query are true. When a media query is true, the corresponding stylesheet or style rules are applied, following the normal cascading rules.

Unless you use the not or only operators, the media type is optional and the all type will be implied.

You can also have different stylesheets for different media:

```
<link rel="stylesheet" media="mediatype and|not|only (expressions)" href="print.css">
```

CSS3 Media Types

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screen readers that "reads" the page out loud

Media Queries Simple Examples

One way to use media queries is to have an alternate CSS section right inside your style sheet.

The following example changes the background-color to light green if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the background-color will be pink):

Example

```
@media screen and (min-width: 480px) {
  body {
    background-color: light green;
  }
}
```

The following example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the menu will be on top of the content):

Example

```
@media screen and (min-width: 480px) {
  #left sidebar {width: 200px; float: left;}
  #main {margin-left: 216px;}
}
```

Wildcard Selectors (*, ^ and \$) in CSS for classes

Wildcard selector is used to select multiple elements simultaneously. It selects a similar type of class name or attribute and uses CSS property. * wildcard also known as containing wildcard.

[attribute*="str"] Selector: The [attribute*="str"] selector is used to select those elements whose attribute value contains the specified sub string *str*. This example shows how to use a wildcard to select all div with a class that contains *str*. This could be at the start, the end or in the middle of the class.

Syntax:

```
[attribute*="value"] {
  // CSS property
}
```

Example:

```

<!DOCTYPE html>
<html>
  <head>
    <style>

      /* Define styles of selected items, h1 and
      rest of the body */
      [class*="str"] { /* WE USE * HERE */
        background: green;
        color: white;
      }
      h1 {
        color:green;
      }
      body {
        text-align:center;
        width:60%;
      }
    </style>
  </head>
  <body>

    <!-- Since we have used * with str, all items with
    str in them are selected -->
    <div class="first_str">The first div element.</div>
    <div class="second">The second div element.</div>
    <div class="my-strt">The third div element.</div>
    <p class="mystr">Paragraph Text</p>
  </body>
</html>

```

Output:

The first div element.

The second div element.

The third div element.

Paragraph Text

[attribute^="str"] Selector: The [attribute^="value"] selector is used to select those elements whose attribute value begins with a specified value *str*. This example shows how to use a wildcard to select all div with a class that starts with *str*.

Syntax:

```
[attribute^="str"] {  
  
  // CSS property  
  
}
```

[attribute\$="str"] Selector: The [attribute\$="value"] selector is used to select those elements whose attribute value ends with a specified value *str*. The following example selects all elements with a class attribute value that ends with *str*.

Syntax:

```
[attribute$="str"] {  
  
  // CSS property  
  
}
```

Differences Between CSS1, CSS2 and CSS3

- CSS1 had a major difficulty with adaptation and consistency across different web browsers. The number of web browsers has also increased largely. CSS2 still has browser extension issues. CSS3 has complete support for almost all recent web browsers.
- CSS1 had limited styling options and old design influences. The properties and

add-ons have increased with CSS2 and further expanded with CSS3. CSS3 has support to add animations to your modern websites.

- CSS3 has compatibility with external font styles through google fonts and typecast. It was not possible with earlier CSS1 and CSS2.
- The selectors in CSS3 has increased while CSS1 and CSS2 only had simple selectors.
- CSS1 AND CSS2 didn't have provision to specifically design the web layout. It is possible with the CSS3 grid system and template layout module. It helped in creating layouts according to user components.
- CSS3 has now been split into various documents known as modules. Previously it was all a single document and limited features. The modularisation helped in working on a particular specification and faster evolution.
- CSS3 is compatible with Internet Explorer 9 whereas CSS1 used to be compatible with Internet Explorer 3.
- CSS3 supports responsive design. Responsive design is referred to as designing a website in such a manner that it looks good on all screen sizes. It should not break or misalign components on changing the screen size.

5.0 Bootstrap



Bootstrap is the popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.

What is Bootstrap

- Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.
- It is absolutely free to download and use.
- It is a front-end framework used for easier and faster web development.
- It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others.
- It can also use JavaScript plug-ins.

- It facilitates you to create responsive designs.



History of Bootstrap

Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter. It was released as an open source product in August 2011 on GitHub.

In June 2014 Bootstrap was the No.1 project on GitHub.

Why use Bootstrap

Following are the main advantage of Bootstrap:

- It is very easy to use. Anybody having basic knowledge of HTML and CSS can use Bootstrap.
- It facilitates users to develop a responsive website.
- It is compatible on most browsers like Chrome, Firefox, Internet Explorer, Safari and Opera etc.

What is a responsive website

A website is called a responsive website which can automatically adjust itself to look good on all devices, from smartphones to desktops etc.

What Bootstrap package contains

Scaffolding: Bootstrap provides a basic structure with Grid System, link styles, and background.

CSS: Bootstrap comes with the feature of global CSS settings, fundamental HTML elements style and an advanced grid system.

Components: Bootstrap contains a lot of reusable components built to provide iconography, dropdowns, navigation, alerts, pop-overs, and much more.

JavaScript Plugins: Bootstrap also contains a lot of custom jQuery plugins. You can easily include them all, or one by one.

Customize: Bootstrap components are customizable and you can customize Bootstrap's components, LESS variables, and jQuery plugins to get your own style.

What is Bootstrap 4?

Bootstrap is the newest and latest version of Bootstrap. It is the most popular HTML, CSS, JavaScript framework for developing responsive, mobile first websites.

Bootstrap 3 vs. Bootstrap 4

Bootstrap 4 has some new components, faster stylesheet, more buttons, effects and more responsiveness.

Bootstrap 4 supports some of the latest, stable releases of all major browsers and platforms.

Note: Internet Explorer 9 and down is not supported by Bootstrap 4. Although Internet Explorer 8-9 supported Bootstrap 3. So, if you have Internet Explorer 8-9, you should use Bootstrap 3. Bootstrap 3 is the most stable version of Bootstrap, and it is still supported by the team for critical bug fixes and documentation changes.

Is Bootstrap Best?

Bootstrap is more than efficient to create a responsive and mobile first website but it is not the best in the industry. There is an alternative of Bootstrap named W3.CSS which is smaller, faster, and easier to use

First Bootstrap Example

Add the HTML 5 doctype: Bootstrap uses HTML elements and CSS properties, so you have to add the HTML 5 doctype at the beginning of the page with lang attribute and correct character set.

Ex:

```
<!DOCTYPE html>
<html lang="en">
<head><meta http-equiv="Content-Type" content="text/html;
charset=windows-1252">
<title>Any title</title>
</head>
<body>
//write code
</body>
</html>
```

Bootstrap is mobile friendly: Bootstrap 3 is designed to be responsive to mobile devices.

Mobile-first styles are part of the core framework of Bootstrap. You have to add the following `<meta>` tag inside the `<head>` element for proper rendering and touch zooming:

1. `<meta name="viewport" content="width=device-width, initial-scale=1">`

Note: The `"width=device-width"` part is used to set the width of the page to follow the screen-width of the device (vary according to the devices).

The `initial-scale=1` part is used to set the initial zoom level when the page is first loaded by the browser.

Containers: container is used to wrap the site contents. There are two container classes.

- The `.container` class provides a responsive fixed width container.
- The `.container-fluid` class provides a full width container, spanning the entire width of the viewport.

Note: A container cannot be placed inside a container.

First Bootstrap Example (with responsive fixed width container)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>This is a Bootstrap example</title>
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css">
</head>
<body>
<div class="container">
  <h1> First Bootstrap web page</h1>
  <p>Write your text here..</p>
</div>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min
.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.m
in.js"></script>
</body>
</html>
```

Bootstrap Container

In Bootstrap, a container is used to set the content's margins dealing with the responsive behaviors of your layout. It contains the row elements and the row elements are the container of columns (known as grid system).

The **container class** is used to create boxed content.

There are two container classes in Bootstrap:

1. container
2. container-fluid

See the basic layout of a container:

```
<html>
<body>
  <div class="container">
    <div class="row">
      <div class="col-md-xx"></div>
      ...
    </div>
    <div class="row">
      <div class="col-md-xx"></div>
      ...
    </div>
  </div>
</body>
</html>
```

Bootstrap container example

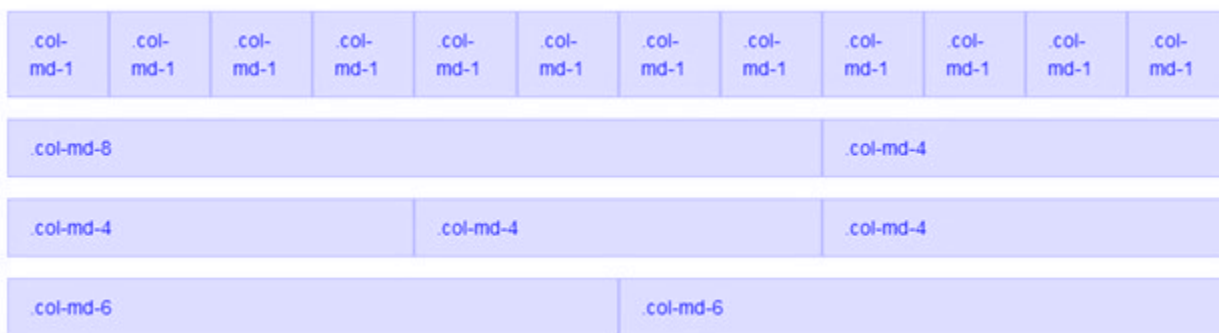
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Job</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css"/>
  </head>
  <body>
<div class="container">
  <h1>Container</h1>
  <p>container content</p>
</div>
<div class="container-fluid">
  <h1>Container-fluid</h1>
  <p>container-fluid content</p>
</div>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min
.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.m
in.js"></script>
  </body>
</html>
```

Bootstrap Grid

"In graphic design, a grid is a structure (usually two-dimensional) made up of a series of intersecting straight (vertical, horizontal) lines used to structure the content. It is widely used to design layout and content structure in print design. In web design, it is a very effective method to create a consistent layout rapidly and effectively using HTML and CSS."

Bootstrap Grid System

The Bootstrap Grid System allows up to 12 columns across the page. You can use all 12 columns individually or you can group the columns together to create wider columns.



Bootstrap Grid System is responsive and the columns are rearranged automatically according to the screen size.

Grid Classes:

There are four classes in Bootstrap Grid System:

- xs (for phones)
- sm (for tablets)
- md (for desktops)
- lg (for larger desktops)

You can combine the above classes to create more dynamic and flexible layouts.

Basic Structure of a Bootstrap Grid:

```

<div class="row">
  <div class="col-*-*"></div>
</div>
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
<div class="row">
  . . .
</div>

```

Follow the below instructions while creating a Bootstrap Grid:

- Create a row (<div class="row">).
- Add the number of columns, you want in the grid (tags with appropriate .col-*-* classes).
- Note that numbers in .col-*-* should always add up to 12 for each row.

Bootstrap Grid Example

For equal columns:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Job</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css"/>
  </head>
  <body>
<div class="container">
  <h1>Grid Example</h1>

  <div class="row">
    <div
class="col-md-3"style="background-color:lavender;">Rahul</div>
    <div

```

```

class="col-md-3"style="background-color:lavenderblush;">Vijay</div>
  <div
class="col-md-3"style="background-color:lavender;">Kartik</div>
  <div
class="col-md-3"style="background-color:lavenderblush;">Ajeet</div>
</div>
</div>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min
.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.m
in.js"></script>
</body>
</html>

```

Bootstrap Tables

We can create different types of Bootstrap tables by using different classes to style them.

Bootstrap Basic Table:

The basic Bootstrap table has a light padding and only horizontal dividers. The **.table class** is used to add basic styling to a table.

Example:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Job</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css"/>
  </head>
  <body>
<div class="container">
  <h1>Basic Table Example</h1>
<table class="table">
  <tr><th>Id</th><th>Name</th><th>Age</th></tr>
  <tr><td>101</td><td>Rahul</td><td>23</td></tr>
  <tr><td>102</td><td>Umesh</td><td>22</td></tr>
  <tr><td>103</td><td>Max</td><td>29</td></tr>
  <tr><td>104</td><td>Ajeet</td><td>21</td></tr>
</table>

```

```

</div>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min
.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.m
in.js"></script>
  </body>
</html>

```

Bootstrap Striped Rows Table:

The **.table-striped** class is used to add zebra-stripes to a table:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Job</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css"/>
  </head>
  <body>
    <div class="container">
      <h1>Striped Table Example</h1>
      <table class="table table-striped">
        <tr><th>Id</th><th>Name</th><th>Age</th></tr>
        <tr><td>101</td><td>Rahul</td><td>23</td></tr>
        <tr><td>102</td><td>Umesh</td><td>22</td></tr>
        <tr><td>103</td><td>Max</td><td>29</td></tr>
        <tr><td>104</td><td>Ajeet</td><td>21</td></tr>
      </table>
    </div>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min
.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.m
in.js"></script>
  </body>
</html>

```

Bootstrap Contextual classes:.

Contextual classes are used to color table rows (<tr>) or table cells (<td>):

Following are the different contextual classes:

Class	Description
.active	It is used to apply the hover color to the table row or table cell
.success	It indicates a successful or positive action
.info	It indicates a neutral informative change or action
.warning	It specifies a warning that might need attention
.danger	It indicates a dangerous or potentially negative action

Bootstrap Forms

In Bootstrap, there are three types of form layouts:

- Vertical form (this is default)
- Horizontal form
- Inline form

Bootstrap Form Rules

There are three standard rules for these 3 form layouts:

- Always use <form role="form"> (helps improve accessibility for people using screen readers)
- Wrap labels and form controls in <div class="form-group"> (needed for optimum spacing)
- Add class .form-control to all textual <input>, <textarea>, and <select> elements

1) Bootstrap Vertical Form (Default)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Bootstrap example</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css"/>
  </head>
  <body>

<div class="container">
  <h1>Vertical Form Example</h1>

<form style="width:300px">
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control"
id="exampleInputEmail1" placeholder="Email">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control"
id="exampleInputPassword1" placeholder="Password">
  </div>

  <button type="submit" class="btn btn-default">Login</button>
</form>

</div>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min
.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.m
in.js"></script>
</body>
</html>
```

2) Bootstrap Inline Form

In Bootstrap Inline forms, all elements are inline, left-aligned, and the labels are alongside.

This example is only applied to forms within viewports that are at least 768px wide!

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css">
</head>
<body>
<div class="container">
  <h2>Inline form Example</h2>
  <form class="form-inline" role="form">
    <form style="width:300px">
      <div class="form-group">
        <label for="exampleInputEmail1">Email address</label>
        <input type="email" class="form-control"
id="exampleInputEmail1" placeholder="Email">
      </div>
      <div class="form-group">
        <label for="exampleInputPassword1">Password</label>
        <input type="password" class="form-control"
id="exampleInputPassword1" placeholder="Password">
      </div>
      <button type="submit" class="btn btn-default">Login</button>
    </form>
  </div>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min
.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.m
in.js"></script>
</body>
</html>
```

3) Bootstrap Horizontal Form

You have to add some additional rules if you want to create a horizontal form.

Additional rules for a horizontal form:

- Add class `.form-horizontal` to the `<form>` element
- Add class `.control-label` to all `<label>` elements

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css">
</head>
<body>

<div class="container">
  <h2>Horizontal form Example</h2>
  <form class="form-horizontal" role="form">
    <form style="width:300px">
      <div class="form-group">
        <label class="control-label col-sm-2"
for="email">Email:</label>
        <div class="col-sm-10">
          <input type="email" class="form-control" id="email"
placeholder="Enter email">
        </div>
      </div>
      <div class="form-group">
        <label class="control-label col-sm-2"
for="pwd">Password:</label>
        <div class="col-sm-10">
          <input type="password" class="form-control" id="pwd"
placeholder="Enter password">
        </div>
      </div>
    </form>
  </div>
</div>
```

```

        <div class="col-sm-offset-2 col-sm-10">
            <button type="submit" class="btn
btn-default">Submit</button>
        </div>
    </div>
</form>
</div>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min
.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.m
in.js"></script>

</body>
</html>

```

Bootstrap Images

Bootstrap supports images. There are three classes in Bootstrap that can be used to apply some simple style to the images.

The following classes add style to the images:

Classes	Uses
.img-rounded	It adds border-radius:6px to give the image rounded corners.
.img-circle	It makes the entire image round by adding border-radius:500px.
.img-thumbnail	It adds a bit of padding and a gray border.

Bootstrap Image-rounded Example

The **class .img-rounded** is used to add rounded corners to an image (IE8 does not support rounded corners).

Example:

```

<!DOCTYPE html>
<html lang="en">

```

```

<head>
  <title>Bootstrap image</title>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css">
</head>
<body>
<div class="container">
  <h2>Rounded Corners</h2>
  
</div>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min
.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.m
in.js"></script>
</body>
</html>

```

Bootstrap Image-circle Example

The **class .img-circle** is used to shape the image to a circle (IE8 does not support rounded corners).

Example:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Bootstrap image</title>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css">
</head>
<body>
<div class="container">
  <h2>Circle</h2>
  
</div>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min

```

```

.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.m
in.js"></script>
</body>
</html>

```

Bootstrap Thumbnail Image Example

The **class .img-thumbnail** is used to shape an image to a thumbnail.

Example:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Bootstrap image</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css">
  </head>
  <body>
    <div class="container">
      <h2>Thumbnail</h2>
      
    </div>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min
.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.m
in.js"></script>
  </body>
</html>

```

Bootstrap Responsive images

The responsive images can adjust themselves automatically to fit the size of screen. You can create responsive images by adding an **.img-responsive class** to the `` tag. The image will then scale nicely to the parent element.

The **.img-responsive class** applies `display: block;` and `max-width: 100%;` and `height: auto;`

to the image.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css">
</head>
<body>
  <div class="container">
    <h2>Responsive Image</h2>
    
  </div>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min
.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.m
in.js"></script>
</body>
</html>
```

Typography

Typography provides some utilities to add additional styles to texts.

These utilities are:

- Text alignment
- Text transform
- Font weight and italics

Text Alignment

Text alignment is used to easily realign text to components with text alignment classes

syntax:

Align text left:

1. `<p class="text-left">Left aligned text.</p>`

Align text center:

1. `<p class="text-center">Center aligned text.</p>`

Align text right:

1. `<p class="text-right">Right aligned text.</p>`

Align text justify:

1. `<p class="text-justify">Justified text.</p>`

Align text no-wrap:

1. `<p class="text-nowrap">No wrap text.</p>`

You can align text on viewports according to their size also.

Left aligned text on viewports sized SM (small) or wider.

1. `<p class="text-sm-left">Left aligned text on viewports sized SM (small) or wider.</p>`

Left aligned text on viewports sized MD (medium) or wider.

1. `<p class="text-md-left">Left aligned text on viewports sized MD (medium) or wider.</p>`

Left aligned text on viewports sized LG (large) or wider.

1. `<p class="text-lg-left">Left aligned text on viewports sized LG (large) or wider.</p>`

Left aligned text on viewports sized XL (extra-large) or wider.

1. `<p class="text-xl-left">Left aligned text on viewports sized XL (extra-large) or wider.</p>`

Text transform

The text capitalization classes are used to transform text in components.

For lowercase text:

Use "text-lowercase" class to make the text appear in lowercase.

Syntax:

1. `<p class="text-lowercase">Lowercase text.</p>`

Output:

lowercase text.

For uppercase text:

Use the "text-uppercase" class to make the text appear in uppercase.

Syntax:

1. `<p class="text-uppercase">Uppercased text.</p>`

Output:

UPPERCASED TEXT.

For capitalized text:

Use "text- capitalize" class to make the text's first letter appear in uppercase.

Syntax:

1. `<p class="text-capitalize">CapiTaliZed text.</p>`

Output:

CapiTaliZed Text.

Font weight and italics

It is used to quickly change the weight (boldness) of text or italicize text.

For bold text:

Use "font-weight-bold" class to make the text weight bold.

Syntax:

1. `<p class="font-weight-bold">Bold text.</p>`

Output:

Bold text.

For normal weight text:

Use "font-weight-normal" class to make the text normal weight.

Syntax:

1. `<p class="font-weight-normal">Normal weight text.</p>`

Output:

Normal weight text.

For Italic text:

Use "font-italic" class to make the text italic.

Syntax:

1. `<p class="font-italic">Italic text.</p>`

6.0 Java Script

Why Study JavaScript?

JavaScript is one of the 3 languages all web developers must learn:

1. HTML to define the content of web pages
2. CSS to specify the layout of web pages
3. JavaScript to program the behavior of web pages

Server-side Scripting Vs Client-side Scripting

BASIS FOR COMPARISON	SERVER-SIDE SCRIPTING	CLIENT-SIDE SCRIPTING
Basic	Works in the back end which could not be visible at the client end.	Works at the front end and script are visible among the users.

Processing	Requires server interaction.	Does not need interaction with the server.
Languages involved	PHP, ASP.net, Ruby on Rails, ColdFusion, Python, etcetera.	HTML, CSS, JavaScript, etc.
Affect	Could effectively customize the web pages and provide dynamic websites.	Can reduce the load to the server.
Security	Relatively secure.	Insecure

Difference between Java and JavaScript

JAVA	JAVASCRIPT
Java is an object oriented programming language.	JavaScript is an object based scripting language.
Java applications can run in any virtual machine(JVM) or browser.	JavaScript code runs on browser only as JavaScript is developed for browser only.
Java program has file extension “.Java” and translates source code into bytetimes which is executed by JVM(Java Virtual Machine).	JavaScript file has file extension “.js” and it is interpreted but not compiled, every browser has the Javascript interpreter to execute JS code.
Java programs use more memory.	JavaScript requires less memory therefore it is used in web pages.

How do I write JavaScript?

```
<script type="text/javascript">  
Your JavaScript code goes here.  
</script>
```

```
<script type="text/javascript">  
  
document.write("Text written using JavaScript code!");  
  
</script>
```

You would then save the file with a .js extension. For instance, you could save it as **write.js**. Once the file is saved, you can call it from the HTML code via the **src** attribute of the opening script tag, as shown below for write.js.

```
<script type="text/javascript" src="write.js"></script>
```

What are JavaScript Identifiers?

JavaScript Identifiers are names given to variables, functions, etc. It is the same as identifiers in other programming languages like C, C++, Java, etc. Let's see identifiers for variable names.

JavaScript Reserved Keywords List

Keywords are reserved words in JavaScript that you cannot use to indicate variable labels or function names. There are a total of 63 keywords that JavaScript provides to programmers. All of them are shown in the below-mentioned table:

abstract	arguments	boolean	break	byte	case	catch	char	const
debugger	default	delete	do	double	else	continue	function	goto

JavaScript Comments

There are two types of comments in JavaScript.

Single-line Comment

Multi-line Comment

<script>

```
// It is single line comment
document.write("hello javascript");
```

</script>

<script>

```
/* It is a multi line comment.
It will not be displayed */
document.write("example of javascript multiline comment");
```

</script>

JavaScript Variables

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore(_), or dollar(\$) sign.
2. After the first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

Correct JavaScript variables

1. var x = 10;
2. var _value="sonoo";

Incorrect JavaScript variables

1. var 123=30;

2. `var *aa=320;`

Example of JavaScript variable

```
<script>  
var x = 10;  
var y = 20;  
var z=x+y;  
document.write(z);  
</script>
```

Output of the above example

30

JavaScript local variable

A JavaScript local variable is declared inside a block or function. It is accessible within the function or block only. For example:

```
<script>  
function abc() {  
var x=10;//local variable  
}  
</script>
```

Or,

```
<script>  
If(10<13) {  
var y=20;//JavaScript local variable  
}  
</script>
```

JavaScript global variable

A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with a window object is known as global variable. For example:

```
<script>
var data=200;//global variable
function a(){
document.writeln(data);
}
function b(){
document.writeln(data);
}
a();//calling JavaScript function
b();
</script>

200 200
```

JavaScript Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, meaning you don't need to specify type of the variable because it is dynamically used by the JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

1. var a=40;//holding number
2. var b="Rahul";//holding string

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript Operators

JavaScript operators are symbols that are used to perform operations on operands. For example:

1. `var sum=10+20;`

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

JavaScript Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
++ /	Increment Division

JavaScript Comparison Operators

Operator	Description
==	Is equal to
!=	Not equal to
>	Greater than
<=	Less than or equal to

JavaScript Logical Operators

Operator	Description
----------	-------------

&&	Logical AND
	Logical OR
!	Logical Not

typeof Operator

The typeof operator is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.

The typeof operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.

Here is a list of the return values for the typeof Operator.

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	Function

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 10;
        var b = "String";
      -->
    </script>
  </body>
</html>
```

```

        var line break = "<br />";
        result = (typeof b == "string" ? "B is String" : "B is
Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);
        result = (typeof a == "string" ? "A is String" : "A is
Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);
    //-->
</script>
<p>Set the variables to different values and different
operators and then try...</p>
</body>
</html>

```

output:

Result => B is String

Result => A is Numeric

Set the variables to different values and different operators and then try...

JavaScript | Type Conversion

The process of converting one data type to another data type is called type conversion. There are two types of type conversion in JavaScript.

- Implicit Conversion
- Explicit Conversion

JavaScript Implicit Conversion

In certain situations, JavaScript automatically converts one data type to another (to the right type). This is known as implicit conversion.

Example 1: Implicit Conversion to String

// numeric string used with + gives string type

```
let result;
result = '3' + 2;
console.log(result) // "32"
result = '3' + true;
console.log(result); // "3true"
result = '3' + undefined;
console.log(result); // "3undefined"
result = '3' + null;
console.log(result); // "3null"
```

Note: When a number is added to a string, JavaScript converts the number to a string before concatenation.

Example 2: Implicit Conversion to Number

// numeric string used with -, /, * results number type

```
let result;

result = '4' - '2';
console.log(result); // 2

result = '4' - 2;
console.log(result); // 2

result = '4' * 2;
console.log(result); // 8

result = '4' / 2;
console.log(result); // 2
```

JavaScript Explicit Conversion

You can also convert one data type to another as per your needs. The type conversion that you do manually is known as explicit type conversion.

In JavaScript, explicit type conversions are done using built-in methods.

Here are some common methods of explicit conversions.

1. Convert to Number Explicitly

To convert numeric strings and boolean values to numbers, you can use `Number()`. For example,

```
let result;

// string to number
result = Number('324');
console.log(result); // 324

result = Number('324e-1')
console.log(result); // 32.4

// boolean to number
result = Number(true);
console.log(result); // 1
```

```
result = Number(false);
console.log(result); // 0
```

In JavaScript, empty strings and null values return 0. For example,

```
let result;
result = Number(null);
console.log(result); // 0
```

```
let result = Number(' ');
console.log(result); // 0
```

If a string is an invalid number, the result will be NaN. For example,

```
let result;
result = Number('hello');
console.log(result); // NaN
```

```
result = Number(undefined);
console.log(result); // NaN
```

```
result = Number(NaN);
console.log(result); // NaN
```

Note: You can also generate numbers from strings using `parseInt()`, `parseFloat()`, unary operator `+` and `Math.floor()`. For example,

```
let result;
result = parseInt('20.01');
console.log(result); // 20

result = parseFloat('20.01');
console.log(result); // 20.01

result = +'20.01';
console.log(result); // 20.01

result = Math.floor('20.01');
console.log(result); // 20
```

Inline vs External JavaScript

In HTML documents and web pages, there are two ways to include JavaScript code. One way is to include the code internally, known as the inline method. Using this method, the code is written inside the `<script>` element.

In the external method, the JavaScript code is contained in an entirely separate external document, with a file extension of `.js`. This extension, as you probably have already guessed, stands for JavaScript. The external file will then be referenced in the HTML document using the syntax: `<script src = "externalFile.js"></script>`

An important point to remember about the above syntax is that if there is any inline JavaScript code included between the `<script>` and `</script>` HTML tags, it will not execute because the browser has been "told" to look for JavaScript code outside the HTML document.

Now let's see an example of each method and discuss the differences in approach.

Inline JavaScript

In the following example, the HTML document contains a form, and underneath it follows the `<script>` element with the JavaScript code. The function in the code is called directly from inside the HTML form, using an event handler.

```
<!DOCTYPE html>
<body>
  <form onsubmit="inline(); return false">
    Password:
```

```



```

Let's see how this works in a little more detail. As mentioned above, the JavaScript code in the <script> element contains the function inline(). The commands in this function will be executed only when the user clicks the "Submit" button. This is specified in the HTML line that begins the form:

```
<form onsubmit="inline(); return false">
```

The inline() function's purpose is to determine if the user has entered a value in the "password" field, and if they have, to alert the message "Your password has been successfully submitted" inside the <textarea> element.

Now let's see how to create an HTML document that will look and work the same (as far as the user is concerned), but will do so using an external JavaScript file.

External JavaScript

This is the HTML document that references the external JavaScript file:

```

<!DOCTYPE html>
<body>
<form>
Password:
<input type = "password">

```



```
Message:
<textarea rows="2" cols="50">
</textarea>
<input type = "submit">
</form>
<script src = "external.js">
</script>
</body>
</html>
```

And this is the code in the external JavaScript file:

```
var submitButton = document.getElementsByTagName("input")[1];
submitButton.addEventListener("click", external);
function external(e) {
var password =
document.getElementsByTagName("input")[0].value;
var input = document.getElementsByTagName("textarea")[0];
if (password != "") {
input.value = "Your password has been successfully
submitted.";
}
e.preventDefault();
}
```

Starting with the HTML document, notice that overall it's a smaller document compared to the first example, because it doesn't need to contain as much information. The two really important differences are:

- The JavaScript code is stored in the external file external.js and is referenced in this line: <script src = "external.js">
- The <form> element no longer contains the onsubmit event handler, because this too is now included in the external JavaScript.

So let's move on to the actual code, in the obviously-named external.js file:

The first line of code identifies the form's "Submit" button and stores it in the submitButton variable:

```
var submitButton = document.getElementsByTagName("input")[1];
```

The next line adds an event listener, so that when the "Submit" button is clicked, it triggers the external() function:

```
submitButton.addEventListener("click", external);
```

The `external()` function is almost identical to the `inline()` function in the first example; it performs the same actions, displaying a message only if the user has entered something in the password field.

There is only one real difference between the two functions. In the external function, we included an argument, 'e'. This is an arbitrary value, only so we could attach to it the `preventDefault()` method in the function's last line:

```
e.preventDefault();
```

This allows the results to remain displayed on the page after the user clicks "Submit". Without this, the results would appear for a split second, then the page would reload.

JavaScript Variable

A JavaScript variable is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore (_), or dollar (\$) sign.
2. After the first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

Correct JavaScript variables

1. `var x = 10;`
2. `var _value="sonoo";`

Incorrect JavaScript variables

1. `var 123=30;`
2. `var *aa=320;`

Example of JavaScript variable

Let's see a simple example of a JavaScript variable.

```
<script>  
var x = 10;
```

```
var y = 20;  
var z=x+y;  
document.write(z);  
</script>
```

Output of the above example

30

JavaScript local variable

A JavaScript local variable is declared inside a block or function. It is accessible within the function or block only. For example:

```
<script>  
function abc() {  
var x=10;//local variable  
}  
</script>
```

Or,

```
<script>  
If(10<13) {  
var y=20;//JavaScript local variable  
}  
</script>
```

JavaScript global variable

A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with a window object is known as global variable. For

example:

```
<script>
var data=200;//global variable
function a(){
document.writeln(data);
}
function b(){
document.writeln(data);
}
a();//calling JavaScript function
b();
</script>
```

Output:

200 200

Declaring JavaScript global variable within function

To declare JavaScript global variables inside a function, you need to use a window **object**.

For example:

```
window.value=90;
```

Now it can be declared inside any function and can be accessed from any function. For

example:

```
function m(){
window.value=100;//declaring global variable by window object
}
function n(){
alert(window.value);//accessing global variable from other function
}
```

JavaScript Array

JavaScript array is an object that represents a collection of similar types of elements.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

1) JavaScript array literal

The syntax of creating array using array literal is given below:

1. `var arrayname=[value1,value2.....valueN];`

As you can see, values are contained inside [] and separated by , (comma).

Let's see the simple example of creating and using arrays in JavaScript.

```
<script>  
var emp=["Sonoo", "Vimal", "Ratan"];  
for (i=0;i<emp.length;i++){  
document.write(emp[i] + "<br/>");  
}  
</script>
```

The .length property returns the length of an array.

Output of the above example

Sonoo

Vimal

Ratan

2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

1. `var arrayname= new Array();`

Here, **new keyword** is used to create instances of arrays.

Let's see the example of creating an array directly.

```
<script>  
var i;  
var emp = new Array();  
emp[0] = "Arun";  
emp[1] = "Varun";  
emp[2] = "John";  
  
for (i=0;i<emp.length;i++){  
document.write(emp[i] + "<br>");  
}  
</script>
```

Output of the above example

Arun

Varun

John

3) JavaScript array constructor (new keyword)

Here, you need to create an array of arrays by passing arguments in constructor so that we don't have to provide value explicitly.

The example of creating objects by array constructor is given below.

```
<script>  
var emp=new Array("Jai","Vijay","Smith");  
for (i=0;i<emp.length;i++){  
document.write(emp[i] + "<br>");  
}  
</script>
```

Output of the above example

Jai

Vijay

Smith

JavaScript Array Methods

Let's see the list of JavaScript array methods with their description.

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
copywithin()	It copies the part of the given array with its own elements and returns the modified array.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the

	provided function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
filter()	It returns the new array containing the elements that pass the provided function conditions.
find()	It returns the value of the first element in the given array that satisfies the specified condition.
findIndex()	It returns the index value of the first element in the given array that satisfies the specified condition.
forEach()	It invokes the provided function once for each element of an array.
includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
isArray()	It tests if the passed value is an array.
join()	It joins the elements of an array as a string.
keys()	It creates an iterator object that contains only the keys of the array, then loops through these keys.
lastIndexOf()	It searches the specified element in the given array and returns the index of the last match.
map()	It calls the specified function for every array element and returns the

	new array
of()	It creates a new array from a variable number of arguments, holding any type of argument.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of the given array.
reduce(function, initial)	It executes a provided function for each value from left to right and reduces the array to a single value.
reduceRight()	It executes a provided function for each value from right to left and reduces the array to a single value.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It adds/removes elements to/from the given array.
toLocaleString()	It returns a string containing all the elements of a specified array.
toString()	It converts the elements of a specified array into string form, without affecting the original array.
unshift()	It adds one or more elements in the beginning of the given array.
values()	It creates a new iterator object carrying values for each index in the array.

JavaScript Functions

JavaScript functions are used to perform operations. We can call a JavaScript function many times to reuse the code.

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

1. **Code reusability:** We can call a function several times so it saves coding.
2. **Less coding:** It makes our program compact. We don't need to write many lines of code each time to perform a common task.

JavaScript Function Syntax

The syntax of the declaring function is given below.

```
function functionName([arg1, arg2, ...argN]) {  
  
    //code to be executed  
  
}
```

JavaScript Functions can have 0 or more arguments.

JavaScript Function Example

Let's see a simple example of a function in JavaScript that does not have arguments.

```
<script>
```

```
function msg() {  
  
alert("hello! this is message");  
  
}
```

```
</script>
```

```
<input type="button" onclick="msg()" value="call function"/>
```

JavaScript Function Arguments

We can call functions by passing arguments. Let's see the example of a function that has one argument.

```
<script>
```

```
function getcube(number) {  
alert(number*number*number);  
}
```

```
</script>
```

```
<form>
```

```
<input type="button" value="click" onclick="getcube(4)"/>
```

```
</form>
```

Function with Return Value

We can call a function that returns a value and use it in our program. Let's see the example of a function that returns value.

```
<script>
```

```
function getInfo() {  
return "hello Students! How r u?";  
}
```

```
</script>
```

```
<script>
document.write(getInfo());
</script>
```

Output of the above example

```
hello Students! How r u?
```

JavaScript Function Object

In JavaScript, the purpose of Function constructor is to create a new Function object. It executes the code globally. However, if we call the constructor directly, a function is created dynamically but in an unsecured way.

Syntax

1. **new Function ([arg1[, arg2[,argn]],] functionBody)**

Parameter

arg1, arg2, , argn - It represents the argument used by function.

functionBody - It represents the function definition.

JavaScript Function Methods

Let's see function methods with description.

Method	Description
apply()	It is used to call a function containing this value and a single array of

	arguments.
bind()	It is used to create a new function.
call()	It is used to call a function containing this value and an argument list.
toString()	It returns the result in a form of a string.

JavaScript Function Object Examples

Example 1

Let's see an example to display the sum of given numbers.

```
<script>
var add=new Function("num1","num2","return num1+num2");
document.writeln(add(2,5));
</script>
```

Output:

7

Example 2

Let's see an example to display the power of provided value.

```
<script>
var pow=new Function("num1","num2","return Math.pow(num1,num2)");
document.writeln(pow(2,3));
</script>
```

Output:

8

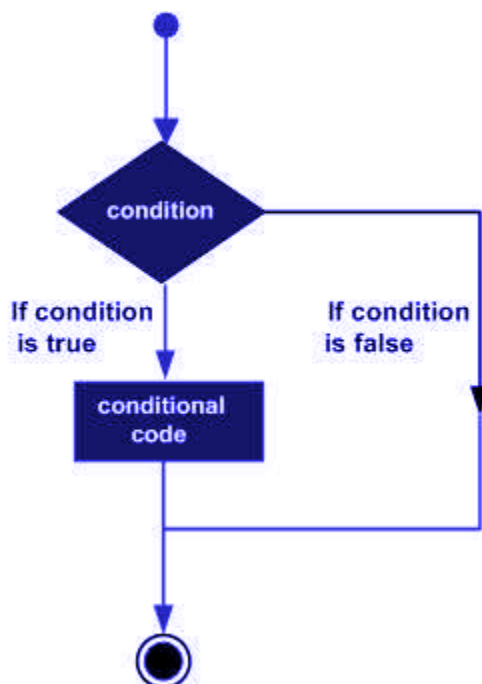
JavaScript - if...else Statement

While writing a program, there may be a situation when you need to adopt one out of a given set of paths. In such cases, you need to use conditional statements that allow your program to make correct decisions and perform right actions.

JavaScript supports conditional statements which are used to perform different actions based on different conditions. Here we will explain the if..else statement.

FlowChart of if-else

The following flow chart shows how the if-else statement works.



JavaScript supports the following forms of if..else statement –

- if statement
- if...else statement
- if...else if... statement.

if statement

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax

The syntax for a basic if statement is as follows –

```
if (expression) {  
    Statement(s) to be executed if expression is true  
}
```

Here a JavaScript expression is evaluated. If the resulting value is true, the given statement(s) are executed. If the expression is false, then no statement would be not executed. Most of the time, you will use comparison operators while making decisions.

Example

Try the following example to understand how the if statement works.

```
<html>  
  <body>  
    <script type = "text/javascript">  
      <!--  
        var age = 20;  
  
        if( age > 18 ) {
```

```
        document.write("<b>Qualifies for driving</b>");
    }
    //-->
</script>
<p>Set the variable to a different value and then try...</p>
</body>
</html>
```

Output

Qualifies for driving

Set the variable to a different value and then try...

if...else statement

The 'if...else' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Syntax

```
if (expression) {
    Statement(s) to be executed if expression is true
} else {
    Statement(s) to be executed if expression is false
}
```

Here the JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the 'if' block are executed. If the expression is false, then the given statement(s) in the else block are executed.

Example

Try the following code to learn how to implement an if-else statement in JavaScript.

```
<html>
```



```

<body>
  <script type = "text/javascript">
    <!--
      var age = 15;

      if( age > 18 ) {
        document.write("<b>Qualifies for driving</b>");
      } else {
        document.write("<b>Does not qualify for
driving</b>");
      }
    </script>
    <p>Set the variable to a different value and then try...</p>
  </body>
</html>

```

Output

Does not qualify for driving

Set the variable to a different value and then try...

if...else if... statement

The if...else if... statement is an advanced form of if...else that allows JavaScript to make a correct decision out of several conditions.

Syntax

The syntax of an if-else-if statement is as follows –

```

if (expression 1) {
  Statement(s) to be executed if expression 1 is true
} else if (expression 2) {
  Statement(s) to be executed if expression 2 is true
} else if (expression 3) {
  Statement(s) to be executed if expression 3 is true

```

```
} else {  
    Statement(s) to be executed if no expression is true  
}
```

There is nothing special about this code. It is just a series of if statements, where each if is a part of the else clause of the previous statement. Statement(s) are executed based on the true condition, if none of the conditions is true, then the else block is executed.

Example

Try the following code to learn how to implement an if-else-if statement in JavaScript.

```
<html>  
  <body>  
    <script type = "text/javascript">  
      <!--  
        var book = "maths";  
        if( book == "history" ) {  
          document.write("<b>History Book</b>");  
        } else if( book == "maths" ) {  
          document.write("<b>Maths Book</b>");  
        } else if( book == "economics" ) {  
          document.write("<b>Economics Book</b>");  
        } else {  
          document.write("<b>Unknown Book</b>");  
        }  
      //-->  
    </script>  
    <p>Set the variable to a different value and then try...</p>  
  </body>  
</html>
```

Output

Maths Book

Set the variable to a different value and then try...

JavaScript - For Loop

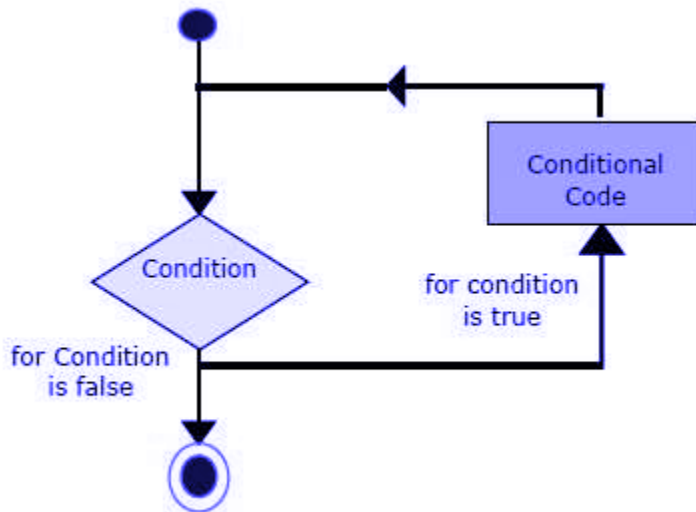
The 'for' loop is the most compact form of looping. It includes the following three important parts –

- The loop initialization where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The test statement which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The iteration statement where you can increase or decrease your counter.

You can put all the three parts in a single line separated by semicolons.

Flow Chart

The flow chart of a for loop in JavaScript would be as follows –



Syntax

The syntax of for loop in JavaScript is as follows –

```

for (initialization; test condition; iteration statement) {
  Statement(s) to be executed if test condition is true
}
  
```

Example

Try the following example to learn how a for loop works in JavaScript.

```

<html>
  <body>
    <script type = "text/javascript">
      <!--
        var count;
        document.write("Starting Loop" + "<br />");

        for(count = 0; count < 10; count++) {
          document.write("Current Count : " + count );
          document.write("<br />");
        }
        document.write("Loop stopped!");
      //-->
    </script>
  </body>
</html>
  
```

```
        </script>
        <p>Set the variable to a different value and then try...</p>
    </body>
</html>
```

Output

Starting Loop

Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Current Count : 9

Loop stopped!

Set the variable to a different value and then try...

JavaScript for...in loop

The for...in loop is used to loop through an object's properties. As we have not discussed Objects yet, you may not feel comfortable with this loop. But once you understand how objects behave in JavaScript, you will find this loop very useful.

Syntax

The syntax of 'for..in' loop is –

```
for (variablename in object) {
    statement or block to execute
```

```
}
```

In each iteration, one property from the object is assigned to variable name and this loop continues till all the properties of the object are exhausted.

Example

Try the following example to implement a 'for-in' loop. It prints the web browser's Navigator object.

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var aProperty;
        document.write("Navigator Object Properties<br /> ");
        for (aProperty in navigator) {
          document.write(aProperty);
          document.write("<br />");
        }
        document.write ("Exiting from the loop!");
      //-->
    </script>
    <p>Set the variable to a different object and then try...</p>
  </body>
</html>
```

Output

```
Navigator Object Properties
serviceWorker
webkitPersistentStorage
webkitTemporaryStorage
geolocation
doNotTrack
onLine
languages
language
userAgent
```

product
platform
appVersion
appName
appCodeName
hardwareConcurrency
maxTouchPoints
vendorSub
vendor
productSub
cookieEnabled
mimeTypes
plugins
javaEnabled
getStorageUpdates
getGamepads
webkitGetUserMedia
vibrate
getBattery
sendBeacon
registerProtocolHandler
unregisterProtocolHandler
Exiting from the loop!
Set the variable to a different object and then try...

JavaScript - Loop Control

JavaScript provides full control to handle loops and switch statements. There may be a situation when you need to come out of a loop without reaching its bottom. There may also be a situation when you want to skip a part of your code block and start the next iteration of the loop.

To handle all such situations, JavaScript provides break and continue statements. These statements are used to immediately come out of any loop or to start the next iteration of

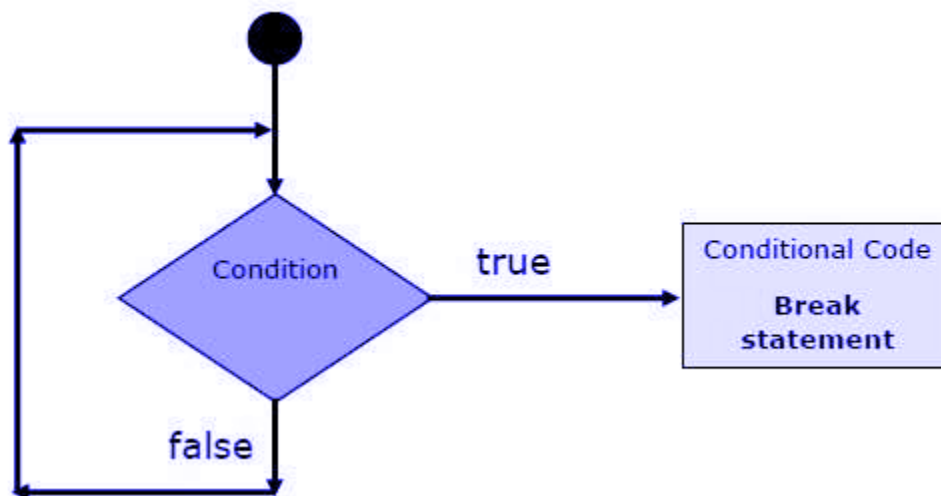
any loop respectively.

The break Statement

The `break` statement, which was briefly introduced with the `switch` statement, is used to exit a loop early, breaking out of the enclosing curly braces.

Flow Chart

The flow chart of a `break` statement would look as follows –



Example

The following example illustrates the use of a `break` statement with a `while` loop. Notice how the loop breaks out early once `x` reaches 5 and reaches to `document.write (..)` statement just below to the closing curly brace –

```
<html>  
  <body>
```



```

<script type = "text/javascript">
  <!--
  var x = 1;
  document.write("Entering the loop<br /> ");

  while (x < 20) {
    if (x == 5) {
      break;    // breaks out of loop completely
    }
    x = x + 1;
    document.write( x + "<br />");
  }
  document.write("Exiting the loop!<br /> ");
  //-->
</script>

  <p>Set the variable to a different value and then try...</p>
</body>
</html>

```

Output

Entering the loop

2
3
4
5

Exiting the loop!

Set the variable to a different value and then try...

The continue Statement

The continue statement tells the interpreter to immediately start the next iteration of the loop and skip the remaining code block. When a continue statement is encountered, the program flow moves to the loop check expression immediately and if the condition

remains true, then it starts the next iteration, otherwise the control comes out of the loop.

Example

This example illustrates the use of a continue statement with a while loop. Notice how the continue statement is used to skip printing when the index held in variable x reaches 5 –

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var x = 1;
        document.write("Entering the loop<br /> ");

        while (x < 10) {
          x = x + 1;

          if (x == 5) {
            continue;    // skip rest of the loop body
          }
          document.write( x + "<br />");
        }
        document.write("Exiting the loop!<br /> ");
      //-->
    </script>
    <p>Set the variable to a different value and then try...</p>
  </body>
</html>
```

Output

Entering the loop

2
3
4
6
7
8

9

10

Exiting the loop!

Set the variable to a different value and then try...

JavaScript Popup Boxes

JavaScript has three kinds of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
window.alert("sometext");
```

The **window.alert()** method can be written without the window prefix.

Example

```
alert("I am an alert box!");
```

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
window.confirm("sometext");
```

The `window.confirm()` method can be written without the window prefix.

Example

```
if (confirm("Press a button!")) {  
    txt = "You pressed OK!";  
} else {  
    txt = "You pressed Cancel!";  
}
```

Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box

returns null.

Syntax

```
window.prompt("sometext","defaultText");
```

The `window.prompt()` method can be written without the `window` prefix.

Example

```
var person = prompt("Please enter your name", "Harry Potter");  
if (person == null || person == "") {  
    txt = "User cancelled the prompt.";  
} else {  
    txt = "Hello " + person + "! How are you today?";  
}
```

JavaScript Objects

A JavaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create classes to get the object. But, we directly create objects.

Creating Objects in JavaScript

There are 3 ways to create objects.

1. By object literal

2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

1) JavaScript Object by object literal

The syntax of creating object using object literal is given below:

1. `object={property1:value1,property2:value2.....propertyN:valueN}`

As you can see, property and value is separated by : (colon).

Let's see the simple example of creating objects in JavaScript.

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

Output of the above example

```
102 Shyam Kumar 40000
```

2) By creating instance of Object

The syntax of creating object directly is given below:

1. `var objectname=new Object();`

Here, new keyword is used to create objects.

Let's see the example of creating objects directly.

```
<script>
```

```
var emp=new Object();  
emp.id=101;  
emp.name="Ravi Malik";  
emp.salary=50000;  
document.write(emp.id+" "+emp.name+" "+emp.salary);  
</script>
```

Output of the above example

```
101 Ravi 50000
```

3) By using an Object constructor

Here, you need to create a function with arguments. Each argument value can be assigned in the current object by using this keyword.

This keyword refers to the current object.

The example of creating object by object constructor is given below.

```
<script>  
function emp(id,name,salary) {  
this.id=id;  
this.name=name;  
this.salary=salary;  
}  
e=new emp(103,"Vimal Jaiswal",30000);  
  
document.write(e.id+" "+e.name+" "+e.salary);  
</script>
```

Output of the above example

```
103 Vimal Jaiswal 30000
```

Defining method in JavaScript Object

We can define methods in JavaScript objects. But before defining a method, we need to add property in the function with the same name as method.

The example of defining method in object is given below.

```
<script>
function emp(id,name,salary) {
this.id=id;
this.name=name;
this.salary=salary;
this.changeSalary=changeSalary;
function changeSalary(otherSalary) {
this.salary=otherSalary;
}
}
e=new emp(103,"Sonoo Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
e.changeSalary(45000);
document.write("<br>" +e.id+" "+e.name+" "+e.salary);
</script>
```

Output of the above example

103 Sonoo Jaiswal 30000

103 Sonoo Jaiswal 45000

JavaScript Object Methods

The various methods of Object are as follows:

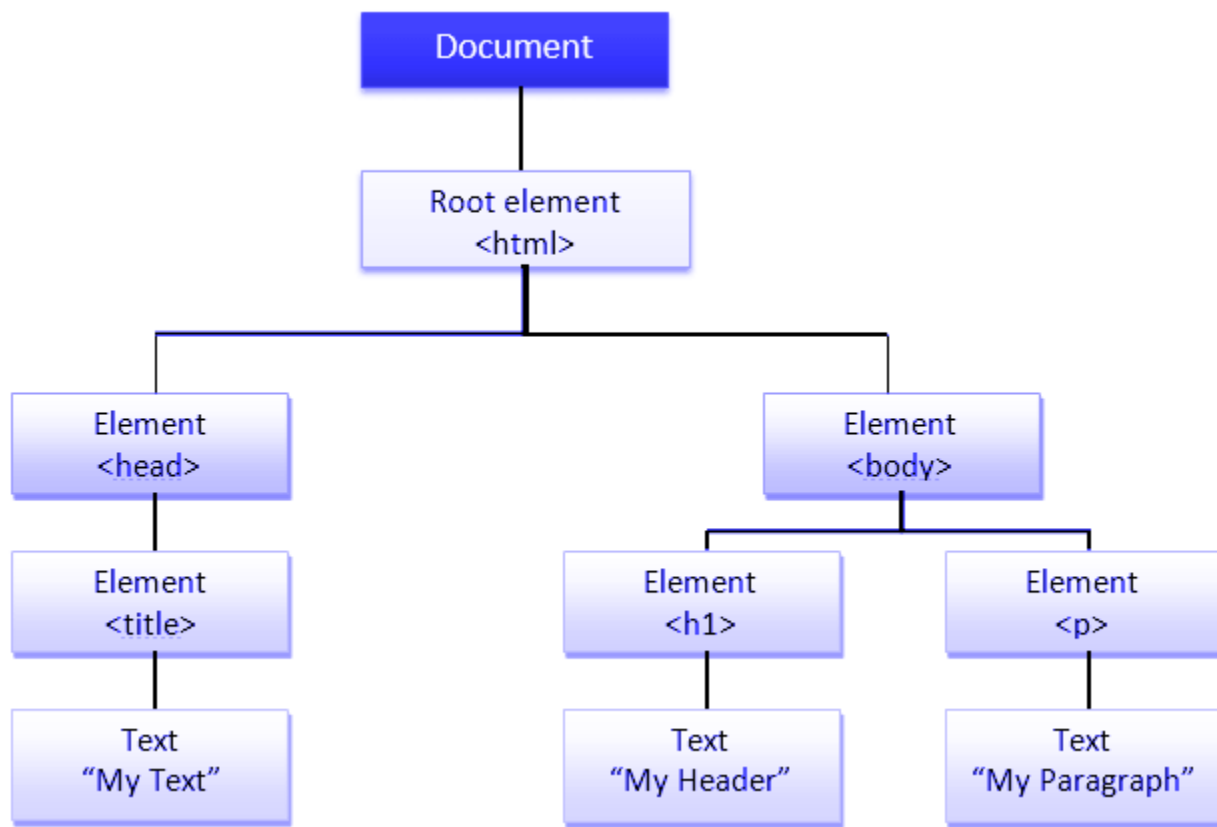
S.No	Methods	Description
------	---------	-------------

1	Object.assign()	This method is used to copy enumerable and own properties from a source object to a target object
2	Object.create()	This method is used to create a new object with the specified prototype object and properties.
3	Object.defineProperty()	This method is used to describe some behavioral attributes of the property.
4	Object.defineProperties()	This method is used to create or configure multiple object properties.
5	Object.entries()	This method returns an array with arrays of the key, value pairs.
6	Object.freeze()	This method prevents existing properties from being removed.
7	Object.getOwnPropertyDescriptor()	This method returns a property descriptor for the specified property of the specified object.
8	Object.getOwnPropertyDescriptors()	This method returns all its own property descriptors of a given object.
9	Object.getOwnPropertyNames()	This method returns an array of all properties (enumerable or not) found.
10	Object.getOwnPropertySymbols()	This method returns an array of all its own symbol key properties.
11	Object.getPrototypeOf()	This method returns the prototype of the specified object.
12	Object.is()	This method determines whether two values are the same value.
13	Object.isExtensible()	This method determines if an object is extensible
14	Object.isFrozen()	This method determines if an object was frozen.

15	<code>Object.isSealed()</code>	This method determines if an object is sealed.
16	<code>Object.keys()</code>	This method returns an array of a given object's own property names.
17	<code>Object.preventExtensions()</code>	This method is used to prevent any extensions of an object.
18	<code>Object.seal()</code>	This method prevents new properties from being added and marks all existing properties as non-configurable.
19	<code>Object.setPrototypeOf()</code>	This method sets the prototype of a specified object to another object.
20	<code>Object.values()</code>	This method returns an array of values.

What is DOM in JavaScript?

JavaScript can access all the elements in a webpage making use of Document Object Model (DOM). In fact, the web browser creates a DOM of the webpage when the page is loaded. The DOM model is created as a tree of objects like this:



How to use DOM and Events

Using DOM, JavaScript can perform multiple tasks. It can create new elements and attributes, change the existing elements and attributes and even remove existing elements and attributes. JavaScript can also react to existing events and create new events in the page.

getElementById, innerHTML Example

1. `getElementById`: To access elements and attributes whose id is set.
2. `innerHTML`: To access the content of an element.

```

<html>
<head>

```

```

        <title>DOM!!!</title>
</head>
<body>
  <h1 id="one">Welcome</h1>
  <p>This is the welcome message.</p>
  <h2>Technology</h2>
  <p>This is the technology section.</p>
  <script type="text/javascript">
    var text = document.getElementById("one").innerHTML;
    alert("The first heading is " + text);
  </script>
</body>
</html>

```

Welcome

This is the welcome message.

Technology

This is the technology section.

getElementsByTagName Example

getElementsByTagName: To access elements and attributes using tag name. This method will return an array of all the items with the same tag name.

```

<html>
<head>
  <title>DOM!!!</title>
</head>
<body>
  <h1>Welcome</h1>
  <p>This is the welcome message.</p>
  <h2>Technology</h2>
  <p id="second">This is the technology section.</p>
  <script type="text/javascript">
    var paragraphs = document.getElementsByTagName("p");
    alert("Content in the second paragraph is " +

```

```
paragraphs[1].innerHTML);
    document.getElementById("second").innerHTML = "The original
message is changed.";
</script>
</body>
</html>
```

Welcome

This is the welcome message.

Technology

The original message is changed.

Event handler Example

1. createElement: To create new element
2. removeChild: Remove an element
3. You can add an event handler to a particular element like this:

```
document.getElementById(id).onclick=function()
{
    lines of code to be executed
}
```

OR

```
document.getElementById(id).addEventListener("click", functionname)
<html>
<head>
    <title>DOM!!!</title>
</head>
<body>
    <input type="button" id="btnClick" value="Click Me!!" />
    <script type="text/javascript">
        document.getElementById("btnClick").addEventListener("click",
clicked);
        function clicked()
```

```
    {
        alert("You clicked me!!!");
    }
</script>
</body>
</html>
```

Functions or Methods?

you can also call these functions methods, since they are methods of the global object where they run. In a web browser, the global object is the browser window. Then `isNaN()` is actually a window method: `window.isNaN()`.

JavaScript `decodeURI()` Function

Example

Decode a URI after encoding it:

```
var uri = "my test.asp?name=ståle&car=saab";
var enc = encodeURIComponent(uri);
var dec = decodeURI(enc);
var res = enc + "<br>" + dec;
```

Encoded URI: `my%20test.asp?name=st%C3%A5le&car=saab`

Decoded URI: `my test.asp?name=ståle&car=saab`

Definition and Usage

The `decodeURI()` function is used to decode a URI.

Tip: Use the `encodeURIComponent()` function to encode a URI.

Syntax

`decodeURI(uri)`

Parameter Values

Parameter	Description
<code>uri</code>	Required. The URI to be decoded

Technical Details

Return Value:	A String, representing the decoded URI
---------------	--

JavaScript `decodeURIComponent()` Function

Example

Decode a URI after encoding it:

```
var uri = "https://Google.com/my test.asp?name=ståle&car=saab";  
var uri_enc = encodeURIComponent(uri);  
var uri_dec = decodeURIComponent(uri_enc);  
var res = uri_enc + "<br>" + uri_dec;
```

Encoded URI:

```
https%3A%2F%2Fw3Google.com%2Fmy%20test.asp%3Fname%3Dst%C3%A5le%26car  
%3Dsaab
```

Decoded URI: <https://Google.com/my test.asp?name=ståle&car=saab>

Definition and Usage

The `decodeURIComponent()` function decodes a URI component.

Tip: Use the `encodeURIComponent()` function to encode a URI component.

Syntax

`decodeURIComponent(uri)`

Parameter Values

Parameter	Description
<i>uri</i>	Required. The URI to be decoded

Technical Details

Return Value:	A String, representing the decoded URI
----------------------	--

JavaScript encodeURIComponent() Function

Example

Encode a URI:


```
var uri = "my test.asp?name=ståle&car=saab";  
var res = encodeURIComponent(uri);
```

my%20test.asp?name=st%C3%A5le&car=saab

Definition and Usage

The `encodeURIComponent()` function is used to encode a URI.

This function encodes special characters, except: `, / ? : @ & = + $ #` (Use `encodeURIComponent()` to encode these characters).

Tip: Use the `decodeURI()` function to decode an encoded URI.

Syntax

`encodeURIComponent(uri)`

Parameter Values

Parameter	Description
<i>uri</i>	Required. The URI to be encoded

Technical Details

Return Value:	A String, representing the encoded URI
---------------	--

JavaScript encodeURIComponent() Function

Example

Encode a URI:

```
var uri = "https://Google.com/my test.asp?name=ståle&car=saab";  
  
var res = encodeURIComponent(uri);
```

Definition and Usage

The `encodeURIComponent()` function encodes a URI component.

This function encodes special characters. In addition, it encodes the following

characters: `, / ? : @ & = + $ #`

Tip: Use the `decodeURIComponent()` function to decode an encoded URI component.

Syntax

`encodeURIComponent(uri)`

Parameter Values

Parameter	Description
<i>uri</i>	Required. The URI to be encoded

Technical Details

Return Value:	A String, representing the encoded URI
---------------	--

JavaScript eval() Function

Example

Evaluate/Execute JavaScript code/expressions:

```
var x = 10;  
var y = 20;  
var a = eval("x * y") + "<br>";  
var b = eval("2 + 2") + "<br>";  
var c = eval("x + 17") + "<br>";  
var res = a + b + c;
```

200

4

27

Definition and Usage

The eval() function evaluates or executes an argument.

If the argument is an expression, eval() evaluates the expression. If the argument is one or more JavaScript statements, eval() executes the statements.

Syntax

eval(*string*)

Parameter Values

Parameter	Description
<i>string</i>	A JavaScript expression, variable, statement, or sequence of statements

JavaScript parseInt() Function

Example

Parse different strings:

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button to parse different strings.</p>
<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var a = parseInt("10") + "<br>";
  var b = parseInt("10.00") + "<br>";
  var c = parseInt("10.33") + "<br>";
  var d = parseInt("34 45 66") + "<br>";
  var e = parseInt(" 60 ") + "<br>";
  var f = parseInt("40 years") + "<br>";
  var g = parseInt("He was 40") + "<br>";

  var h = parseInt("10", 10)+ "<br>";
  var i = parseInt("010")+ "<br>";
  var j = parseInt("10", 8)+ "<br>";
  var k = parseInt("0x10")+ "<br>";
  var l = parseInt("10", 16)+ "<br>";

  var n = a + b + c + d + e + f + g + "<br>" + h + i + j + k + l;
  document.getElementById("demo").innerHTML = n;
}
```

```
}  
</script>  
  
</body>  
</html>
```

Click the button to parse different strings.

Try it

```
10  
10  
10  
34  
60  
40  
NaN  
10  
10  
8  
16  
16
```

Definition and Usage

The `parseInt()` function parses a string and returns an integer.

The `radix` parameter is used to specify which numeral system to be used, for example, a `radix` of 16 (hexadecimal) indicates that the number in the string should be parsed from a hexadecimal number to a decimal number.

If the `radix` parameter is omitted, JavaScript assumes the following:

- If the string begins with "0x", the radix is 16 (hexadecimal)
- If the string begins with "0", the radix is 8 (octal). This feature is deprecated

- If the string begins with any other value, the radix is 10 (decimal)

Note: Only the first number in the string is returned!

Note: Leading and trailing spaces are allowed.

Note: If the first character cannot be converted to a number, `parseInt()` returns NaN.

Note: Older browsers will result in `parseInt("010")` as 8, because older versions of ECMAScript, (older than ECMAScript 5, uses the octal radix (8) as default when the string begins with "0". As of ECMAScript 5, the default is the decimal radix (10).

Syntax

`parseInt(string, radix)`

Parameter Values

Parameter	Description
<i>string</i>	Required. The string to be parsed
<i>radix</i>	Optional. A number (from 2 to 36) that represents the numeral system to be used

JavaScript Form Validation

It is important to validate the form submitted by the user because it can have

inappropriate values. So, validation is must to authenticate users.

JavaScript provides a facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation.

Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

JavaScript Form Validation Example

In this example, we are going to validate the name and password. The name can't be empty and the password can't be less than 6 characters long.

Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
    alert("Name can't be blank");
    return false;
}else if(password.length<6){
    alert("Password must be at least 6 characters long.");
    return false;
}
}
</script>
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return
validateform()" >
Name: <input type="text" name="name"><br/>
```

```
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
```

JavaScript Retype Password Validation

```
<script type="text/javascript">
function matchpass() {
var firstpassword=document.f1.password.value;
var secondpassword=document.f1.password2.value;

if(firstpassword==secondpassword) {
return true;
}
else{
alert("password must be same!");
return false;
}
}
}
</script>

<form name="f1" action="register.jsp" onsubmit="return
matchpass()">
Password:<input type="password" name="password" /><br/>
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form>
```

JavaScript Number Validation

Let's validate the text field for numeric value only. Here, we are using isNaN() function.

```
<script>
function validate() {
var num=document.myform.num.value;
if (isNaN(num)) {
    document.getElementById("numloc").innerHTML="Enter Numeric value
only";
    return false;
}
```



```

}else{
    return true;
}
}
</script>
<form name="myform" onsubmit="return validate()" >
Number: <input type="text" name="num"><span
id="numloc"></span><br/>
<input type="submit" value="submit">
</form>

```

JavaScript validation with image

Let's see an interactive JavaScript form validation example that displays correct and incorrect images if input is correct or incorrect.

```

<script>
function validate(){
var name=document.f1.name.value;
var password=document.f1.password.value;
var status=false;

if(name.length<1){
document.getElementById("nameloc").innerHTML=
" <img src='unchecked.gif' /> Please enter your name";
status=false;
}else{
document.getElementById("nameloc").innerHTML=" <img
src='checked.gif' />";
status=true;
}
if(password.length<6){
document.getElementById("passwordloc").innerHTML=
" <img src='unchecked.gif' /> Password must be at least 6 char
long";

```

```

status=false;
}else{
document.getElementById("passwordloc").innerHTML=" <img
src='checked.gif' />";
}
return status;
}
</script>

<form name="f1" action="#" onsubmit="return validate()">
<table>
<tr><td>Enter Name:</td><td><input type="text" name="name"/>
<span id="nameloc"></span></td></tr>
<tr><td>Enter Password:</td><td><input type="password"
name="password" />
<span id="passwordloc"></span></td></tr>
<tr><td colspan="2"><input type="submit"
value="register" /></td></tr>
</table>
</form>

```

JavaScript email validation

We can validate the email by the help of JavaScript.

There are many criteria that need to be follow to validate the email id such as:

- email id must contain the @ and . character
- There must be at least one character before and after the @.
- There must be at least two characters after . (dot).

Let's see the simple example to validate the email field.

```

<script>
function validateemail ()
{
var x=document.myform.email.value;
var atposition=x.indexOf("@");

```

```

var dotposition=x.lastIndexOf(".");
if (atposition<1 || dotposition<atposition+2 ||
dotposition+2>=x.length){
    alert("Please enter a valid e-mail address \n
atpostion:"+atposition+"\n dotposition:"+dotposition);
    return false;
}
}
</script>
<body>
<form name="myform" method="post" action="#" onsubmit="return
validateemail();">
Email: <input type="text" name="email"><br/>

<input type="submit" value="register">
</form>

```

JavaScript RegExp Reference

RegExp Object

A regular expression is an object that describes a pattern of characters.

Regular expressions are used to perform pattern-matching and "search-and-replace" functions on text.

Syntax

/pattern/modifiers;

Example

```

<!DOCTYPE html>
<html>
<body>

```

```
<h2>JavaScript Regular Expressions</h2>
```

```
<p>Click the button to do a case-insensitive search for "AIT" in a string.</p>
```

```
<button onclick="myFunction()">Try it</button>
```

```
<p id="demo"></p>
```

```
<script>
```

```
function myFunction() {  
    var str = "Visit AIT";  
    var patt = /AIT/i;  
    var result = str.match(patt);  
    document.getElementById("demo").innerHTML = result;  
}
```

```
</script>
```

```
</body>
```

```
</html>
```

JavaScript Regular Expressions

Click the button to do a case-insensitive search for "AIT" in a string.

Try it

AIT

Example explained:

- /AIT/i is a regular expression.
- AIT is a pattern (to be used in a search).
- i is a modifier (modifies the search to be case-insensitive).

Modifiers

Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
g	Perform a global match (find all matches rather than stopping after the first match)
i	Perform case-insensitive matching
m	Perform multiline matching

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
(x y)	Find any of the alternatives specified

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word, beginning like this: \bHI, end like this: HI\b
\B	Find a match, but not at the beginning/end of a word
\0	Find a NULL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\udddd	Find the Unicode character specified by a hexadecimal number dddd

Quantifiers

Quantifier	Description
n+	Matches any string that contains at least one <i>n</i>
n*	Matches any string that contains zero or more occurrences of <i>n</i>
n?	Matches any string that contains zero or one occurrences of <i>n</i>
n{X}	Matches any string that contains a sequence of <i>X n</i> 's
n{X,Y}	Matches any string that contains a sequence of <i>X</i> to <i>Y n</i> 's
n{X,}	Matches any string that contains a sequence of at least <i>X n</i> 's
n\$	Matches any string with <i>n</i> at the end of it
^n	Matches any string with <i>n</i> at the beginning of it
?=n	Matches any string that is followed by a specific string <i>n</i>
?!n	Matches any string that is not followed by a specific string <i>n</i>

RegExp Object Properties

Property	Description
constructor	Returns the function that created the RegExp object's prototype
global	Checks whether the "g" modifier is set
ignoreCase	Checks whether the "i" modifier is set
lastIndex	Specifies the index at which to start the next match
multiline	Checks whether the "m" modifier is set
source	Returns the text of the RegExp pattern

RegExp Object Methods

Method	Description
compile()	Deprecated in version 1.5. Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false
toString()	Returns the string value of the regular expression

```
<!DOCTYPE html>
<html>
<body>
```

```
<p>Click the button to do a global search for the numbers 1 to 4 in
a string.</p>
```

```
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var str = "123456789";
  var patt1 = /[1-4]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Click the button to do a global search for the numbers 1 to 4 in a string.

Try it

1,2,3,4

JavaScript Events

The change in the state of an object is known as an Event. In html, there are various events which represent that some activity is performed by the user or by the browser.

When javascript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called Event Handling. Thus, js handles the HTML events via Event Handlers.

For example, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Some of the HTML events and their event handlers are:

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Keyboard events:

Event Performed	Event Handler	Description
-----------------	---------------	-------------

Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key
----------------------------	--------------------------------	---

Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Window/Document events

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

Let's discuss some examples over events and their handlers.

Click Event

```
<html>
<head> Javascript Events </head>
<body>
<script language="Javascript" type="text/Javascript">
```

```

    <!--
    function clickevent()
    {
        document.write("This is JavaTpoint");
    }
    //-->
</script>
<form>
<input type="button" onclick="clickevent()" value="Who's this?"/>
</form>
</body>
</html>

```

MouseOver Event

```

<html>
<head>
<h1> Javascript Events </h1>
</head>
<body>
<script language="Javascript" type="text/Javascript">
    <!--
    function mouseoverevent()
    {
        alert("This is JavaTpoint");
    }
    //-->
</script>
<p onmouseover="mouseoverevent()"> Keep cursor over me</p>
</body>
</html>

```

Focus Event

```

<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onfocus="focusevent()"/>
<script>
<!--
    function focusevent()
    {
        document.getElementById("input1").style.background=" aqua";
    }

```

```
//-->
</script>
</body>
</html>
```

Keydown Event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onkeydown="keydownevent()" />
<script>
<!--
    function keydownevent()
    {
        document.getElementById("input1");
        alert("Pressed a key");
    }
-->
</script>
</body>
</html>
```

Load event

```
<html>
<head>Javascript Events</head>
</br>
<body onload="window.alert('Page successfully loaded');">
<script>
<!--
document.write("The page is loaded successfully");
-->
</script>
</body>
</html>
```

JavaScript | Callbacks

Callbacks are a great way to handle something after something else has been completed.

By something here we mean a function execution. If we want to execute a function right after the return of some other function, then callbacks can be used.

JavaScript functions have the type of Objects. So, much like any other objects (String, Arrays etc.), They can be passed as an argument to any other function while calling.

JavaScript code to show the working of callback:

Code #1:

```
<script>

    // add() function is called with arguments a, b
    // and callback, callback will be executed just
    // after ending of add() function
    function add(a, b , callback){
        document.write(`The sum of ${a} and ${b} is ${a+b}.`
+"<br>");
        callback();
    }

    // disp() function is called just
    // after the ending of add() function
    function disp(){
        document.write('This must be printed after addition');
    }

    // Calling add() function
    add(5,6,disp);

</script>
```

Output:

The sum of 5 and 6 is 11.

This must be printed after addition

Explanation:

Here are the two functions – add(a, b, callback) and disp(). Here add() is called with the disp() function i.e. passed in as the third argument to the add function along with two numbers.

As a result, the add() is invoked with 1, 2 and the disp() which is the callback. The add() prints the addition of the two numbers and as soon as that is done, the callback function is fired!

Code #2:

An alternate way to implement above code is shown below with anonymous functions being passed.

```
<script>

    // add() function is called with arguments a, b
    // and callback, callback will be executed just
    // after ending of add() function
    function add(a, b , callback){
        document.write(`The sum of ${a} and ${b} is ${a+b}.`
+"<br>");
        callback();
    }

    // add() function is called with arguments given below
    add(5,6,function disp(){
        document.write('This must be printed after addition. ');
    });

</script>
```

Output:

The sum of 5 and 6 is 11.

This must be printed after addition.

Callbacks are primarily used while handling asynchronous operations like – making an API request to Google Maps, fetching/writing some data from/into a file, registering event listeners and related stuff. All the operations mentioned use callbacks. This way once the data/error from the asynchronous operation is returned, the callbacks are used to do something with that inside our code.

JavaScript Math Object

The JavaScript Math object allows you to perform mathematical tasks on numbers.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Math.PI</h2>
<p>Math.PI returns the ratio of a circle's circumference to its
diameter:</p>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = Math.PI;
</script>
</body>
</html>
```

JavaScript Math.PI

Math.PI returns the ratio of a circle's circumference to its diameter:

3.141592653589793

The Math Object

Unlike other objects, the Math object has no constructor.

The Math object is static.

All methods and properties can be used without creating a Math object first.

Math Properties (Constants)

The syntax for any Math property is : *Math.property*.

JavaScript provides 8 mathematical constants that can be accessed as Math properties:

Example

```
Math.E          // returns Euler's number
Math.PI         // returns PI
Math.SQRT2      // returns the square root of 2
Math.SQRT1_2    // returns the square root of 1/2
Math.LN2        // returns the natural logarithm of 2
Math.LN10       // returns the natural logarithm of 10
Math.LOG2E      // returns base 2 logarithm of E
Math.LOG10E     // returns base 10 logarithm of E
```

Math Methods

The syntax for Math any methods is : *Math.method.(number)*

Number to Integer

There are 4 common methods to round a number to an integer:

<code>Math.round(x)</code>	Returns x rounded to its nearest integer
<code>Math.ceil(x)</code>	Returns x rounded up to its nearest integer

<code>Math.floor(x)</code>	Returns x rounded down to its nearest integer
<code>Math.trunc(x)</code>	Returns the integer part of x (new in ES6)

`Math.round()`

`Math.round(x)` returns the nearest integer:

Example

```
Math.round(4.9);    // returns 5
Math.round(4.7);    // returns 5
Math.round(4.4);    // returns 4
Math.round(4.2);    // returns 4
Math.round(-4.2);   // returns -4
```

`Math.ceil()`

`Math.ceil(x)` returns the value of x rounded up to its nearest integer:

Example

```
Math.ceil(4.9);     // returns 5
Math.ceil(4.7);     // returns 5
Math.ceil(4.4);     // returns 5
Math.ceil(4.2);     // returns 5
Math.ceil(-4.2);    // returns -4
```

`Math.floor()`

`Math.floor(x)` returns the value of x rounded down to its nearest integer:

Example

```
Math.floor(4.9);    // returns 4
Math.floor(4.7);    // returns 4
Math.floor(4.4);    // returns 4
Math.floor(4.2);    // returns 4
Math.floor(-4.2);   // returns -5
```

Math.trunc()

Math.trunc(x) returns the integer part of x:

Math.sign()

Math.sign(x) returns if x is negative, null or positive:

Math.pow()

Math.pow(x, y) returns the value of x to the power of y:

Math.sqrt()

Math.sqrt(x) returns the square root of x:

Math.abs()

Math.abs(x) returns the absolute (positive) value of x:

Math.sin()

Math.sin(x) returns the sine (a value between -1 and 1) of the angle x (given in radians).

If you want to use degrees instead of radians, you have to convert degrees to radians:

Angle in radians = Angle in degrees x PI / 180.

Math.cos()

Math.cos(x) returns the cosine (a value between -1 and 1) of the angle x (given in radians).

If you want to use degrees instead of radians, you have to convert degrees to radians:

Angle in radians = Angle in degrees x PI / 180.

Math.min() and Math.max()

Math.min() and Math.max() can be used to find the lowest or highest value in a list of arguments:

Math.random()

Math.random() returns a random number between 0 (inclusive), and 1 (exclusive):

The Math.log() Method

Math.log(x) returns the natural logarithm of x:

The natural logarithm returns the time needed to reach a certain level of growth.

Math.E and Math.log() are twins.

How many times must we multiply Math.E to get 10?

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
acosh(x)	Returns the hyperbolic arccosine of x
asin(x)	Returns the arcsine of x, in radians
asinh(x)	Returns the hyperbolic arcsine of x
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y, x)	Returns the arctangent of the quotient of its arguments
atanh(x)	Returns the hyperbolic arctangent of x
cbrt(x)	Returns the cubic root of x
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
cosh(x)	Returns the hyperbolic cosine of x
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x, y, z, ..., n)	Returns the number with the highest value
min(x, y, z, ..., n)	Returns the number with the lowest value

pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sign(x)	Returns if x is negative, null or positive (-1, 0, 1)
sin(x)	Returns the sine of x (x is in radians)
sinh(x)	Returns the hyperbolic sine of x
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle
tanh(x)	Returns the hyperbolic tangent of a number
trunc(x)	Returns the integer part of a number (x)

JavaScript Get Date Methods

These methods can be used for getting information from a date object:

Method	Description
getFullYear()	Get the year as a four digit number (yyyy)
getMonth()	Get the month as a number (0-11)
getDate()	Get the day as a number (1-31)
getHours()	Get the hour (0-23)
getMinutes()	Get the minute (0-59)
getSeconds()	Get the second (0-59)

<code>getMilliseconds()</code>	Get the millisecond (0-999)
<code>getTime()</code>	Get the time (milliseconds since January 1, 1970)
<code>getDay()</code>	Get the weekday as a number (0-6)
<code>Date.now()</code>	Get the time. ECMAScript 5.

The `getTime()` Method

The `getTime()` method returns the number of milliseconds since January 1, 1970:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript getTime()</h2>
<p>The internal clock in JavaScript counts from midnight January 1,
1970.</p>
<p>The getTime() function returns the number of milliseconds since
then:</p>

<p id="demo"></p>

<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.getTime();
</script>

</body>
</html>
```

JavaScript `getTime()`

The internal clock in JavaScript counts from midnight January 1, 1970.

The `getTime()` function returns the number of milliseconds since then:

1616490648877

UTC Date Methods

UTC date methods are used for working with UTC dates (Universal Time Zone dates):

Method	Description
<code>getUTCDate()</code>	Same as <code>getDate()</code> , but returns the UTC date
<code>getUTCDay()</code>	Same as <code>getDay()</code> , but returns the UTC day
<code>getUTCFullYear()</code>	Same as <code>getFullYear()</code> , but returns the UTC year
<code>getUTCHours()</code>	Same as <code>getHours()</code> , but returns the UTC hour
<code>getUTCMilliseconds()</code>	Same as <code>getMilliseconds()</code> , but returns the UTC milliseconds
<code>getUTCMinutes()</code>	Same as <code>getMinutes()</code> , but returns the UTC minutes
<code>getUTCMonth()</code>	Same as <code>getMonth()</code> , but returns the UTC month
<code>getUTCSeconds()</code>	Same as <code>getSeconds()</code> , but returns the UTC seconds

JavaScript Booleans

A JavaScript Boolean represents one of two values: true or false.

Boolean Values

Very often, in programming, you will need a data type that can only have one of two values, like

- YES / NO

- ON / OFF
- TRUE / FALSE

For this, JavaScript has a Boolean data type. It can only take the values true or false.

The Boolean() Function

You can use the Boolean() function to find out if an expression (or a variable) is true:

```
<!DOCTYPE html>
<html>
<body>
<p>Display the value of Boolean(10 > 9):</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = Boolean(10 > 9);
}
</script>
</body>
</html>
```

Display the value of Boolean(10 > 9):

Try it

true

Comparisons and Conditions

The chapter JS Comparisons gives a full overview of comparison operators.

Here are some examples:

Operator	Description	Example
==	equal to	if (day == "Monday")
>	greater than	if (salary > 9000)
<	less than	if (age < 18)

Booleans Can be Objects

Normally JavaScript booleans are primitive values created from literals:

```
var x = false;
```

But booleans can also be defined as objects with the keyword `new`:

```
var y = new Boolean(false);
```

```
<!DOCTYPE html>
<html>
<body>
<p>Never create booleans as objects.</p>
<p>Booleans and objects cannot be safely compared.</p>
<p id="demo"></p>
<script>
var x = false;           // x is a boolean
var y = new Boolean(false); // y is an object
document.getElementById("demo").innerHTML = typeof x + "<br>" +
typeof y;
</script>
</body>
</html>
```

Never create booleans as objects.

Booleans and objects cannot be safely compared.

boolean

object

What is JSON

JSON is an open standard for exchanging data on the web. It supports data structures like objects and arrays. So it is easy to write and read data from JSON.

What is JSON

- **JSON stands for JavaScript Object Notation.**
- **JSON is an open standard data-interchange format.**
- **JSON is lightweight and self describing.**
- **JSON originated from JavaScript.**
- **JSON is easy to read and write.**
- **JSON is language independent.**
- **JSON supports data structures such as arrays and objects.**

Features of JSON

- 1. Simplicity**
- 2. Openness**
- 3. Self Describing**
- 4. Internationalization**
- 5. Extensibility**
- 6. Interoperability**

JSON Example

JSON examples can be created by object and array. Each object can have different data such as text, number, boolean etc. Let's see different JSON examples using objects and

arrays.

JSON Object Example

A JSON object contains data in the form of a key/value pair. The keys are strings and the values are the JSON types. Keys and values are separated by colon. Each entry (key/value pair) is separated by comma.

The { (curly brace) represents the JSON object.

```
{
  "employee": {
    "name":      "sonoo",
    "salary":    56000,
    "married":   true
  }
}
```

JSON Array example

The [(square bracket) represents the JSON array. A JSON array can have values and objects.

Let's see the example of a JSON array having values.

1. ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

Let's see the example of a JSON array having objects.

```
[
  {"name": "Ram", "email": "Ram@gmail.com"},
  {"name": "Bob", "email": "bob32@gmail.com"}
]
```

JSON Example 1

```

{"employees": [
  {"name": "Shyam", "email": "shyamjaiswal@gmail.com"},
  {"name": "Bob", "email": "bob32@gmail.com"},
  {"name": "Jai", "email": "jai87@gmail.com"}
]}

```

The XML representation of the above JSON example is given below.

```

<employees>
  <employee>
    <name>Shyam</name>
    <email>shyamjaiswal@gmail.com</email>
  </employee>
  <employee>
    <name>Bob</name>
    <email>bob32@gmail.com</email>
  </employee>
  <employee>
    <name>Jai</name>
    <email>jai87@gmail.com</email>
  </employee>
</employees>

```

JSON Example 2

```

{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Save", "onclick": "SaveDoc()"}
    ]
  }
}}

```

The XML representation of the above JSON example is given below.

```

<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
  </popup>
</menu>

```

```
    <menuitem value="Save" onclick="SaveDoc()" />
  </popup>
</menu>
```

JSON Object

JSON object holds key/value pair. Each key is represented as a string in JSON and value can be of any type. The keys and values are separated by colon. Each key/value pair is separated by comma.

The curly brace { represents a JSON object.

Let's see an example of a JSON object.

```
{
  "employee": {
    "name":      "sonoo",
    "salary":    56000,
    "married":   true
  }
}
```

In the above example, an employee is an object in which "name", "salary" and "married" are the key. In this example, there are string, number and boolean values for the keys.

JSON Object with Strings

The string value must be enclosed within a double quote.

```
{
  "name":      "sonoo",
  "email":     "sonoojaiswal1987@gmail.com"
}
```

JSON Object with Numbers

JSON supports numbers in double precision floating-point format. The number can be digits (0-9), fractions (.33, .532 etc) and exponents (e, e+, e-,E, E+, E-).

```
{  
  "integer": 34,  
  "fraction": .2145,  
  "exponent": 6.61789e+0  
}
```

JSON Object with Booleans

JSON also supports boolean values *true* or *false*.

```
{  
  "first": true,  
  "second": false  
}
```

JSON Nested Object Example

A JSON object can have another object also. Let's see a simple example of a JSON object having another object.

```
{  
  "firstName": "Sonoo",  
  "lastName": "Jaiswal",  
  "age": 27,  
  "address" : {  
    "streetAddress": "Plot-6, Mohan Nagar",  
    "city": "Ghaziabad",  
    "state": "UP",  
    "postalCode": "201007"  
  }  
}
```

```
}
```

JSON Array

The JSON array represents an ordered list of values. JSON array can store multiple values. It can store string, number, boolean or object in JSON array.

In JSON arrays, values must be separated by comma.

The [(square bracket) represents JSON array.

JSON Array of Strings

Let's see an example of JSON arrays storing string values.

```
["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]
```

JSON Array of Numbers

Let's see an example of JSON arrays storing number values.

```
[12, 34, 56, 43, 95]
```

JSON Array of Booleans

Let's see an example of JSON arrays storing boolean values.

```
[true, true, false, false, true]
```

JSON Array of Objects

Let's see a simple JSON array example having 4 objects.

```
{ "employees": [  
    { "name": "Ram", "email": "ram@gmail.com", "age": 23 },  
    { "name": "Shyam", "email": "shyam23@gmail.com", "age": 28 },  
    { "name": "John", "email": "john@gmail.com", "age": 33 },  
    { "name": "Bob", "email": "bob32@gmail.com", "age": 41 }  
] }
```

JSON Multidimensional Array

We can store arrays inside JSON arrays, it is known as an array of arrays or multidimensional arrays.

```
[  
  ["a", "b", "c" ],  
  ["m", "n", "o" ],  
  ["x", "y", "z" ]  
]
```

JSON Function Files

A common use of JSON is to read data from a web server, and display the data in a web page.

JSON Example

This example reads a menu from myTutorials.js, and displays the menu in a web page:

```
<!DOCTYPE html>  
<html>  
<body>  
<div id="id01"></div>  
<script>  
function myFunction(arr) {  
  var out = "";  
  var i;  
  for(i = 0; i<arr.length; i++) {  
    out += '<a href="' + arr[i].url + '>' +  
      arr[i].display + '</a><br>';  
  }  
  document.getElementById("id01").innerHTML = out;  
}  
</script>  
  
<script src="myTutorials.js"></script>
```



```
</body>
</html>
```

JSON Http Request

A common use of JSON is to read data from a web server, and display the data in a web page.

JSON Example

This example reads a menu from myTutorials.txt, and displays the menu in a web page:

JSON Example

```
<!DOCTYPE html>
<html>
<body>
<div id="id01"></div>
<script>
var xmlhttp = new XMLHttpRequest();
var url = "myTutorials.txt";
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myArr = JSON.parse(this.responseText);
        myFunction(myArr);
    }
};
xmlhttp.open("GET", url, true);
xmlhttp.send();
function myFunction(arr) {
    var out = "";
    var i;
    for(i = 0; i < arr.length; i++) {
        out += '<a href="' + arr[i].url + '>' +
            arr[i].display + '</a><br>';
    }
}
```

```
    document.getElementById("id01").innerHTML = out;
}
</script>
</body>
</html>
```

Server Side Programming with PHP

What is PHP

PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development. Therefore, it is used to develop web applications (an application that executes on the server and generates the dynamic page.).

PHP was created by **Rasmus Lerdorf in 1994** but appeared in the market in 1995. **PHP 7.4.0** is the latest version of PHP, which was released on **28 November**. Some important points that need to be noticed about PHP are as followed:

- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn the language

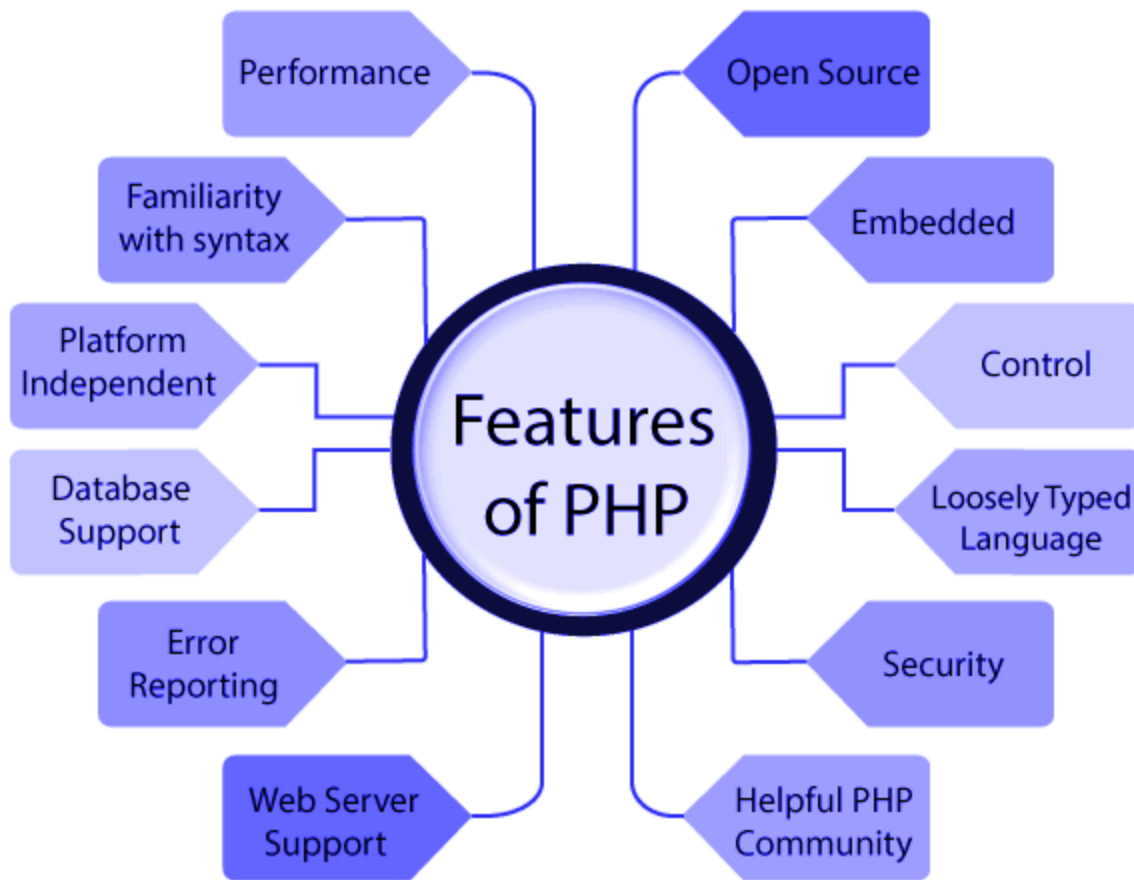
Why use PHP

PHP is a server-side scripting language, which is used to design dynamic web applications with MySQL database.

- It handles dynamic content, database as well as session tracking for the website.
- You can create sessions in PHP.
- It can access cookie variables and also set cookies.
- It helps to encrypt the data and apply validation.
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.
- Using PHP language, you can control the user to access some pages of your website.
- As PHP is easy to install and set up, this is the main reason why PHP is the best language to learn.
- PHP can handle the forms, such as - collect the data from users using forms, save it into the database, and return useful information to the user. **For example -** Registration form.

PHP Features

PHP is a very popular language because of its simplicity and open source. There are some important features of PHP given below:



Performance:

PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.

Open Source:

PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

Familiarity with the syntax:

PHP has easily understandable syntax. Programmers are comfortable coding with it.

Embedded:

PHP code can be easily embedded within HTML tags and script.

Platform Independent:

PHP is available for WINDOWS, MAC, LINUX & UNIX operating systems. A PHP application developed in one OS can be easily executed in other OS also.

Database Support:

PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.

Error Reporting -

PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.

Loosely Typed Language:

PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.

Web servers Support:

PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.

Security:

PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threats and malicious attacks.

Control:

Different programming languages require long scripts or codes, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

A Helpful PHP Community:

It has a large community of developers who regularly updates documentation, tutorials, online help, and FAQs. Learning PHP from the communities is one of the significant benefits.

PHP Echo

PHP echo is a language construct, not a function. Therefore, you don't need to use parentheses with it. But if you want to use more than one parameter, it is required to use parenthesis.

The syntax of PHP echo is given below:

1. `void echo (string $arg1 [, string $...])`

PHP echo statements can be used to print the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:

- echo is a statement, which is used to display the output.
- echo can be used with or without parentheses: `echo()`, and `echo`.
- echo does not return any value.
- We can pass multiple strings separated by a comma (,) in echo.
- echo is faster than the print statement.

PHP echo: printing string

File: echo1.php

```
<?php
echo "Hello by PHP echo";
?>
```

Output:

Hello by PHP echo

PHP echo: printing multi line string

File: echo2.php

<?php

```
echo "Hello by PHP echo  
this is multi line  
text printed by  
PHP echo statement  
";  
?>
```

Output:

Hello by PHP echo this is multi line text printed by PHP echo statement

PHP Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Basic PHP Syntax

A PHP script can be placed anywhere in the document.

A PHP script starts with <?php and ends with ?>:

```
<?php  
// PHP code goes here  
?>
```

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

My first PHP page

Hello World!

Note: PHP statements end with a semicolon (;).

PHP Case Sensitivity

In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.

In the example below, all three echo statements below are equal and legal:

```
<!DOCTYPE html>
<html>
<body>
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
</body>
</html>
```

Hello World!

Hello World!

Hello World!

Note: However; all variable names are case-sensitive!

PHP Comments

Comments in PHP

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

Comments can be used to:

- Let others understand your code
- Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

PHP supports several ways of commenting:

Example

Syntax for single-line comments:

```
<!DOCTYPE html>
<html>
<body>
<?php
// This is a single-line comment
# This is also a single-line comment
?>
</body>
</html>
```

Example

Syntax for multiple-line comments:

```
<!DOCTYPE html>
<html>
<body>
```

```
<?php
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
?>
</body>
</html>
```

Example

Using comments to leave out parts of the code:

```
<!DOCTYPE html>
<html>
<body>
<?php
// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>
</body>
</html>
```

PHP - Variable Types

The main way to store information in the middle of a PHP program is by using a variable.

Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign (\$).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.

- PHP does a good job of automatically converting types from one to another when necessary.
- PHP variables are Perl-like.

PHP has a total of eight data types which we use to construct our variables –

- Integers – are whole numbers, without a decimal point, like 4195.
- Doubles – are floating-point numbers, like 3.14159 or 49.1.
- Booleans – have only two possible values either true or false.
- NULL – is a special type that only has one value: NULL.
- Strings – are sequences of characters, like 'PHP supports string operations.'
- Arrays – are named and indexed collections of other values.
- Objects – are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- Resources – are special variables that hold references to resources external to PHP (such as database connections).

The first five are *simple types*, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

Integers

They are whole numbers, without a decimal point, like 4195. They are the simplest type .they correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so –

```
$int_var = 12345;
```

```
$another_int = -12345 + 12345;
```

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x.

For most common platforms, the largest integer is $(2^{31} - 1)$ (or 2,147,483,647), and the smallest (most negative) integer is $-(2^{31} - 1)$ (or -2,147,483,647).

Doubles

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code –

```
<?php
    $many = 2.2888800;
    $many_2 = 2.2111200;
    $few = $many + $many_2;
    print("$many + $many_2 = $few <br>");
?>
```

It produces the following browser output –

```
2.28888 + 2.21112 = 4.5
```

Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types –

- Local variables
- Function parameters
- Global variables
- Static variables

PHP - Local Variables

Local Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function –

```
<?php
    $x = 4;
    function assignx () {
        $x = 0;
        print "\$x inside function is $x. <br />";
    }
    assignx();
    print "\$x outside of function is $x. <br />";
?>
```

This will produce the following result –

\$x inside function is 0.

\$x outside of function is 4.

PHP - Function Parameters

But in short a function is a small unit of program which can take some input in the form of parameters and does some processing and may return some value.

Function Parameters

Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable would be –

```
<?php
    // multiply a value by 10 and return it to the caller
    function multiply ($value) {
        $value = $value * 10;
        return $value;
    }
    $retval = multiply (10);
    Print "Return value is $retval\n";
?>
```

This will produce the following result –

Return value is 100

PHP - Global Variables

Global Variables

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished, conveniently enough, by placing the keyword GLOBAL in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example –

```
<?php
    $somevar = 15;
    function addit() {
        GLOBAL $somevar;
        $somevar++;
        print "Somevar is $somevar";
    }
    addit();
?>
```

This will produce the following result –

Somevar is 16

PHP - Static Variables

Static Variables

The final type of variable scoping that I discuss is known as static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

You can declare a variable to be static simply by placing the keyword `STATIC` in front of the variable name.

```
<?php
    function keep_track() {
        STATIC $count = 0;
        $count++;
        print $count;
        print "<br />";
    }
    keep_track();
    keep_track();
    keep_track();
?>
```

This will produce the following result –

1

2
3

PHP | Loops

Like any other language, loop in PHP is used to execute a statement or a block of statements, multiple times until and unless a specific condition is met. This helps the user to save both time and effort of writing the same code multiple times.

PHP supports four types of looping techniques;

1. for loop
2. while loop
3. do-while loop
4. foreach loop

Let us now learn about each of the above mentioned loops in details:

1. **for loop**: This type of loop is used when the user knows in advance, how many times the block needs to execute. That is, the number of iterations is known beforehand. These types of loops are also known as entry-controlled loops. There are three main parameters to the code, namely the initialization, the test condition and the counter.

Syntax:

```
for (initialization expression; test condition; update expression) {  
    // code to be executed  
}
```


In for loop, a loop variable is used to control the loop. First initialize this loop variable to some value, then check whether this variable is less than or greater than counter value. If the statement is true, then the loop body is executed and the loop variable gets updated . Steps are repeated till exit condition comes.

- **Initialization Expression:** In this expression we have to initialize the loop counter to some value. for example: \$num = 1;
- **Test Expression:** In this expression we have to test the condition. If the condition evaluates to true then we will execute the body of the loop and go to update expression otherwise we will exit from the for loop. For example:
\$num <= 10;
- **Update Expression:** After executing loop body this expression increments/decrements the loop variable by some value. for example: \$num += 2;

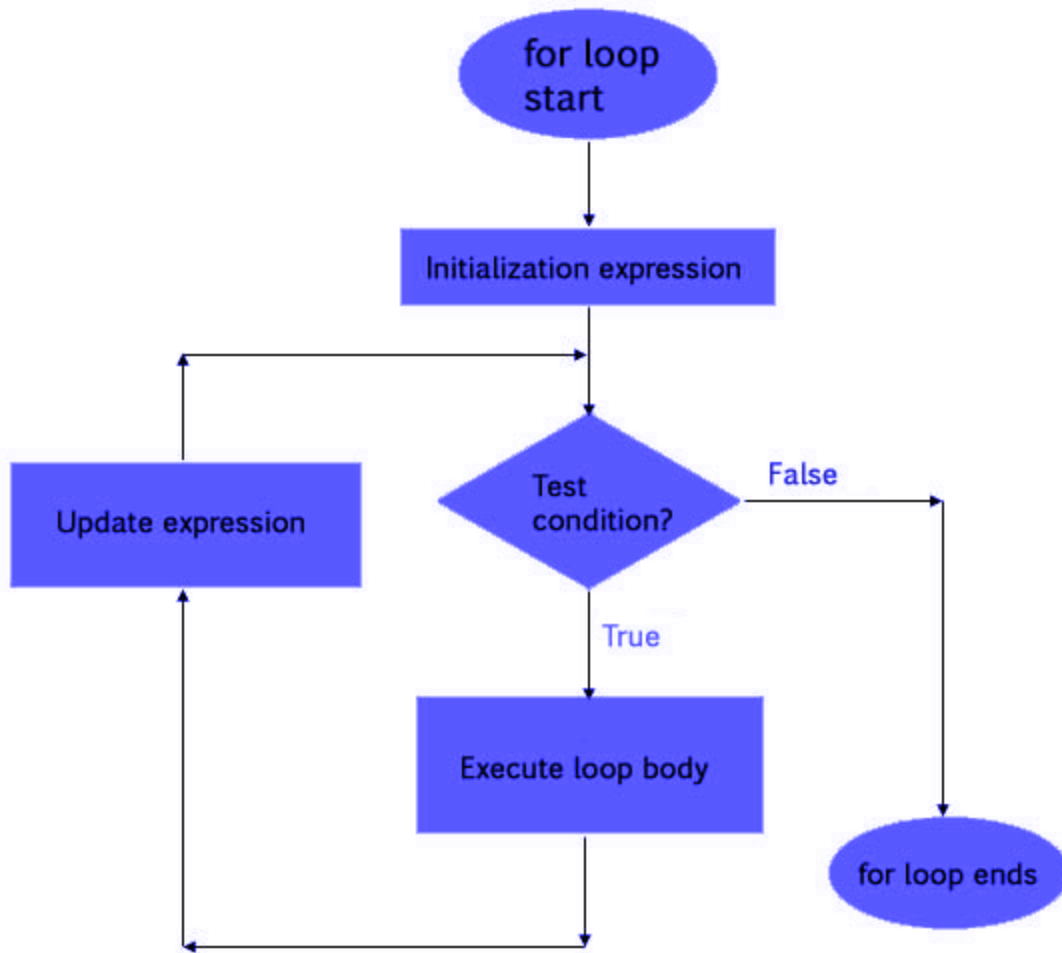
Example:

```
<?php
// code to illustrate for loop
for ($num = 1; $num <= 10; $num +=
2) {
    echo "$num \n";
}
?>
```

Output:

```
1
3
5
7
9
```

Flow Diagram:



while loop: The while loop is also an entry control loop like for loops i.e., it first checks the condition at the start of the loop and if it's true then it enters the loop and executes the block of statements, and goes on executing it as long as the condition holds true.

Syntax:

```

while (if the condition is true) {
    // code is executed
}
  
```

Example:

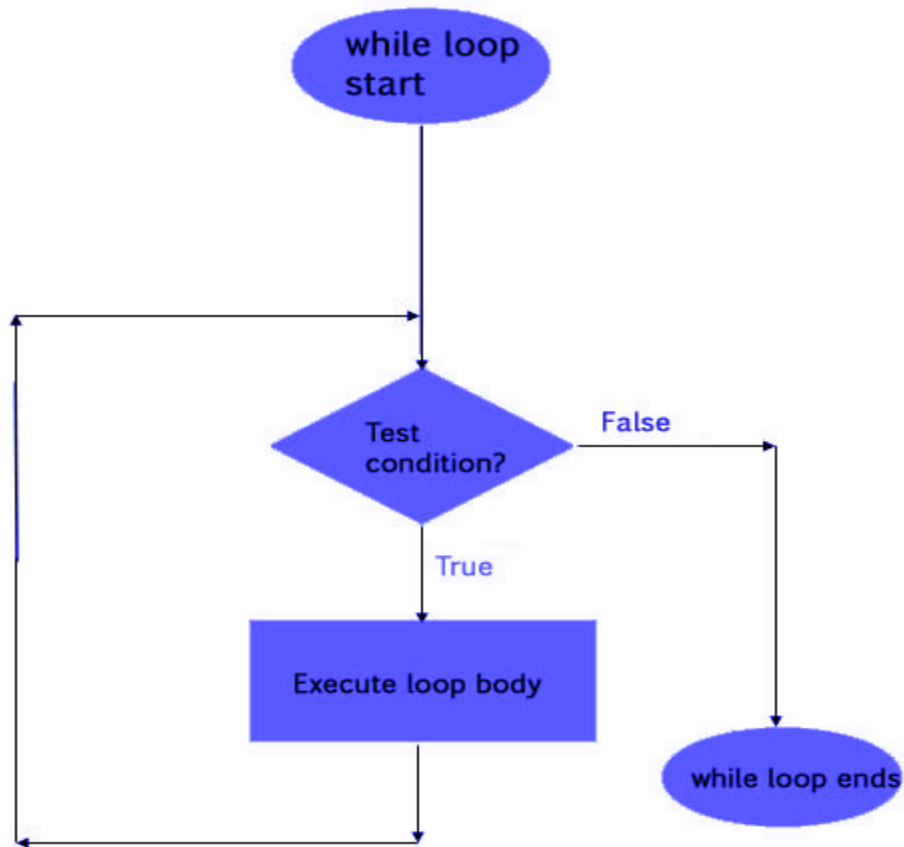
```
<?php
// PHP code to illustrate while
loops
$num = 2;

while ($num < 12) {
    $num += 2;
    echo $num, "\n";
}
?>
```

Output:

4
6
8
10
12

Flowchart:



do-while loop: This is an exit control loop which means that it first enters the loop, executes the statements, and then checks the condition. Therefore, a statement is executed at least once on using the do...while loop. After executing once, the program is executed as long as the condition holds true.

Syntax:

```

do {
    //code is executed
} while (if condition is true);
  
```

Example:

```
<?php
// PHP code to illustrate do...while
loops
$num = 2;
do {
    $num += 2;
    echo $num, "\n";
} while ($num < 12);
?>
```

Output:

```
4
6
8
10
12
```

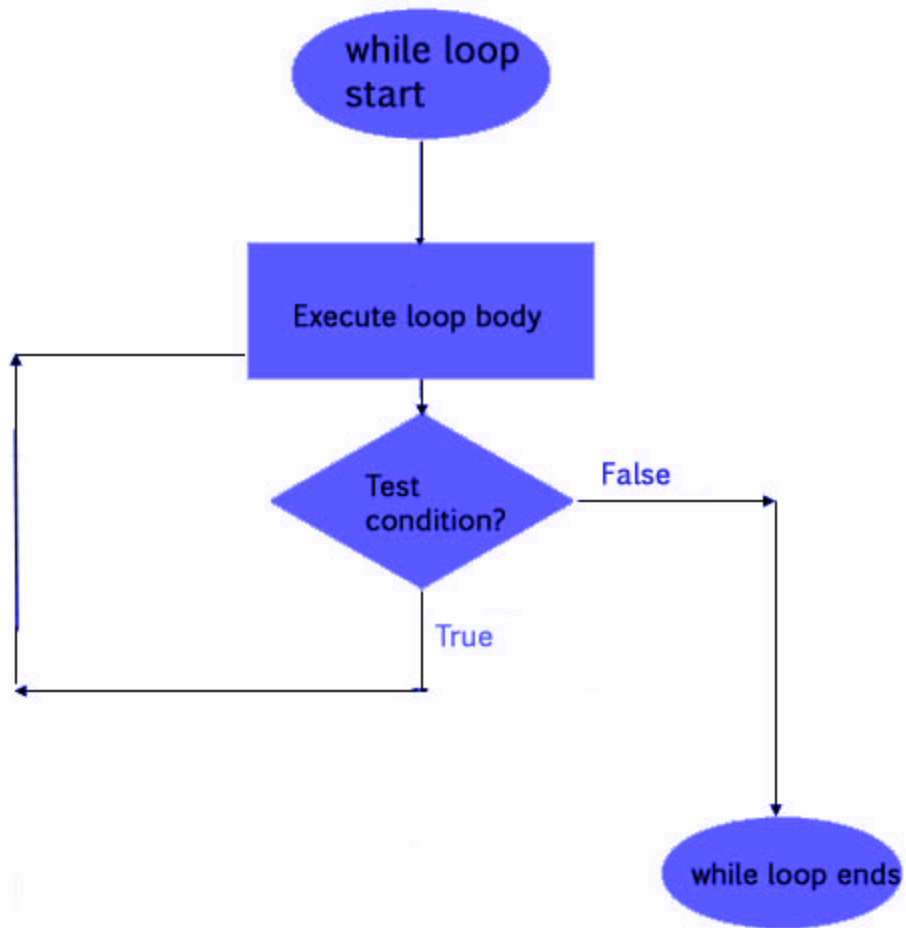
This code would show the difference between while and do...while loop.

```
<?php
// PHP code to illustrate the difference of two
loops
$num = 2;
// In case of while
while ($num != 2) {
    echo "In case of while the code is skipped";
    echo $num, "\n";
}
// In case of do...while
do {
    $num++;
    echo "The do...while code is executed at least
once ";
} while($num == 2);
?>
```

Output:

The code is executed at least once

Flowchart:



foreach loop: This loop is used to iterate over arrays. For every counter of the loop, an array element is assigned and the next counter is shifted to the next element.

Syntax:

```
foreach (array_element as value) {  
    //code to be executed  
}
```

Example:

```
<?php
    $arr = array (10, 20, 30, 40, 50,
60);
    foreach ($arr as $val) {
        echo "$val \n";
    }
    $arr = array ("Ram", "Laxman",
"Sita");
    foreach ($arr as $val) {
        echo "$val \n";
    }
?>
```

Output:

10

20

30

40

50

60

Ram

Laxman

Sita

PHP | Decision Making

PHP allows us to perform actions based on some type of conditions that may be logical or comparative. Based on the result of these conditions i.e., either TRUE or FALSE, an action would be performed as asked by the user. It's just like a two- way path. If you want something then go this way or else turn that way. To use this feature, PHP provides us with four conditional statements:

- **if** statement
- **if...else** statement
- **if...elseif...else** statement
- **switch** statement

Let us now look at each one of these in details:

if Statement: This statement allows us to set a condition. On being TRUE, the following block of code enclosed within the if clause will be executed.

Syntax :

```
if (condition){  
    // if TRUE then execute this code  
}
```

1.

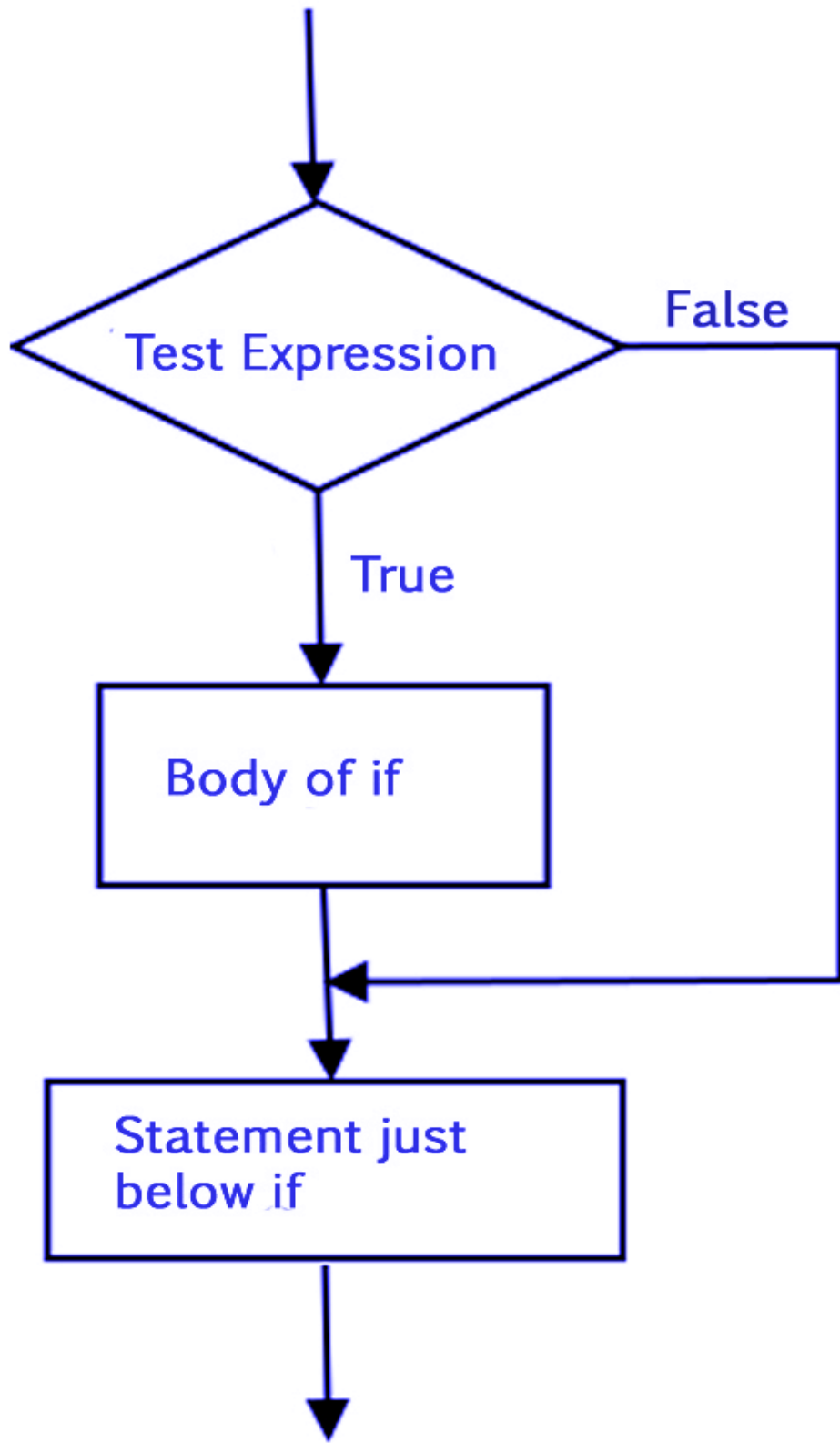
Example:

```
<?php  
$x = 12;  
  
if ($x > 0) {  
    echo "The number is  
positive";  
}  
?>
```

Output:

The number is positive

Flowchart:



if...else Statement: We understood that if a condition will hold i.e., TRUE, then the block of code within if will be executed. But what if the condition is not TRUE and we want to perform an action? This is where else comes into play. If a condition is TRUE then if block gets executed, otherwise else block gets executed.

Syntax:

```
if (condition) {  
    // if TRUE then execute this code  
}  
else{  
    // if FALSE then execute this code  
}
```

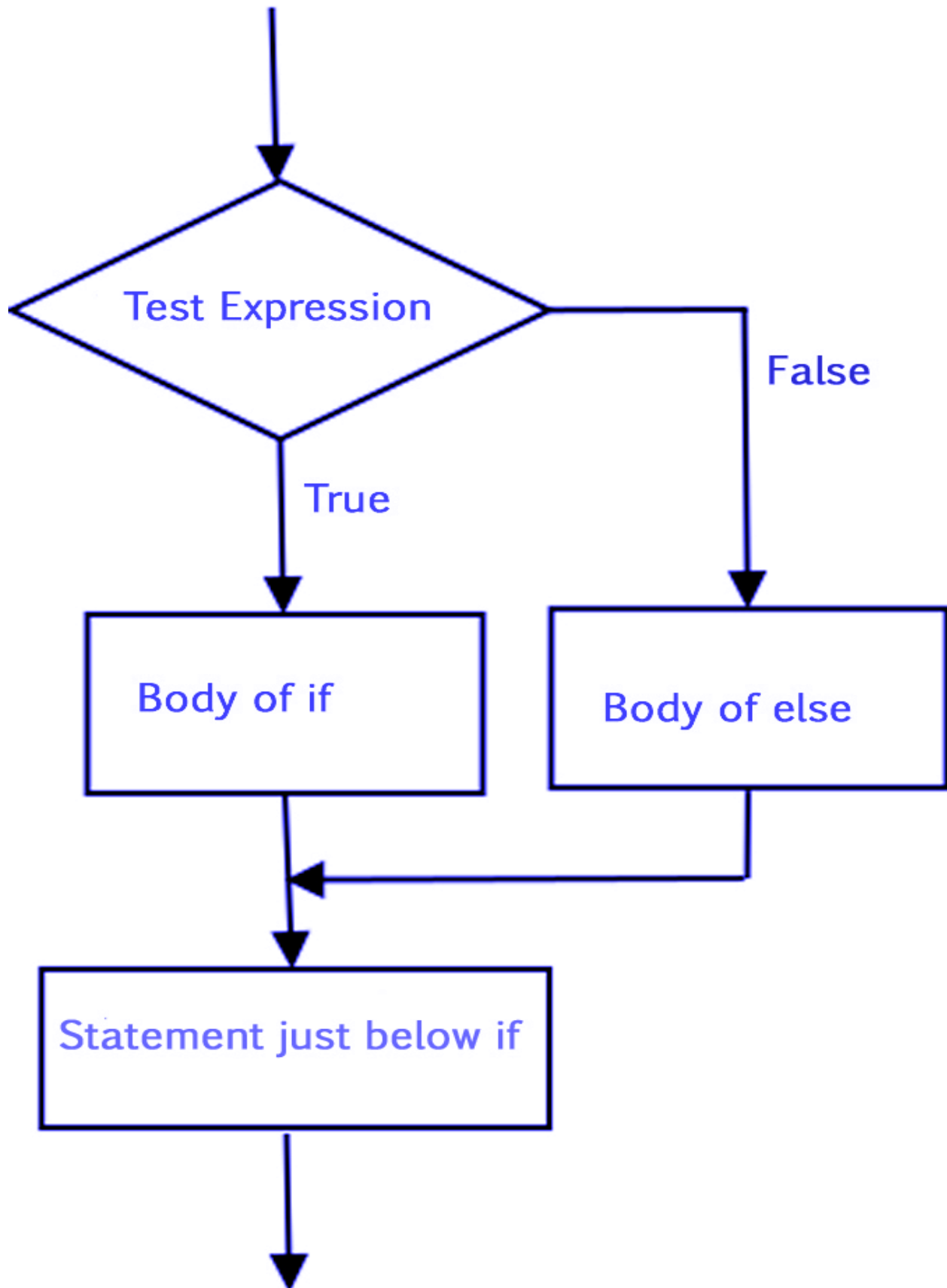
Example:

```
<?php  
$x = -12;  
  
if ($x > 0) {  
    echo "The number is  
positive";  
}  
  
else{  
    echo "The number is  
negative";  
}  
?>
```

Output:

The number is negative

Flowchart:



if...elseif...else Statement: This allows us to use multiple if...else statements. We use this when there are multiple conditions of TRUE cases.

Syntax:

```
if (condition) {
    // if TRUE then execute this code
}
elseif {
    // if TRUE then execute this code
}
elseif {
    // if TRUE then execute this code
}
else {
    // if FALSE then execute this code
}
```

Example:

```
<?php
$x = "August";

if ($x == "January") {
    echo "Happy Republic Day";
}

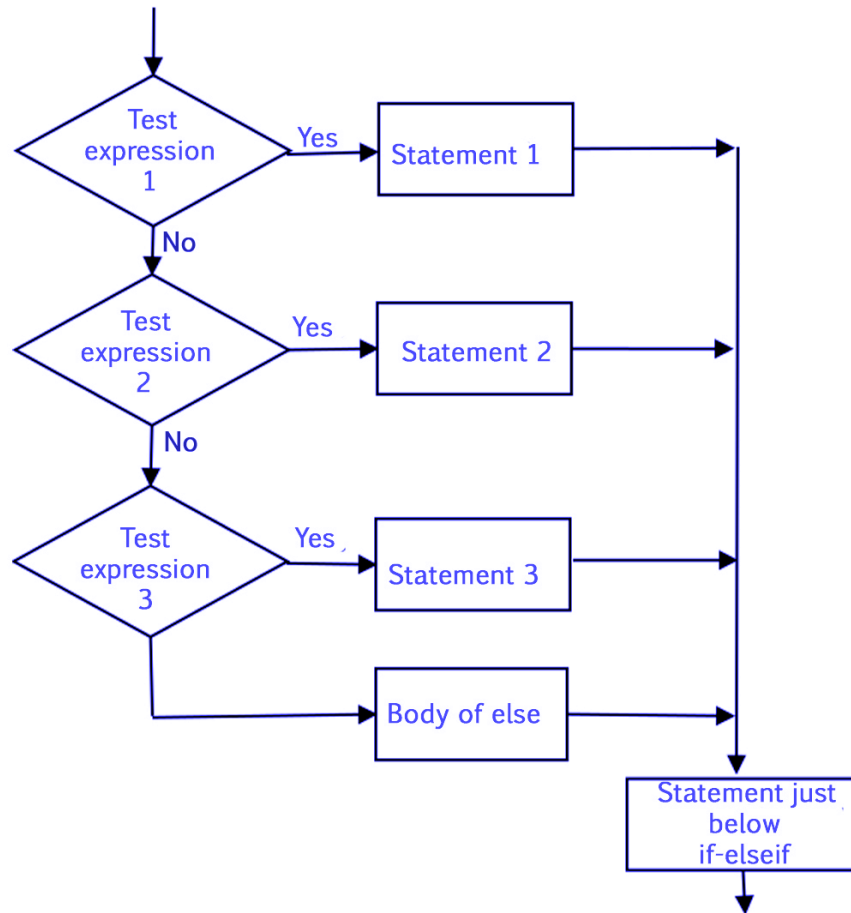
elseif ($x == "August") {
    echo "Happy Independence
Day!!!";
}

else{
    echo "Nothing to show";
}
?>
```

Output:

Happy Independence Day!!!

Flowchart:



switch Statement: The “switch” performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares with the values of each case. If a case matches then the same case is executed. To use switch, we need to get familiar with two different keywords namely, **break** and **default**.

1. The **break** statement is used to stop the automatic control flow into the next cases and exit from the switch case.
2. The **default** statement contains the code that would execute if none of the cases match.

Syntax:

```
switch(n) {
```

```
case statement1:
    code to be executed if n==statement1;
    break;
case statement2:
    code to be executed if n==statement2;
    break;
case statement3:
    code to be executed if n==statement3;
    break;
case statement4:
    code to be executed if n==statement4;
    break;
.....
default:
    code to be executed if n != any case;
```

Example:

```

<?php
$n = "February";

switch($n) {
    case "January":
        echo "Its January";
        break;
    case "February":
        echo "Its
February";
        break;
    case "March":
        echo "Its March";
        break;
    case "April":
        echo "Its April";
        break;
    case "May":
        echo "Its May";
        break;
    case "June":
        echo "Its June";
        break;
    case "July":
        echo "Its July";
        break;
    case "August":
        echo "Its August";
        break;
    case "September":
        echo "Its
September";
        break;
    case "October":
        echo "Its October";
        break;
    case "November":
        echo "Its
November";
        break;
    case "December":
        echo "Its
December";
        break;
}

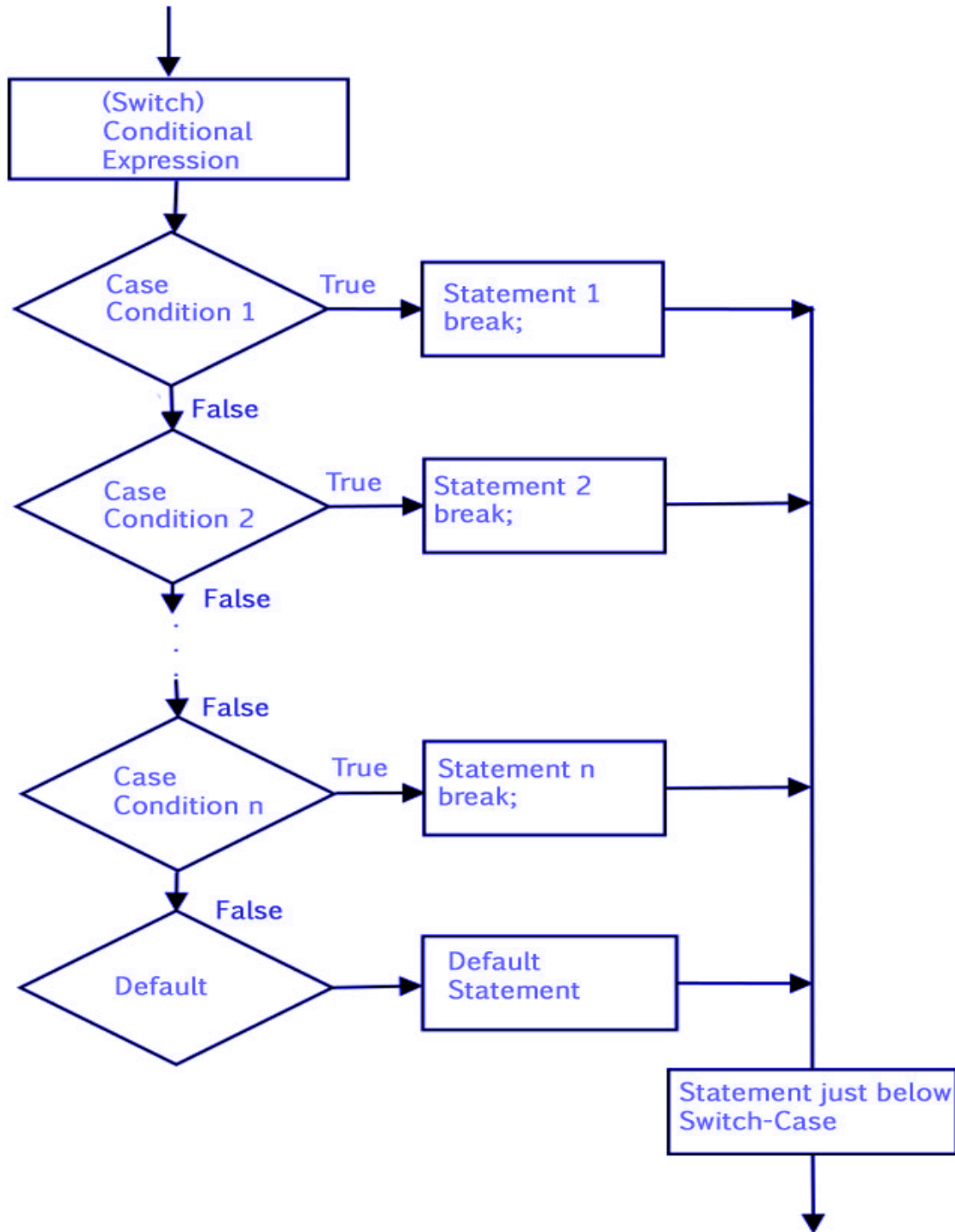
```

```
default:
    echo "Doesn't
exist";
}
?>
```

Output:

Its February

Flowchart:



Ternary Operators

In addition to all these conditional statements, PHP provides a shorthand way of writing if...else, called Ternary Operators. The statement uses a question mark (?) and a colon (:) and takes three operands: a condition to check, a result for TRUE and a result for FALSE.

Syntax:

(condition) ? if TRUE execute this : otherwise execute this;

Example:

```
<?php
$x = -12;
if ($x > 0) {
    echo "The number is positive \n";
}
else {
    echo "The number is negative \n";
}
// This whole lot can be written in a
// single line using ternary operator
echo ($x > 0) ? 'The number is
positive' :
                'The number is
negative';
?>
```

Output:

The number is negative
The number is negative

PHP | Arrays

Arrays in PHP are a type of data structure that allows us to store multiple elements of similar data type under a single variable thereby saving us the effort of creating a different variable for every data. The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key. Suppose we want to store five names and print them accordingly. This can be easily done by the use of five different string variables. But if instead of five, the number rises to a hundred, then it would be really difficult for the user or developer to create so many different variables. Here array comes into play and helps us to store every element within a single variable

and also allows easy access using an index or a key. An array is created using an **array()** function in PHP.

There are basically three types of arrays in PHP:

- **Indexed or Numeric Arrays:** An array with a numeric index where values are stored linearly.
- **Associative Arrays:** An array with a string index where instead of linear storage, each value can be assigned a specific key.
- **Multidimensional Arrays:** An array which contains a single or multiple array within it and can be accessed via multiple indices.

Indexed or Numeric Arrays

These types of arrays can be used to store any type of elements, but an index is always a number. By default, the index starts at zero. These arrays can be created in two different ways as shown in the following example:

```
<?php

// One way to create an indexed array
$name_one = array("Zack", "Anthony", "Ram", "Salim",
"Raghav");
// Accessing the elements directly
echo "Accessing the 1st array elements directly:\n";
echo $name_one[2], "\n";
echo $name_one[0], "\n";
echo $name_one[4], "\n";
// Second way to create an indexed array
$name_two[0] = "ZACK";
$name_two[1] = "ANTHONY";
$name_two[2] = "RAM";
$name_two[3] = "SALIM";
$name_two[4] = "RAGHAV";
// Accessing the elements directly
echo "Accessing the 2nd array elements directly:\n";
echo $name_two[2], "\n";
echo $name_two[0], "\n";
echo $name_two[4], "\n";
?>
```

Output:

Accessing the 1st array elements directly:

Ram

Zack

Raghav

Accessing the 2nd array elements directly:

RAM

ZACK

RAGHAV

Example:

```

<?php
// Creating an indexed array
$name_one = array("Zack", "Anthony", "Ram", "Salim",
"Raghav");
// One way of Looping through an array using foreach
echo "Looping using foreach: \n";
foreach ($name_one as $val){
    echo $val. "\n";
}
// count() function is used to count
// the number of elements in an array
$round = count($name_one);
echo "\nThe number of elements are $round \n";
// Another way to loop through the array using for
echo "Looping using for: \n";
for($n = 0; $n < $round; $n++){
    echo $name_one[$n], "\n";
}
?>

```

Output:

Looping using foreach:

Zack

Anthony

Ram

Salim

Raghav

The number of elements is 5

Looping using for:

ZACK

ANTHONY

RAM

SALIM

RAGHAV

Associative Arrays

These types of arrays are similar to the indexed arrays but instead of linear storage, every value can be assigned with a user-defined key of string type.

Example:

```
<?php
// One way to create an associative array
$name_one = array("Zack"=>"Zara",
"Anthony"=>"Any",
"Ram"=>"Rani",
"Salim"=>"Sara",
"Raghav"=>"Ravina");
// Second way to create an associative array
$name_two["zack"] = "zara";
$name_two["anthony"] = "any";
$name_two["ram"] = "rani";
$name_two["salim"] = "sara";
$name_two["raghav"] = "ravina";
// Accessing the elements directly
echo "Accessing the elements directly:\n";
echo $name_two["zack"], "\n";
echo $name_two["salim"], "\n";
echo $name_two["anthony"], "\n";
echo $name_one["Ram"], "\n";
echo $name_one["Raghav"], "\n";
?>
```

Output:

Accessing the elements directly:

zara

sara

any

Rani

Ravina

Example:

```

<?php
// Creating an associative array
$name_one = array("Zack"=>"Zara", "Anthony"=>"Any",
                 "Ram"=>"Rani", "Salim"=>"Sara",
                 "Raghav"=>"Ravina");

// Looping through an array using foreach
echo "Looping using foreach: \n";
foreach ($name_one as $val => $val_value){
    echo "Husband is ".$val." and Wife is
    ".$val_value."\n";
}
// Looping through an array using for
echo "\nLooping using for: \n";
$keys = array_keys($name_two);
$round = count($name_two);

for($i=0; $i < $round; ++$i) {
    echo $keys[$i] . ' ' . $name_two[$keys[$i]] .
    "\n";
}
?>

```

Output:

Looping using foreach:

Husband is Zack and Wife is Zara
Husband is Anthony and Wife is Any
Husband is Ram and Wife is Rani
Husband is Salim and Wife is Sara
Husband is Raghav and Wife is Ravina

Looping using for:

zack zara
anthony any
ram rani
salim sara
raghav ravina

Multidimensional Arrays

Multi-dimensional arrays are such arrays that store another array at each index instead of a single element. In other words, we can define multi-dimensional arrays as an array of arrays. As the name suggests, every element in this array can be an array and they can also hold other sub-arrays within. Arrays or sub-arrays in multidimensional arrays can be accessed using multiple dimensions.

Example:

```
<?php
// Defining a multidimensional array
$favorites = array(
    array(
        "name" => "Dave Punk",
        "mob" => "5689741523",
        "email" => "davepunk@gmail.com",
    ),
    array(
        "name" => "Monty Smith",
        "mob" => "2584369721",
        "email" => "montysmith@gmail.com",
    ),
    array(
        "name" => "John Flinch",
        "mob" => "9875147536",
        "email" => "johnflinch@gmail.com",
    )
);
// Accessing elements
echo "Dave Punk email-id is: " . $favorites[0]["email"],
"\n";
echo "John Flinch mobile number is: " .
$favorites[2]["mob"];
?>
```

Output:

Dave Punk email-id is: davepunk@gmail.com

John Flinch mobile number is: 9875147536

Example:


```

<?php
// Defining a multidimensional array
$favorites = array(
    "Dave Punk" => array(
        "mob" => "5689741523",
        "email" => "davepunk@gmail.com",
    ),
    "Dave Punk" => array(
        "mob" => "2584369721",
        "email" => "montysmith@gmail.com",
    ),
    "John Flinch" => array(
        "mob" => "9875147536",
        "email" => "johnflinch@gmail.com",
    )
);
// Using for and foreach in nested form
$keys = array_keys($favorites);
for($i = 0; $i < count($favorites); $i++) {
    echo $keys[$i] . "\n";
    foreach($favorites[$keys[$i]] as $key =>
$value) {
        echo $key . " : " . $value . "\n";
    }
    echo "\n";
}
?>

```

Output:

```

Dave Punk
mob : 2584369721
email : montysmith@gmail.com

```

```

John Flinch
mob : 9875147536
email : johnflinch@gmail.com

```

PHP | array() Function

The array() function is an inbuilt function in PHP which is used to create an array. There are three types of array in PHP:

- **Indexed array:** The array which contains numeric index.

Syntax:

```
array( val1, val2, val3, ... )
```

- **Associative array:** The array which contains name as keys.

Syntax:

```
array( key=>val, key=>val, key=>value, ... )
```

- **Multidimensional array:** The array which contains one or more arrays.

Syntax:

```
array( array( val11, val12, ...
array( val21, val22, ... ) ... )
```

Parameters: This function accepts at most two parameters as mentioned above and described below:

- **val:** This parameter is used to hold the value of the array.
- **key:** This parameter is used to hold the key value:

Return Value: This function returns an array of parameters.

Below programs illustrates the array() function in PHP:

Program 1: This example illustrates the Indexed array.

```
<?php
// Create an array
$sub = array("DBMS", "Algorithm", "C++",
"JAVA");
// Find length of array
$len = count( $sub );
// Loop to print array elements
for( $i = 0; $i < $len; $i++) {
    echo $sub[$i] . "\n";
}
?>
```

Output:

DBMS

Algorithm

C++

JAVA

Program 2: This example illustrates the Associative array.

```
<?php
// Declare an associative array
$detail = array( "Name"=>"AIT",
                "Address"=>"AHMEDABAD",
                "Type"=>"Education
Institute");
// Display the output
var_dump ($detail);
?>
```

Output:

```
array(3) {
  ["Name"]=>
  string(3) "AIT"
  ["Address"]=>
  string(9) "AHMEDABAD"
  ["Type"]=>
  string(19) "Education Institute"
}
```

Program 3: This example illustrates the Multidimensional array.

```
<?php
// Declare 2D array
$detail = array(array(1, 2, 3,
4),
                array(5, 6, 7,
8));
// Display the output
var_dump ($detail);
?>
```

Output:

```
array(2) {
  [0]=>
  array(4) {
    [0]=>
    int(1)
    [1]=>
    int(2)
    [2]=>
    int(3)
    [3]=>
    int(4)
  }
  [1]=>
  array(4) {
    [0]=>
    int(5)
    [1]=>
    int(6)
    [2]=>
    int(7)
    [3]=>
    int(8)
  }
}
```

PHP Functions

PHP function is a piece of code that can be reused many times. It can take input as an argument list and return value. There are thousands of built-in functions in PHP.

In PHP, we can define **Conditional function**, **Function within Function** and **Recursive function** also

Advantage of PHP Functions

Code Reusability: PHP functions are defined only once and can be invoked many times, like in other programming languages.

Less Code: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.

Easy to understand: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP User-defined Functions

We can declare and call user-defined functions easily. Let's see the syntax to declare user-defined functions.

Syntax

```
function functionname(){  
  
//code to be executed  
  
}
```

Note: Function name must be started with letter and underscore only like other labels in PHP. It can't be started with numbers or special symbols.

PHP Functions Example

File: function1.php

```
<?php
function sayHello() {
echo "Hello PHP Function";
}
sayHello();//calling function
?>
```

Output:

Hello PHP Function

PHP Function Arguments

We can pass the information in the PHP function through arguments which are separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.

Let's see the example to pass a single argument in the PHP function.

File: functionarg.php

```
<?php
function sayHello($name) {
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>
```

Output:

Hello Sonoo
Hello Vimal
Hello John

Let's see the example to pass two arguments in the PHP function.

File: functionarg2.php

```
<?php
function sayHello($name,$age) {
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>
```

Output:

Hello Sonoo, you are 27 years old
Hello Vimal, you are 29 years old
Hello John, you are 23 years old

PHP Call By Reference

Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.

By default, value passed to the function is called by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

Let's see a simple example of a call by reference in PHP.

File: functionref.php

```
<?php
```

```

function adder(&$str2)
{
    $str2 .= 'Call By Reference';
}
$str = 'Hello ';
adder($str);
echo $str;
?>

```

Output:

Hello Call By Reference

PHP Function: Default Argument Value

We can specify a default argument value in function. While calling a PHP function if you don't specify any argument, it will take the default argument. Let's see a simple example of using default argument values in PHP function.

File: functiondefaultarg.php

```

<?php
function sayHello($name="Sonoo") {
    echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");
?>

```

Output:

Hello Rajesh
Hello Sonoo

Hello John

PHP Function: Returning Value

Let's see an example of a PHP function that returns value.

File: functiondefaultarg.php

```
<?php
function cube($n) {
return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);
?>
```

Output:

Cube of 3 is: 27

PHP Parameterized Function

PHP Parameterized functions are the functions with parameters. You can pass any number of parameters inside a function. These passed parameters act as variables inside your function.

They are specified inside the parentheses, after the function name.

The output depends upon the dynamic values passed as the parameters into the function.

PHP Parameterized Example 1

Addition and Subtraction

In this example, we have passed two parameters **\$x** and **\$y** inside two functions **add()** and **sub()**.

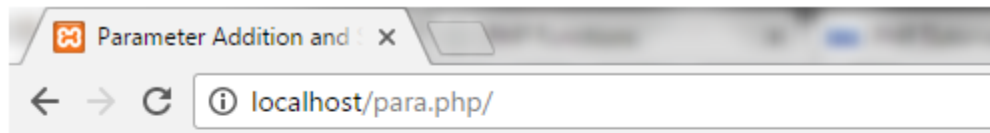
```
<!DOCTYPE html>
```

```

<html>
<head>
    <title>Parameter Addition and Subtraction Example</title>
</head>
<body>
<?php
    //Adding two numbers
    function add($x, $y) {
        $sum = $x + $y;
        echo "Sum of two numbers is = $sum <br><br>";
    }
    add(467, 943);
    //Subtracting two numbers
    function sub($x, $y) {
        $diff = $x - $y;
        echo "Difference between two numbers is = $diff";
    }
    sub(943, 467);
?>
</body>
</html>

```

Output:



Sum of two numbers is = 1410

Difference between two numbers is = 476

PHP Call By Value

PHP allows you to call functions by value and reference both. In case of PHP call by value, actual value is not modified if it is modified inside the function.

Let's understand the concept of call by value by the help of examples.

Example 1

In this example, variable \$str is passed to the adder function where it is concatenated with the 'Call By Value' string. But, printing \$str variable results 'Hello' only. It is because changes are done in the local variable \$str2 only. It doesn't reflect the \$str variable.

```
<?php
function adder($str2)
{
    $str2 .= 'Call By Value';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output:

Hello

PHP Call By Reference

In case of PHP call by reference, actual value is modified if it is modified inside the function. In such a case, you need to use the & (ampersand) symbol with formal arguments. The & represents reference of the variable.

Let's understand the concept of call by reference by the help of examples.

Example 1

In this example, variable \$str is passed to the adder function where it is concatenated with the 'Call By Reference' string. Here, printing \$str variable results 'This is Call By Reference'. It is because changes are done in the actual variable \$str.

```
<?php
function adder(&$str2)
{
    $str2 .= 'Call By Reference';
}
$str = 'This is ';
adder($str);
```

```
echo $str;  
?>
```

Output:

This is Call By Reference

PHP Default Argument Values Function

PHP allows you to define C++ style default argument values. In such a case, if you don't pass any value to the function, it will use the default argument value.

Let's see the simple example of using PHP default arguments in function.

Example 1

```
<?php  
function sayHello($name="Ram") {  
echo "Hello $name<br/>";  
}  
sayHello("Sonoo");  
sayHello();//passing no value  
sayHello("Vimal");  
?>
```

Output:

Hello Sonoo

Hello Ram

Hello Vimal

PHP Strings

A string is a sequence of characters, like "Hello world!".

PHP String Functions

In this chapter we will look at some commonly used functions to manipulate strings.

strlen() - Return the Length of a String

The PHP strlen() function returns the length of a string.

Example

Return the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

str_word_count() - Count Words in a String

The PHP str_word_count() function counts the number of words in a string.

Example

Count the number of word in the string "Hello world!":

```
<!DOCTYPE html>
<html>
<body>
<?php
echo str_word_count("Hello world!");
?>
</body>
</html>
```

Output:

2

strrev() - Reverse a String

The PHP strrev() function reverses a string.

Example

Reverse the string "Hello world!":

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

strpos() - Search For a Text Within a String

The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

Example

Search for the text "world" in the string "Hello world!":

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

Tip: The first character position in a string is 0 (not 1).

str_replace() - Replace Text Within a String

The PHP str_replace() function replaces some characters with some other characters in a string.

Example

Replace the text "world" with "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

PHP Form Handling

We can create and use forms in PHP. To get form data, we need to use PHP superglobals \$_GET and \$_POST.

The form request may be get or post. To retrieve data from a get request, we need to use \$_GET, for post request \$_POST.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send a limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

File: form1.html

```
<form action="welcome.php" method="get">
Name: <input type="text" name="name"/>
<input type="submit" value="visit"/>
</form>
```

File: welcome.php

```
<?php
$name=$_GET["name"];//receiving name field value in $name
variable
echo "Welcome, $name";
?>
```

PHP Post Form

Post request is widely used to submit forms that have a large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send a large amount of data through post requests.

Let's see a simple example to receive data from a post request in PHP.

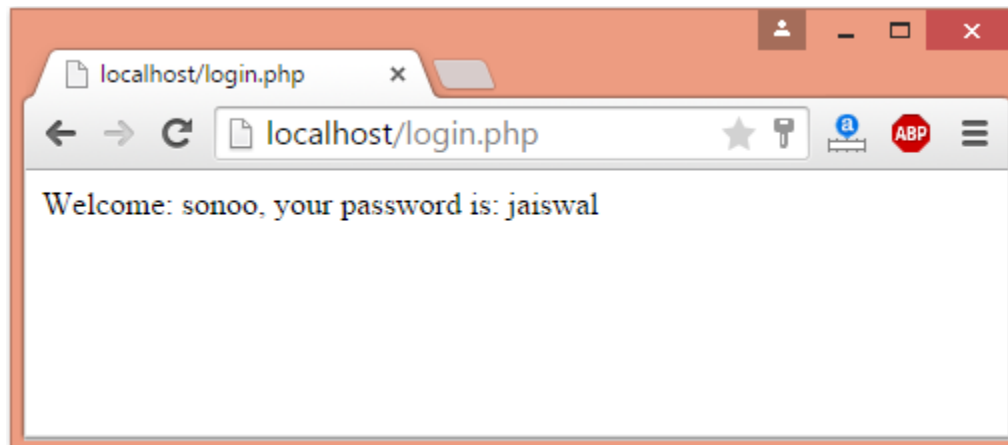
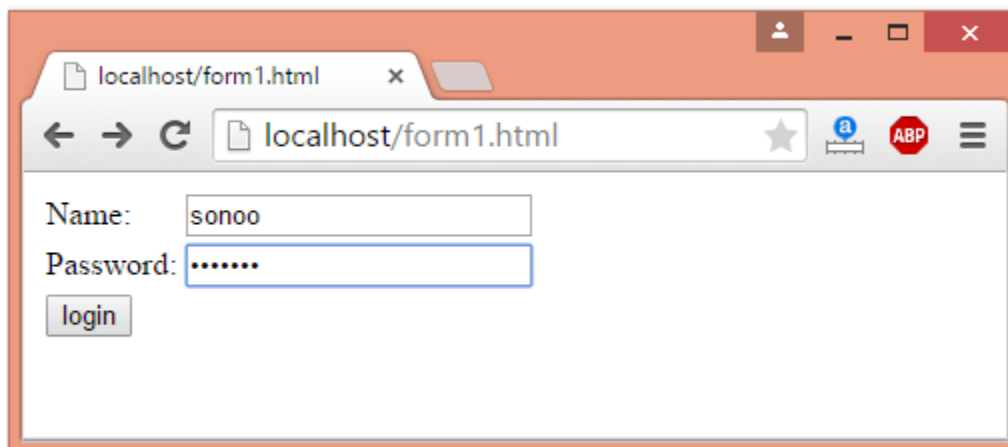
File: form1.html

```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td>
<input type="text"
name="name"/></td></tr>
<tr><td>Password:</td><td>
<input type="password"
name="password"/></td></tr>
<tr><td colspan="2"><input type="submit" value="login"/>
</td></tr>
</table>
</form>
```

File: login.php

```
<?php
$name=$_POST["name");//receiving name field value in $name
variable
$password=$_POST["password");//receiving password field value in
$password variable
echo "Welcome: $name, your password is: $password";
?>
```

Output:



Attributes of Form Tag:

Attribute	Description
-----------	-------------

name or id	It specifies the name of the form and is used to identify individual forms.
action	It specifies the location to which the form data has to be sent when the form is submitted.
method	It specifies the HTTP method that is to be used when the form is submitted. The possible values are get and post . If get method is used, the form data is visible to the users in the url. Default HTTP method is get .
encType	It specifies the encryption type for the form data when the form is submitted.
novalidate	It implies the server not to verify the form data when the form is submitted.

Controls used in forms: Form processing contains a set of controls through which the client and server can communicate and share information. The controls used in forms are:

- **Textbox:** Textbox allows the user to provide single-line input, which can be used for getting values such as names, search menu etc.
- **Textarea:** Textarea allows the user to provide multi-line input, which can be used for getting values such as an address, message etc.
- **DropDown:** Dropdown or combobox allows the user to select a value from a list of values.
- **Radio Buttons:** Radio buttons allow the user to select only one option from the given set of options.

- **CheckBox:** Checkbox allows the user to select multiple options from the set of given options.
- **Buttons:** Buttons are the clickable controls that can be used to submit the form.

Creating a simple HTML Form: All the form controls given above is designed by using the **input** tag based on the **type** attribute of the tag. In the below script, when the form is submitted, no event handling mechanism is done. Event handling refers to the process done while the form is submitted. These event handling mechanisms can be done by using javaScript or PHP. However, JavaScript provides only client-side validation. Hence, we can use PHP for form processing.

HTML Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple Form Processing</title>
</head>
<body>
  <form id="form1" method="post">
    FirstName:
    <input type="text" name="firstname" required/>
    <br>
    <br>
    LastName
    <input type="text" name="lastname" required/>
    <br>
    <br>
    Address
    <input type="text" name="address" required/>
    <br>
    <br>
    Email Address:
    <input type="email" name="emailaddress" required/>
    <br>
    <br>
    Password:
```

```

        <input type="password" name="password" required/>
        <br>
        <br>
        <input type="submit" value="Submit"/>
    </form>
</body>
</html>

```

Form Validation: Form validation is done to ensure that the user has provided the relevant information. Basic validation can be done using HTML elements. For example, in the above script, the email address text box is having a type value as “email”, which prevents the user from entering the incorrect value for an email. Every form field in the above script is followed by a required attribute, which will intimate the user not to leave any field empty before submitting the form. PHP methods and arrays used in form processing are:

- **isset():** This function is used to determine whether the variable or a form control is having a value or not.
- **\$_GET[]:** It is used to retrieve the information from the form control through the parameters sent in the URL. It takes the attribute given in the url as the parameter.
- **\$_POST[]:** It is used to retrieve the information from the form control through the HTTP POST method. IT takes the name attribute of corresponding form control as the parameter.
- **\$_REQUEST[]:** It is used to retrieve information while using a database.

Form Processing using PHP: Above HTML script is rewritten using the above mentioned functions and array. The rewritten script validates all the form fields and if there are no errors, it displays the received information in a tabular form.

```

<?php
if (isset($_POST['submit']))
{

```

```

        if          ((!isset($_POST['firstname']))          ||
(!isset($_POST['lastname'])) ||
        (!isset($_POST['address']))          ||
(!isset($_POST['emailaddress'])) ||
        (!isset($_POST['password']))          ||
(!isset($_POST['gender'])))
        {
            $error = "*" . "Please fill all the required fields";
        }
    else
    {
        $firstname = $_POST['firstname'];
        $lastname = $_POST['lastname'];
        $address = $_POST['address'];
        $emailaddress = $_POST['emailaddress'];
        $password = $_POST['password'];
        $gender = $_POST['gender'];
    }
}
?>
<html>

<head>
    <title>Simple Form Processing</title>
</head>

<body>
    <h1>Form Processing using PHP</h1>
    <fieldset>
        <form id="form1" method="post" action="form.php">
            <?php
                if (isset($_POST['submit']))
                {
                    if (isset($error))
                    {
                        echo "<p style='color:red;'"
                            . $error . "</p>";
                    }
                }
            ?>

            FirstName:
            <input type="text" name="firstname"/>
            <span style="color:red;">*</span>
            <br>
            <br>

```

```

Last Name:
<input type="text" name="lastname"/>
<span style="color:red;*></span>
<br>
<br>
Address:
<input type="text" name="address"/>
<span style="color:red;*></span>
<br>
<br>
Email:
<input type="email" name="emailaddress"/>
<span style="color:red;*></span>
<br>
<br>
Password:
<input type="password" name="password"/>
<span style="color:red;*></span>
<br>
<br>
Gender:
<input type="radio"
      value="Male"
      name="gender"> Male
<input type="radio"
      value="Female"
      name="gender">Female
<br>
<br>
<input      type="submit"      value="Submit"
name="submit" />
</form>
</fieldset>
<?php
if(isset($_POST['submit']))
{
    if(!isset($error))
    {
        echo"<h1>INPUT RECEIVED</h1><br>";
        echo "<table border='1'>";
        echo "<thead>";
        echo "<th>Parameter</th>";
        echo "<th>Value</th>";
        echo "</thead>";
        echo "<tr>";
        echo "<td>First Name</td>";
    }
}

```

```

        echo "<td>".$firstname."</td>";
        echo "</tr>";
        echo "<tr>";
        echo "<td>Last Name</td>";
        echo "<td>".$lastname."</td>";
        echo "</tr>";
        echo "<tr>";
        echo "<td>Address</td>";
        echo "<td>".$address."</td>";
        echo "</tr>";
        echo "<tr>";
        echo "<td>Email Address</td>";
        echo "<td>".$emailaddress."</td>";
        echo "</tr>";
        echo "<tr>";
        echo "<td>Password</td>";
        echo "<td>".$password."</td>";
        echo "</tr>";
        echo "<tr>";
        echo "<td>Gender</td>";
        echo "<td>".$gender."</td>";
        echo "</tr>";
        echo "</table>";
    }
}
?>
</body>

</html>

```

Output:

The screenshot shows a web browser window displaying a form titled "Form Processing using PHP". The form is enclosed in a rectangular border and contains the following elements:

- Five text input fields, each with a red arrow pointing to the right, labeled "FirstName", "Last Name", "Address", "Email", and "Password".
- A "Gender" section with two radio buttons: "Male" (which is selected) and "Female".
- A "Submit" button located at the bottom left of the form area.

Note: When the PHP and HTML are coded in a single file, the file should be saved as PHP. In the form, the value for the action parameter should be a file name.

PHP File Upload

PHP allows you to upload single and multiple files through a few lines of code only.

PHP file upload features allow you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

PHP \$_FILES

The PHP global \$_FILES contains all the information of the file. By the help of \$_FILES global, we can get file name, file type, file size, temp file name and errors associated with the file.

Here, we are assuming that file name is *filename*.

```
$_FILES['filename']['name']
```

returns file name.

```
$_FILES['filename']['type']
```

returns MIME type of the file.

```
$_FILES['filename']['size']
```

returns size of the file (in bytes).

```
$_FILES['filename']['tmp_name']
```

returns the temporary file name of the file which was stored on the server.

```
$_FILES['filename']['error']
```

returns error code associated with this file.

`move_uploaded_file()` function

The `move_uploaded_file()` function moves the uploaded file to a new location. The `move_uploaded_file()` function checks internally if the file is uploaded through the POST request. It moves the file if it is uploaded through the POST request.

Syntax

1. `bool move_uploaded_file (string $filename , string $destination)`

PHP File Upload Example

File: uploadform.html

```
<form action="uploader.php" method="post"
enctype="multipart/form-data">
    Select File:
    <input type="file" name="fileToUpload"/>
    <input type="submit" value="Upload Image" name="submit"/>
</form>
```

File: uploader.php

```
<?php
$target_path = "e:/";
$target_path = $_FILES['fileToUpload']['tmp_name'];
$target_path = $target_path.basename(
$_FILES['fileToUpload']['name']);

if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'],
$target_path)) {
    echo "File uploaded successfully!";
} else{
    echo "Sorry, file not uploaded, please try again!";
}
?>
```

PHP Download File

PHP enables you to download files easily using the built-in `readfile()` function. The `readfile()` function reads a file and writes it to the output buffer.

PHP readfile() function

Syntax

1. `int readfile (string $filename [, bool $use_include_path = false [, resource $context]])`

\$filename: represents the file name

\$use_include_path: it is the optional parameter. It is by default false. You can set it to true to search the file in the included_path.

\$context: represents the context stream resource.

int: it returns the number of bytes read from the file.

PHP Download File Example: Text File

File: download1.php

```
<?php
$file_url = 'http://www.google.com/f.txt';
header('Content-Type: application/octet-stream');
header("Content-Transfer-Encoding: utf-8");
header("Content-disposition: attachment; filename=\"\"");
basename($file_url) . "\"");
readfile($file_url);
?>
```

PHP Download File Example: Binary File

File: download2.php

```
<?php
$file_url = 'http://www.myremoteserver.com/file.exe';
header('Content-Type: application/octet-stream');
header("Content-Transfer-Encoding: Binary");
header("Content-disposition: attachment; filename=\"\"");
basename($file_url) . "\"");
readfile($file_url);
?>
```

PHP Date and Time

The PHP date() function is used to format a date and/or a time.

The PHP Date() Function

The PHP date() function formats a timestamp to a more readable date and time.

Syntax

`date(format,timestamp)`

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time

A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.

Get a Date

The required *format* parameter of the date() function specifies how to format the date (or time).

Here are some characters that are commonly used for dates:

- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)
- l (lowercase 'L') - Represents the day of the week

Other characters, like "/", ".", or "-" can also be inserted between the characters to add additional formatting.

The example below formats today's date in three different ways:

Example

```
<!DOCTYPE html>
<html>
<body>
<?php
```

```
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>
</body>
</html>
```

Today is 2020/11/03

Today is 2020.11.03

Today is 2020-11-03

Today is Tuesday

PHP Tip - Automatic Copyright Year

Use the date() function to automatically update the copyright year on your website:

Example

```
&copy; 2010-<?php echo date("Y");?>
```

Get a Time

Here are some characters that are commonly used for times:

- H - 24-hour format of an hour (00 to 23)
- h - 12-hour format of an hour with leading zeros (01 to 12)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds with leading zeros (00 to 59)
- a - Lowercase Ante meridiem and Post meridiem (am or pm)

The example below outputs the current time in the specified format:

Example

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

Note that the PHP date() function will return the current date/time of the server!

Get Your Time Zone

If the time you got back from the code is not correct, it's probably because your server is in another country or set up for a different timezone.

So, if you need the time to be correct according to a specific location, you can set the timezone you want to use.

The example below sets the timezone to "America/New_York", then outputs the current time in the specified format:

Example

```
<?php
date_default_timezone_set("America/New_York");
echo "The time is " . date("h:i:sa");
?>
```

Create a Date With mktime()

The optional *timestamp* parameter in the `date()` function specifies a timestamp. If omitted, the current date and time will be used (as in the examples above).

The PHP `mktime()` function returns the Unix timestamp for a date. The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

Syntax

```
mktime(hour, minute, second, month, day, year)
```

The example below creates a date and time with the `date()` function from a number of parameters in the `mktime()` function:

Example

```
<?php
$d=mktime(11, 14, 54, 8, 12, 2014);
echo "Created date is " . date("Y-m-d h:i:sa", $d);
```

?>

Create a Date From a String With strtotime()

The PHP strtotime() function is used to convert a human readable date string into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT).

Syntax

strtotime(*time, now*)

The example below creates a date and time from the strtotime() function:

Example

```
<?php
$d=strtotime("10:30pm April 15 2014");
echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

PHP is quite clever about converting a string to a date, so you can put in various values:

Example

```
<?php
$d=strtotime("tomorrow");
echo date("Y-m-d h:i:sa", $d) . "<br>";
$d=strtotime("next Saturday");
echo date("Y-m-d h:i:sa", $d) . "<br>";
$d=strtotime("+3 Months");
echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

However, strtotime() is not perfect, so remember to check the strings you put in there.

PHP Regular Expressions

What is a Regular Expression?

A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.

A regular expression can be a single character, or a more complicated pattern.

Regular expressions can be used to perform all types of text search and text replace operations.

Syntax

In PHP, regular expressions are strings composed of delimiters, a pattern and optional modifiers.

```
$exp = "/AIT/i";
```

In the example above, / is the delimiter, *AIT* is the pattern that is being searched for, and *i* is a modifier that makes the search case-insensitive.

The delimiter can be any character that is not a letter, number, backslash or space. The most common delimiter is the forward slash (/), but when your pattern contains forward slashes it is convenient to choose other delimiters such as # or ~.

Regular Expression Functions

PHP provides a variety of functions that allow you to use regular expressions. The `preg_match()`, `preg_match_all()` and `preg_replace()` functions are some of the most commonly used ones:

Function	Description
<code>preg_match()</code>	Returns 1 if the pattern was found in the string and 0 if not
<code>preg_match_all()</code>	Returns the number of times the pattern was found in the string, which may also be 0

<code>preg_replace()</code>	Returns a new string where matched patterns have been replaced with another string
-----------------------------	--

Using `preg_match()`

The `preg_match()` function will tell you whether a string contains matches of a pattern.

Example

Use a regular expression to do a case-insensitive search for "AIT" in a string:

```
<!DOCTYPE html>
<html>
<body>

<?php
$str = "Visit AIT";
$pattern = "/AIT/i";
echo preg_match($pattern, $str);
?>
</body>
</html>
1
```

Using `preg_match_all()`

The `preg_match_all()` function will tell you how many matches were found for a pattern in a string.

Example

Use a regular expression to do a case-insensitive count of the number of occurrences of "ain" in a string:

```
<?php
$str = "The rain in SPAIN falls mainly on the plains.";
$pattern = "/ain/i";
```

```
echo preg_match_all($pattern, $str); // Outputs 4
?>
```

Using preg_replace()

The preg_replace() function will replace all of the matches of the pattern in a string with another string.

Example

Use a case-insensitive regular expression to replace Microsoft with AIT in a string:

```
<?php
$str = "Visit Microsoft!";
$pattern = "/microsoft/i";
echo preg_replace($pattern, "AIT", $str); // Outputs "Visit AIT!"
?>
```

Regular Expression Modifiers

Modifiers can change how a search is performed.

Modifier	Description
i	Performs a case-insensitive search
m	Performs a multiline search (patterns that search for the beginning or end of a string will match the beginning or end of each line)
u	Enables correct matching of UTF-8 encoded patterns

Regular Expression Patterns

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find one character from the options between the brackets
[^abc]	Find any character NOT between the brackets

[0-9]	Find one character from the range 0 to 9
-------	--

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
	Find a match for any one of the patterns separated by as in: cat dog fish
.	Find just one instance of any character
^	Finds a match as the beginning of a string as in: ^Hello
\$	Finds a match at the end of the string as in: World\$
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers

Quantifiers define quantities:

Quantifier	Description
n+	Matches any string that contains at least one <i>n</i>
n*	Matches any string that contains zero or more occurrences of <i>n</i>
n?	Matches any string that contains zero or one occurrences of <i>n</i>

n{x}	Matches any string that contains a sequence of X n's
n{x,y}	Matches any string that contains a sequence of X to Y n's
n{x,}	Matches any string that contains a sequence of at least X n's

Note: If your expression needs to search for one of the special characters you can use a backslash (\) to escape them. For example, to search for one or more question marks you can use the following expression: `$pattern = '/\?+/';`

Grouping

You can use parentheses () to apply quantifiers to entire patterns. They also can be used to select parts of the pattern to be used as a match.

Example

Use grouping to search for the word "banana" by looking for *ba* followed by two instances of *na*:

```
<?php
$str = "Apples and bananas.";
$pattern = "/ba(na){2}/i";
echo preg_match($pattern, $str); // Outputs 1
?>
```

Exception Handling in PHP

An exception is an unexpected program result that can be handled by the program itself. Exception Handling in PHP is almost similar to exception handling in all programming languages.

PHP provides the following specialized keywords for this purpose.

- **try:** It represents a block of code in which an exception can arise.
- **catch:** It represents a block of code that will be executed when a particular exception has been thrown.
- **throw:** It is used to throw an exception. It is also used to list the exceptions that a function throws, but doesn't handle itself.

- **finally:** It is used in place of catch block or after catch block basically it is put for cleanup activity in PHP code.

Why Exception Handling in PHP ?

Following are the main advantages of exception handling over error handling

- **Separation of error handling code from normal code:** In traditional error handling code there is always if else block to handle errors. These conditions and code to handle errors got mixed so that becomes unreadable. With try Catch block code becomes readable.
- **Grouping of error types:** In PHP both basic types and objects can be thrown as exceptions. It can create a hierarchy of exception objects, group exceptions in namespaces or classes, categorize them according to types.

Exception handling in PHP:

Following code explains the flow of normal try catch block in PHP:

```
<?php

// PHP Program to illustrate normal
// try catch block code
function demo($var) {
    echo " Before try block";
    try {
        echo "\n Inside try block";

        // If var is zero then only if will be executed
        if($var == 0)
        {

            // If var is zero then only exception is thrown
```

```

        throw new Exception('Number is zero.');
```

```

        // This line will never be executed
        echo "\n After throw (It will never be executed)";
    }
}

// Catch block will be executed only
// When Exception has been thrown by try block
catch(Exception $e) {
    echo "\n Exception Caught", $e->getMessage();
}

// This line will be executed whether
// Exception has been thrown or not
echo "\n After catch (will be always executed)";
}

// Exception will not be raised
demo(5);

// Exception will be raised here
demo(0);
?>
```

Output:

Before try block

Inside try block

After catch (will be always executed)

Before try block

Inside try block

Exception CaughtNumber is zero.

After catch (will be always executed)

Following code explains the flow of normal try catch and finally block in PHP

```
<?php

// PHP Program to illustrate normal
// try catch block code
function demo($var) {

    echo " Before try block";
    try {
        echo "\n Inside try block";

        // If var is zero then only if will be executed
        if($var == 0) {

            // If var is zero then only exception is thrown
            throw new Exception('Number is zero.');
```

```
            // This line will never be executed
            echo "\n After throw it will never be executed";
        }
    }

    // Catch block will be executed only
    // When Exception has been thrown by try block
    catch(Exception $e) {
        echo "\n Exception Caught" . $e->getMessage();
    }
    finally {
        echo "\n Here cleanup activity will be done";
    }

    // This line will be executed whether
    // Exception has been thrown or not
    echo "\n After catch it will be always executed";
}

// Exception will not be raised
demo(5);

// Exception will be raised here
demo(0);
```

?>

Output:

Before try block

Inside try block

Here cleanup activity will be done

After catch (will be always executed)

Before try block

Inside try block

Exception CaughtNumber is zero.

Here cleanup activity will be done

After catch (will be always executed)

Using Custom Exception Class

```
<?php
class myException extends Exception {
    function get_Message() {

        // Error message
        $errorMsg = 'Error on line '.$this->getLine().
                    ' in '.$this->getFile()
                    .' is number zero';
        return $errorMsg;
    }
}

function demo($a) {
    try {
```

```

        // Check if
        if($a == 0) {
            throw new myException($a);
        }
    }

    catch (myException $e) {

        // Display custom message
        echo $e->get_Message();
    }
}

// This will not generate any exception
demo(5);

// It will cause an exception
demo(0);
?>

```

Output:

Error on line 20 in /home/45ae8dc582d50df2790517e912980806.php0 is number zero

Set a Top Level Exception Handler: The `set_exception_handler()` function set all user defined functions to all uncaught exceptions.

```

<?php
// PHP Program to illustrate normal
// try catch block code

// Function for Uncaught Exception
function myException($exception) {

    // Details of Uncaught Exception
    echo "\nException: " . $exception->getMessage();
}

// Set Uncaught Exception handler
set_exception_handler('myException');
function demo($var) {

```

```

echo " Before try block";
try {
    echo "\n Inside try block";

    // If var is zero then only if will be executed
    if($var == 0)
    {

        // If var is zero then only exception is thrown
        throw new Exception('Number is zero.');
```

// This line will never be executed

```
        echo "\n After throw (it will never be executed)";
    }
}

// Catch block will be executed only
// When Exception has been thrown by try block
catch(Exception $e) {
    echo "\n Exception Caught", $e->getMessage();
}

// This line will be executed whether
// Exception has been thrown or not
echo "\n After catch (will be always executed)";

if($var < 0) {

    // Uncaught Exception
    throw new Exception('Uncaught Exception occurred');
}
}

// Exception will not be raised
demo(5);

// Exception will be raised here
demo(0);

// Uncaught Exception
demo (-3);
?>
```

Output:

Before try block

Inside try block

After catch (will be always executed)

Before try block

Inside try block

Exception CaughtNumber is zero.

After catch (will be always executed)

Before try block

Inside try block

After catch (will be always executed)

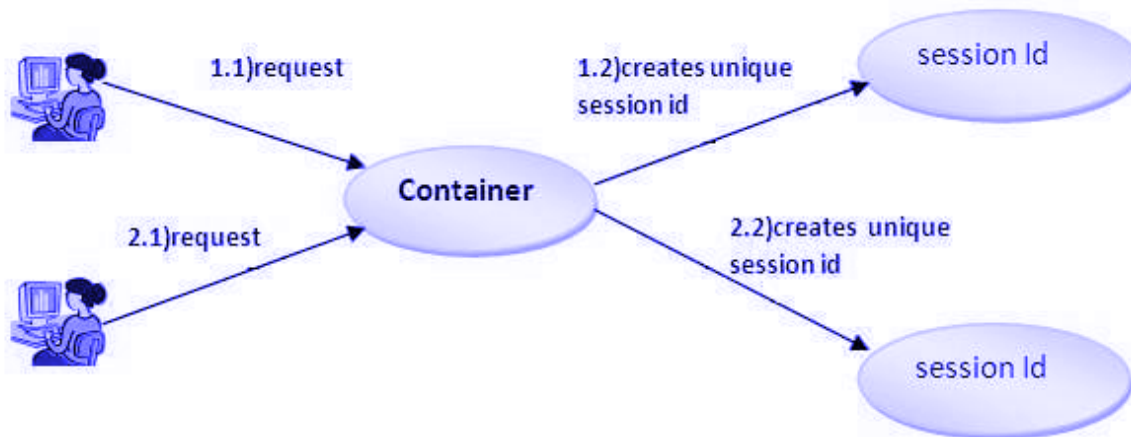
Exception: Uncaught Exception occurred

PHP Session

A PHP session is used to store and pass information from one page to another temporarily (until the user closes the website).

PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

PHP session creates a unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



PHP session_start() function

PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns an existing session if the session is created already. If a session is not available, it creates and returns a new session.

Syntax

1. bool session_start (void)

Example

1. session_start();

PHP \$_SESSION

PHP \$_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

Example: Store information

1. \$_SESSION["user"] = "Sachin";

Example: Get information

1. `echo $_SESSION["user"];`

PHP Session Example

File: session1.php

```
<?php
session_start();
?>
<html>
<body>
<?php
$_SESSION["user"] = "Sachin";
echo "Session information are set successfully.<br/>";
?>
<a href="session2.php">Visit next page</a>
</body>
</html>
```

File: session2.php

```
<?php
session_start();
?>
<html>
<body>
<?php
echo "User is: ".$_SESSION["user"];
?>
</body>
</html>
```

PHP Session Counter Example

File: sessioncounter.php

```
<?php
    session_start();
```

```

    if (!isset($_SESSION['counter'])) {
        $_SESSION['counter'] = 1;
    } else {
        $_SESSION['counter']++;
    }
    echo ("Page Views: ".$_SESSION['counter']);
?>

```

PHP Destroying Session

PHP `session_destroy()` function is used to destroy all session variables completely.

File: session3.php

```

<?php
session_start();
session_destroy();
?>

```

HTML `<input type="hidden">`

Example

Define a hidden field:

```

<!DOCTYPE html>
<html>
<body>

<h1>A Hidden Field (look in source code)</h1>

<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="hidden" id="custId" name="custId" value="3487">
  <input type="submit" value="Submit">
</form>
<p><strong>Note:</strong> The hidden field is not shown to the user,
but the data is sent when the form is submitted.</p>
</body>
</html>

```

Definition and Usage

The `<input type="hidden">` defines a hidden input field.

A hidden field let web developers include data that cannot be seen or modified by users when a form is submitted.

A hidden field often stores what database record that needs to be updated when the form is submitted.

Note: While the value is not displayed to the user in the page's content, it is visible (and can be edited) using any browser's developer tools or "View Source" functionality. Do not use hidden inputs as a form of security!

Syntax

```
<input type="hidden">
```

PHP Cookies

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the `setcookie()` function.

Syntax

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available on the entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable \$_COOKIE). We also use the isset() function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30),
"/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Cookie named 'user' is not set!

Note: You might have to reload the page to see the value of the cookie.

Note: The setcookie() function must appear BEFORE the <html> tag.

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30),
"/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Cookie named 'user' is not set!

Note: You might have to reload the page to see the new value of the cookie.

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example

```
<!DOCTYPE html>
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
```

```
echo "Cookie 'user' is deleted.";
?>
```

```
</body>
</html>
```

Cookie 'user' is deleted.

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the `setcookie()` function, then count the `$_COOKIE` array variable:

Example

```
<!DOCTYPE html>
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

Cookies are enabled.

Database programming with PHP and MySQL

What is a Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching, and replicating the data it holds.

Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage a huge volume of data. This is called a relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

A Relational DataBase Management System (RDBMS) is a software that –

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

RDBMS Terminology

Before we proceed to explain the MySQL database system, let us revise a few definitions related to the database.

- Database – A database is a collection of tables, with related data.
- Table – A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- Column – One column (data element) contains data of one and the same kind, for example the column postcode.
- Row – A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
- Redundancy – Storing data twice, redundantly to make the system faster.

- Primary Key – A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row.
- Foreign Key – A foreign key is the linking pin between two tables.
- Compound Key – A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- Index – An index in a database resembles an index at the back of a book.
- Referential Integrity – Referential Integrity makes sure that a foreign key value always points to an existing row.

MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons –

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

All downloads for MySQL are located at MySQL Downloads. Pick the version number of MySQL Community Server which is required along with the platform you will be running it on.

Installing MySQL on Linux/UNIX

The recommended way to install MySQL on a Linux system is via RPM. MySQL AB makes the following RPMs available for download on its website –

- MySQL – The MySQL database server manages the databases and tables, controls user access and processes the SQL queries.
- MySQL-client – MySQL client programs, which make it possible to connect to and interact with the server.
- MySQL-devel – Libraries and header files that come in handy when compiling other programs that use MySQL.
- MySQL-shared – Shared libraries for the MySQL client.
- MySQL-bench – Benchmark and performance testing tools for the MySQL database server.

The MySQL RPMs listed here are all built on a SuSE Linux system, but they will usually work on other Linux variants with no difficulty.

Now, you will need to adhere to the steps given below, to proceed with the installation –

- Login to the system using the root user.
- Switch to the directory containing the RPMs.
- Install the MySQL database server by executing the following command.
Remember to replace the filename in italics with the file name of your RPM.

```
[root@host]# rpm -i MySQL-5.0.9-0.i386.rpm
```

The above command takes care of installing the MySQL server, creating a user of MySQL, creating necessary configuration and starting the MySQL server automatically.

You can find all the MySQL related binaries in /usr/bin and /usr/sbin. All the tables and databases will be created in the /var/lib/mysql directory.

The following code box has an optional but recommended step to install the remaining RPMs in the same manner –

```
[root@host]# rpm -i MySQL-client-5.0.9-0.i386.rpm
```

```
[root@host]# rpm -i MySQL-devel-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-shared-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-bench-5.0.9-0.i386.rpm
```

Installing MySQL on Windows

The default installation on any version of Windows is now much easier than it used to be, as MySQL now comes neatly packaged with an installer. Simply download the installer package, unzip it anywhere and run the setup.exe file.

The default installer setup.exe will walk you through the trivial process and by default will install everything under C:\mysql.

Test the server by firing it up from the command prompt the first time. Go to the location of the mysqld server which is probably C:\mysql\bin, and type –

```
mysqld.exe --console
```

NOTE – If you are on NT, then you will have to use mysqld-nt.exe instead of mysqld.exe

If all went well, you will see some messages about startup and InnoDB. If not, you may have a permissions issue. Make sure that the directory that holds your data is accessible to whatever user (probably MySQL) the database processes run under.

MySQL will not add itself to the start menu, and there is no particularly nice GUI way to stop the server either. Therefore, if you tend to start the server by double clicking the mysqld executable, you should remember to halt the process by hand by using mysqladmin, Task List, Task Manager, or other Windows-specific means.

Verifying MySQL Installation

After MySQL has been successfully installed, the base tables have been initialized and the server has been started: you can verify that everything is working as it should be via some simple tests.

Use the mysqladmin Utility to Obtain Server Status

Use mysqladmin binary to check the server version. This binary would be available in /usr/bin on linux and in C:\mysql\bin on windows.

```
[root@host]# mysqladmin --version
```

It will produce the following result on Linux. It may vary depending on your installation –

```
mysqladmin Ver 8.23 Distrib 5.0.9-0, for redhat-linux-gnu on i386
```

If you do not get such a message, then there may be some problem in your installation and you would need some help to fix it.

Execute simple SQL commands using the MySQL Client

You can connect to your MySQL server through the MySQL client and by using the mysql command. At this moment, you do not need to give any password as by default it will be set as blank.

You can just use following command –

```
[root@host]# mysql
```

It should be rewarded with a mysql> prompt. Now, you are connected to the MySQL server and you can execute all the SQL commands at the mysql> prompt as follows –

```
mysql> SHOW DATABASES;
```

```
+-----+  
| Database |  
+-----+  
| mysql |  
| test |  
+-----+
```

```
2 rows in set (0.13 sec)
```

Post-installation Steps

MySQL ships with a blank password for the root MySQL user. As soon as you have successfully installed the database and the client, you need to set a root password as given in the following code block –

```
[root@host]# mysqladmin -u root password "new_password";
```

Now to make a connection to your MySQL server, you would have to use the following command –

```
[root@host]# mysql -u root -p  
Enter password:*****
```

UNIX users will also want to put your MySQL directory in your PATH, so you won't have to keep typing out the full path everytime you want to use the command-line client.

For bash, it would be something like –

```
export PATH = $PATH:/usr/bin:/usr/sbin
```

Running MySQL at Boot Time

If you want to run the MySQL server at boot time, then make sure you have the following entry in the /etc/rc.local file.

```
/etc/init.d/mysqld start
```

Also, you should have the mysqld binary in the /etc/init.d/ directory.

MySQL - Administration

Running and Shutting down MySQL Server

First check if your MySQL server is running or not. You can use the following command to check it –

```
ps -ef | grep mysqld
```

If your MySQL is running, then you will see the mysqld process listed out in your result. If server is not running, then you can start it by using the following command –

```
root@host# cd /usr/bin
```

```
./safe_mysqld &
```

Now, if you want to shut down an already running MySQL server, then you can do it by using the following command –

```
root@host# cd /usr/bin
./mysqladmin -u root -p shutdown
Enter password: *****
```

Setting Up a MySQL User Account

For adding a new user to MySQL, you just need to add a new entry to the user table in the database mysql.

The following program is an example of adding a new user guest with SELECT, INSERT and UPDATE privileges with the password guest123; the SQL query is –

```
root@host# mysql -u root -p
Enter password:*****
mysql> use mysql;
Database changed

mysql> INSERT INTO user
  (host, user, password,
  select_priv, insert_priv, update_priv)
  VALUES ('localhost', 'guest',
  PASSWORD('guest123'), 'Y', 'Y', 'Y');
Query OK, 1 row affected (0.20 sec)
```

```
mysql> FLUSH PRIVILEGES;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> SELECT host, user, password FROM user WHERE user = 'guest';
+-----+-----+-----+
| host | user | password |
+-----+-----+-----+
```

```
| localhost | guest | 6f8c114b58f2ce9e |
```

```
+-----+-----+-----+
```

```
1 row in set (0.00 sec)
```

When adding a new user, remember to encrypt the new password using the PASSWORD() function provided by MySQL. As you can see in the above example, the password mypass is encrypted to 6f8c114b58f2ce9e.

Notice the FLUSH PRIVILEGES statement. This tells the server to reload the grant tables. If you don't use it, then you won't be able to connect to MySQL using the new user account at least until the server is rebooted.

You can also specify other privileges to a new user by setting the values of following columns in the user table to 'Y' when executing the INSERT query or you can update them later using an UPDATE query.

- Select_priv
- Insert_priv
- Update_priv
- Delete_priv
- Create_priv
- Drop_priv
- Reload_priv
- Shutdown_priv
- Process_priv
- File_priv
- Grant_priv
- References_priv
- Index_priv
- Alter_priv

Another way of adding a user account is by using GRANT SQL command. The following example will add user zara with password zara123 for a particular database, which is named as TUTORIALS.

```
root@host# mysql -u root -p password;
```

```
Enter password:*****
```



```
mysql> use mysql;
Database changed
```

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
-> ON TUTORIALS.*
-> TO 'zara'@'localhost'
-> IDENTIFIED BY 'zara123';
```

This will also create an entry in the MySQL database table called as user.

NOTE – MySQL does not terminate a command until you give a semicolon (;) at the end of the SQL command.

The /etc/my.cnf File Configuration

In most of the cases, you should not touch this file. By default, it will have the following entries –

```
[mysqld]
datadir = /var/lib/mysql
socket = /var/lib/mysql/mysql.sock

[mysql.server]
user = mysql
basedir = /var/lib

[safe_mysqld]
err-log = /var/log/mysqld.log
pid-file = /var/run/mysqld/mysqld.pid
```

Here, you can specify a different directory for the error log, otherwise you should not change any entry in this table.

Administrative MySQL Command

Here is the list of the important MySQL commands, which you will use time to time to work with MySQL database –

- USE Databasename – This will be used to select a database in the MySQL work area.
- SHOW DATABASES – Lists out the databases that are accessible by the MySQL DBMS.
- SHOW TABLES – Shows the tables in the database once a database has been selected with the use command.
- SHOW COLUMNS FROM *tablename*: Shows the attributes, types of attributes, key information, whether NULL is permitted, defaults, and other information for a table.
- SHOW INDEX FROM *tablename* – Presents the details of all indexes on the table, including the PRIMARY KEY.
- SHOW TABLE STATUS LIKE *table name* \G – Reports details of the MySQL DBMS performance and statistics.

MySQL - PHP Syntax

MySQL works very well in combination with various programming languages like PERL, C, C++, JAVA and PHP. Out of these languages, PHP is the most popular one because of its web application development capabilities.

PHP provides various functions to access the MySQL database and to manipulate the data records inside the MySQL database. You would require to call the PHP functions in the same way you call any other PHP function.

The PHP functions for use with MySQL have the following general format –

```
mysql_<function>(value,value,...);
```

The second part of the function name is specific to the function, usually a word that describes what the function does. The following are two of the functions, which we will use in our tutorial –

```
mysqli_connect($connect);
mysqli_query($connect,"SQL statement");
```

The following example shows a generic syntax of PHP to call any MySQL function.

```
<html>
  <head>
    <title>PHP with MySQL</title>
  </head>

  <body>
    <?php
      $retval = mysql_function(value, [value,...]);
      if( !$retval ) {
        die ( "Error: a related error message" );
      }
      // Otherwise MySQL or PHP Statements
    ?>
  </body>
</html>
```

MySQL - Connection

MySQL Connection Using MySQL Binary

You can establish the MySQL database using the mysql binary at the command prompt.

Example

Here is a simple example to connect to the MySQL server from the command prompt –

```
[root@host]# mysql -u root -p
Enter password:*****
```

This will give you the mysql> command prompt where you will be able to execute any SQL command. Following is the result of above command –

The following code block shows the result of above code –

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2854760 to server version: 5.0.9
```

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

In the above example, we have used root as a user but you can use any other user as well. Any user will be able to perform all the SQL operations, which are allowed to that user.

You can disconnect from the MySQL database any time using the exit command at mysql> prompt.

```
mysql> exit
```

Bye

MySQL Connection Using PHP Script

PHP provides `mysql_connect()` function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success or FALSE on failure.

Syntax

```
connection mysql_connect(server,user,password,new_link,client_flag);
```

Sr.No.	Parameter & Description
1	server Optional – The host name running the database server. If not specified, then the default value will be localhost:3306.
2	user Optional – The username accessing the database. If not specified, then the default will be the name of the user that owns the server process.

3	<p>passwd</p> <p>Optional – The password of the user accessing the database. If not specified, then the default will be an empty password.</p>
4	<p>new_link</p> <p>Optional – If a second call is made to <code>mysql_connect()</code> with the same arguments, no new connection will be established; instead, the identifier of the already opened connection will be returned.</p>
5	<p>client_flags</p> <p>Optional – A combination of the following constants –</p> <ul style="list-style-type: none"> ● <code>MYSQL_CLIENT_SSL</code> – Use SSL encryption. ● <code>MYSQL_CLIENT_COMPRESS</code> – Use compression protocol. ● <code>MYSQL_CLIENT_IGNORE_SPACE</code> – Allow space after function names. ● <code>MYSQL_CLIENT_INTERACTIVE</code> – Allow interactive timeout seconds of inactivity before closing the connection.

You can disconnect from the MySQL database anytime using another PHP function `mysql_close()`. This function takes a single parameter, which is a connection returned by the `mysql_connect()` function.

Syntax

```
bool mysql_close ( resource $link_identifier );
```

If a resource is not specified, then the last opened database is closed. This function returns true if it closes the connection successfully otherwise it returns false.

Example

Try the following example to connect to a MySQL server –

```
<html>
  <head>
    <title>Connecting MySQL Server</title>
  </head>
  <body>
    <?php
      $dbhost = 'localhost:3306';
      $dbuser = 'guest';
      $dbpass = 'guest123';
      $conn = mysql_connect($dbhost, $dbuser, $dbpass);

      if(! $conn ) {
        die('Could not connect: ' . mysql_error());
      }
      echo 'Connected successfully';
      mysql_close($conn);
    ?>
  </body>
</html>
```

MySQL - Create Database

Create Database Using mysqladmin

You would need special privileges to create or to delete a MySQL database. So assuming you have access to the root user, you can create any database using the mysql mysqladmin binary.

Example

Here is a simple example to create a database called TUTORIALS –

```
[root@host]# mysqladmin -u root -p create TUTORIALS
Enter password:*****
```

This will create a MySQL database called TUTORIALS.

Create a Database using PHP Script

PHP uses `mysql_query` function to create or delete a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax

```
bool mysql_query( sql, connection );
```

Sr.No.	Parameter & Description
1	sql Required - SQL query to create or delete a MySQL database
2	connection Optional - if not specified, then the last opened connection by <code>mysql_connect</code> will be used.

Example

The following example to create a database -

```
<html>
  <head>
    <title>Creating MySQL Database</title>
  </head>

  <body>
    <?php
      $dbhost = 'localhost:3036';
      $dbuser = 'root';
      $dbpass = 'rootpassword';
      $conn = mysql_connect($dbhost, $dbuser, $dbpass);

      if(! $conn ) {
```

```

        die('Could not connect: ' . mysql_error());
    }
    echo 'Connected successfully<br />';
    $sql = 'CREATE DATABASE TUTORIALS';
    $retval = mysql_query( $sql, $conn );

    if(! $retval ) {
        die('Could not create database: ' . mysql_error());
    }
    echo "Database TUTORIALS created successfully\n";
    mysql_close($conn);
?>
</body>
</html>

```

Drop MySQL Database

Drop a Database using mysqladmin

You would need special privileges to create or to delete a MySQL database. So, assuming you have access to the root user, you can create any database using the mysql mysqladmin binary.

Be careful while deleting any database because you will lose all the data available in your database.

```

[root@host]# mysqladmin -u root -p drop TUTORIALS
Enter password:*****

```

This will give you a warning and it will confirm if you really want to delete this database or not.

Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.

```

Do you really want to drop the 'TUTORIALS' database [y/N] y
Database "TUTORIALS" dropped

```


Drop Database using PHP Script

PHP uses `mysql_query` function to create or delete a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax

```
bool mysql_query( sql, connection );
```

Sr.No	Parameter & Description
1	<p>sql</p> <p>Required – SQL query to create or delete a MySQL database</p>
2	<p>connection</p> <p>Optional – if not specified, then the last opened connection by <code>mysql_connect</code> will be used.</p>

Example

```
<html>
  <head>
    <title>Deleting MySQL Database</title>
  </head>

  <body>
    <?php
      $dbhost = 'localhost:3036';
      $dbuser = 'root';
      $dbpass = 'rootpassword';
```

```

$conn = mysql_connect($dbhost, $dbuser, $dbpass);

if(! $conn ) {
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br />';
$sql = 'DROP DATABASE TUTORIALS';
$retval = mysql_query( $sql, $conn );

if(! $retval ) {
    die('Could not delete database: ' . mysql_error());
}
echo "Database TUTORIALS deleted successfully\n";
mysql_close($conn);
?>
</body>
</html>

```

WARNING – While deleting a database using the PHP script, it does not prompt you for any confirmation. So be careful while deleting a MySQL database.

Selecting MySQL Database

Once you get connected with the MySQL server, it is required to select a database to work with. This is because there might be more than one database available with the MySQL Server.

Selecting MySQL Database from the Command Prompt

It is very simple to select a database from the mysql> prompt. You can use the SQL command used to select a database.

Example

Here is an example to select a database called TUTORIALS –

```

[root@host]# mysql -u root -p
Enter password:*****
mysql> use TUTORIALS;
Database changed

```

mysql>

Now, you have selected the TUTORIALS database and all the subsequent operations will be performed on the TUTORIALS database.

NOTE – All the database names, table names, table fields names are case sensitive. So you would have to use the proper names while giving any SQL command.

Selecting a MySQL Database Using PHP Script

PHP provides a function `mysql_select_db` to select a database. It returns TRUE on success or FALSE on failure.

Syntax

```
bool mysql_select_db( db_name, connection );
```

Sr.No.	Parameter & Description
1	<code>db_name</code> Required – MySQL Database name to be selected
2	<code>connection</code> Optional – if not specified, then the last opened connection by <code>mysql_connect</code> will be used.

Example

```
<html>  
  <head>  
    <title>Selecting MySQL Database</title>
```

```

</head>

<body>
  <?php
    $dbhost = 'localhost:3036';
    $dbuser = 'guest';
    $dbpass = 'guest123';
    $conn = mysql_connect($dbhost, $dbuser, $dbpass);
    if(! $conn ) {
        die('Could not connect: ' . mysql_error());
    }
    echo 'Connected successfully';
    mysql_select_db( 'TUTORIALS' );
    mysql_close($conn);
  ?>
</body>
</html>

```

PHP MySQL Create Table

A database table has its own unique name and consists of columns and rows.

Create a MySQL Table Using MySQLi and PDO

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

```

CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)

```

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

The following examples shows how to create the table in PHP:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";
```

```

if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

$conn->close();
?>

```

Example (MySQLi Procedural)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

Example (PDO)

```

<?php

```

```

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // sql to create table
    $sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";

    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Table MyGuests created successfully";
} catch(PDOException $e) {
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>

```

PHP MySQL Insert Data

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

To learn more about SQL, please visit our [SQL tutorial](#).

In the previous chapter we created an empty table named "MyGuests" with five columns: "id", "firstname", "lastname", "email" and "reg_date". Now, let us fill the table with data.

Note: If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP with default update of current_timestamp (like the "reg_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

The following examples add a new record to the "MyGuests" table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com)";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```


PHP MySQL Select Data

Select Data From a MySQL Database

The SELECT statement is used to select data from one or more tables:

```
SELECT column_name(s) FROM table_name
```

or we can use the * character to select ALL columns from a table:

```
SELECT * FROM table_name
```

Select Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
        $row["lastname"]. "<br>";
    }
} else {
```

```
    echo "0 results";  
}  
$conn->close();  
?>
```

Code lines to explain from the example above:

First, we set up an SQL query that selects the id, firstname and lastname columns from the MyGuests table. The next line of code runs the query and puts the resulting data into a variable called \$result.

Then, the function num_rows() checks if there are more than zero rows returned.

If there are more than zero rows returned, the function fetch_assoc() puts all the results into an associative array that we can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

PHP MySQL Use The WHERE Clause

Select and Filter Data From a MySQL Database

The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

```
SELECT column_name(s) FROM table_name WHERE column_name operator value
```

Select and Filter Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table where the last name is "Doe", and displays it on the page:

Example (MySQLi Object-oriented)

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";
```

```

$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE
lastname='Doe'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>

```

Code lines to explain from the example above:

First, we set up the SQL query that selects the id, firstname and lastname columns from the MyGuests table where the last name is "Doe". The next line of code runs the query and puts the resulting data into a variable called \$result.

Then, the function num_rows() checks if there are more than zero rows returned.

If there are more than zero rows returned, the function fetch_assoc() puts all the results into an associative array that we can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

PHP MySQL Update Data

The UPDATE statement is used to update existing records in a table:

```
UPDATE table_name
```

SET column1=value, column2=value2,...

WHERE some_column=some_value

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

Let's look at the "MyGuests" table:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

The following examples update the record with id=2 in the "MyGuests" table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```

PHP MySQL Delete Data

The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
```

```
WHERE some_column = some_value
```

Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Let's look at the "MyGuests" table:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

The following examples delete the record with id=3 in the "MyGuests" table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
```

```
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

PHP MySQL Prepared Statements

Prepared statements are very useful against SQL injections.

Prepared Statements and Bound Parameters

A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.

Prepared statements basically work like this:

1. **Prepare:** An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?)
2. The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
3. **Execute:** At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

Compared to executing SQL statements directly, prepared statements have three main advantages:

- Prepared statements reduce parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
- Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query
- Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be

correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

Prepared Statements in MySQLi

The following example uses prepared statements and bound parameters in MySQLi:

Example (MySQLi with Prepared Statements)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname,
email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";
```

```
$stmt->close();  
$conn->close();  
?>
```

Code lines to explain from the example above:

```
"INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)"
```

In our SQL, we insert a question mark (?) where we want to substitute in an integer, string, double or blob value.

Then, have a look at the `bind_param()` function:

```
$stmt->bind_param("sss", $firstname, $lastname, $email);
```

This function binds the parameters to the SQL query and tells the database what the parameters are. The "sss" argument lists the types of data that the parameters are. The s character tells mysql that the parameter is a string.

The argument may be one of four types:

- i - integer
- d - double
- s - string
- b - BLOB

We must have one of these for each parameter.

By telling mysql what type of data to expect, we minimize the risk of SQL injections.

Note: If we want to insert any data from external sources (like user input), it is very important that the data is sanitized and validated.

Prepared Statements in PDO

The following example uses prepared statements and bound parameters in PDO:

Example (PDO with Prepared Statements)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // prepare sql and bind parameters
    $stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname,
email)
VALUES (:firstname, :lastname, :email)");
    $stmt->bindParam(':firstname', $firstname);
    $stmt->bindParam(':lastname', $lastname);
    $stmt->bindParam(':email', $email);

    // insert a row
    $firstname = "John";
    $lastname = "Doe";
    $email = "john@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Mary";
    $lastname = "Moe";
    $email = "mary@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Julie";
    $lastname = "Dooley";
    $email = "julie@example.com";
    $stmt->execute();

    echo "New records created successfully";
} catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
?>
```

Advanced Web Programming concepts

What is AJAX?

AJAX = Asynchronous JavaScript and XML.

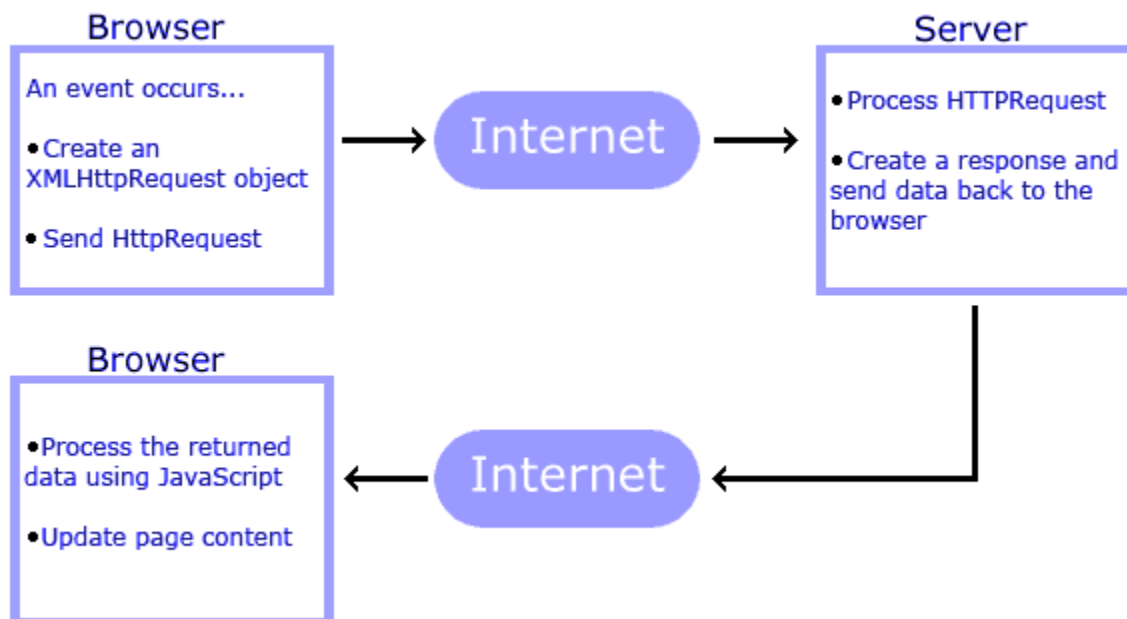
AJAX is a technique for creating fast and dynamic web pages.

AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.

Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

How AJAX Works



AJAX is Based on Internet Standards

AJAX is based on internet standards, and uses a combination of:

- XMLHttpRequest object (to exchange data asynchronously with a server)
- JavaScript/DOM (to display/interact with the information)
- CSS (to style the data)
- XML (often used as the format for transferring data)

AJAX applications are browser- and platform-independent!

Understanding XMLHttpRequest

An object of XMLHttpRequest is used for asynchronous communication between client and server.

It performs following operations:

1. Sends data from the client in the background
2. Receives the data from the server
3. Updates the web page without reloading it.

Properties of XMLHttpRequest object

The common properties of XMLHttpRequest object are as follows:

Property	Description
onReadyStateChange	It is called whenever the readystate attribute changes. It must not be used with synchronous requests.

readyState	<p>represents the state of the request. It ranges from 0 to 4.</p> <p>0 UNOPENED open() is not called.</p> <p>1 OPENED open is called but send() is not called.</p> <p>2 HEADERS_RECEIVED send() is called, and headers and status are available.</p> <p>3 LOADING Downloading data; responseText holds the data.</p> <p>4 DONE The operation is completed fully.</p>
responseText	returns response as text.
responseXML	returns response as XML

Methods of XMLHttpRequest object

The important methods of XMLHttpRequest object are as follows:

Method	Description
void open(method, URL)	opens the request specifying get or post method and url.
void open(method, URL, async)	same as above but specifies asynchronous or not.
void open(method, URL, async, username, password)	same as above but specifies username and password.

void send()	send get request.
void send(string)	send a post request.
setRequestHeader(header,value)	it adds request headers.

How AJAX Works?

1. User sends a request from the UI and a javascript call goes to XMLHttpRequest object.
2. HTTP Request is sent to the server by XMLHttpRequest object.
3. Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.
4. Data is retrieved.
5. Server sends XML data or JSON data to the XMLHttpRequest callback function.
6. HTML and CSS data is displayed on the browser.

jQuery

jQuery is a fast, small, cross-platform and feature-rich JavaScript library. It is designed to simplify the client-side scripting of HTML. It makes things like HTML document traversal and manipulation, animation, event handling, and AJAX very simple with an easy-to-use API that works on a lot of different type of browsers.

The main purpose of jQuery is to provide an easy way to use JavaScript on your website to make it more interactive and attractive. It is also used to add animation.

What is jQuery

jQuery is a small, light-weight and fast JavaScript library. It is cross-platform and supports different types of browsers. It is also referred as "write less do more" because it takes a lot of common tasks that requires many lines of JavaScript code to accomplish, and binds them into methods that can be called with a single line of code whenever needed. It is also very useful to simplify a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

- jQuery is a small, fast and lightweight JavaScript library.
- jQuery is platform-independent.
- jQuery means "write less do more".
- jQuery simplifies AJAX call and DOM manipulation.

jQuery Features

Following are the important features of jQuery.

- HTML manipulation
- DOM manipulation
- DOM element selection
- CSS manipulation
- Effects and Animations
- Utilities
- AJAX
- HTML event methods
- JSON Parsing
- Extensibility through plug-ins

Why jQuery is required

Sometimes, a question can arise: what is the need of jQuery or what difference does it make on bringing jQuery instead of AJAX/ JavaScript? If jQuery is the replacement of AJAX and JavaScript?

- It is very fast and extensible.

- It facilitates the users to write UI related function codes in minimum possible lines.
- It improves the performance of an application.
- Browser's compatible web applications can be developed.
- It uses mostly new features of new browsers.

So, you can say that out of the lot of JavaScript frameworks, jQuery is the most popular and the most extendable. Many of the biggest companies on the web use jQuery.

Some of these companies are:

- Microsoft
- Google
- IBM
- Netflix

jQuery Example

jQuery is developed by Google. To create the first jQuery example, you need to use JavaScript file for jQuery. You can download the jQuery file from jquery.com or use the absolute URL of jQuery file.

In this jQuery example, we are using the absolute URL of jQuery file. The jQuery example is written inside the script tag.

Let's see a simple example of jQuery.

File: firstjquery.html

```
<!DOCTYPE html>

<html>

<head>

  <title>First jQuery Example</title>
```

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.mi
n.js">
</script>
<script type="text/javascript" language="javascript">
$(document).ready(function() {
$("p").css("background-color", "cyan");
});
</script>
</head>
<body>
<p>The first paragraph is selected.</p>
<p>The second paragraph is selected.</p>
<p>The third paragraph is selected.</p>
</body>
</html>
```

Output:

```
The first paragraph is selected.
The second paragraph is selected.
The third paragraph is selected.
```

\$(document).ready() and \$()

The code inserted between `$(document).ready()` is executed only once when page is ready for JavaScript code to execute.

In place of `$(document).ready()`, you can use shorthand notation `$(())` only.

```
$(document).ready(function() {
```

```
    $("p").css("color", "red");
```

```
});
```

The above code is equivalent to this code.

```
$(function() {
```

```
    $("p").css("color", "red");
```

```
});
```

Let's see the full example of jQuery using shorthand notation `$(())`.

File: shortjquery.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>Second jQuery Example</title>
```

```
    <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.mi
n.js">
```

```
    </script>
```

```
    <script type="text/javascript" language="javascript">
```

```
        $(function() {
```

```
            $("p").css("color", "red");
```

```
});  
</script>  
</head>  
<body>  
<p>The first paragraph is selected.</p>  
<p>The second paragraph is selected.</p>  
<p>The third paragraph is selected.</p>  
</body>  
</html>
```

Output:

The first paragraph is selected.

The second paragraph is selected.

The third paragraph is selected.

```
function() { $("p").css("background-color", "cyan"); }
```

It changes the background-color of all <p> tags or paragraphs to cyan.

jQuery Selectors

jQuery Selectors are used to select and manipulate HTML elements. They are a very important part of jQuery library.

With jQuery selectors, you can find or select HTML elements based on their id, classes, attributes, types and much more from a DOM.

In simple words, you can say that selectors are used to select one or more HTML elements using jQuery and once the element is selected then you can perform various operation on that.

All jQuery selectors start with a dollar sign and parentheses e.g. \$(). It is known as the factory function.

The \$() factory function

Every jQuery selector start with this sign \$(). This sign is known as the factory function. It uses the three basic building blocks while selecting an element in a given document.

S. No	Select or	Description
1)	Tag Name:	It represents a tag name available in the DOM. For example: \$('p') selects all paragraphs in the document.
2)	Tag ID:	It represents a tag available with a specific ID in the DOM. For example: \$('#real-id') selects a specific element in the document that has an ID of real-id.
3)	Tag Class:	It represents a tag available with a specific class in the DOM. For example: \$('real-class') selects all elements in the document that have a class of real-class.

Let's take a simple example to see the use of Tag selector. This would select all the elements with a tag name

and the background color is set to "pink".

File: firstjquery.html

```
<!DOCTYPE html>
<html>
<head>
  <title>First jQuery Example</title>
  <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.mi
n.js">
  </script>
  <script type="text/javascript" language="javascript">
```

```
$(document).ready(function() {
  $("p").css("background-color", "pink");
});
</script>
</head>
<body>
<p>This is first paragraph.</p>
<p>This is second paragraph.</p>
<p>This is third paragraph.</p>
</body>
</html>
```

Test it Now

Output:

This is first paragraph.

This is second paragraph.

This is third paragraph.

Note: 1. All of the above discussed selectors can be used alone or with the combination of other selectors.

Note: 2. If you have any conflict with the use of dollar sign \$ in any JavaScript library then you can use jQuery() function instead of factory function \$(). The factory function \$() and the jQuery function is the same.

How to use Selectors

The jQuery selectors can be used single or with the combination of other selectors. They are required at every steps while using jQuery. They are used to select the exact element that you want from your HTML document.

S.N	Selector	Description
1)	Name:	It selects all elements that match with the given element name.
2)	#ID:	It selects a single element that matches with the given id.
3)	.Class:	It selects all elements that matches with the given class.
4)	Universal(*)	It selects all elements available in a DOM.
5)	Multiple Elements A,B,C	It selects the combined results of all the specified selectors A,B and C.

Different jQuery Selectors

Selector	Example	Description
*	\$("*")	It is used to select all elements.
#id	\$("#firstname")	It will select the element with id="firstname"
.class	\$(".primary")	It will select all elements with class="primary"
class,.class	\$(".primary,.secondary")	It will select all elements with the class "primary" or "secondary"

element	\$("p")	It will select all p elements.
e1,e2,e3	\$("h1,div,p")	It will select all h1, div, and p elements.
:first	\$("p:first")	This will select the first p element
:last	\$("p:last")	This will select the last p element
:even	\$("tr:even")	This will select all even tr elements
:odd	\$("tr:odd")	This will select all odd tr elements
:first-child	\$("p:first-child")	It will select all p elements that are the first child of their parent
:first-of-type	\$("p:first-of-type")	It will select all p elements that are the first p element of their parent
:last-child	\$("p:last-child")	It will select all p elements that are the last child of their parent
:last-of-type	\$("p:last-of-type")	It will select all p elements that are the last p element of their parent
:nth-child(n)	\$("p:nth-child(2)")	This will select all p elements that are the 2nd child of their parent

:nth-last-child(n)	\$("p:nth-last-child(2)")	This will select all p elements that are the 2nd child of their parent, counting from the last child
:nth-of-type(n)	\$("p:nth-of-type(2)")	It will select all p elements that are the 2nd p element of their parent
:nth-last-of-type(n)	\$("p:nth-last-of-type(2)")	This will select all p elements that are the 2nd p element of their parent, counting from the last child
:only-child	\$("p:only-child")	It will select all p elements that are the only child of their parent
:only-of-type	\$("p:only-of-type")	It will select all p elements that are the only child, of its type, of their parent
parent > child	\$("div > p")	It will select all p elements that are a direct child of a div element
parent descendant	\$("div p")	It will select all p elements that are descendants of a div element
element + next	\$("div + p")	It selects the p element that are next to each div elements

element ~ siblings	<code>\$("div ~ p")</code>	It selects all p elements that are siblings of a div element
<code>:eq(index)</code>	<code>\$("ul li:eq(3)")</code>	It will select the fourth element in a list (index starts at 0)
<code>:gt(no)</code>	<code>\$("ul li:gt(3)")</code>	Select the list elements with an index greater than 3
<code>:lt(no)</code>	<code>\$("ul li:lt(3)")</code>	Select the list elements with an index less than 3
<code>:not(selector)</code>	<code>\$("input:not(:empty)")</code>	Select all input elements that are not empty
<code>:header</code>	<code>\$(":header")</code>	Select all header elements h1, h2 ...
<code>:animated</code>	<code>\$(":animated")</code>	Select all animated elements
<code>:focus</code>	<code>\$(":focus")</code>	Select the element that currently has focus
<code>:contains(text)</code>	<code>\$(":contains('Hello'))</code>	Select all elements which contains the text "Hello"
<code>:has(selector)</code>	<code>\$("div:has(p)")</code>	Select all div elements that have a p element
<code>:empty</code>	<code>\$(":empty")</code>	Select all elements that are empty

:parent	\$(":parent")	Select all elements that are a parent of another element
:hidden	\$("p:hidden")	Select all hidden p elements
:visible	\$("table:visible")	Select all visible tables
:root	\$(":root")	It will select the document's root element
:lang(language)	\$("p:lang(de)")	Select all p elements with a lang attribute value starting with "de"
[attribute]	\$("[href]")	Select all elements with a href attribute
[attribute=value]	\$("[href='default.htm']")	Select all elements with a href attribute value equal to "default.htm"
[attribute!=value]	\$("[href!='default.htm']")	It will select all elements with a href attribute value not equal to "default.htm"
[attribute\$=value]	\$("[href\$='.jpg']")	It will select all elements with a href attribute value ending with ".jpg"
[attribute =value]	\$("[title ='Tomorrow']")	Select all elements with a title attribute value equal to 'Tomorrow', or starting with 'Tomorrow' followed by a hyphen

[attribute^= value]	\$("[title^='Tom']")	Select all elements with a title attribute value starting with "Tom"
[attribute~= value]	\$("[title~='hello']")	Select all elements with a title attribute value containing the specific word "hello"
[attribute*=v alue]	\$("[title*='hello']")	Select all elements with a title attribute value containing the word "hello"
:input	\$(":input")	It will select all input elements
:text	\$(":text")	It will select all input elements with type="text"
:password	\$(":password")	It will select all input elements with type="password"
:radio	\$(":radio")	It will select all input elements with type="radio"
:checkbox	\$(":checkbox")	It will select all input elements with type="checkbox"
:submit	\$(":submit")	It will select all input elements with type="submit"
:reset	\$(":reset")	It will select all input elements with type="reset"
:button	\$(":button")	It will select all input elements with type="button"

:image	\$(":image")	It will select all input elements with type="image"
:file	\$(":file")	It will select all input elements with type="file"
:enabled	\$(":enabled")	Select all enabled input elements
:disabled	\$(":disabled")	It will select all disabled input elements
:selected	\$(":selected")	It will select all selected input elements
:checked	\$(":checked")	It will select all checked input elements

jQuery - Plugins

A plug-in is a piece of code written in a standard JavaScript file. These files provide useful jQuery methods which can be used along with jQuery library methods.

There are plenty of jQuery plug-in available which you can download from the repository link at <https://jquery.com/plugins>.

How to use Plugins

To make a plug-in's methods available to us, we include a plug-in file very similar to jQuery library file in the <head> of the document.

We must ensure that it appears after the main jQuery source file, and before our custom JavaScript code.

Following example shows how to include jquery.plugin.js plugin –

```
<html>
  <head>
    <title>The jQuery Example</title>
```

```

    <script type = "text/javascript"
        src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
    >
    </script>

    <script src = "jquery.plugin.js" type =
"text/javascript"></script>
    <script src = "custom.js" type =
"text/javascript"></script>

    <script type = "text/javascript" language = "javascript">
        $(document).ready(function() {
            .....your custom code.....
        });
    </script>
</head>

<body>
    .....
</body>
</html>

```

How to develop a Plug-in

This is very simple to write your own plug-in. Following is the syntax to create a method –

```
jQuery.fn.methodName = methodDefinition;
```

Here methodNameM is the name of new method and methodDefinition is actual method definition.

The guideline recommended by the jQuery team is as follows –

- Any methods or functions you attach must have a semicolon (;) at the end.
- Your method must return the jQuery object, unless explicitly noted otherwise.
- You should use this.each to iterate over the current set of matched elements - it produces clean and compatible code that way.

- Prefix the filename with `jquery`, follow that with the name of the plugin and conclude with `.js`.
- Always attach the plugin to jQuery directly instead of `$`, so users can use a custom alias via `noConflict()` method.

For example, if we write a plugin that we want to name *debug*, our JavaScript filename for this plugin is –

`jquery.debug.js`

The use of the `jquery.` prefix eliminates any possible name collisions with files intended for use with other libraries.

Example

Following is a small plug-in to have a warning method for debugging purposes. Keep this code in *jquery.debug.js* file –

```
jQuery.fn.warning = function() {
  return this.each(function() {
    alert("Tag Name:" + $(this).prop("tagName") + ".");
  });
};
```

Here is the example showing usage of `warning()` method. Assuming we put `jquery.debug.js` file in same directory of html page.

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"
      src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
    >
  </script>
  <script src = "jquery.debug.js" type = "text/javascript">
  </script>
  <script type = "text/javascript" language = "javascript">
    $(document).ready(function() {
```

```

        $ ("div").warning();
        $ ("p").warning();
    });
</script>
</head>
<body>
    <p>This is paragraph</p>
    <div>This is division</div>
</body>
</html>

```

This would alert you with following result –

This is paragraph

This is division

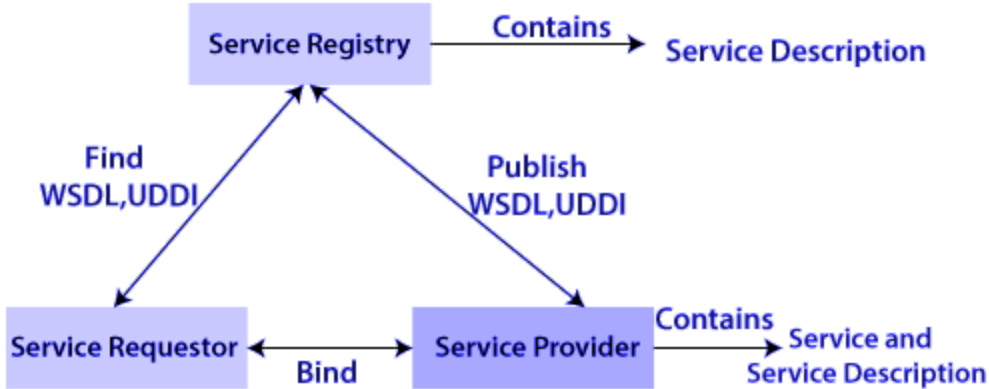
Architecture of Web Services

The Web Services architecture describes how to instantiate the elements and implement the operations in an interoperable manner.

The architecture of web service interacts among three roles: **service provider**, **service requester**, and **service registry**. The interaction involves the three operations: **publish**, **find**, and **bind**. These operations and roles act upon the **web services artifacts**. The web service artifacts are the web service software module and its description.

The service provider hosts a network-associable module (web service). It defines a service description for the web service and publishes it to a service requestor or service registry. These service requestor uses a find operation to retrieve the service description locally or from the service registry. It uses the service description to bind with the service provider and invoke with the web service implementation.

The following figure illustrates the operations, roles, and their interaction.



Web Service Roles, Operations and Artifacts

There are three roles in web service architecture:

- Service Provider
- Service Requestor
- Service Registry

Service Provider

From an architectural perspective, it is the platform that hosts the services.

Service Requestor

Service requestor is the application that is looking for and invoking or initiating an interaction with a service. The browser plays the requester role, driven by a consumer or a program without a user interface.

Service Registry

Service requestors find service and obtain binding information for services during development.

Operations in a Web Service Architecture

Three behaviors that take place in the microservices:

- Publication of service descriptions (**Publish**)
- Finding of services descriptions (**Find**)
- Invoking of service based on service descriptions (**Bind**)

Publish: In the publish operation, a service description must be published so that a service requester can find the service.

Find: In the find operation, the service requestor retrieves the service description directly. It can be involved in two different lifecycle phases for the service requestor:

- At design, time to retrieve the service's interface description for program development.
- And, at the runtime to retrieve the service's binding and location description for invocation.

Bind: In the bind operation, the service requestor invokes or initiates an interaction with the service at runtime using the binding details in the service description to locate, contact, and invoke the service.

Artifacts of the web service

There are two artifacts of web services:

- Service
- Service Registry

Service: A service is an **interface** described by a service description. The service description is the implementation of the service. A service is a software module deployed on network-accessible platforms provided by the service provider. It interacts with a service requestor. Sometimes it also functions as a requestor, using other Web Services in its implementation.

Service Description: The service description comprises the details of the **interface** and **implementation** of the service. It includes its **data types, operations, binding information**, and **network location**. It can also categorize other metadata to enable discovery and utilize by service requestors. It can be published to a service requestor or a service registry.

RESTful Web Services - Introduction

What is REST architecture?

REST stands for REpresentational State Transfer. REST is web standards based architecture and uses HTTP Protocol. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in 2000.

In REST architecture, a REST Server simply provides access to resources and REST client accesses and modifies the resources. Here each resource is identified by URIs/global IDs. REST uses various representation to represent a resource like text, JSON, XML. JSON is the most popular one.

HTTP methods

Following four HTTP methods are commonly used in REST based architecture.

- GET – Provides a read only access to a resource.
- POST – Used to create a new resource.
- DELETE – Used to remove a resource.
- PUT – Used to update a existing resource or create a new resource.

Introduction to RESTful web services

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

Web services based on REST Architecture are known as RESTful web services. These webservice uses HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation such as JSON and set of HTTP Methods.

Creating RESTful Webservice

In next chapters, we'll create a webservice say user management with following functionalities –

Sr.No	URI	HTTP Method	POST body	Result
1	/UserService/users	GET	empty	Show list of all the users.
2	/UserService/addUser	POST	JSON String	Add details of new user.
3	/UserService/getUser/:id	GET	empty	Show details of a user.

Introduction to PHP PDO

PHP is an open-source general-purpose scripting language, which is widely used for creating dynamic and interactive web pages. PHP can access a large range of relational database management systems such as **MYSQL**, **SQLite**, and **PostgreSQL**. The **PHP 5.1**

version offered a new database connection abstraction library, which is **PHP Data Objects (PDO)**.

What is PDO?

PDO refers to **PHP Data Object**, which is a PHP extension that defines a lightweight and consistent interface for accessing a database in PHP. It is a set of PHP extensions which provide a core PDO class and database-specific driver. Each database driver can expose database-specific features as a regular extension function that implements the PDO interface.

Note: We cannot perform any type of database function by using the PDO extension itself. To access a database server, we must use a database-specific PDO driver.

PDO mainly focuses on data access abstraction rather than database abstraction. It provides **data-access abstraction layer**, which means, regardless of which database we are using, we have to use the same functions provided by that database to issue queries and fetch data. PDO does not provide data abstraction, as it does not rewrite the SQL or emulate missing features.

Benefits of using PDO

PDO is the native database driver. There are some benefits of using PDO that are given below:

- **Usability** - It contains many helper functions to operate automatic routine operations.
- **Reusability** - It offers the unified API to access multiple databases.
- **Security** - It uses a prepared statement which protects from SQL injection. A prepared statement is a pre-compiled SQL statement that separates the instruction of the SQL statement from the data.

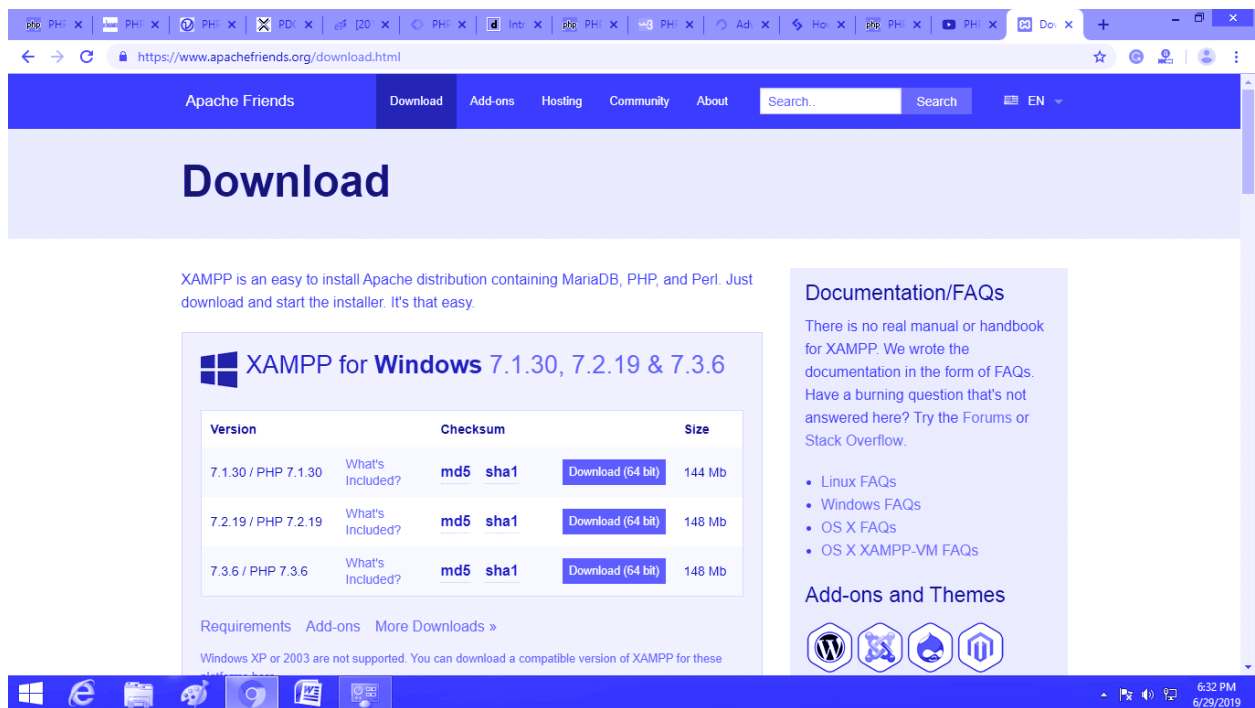
Requirement

There is no need of external libraries to build this extension.

Installation Process

Step 1: Download the latest XAMPP server from here

<https://www.apachefriends.org/download.html> for different platforms like Windows, Linux, and MacOS.



The screenshot shows the Apache Friends website's download page for Windows. The page title is "Download" and it features a table of XAMPP for Windows versions. The table lists three versions: 7.1.30 / PHP 7.1.30 (144 Mb), 7.2.19 / PHP 7.2.19 (148 Mb), and 7.3.6 / PHP 7.3.6 (148 Mb). Each row includes a "Download (64 bit)" button. To the right of the table, there is a "Documentation/FAQs" section with a list of links for Linux, Windows, OS X, and OS X XAMPP-VM FAQs. Below that is an "Add-ons and Themes" section with icons for WordPress, Joomla, Drupal, and Magento. The page also includes a search bar and navigation links like "Download", "Add-ons", "Hosting", "Community", and "About".

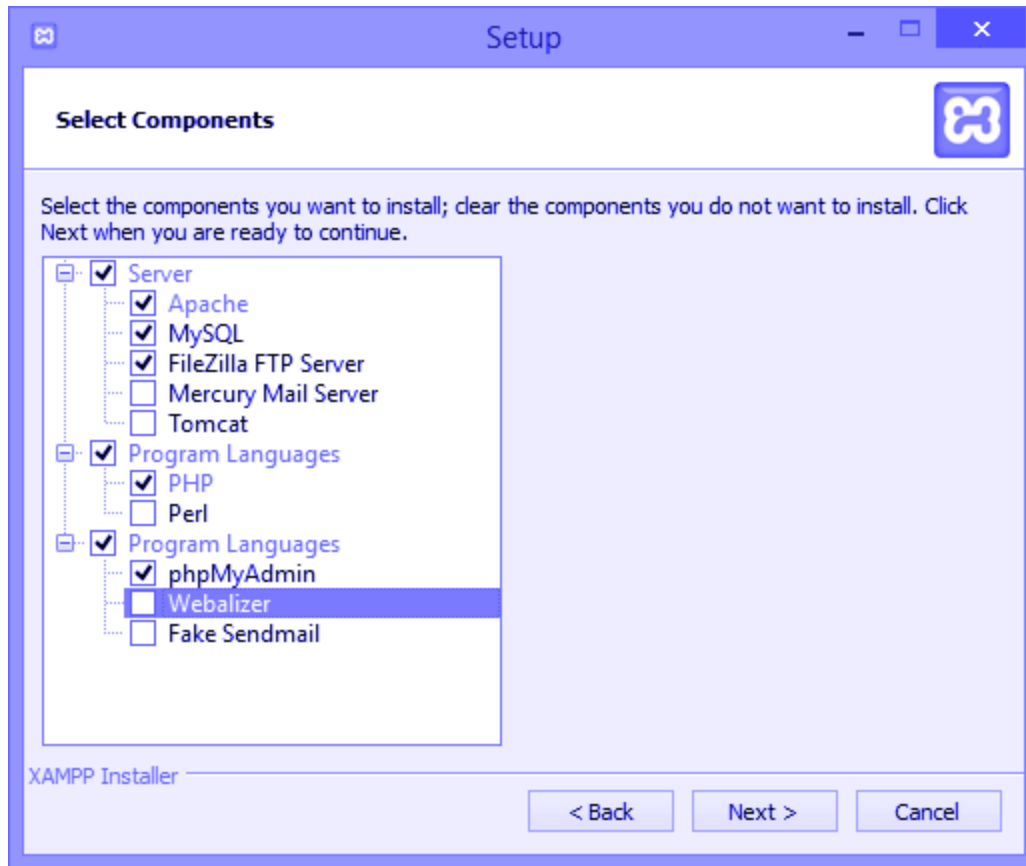
Version	Checksum	Size
7.1.30 / PHP 7.1.30	What's Included? md5 sha1	Download (64 bit) 144 Mb
7.2.19 / PHP 7.2.19	What's Included? md5 sha1	Download (64 bit) 148 Mb
7.3.6 / PHP 7.3.6	What's Included? md5 sha1	Download (64 bit) 148 Mb

Note: Here we will discuss the installation process for Windows OS only.

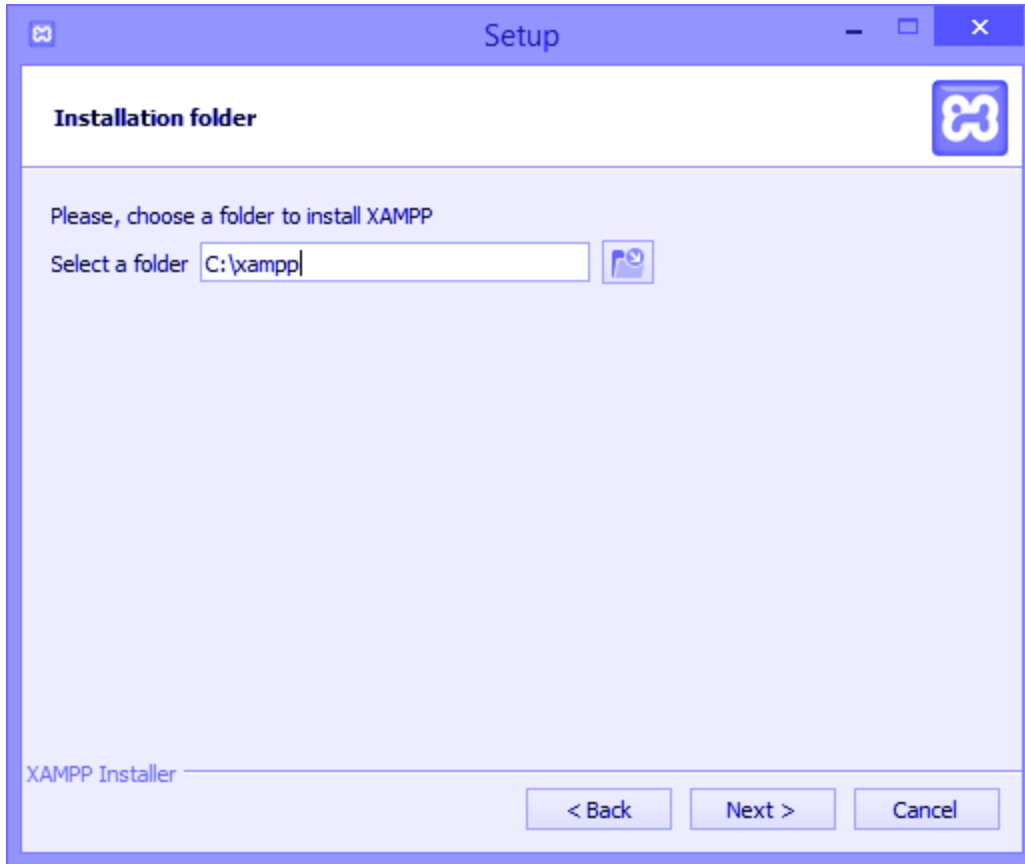
Step 2: Install the XAMPP Server on your system by following these steps.



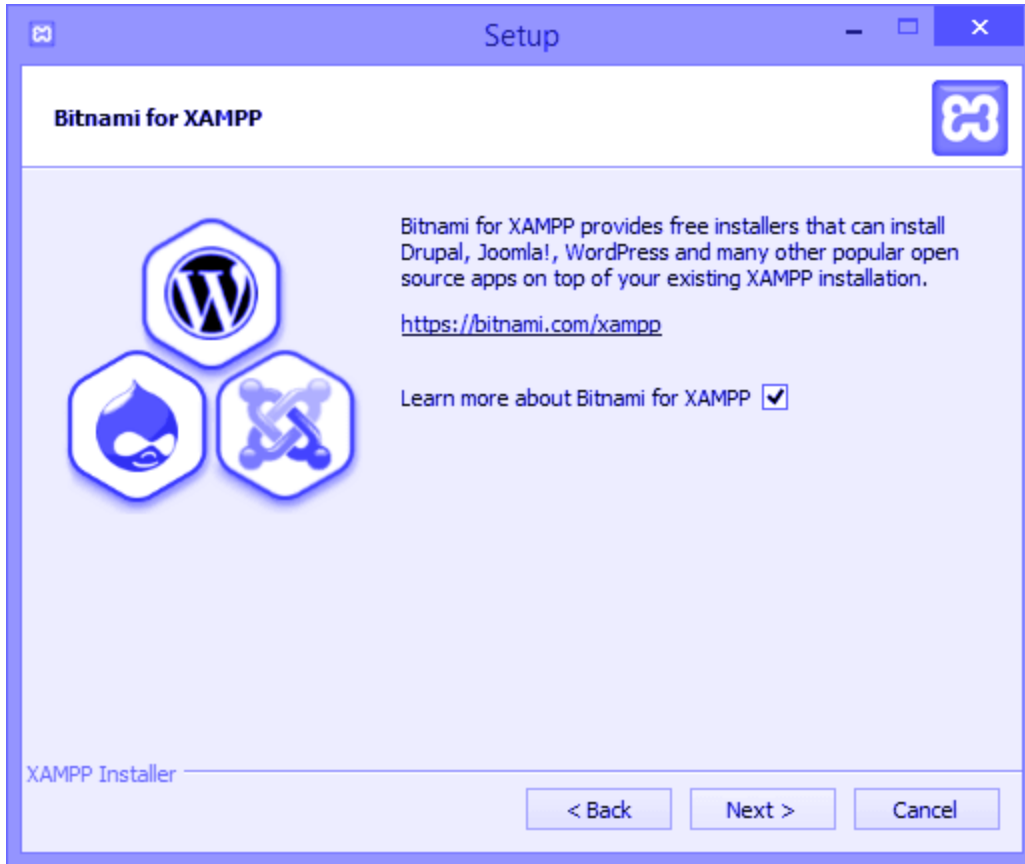
Step 3: Select the components which you want to install and click on the Next button.



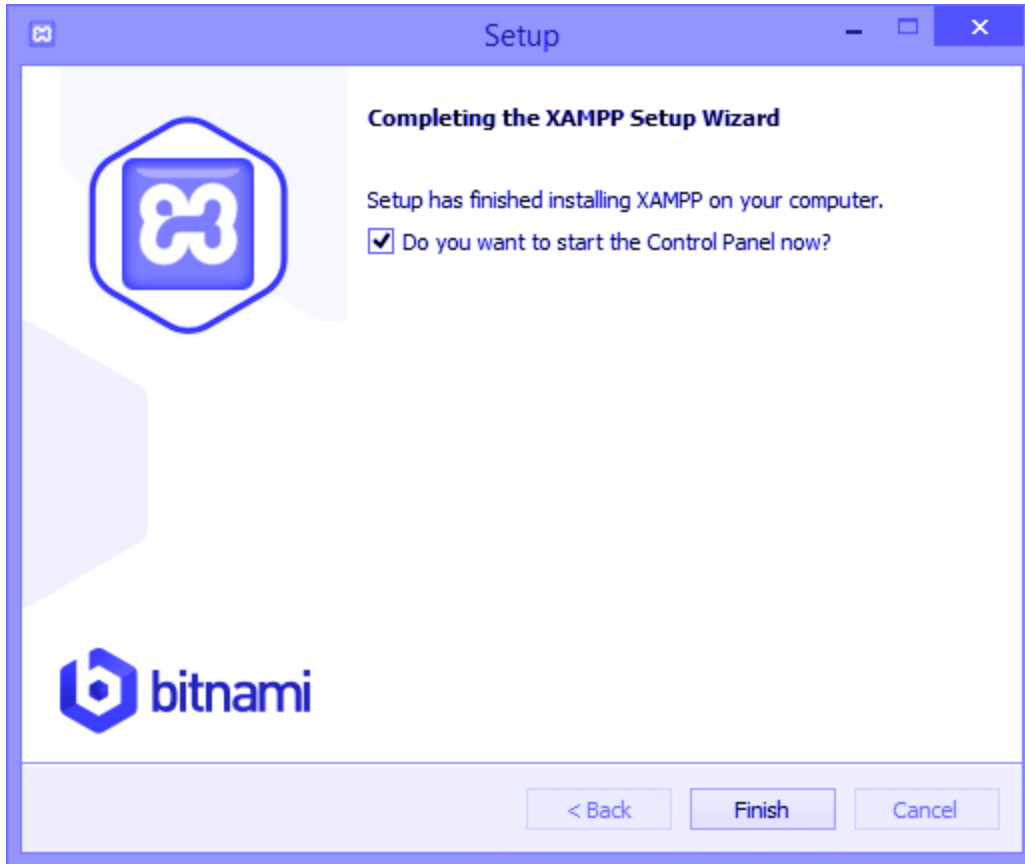
Step 4: Create the new folder with the name xampp at the location where you want to install XAMPP.



Step 5: Click on Next here and move forward. Installation of the XAMPP server will start from here.



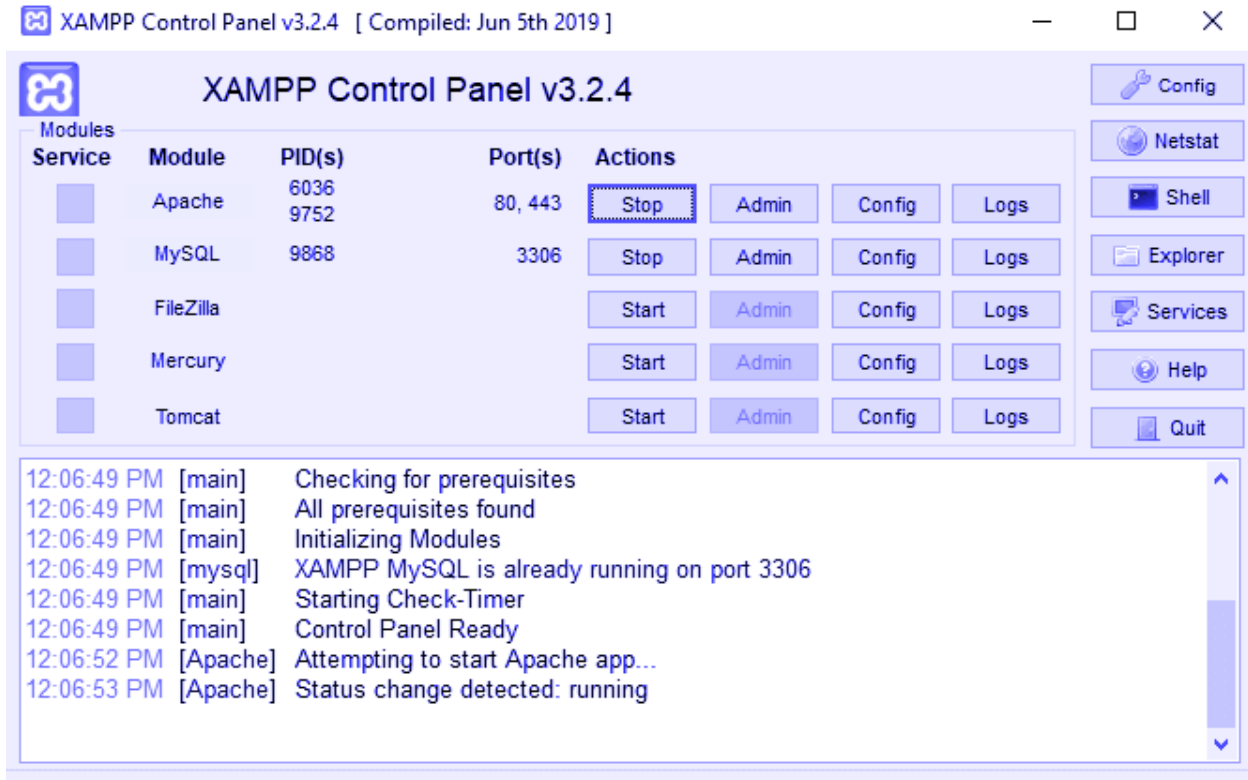
Step 6: XAMPP is installed successfully. Click on Finish button.



Step 7: Select the preferred Language.



Step 8: Run the Apache server and MySQL from here (as per the given screenshot).



Step 9: Now, open php.ini from **C:/xampp/php/php.ini** (where you have installed your XAMPP) and uncomment the extension "**php_pdo_mysql.dll**" and "**php_pdo.dll**" (if you are working with MySQL database), or "**php_pdo_oci.dll**" (if you are working with Oracle database). Now, start working with the database. In PHP 5.1 upper version, it is already set.