



Presidency University, Bengaluru

# CSE 256 INTERNET TECHNOLOGIES

## TUTORIAL SHEET-1

Welcome to the exciting and rapidly evolving world of  
Internet and web programming!

# Introduction.....course overview

---



You'll begin by learning the client-side programming technologies used to build web pages and applications that are run on the client (i.e., in the browser on the user's device).

You'll use Hyper Text Markup Language 5 (HTML5) and Cascading Style Sheets 3 (CSS3)

The recent releases of HTML and CSS technologies—to add powerful, dynamic and fun features and effects to web pages and web applications, such as audio, video, animation, drawing, image manipulation, designing pages for multiple screen sizes

You'll learn JavaScript—the language of choice for implementing the client side of Internet-based applications



---

you'll learn server-side programming—the applications that respond to requests from client-side web browsers, such as searching the Internet, checking your bank-account balance, ordering a book from Amazon, bidding on an eBay auction and ordering concert tickets.

there's also an emphasis on Ajax development, which helps you create better-performing, more usable applications.

# A brief history of internet

---



The Internet—a global network of computers—was made possible by the convergence of computing and communications technologies.

In the late 1960s, ARPA (the Advanced Research Projects Agency) rolled out blueprints for networking the main computer systems of about a dozen ARPA-funded universities and research institutions.

The history of telecommunication and data transport is a long one.

The Internet is not alone in providing instantaneous digital communication. Earlier technologies like radio, telegraph, and the telephone provided the same speed of communication in an analog form.

# What Is the Internet?

---



The Internet is a huge collection of computers connected in a communications network.

These computers are of every imaginable size, configuration, and manufacturer.

In fact, some of the devices connected to the Internet—such as plotters and printers—are not computers at all.

The innovation that allows all these diverse devices to communicate with each other is a single, low-level protocol named Transmission Control Protocol/Internet Protocol (TCP/IP).

# Circuit switching      or      Packet Switching

---



Earlier days of network connections are referred to as circuit switching.

Low bandwidth was one of the major issues with circuit switching

Bandwidth is a measurement of how much data can (maximally) be transmitted along an Internet connection.

Normally measured in bits per second (bps), this measurement differs according to the type of Internet access technology you are using.

A dial-up 56-Kbps modem has far less bandwidth than a 10-Gbps fiber optic connection.



---

A packet-switched network does not require a continuous connection.

Instead it splits the messages into smaller chunks called packets and routes them to the appropriate place based on the destination address.

The packets contained address, error-control and sequencing information.

The packets can take different routes to the destination

# Internet Protocol Addresses

---



To promote the growth and unification of the disparate networks, a suite of protocols was invented to unify the networks.

A protocol is the name given to a formal set of publicly available rules that manage data exchange between two points.

Communications protocols allow any two computers to talk to one another, so long as they implement the protocol.

The protocol for communicating over the ARPANET became known as TCP—the Transmission Control Protocol.

TCP ensured that messages were properly routed from sender to receiver and that they arrived intact.



As the Internet evolved, organizations worldwide were implementing their own networks for both intraorganization (i.e., within the organization) and interorganization (i.e., between organizations) communications.

A wide variety of networking hardware and software appeared.

One challenge was to get these different networks to communicate.

ARPA accomplished this with the development of IP—the Internet Protocol, truly creating a network of networks, the current architecture of the Internet.

The combined set of protocols is now commonly called TCP/IP.



---

Each computer on the Internet has a unique IP address.

The Internet Protocol (IP) address of a machine connected to the Internet is a unique 32-bit number.

IP addresses usually are written (and thought of) as four 8-bit numbers, separated by periods.

The four parts are separately used by Internet-routing computers to decide where a message must go next to get to its destination.



# Domain Names

---

Because people have difficulty dealing with and remembering numbers, machines on the Internet also have textual names.

These names begin with the name of the host machine, followed by progressively larger enclosing collections of machines, called domains.

There may be two, three, or more domain names.



Because IP addresses are the addresses used internally by the Internet, the fully qualified domain name of the destination for a message, which is what is given by a browser user, must be converted to an IP address before the message can be transmitted over the Internet to the destination.

These conversions are done by software systems called name servers, which implement the Domain Name System (DNS).

# The World Wide Web

---



In 1989, a small group of people led by Tim Berners-Lee at Conseil Européen pour la Recherche Nucléaire (CERN) or European Organization for Particle Physics proposed a new protocol for the Internet, as well as a system of document access to use it.

The intent of this new system, which the group named the World Wide Web, was to allow scientists around the world to use the Internet to exchange documents describing their work.

Main postulates of WWW are as follows

1. A technology for sharing information via hyperlinked text documents. Berners-Lee called his invention the HyperTextMarkup Language (HTML).



- 
2. He also wrote communication protocols to form the backbone of his new information system, In particular, he wrote the Hypertext Transfer Protocol (HTTP)—a communications protocol used to send information over the web.
  3. The units of information on the Web have been referred to by several different names; among them, the most common are pages, documents, and resources.
  4. A Uniform Resource Locator (URL) to uniquely identify a resource on the WWW. Each web page on the Internet is associated with a unique URL.



- 
5. A software program (later called web server software) that can respond to HTTP requests.
  6. A program (later called a browser) that can make HTTP requests from URLs and that can display the HTML it receives.



# Web Browsers

When two computers communicate over some network, in many cases one acts as a client and the other as a server.

The client initiates the communication, which is often a request for information stored on the server, which then sends that information back to the client.

The Web, as well as many other systems, operates in this client-server configuration.

Documents provided by servers on the Web are requested by browsers, which are programs running on client machines.

They are called browsers because they allow the user to browse the resources available on servers.



# Web Servers

---

Web servers are programs that provide documents to requesting browsers.

Servers are slave programs: They act only when requests are made to them by browsers running on other computers on the Internet.

The most commonly used Web servers are Apache, which has been implemented for a variety of computer platforms,

and Microsoft's Internet Information Server (IIS), which runs under Windows operating systems

# World Wide Web Consortium

---



In October 1994, Tim Berners-Lee founded an organization—the World Wide Web Consortium (W3C)—devoted to developing nonproprietary, interoperable technologies for the World Wide Web.

One of the W3C's primary goals is to make the web universally accessible—regardless of disability, language or culture.

The W3C is also a standards organization. Web technologies standardized by the W3C are called Recommendations.

Current and forthcoming W3C Recommendations include the Hyper Text Markup Language 5 (HTML5), Cascading Style Sheets 3 (CSS3) and the Extensible Markup Language (XML).



# Governance

---

Internet Engineering Task Force (IETF)- The technical underpinning and standardization of the core protocols (IPv4 and IPv6)

Internet Corporation for Assigned Names and Numbers (ICANN)- coordinates the assignment of unique identifiers for use on the Internet, including domain names, Internet Protocol (IP) addresses, application port numbers in the transport protocols etc...



# Intranet or Internet

---

One of the more common terms you might encounter in web development is the term “intranet” (with an “a”), which refers to an Internet network that is local to an organization or business.

Intranet resources are often private, meaning that only employees (or authorized external parties such as customers or suppliers) have access to those resources.

Internet is a broader term that encompasses both private (intranet) and public networked resources.



Presidency University, Bengaluru

# CSE 256 INTERNET TECHNOLOGIES

## TUTORIAL SHEET-2

Welcome to the exciting and rapidly evolving world of  
Internet and web programming!

# The Client-Server Model



The web is sometimes referred to as a client-server model of communications.

In the client-server model, there are two types of actors: clients and servers.

The server is a computer agent that is normally active 24 hours a day, 7 days a week, listening for queries from any client who make a request.

A client is a computer agent that makes requests and receives responses from the server, in the form of response codes, images, text files, and other data.



Client machines are the desktops, laptops, smart phones, and tablets you see everywhere in daily life.

These machines have a broad range of specifications regarding operating system, processing speed, screen size, available memory, and storage.

In the most familiar scenario, client requests for web pages come through a web browser.

The essential characteristic of a client is that it can make requests to particular servers for particular resources using URLs and then wait for the response.

These requests are processed in some way by the server.



---

The server in this model is the central repository, the command center, and the central hub of the client-server model.

It hosts web applications, stores user and program data, and performs security authorization tasks.

The essential characteristic of a server is that it is listening for requests, and upon getting one, responds with a message.

The exchange of information between the client and server is often referred as the request-response loop.



# Server Types

- Web servers. A web server is a computer servicing HTTP requests. This typically refers to a computer running web server software such as Apache or Microsoft IIS (Internet Information Services).
- Application servers. An application server is a computer that hosts and executes web applications, which may be created in PHP, ASP.NET, Ruby on Rails, or some other web development technology.
- Database servers. A database server is a computer that is devoted to running a Database Management System (DBMS), such as MySQL, Oracle, or SQL Server, that is being used by web applications.



- Mail servers. A mail server is a computer creating and satisfying mail requests, typically using the Simple Mail Transfer Protocol (SMTP).
- Media servers. A media server (also called a streaming server) is a special type of server dedicated to servicing requests for images and videos. It may run special software that allows video content to be streamed to clients.

# Web Basics



a web page is nothing more than an HTML (HyperText Markup Language) document (with the extension .html or .htm) that describes to a web browser the document's content and structure.

## Hyperlinks

HTML documents normally contain hyperlinks, which, when clicked, load a specified web document.

Both images and text may be hyperlinked.

When the mouse pointer hovers over a hyperlink, the default arrow pointer changes into a hand with the index finger pointing upward.

Often hyperlinked text appears underlined and in a different color from regular text in a web page.



## URL Formats

All URLs have the same general format:

The first part of the URL is the protocol that we are using.



The domain identifies the server from which we are requesting resources. Since the DNS system is case insensitive, this part of the URL is case insensitive. Alternatively, an IP address can be used for the domain.



The path is a familiar concept to anyone who has ever used a computer file system. The root of a web server corresponds to a folder somewhere on that server.

\*Query strings is a way of passing information such as user form input from the client to the server. In URLs, they are encoded as key-value pairs delimited by “&” symbols and preceded by the “?” symbol.

The last part of a URL is the optional fragment. This is used as a way of requesting a portion of a page.

*\*[will be explained in later sessions]*

# The Hypertext Transfer Protocol

---



All Web communications transactions use the same protocol: the Hypertext Transfer Protocol (HTTP).

HTTP consists of two phases: the request and the response.

Each HTTP communication (request or response) between a browser and a Web server consists of two parts: a header and a body.

The header contains information about the communication

The body contains the data of the communication if there is any.

The HTTP establishes a TCP connection on port 80 (by default).



---

The HTTP protocol defines several different types of requests, each with a different intent and characteristics.

The most common requests are the GET and POST request.

### GET request

In this request one is asking for a resource located at a specified URL to be retrieved. Whenever you click on a link, type in a URL in your browser, or click on a book mark, you are usually making a GET request.



## POST request

This method is normally used to transmit data to the server using an HTML form. A post request sends form data as part of the HTTP message, not as part of the URL. since the data is not transmitted in the URL, it is seen to be a safer way of transmitting data

**Response codes** are integer values returned by the server as part of the response header. These codes describe the state of the request, including whether it was successful, had errors, requires permission, and more.



---

200: OK :-The 200 response code means that the request was successful.

301: Moved Permanently:- Tells the client that the requested resource has permanently moved. Codes like this allow search engines to update their databases to reflect the new location of the resource. Normally the new location for that resource is returned in the response.

304: Not Modified:- If the client so requested a resource with appropriate Cache-Control headers, the response might say that the resource on the server is no newer than the one in the client cache. A response like this is just a header, since we expect the client to use a cached copy of the resource.



401: Unauthorized:- Some web resources are protected and require the user to provide credentials to access the resource. If the client gets a 401 code, the request will have to be resent, and the user will need to provide those credentials.

404: Not found:- 404 codes are one of the only ones known to web users. Many browsers will display an HTML page with the 404 code to them when the requested resource was not found.

414: Request URI too long:- URLs have a length limitation, which varies depending on the server software in place. A 414 response code likely means too much data is likely trying to be submitted via the URL.



---

307: Temporary redirect :-This code is similar to 301, except the redirection should be considered temporary.

400: Bad Request:- If something about the headers or HTTP request in general is not correctly adhering to HTTP protocol, the 400 response code will inform the client.

500: Internal server error:- This error provides almost no information to the client except to say the server has encountered an error.

# Web Applications in Comparison to Desktop Applications



advantages of web applications include:

- Accessible from any Internet-enabled computer.
- Usable with different operating systems and browser applications.
- Easier to roll out program updates since only software on the server needs to be updated and not on every desktop in the organization.
- Centralized storage on the server means fewer security concerns about local.



---

Some of these disadvantages include:

- Requirement to have an active Internet connection (the Internet is not always available everywhere at all times).
- Security concerns about sensitive private data being transmitted over the Internet.
- Concerns over the storage, licensing, and use of uploaded data.
- Problems with certain websites on certain browsers not looking quite right.

# Static Websites versus Dynamic Websites

---



Static website consists only of HTML pages that look identical for all users at all times.

dynamic website page content is being created at run time by a program created by a programmer; this page content can vary from user to user.



# Client-Side Scripting versus Server-Side Scripting

---

Client-side scripting:- method of interacting with web browser

Client-side scripting with JavaScript can be used to validate user input, to interact with the browser, to enhance web pages, and to add client/server communication between a browser and a web server.

server-side scripts:- allow user to interact with server



Presidency University, Bengaluru

# CSE 256 INTERNET TECHNOLOGIES

## TUTORIAL SHEET-3

**Where Is the Internet?**

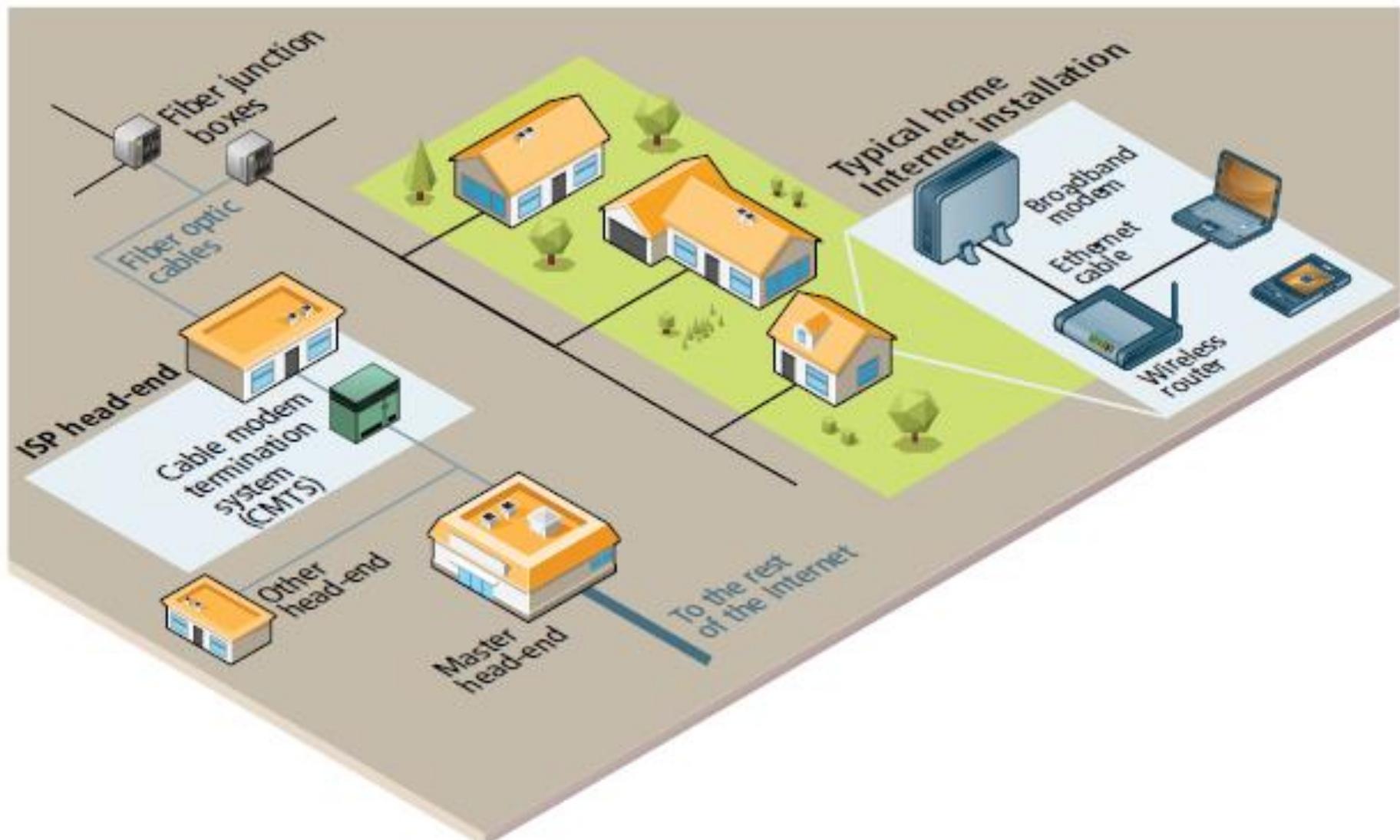
**Let your imagination fly beyond the computer/mobile screen**

---



The global network of networks is implemented via millions of miles of copper wires and fiber optic cables, as well as via hundreds of thousands or even millions of server computers and probably an equal number of routers, switches, and other networked devices, along with many thousands of air conditioning units and specially constructed server rooms and buildings.

# Internet hardware from the home computer to the local Internet provider





The **broadband modem** (also called a cable modem or DSL modem) is a **bridge** between the network hardware outside the house (typically controlled by a phone or cable company) and the network hardware inside the house.

These devices are often supplied by the ISP.



## 2. router

---

The wireless router is perhaps the most visible manifestation of the Internet in one's home, in that it is a device we typically need to purchase and install.

Routers are in fact one of the most important and ubiquitous hardware devices that make the Internet work.

At its simplest, a router is a hardware device that forwards data packets from one network to another network.

When the router receives a data packet, it examines the packet's destination address and then forwards it to another destination by deciding the best path to send the packets.



### 3. Fiber optic cable

Fiber optic cable (or simply optical fiber) is a glass-based wire that transmits light and has significantly greater bandwidth and speed in comparison to metal wires.

In some cities (or large buildings), you may have fiber optic cable going directly into individual buildings; in such a case the fiber junction box will reside in the building.

These fiber optic cables eventually make their way to an ISP's head-end, which is a facility that may contain a cable modem termination system (CMTS) or a digital subscriber line access multiplexer (DSLAM) in a DSL-based system.

# Internet Protocols- An OS abstraction



The Internet exists today because of a suite of interrelated communications protocols.

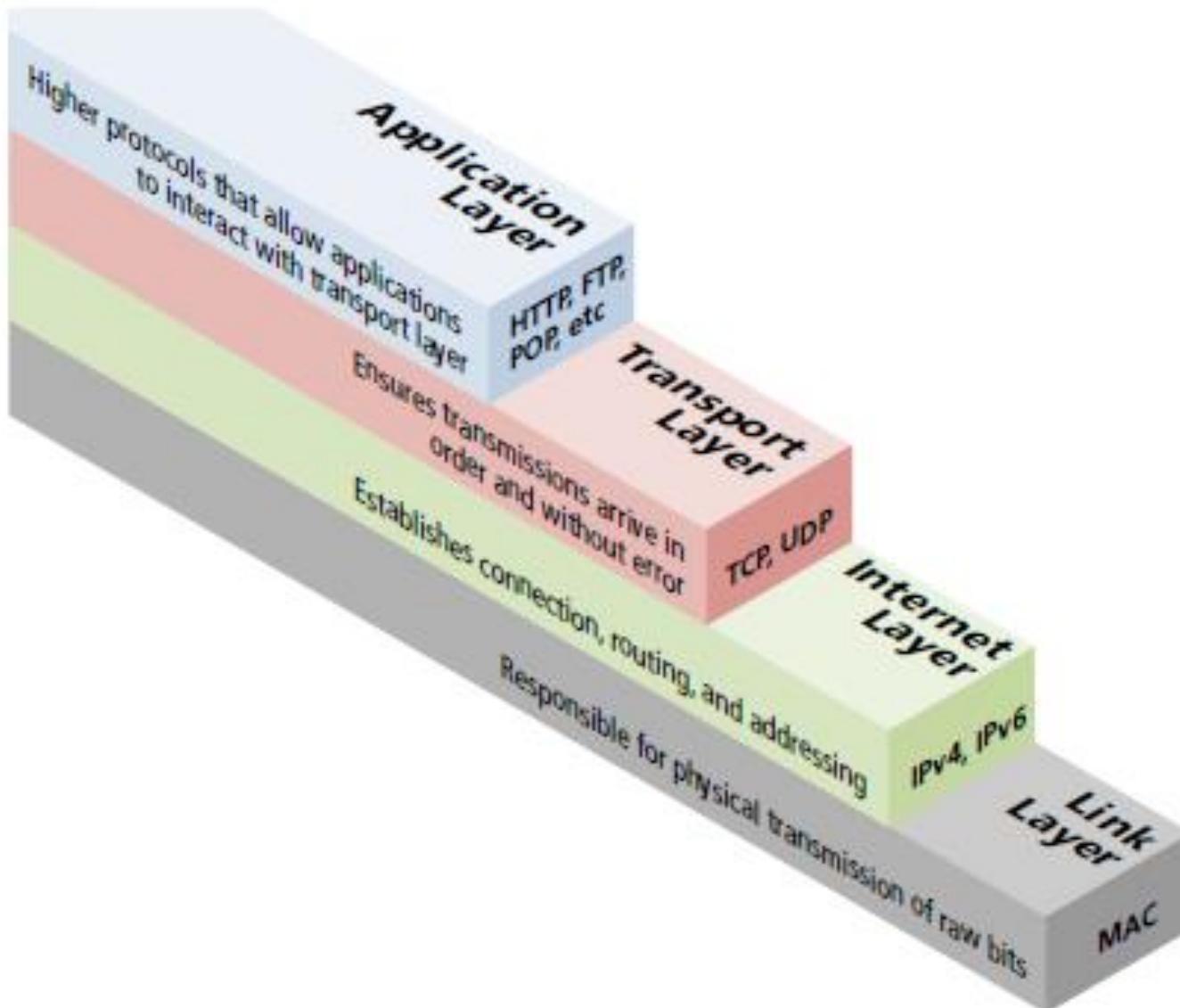
**A protocol is a set of rules that partners in communication use when they communicate.**

We have already mentioned one of these essential Internet protocols, namely TCP/IP.

These protocols have been implemented in every operating system



# Layered architecture- A quick revision





# link layer

---

The link layer is the lowest layer, responsible for both the physical transmission across media (wires, wireless) and establishing logical links.

It handles issues like packet creation, transmission, reception, error detection, collisions, line sharing

The MAC (media access control) addresses:- unique 48- or 64-bit identifiers assigned to network hardware and which are used at the physical networking level



# The Internet layer

The Internet layer (sometimes also called the IP Layer) routes packets between communication partners across networks.

The Internet layer provides “best effort” communication.

It sends out the message to the destination, but expects no reply, and provides no guarantee the message will arrive intact, or at all.

The Internet uses the Internet Protocol (IP) addresses to identify destinations on the Internet.

Every device connected to the Internet has an IP address, which is a numeric code that is meant to uniquely identify it.



# The transport layer

The transport layer ensures transmissions arrive in order and without error.

The data is broken into packets formatted according to the Transmission Control Protocol (TCP).

Each data packet has a header that includes a sequence number, so the receiver can put the original message back in order, no matter when they arrive.

Each packet is acknowledged back to the sender so in the event of a lost packet, the transmitter will realize a packet has been lost since no ACK arrived for that packet.

That packet is retransmitted, and although out of order, is reordered at the destination



# Application layer

Application layer protocols implement process-to-process communication and are at a higher level of abstraction in comparison to the low-level packet and IP address protocols in the layers below it.

There are many application layer protocols. A few that are useful to web developers include:

- HTTP. The Hypertext Transfer Protocol is used for web communication.
- SSH. The Secure Shell Protocol allows remote command-line connections to servers.
- FTP. The File Transfer Protocol is used for transferring files between computers.
- POP/IMAP/SMTP. Email-related protocols for transferring and storing email.
- DNS. The Domain Name System protocol used for resolving domain names to IP addresses.



Presidency University, Bengaluru

# CSE 256 INTERNET TECHNOLOGIES

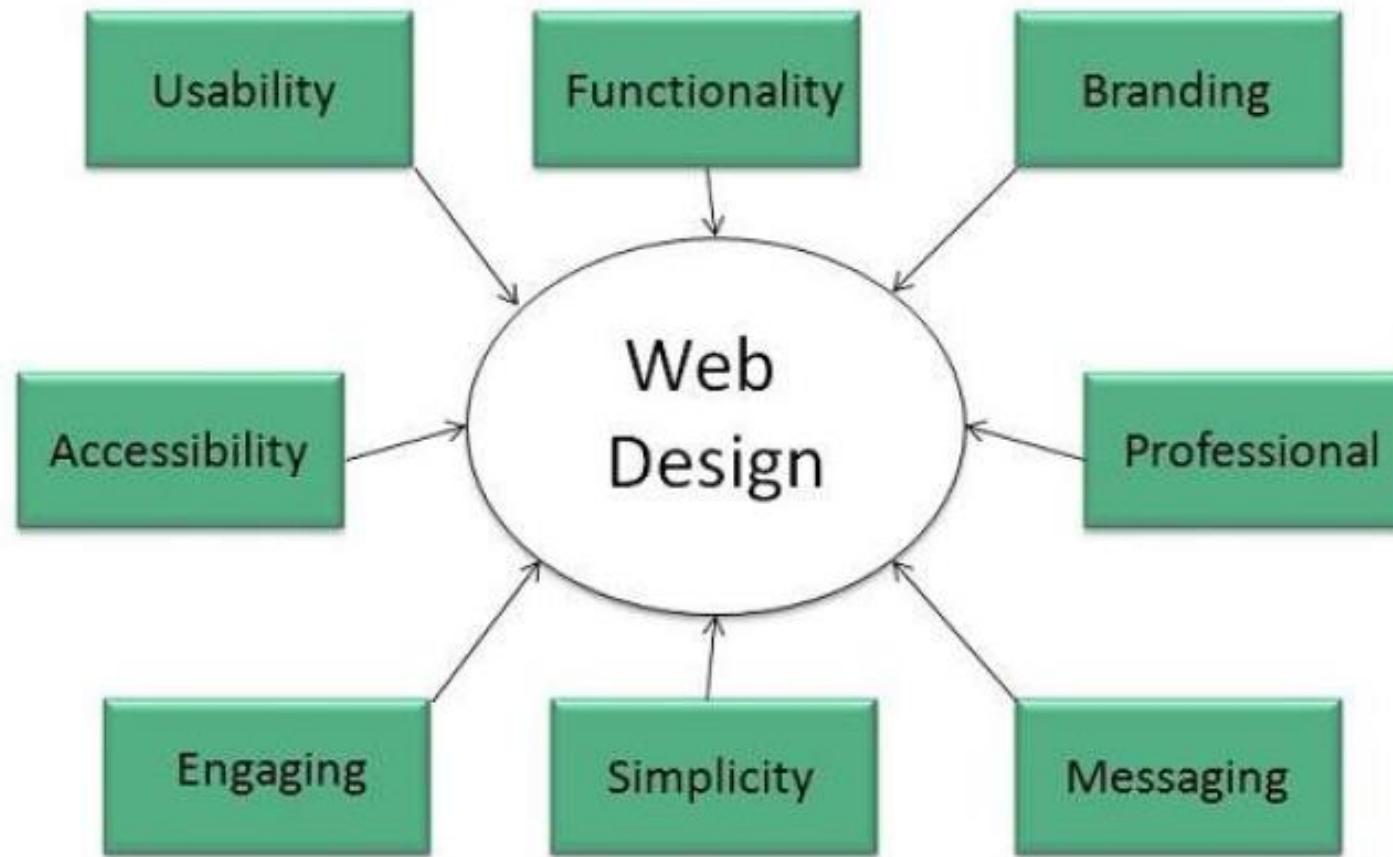
## TUTORIAL SHEET-4

**Website Designing**



# Website Designing

Web designing has direct link to visual aspect of a web site. Effective web design is necessary to communicate ideas effectively.





# Web Page Anatomy

## Containing Block

Container can be in the form of page's body tag, an all containing div tag. Without container there would be no place to put the contents of a web page.

## Logo

Logo refers to the identity of a website and is used across a company's various forms of marketing such as business cards, letterhead, broachers and so on.

## Naviagation

The site's navigation system should be easy to find and use. Often the navigation is placed right at the top of the page.



# Web Page Anatomy

## Content

The content on a web site should be relevant to the purpose of the web site.

## Footer

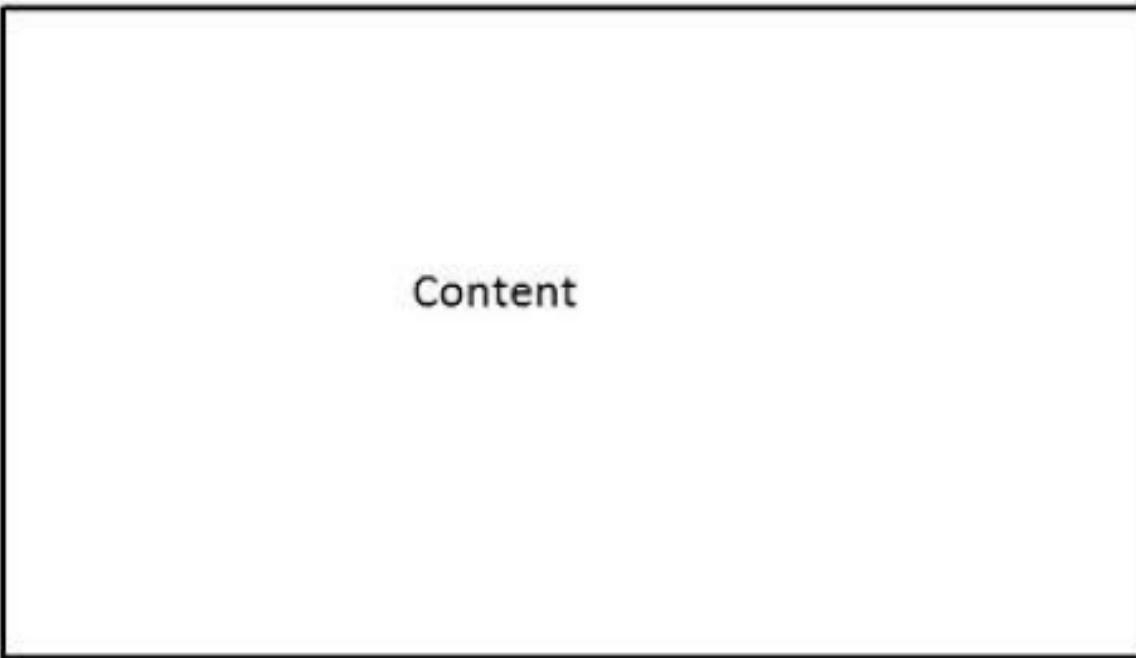
Footer is located at the bottom of the page. It usually contains copyright, contract and legal information as well as few links to the main sections of the site.

## Whitespace

It is also called as negative space and refers to any area of page that is not covered by type or illustrations.



LOGO



White Space





# Web development

Web development refers to building website and deploying on the web. Web development requires use of scripting languages both at the server end as well as at client end.

Before developing a web site once should keep several aspects in mind like:

- What to put on the web site?
- Who will host it?
- How to make it interactive?
- How to code it?
- How to create search engine friendly web site?
- How to secure the source code frequently?
- Will the web site design display well in different browsers?

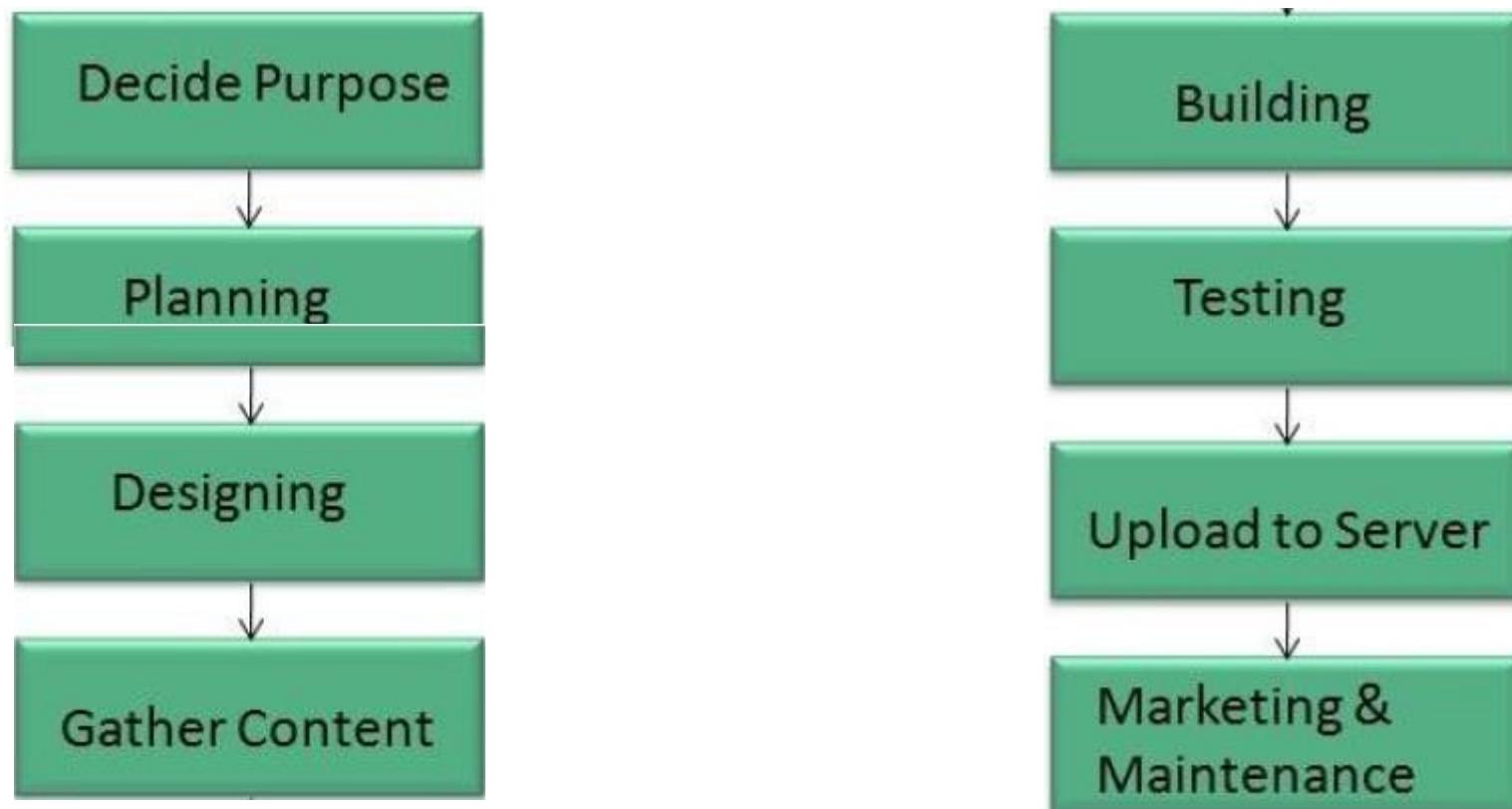


- Will the navigation menus be easy to use?
- Will the web site loads quickly?
- How easily will the site pages print?
- How easily will visitors find important details specific to the web site?
- How effectively the style sheets be used on your web sites?

# Web Development Process



Web development process includes all the steps that are good to take to build an attractive, effective and responsive website.





# Websites Hosting

Web hosting is a service of providing online space for storage of web pages. These web pages are made available via World Wide Web.

The companies which offer website hosting are known as Web hosts.

The servers on which web site is hosted remain switched on 24 x7.

These servers are run by web hosting companies.  
Each server has its own IP address.

Since IP addresses are difficult to remember therefore, webmaster points their domain name to the IP address of the server their website is stored on.



# Types of Hosting

## Shared Hosting

In shared hosting, the hosting company puts thousand of website on the same physical server.

Each customer has their own allocation of physical web space and a set of bandwidth limit.

As all websites share same physical memory, MYSQL server and Apache server, one website on the server experiencing high traffic load will affect performance of all websites on the server.



## **Virtual Private Server VPS**

It is also known as Virtual Dedicated Server. It is a server which is partitioned into smaller servers.

In this customer is given their own partition, which is installed with its own operating system.

Unlike shared hosting, VPS doesn't share memory or processor time rather it allocates certain amount of memory and CPU to use which means that any problem on a VPS partition on the same drive will not affect other VPS customers.



## Dedicated Server

In this kind of hosting, single dedicated server is setup for just one customer.

It is commonly used by the businesses that need the power, control and security that a dedicated server offers.

## Reseller Hosting

A reseller acts as a middle man and sells hosting space of someone else's server.

## Grid Hosting

Instead of utilizing one server, Grid Hosting spreads resources over a large number of servers. It is quite stable and flexible. The servers can be added or taken away from the grid without crashing the system.

# Website Security Considerations

---



Websites are always prone to security risks.

Cyber crime impacts your business by hacking your website.

Your website is then used for hacking assaults that install malicious software or malware on your visitor's computer.

## Updated Software

It is mandatory to keep your software updated. It plays vital role in keeping your website secure.



# Security Considerations

## SQL Injection

It is an attempt by the hackers to manipulate your database.

It is easy to insert rogue code into your query that can be used to manipulate your database such as change tables, get information or delete data.

## Cross Site Scripting XSS

It allows the attackers to inject client side script into web pages. Therefore, while creating a form It is good to endure that you check the data being submitted and encode or strip out any HTML.



# Security Considerations

## Error Messages

You need to be careful about how much information to be given in the error messages.

For example, if the user fails to log in the error message should not let the user know which field is incorrect: username or password.

## Validation of Data

The validation should be performed on both server side and client side.



# Security Considerations

## Passwords

It is good to enforce password requirements such as of minimum of eight characters, including upper case, lower case and special character. It will help to protect user's information in long run.

## Upload files

The file uploaded by the user may contain a script that when executed on the server opens up your website.

## SSL

It is good practice to use SSL protocol while passing personal information between website and web server or database.

## HTML exercises

Develop an HTML page which displays the following:

1. Prints your name in a Tahoma font.
2. Print a paragraph that is a description of a book, include the title of the book as well as its author. Names and titles should be underlined, adjectives should be italicized and bolded.
3. Print the squares of the numbers 1 - 5. Each number should be on a separate line, next to it the number 2 superscripted, an equal sign and the result.
4. Write HTML page containing a text area as follows:  
Tell us about your interest (maximum 100 words)
5. Write HTML page to create the following lists:
  - Pentium I
  - Pentium II
    - 1. 1.6 GHz
    - 2. 2.4 GHz
    - 3. 3.2 GHz
6. Write an HTML page to display the following:  
First name: [.....]  
Middle name: [.....]  
Last name: [.....]
7. Create links to five different pages on five different websites that should all open in a new window.
8. Create a page with a link at the top of it that when clicked will jump all the way to the bottom of the page. At the bottom of the page there should be a link to jump back to the top of the page.
9. Display an image that when clicked will link to itself and will display the image in the browser by itself.

-----END-----

# JavaScript

- JavaScript is used to create client-side dynamic pages.
- JavaScript is *an object-based scripting language* which is lightweight and cross-platform.
- JavaScript is not a compiled language, but it is a translated language.
- The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.

# History

- It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.
- It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser.
- Since then, it has been adopted by all other graphical web browsers.

- With JavaScript, users can build modern web applications to interact directly without reloading the page every time.
- JavaScript has no connectivity with Java programming language.
- In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

- In 1993, **Mosaic**, the first popular web browser, came into existence.
- In the year **1994**, **Netscape** was founded by **Marc Andreessen**.
- He realized that the web needed to become more dynamic. Thus, a 'glue language' was believed to be provided to HTML to make web designing easy for designers and part-time programmers.

- Consequently, in 1995, the company recruited **Brendan Eich** intending to implement and embed Scheme programming language to the browser.
- But, before Brendan could start, the company merged with **Sun Microsystems** for adding Java into its Navigator so that it could compete with Microsoft over the web technologies and platforms.

- Finally, in May 1995, Marc Andreessen coined the first code of Javascript named '**Mocha**'.
- Later, the marketing team replaced the name with '**LiveScript**'.
- But, due to trademark reasons and certain other reasons, in December 1995, the language was finally renamed to '**JavaScript**'. From then, JavaScript came into existence.

# Application of JavaScript

- JavaScript is used to create interactive websites.  
It is mainly used for:
- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

# JavaScript Example

```
<html>
<body>
<h2>Welcome to JavaScript</h2>
<script>
document.write("Hello JavaScript by JavaScript");
</script>
</body>
</html>
```

# OUTPUT

**Welcome to JavaScript**

Hello JavaScript by JavaScript

# DataTypes

- JavaScript provides different **data types** to hold different types of values.
- There are two types of data types in JavaScript.
  - Primitive data type
  - Non-primitive (reference) data type
- JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine.

- **var** here to specify the data type.
- It can hold any type of values such as numbers, strings etc.
- For example:
- `var a=40;//holding number`
- `var b="Rahul";//holding string`

# primitive data types

---

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

# Non-primitive data types

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

# Examples

---

```
var length = 16;                      // Number
var lastName = "Johnson";              // String
var x = {firstName:"John", lastName:"Doe"}; // Object

var x;                                // Now x is undefined
x = 5;                                // Now x is a Number
x = "John";                            // Now x is a String
```

# 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).

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

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

Output: 30

# JavaScript local variable

- A JavaScript local variable is declared inside 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 window object is known as global variable.
- A **JavaScript global variable** is declared outside the function or declared with window object. It can be accessed from any function.
- For example:

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

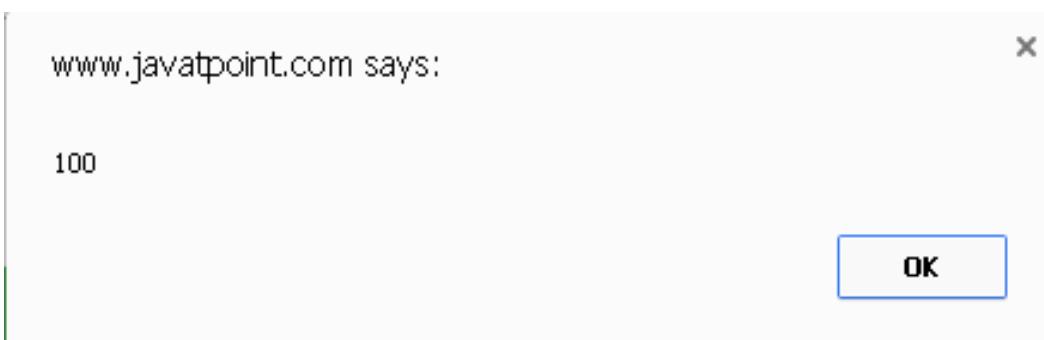
</script>
</body>
</html>
```

200 200

# Declaring JavaScript global variable within function

- To declare JavaScript global variables inside function, you need to use **window object**.
- For example: **window.value=90;**

```
<html>
<body>
<script>
function m(){
window.value=100;//declaring global variable by window object
}
function n(){
alert(window.value);//accessing global variable from other function
}
m();
n();
</script>
</body>
</html>
```



# JavaScript Operators

- JavaScript operators are symbols that are used to perform operations on operands.
- For example:
- `var sum=10+20;`
- Here, `+` is the arithmetic operator and `=` is the assignment operator.

- Arithmetic Operators
- Comparison (Relational) Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Special Operators



# Arithmetic Operators

- Arithmetic operators are used to perform arithmetic operations on the operands.
- The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	$10+20 = 30$
-	Subtraction	$20-10 = 10$
*	Multiplication	$10*20 = 200$
/	Division	$20/10 = 2$
%	Modulus (Remainder)	$20\%10 = 0$
++	Increment	<code>var a=10; a++; Now a = 11</code>
--	Decrement	<code>var a=10; a--; Now a = 9</code>

# Comparison Operators

- The JavaScript comparison operator compares the two operands.

Operator	Description	Example
<code>==</code>	Is equal to	<code>10==20 = false</code>
<code>===</code>	Identical (equal and of same type)	<code>10===20 = false</code>
<code>!=</code>	Not equal to	<code>10!=20 = true</code>
<code>!==</code>	Not Identical	<code>20!==20 = false</code>
<code>&gt;</code>	Greater than	<code>20&gt;10 = true</code>
<code>&gt;=</code>	Greater than or equal to	<code>20&gt;=10 = true</code>
<code>&lt;</code>	Less than	<code>20&lt;10 = false</code>
<code>&lt;=</code>	Less than or equal to	<code>20&lt;=10 = false</code>

# Bitwise Operators

- The bitwise operators perform bitwise operations on operands.

Operator	Description	Example
&	Bitwise AND	$(10 == 20 \& 20 == 33) = \text{false}$
	Bitwise OR	$(10 == 20   20 == 33) = \text{false}$
^	Bitwise XOR	$(10 == 20 ^ 20 == 33) = \text{false}$
~	Bitwise NOT	$(\sim 10) = -10$
<<	Bitwise Left Shift	$(10 << 2) = 40$
>>	Bitwise Right Shift	$(10 >> 2) = 2$
>>>	Bitwise Right Shift with Zero	$(10 >>> 2) = 2$

# Logical Operators

Operator	Description	Example
<code>&amp;&amp;</code>	Logical AND	<code>(10==20 &amp;&amp; 20==33) = false</code>
<code>  </code>	Logical OR	<code>(10==20    20==33) = false</code>
<code>!</code>	Logical Not	<code>!(10==20) = true</code>

# Assignment Operators

Operator	Description	Example
=	Assign	$10+10 = 20$
+=	Add and assign	<code>var a=10; a+=20; Now a = 30</code>
-=	Subtract and assign	<code>var a=20; a-=10; Now a = 10</code>
*=	Multiply and assign	<code>var a=10; a*=20; Now a = 200</code>
/=	Divide and assign	<code>var a=10; a/=2; Now a = 5</code>
%=	Modulus and assign	<code>var a=10; a%=2; Now a = 0</code>

# Special Operators

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.

# Control Structure

- Control structure actually controls the flow of execution of a program.
- if ... else
- switch case
- do while loop
- while loop
- for loop

# If ... else

- The **if statement** is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.
- The **else statement** to specify a block of code to be executed if the condition is false.

```
if (hour < 18) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}
```

---

Good day

---

# Switch case

- The switch statement is used to perform different actions based on different conditions.
- The switch statement to select one of many code blocks to be executed.

## Syntax

```
switch(expression) {  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    default:  
        // code block  
}
```

# Switch case

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

```
<script type="text/javascript">
<!--
    var grade='A';
    document.write("Entering switch block<br/>");
    switch (grade) {
        case 'A': document.write("Good job<br/>");
                    break;
        case 'B': document.write("Pretty good<br/>");
                    break;
        case 'C': document.write("Passed<br/>");
                    break;
        case 'D': document.write("Not so good<br/>");
                    break;
        case 'F': document.write("Failed<br/>");
                    break;
        default:  document.write("Unknown grade<br/>")
    }
    document.write("Exiting switch block");
//-->
</script>
```

# Do while Loop

- The do loop will always be executed at least once, even if the while condition is false.

```
do {  
    Statement(s) to be executed;  
} while (expression);
```

# Example

```
<script type="text/javascript">  
    <!--  
        var count = 0;  
        document.write("Starting Loop" + "<br/>");  
        do{  
            document.write("Current Count : " + count + "<br/>");  
            count++;  
        }while (count < 0);  
        document.write("Loop stopped!");  
    //-->  
</script>
```

Starting Loop  
Current Count : 0  
Loop stopped!

# While Loop

- The purpose of a while loop is to execute a statement or code block repeatedly as long as expression is true.
- Once expression becomes false, the loop will be exited.

```
while (expression) {  
    Statement(s) to be executed if expression is true  
}
```

```
<script type="text/javascript">
<!--
    var count = 0;
    document.write("Starting Loop" + "<br/>");
    while (count < 10){
        document.write("Current Count : " + count + "<br/>");
        count++;
    }
    document.write("Loop stopped!");
//-->
</script>
```

```
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!
```

# For Loop

- The for loop is the most compact form of looping and 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 the given condition is true or not. If condition is true then code given inside the loop will be executed otherwise loop will come out.
- The iteration statement where you can increase or decrease your counter.

# Syntax and Example

```
for (initialization; test condition; iteration statement) {
    Statement(s) to be executed if test condition is true
}

<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>
```

```
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!
```

# JavaScript Functions

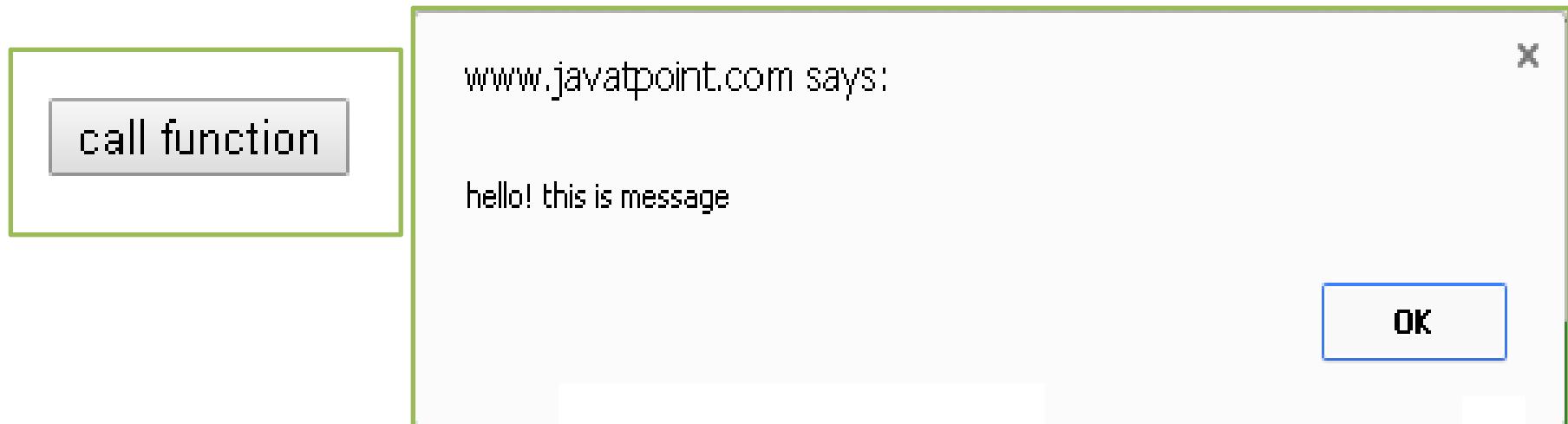
- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).
- **JavaScript functions** are used to perform operations. We can call JavaScript function many times to reuse the code.

# Advantage of JavaScript function

- There are mainly two advantages of JavaScript functions.
- **Code reusability:** We can call a function several times so it save coding.
- **Less coding:** It makes our program compact. We don't need to write many lines of code each time to perform a common task.

```
function functionName([arg1, arg2, ...argN]){
    //code to be executed
}
```

```
html>
<body>
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
</body>
</html>
```



# JavaScript Function Arguments

```
<html>
<body>
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcube(4)"/>
</form>
</body>
</html>
```

click

64

OK

# Function with Return Value

```
<html>
<body>
<script>
function getInfo(){
return "hello javatpoint! How r u?";
}
</script>
<script>
document.write(getInfo());
</script>
</body>
</html>
```

hello javatpoint! How r u?

# JavaScript Array

- **JavaScript array** is an object that represents a collection of similar type of elements.
- There are 3 ways to construct array in JavaScript
  - By array literal
  - By creating instance of Array directly (using new keyword)
  - By using an Array constructor (using new keyword)

# 1) JavaScript array literal:

```
var arrayname=[value1,value2....valueN];
```

```
<html>
<body>
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
</body>
</html>
```

Sonoo  
Vimal  
Ratan

# JavaScript Array directly (new keyword)

```
var arrayname=new Array();
```

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

Arun  
Varun  
John

# JavaScript array constructor (new keyword)

- create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

```
<html>
<body>
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
</body>
</html>
```

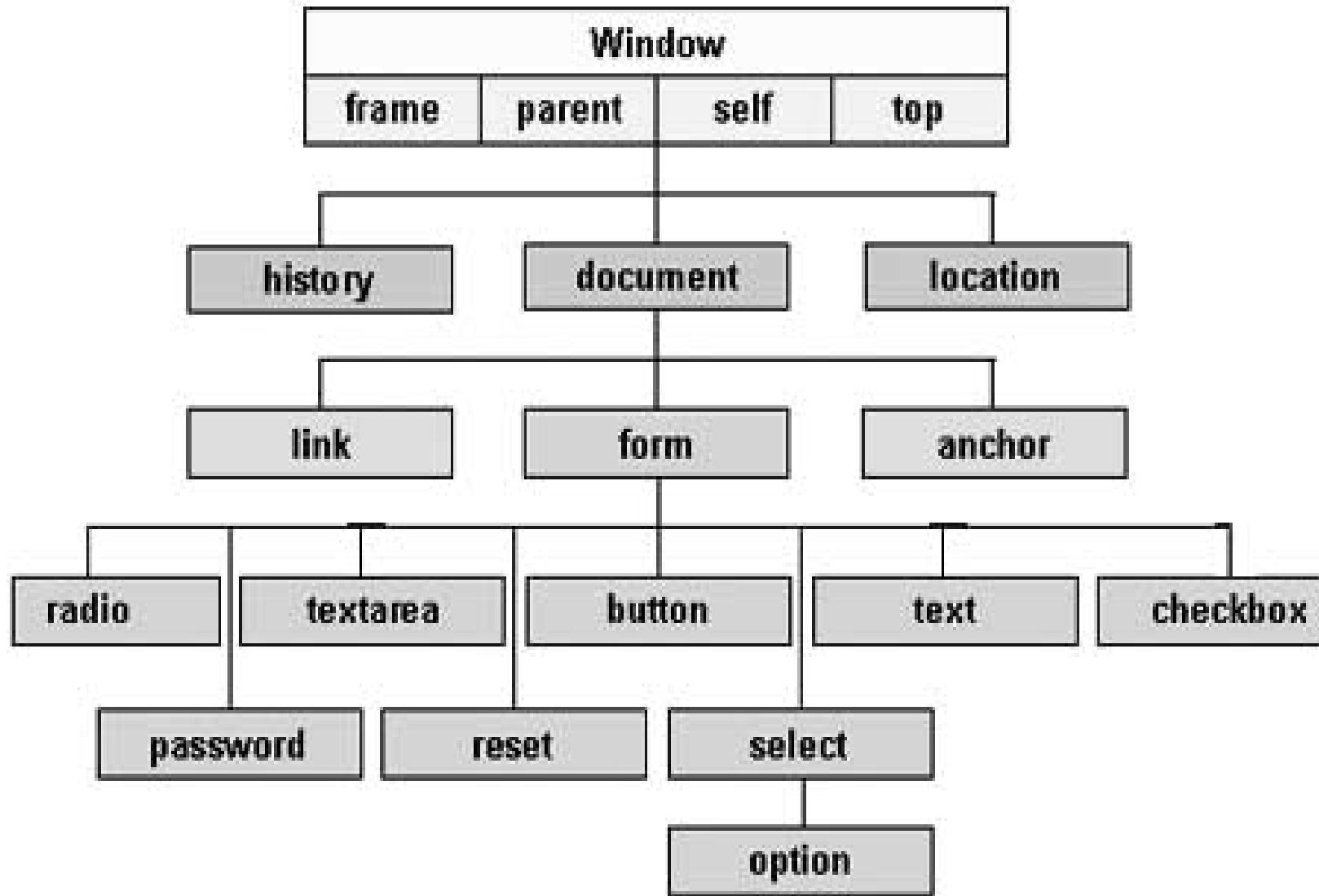
Jai  
Vijay  
Smith

# What is the DOM?

- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

- Every web page resides inside a browser window which can be considered as an object.
- A Document object represents the HTML document that is displayed in that window.
- The Document object has various properties that refer to other objects which allow access to and modification of document content.
- The way a document content is accessed and modified is called the **Document Object Model**, or **DOM**.
- The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.

- **Window object** – Top of the hierarchy. It is the outmost element of the object hierarchy.
- **Document object** – Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.
- **Form object** – Everything enclosed in the `<form>...</form>` tags sets the form object.
- **Form control elements** – The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.



# Date

- The **JavaScript date** object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.
- You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

# Date-Constructor

- 4 variant of Date constructor to create date object.
- Date objects are created with the **new Date()** constructor.
- Date()
- Date(milliseconds)
- Date(dateString)
- Date(year, month, day, hours, minutes, seconds, milliseconds)

- `new Date()` creates a new date object with the **current date and time**.
- `var d = new Date();`
- `new Date(year, month, ...)` `new Date(year, month, ...)` creates a new date object with a **specified date and time**.
- 7 numbers specify year, month, day, hour, minute, second, and millisecond (in that order)
- `var d = new Date(2018, 11, 24, 10, 33, 30, 0);`
- **Note:** JavaScript counts months from 0 to 11.
- January is 0. December is 11.

## JavaScript new Date()

Using `new Date(7 numbers)`, creates a new date object with the specified date and time:

Mon Dec 24 2018 10:33:30 GMT+0530 (India Standard Time)

- `new Date(dateString)` creates a new date object from a **date string**

```
var d = new Date("October 13, 2014 11:13:00");
```

## JavaScript new Date()

A Date object can be created with a specified date and time:

Mon Oct 13 2014 11:13:00 GMT+0530 (India Standard Time)

- `new Date(milliseconds)` creates a new date object as **zero time plus milliseconds**
- `var d = new Date(0);`

## JavaScript new Date()

Using `new Date(milliseconds)`, creates a new date object as January 1, 1970, 00:00:00 Universal Time (UTC) plus the milliseconds:

`Thu Jan 01 1970 05:30:00 GMT+0530 (India Standard Time)`

# 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 class to get the object. But, we direct create objects.

In real life, a car is an **object**.

A car has **properties** like weight and color, and **methods** like start and stop:

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

All cars have the same **properties**, but the property **values** differ from car to car.

All cars have the same **methods**, but the methods are performed **at different times**.

# Creating Objects in JavaScript

- There are 3 ways to create objects.
- By object literal
- By creating instance of Object directly (using new keyword)
- 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:
- `var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};`  
or
- `var person = {  
 firstName: "John",  
 lastName: "Doe",  
 age: 50,  
 eyeColor: "blue"  
};`

# Object Properties

- The **name:values** pairs in JavaScript objects are called **properties**

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

## 2) By creating instance of Object

- The syntax of creating object directly is given below:
- var objectname=new Object();**
- Here, **new keyword** is used to create object.

```
<html>
<body>
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
</body>
</html>
```

101 Ravi Malik 50000

- var person = new Object();  
person.firstName = "John";  
person.lastName = "Doe";  
person.age = 50;  
person.eyeColor = "blue";

### 3) By using an Object constructor

- Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.
- The **this keyword** refers to the current object.
-

```
<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>
```

103 Vimal Jaiswal 30000

Thank You

## **Javascript**

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.

### **Features of JavaScript**

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

### **Application of JavaScript**

JavaScript is used to create interactive websites. It is mainly used for:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

### **Example**

```
<html>
<body>
<h2>Welcome to JavaScript</h2>
<script>
```

```
document.write("Hello JavaScript by JavaScript");
</script>
</body>
</html>
```

### **3 Places to put JavaScript code**

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)

#### **Between the body tag**

```
<html>
<body>
<script type="text/javascript">
  alert("Hello Javatpoint");
</script>
</body>
</html>
```

#### **Between the head tag**

```
<html>
<head>
<script type="text/javascript">
function msg(){
  alert("Hello Javatpoint");
}
</script>
</head>
<body>
<p>Welcome to Javascript</p>
<form>
<input type="button" value="click" onclick="msg()"/>
```

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

## External javascript

Message.js

```
function msg(){
    alert("Hello Javatpoint");
}
```

```
<html>
<head>
<script type="text/javascript" src="message.js"></script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</body>
</html>
```

## 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**, means you don't need to specify type of the variable because it is dynamically used by 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:

```
var a=40;//holding number
```

```
var b="Rahul";//holding string
```

## Primitive data types

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

## Non-primitive data types

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

## JavaScript Operators

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

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 Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.

## JavaScript Functions

**JavaScript functions** are used to perform operations. We can call JavaScript function many times to reuse the code.

### Syntax

```
function functionName([arg1, arg2, ...argN]){\n    //code to be executed\n}
```

### example of function in JavaScript

```
<script>\nfunction msg(){\n    alert("hello! this is message");\n}
```

```
}
```

```
</script>
```

```
<input type="button" onclick="msg()" value="call function"/>
```

### **function by passing arguments**

```
<html>
```

```
<body>
```

```
<script>
```

```
function getcube(number){
```

```
    alert(number*number*number);
```

```
}
```

```
</script>
```

```
<form>
```

```
<input type="button" value="click" onclick="getcube(4)"/>
```

```
</form>
```

```
</body>
```

```
</html>
```

### **function that returns a value**

```
<html>
```

```
<body>
```

```
<script>
```

```
function getInfo(){
```

```
    return "hello javatpoint! How r u?";
```

```
}
```

```
</script>
```

```
<script>
```

```
document.write(getInfo());
```

```
</script>
```

```
</body>
```

```
</html>
```

## 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

```
new Function ([arg1[, arg2[, ....argn]]],] functionBody)
```

**arg1, arg2, .... , argn** - It represents the argument used by function.

**functionBody** - It represents the function definition

### Function Methods

Method	Description
apply()	It is used to call a function contains 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 contains this value and an argument list.
toString()	It returns the result in a form of a string.

### example to display the sum of given numbers

```
<!DOCTYPE html>

<html>
<body>

<script>
var add=new Function("num1","num2","return num1+num2");
document.writeln(add(2,5));
</script>
```

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

## 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 template based not class based. Here, we don't create class to get the object. But, we direct create objects

**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)

### **syntax of creating object using object literal**

object={property1:value1,property2:value2.....propertyN:valueN}

property and value is separated by : (colon).

### **example of creating object**

```
<html>  
<body>  
<script>  
emp={id:102,name:"Shyam Kumar",salary:40000}  
document.write(emp.id+" "+emp.name+" "+emp.salary);  
</script>  
</body>  
</html>
```

### **By creating instance of Object**

#### **syntax of creating object directly**

```
var objectname=new Object();
```

### **example of creating object directly**

```
<html>  
<body>  
<script>
```

```
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
</body>
</html>
```

### **By using an Object constructor**

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

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

The **this keyword** refers to the current object

### **Defining method in JavaScript Object**

We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.

example of defining method in object

```

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

```

### **JavaScript Object Methods**

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 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 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	Object.isSealed()	This method determines if an object is sealed.
16	Object.keys()	This method returns an array of a given object's

		own property names.
17	Object.preventExtensions()	This method is used to prevent any extensions of an object.
18	Object.seal()	This method prevents new properties from being added and marks all existing properties as non-configurable.
19	Object.setPrototypeOf()	This method sets the prototype of a specified object to another object.
20	Object.values()	This method returns an array of values.

## Array

**JavaScript array** is an object that represents a collection of similar type 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)

### **syntax of creating array using array literal**

```
var arrayname=[value1,value2.....valueN];
```

example of creating and using array in JavaScript

```
<html>
<body>
<script>

var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
</body>
</html>
```

### **syntax of creating array directly**

```
var arrayname=new Array();
```

**new keyword** is used to create instance of array

```
<html>
```

```
<body>
```

```
<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>
```

```
</body>
```

```
</html>
```

### **JavaScript array constructor**

#### **example of creating object by array constructor**

```
<html>
```

```
<body>
```

```
<script>
```

```
var emp=new Array("Jai","Vijay","Smith");
```

```
for (i=0;i<emp.length;i++){
```

```
document.write(emp[i] + "<br>");
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

## Array Methods

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()	<b>It creates a new array carrying the exact copy of another array element.</b>
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 ia an array.
join()	It joins the elements of an array as a string.
keys()	<b>It creates an iterator object that contains only the keys of the array, then loops through these keys.</b>
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 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 add/remove 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.

## Event Handling

The change in the state of an object is known as an **Event**. In html, there are various events which represents 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**.

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc

Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

### 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	Event Handler	Description

Performed		
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

### 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>
```

### **Exception Handling in JavaScript**

An exception signifies the presence of an abnormal condition which requires special operable techniques. In programming terms, an exception is the anomalous code that breaks the normal flow of the code. Such exceptions require specialized programming constructs for its execution.

## Types of Errors

While coding, there can be three types of errors in the code:

1. **Syntax Error:** When a user makes a mistake in the pre-defined syntax of a programming language, a syntax error may appear.
2. **Runtime Error:** When an error occurs during the execution of the program, such an error is known as Runtime error. The codes which create runtime errors are known as Exceptions. Thus, exception handlers are used for handling runtime errors.
3. **Logical Error:** An error which occurs when there is any logical mistake in the program that may not produce the desired output, and may terminate abnormally. Such an error is known as Logical error.

## Error Object

When a runtime error occurs, it creates and throws an Error object. Such an object can be used as a base for the user-defined exceptions too. An error object has two properties:

1. **name:** This is an object property that sets or returns an error name.
2. **message:** This property returns an error message in the string form.

Although Error is a generic constructor, there are following standard built-in error types or error constructors beside it:

1. **EvalError:** It creates an instance for the error that occurred in the eval(), which is a global function used for evaluating the js string code.
2. **InternalError:** It creates an instance when the js engine throws an internal error.
3. **RangeError:** It creates an instance for the error that occurs when a numeric variable or parameter is out of its valid range.
4. **ReferenceError:** It creates an instance for the error that occurs when an invalid reference is de-referenced.
5. **SyntaxError:** An instance is created for the syntax error that may occur while parsing the eval().
6. **TypeError:** When a variable is not a valid type, an instance is created for such an error.
7. **URIError:** An instance is created for the error that occurs when invalid parameters are passed in **encodeURI()** or **decodeURI()**.

There are following statements that handle if any exception occurs:

- o throw statements
- o try...catch statements
- o try...catch...finally statements.

## try...catch

A try...catch is a commonly used statement in various programming languages. Basically, it is used to handle the error-prone part of the code. It initially tests the code for all possible errors it may contain, then it implements actions to tackle those errors (if occur). A good programming approach is to keep the complex code within the try...catch statements.

**try{} statement:** Here, the code which needs possible error testing is kept within the try block. In case any error occur, it passes to the catch{} block for taking suitable actions and handle the error. Otherwise, it executes the code written within

**catch{} statement:** This block handles the error of the code by executing the set of statements written within the block. This block contains either the user-defined exception handler or the built-in handler. This block executes only when any error-prone code needs to be handled in the try block. Otherwise, the catch block is skipped.

### Syntax:

```
try{  
    expression;  
} //code to be written.  
catch(error){  
    expression;  
} // code for handling the error.
```

```
<html>  
    <head>  
        <script type = "text/javascript">  
            function myFunc() {  
                var a = 100;  
                try {  
                    alert("Value of variable a is : " + a );  
                }  
                catch ( e ) {  
                    alert("Error: " + e.description );  
                }  
            }  
        </script>
```

```
</head>

<body>
    <p>Click the following to see the result:</p>
    <form>
        <input type = "button" value = "Click Me" onclick = "myFunc();"/>
    </form>

</body>
</html>
```

### **throw Statement**

Throw statements are used for throwing user-defined errors. User can define and throw their own custom errors. When throw statement is executed, the statements present after it will not execute. The control will directly pass to the catch block.

### **Syntax**

```
throw exception;

try{
    throw exception; // user can define their own exception
}

catch(error){
    expression; } // code for handling exception.
```

The exception can be a string, number, object, or boolean value.

```
<html>
    <head>Exception Handling</head>
    <body>
        <script>
            try {
                throw new Error('This is the throw keyword'); //user-defined throw statement.
            }
            catch (e) {
                document.write(e.message); // This will generate an error message
            }
        </script>
    </body>
</html>
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

### **try...catch...finally statements**

Finally is an optional block of statements which is executed after the execution of try and catch statements. Finally block does not hold for the exception to be thrown. Any exception is thrown or not, finally block code, if present, will definitely execute. It does not care for the output too

#### **Syntax**

```
try{
```

```
    expression;
```

```
}
```

```
catch(error){
```

```
    expression;
```

```
}
```

```
finally{
```

```
    expression; } //Executable code
```

#### **Example**

```
<html>
```

```
<head>Exception Handling</head>
```

```
<body>
```

```
<script>
```

```
try{
```

```
    var a=2;
```

```
    if(a==2)
```

```
        document.write("ok");
```

```
}
```

```
catch(Error){
```

```
    document.write("Error found"+e.message);
```

```
}

finally{
    document.write("Value of a is 2 ");
}

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

**Reference:** The contents are taken from [www.javatpoint.com](http://www.javatpoint.com)

## JavaScript – Objects

JavaScript has several built-in or native objects. These objects are accessible anywhere in your program and will work the same way in any browser running in any operating system.

Here is the list of all important JavaScript Native Objects –

- JavaScript Number Object
- JavaScript Boolean Object
- JavaScript String Object
- JavaScript Array Object
- JavaScript Date Object
- JavaScript Math Object
- JavaScript RegExp Object

## The Number Object

The **Number** object represents numerical date, either integers or floating-point numbers. In general, you do not need to worry about **Number** objects because the browser automatically converts number literals to instances of the number class.

### Syntax

The syntax for creating a **number** object is as follows –

```
var val = new Number(number);
```

In the place of number, if you provide any non-number argument, then the argument cannot be converted into a number, it returns **NaN** (Not-a-Number).

### Number Properties

Here is a list of each property and their description.

Property	Description
<a href="#"><u>MAX_VALUE</u></a>	The largest possible value a number in JavaScript can have 1.7976931348623157E+308
<a href="#"><u>MIN_VALUE</u></a>	The smallest possible value a number in JavaScript can have 5E-324
<a href="#"><u>NaN</u></a>	Equal to a value that is not a number.
<a href="#"><u>NEGATIVE_INFINITY</u></a>	A value that is less than MIN_VALUE.

<a href="#">POSITIVE_INFINITY</a>	A value that is greater than MAX_VALUE
<a href="#">prototype</a>	A static property of the Number object. Use the prototype property to assign new properties and methods to the Number object in the current document
<a href="#">constructor</a>	Returns the function that created this object's instance. By default this is the Number object.

In the following sections, we will take a few examples to demonstrate the properties of Number.

## Number Methods

The Number object contains only the default methods that are a part of every object's definition.

Method	Description
<a href="#">toExponential()</a>	Forces a number to display in exponential notation, even if the number is in the range in which JavaScript normally uses standard notation.
<a href="#">toFixed()</a>	Formats a number with a specific number of digits to the right of the decimal.
<a href="#">toLocaleString()</a>	Returns a string value version of the current number in a format that may vary according to a browser's local settings.
<a href="#">toPrecision()</a>	Defines how many total digits (including digits to the left and right of the decimal) to display of a number.
<a href="#">toString()</a>	Returns the string representation of the number's value.
<a href="#">valueOf()</a>	Returns the number's value.

## The Boolean Object

The **Boolean** object represents two values, either "true" or "false". If *value* parameter is omitted or is 0, -0, null, false, **NaN**, undefined, or the empty string (""), the object has an initial value of false.

### Syntax

Use the following syntax to create a **boolean** object.

```
var val = new Boolean(value);
```

## Boolean Properties

Here is a list of the properties of Boolean object –

Property	Description
<a href="#">constructor</a>	Returns a reference to the Boolean function that created the object.
<a href="#">prototype</a>	The prototype property allows you to add properties and methods to an object.

In the following sections, we will have a few examples to illustrate the properties of Boolean object.

## Boolean Methods

Here is a list of the methods of Boolean object and their description.

Method	Description
<a href="#">toSource()</a>	Returns a string containing the source of the Boolean object; you can use this string to create an equivalent object.
<a href="#">toString()</a>	Returns a string of either "true" or "false" depending upon the value of the object.
<a href="#">valueOf()</a>	Returns the primitive value of the Boolean object.

## The Strings Object

The **String** object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods.

As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive.

### Syntax

Use the following syntax to create a String object –

```
var val = new String(string);
```

The **String** parameter is a series of characters that has been properly encoded.

## String Properties

Here is a list of the properties of String object and their description.

Property	Description
<a href="#">constructor</a>	Returns a reference to the String function that created the object.
<a href="#">length</a>	Returns the length of the string.
<a href="#">prototype</a>	The prototype property allows you to add properties and methods to an object.

In the following sections, we will have a few examples to demonstrate the usage of String properties.

## String Methods

Here is a list of the methods available in String object along with their description.

Method	Description
<a href="#">charAt()</a>	Returns the character at the specified index.
<a href="#">charCodeAt()</a>	Returns a number indicating the Unicode value of the character at the given index.
<a href="#">concat()</a>	Combines the text of two strings and returns a new string.
<a href="#">indexOf()</a>	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
<a href="#">lastIndexOf()</a>	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
<a href="#">localeCompare()</a>	Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.
<a href="#">match()</a>	Used to match a regular expression against a string.
<a href="#">replace()</a>	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
<a href="#">search()</a>	Executes the search for a match between a regular expression and a

	specified string.
<a href="#">slice()</a>	Extracts a section of a string and returns a new string.
<a href="#">split()</a>	Splits a String object into an array of strings by separating the string into substrings.
<a href="#">substr()</a>	Returns the characters in a string beginning at the specified location through the specified number of characters.
<a href="#">substring()</a>	Returns the characters in a string between two indexes into the string.
<a href="#">toLocaleLowerCase()</a>	The characters within a string are converted to lower case while respecting the current locale.
<a href="#">toLocaleUpperCase()</a>	The characters within a string are converted to upper case while respecting the current locale.
<a href="#">toLowerCase()</a>	Returns the calling string value converted to lower case.
<a href="#">toString()</a>	Returns a string representing the specified object.
<a href="#">toUpperCase()</a>	Returns the calling string value converted to uppercase.
<a href="#">valueOf()</a>	Returns the primitive value of the specified object.

## String HTML Wrappers

Here is a list of the methods that return a copy of the string wrapped inside an appropriate HTML tag.

Method	Description
<a href="#">anchor()</a>	Creates an HTML anchor that is used as a hypertext target.
<a href="#">big()</a>	Creates a string to be displayed in a big font as if it were in a <big> tag.
<a href="#">blink()</a>	Creates a string to blink as if it were in a <blink> tag.
<a href="#">bold()</a>	Creates a string to be displayed as bold as if it were in a <b> tag.
<a href="#">fixed()</a>	Causes a string to be displayed in fixed-pitch font as if it were in a <tt> tag
<a href="#">fontcolor()</a>	Causes a string to be displayed in the specified color as if it were in a <font>

	color="color"> tag.
<a href="#"><u>fontsize()</u></a>	Causes a string to be displayed in the specified font size as if it were in a <font size="size"> tag.
<a href="#"><u>italics()</u></a>	Causes a string to be italic, as if it were in an <i> tag.
<a href="#"><u>link()</u></a>	Creates an HTML hypertext link that requests another URL.
<a href="#"><u>small()</u></a>	Causes a string to be displayed in a small font, as if it were in a <small> tag.
<a href="#"><u>strike()</u></a>	Causes a string to be displayed as struck-out text, as if it were in a <strike> tag.
<a href="#"><u>sub()</u></a>	Causes a string to be displayed as a subscript, as if it were in a <sub> tag
<a href="#"><u>sup()</u></a>	Causes a string to be displayed as a superscript, as if it were in a <sup> tag

## The Arrays Object

The **Array** object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

### Syntax

Use the following syntax to create an **Array** object –

```
var fruits = new Array( "apple", "orange", "mango" );
```

The **Array** parameter is a list of strings or integers. When you specify a single numeric parameter with the Array constructor, you specify the initial length of the array. The maximum length allowed for an array is 4,294,967,295.

You can create array by simply assigning values as follows –

```
var fruits = [ "apple", "orange", "mango" ];
```

You will use ordinal numbers to access and to set values inside an array as follows.

```
fruits[0] is the first element  
fruits[1] is the second element  
fruits[2] is the third element
```

## Array Properties

Here is a list of the properties of the Array object along with their description.

Property	Description
<a href="#">constructor</a>	Returns a reference to the array function that created the object.
<a href="#">index</a>	The property represents the zero-based index of the match in the string
<a href="#">input</a>	This property is only present in arrays created by regular expression matches.
<a href="#">length</a>	Reflects the number of elements in an array.
<a href="#">prototype</a>	The prototype property allows you to add properties and methods to an object.

In the following sections, we will have a few examples to illustrate the usage of Array properties.

## Array Methods

Here is a list of the methods of the Array object along with their description.

Method	Description
<a href="#">concat()</a>	Returns a new array comprised of this array joined with other array(s) and/or value(s).
<a href="#">every()</a>	Returns true if every element in this array satisfies the provided testing function.
<a href="#">filter()</a>	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
<a href="#">forEach()</a>	Calls a function for each element in the array.
<a href="#">indexOf()</a>	Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
<a href="#">join()</a>	Joins all elements of an array into a string.
<a href="#">lastIndexOf()</a>	Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.
<a href="#">map()</a>	Creates a new array with the results of calling a provided function on every

	element in this array.
<a href="#">pop()</a>	Removes the last element from an array and returns that element.
<a href="#">push()</a>	Adds one or more elements to the end of an array and returns the new length of the array.
<a href="#">reduce()</a>	Apply a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.
<a href="#">reduceRight()</a>	Apply a function simultaneously against two values of the array (from right-to-left) as to reduce it to a single value.
<a href="#">reverse()</a>	Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
<a href="#">shift()</a>	Removes the first element from an array and returns that element.
<a href="#">slice()</a>	Extracts a section of an array and returns a new array.
<a href="#">some()</a>	Returns true if at least one element in this array satisfies the provided testing function.
<a href="#">toSource()</a>	Represents the source code of an object
<a href="#">sort()</a>	Represents the source code of an object
<a href="#">splice()</a>	Adds and/or removes elements from an array.
<a href="#">toString()</a>	Returns a string representing the array and its elements.
<a href="#">unshift()</a>	Adds one or more elements to the front of an array and returns the new length of the array.

## The Date Object

The Date object is a datatype built into the JavaScript language. Date objects are created with the **new Date( )** as shown below.

Once a Date object is created, a number of methods allow you to operate on it. Most methods simply allow you to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal, or GMT) time.

## Syntax

You can use any of the following syntaxes to create a Date object using Date() constructor.

```
new Date( )
new Date(milliseconds)
new Date(datestring)
new Date(year,month,date[,hour,minute,second,millisecond ])
```

## Date Methods

Here is a list of the methods used with **Date** and their description.

Method	Description
<a href="#">Date()</a>	Returns today's date and time
<a href="#">getDate()</a>	Returns the day of the month for the specified date according to local time.
<a href="#">getDay()</a>	Returns the day of the week for the specified date according to local time.
<a href="#">getFullYear()</a>	Returns the year of the specified date according to local time.
<a href="#">getHours()</a>	Returns the hour in the specified date according to local time.
<a href="#">getMilliseconds()</a>	Returns the milliseconds in the specified date according to local time.
<a href="#">getMinutes()</a>	Returns the minutes in the specified date according to local time.
<a href="#">getMonth()</a>	Returns the month in the specified date according to local time.
<a href="#">getSeconds()</a>	Returns the seconds in the specified date according to local time.
<a href="#">getTime()</a>	Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC.
<a href="#">getTimezoneOffset()</a>	Returns the time-zone offset in minutes for the current locale.
<a href="#">getUTCDate()</a>	Returns the day (date) of the month in the specified date according to universal time.
<a href="#">getUTCDay()</a>	Returns the day of the week in the specified date according to universal time.

<a href="#">getUTCFullYear()</a>	Returns the year in the specified date according to universal time.
<a href="#">getUTCHours()</a>	Returns the hours in the specified date according to universal time.
<a href="#">getUTCMilliseconds()</a>	Returns the milliseconds in the specified date according to universal time.
<a href="#">getUTCMilliseconds()</a>	Returns the minutes in the specified date according to universal time.
<a href="#">getUTCMonth()</a>	Returns the month in the specified date according to universal time.
<a href="#">getUTCSeconds()</a>	Returns the seconds in the specified date according to universal time.
<a href="#">getYear()</a>	<b>Deprecated</b> - Returns the year in the specified date according to local time. Use <code>getFullYear</code> instead.
<a href="#"> setDate()</a>	Sets the day of the month for a specified date according to local time.
<a href="#">setFullYear()</a>	Sets the full year for a specified date according to local time.
<a href="#">setHours()</a>	Sets the hours for a specified date according to local time.
<a href="#">setMilliseconds()</a>	Sets the milliseconds for a specified date according to local time.
<a href="#">setMinutes()</a>	Sets the minutes for a specified date according to local time.
<a href="#">setMonth()</a>	Sets the month for a specified date according to local time.
<a href="#">setSeconds()</a>	Sets the seconds for a specified date according to local time.
<a href="#"> setTime()</a>	Sets the Date object to the time represented by a number of milliseconds since January 1, 1970, 00:00:00 UTC.
<a href="#">setUTCDate()</a>	Sets the day of the month for a specified date according to universal time.
<a href="#">setUTCFullYear()</a>	Sets the full year for a specified date according to universal time.
<a href="#">setUTCHours()</a>	Sets the hour for a specified date according to universal time.
<a href="#">setUTCMilliseconds()</a>	Sets the milliseconds for a specified date according to universal time.
<a href="#">setUTCMilliseconds()</a>	Sets the minutes for a specified date according to universal time.
<a href="#">setUTCMonth()</a>	Sets the month for a specified date according to universal time.

<a href="#"><u>setUTCSeconds()</u></a>	Sets the seconds for a specified date according to universal time.
<a href="#"><u>setYear()</u></a>	<b>Deprecated</b> - Sets the year for a specified date according to local time. Use setFullYear instead.
<a href="#"><u>toDateString()</u></a>	Returns the "date" portion of the Date as a human-readable string.
<a href="#"><u>toGMTString()</u></a>	<b>Deprecated</b> - Converts a date to a string, using the Internet GMT conventions. Use toUTCString instead.
<a href="#"><u>toLocaleDateString()</u></a>	Returns the "date" portion of the Date as a string, using the current locale's conventions.
<a href="#"><u>toLocaleFormat()</u></a>	Converts a date to a string, using a format string.
<a href="#"><u>toLocaleString()</u></a>	Converts a date to a string, using the current locale's conventions.
<a href="#"><u>toLocaleTimeString()</u></a>	Returns the "time" portion of the Date as a string, using the current locale's conventions.
<a href="#"><u>toSource()</u></a>	Returns a string representing the source for an equivalent Date object; you can use this value to create a new object.
<a href="#"><u>toString()</u></a>	Returns a string representing the specified Date object.
<a href="#"><u>toTimeString()</u></a>	Returns the "time" portion of the Date as a human-readable string.
<a href="#"><u>toUTCString()</u></a>	Converts a date to a string, using the universal time convention.
<a href="#"><u>valueOf()</u></a>	Returns the primitive value of a Date object.

Example :

```
function start()
{
    var current = new Date();

    // string formatting methods and valueOf
    document.getElementById( "strings" ).innerHTML =
        "<p>toString: " + current.toString() + "</p>" +
        "<p>toLocaleString: " + current.toLocaleString() + "</p>" +
        "<p>toUTCString: " + current.toUTCString() + "</p>" +
        "<p>valueOf: " + current.valueOf() + "</p>";

    // get methods
    document.getElementById( "getMethods" ).innerHTML =
```

```

"<p>getDate: " + current.getDate() + "</p>" +
"<p>getDay: " + current.getDay() + "</p>" +
"<p>getMonth: " + current.getMonth() + "</p>" +
"<p>getFullYear: " + current.getFullYear() + "</p>" +
"<p>getTime: " + current.getTime() + "</p>" +
"<p>getHours: " + current.getHours() + "</p>" +
"<p>getMinutes: " + current.getMinutes() + "</p>" +
"<p>getSeconds: " + current.getSeconds() + "</p>" +
"<p>getMilliseconds: " + current.getMilliseconds() + "</p>" +
"<p>getTimezoneOffset: " + current.getTimezoneOffset() + "</p>";

// creating a Date
var anotherDate = new Date( 2011, 2, 18, 1, 5, 0, 0 );
document.getElementById( "newArguments" ).innerHTML =
"<p>Date: " + anotherDate + "</p>";

// set methods
anotherDate.setDate( 31 );
anotherDate.setMonth( 11 );
anotherDate.setFullYear( 2011 );
anotherDate.setHours( 23 );
anotherDate.setMinutes( 59 );
anotherDate.setSeconds( 59 );
document.getElementById( "setMethods" ).innerHTML =
"<p>Modified date: " + anotherDate + "</p>";
} // end function start

window.addEventListener( "load", start, false );

```

## The Math Object

The **math** object provides you properties and methods for mathematical constants and functions. Unlike other global objects, **Math** is not a constructor. All the properties and methods of **Math** are static and can be called by using Math as an object without creating it.

### Syntax

The syntax to call the properties and methods of Math are as follows

```
var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

### Math Properties

Here is a list of all the properties of Math and their description.

Property	Description
E	Euler's constant and the base of natural logarithms, approximately 2.718.

<a href="#">LN2</a>	Natural logarithm of 2, approximately 0.693.
<a href="#">LN10</a>	Natural logarithm of 10, approximately 2.302.
<a href="#">LOG2E</a>	Base 2 logarithm of E, approximately 1.442.
<a href="#">LOG10E</a>	Base 10 logarithm of E, approximately 0.434.
<a href="#">PI</a>	Ratio of the circumference of a circle to its diameter, approximately 3.14159.
<a href="#">SQRT1_2</a>	Square root of 1/2; equivalently, 1 over the square root of 2, approximately 0.707.
<a href="#">SQRT2</a>	Square root of 2, approximately 1.414.

In the following sections, we will have a few examples to demonstrate the usage of Math properties.

## Math Methods

Here is a list of the methods associated with Math object and their description

Method	Description
<a href="#">abs()</a>	Returns the absolute value of a number.
<a href="#">acos()</a>	Returns the arccosine (in radians) of a number.
<a href="#">asin()</a>	Returns the arcsine (in radians) of a number.
<a href="#">atan()</a>	Returns the arctangent (in radians) of a number.
<a href="#">atan2()</a>	Returns the arctangent of the quotient of its arguments.
<a href="#">ceil()</a>	Returns the smallest integer greater than or equal to a number.
<a href="#">cos()</a>	Returns the cosine of a number.
<a href="#">exp()</a>	Returns $E^N$ , where N is the argument, and E is Euler's constant, the base of the natural logarithm.
<a href="#">floor()</a>	Returns the largest integer less than or equal to a number.
<a href="#">log()</a>	Returns the natural logarithm (base E) of a number.

<a href="#"><u>max()</u></a>	Returns the largest of zero or more numbers.
<a href="#"><u>min()</u></a>	Returns the smallest of zero or more numbers.
<a href="#"><u>pow()</u></a>	Returns base to the exponent power, that is, base exponent.
<a href="#"><u>random()</u></a>	Returns a pseudo-random number between 0 and 1.
<a href="#"><u>round()</u></a>	Returns the value of a number rounded to the nearest integer.
<a href="#"><u>sin()</u></a>	Returns the sine of a number.
<a href="#"><u>sqrt()</u></a>	Returns the square root of a number.
<a href="#"><u>tan()</u></a>	Returns the tangent of a number.
<a href="#"><u>toSource()</u></a>	Returns the string "Math".

## Regular Expressions and RegExp Object

A regular expression is an object that describes a pattern of characters.

The JavaScript **RegExp** class represents regular expressions, and both **String** and **RegExp** define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.

### Syntax

A regular expression could be defined with the **RegExp ()** constructor, as follows –

```
var pattern = new RegExp(pattern, attributes);
```

or simply

```
var pattern = /pattern/attributes;
```

Here is the description of the parameters –

- **pattern** – A string that specifies the pattern of the regular expression or another regular expression.
- **attributes** – An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multiline matches, respectively.

## Brackets

Brackets ([])) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

Expression	Description
[...]	Any one character between the brackets.
[^...]	Any one character not between the brackets.
[0-9]	It matches any decimal digit from 0 through 9.
[a-z]	It matches any character from lowercase a through lowercase z.
[A-Z]	It matches any character from uppercase A through uppercase Z.
[a-Z]	It matches any character from lowercase a through uppercase Z.

The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from **b** through **v**.

## Quantifiers

The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character has a specific connotation. The +, \*, ?, and \$ flags all follow a character sequence.

Expression	Description
p+	It matches any string containing at least one p.
p*	It matches any string containing zero or more p's.
p?	It matches any string containing one or more p's.
p{N}	It matches any string containing a sequence of <b>N</b> p's
p{2,3}	It matches any string containing a sequence of two or three p's.
p{2, }	It matches any string containing a sequence of at least two p's.
p\$	It matches any string with p at the end of it.

<code>^p</code>	It matches any string with p at the beginning of it.
-----------------	--

## Examples

Following examples explain more about matching characters.

Expression	Description
<code>[^a-zA-Z]</code>	It matches any string not containing any of the characters ranging from <b>a</b> through <b>z</b> and <b>A</b> through <b>Z</b> .
<code>p.p</code>	It matches any string containing <b>p</b> , followed by any character, in turn followed by another <b>p</b> .
<code>^.{2}\$</code>	It matches any string containing exactly two characters.
<code>&lt;b&gt;(.*)&lt;/b&gt;</code>	It matches any string enclosed within <code>&lt;b&gt;</code> and <code>&lt;/b&gt;</code> .
<code>p(hp)*</code>	It matches any string containing a <b>p</b> followed by zero or more instances of the sequence <b>hp</b> .

## Literal characters

Character	Description
Alphanumeric	Itself
<code>\0</code>	The NUL character ( <code>\u0000</code> )
<code>\t</code>	Tab ( <code>\u0009</code> )
<code>\n</code>	Newline ( <code>\u000A</code> )
<code>\v</code>	Vertical tab ( <code>\u000B</code> )
<code>\f</code>	Form feed ( <code>\u000C</code> )
<code>\r</code>	Carriage return ( <code>\u000D</code> )
<code>\xnn</code>	The Latin character specified by the hexadecimal number nn; for example, <code>\x0A</code> is the same as <code>\n</code>
<code>\uxxxx</code>	The Unicode character specified by the hexadecimal number xxxx; for example, <code>\u0009</code> is the same as <code>\t</code>

\cx	The control character ^X; for example, \cJ is equivalent to the newline character \n
-----	--

## Metacharacters

A metacharacter is simply an alphabetical character preceded by a backslash that acts to give the combination a special meaning.

For instance, you can search for a large sum of money using the '\d' metacharacter:  
/(\d+)000/. Here \d will search for any string of numerical character.

The following table lists a set of metacharacters which can be used in PERL Style Regular Expressions.

Character	Description
.	a single character
\s	a whitespace character (space, tab, newline)
\S	non-whitespace character
\d	a digit (0-9)
\D	a non-digit
\w	a word character (a-z, A-Z, 0-9, _)
\W	a non-word character
[\b]	a literal backspace (special case).
[aeiou]	matches a single character in the given set
[^aeiou]	matches a single character outside the given set
(foo bar baz)	matches any of the alternatives specified

## Modifiers

Several modifiers are available that can simplify the way you work with **regexp**s, like case sensitivity, searching in multiple lines, etc.

Modifier	Description

i	Perform case-insensitive matching.
m	Specifies that if the string has newline or carriage return characters, the ^ and \$ operators will now match against a newline boundary, instead of a string boundary
g	Performs a global match that is, find all matches rather than stopping after the first match.

## RegExp Properties

Here is a list of the properties associated with RegExp and their description.

Property	Description
<a href="#">constructor</a>	Specifies the function that creates an object's prototype.
<a href="#">global</a>	Specifies if the "g" modifier is set.
<a href="#">ignoreCase</a>	Specifies if the "i" modifier is set.
<a href="#">lastIndex</a>	The index at which to start the next match.
<a href="#">multiline</a>	Specifies if the "m" modifier is set.
<a href="#">source</a>	The text of the pattern.

In the following sections, we will have a few examples to demonstrate the usage of RegExp properties.

## RegExp Methods

Here is a list of the methods associated with RegExp along with their description.

Method	Description
<a href="#">exec()</a>	Executes a search for a match in its string parameter.
<a href="#">test()</a>	Tests for a match in its string parameter.
<a href="#">toSource()</a>	Returns an object literal representing the specified object; you can use this value to create a new object.
<a href="#">toString()</a>	Returns a string representing the specified object.

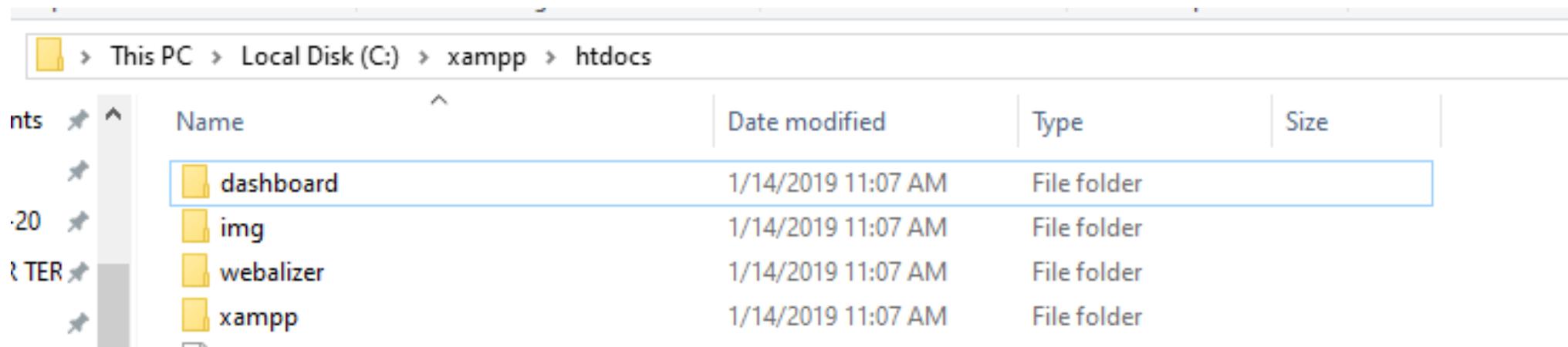
# PHP

## How to write my first php script

# Run Your First PHP Script

Step 1: Go to XAMPP server directory

I'm using Windows, so my root server directory is "C:\xampp\htdocs".



## Step 2: Create hello.php

Create a file and name it "hello.php"

 favicon	7/16/2015 9:02 PM	Icon	31 KB
 hello	7/16/2015 9:02 PM	PHP File	1 KB
 index	7/16/2015 9:02 PM	PHP File	1 KB

## Step 3: Code Inside hello.php

Open hello.php and put the following code.

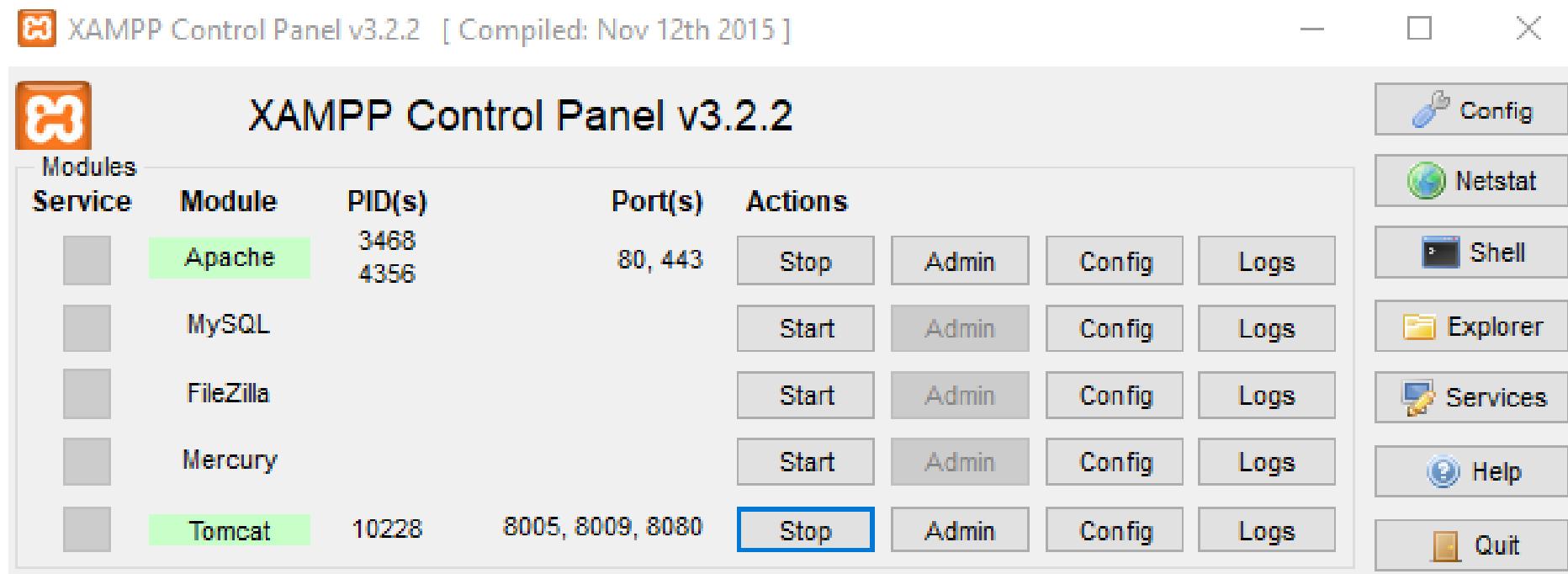


The image shows a screenshot of a Windows Notepad window titled '\*hello - Notepad'. The menu bar includes File, Edit, Format, View, and Help. The main text area contains the following PHP code:

```
<?php  
echo "Hello World!";  
?>
```

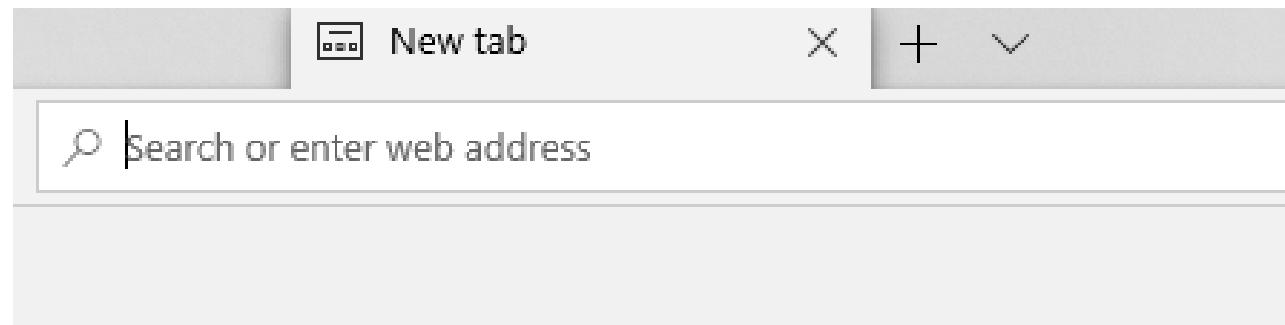
## Step 4: Run your XAMPP server

Open xampp application and click start required module



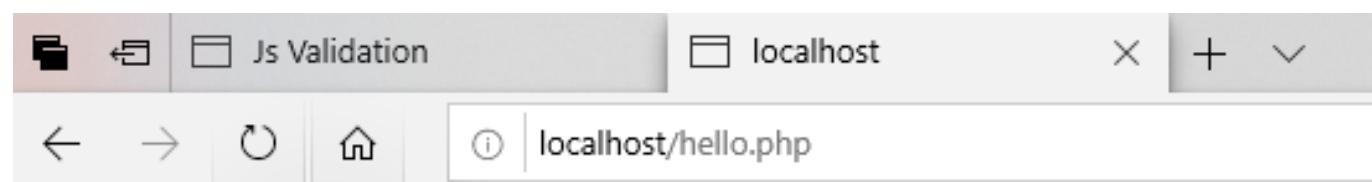
## Step 5: Open New Tab

Run it by opening a new tab in your browser



## Step 6: Load hello.php

In your browser window, type `http://localhost/hello.php`



## "Hello World" Script in PHP

```
<html>  
  <head>  
    <title>Hello World</title>  
  </head>  
  
  <body>  
    <?php echo "Hello, World!";?>  
  </body>  
  
</html>
```

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

A PHP script can be placed anywhere in the document.

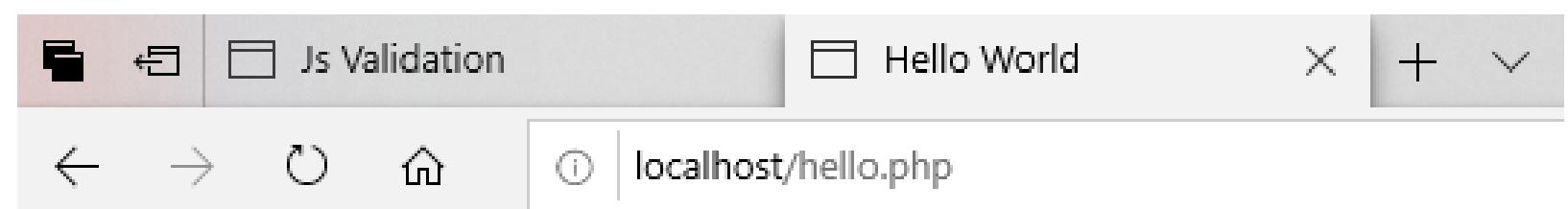
A PHP script starts with **<?php** and ends with **?>**:

```
<html>
```

```
  <head>
    <title>Hello World</title>
  </head>
```

```
  <body>
    <?php echo "Hello, World!";?>
  </body>
```

```
</html>
```



Hello, World!

# What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

# **PHP - Environment Setup**

Web Server

Database

PHP Parser

## **Canonical PHP tags**

The most universally effective PHP tag style is –

<?php...?>

# **Commenting PHP Code**

Single-line comments -//  
Multi-lines comments -/\*.....\*/

## **PHP is whitespace insensitive**

PHP whitespace insensitive means that it almost never matters how many whitespace characters you have in a row

## **PHP is case sensitive**

Yeah it is true that PHP is a case sensitive language.

Statements are expressions terminated by semicolons

# **Expressions are combinations of tokens**

The smallest building blocks of PHP are the indivisible tokens, such as numbers (3.14159), strings (.two.), variables (\$two), constants (TRUE), and the special words that make up the syntax of PHP itself like if, else, while, for and so forth

## **Braces make blocks**

We can always put a sequence of statements anywhere a statement can go by enclosing them in a set of curly braces.

# **PHP - Variable Types**

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

# **Variable Naming**

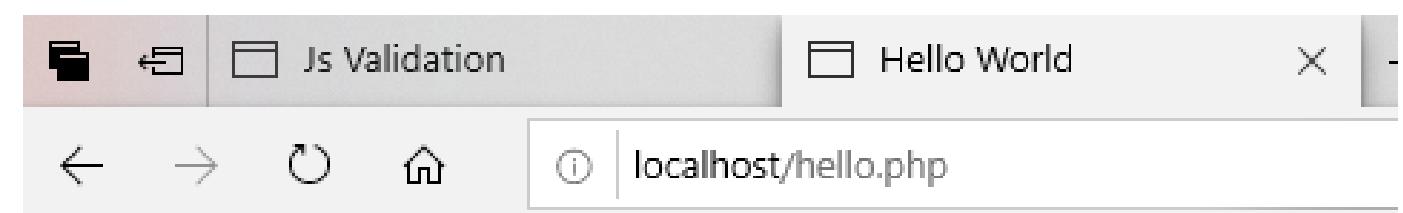
Rules for naming a variable is –

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , ( , ) . & , etc

- **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).

- If the value is a number, it is false if exactly equal to zero and true otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <?php
      $x=15;$y=30;
      echo "Sum: ",$x+$y;
    ?>
  </body>
</html>
```



# **Variable Scope**

Scope can be defined as the range of availability a variable has to the program in which it is declared.

Local variables

Function parameters

Global variables

Static variables

```
<?php  
$x = 5;  
$y = 10;  
  
function myTest() {  
    global $x, $y;  
    $y = $x + $y;  
}  
  
myTest();  
echo $y; // outputs 15  
?>
```

```
<?php  
function myTest() {  
    static $x = 0;  
    echo $x;  
    $x++;  
}  
myTest();  
myTest();  
myTest();  
?>
```

Output

0  
1  
2

# **PHP - Constants Types**

A constant is a name or an identifier for a simple value.

A constant value cannot change during the execution of the script.

By default, a constant is case-sensitive.

By convention, constant identifiers are always uppercase.

A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

To define a constant you have to use define() function and to retrieve the value of a constant, you have to simply specifying its name.

```
<?php
define("MINSIZE", 50);

echo MINSIZE;
echo constant("MINSIZE"); // same thing as the previous line
?>
```

## **PHP - Operator Types**

Arithmetric Operators

Comparison Operators

Logical (or Relational) Operators

Assignment Operators

Conditional (or ternary) Operators

## Precedence of PHP Operators

Category	Operator	Associativity
Unary	<code>! ++ --</code>	Right to left
Multiplicative	<code>* / %</code>	Left to right
Additive	<code>+ -</code>	Left to right
Relational	<code>&lt; &lt;= &gt; &gt;=</code>	Left to right
Equality	<code>== !=</code>	Left to right
Logical AND	<code>&amp;&amp;</code>	Left to right
Logical OR	<code>  </code>	Left to right
Conditional	<code>?:</code>	Right to left
Assignment	<code>= += -= *= /= %=</code>	Right to left

Example for practice

```
<?php
    $a = 42;
    $b = 20;
    $c = $a + $b;
    echo "Addtion Operation Result: $c <br/>";
    $c = $a - $b;
    echo "Substraction Operation Result: $c <br/>";
    $c = $a * $b;
    echo "Multiplication Operation Result: $c <br/>";
    $c = $a / $b;
    echo "Division Operation Result: $c <br/>";
    $c = $a % $b;
    echo "Modulus Operation Result: $c <br/>";
    $c = $a++;
```

# PHP - Decision Making

**if...else statement** – use this statement if you want to execute a set of code when a condition is true and another if the condition is not true

**elseif statement** – is used with the if...else statement to execute a set of code if **one** of the several condition is true

**switch statement** – is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

**Syntax remain same as C language**

Example for practice

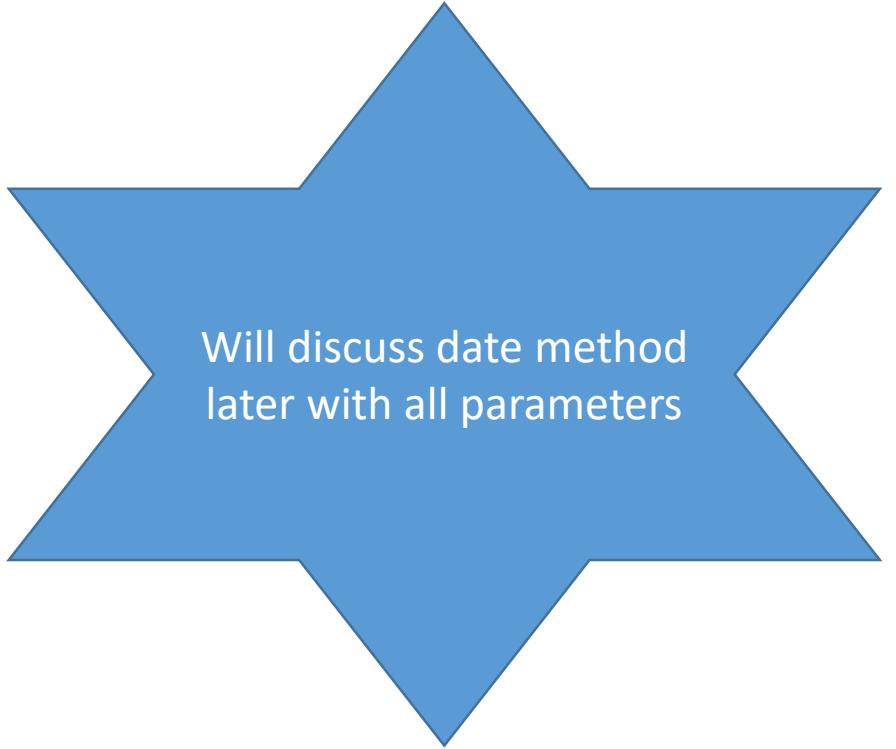
```
<?php
    $d = date("D");//date will return the currrent day

    if ($d == "Fri")
        echo "Have a nice weekend!";

    elseif ($d == "Sun")
        echo "Have a nice Sunday!";

    else
        echo "Have a nice day!";

?>
```



Will discuss date method  
later with all parameters

# **PHP - Loop Types**

**for** – loops through a block of code a specified number of times.

**while** – loops through a block of code if and as long as a specified condition is true.

**do...while** – loops through a block of code once, and then repeats the loop as long as a special condition is true.

**foreach** – loops through a block of code for each element in an array.

Example for practice

```
<?php
    $a = 0;
    $b = 0;

    for( $i = 0; $i<5; $i++ ) {
        $a += 10;
        $b += 5;
    }

    echo ("At the end of the loop a = $a and b = $b" );
?>
```

```
<html>
<body>
<?php
$i = 0;
$num = 50;

while( $i < 10 ) {
    $num--;
    $i++;
}

echo ("Loop stopped at i = $i and num = $num" );
?>
</body>
</html>
```

```
<html>
<body>
<?php
$i = 0;
$num = 0;

do {
    $i++;
}

while( $i < 10 );
echo ("Loop stopped at i = $i" );
?>
</body>
</html>
```

## Example foreach

```
<html>
```

```
  <body>
```

```
    <?php
      $array = array( 1, 2, 3, 4, 5);

      foreach( $array as $value ) {
        echo "Value is $value <br />";
      }
    ?>
```

## OUTPUT

Value is 1

Value is 2

Value is 3

Value is 4

Value is 5

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

## The **break** statement

The PHP **break** keyword is used to terminate the execution of a loop prematurely.

## The **continue** statement

The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.

# PHP – Arrays

An array is a data structure that stores one or more similar type of values in a single name.

**Numeric array** – An array with a numeric index. Values are stored and accessed in linear fashion.

**Associative array** – An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.

**Multidimensional array** – An array containing one or more arrays and values are accessed using multiple indices

Example for practice. Two methods of using array

```
<?php
/* First method to create array.*/
$numbers = array( 1, 2, 3, 4, 5);

foreach( $numbers as $value ) {
    echo "Value is $value <br />";
}

?>
```

```
<?php

$numbers[0] = "one";
$numbers[1] = "two";
$numbers[2] = "three";
$numbers[3] = "four";
$numbers[4] = "five";

foreach( $numbers as $value ) {
    echo "Value is $value <br />";
}

?>
```

# Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index.

Associative array will have their index as string so that you can establish a strong association between key and values.

```
$salaries = array("James" => 1000, "Shweta" => 1500, "Sunil" => 2000);
```

```
echo "Salary of james is ". $salaries['James'] .;
```

# String Concatenation Operator

To concatenate two string variables together, use the dot (.) operator

```
<?php  
$string1="Hello World";  
$string2="1234";  
  
echo $string1 . " " . $string2;  
?>
```

Hello World 1234

Example for practice, addition of two numbers using form

```
<body>
<form>
Enter First Number:
<input type="number" name="number1" /><br><br>
Enter Second Number:
<input type="number" name="number2" /><br><br>
<input type="submit" name="submit" value="Add">
</form>
</body>
<?php
$number1=$_GET['number1'];
$number2=$_GET['number2'];
$result=$number1+$number2;
echo "Sum of $number1 and $number2 is ".$result;
?>
```

A screenshot of a web browser window. The title bar says "Js Validation" and the address bar says "localhost/localhost/hello.php". The page contains the following form:

Enter First Number:

Enter Second Number:

A screenshot of a web browser window. The title bar says "Js Validation" and the address bar says "localhost/localhost/hello.php?number1=22&number2=". The page displays the results of the addition:

Enter First Number:

Enter Second Number:

Sum of 22 and 10 is=32

# **PHP - GET & POST Methods**

There are two ways the browser client can send information to the web server.

The GET Method

The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding.

In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

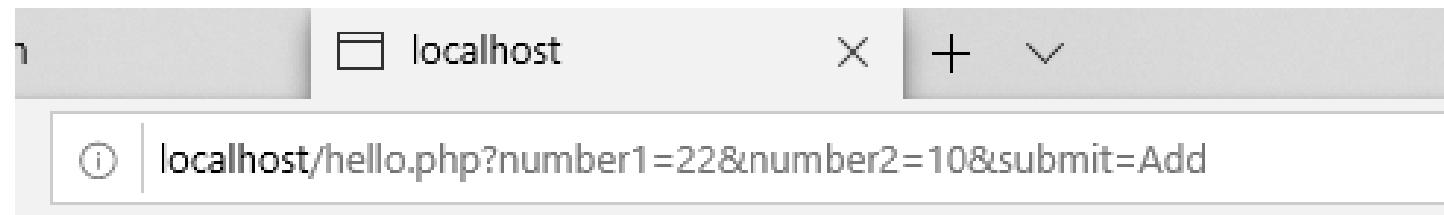
Spaces are removed and replaced with the + character and any other nonalphanumeric characters are replaced with a hexadecimal values.

After the information is encoded it is sent to the server.

# The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?character**.

From our previous example



- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.

Task:-Write a php program to read and display your name age using GET\_METHOD

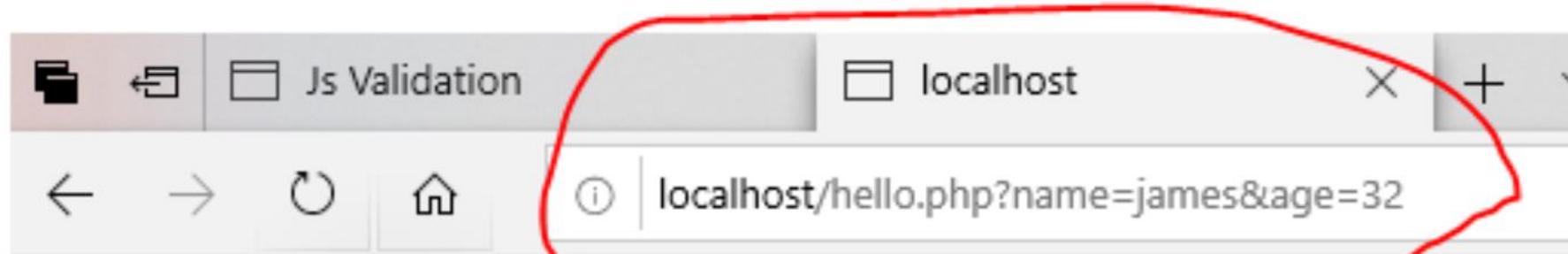
```
<?php
if( $_GET["name"] || $_GET["age"] ) {
    echo "Welcome ". $_GET['name']. "<br />";
    echo "You are ". $_GET['age']. " years old.";

    exit();
}
?>
<html>
    <body>

        <form action = "<?php $_PHP_SELF ?>" method = "GET">
            Name: <input type = "text" name = "name" />
            Age: <input type = "text" name = "age" />
            <input type = "submit" />
        </form>

    </body>
</html>
```

Name:  Age:



Welcome james  
You are 32 years old.

# **The POST Method**

The POST method transfers information via HTTP headers.

The POST method does not have any restriction on data size to be sent.

The POST method can be used to send ASCII as well as binary data.

The data sent by POST method goes through HTTP header so security depends on HTTP protocol.

The PHP provides **`$_POST`** associative array to access all the sent information using POST method.

The PHP **`$_REQUEST`** variable contains the contents of both **`$_GET`, `$_POST`**

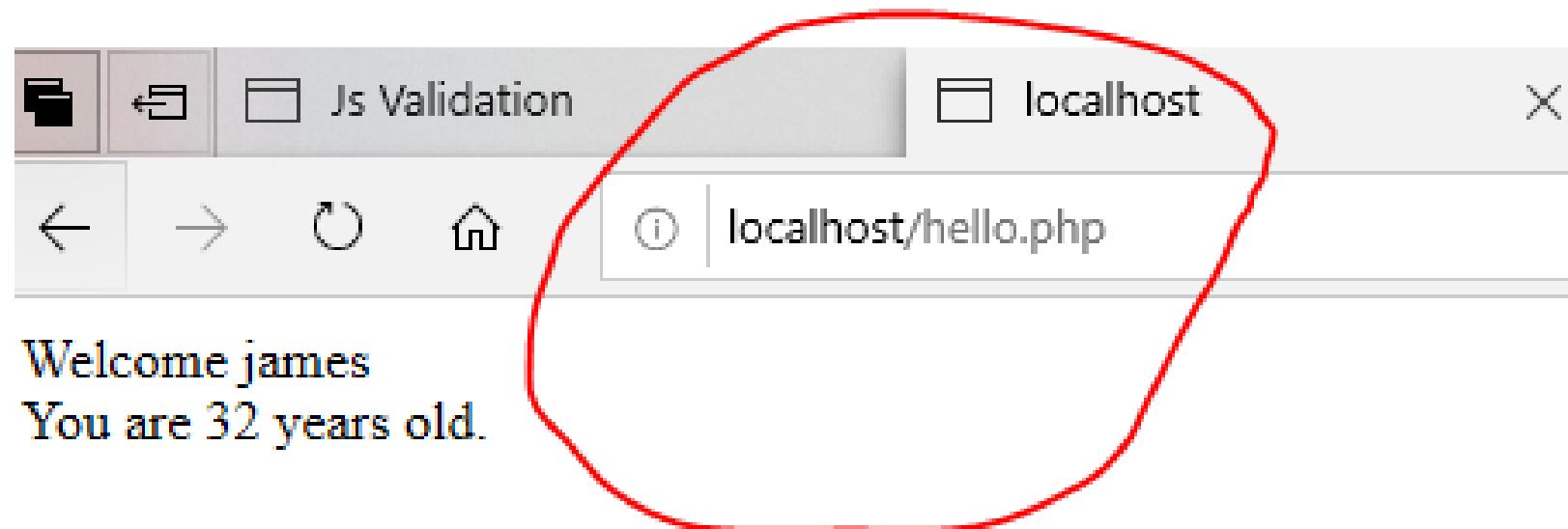
The PHP **`$_REQUEST`** variable can be used to get the result from form data sent with both the GET and POST methods.

```
<?php
if( $_REQUEST["name"] || $_REQUEST["age"] ) {
    echo "Welcome ". $_REQUEST['name']. "<br />";
    echo "You are ". $_REQUEST['age']. " years old.";
    exit();
}
?>
<html>
<body>

<form action = "<?php $_PHP_SELF ?>" method = "POST">
    Name: <input type = "text" name = "name" />
    Age: <input type = "text" name = "age" />
    <input type = "submit" />
</form>

</body>
</html>
```

Name:  Age:



# PHP - File Inclusion

You can include the content of a PHP file into another PHP file before the server executes it.

The `include()` Function  
The `require()` Function

This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages. This will help developers to make it easy to change the layout of complete website with minimal effort.

# The include() Function

The include() function takes all the text in a specified file and copies it into the file that uses the include function.

If there is any problem in loading a file then the **include()** function generates a warning but the script will continue execution.

```
<html>
  <body>

    <?php include("menu.php"); ?>
    <p>This is an example to show how to include PHP file!</p>

  </body>
</html>
```

# The require() Function

The require() function takes all the text in a specified file and copies it into the file that uses the include function.

If there is any problem in loading a file then the **require()** function generates a fatal error and halt the execution of the script.

# **PHP – Functions**

PHP functions are similar to other programming languages. A function is a piece of code which takes one or more input in the form of parameter and does some processing and returns a value.

## **Creating PHP Function**

It's very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it.

```
<html>
```

```
  <head>
    <title>Writing PHP Function</title>
  </head>
```

```
  <body>
```

```
    <?php
```

**/\* Defining a PHP Function \*/**

```
      function writeMessage() {
        echo "You are really a nice person, Have a nice time!";
      }
```

**/\* Calling a PHP Function \*/**

```
      writeMessage();
    ?>
```

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

# PHP Functions with Parameters

```
<html>

<head>
    <title>Writing PHP Function with Parameters</title>
</head>

<body>

<?php
    function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        echo "Sum of the two numbers is : $sum";
    }

    addFunction(10, 20);
?>

</body>
</html>
```

# Passing Arguments by Reference

```
<html>

<head>
    <title>Passing Argument by Reference</title>
</head>

<body>
    $orignum = 10;
    addFive( $orignum );

    echo "Original Value is $orignum<br />";

    addSix( $orignum );
    echo "Original Value is $orignum<br />";
    ?>

</body>
</html>

<?php
    function addFive($num) {
        $num += 5;
    }

    function addSix(&$num) {
        $num += 6;
    }
</?php>
```

# PHP Functions returning value

```
<html>

<head>
    <title>Writing PHP Function which returns value</title>
</head>

<body>

<?php
    function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        return $sum;
    }
    $return_value = addFunction(10, 20);

    echo "Returned value from the function : $return_value";
?>

</body>
</html>
```

# PHP

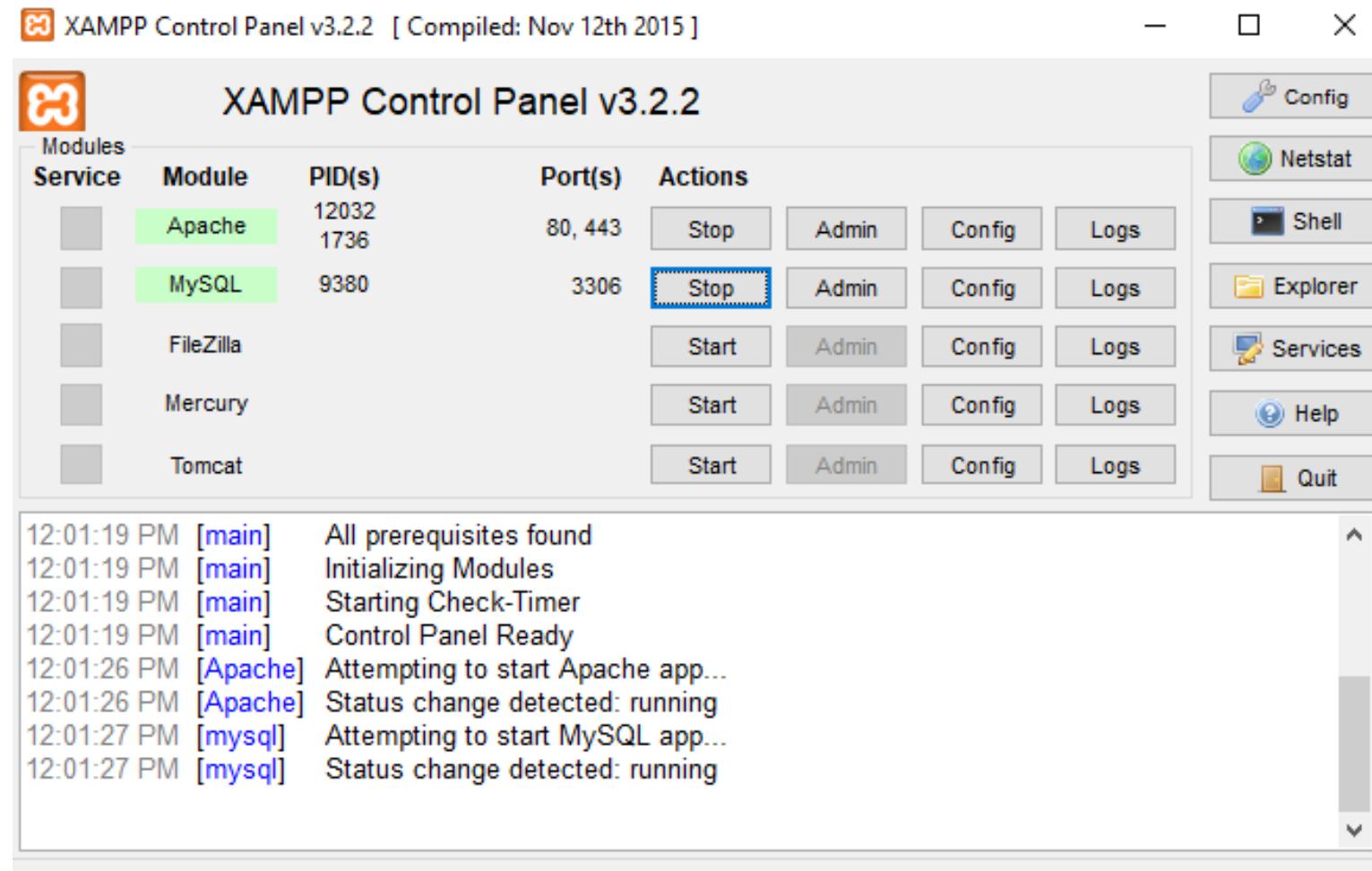
## Manage Database with PhpMyAdmin

MySQL is an open-source relational database management system (RDBMS). MySQL is a popular choice of database for use in web applications.

phpMyAdmin is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a web browser.

In the following examples, we will see how easy we can handle MySQL with PhpMyAdmin.

## Step 1: run XAMPP application with MySQL



## Step 2: Create a Database

Go to <http://localhost/phpmyadmin/>

Click the "New" link on the upper left corner (under recent tables)

Fill out the "Database Name" field with "my\_first\_database". // I used Demo

Click the "Create" button

The screenshot shows the phpMyAdmin interface for MySQL version 5.7.23. The browser address bar shows the URL `localhost/phpmyadmin/server_databases.php?server=1`. The main menu bar includes links for Js Validation, localhost / 127.0.0.1 | pt, +, and a dropdown arrow. Below the menu is a toolbar with icons for Home, Recent, Favorites, and a New database button. The left sidebar displays a tree view of databases: information\_schema, mysql, performance\_schema, phpmyadmin, and test. The main content area is titled "Databases" and features a "Create database" form with a "Demo" input field and a "Collation" dropdown set to "Collation". A "Create" button is located to the right of the form. Below the form is a table showing existing databases:

Database	Collation	Action
information_schema	utf8_general_ci	Check privileges
mysql	latin1_swedish_ci	Check privileges

### Step 3: Create a Table

Click "my\_first\_database" on the left side of the screen. // in my case its Demo

On the "Create Table" section, fill out the Name with "user\_info" and Number of Columns with "4"

Click "Go" button

The screenshot shows the phpMyAdmin interface for MySQL. On the left, the database tree shows several databases: demo, information\_schema, mysql, performance\_schema, phpmyadmin, and test. The demo database is selected. The main area is titled 'Table: user\_info'. The 'Structure' tab is active. The 'Table name:' field contains 'user\_info'. The 'Add' field shows '1 column(s)'. Below this, there are four rows for defining columns. Each row has fields for 'Name', 'Type' (set to 'INT'), 'Length/Values', 'Default' (set to 'None'), and 'Collation'. The first three rows have a 'Pick from Central Columns' link below them. The 'Go' button is visible at the top right of the form.

## Step 4: complete the table schema

Fill out the fields with id, name, age and gender etc.

Click the "Save" button

The screenshot shows the phpMyAdmin interface with the following details:

- Left Panel:** Shows the database tree with databases like demo, information\_schema, mysql, performance\_schema, phpmyadmin, and test.
- Header:** Shows the URL as `localhost/phpmyadmin/db_structure.php?server=1&db=de`.
- Toolbar:** Includes links for Home, Recent, Favorites, and various database operations.
- Table Structure View:** The table name is set to `user_info`. The columns are defined as follows:

Name	Type
id	INT
name	VARCHAR
age	SMALLINT
gender	CHAR
- Bottom:** A note says "Table comments: " followed by a text input field.

Js Validation    localhost / 127.0.0.1 / d × +

localhost/phpmyadmin/db\_structure.php?server=1&db=demo&table=user\_info

# phpMyAdmin

Server: 127.0.0.1 » Database: demo » Table: user\_info

Browse Structure SQL Search Insert Export Import Privileges Opera

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id</b>	int(11)			No	None			Change  Drop  More
2	<b>name</b>	varchar(20)	latin1_swedish_ci		No	None			Change  Drop  More
3	<b>age</b>	smallint(6)			No	None			Change  Drop  More
4	<b>gender</b>	char(1)	latin1_swedish_ci		No	None			Change  Drop  More

Check all    With selected: Browse Change Drop Primary Unique Index Fulltext  
 Remove from central columns

New  
demo  
New  
user\_info  
information\_schema  
mysql  
performance\_schema  
phpmyadmin  
test

## step 5: Insert Data

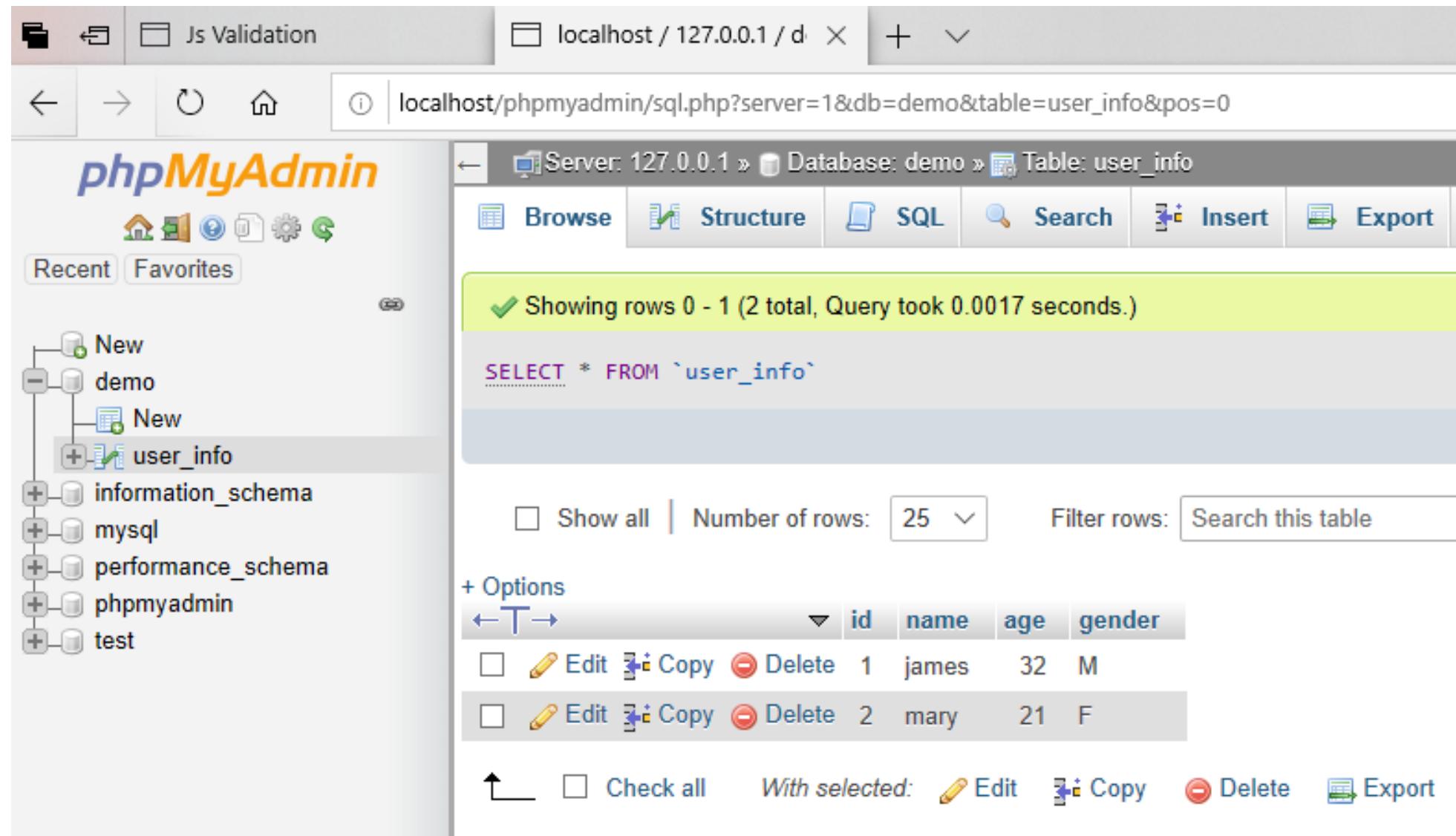
Click the “user\_info” table. From the left pane, then click on “insert” tab, fill all the values and clock “Go”

The screenshot shows the phpMyAdmin interface for a database named 'demo'. The left sidebar lists databases like 'information\_schema', 'mysql', and 'phpmyadmin'. The 'user\_info' table is selected in the main area. The 'Insert' tab is active, showing two sets of input fields for inserting new rows. The first row has values: id=1, name=james, age=32, gender=M. The second row has values: id=2, name=mary, age=21, gender=F. Both rows have a 'Go' button at the bottom right.

Column	Type	Value
id	int(11)	1
name	varchar(20)	james
age	smallint(6)	32
gender	char(1)	M

Column	Type	Value
id	int(11)	2
name	varchar(20)	mary
age	smallint(6)	21
gender	char(1)	F



# PHP

**Manage more options of Database  
through PhpMyAdmin and php script**

# Change Admin password

Step 1: Run XAMPP Application

Step 2: open PhpMyAdmin page on a new tab

Step 3: click on user accounts

Step 4: click on Edit privileges

The screenshot shows the PhpMyAdmin interface for managing user accounts. At the top, there's a navigation bar with tabs for Databases, SQL, Status, User accounts (which is selected), Export, Import, and Settings. Below the navigation bar, there are two buttons: 'User accounts overview' and 'User groups'. The main content area is titled 'User accounts overview'. A warning message is displayed: '⚠ A user account allowing any user from localhost to connect is present. This will prevent other users from connecting if host. ⓘ'. The table below lists user accounts:

User name	Host name	Password	Global privileges ⓘ	User group	Grant	Action
Any	%	No ⓘ	USAGE		No	<a href="#">Edit privileges</a> <a href="#">Export</a>
Any	localhost	No	USAGE		No	<a href="#">Edit privileges</a> <a href="#">Export</a>
pma	localhost	No	USAGE		No	<a href="#">Edit privileges</a> <a href="#">Export</a>
root	127.0.0.1	Yes	ALL PRIVILEGES		Yes	<a href="#">Edit privileges</a> <a href="#">Export</a>
root	::1	No	ALL PRIVILEGES		Yes	<a href="#">Edit privileges</a> <a href="#">Export</a>
root	localhost	Yes	ALL PRIVILEGES		Yes	<a href="#">Edit privileges</a> <a href="#">Export</a>

Step 5: click on change password, then click on “Go”

The screenshot shows the MySQL Workbench interface for a server at 127.0.0.1. The 'User accounts' tab is selected. Below it, the 'Change password' tab is highlighted with a red arrow pointing to it. The main area displays the message "Edit privileges: User account 'root'@'localhost'". A warning message in a yellow box states: "Note: You are attempting to edit privileges of the user with which you are currently logged in." The 'Change password' form is visible, containing fields for entering a new password and re-typing it, along with a strength meter indicating "Very weak".

Server: 127.0.0.1

Databases SQL Status User accounts Export

Global Database Change password Login Information

Edit privileges: User account 'root'@'localhost'

Note: You are attempting to edit privileges of the user with which you are currently logged in.

Change password

No Password

Enter:  Strength: █ Very weak

Password:

Re-type:

Password Hashing: Native MySQL authentication

Generate password

This error message will appear if you try to access phpmyadmin after changing password, so we need to update configuration file

A screenshot of a web browser window titled "Access denied!". The address bar shows "localhost/phpmyadmin/". The main content area displays the "Welcome to phpMyAdmin" page, but it is overlaid by an "Error" message. The error message is contained within a red-bordered box with three stacked sections. The first section says "MySQL said: ⓘ Cannot connect: invalid settings.". The second section says "ⓘ mysqli\_real\_connect(): (HY000/1045): Access denied for user 'root'@'localhost' (using password: NO)". The third section says "ⓘ phpMyAdmin tried to connect to the MySQL server, and the server rejected the connection. You should check the host, username and password in your configuration and make sure that they correspond to the information given by the administrator of the MySQL server." At the bottom left of the error box is a "Retry to connect" button.

Welcome to phpMyAdmin

Error

MySQL said: ⓘ  
Cannot connect: invalid settings.

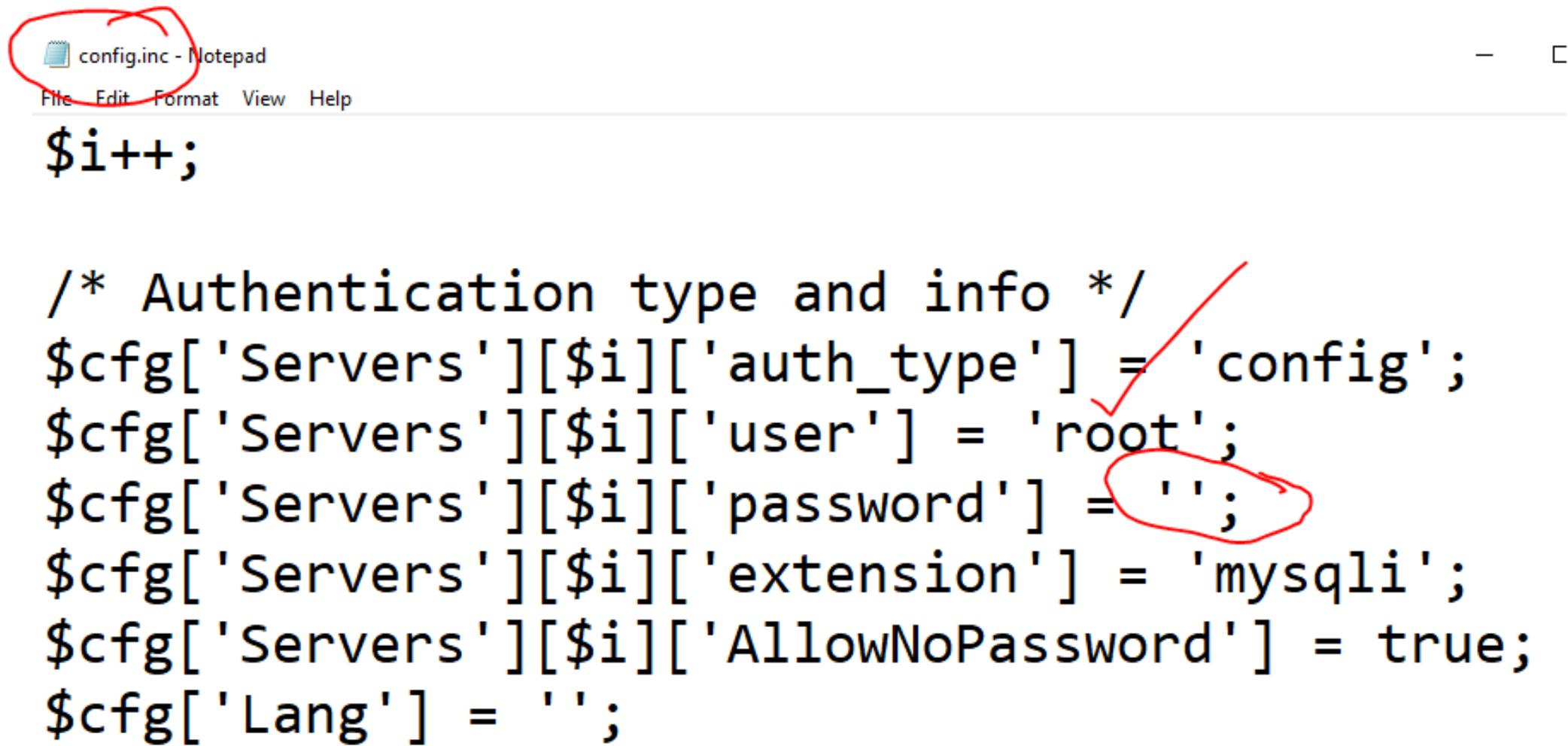
ⓘ mysqli\_real\_connect(): (HY000/1045): Access denied for user 'root'@'localhost' (using password: NO)

ⓘ phpMyAdmin tried to connect to the MySQL server, and the server rejected the connection. You should check the host, username and password in your configuration and make sure that they correspond to the information given by the administrator of the MySQL server.

Retry to connect

Step 6: select config.inc.php file from C->xampp->PhpMyAdmin

Step 7: update password field

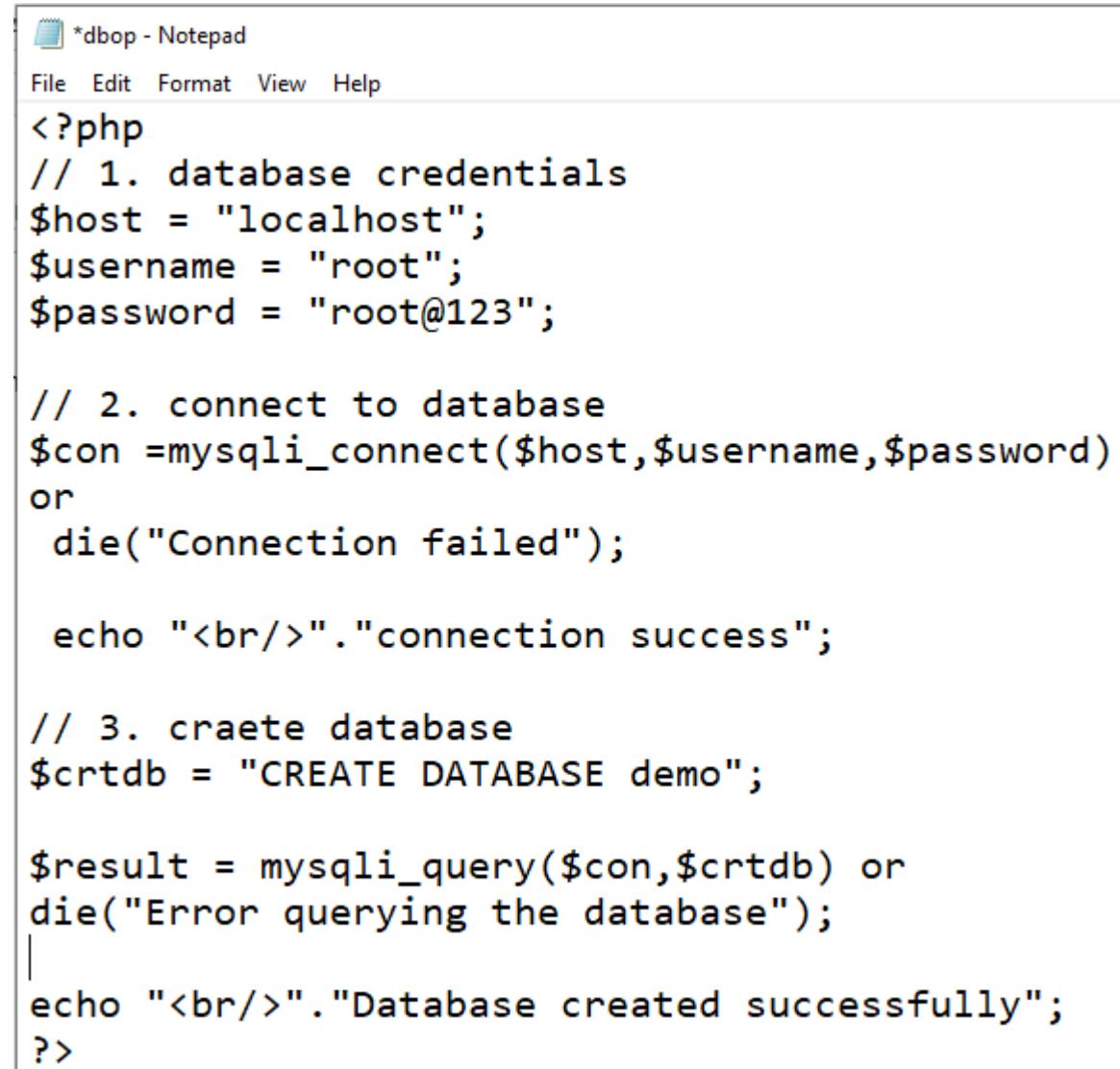


```
$i++;

/* Authentication type and info */
$cfg['Servers'][$i]['auth_type'] = 'config';
$cfg['Servers'][$i]['user'] = 'root';
$cfg['Servers'][$i]['password'] = '';
$cfg['Servers'][$i]['extension'] = 'mysqli';
$cfg['Servers'][$i]['AllowNoPassword'] = true;
$cfg['Lang'] = '';
```

# Create database

Open another file dbop.php in C->xampp->htdocs and add the content



The image shows a screenshot of a Windows Notepad window titled '\*dbop - Notepad'. The window contains PHP code for creating a database. The code includes comments for database credentials, connecting to the database, creating a database named 'demo', and echoing success messages.

```
<?php
// 1. database credentials
$host = "localhost";
$username = "root";
$password = "root@123";

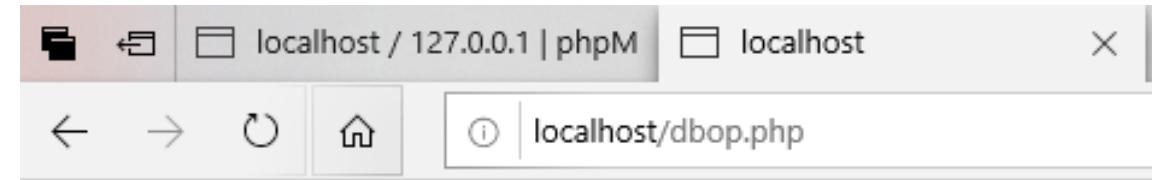
// 2. connect to database
$con =mysqli_connect($host,$username,$password)
or
die("Connection failed");

echo "<br/>". "connection success";

// 3. craete database
$crtdb = "CREATE DATABASE demo";

$result = mysqli_query($con,$crtdb) or
die("Error querying the database");
|
echo "<br/>". "Database created successfully";
?>
```

Run dbop.php in a new tab

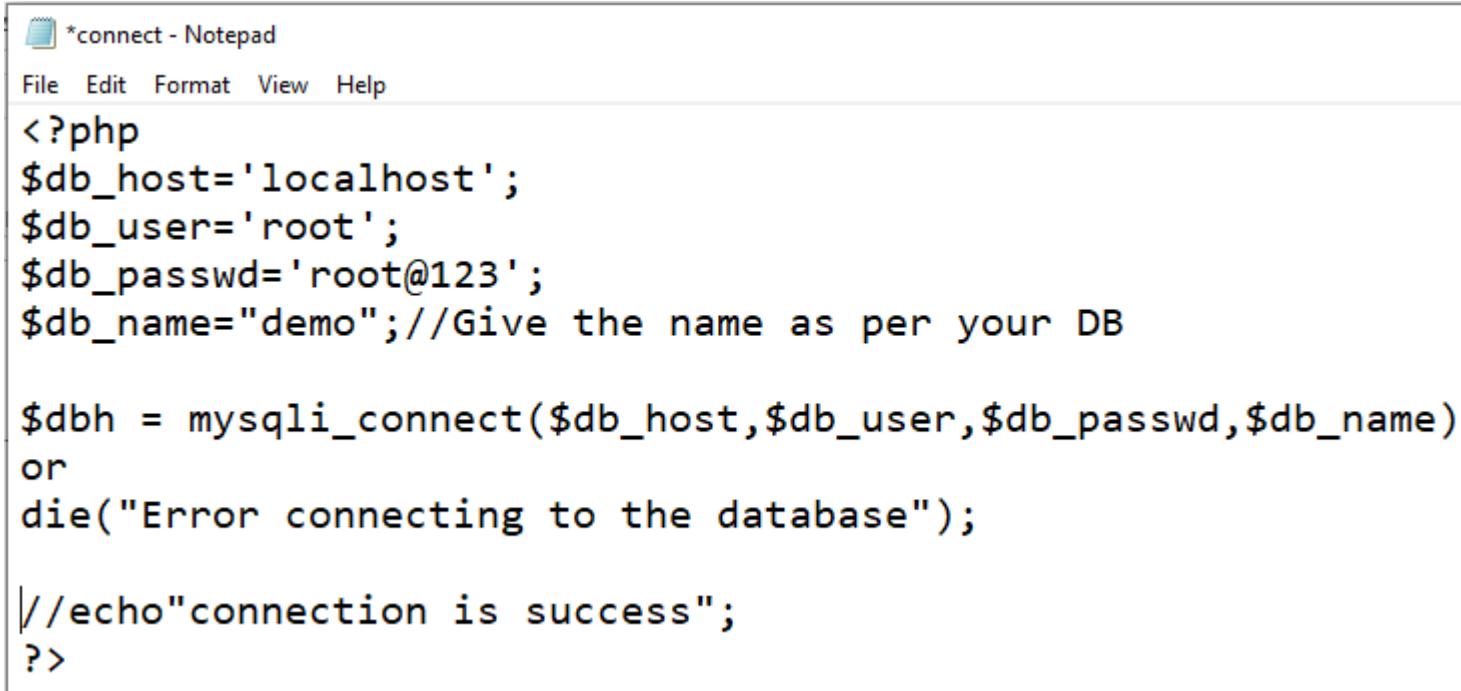


Go and refresh phpmyadmin page you can see that the new database demo is created

Database	Collation	Action
demo	latin1_swedish_ci	<a href="#">Check privileges</a>
information schema	utf8_general_ci	<a href="#">Check privileges</a>

# Create table

Maintain two files, one for storing connection (connect.php) details and another (dbop.php)for other activities



The screenshot shows a Notepad window titled '\*connect - Notepad'. The menu bar includes File, Edit, Format, View, and Help. The code in the editor is as follows:

```
<?php
$db_host='localhost';
$db_user='root';
$db_passwd='root@123';
$db_name="demo";//Give the name as per your DB

$dh = mysqli_connect($db_host,$db_user,$db_passwd,$db_name)
or
die("Error connecting to the database");

//echo"connection is success";
?>
```



dbop - Notepad

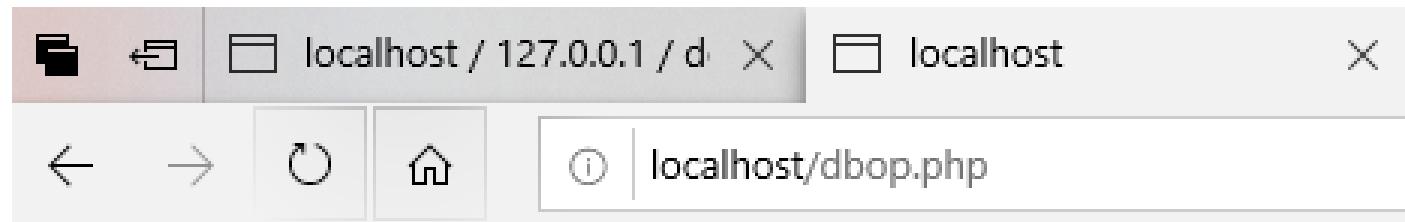
File Edit Format View Help

```
<?php
include("connect.php");

$crttb = "CREATE TABLE user_info(
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(20) NOT NULL,
age INT(6) NOT NULL,
gender CHAR(1) NOT NULL)";

$result = mysqli_query($dbh,$crttb) or
die("Error querying the database");

echo "<br/>". "table created successfully";
?>
```



Go and refresh phpmyadmin page you can see that the new table user-info is created under demo database

A screenshot of the phpMyAdmin interface. The top navigation bar shows the server as "localhost / 127.0.0.1 / d" and the database as "localhost". The main title is "phpMyAdmin". On the left, the database tree shows "demo" expanded, with "user\_info" selected. Other databases listed are "information\_schema", "mysql", and "performance\_schema". The right panel has tabs for "Browse", "Structure", "SQL", and "Search". The "Structure" tab is active. A message box at the top right says "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 sec)". Below it is a SQL query: "SELECT \* FROM `user\_info`". At the bottom, there is a table header with columns "id", "name", "age", and "gender".

# Insert data

dbop - Notepad

File Edit Format View Help

```
<?php  
include("connect.php");  
  
$ins = "INSERT INTO user_info (name, age, gender)  
VALUES ('james', '32', 'M');  
  
$result = mysqli_query($dbh,$ins) or  
die("Error querying the database");  
  
echo "<br/>". "data inserted| successfully";  
?>
```



No need of value for id as we made it as auto increment

Go and refresh  
phpmyadmin page

The screenshot shows the phpMyAdmin interface for a database named 'demo' with a table named 'user\_info'. The browser tab at the top displays 'Server: 127.0.0.1 » Database: demo » Table: user\_info'. Below the tabs, there are several buttons: 'Browse' (selected), 'Structure', 'SQL', 'Search', 'Insert', 'Export', and 'In'. A large green success message box at the top states 'Showing rows 0 - 0 (1 total, Query took 0.0022 seconds.)'. Below it, the SQL query 'SELECT \* FROM `user\_info`' is shown. At the bottom, there are options to 'Show all' (unchecked), set the 'Number of rows' to 25 (selected), and a 'Filter rows' input field with placeholder 'Search this table'. The table itself has columns: id, name, age, and gender. One row is displayed with values: id=1, name=james, age=32, gender=M. There are buttons for 'Edit', 'Copy', and 'Delete' next to the row.

Task:- Try the same insert command with different values

Present status of user\_info table

The screenshot shows the MySQL Workbench interface with the following details:

- Server: 127.0.0.1
- Database: demo
- Table: user\_info
- Toolbar buttons: Browse (selected), Structure, SQL, Search, Insert.
- Message bar: ✓ Showing rows 0 - 2 (3 total, Query took 0.0017 seconds.)
- SQL pane: SELECT \* FROM `user\_info`
- Filtering options: Show all, Number of rows: 25, Filter rows: Search this
- Table view:
  - Header: + Options, id, name, age, gender
  - Data rows:

	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	id	name	age	gender
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	1	james	32	M
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	2	joseph	35	M
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	3	Mary	28	F

# Retrieve data and print

```
*dbop - Notepad
File Edit Format View Help
<?php
include("connect.php");

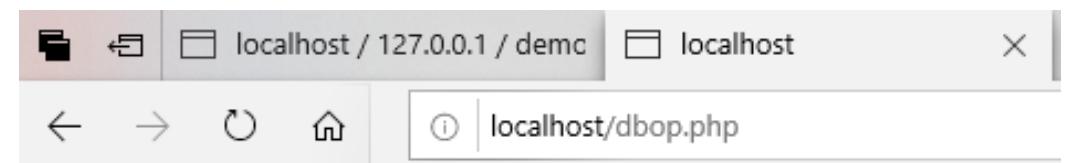
$sel = "SELECT name,age,gender FROM user_info";

$result = mysqli_query($dbh,$sel) or
die("Error querying the database");

while($row = mysqli_fetch_assoc($result))
{
echo "NAME:". $row[ 'name' ]."AGE:". $row[ 'age' ]."Gender:". $row[ 'gender' ]."<br/>";
}

?>
```

Run and verify the output



NAME:jamesAGE:32Gender:M  
NAME:josephAGE:35Gender:M  
NAME:MaryAGE:28Gender:F

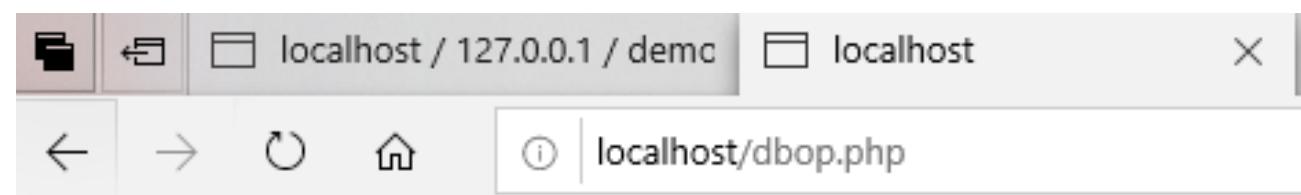
```
<?php
include("connect.php");

$sel = "SELECT name,age |FROM user_info WHERE gender='F'";

$result = mysqli_query($dbh,$sel) or
die("Error querying the database");

while($row = mysqli_fetch_assoc($result))
{
echo "NAME:". $row['name'] ."AGE:". $row['age'] . "<br/>";
}

?>
```



NAME:MaryAGE:28

# Update data

dbop - Notepad

File Edit Format View Help

```
<?php
include("connect.php");

$sel = " UPDATE user_info SET age='30' WHERE id='3'"';

$result = mysqli_query($dbh,$sel) or
die("Error querying the database");

?>
```

Before update

Server: 127.0.0.1 » Database: demo » Table: user\_info

Browse Structure SQL Search Insert

Showing rows 0 - 2 (3 total, Query took 0.0017 seconds.)

```
SELECT * FROM `user_info`
```

Show all Number of rows: 25 Filter rows: Search this

+ Options

		id	name	age	gender
<input type="checkbox"/>	Edit  Copy  Delete	1	james	32	M
<input type="checkbox"/>	Edit  Copy  Delete	2	joseph	35	M
<input type="checkbox"/>	Edit  Copy  Delete	3	Mary	28	F

After update

Server: 127.0.0.1 » Database: demo » Table: user\_info

Browse Structure SQL Search Insert

Showing rows 0 - 2 (3 total, Query took 0.0142 seconds.)

```
SELECT * FROM `user_info`
```

Show all Number of rows: 25 Filter rows: Search this

+ Options

		id	name	age	gender
<input type="checkbox"/>	Edit  Copy  Delete	1	james	32	M
<input type="checkbox"/>	Edit  Copy  Delete	2	joseph	35	M
<input type="checkbox"/>	Edit  Copy  Delete	3	Mary	30	F

# PHP

**Manage more options of Database  
through PhpMyAdmin and php script**

# Change Admin password

Step 1: Run XAMPP Application

Step 2: open PhpMyAdmin page on a new tab

Step 3: click on user accounts

Step 4: click on Edit privileges

The screenshot shows the PhpMyAdmin interface for managing user accounts. At the top, there's a navigation bar with tabs for Databases, SQL, Status, User accounts (which is selected), Export, Import, and Settings. Below the navigation bar, there are two buttons: 'User accounts overview' and 'User groups'. The main content area is titled 'User accounts overview'. A warning message is displayed: '⚠ A user account allowing any user from localhost to connect is present. This will prevent other users from connecting if host. ⓘ'. The table below lists user accounts:

User name	Host name	Password	Global privileges ⓘ	User group	Grant	Action
Any	%	No ⓘ	USAGE		No	<a href="#">Edit privileges</a> <a href="#">Export</a>
Any	localhost	No	USAGE		No	<a href="#">Edit privileges</a> <a href="#">Export</a>
pma	localhost	No	USAGE		No	<a href="#">Edit privileges</a> <a href="#">Export</a>
root	127.0.0.1	Yes	ALL PRIVILEGES		Yes	<a href="#">Edit privileges</a> <a href="#">Export</a>
root	::1	No	ALL PRIVILEGES		Yes	<a href="#">Edit privileges</a> <a href="#">Export</a>
root	localhost	Yes	ALL PRIVILEGES		Yes	<a href="#">Edit privileges</a> <a href="#">Export</a>

Step 5: click on change password, then click on “Go”

Server: 127.0.0.1

Databases SQL Status User accounts Export

Global Database Change password Login Information

Edit privileges: User account 'root'@'localhost'

**Change password**

Note: You are attempting to edit privileges of the user with which you are currently logged in.

No Password  
 Password:

Enter:  Strength:  Very weak

Re-type:

Password Hashing: Native MySQL authentication

Generate password

This error message will appear if you try to access phpmyadmin after changing password, so we need to update configuration file

A screenshot of a web browser window titled "Access denied!". The address bar shows "localhost/phpmyadmin/". The main content area displays the "Welcome to phpMyAdmin" page, but it is overlaid by an "Error" message. The error message is contained within a red-bordered box with three stacked sections. The first section says "MySQL said: ⓘ Cannot connect: invalid settings.". The second section says "ⓘ mysqli\_real\_connect(): (HY000/1045): Access denied for user 'root'@'localhost' (using password: NO)". The third section says "ⓘ phpMyAdmin tried to connect to the MySQL server, and the server rejected the connection. You should check the host, username and password in your configuration and make sure that they correspond to the information given by the administrator of the MySQL server." At the bottom left of the error box is a "Retry to connect" button.

Welcome to phpMyAdmin

Error

MySQL said: ⓘ  
Cannot connect: invalid settings.

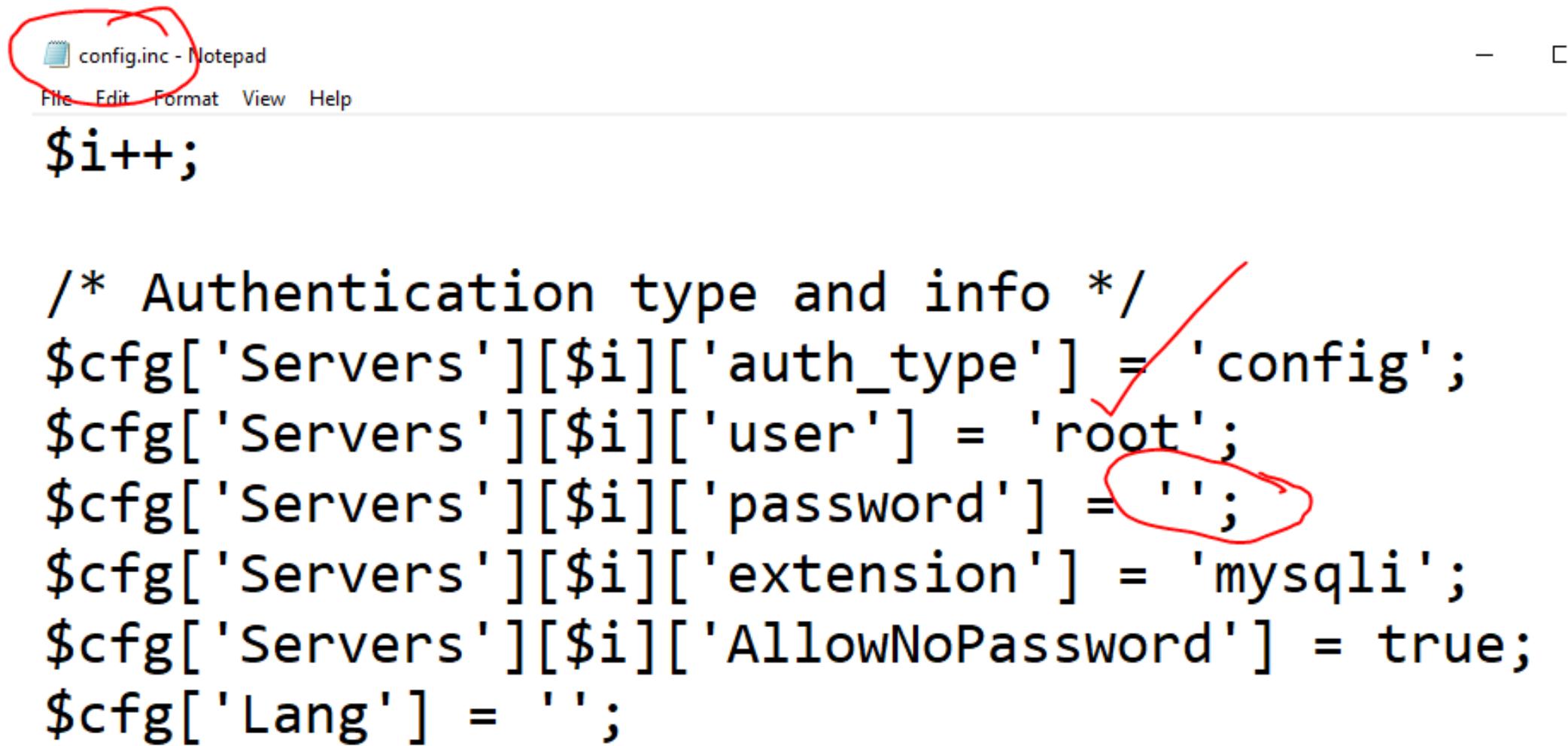
ⓘ mysqli\_real\_connect(): (HY000/1045): Access denied for user 'root'@'localhost' (using password: NO)

ⓘ phpMyAdmin tried to connect to the MySQL server, and the server rejected the connection. You should check the host, username and password in your configuration and make sure that they correspond to the information given by the administrator of the MySQL server.

Retry to connect

Step 6: select config.inc.php file from C->xampp->PhpMyAdmin

Step 7: update password field

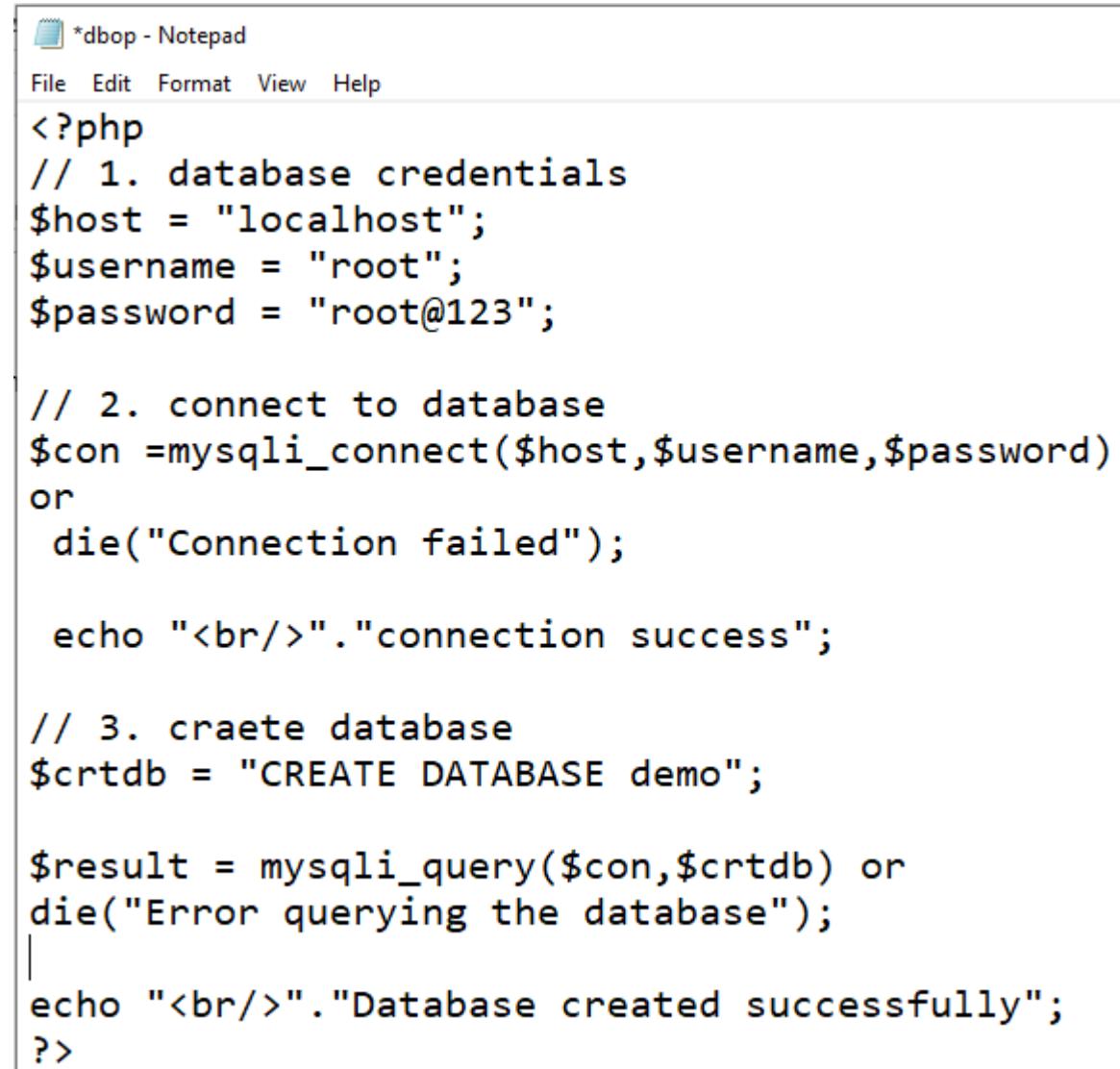


```
$i++;

/* Authentication type and info */
$cfg['Servers'][$i]['auth_type'] = 'config';
$cfg['Servers'][$i]['user'] = 'root';
$cfg['Servers'][$i]['password'] = '';
$cfg['Servers'][$i]['extension'] = 'mysqli';
$cfg['Servers'][$i]['AllowNoPassword'] = true;
$cfg['Lang'] = '';
```

# Create database

Open another file dbop.php in C->xampp->htdocs and add the content



The image shows a screenshot of a Windows Notepad window titled '\*dbop - Notepad'. The window contains PHP code for creating a database. The code includes comments for database credentials, connecting to the database, creating a database named 'demo', and echoing success messages.

```
<?php
// 1. database credentials
$host = "localhost";
$username = "root";
$password = "root@123";

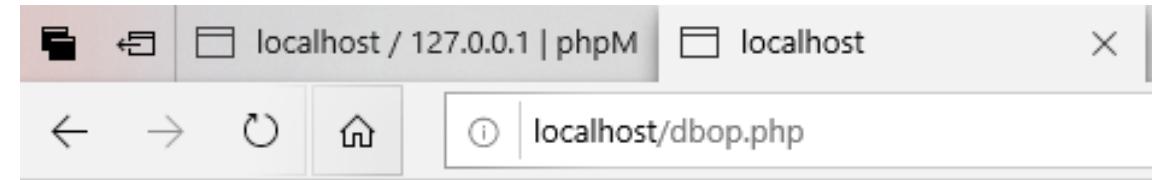
// 2. connect to database
$con =mysqli_connect($host,$username,$password)
or
die("Connection failed");

echo "<br/>". "connection success";

// 3. craete database
$crtdb = "CREATE DATABASE demo";

$result = mysqli_query($con,$crtdb) or
die("Error querying the database");
|
echo "<br/>". "Database created successfully";
?>
```

Run dbop.php in a new tab

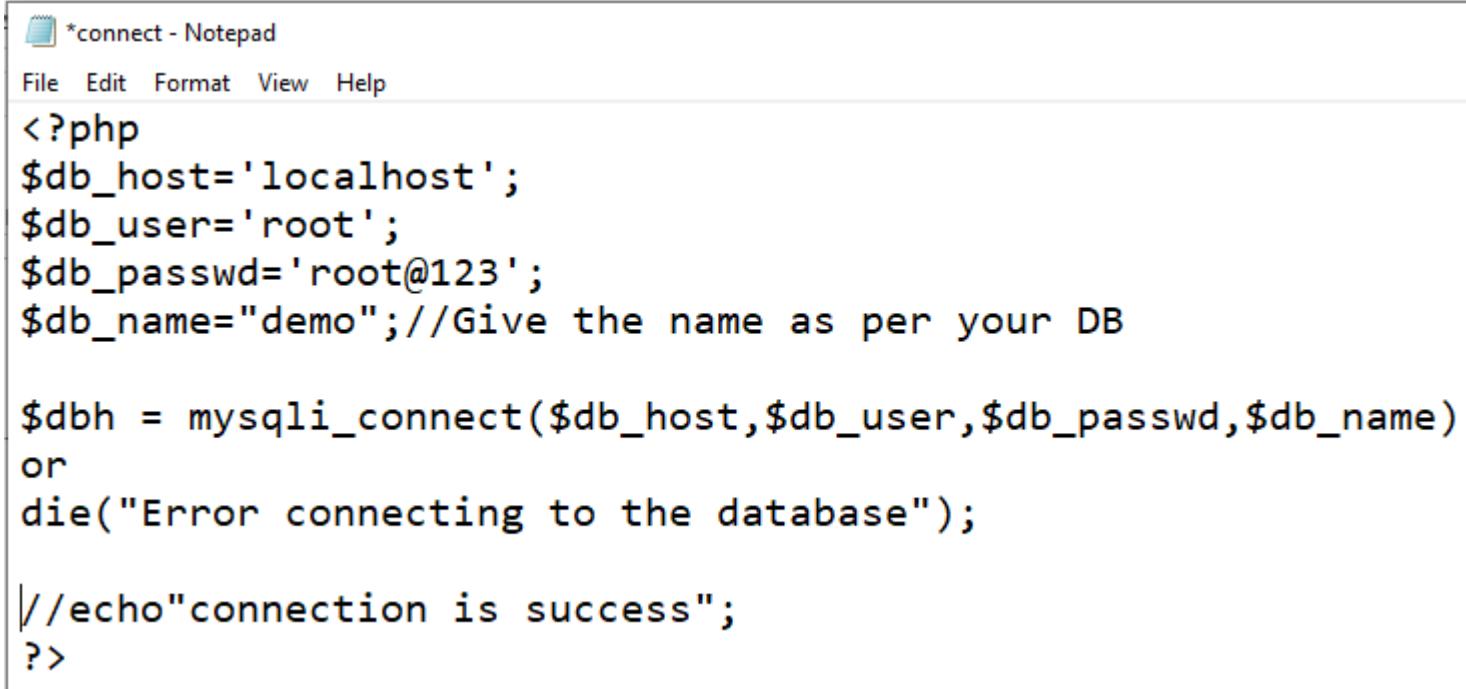


Go and refresh phpmyadmin page you can see that the new database demo is created

Database	Collation	Action
demo	latin1_swedish_ci	<a href="#">Check privileges</a>
information schema	utf8_general_ci	<a href="#">Check privileges</a>

# Create table

Maintain two files, one for storing connection (connect.php) details and another (dbop.php)for other activities



```
*connect - Notepad
File Edit Format View Help
<?php
$db_host='localhost';
$db_user='root';
$db_passwd='root@123';
$db_name="demo";//Give the name as per your DB

$dbh = mysqli_connect($db_host,$db_user,$db_passwd,$db_name)
or
die("Error connecting to the database");

//echo"connection is success";
?>
```



dbop - Notepad

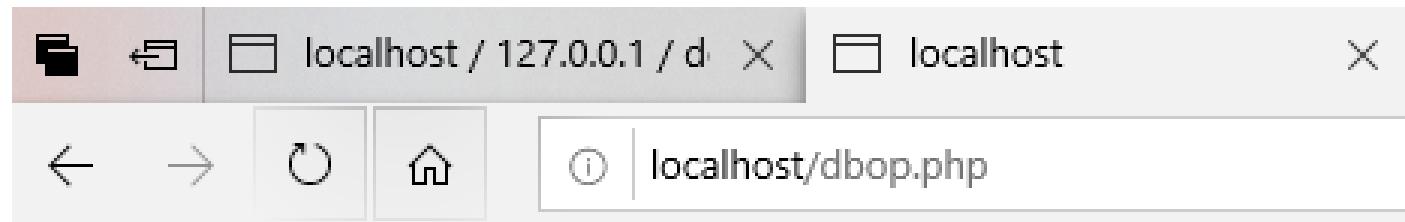
File Edit Format View Help

```
<?php
include("connect.php");

$crttb = "CREATE TABLE user_info(
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(20) NOT NULL,
age INT(6) NOT NULL,
gender CHAR(1) NOT NULL)";

$result = mysqli_query($dbh,$crttb) or
die("Error querying the database");

echo "<br/>". "table created successfully";
?>
```



Go and refresh phpmyadmin page you can see that the new table user-info is created under demo database

A screenshot of the phpMyAdmin interface. The top navigation bar shows the server as "localhost / 127.0.0.1 / d" and the database as "localhost". The main title is "phpMyAdmin". On the left, the database tree shows "demo" expanded, with "user\_info" selected. Other databases listed are "information\_schema", "mysql", and "performance\_schema". The right panel has tabs for "Browse", "Structure", "SQL", and "Search". The "Structure" tab is active. A message box at the top right says "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 sec)". Below it is an SQL query: "SELECT \* FROM `user\_info`". At the bottom, there is a table header with columns "id", "name", "age", and "gender".

# Insert data

dbop - Notepad

File Edit Format View Help

```
<?php  
include("connect.php");  
  
$ins = "INSERT INTO user_info (name, age, gender)  
VALUES ('james', '32', 'M');  
  
$result = mysqli_query($dbh,$ins) or  
die("Error querying the database");  
  
echo "<br/>". "data inserted| successfully";  
?>
```



No need of value for id as we made it as auto increment

Go and refresh  
phpmyadmin page

The screenshot shows the phpMyAdmin interface for a database named 'demo' with a table named 'user\_info'. The browser tab at the top displays 'Server: 127.0.0.1 » Database: demo » Table: user\_info'. Below the tabs, there are several buttons: 'Browse' (selected), 'Structure', 'SQL', 'Search', 'Insert', 'Export', and 'In'. A large green success message box at the top states 'Showing rows 0 - 0 (1 total, Query took 0.0022 seconds.)'. Below this, the SQL query 'SELECT \* FROM `user\_info`' is shown. At the bottom, there are options to 'Show all' (unchecked), set the 'Number of rows' to 25 (selected), and a 'Filter rows' input field with placeholder 'Search this table'. The table itself has columns: id, name, age, and gender. One row is displayed with values: id=1, name=james, age=32, gender=M. There are buttons for 'Edit', 'Copy', and 'Delete' next to the row.

Task:- Try the same insert command with different values

Present status of user\_info table

The screenshot shows the MySQL Workbench interface with the following details:

- Server: 127.0.0.1
- Database: demo
- Table: user\_info
- Toolbar buttons: Browse (selected), Structure, SQL, Search, Insert.
- Message bar: ✓ Showing rows 0 - 2 (3 total, Query took 0.0017 seconds.)
- SQL pane: SELECT \* FROM `user\_info`
- Filtering options: Show all, Number of rows: 25, Filter rows: Search this
- Table view:
  - Header: + Options, id, name, age, gender
  - Data rows:

	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	id	name	age	gender
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	1	james	32	M
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	2	joseph	35	M
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	3	Mary	28	F

# Retrieve data and print

```
*dbop - Notepad
File Edit Format View Help
<?php
include("connect.php");

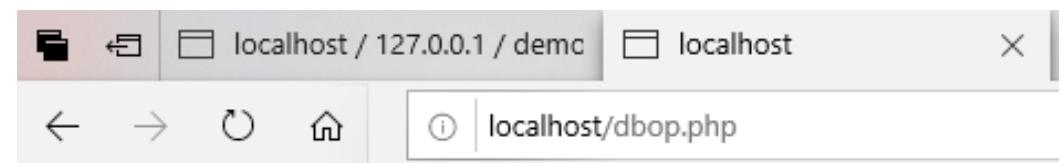
$sel = "SELECT name,age,gender FROM user_info";

$result = mysqli_query($dbh,$sel) or
die("Error querying the database");

while($row = mysqli_fetch_assoc($result))
{
echo "NAME:". $row[ 'name' ]."AGE:". $row[ 'age' ]."Gender:". $row[ 'gender' ]."<br/>";
}

?>
```

Run and verify the output



NAME:jamesAGE:32Gender:M  
NAME:josephAGE:35Gender:M  
NAME:MaryAGE:28Gender:F

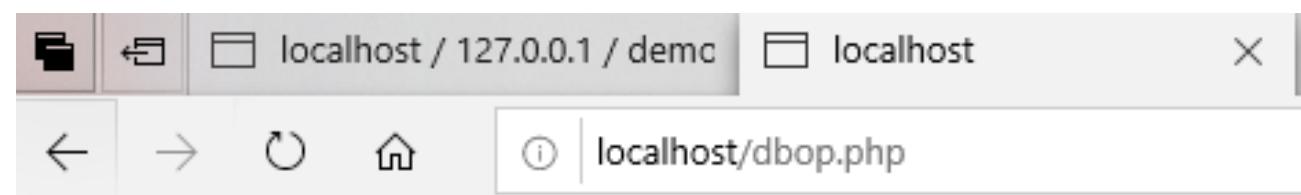
```
<?php
include("connect.php");

$sel = "SELECT name,age |FROM user_info WHERE gender='F'";

$result = mysqli_query($dbh,$sel) or
die("Error querying the database");

while($row = mysqli_fetch_assoc($result))
{
echo "NAME:". $row['name'] ."AGE:". $row['age'] . "<br/>";
}

?>
```



NAME:MaryAGE:28

# Update data

dbop - Notepad

File Edit Format View Help

```
<?php
include("connect.php");

$sel = " UPDATE user_info SET age='30' WHERE id='3'"';

$result = mysqli_query($dbh,$sel) or
die("Error querying the database");

?>
```

Before update

Server: 127.0.0.1 » Database: demo » Table: user\_info

Browse Structure SQL Search Insert

Showing rows 0 - 2 (3 total, Query took 0.0017 seconds.)

```
SELECT * FROM `user_info`
```

Show all Number of rows: 25 Filter rows: Search this

+ Options

		id	name	age	gender
<input type="checkbox"/>	Edit  Copy  Delete	1	james	32	M
<input type="checkbox"/>	Edit  Copy  Delete	2	joseph	35	M
<input type="checkbox"/>	Edit  Copy  Delete	3	Mary	28	F

After update

Server: 127.0.0.1 » Database: demo » Table: user\_info

Browse Structure SQL Search Insert

Showing rows 0 - 2 (3 total, Query took 0.0142 seconds.)

```
SELECT * FROM `user_info`
```

Show all Number of rows: 25 Filter rows: Search this

+ Options

		id	name	age	gender
<input type="checkbox"/>	Edit  Copy  Delete	1	james	32	M
<input type="checkbox"/>	Edit  Copy  Delete	2	joseph	35	M
<input type="checkbox"/>	Edit  Copy  Delete	3	Mary	30	F

# Introduction to HTML

**HTML** stands for **Hyper Text Markup Language**, which is the most widely used language on Web to develop web pages.

**HTML 2.0** was the first standard HTML specification which was published in 1995.

**HTML 4.01** was a major version of HTML and it was published in late 1999.

currently we are having **HTML-5** version which was published in 2012.

**HTML** stands for **Hypertext Markup Language**, and it is the most widely used language to write Web Pages.

**Hypertext** refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.

As its name suggests, HTML is a **Markup Language** which means we use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

## **Basic HTML Document**

```
<html>
  <head>
    <title>This is document title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>Document content goes here.....</p>
  </body>
</html>
```

## **output**

This is a heading  
Document content goes here.....

# HTML Tags

HTML is a markup language and makes use of various tags to format the content.

These tags are enclosed within angle braces **<Tag Name>**.

Tags have their corresponding closing tags.

For example, **<html>** has its closing tag **</html>** and **<body>** tag has its closing tag **</body>** tag etc.

# Heading Tags

Any document starts with a heading.

We can use different sizes for your headings.

HTML also has six levels of headings, which use the elements **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>**, and **<h6>**.

While displaying any heading, browser adds one line before and one line after that heading.

```
<html>  
  <head>  
    <title>Heading Example</title>  
  </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>  
    <h5>This is heading 5</h5>  
    <h6>This is heading 6</h6>  
  </body>  
  
</html>
```

output

**This is heading 1**

**This is heading 2**

**This is heading 3**

**This is heading 4**

**This is heading 5**

**This is heading 6**

## Paragraph Tag

The **<p>** tag offers a way to structure your text into different paragraphs.

Each paragraph of text should go in between an opening **<p>** and a closing **</p>** tag

```
<html>
```

```
  <head>
    <title>Paragraph Example</title>
  </head>
```

```
  <body>
    <p>Here is a first paragraph of text.</p>
    <p>Here is a second paragraph of text.</p>
    <p>Here is a third paragraph of text.</p>
  </body>
```

```
</html>
```

Output

Here is a first paragraph of text.

Here is a second paragraph of text.

Here is a third paragraph of text.

## Line Break Tag

Whenever you use the **<br />** element, anything following it starts from the next line.

The **<br />** tag has a space between the characters **br** and the forward slash.

If you omit this space, older browsers will have trouble rendering the line break, while if you miss the forward slash character and just use **<br>** it is not valid in XHTML.

```
<html>
```

```
  <head>
```

```
    <title>Line Break Example</title>
```

```
  </head>
```

```
  <body>
```

```
    <p>Hello<br />
```

```
      You delivered your assignment on time.<br />
```

```
    Thanks<br />
```

```
    james</p>
```

```
  </body>
```

```
</html>
```

Output

Hello

You delivered your assignment on time.

Thanks

james

## Centering Content

You can use **<center>** tag to put any content in the center of the page or any table cell.

```
<html>
```

```
  <head>
    <title>Centring Content Example</title>
  </head>
```

```
<body>
  <p>This text is not in the center.</p>

  <center>
    <p>This text is in the center.</p>
  </center>
</body>
```

```
</html>
```

### Output

This text is not in the center.

This text is in the center.

## Horizontal Lines

Horizontal lines are used to visually break-up sections of a document.

The **<hr>** tag creates a line from the current position in the document to the right margin and breaks the line accordingly.

```
<html>
  <head>
    <title>Horizontal Line Example</title>
  </head>

  <body>
    <p>This is paragraph one and should be on top</p>
    <hr />
    <p>This is paragraph two and should be at
bottom</p>
  </body>

</html>
```

This is paragraph one and should be on top

---

This is paragraph two and should be at bottom

## Preserve Formatting

Sometimes, we want your text to follow the exact format of how it is written in the HTML document. In these cases, you can use the preformatted tag **<pre>**.

Any text between the opening **<pre>** tag and the closing **</pre>** tag will preserve the formatting of the source document.

```
<html>  
  
<head>  
  <title>Preserve Formatting Example</title>  
</head>  
  
<body>  
  <pre>  
    function testFunction( strText ){  
      alert (strText)  
    }  
  </pre>  
</body>  
  
</html>
```

Output

```
function testFunction( strText ){  
  alert (strText)  
}
```

align attribute: **left**, **center** and **right**.

```
<html>

<head>
    <title>Align Attribute Example</title>
</head>

<body>
    <p align = "left">This is left aligned</p>
    <p align = "center">This is center aligned</p>
    <p align = "right">This is right aligned</p>
</body>

</html>
```

Output

This is left aligned

This is center aligned

This is right aligned

Valign-top, middle, bottom: Vertically aligns tags within an HTML element.

## The title Attribute

The **title** attribute gives a suggested title for the element.

The behavior of this attribute will depend upon the element that carries it, although it is often displayed as a tooltip when cursor comes over the element or while the element is loading.

```
<html>  
  <head>  
    <title>The title Attribute Example</title>  
  </head>  
  
  <body>  
    <h3 title = "Hello HTML!">Titled Heading Tag  
Example</h3>  
  </body>  
  
</html>
```

## The dir Attribute

The **dir** attribute allows you to indicate to the browser about the direction in which the text should flow.

The dir attribute can take one of two values: ltr and rtl

```
<html dir = "rtl">
```

```
  <head>
    <title>Display Directions</title>
  </head>
```

```
  <body>
    This is an example for right-to-left directed
    text.
```

```
  </body>
```

```
</html>
```

## Background color

Bgcolor-numeric, hexidecimal, RGB values:Places a background color behind an element

## Bold Text

Anything that appears within **<b>...</b>** element, is displayed in bold

## Italic Text

Anything that appears within *<i>...</i>* element is displayed in italicized

## Underlined Text

Anything that appears within <u>...</u> element, is displayed with underline

## Strike Text

Anything that appears within **<strike>...</strike>** element is displayed with strikethrough

## Monospaced Font

The content of a **<tt>...</tt>** element is written in monospaced font. Most of the fonts are known as variable-width fonts because different letters are of different widths (for example, the letter 'm' is wider than the letter 'i'). In a monospaced font, however, each letter has the same width.

## Superscript Text

The content of a **<sup>...</sup>** element is written in superscript; the font size used is the same size as the characters surrounding it but is displayed half a character's height above the other characters.

## Subscript Text

The content of a **<sub>...</sub>** element is written in subscript; the font size used is the same as the characters surrounding it, but is displayed half a character's height beneath the other characters.

## Inserted Text

Anything that appears within `<ins>...</ins>` element is displayed as inserted text.

## Deleted Text

Anything that appears within `<del>...</del>` element, is displayed as deleted text.

## Larger Text

The content of the `<big>...</big>` element is displayed one font size larger than the rest of the text surrounding it

## Smaller Text

The content of the `<small>...</small>` element is displayed one font size smaller than the rest of the text surrounding it

## Grouping Content

The `<div>` and `<span>` elements allow you to group together several elements to create sections or subsections of a page.

For example, you might want to put all of the footnotes on a page within a `<div>` element to indicate that all of the elements within that `<div>` element relate to the footnotes. You might then attach a style to this `<div>` element so that they appear using a special set of style rules.

The `<span>` element, on the other hand, can be used to group inline elements only. So, if you have a part of a sentence or paragraph which you want to group together, you could use the `<span>`

```
<html>

<head>
  <title>Div Tag Example</title>
</head>

<body>
  <div id = "menu" align = "middle" >
    <a href = "/index.htm">HOME</a> |
    <a href = "/about/contact_us.htm">CONTACT</a> |
    <a href = "/about/index.htm">ABOUT</a>
  </div>

  <div id = "content" align = "left" bgcolor = "white"> </html>
    <h5>Content Articles</h5>
    <p>Actual content goes here.....</p>
  </div>
</body>

</html>
```

```
<html>

<head>
  <title>Span Tag Example</title>
</head>

<body>
  <p>This is the example of <span style =
"color:green">span tag</span>
  and the <span style = "color:red">div tag</span>
  alongwith CSS</p>
</body>
```

## Emphasized Text

Anything that appears within `<em>...</em>` element is displayed as emphasized text.

## Marked Text

Anything that appears with-in `<mark>...</mark>` element, is displayed as marked with yellow ink.

## Strong Text

Anything that appears within `<strong>...</strong>` element is displayed as important text.

## Text Direction

The `<bdo>...</bdo>` element stands for Bi-Directional Override and it is used to override the current text direction.

## Short Quotations

The `<q>...</q>` element is used when you want to add a double quote within a sentence.

## Address Text

The `<address>...</address>` element is used to contain any address.

```
<bdo dir="rtl">  
This text will go right-to-left.  
</bdo>
```

HTML has several list elements. Most list elements are composed of one or more <LI> (List Item) elements.

UL : Unordered List. Items in this list start with a list mark such as a bullet. Browsers will usually change the list mark in nested lists.

```
<UL>  
<LI> List item ...</LI>  
<LI> List item ...</LI>  
</UL>
```

Choice of three bullet types: **disc(default), circle, square.**

These are controlled in Netscape Navigator by the “TYPE” attribute for the <UL> element.

```
<UL TYPE="square">
<LI> List item ...
<LI> List item ...
<LI> List item ...
</UL>
```

OL: Ordered List. Items in this list are numbered automatically by the browser.

```
<OL>
<LI> List item ...
<LI> List item ...
<LI> List item ...
</OL>
```

choice of setting the TYPE Attribute to one of five numbering styles.

TYPE	Numbering Styles	
1	Arabic numbers	1,2,3, .....
a	Lower alpha	a, b, c, .....
A	Upper alpha	A, B, C, .....
i	Lower roman	i, ii, iii, .....
I	Upper roman	I, II, III, .....

specify a starting number for an ordered list.

**<OL TYPE =“i”>**

<LI> List item ...</LI>

<LI> List item ...</LI>

**</OL>**

<P> text ....</P>

**<OL TYPE=“i” START=“3”>**

**<LI> List item ...</LI>**

**</OL>**

**DL: Definition List.** This kind of list is different from the others. Each item in a DL consists of one or more **Definition Terms (DT elements)**, followed by one or more **Definition Description (DD elements)**.

```
<DL>
<DT> HTML </DT>
<DD> Hyper Text Markup Language </DD>
<DT> DOG </DT>
<DD> A human's best friend!</DD>
</DL>
```

HTML

Hyper Text Markup Language

DOG

A human's best friend!

can nest lists by inserting a UL, OL, etc., inside a list item (LI).

## EXample

```
<UL TYPE = "square">
<LI> List item ...</LI>
<LI> List item ...
<OL TYPE="i" START="3">
<LI> List item ...</LI>
</OL>
</LI>
<LI> List item ...</LI>
</UL>
```

# SAFETY TIPS FOR

## **OL TYPE="a" START="2">**

<LI>Be able to swim </LI>

<LI>Wear a life jacket at all times </LI>

<LI>Don't stand up or move around. If canoe tips,

### **UL**

<LI>Hang on to the canoe </LI>

<LI>Use the canoe for support and </LI>

<LI>Swim to shore

### **UL** </LI>

<LI>Don't overexert yourself </LI>

<LI>Use a bow light at night </LI>

## **OL**

The <TABLE></TABLE> element has four sub-elements:

- Table Row <TR></TR>.
- Table Header <TH></TH>.
- Table Data <TD></TD>.
- Caption <CAPTION></CAPTION>.

The table row elements usually contain table header elements or table data elements.

```
<table border="1">  
<tr>  
  <th> Column 1 header </th>  
  <th> Column 2 header </th>  
</tr>  
<tr>  
  <td> Row1, Col1 </td>  
  <td> Row1, Col2 </td>  
</tr>  
<tr>  
  <td> Row2, Col1 </td>  
  <td> Row2, Col2 </td>  
</tr>  
</table>
```

<b>Column 1 Header</b>	<b>Column 2 Header</b>
Row1, Col1	Row1, Col2
Row2, Col1	Row2, Col2

## TABLE ATTRIBUTES

**BGColor:** Some browsers support background colors in a table.

**Width:** you can specify the table width as an absolute number of pixels or a percentage of the document width. You can set the width for the table cells as well.

**Border:** You can choose a numerical value for the border width, which specifies the border in pixels.

**CellSpacing:** Cell Spacing represents the space between cells and is specified in pixels.

## TABLE ATTRIBUTES

**CellPadding:** Cell Padding is the space between the cell border and the cell contents and is specified in pixels.

**Align:** tables can have left, right, or center alignment.

**Background:** Background Image, will be titled in IE3.0 and above.

**BorderColor, BorderColorDark.**

A table caption allows you to specify a line of text that will appear centered above or below the table.

```
<TABLE BORDER=1 CELLPADDING=2>  
<CAPTION ALIGN="BOTTOM"> Label For My Table  
</CAPTION>
```

The Caption element has one attribute ALIGN that can be either TOP (Above the table) or BOTTOM (below the table).

```
<html>

    <head>
        <title>HTML Table Cellpadding</title>
    </head>

    <body>
        <table border = "1" cellpadding = "5" cellspacing = "5">
            <tr>
                <th>Name</th>
                <th>Salary</th>
            </tr>
            <tr>
                <td>Ramesh Raman</td>
                <td>5000</td>
            </tr>
            <tr>
                <td>Shabbir Hussein</td>
                <td>7000</td>
            </tr>
        </table>
    </body>

</html>
```

## Reference tag/anchor tag

The tags used to produce links are the `<A>` and `</A>`.

The `<A>` tells where the link should start and  
the `</A>` indicates where the link ends.

Everything between these two will work as a link.

The example below shows how to make the word **Here** work as a link to yahoo.

Click `<A HREF="http://www.yahoo.com">here</A>`

Internal Links : Links can also be created inside large documents to simplify navigation.

Select some text at a place in the document that you would like to create a link to, then add an anchor to link to like this:

`<A NAME="bookmark_name"></A>`

The Name attribute of an anchor element specifies a location in the document that we link to shortly. All NAME attributes in a document must be unique.

Next select the text that you would like to create as a link to the location created above.

`<A HREF="#bookmark_name">Go To Book Mark</A>`

**HTML comments** are placed in between `<!-- ... -->` tags. So, any content placed with-in `<!-- ... -->` tags will be treated as comment and will be completely ignored by the browser.

```
<html>

  <head> <!-- Document Header Starts -->
    <title>This is document title</title>
  </head> <!-- Document Header Ends -->

  <body>
    <p>Document content goes here....</p>
  </body>

</html>
```

# Insert Image

You can insert any image in your web page by using **<img>** tag.

```
<html>
  <head>
    <title>Using Image in Webpage</title>
  </head>

  <body>
    <p>Simple Image Insert</p>
    <img src = "/html/images/test.png" />
  </body>

</html>
```

## Set Image Width/Height

You can set image width and height based on your requirement using width and height attributes.

## Set Image Border

By default, image will have a border around it, you can specify border thickness in terms of pixels using border attribute.

## Set Image Alignment

By default, image will align at the left side of the page, but you can use **align** attribute to set it in the center or right.

```
<html>
```

```
<head>
```

```
    <title>Set Image Alignment</title>
```

```
</head>
```

```
<body>
```

```
    <p>Setting image Alignment</p>
```

```
    <img src = "/html/images/test.png"
        alt = "Test Image" border = "3" align =
        "right"/>
    </body>
```

```
</html>
```

## Colspan and Rowspan Attributes

We will use **colspan** attribute if you want to merge two or more columns into a single column. Similar way use **rowspan** if we want to merge two or more rows.

```
<html>
  <head>
    <title>HTML Table Colspan/Rowspan</title>
  </head>
  <body>
    <table border = "1">
      <tr>
        <th>Column 1</th>
        <th>Column 2</th>
        <th>Column 3</th>
      </tr>
      <tr>
        <td rowspan = "2">Row 1 Cell 1</td>
        <td>Row 1 Cell 2</td>
        <td>Row 1 Cell 3</td>
      </tr>
      <tr>
        <td>Row 2 Cell 2</td>
        <td>Row 2 Cell 3</td>
      </tr>
      <tr>
        <td colspan = "3">Row 3 Cell 1</td>
      </tr>
    </table>
  </body>
</html>
```

## Table Height and Width

You can set a table width and height using width and height attributes.

```
<html>
  <head>
    <title>HTML Table Width/Height</title>
  </head>
  <body>
    <table border = "1" width = "400" height =
    "150">
      <tr>
        <td>Row 1, Column 1</td>
        <td>Row 1, Column 2</td>
      </tr>
      <tr>
        <td>Row 2, Column 1</td>
        <td>Row 2, Column 2</td>
      </tr>
    </table>
  </body>
</html>
```

## **Table Header, Body, and Footer**

Tables can be divided into three portions – a header, a body, and a foot.

- <**thead**> – to create a separate table header.
- <**tbody**> – to indicate the main body of the table.
- <**tfoot**> – to create a separate table footer.

## HTML Email Tag

HTML `<a>` tag provides you option to specify an email address to send an email. While using `<a>` tag as an email tag, you will use **mailto: email address** along with `href` attribute.

```
<a href = "mailto: abc@example.com">Send Email</a>
```

# Frames

- Frames are rectangular sections of the display window, each of which can display a different document
- The <frameset> tag specifies the number of frames and their layout in the window
- <frameset> takes the place of <body>
- Cannot have both!
- <frameset> must have either a rows attribute or a cols attribute, or both (usually the case)
- Default is 1
- The possible values for rows and cols are numbers, percentages, and asterisks
- A number value specifies the row height in pixels - Not terribly useful!
- A percentage specifies the percentage of total window height for the row - Very useful!

# Frames (continued)

- An asterisk after some other specification gives the remainder of the height of the window
- Examples:

```
<frameset rows = "150, 200, 300">
```

```
<frameset rows = "25%, 50%, 25%">
```

```
<frameset rows = "50%, 20%, *" >
```

```
<frameset rows = "50%, 25%, 25%"  
          cols = "40%, *">
```

The <frame> tag specifies the content of a frame

The first <frame> tag in a <frameset> specifies the content of the first frame, etc.

Row-major order is used

Frame content is specified with the src attribute

Without a src attribute, the frame will be empty (such a frame CANNOT be filled later)

If <frameset> has fewer <frame> tags than frames, the extra frames are empty

# Creating Frames

To use frames on a page we use <frameset> tag instead of <body> tag.

The <frameset> tag defines, how to divide the window into frames.

The **rows** attribute of <frameset> tag defines horizontal frames and **cols** attribute defines vertical frames.

**Note:** HTML 5 is not supporting frame tag

```
<html>
```

```
  <head>
```

```
    <title>HTML Frames</title>
```

```
  </head>
```

```
  <frameset rows = "10%,80%,10%">
```

```
    <frame name = "top" src = "/html/top_frame.htm" />
```

```
    <frame name = "main" src = "/html/main_frame.htm" />
```

```
    <frame name = "bottom" src = "/html/bottom_frame.htm" />
```

```
  <noframes>
```

```
    <body>Your browser does not support frames.</body>
```

```
  </noframes>
```

```
</frameset>
```

```
</html>
```

We can define an inline frame with HTML tag **<iframe>**.

The **<iframe>** tag is not somehow related to **<frameset>** tag, instead, it can appear anywhere in your document.

The **<iframe>** tag defines a rectangular region within the document in which the browser can display a separate document, including scrollbars and borders.

An inline frame is used to embed another document within the current HTML document.

```
<html>

<head>
    <title>HTML Iframes</title>
</head>

<body>
    <p>Document content goes here...</p>

    <iframe src = "/html/menu.htm" width = "555" height = "200">
        Sorry your browser does not support inline frames.
    </iframe>

    <p>Document content also go here...</p>
</body>

</html>
```

## Set Font Size

You can set content font size using **size** attribute. The range of accepted values is from 1(smallest) to 7(largest). The default size of a font is 3.

```
<html>

  <head>
    <title>Setting Font Size</title>
  </head>

  <body>
    <font size = "1">Font size = "1"</font><br />
    <font size = "2">Font size = "2"</font><br />
    <font size = "3">Font size = "3"</font><br />
    <font size = "4">Font size = "4"</font><br />
    <font size = "5">Font size = "5"</font><br />
    <font size = "6">Font size = "6"</font><br />
    <font size = "7">Font size = "7"</font>
  </body>

</html>
```

# The <marquee> Tag

An HTML marquee is a scrolling piece of text displayed either horizontally across or vertically down your webpage depending on the settings.

```
<html>

    <head>
        <title>HTML marquee Tag</title>
    </head>

    <body>
        <marquee>This is basic example of marquee</marquee>
    </body>

</html>
```

```
<body>
  <marquee direction = "right">This text will scroll from left to right</marquee>
</body>
```

```
<body>
  <marquee direction = "up">This text will scroll from bottom to up</marquee>
</body>
```

# The HTML Style Attribute

```
<tagname style="property:value;">  
  
    <body style="background-color:powderblue;">  
        <h1 style="color:blue;">This is a heading</h1>  
        <p style="color:red;">This is a paragraph.</p>  
        <h1 style="font-family:verdana;">This is a heading</h1>  
        <p style="font-family:courier;">This is a paragraph.</p>  
        <h1 style="font-size:300%;">This is a heading</h1>  
        <p style="font-size:160%;">This is a paragraph.</p>  
        <h1 style="text-align:center;">Centered Heading</h1>  
        <p style="text-align:center;">Centered paragraph.</p>
```

## HTML Colors

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>
```

```
<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

## Image as a Link

```
<a href="default.asp">  
    
</a>
```

## Image Floating

```
<p>   
The image will float to the right of the text.</p>
```

```
<p>  
The image will float to the left of the text.</p>
```

# Image Maps

The <map> tag defines an image-map. An image-map is an image with clickable areas.

In the image below, click on the computer, the phone, or the cup of coffee:



```


<map name="workmap">

  <area shape="rect" coords="34,44,270,350" alt="Computer"
    href="computer.htm">

  <area shape="rect" coords="290,172,333,250" alt="Phone"
    href="phone.htm">

  <area shape="circle" coords="337,300,44" alt="Coffee"
    href="coffee.htm">

</map>
```

# Background Image

```
<body style="background-image:url('clouds.jpg');">
```

```
<h2>Background Image</h2>
```

```
</body>
```

```
<p style="background-image:url('clouds.jpg');">
```

```
...
```

```
</p>
```

# The <div> Element

The <div> element is often used as a container for other HTML elements.

```
<div style="background-color:black;color:white;padding:20px;">
  <h2>London</h2>
  <p>London is the capital city of England. It is the most populous
city in the United Kingdom, with a metropolitan area of over 13
million inhabitants.</p>
</div>
```

## The <span> Element

The <span> element is often used as a container for some text.

```
<h1>My <span style="color:red">Important</span> Heading</h1>
```

# HTML Iframes

An iframe is used to display a web page within a web page.

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

```
<iframe src="demo_iframe.htm" height="200"  
width="300"></iframe>
```

```
<iframe src="demo_iframe.htm"  
style="height:200px;width:300px;"></iframe>
```

```
<iframe src="demo_iframe.htm" style="border:none;"></iframe>
```

```
<iframe src="demo_iframe.htm" style="border:2px solid  
red;"></iframe>
```

## Iframe - Target for a Link

An iframe can be used as the target frame for a link.

The target attribute of the link must refer to the name attribute of the iframe:

```
<iframe src="demo_iframe.htm"  
name="iframe_a"></iframe>
```

```
<p><a href="https://www.google.com"  
target="iframe_a">Load new web page</a></p>
```

# HTML Entities

Result	Description	Entity Name
	non-breaking space	&nbsp;
<	less than	&lt;
>	greater than	&gt;
&	ampersand	&amp;
"	double quotation mark	&quot;
'	single quotation mark (apostrophe)	&apos;
¢	cent	&cent;
£	pound	&pound;
¥	yen	&yen;
€	euro	&euro;
©	copyright	&copy;
®	registered trademark	&reg;

## HTML - Embed Multimedia

Sometimes you need to add music or video into your web page. The easiest way to add video or sound to your web site is to include the special HTML tag called <embed>. This tag causes the browser itself to include controls for the multimedia automatically provided browser supports <embed> tag and given media type.

You can also include a <noembed> tag for the browsers which don't recognize the <embed> tag. You could, for example, use <embed> to display a movie of your choice, and <noembed> to display a single JPG image if browser does not support <embed> tag.

```
<html>

<head>
    <title>HTML embed Tag</title>
</head>

<body>
    <embed src = "/html/yourfile.mid" width = "100%" height = "60" >
        <noembed><img src = "yourimage.gif" alt = "Alternative Media" ></noembed>
    </embed>
</body>

</html>
```

Sr.No	Attribute & Description
1	<b>align</b> Determines how to align the object. It can be set to either center, <i>left or right</i> .
2	<b>autoplay</b> This boolean attribute indicates if the media should start automatically. You can set it either true or false.
3	<b>loop</b> Specifies if the sound should be played continuously (set loop to true), a certain number of times (a positive value) or not at all (false)
4	<b>playcount</b> Specifies the number of times to play the sound. This is alternate option for <i>loop</i> if you are usiong IE.
5	<b>hidden</b> Specifies if the multimedia object should be shown on the page. A false value means no and true values means yes.
6	<b>width</b> Width of the object in pixels
7	<b>height</b> Height of the object in pixels
8	<b>name</b> A name used to reference the object.
9	<b>src</b> URL of the object to be embedded.
10	<b>volume</b> Controls volume of the sound. Can be from 0 (off) to 100 (full volume).

## Background Audio

You can use HTML `<bgsound src="">` tag to play a soundtrack in the background of your webpage. This tag is supported by Internet Explorer only and most of the other browsers ignore this tag. It downloads and plays an audio file when the host document is first downloaded by the user and displayed. The background sound file also will replay whenever the user refreshes the browser.

# CSS

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page.

Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs.

## Advantages of CSS

**CSS saves time** – We can write CSS once and then reuse same sheet in multiple HTML pages. We can define a style for each HTML element and apply it to as many Web pages as we want.

**Pages load faster** – We do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.

**Easy maintenance** – To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.

**Superior styles to HTML** – CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

**Global web standards** – Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

**Platform Independence** – The Script offer consistent platform independence and can support latest browsers as well.

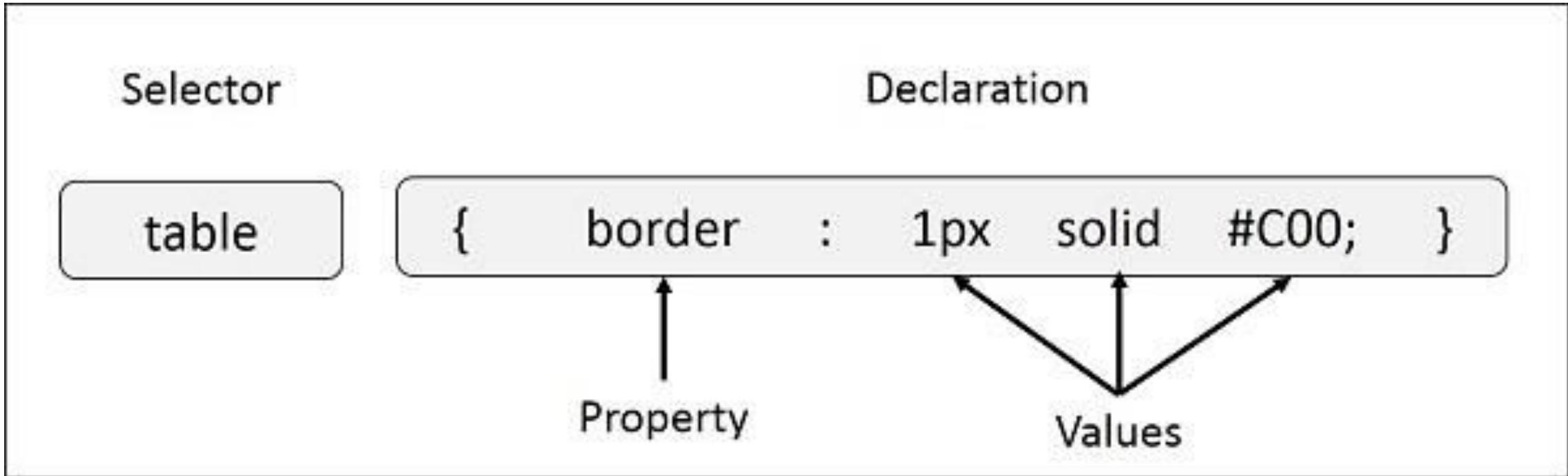
A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts –

**Selector** – A selector is an HTML tag at which a style will be applied. This could be any tag like `<h1>` or `<table>` etc.

**Property** - A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.

**Value** - Values are assigned to properties. For example, color property can have value either red or `#F1F1F1` etc.

selector { property: value }



## The Type Selectors

```
h1 {  
    color: #36CFFF;  
}
```

## The Universal Selectors

```
* {  
    color: #000000;  
}
```

## The Descendant Selectors

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element.

```
ul em {  
    color: #000000;  
}
```

## The Class Selectors

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {  
    color: #000000;  
}
```

```
h1.black {  
    color: #000000;  
}
```

## The ID Selectors

You can define style rules based on the id attribute of the elements. All the elements having that id will be formatted according to the defined rule.

```
#black {  
    color: #000000;  
}
```

```
h1#black {  
    color: #000000;  
}
```

## The Child Selectors

```
body > p {  
    color: #000000;  
}
```

This rule will render all the paragraphs in black if they are direct child of `<body>` element.

Other paragraphs put inside other elements like `<div>` or `<td>` would not have any effect of this rule.

**Multiple Style Rules:** combine multiple properties and corresponding values into a single block

```
h1 {  
    color: #36C;  
    font-weight: normal;  
    letter-spacing: .4em;  
    margin-bottom: 1em;  
    text-transform: lowercase;  
}
```

All the property and value pairs are separated by a semi colon (;).

Grouping Selectors: apply a style to many selectors . Just separate the selectors with a comma

```
h1, h2, h3 {  
    color: #36C;  
    font-weight: normal;  
    letter-spacing: .4em;  
    margin-bottom: 1em;  
    text-transform: lowercase;  
}
```

## Embedded CSS - The <style> Element

We can put your CSS rules into an HTML document using the <style> element.

This tag is placed inside <head>...</head> tags.

Rules defined using this syntax will be applied to all the elements available in the document.

```
<html>
  <head>

    <style type = "text/css" media = "all">
      body {
        background-color: linen;
      }
      h1 {
        color: maroon;
        margin-left: 40px;
      }
    </style>

  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

## Inline CSS - The style Attribute

We can use style attribute of any HTML element to define style rules.

These rules will be applied to that element only.

```
<html>
  <head>
  </head>
  <body>
    <h1 style = "color:#36C;"> This is inline CSS </h1>
  </body>
</html>
```

## External CSS - The <link> Element

The <link> element can be used to include an external stylesheet file in your HTML document.

An external style sheet is a separate text file **with .css extension.**

We define all the Style rules within this text file and then you can include this file in any HTML document using <link> element.

```
<head>
  <link rel="stylesheet" type = "text/css" href = "mystyle.css" />
</head>
```

## Imported CSS - @import Rule

@import is used to import an external stylesheet in a manner similar to the <link> element.

```
<head>
  @import "mystyle.css";
</head>
```

## CSS Rules Overriding

Any inline style sheet takes highest priority. So, it will override any rule defined in `<style>...</style>` tags or rules defined in any external style sheet file.

Any rule defined in `<style>...</style>` tags will override rules defined in any external style sheet file.

Any rule defined in external style sheet file takes lowest priority, and rules defined in this file will be applied only when above two rules are not applicable.

# CSS - Measurement Units

CSS supports a number of measurements including absolute units such as inches, centimeters, points, and so on, as well as relative measures such as percentages and em units.

%	Defines a measurement as a percentage relative to another value, typically an enclosing element.	<pre>p {font-size: 16pt; line-height: 125%;}</pre>
cm	Defines a measurement in centimeters.	<pre>div {margin-bottom: 2cm;}</pre>

em	A relative measurement for the height of a font in em spaces. Because an em unit is equivalent to the size of a given font, if you assign a font to 12pt, each "em" unit would be 12pt; thus, 2em would be 24pt.	p {letter-spacing: 7em;}
ex	This value defines a measurement relative to a font's x-height. The x-height is determined by the height of the font's lowercase letter x.	p {font-size: 24pt; line-height: 3ex;}
in	Defines a measurement in inches.	p {word-spacing: .15in;}

mm	Defines a measurement in millimeters.	p {word-spacing: 15mm;}
pc	Defines a measurement in picas. A pica is equivalent to 12 points; thus, there are 6 picas per inch.	p {font-size: 20pc;}
pt	Defines a measurement in points. A point is defined as 1/72nd of an inch.	body {font-size: 18pt;}
px	Defines a measurement in screen pixels.	p {padding: 25px;}

## CSS Colors - Hex Codes

A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).

CSS uses color values to specify a color.

Typically, these are used to set a color either for the foreground of an element (i.e., its text) or else for the background of the element.

They can also be used to affect the color of borders and other decorative effects.

Format	Syntax	Example
Hex Code	#RRGGBB	p{color:#FF0000;}
Short Hex Code	#RGB	p{color:#6A7;}
RGB %	rgb(rrr%,ggg%,bbb%)	p{color:rgb(50%,50%,50%);}
RGB Absolute	rgb(rrr,ggg,bbb)	p{color:rgb(0,0,255);}
keyword	aqua, black, etc.	p{color:teal;}

## Set the Background Color

```
<html>
  <head>
  <body>
    <p style = "background-color:yellow;">
      This text has a yellow background color.</p>
    </body>
  </head>
<html>
```

## Set the Background Image

```
<html>
  <head>
    <style>
      body {
        background-image: url("/css/images/css.jpg");
      }
    </style>
  <body>
    <h1>Hello World!</h1>
  </body>
</head>
<html>
```

## Repeat the Background Image

The following example demonstrates how to repeat the background image if an image is small.

```
<head>
    <style>
        body {
            background-image: url("/css/images/css.jpg");
            background-repeat: repeat;
        }
    </style>
</head>
<body>
    <p>Tutorials point</p>
</body>
```

To repeat the background image vertically.

```
<style>
  body {
    background-image: url("/css/images/css.jpg");
    background-repeat: repeat-y;
  }
</style>
```

To repeat the background image horizontally.

```
<style>
  body {
    background-image: url("/css/images/css.jpg");
    background-repeat: repeat-x;
  }
</style>
```

## Set the Background Attachment

Background attachment determines whether a background image is fixed or scrolls with the rest of the page.

```
<style>
  body {
    background-image: url('/css/images/css.jpg');
    background-repeat: no-repeat;
    background-attachment: fixed;
  }
</style>
```

```
<style>
  body {
    background-image: url('/css/images/css.jpg');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-attachment: scroll;
  }
</style>
```

## Set the Font Style

```
<body>
```

```
  <p style="font-style:italic;">
```

```
    This text will be rendered in italic style
```

```
  </p>
```

```
</body>
```

## Set the Font Family

```
<body>
```

```
  <p style="font-family:georgia, garamond, serif;">
```

```
    This text is rendered in either georgia, garamond, or the default serif font  
    depending on which font you have at your system.
```

```
  </p>
```

```
</body>
```

## Set the Font Size

Possible values could be *xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, size in pixels or in %.*

```
<body>
  <p style="font-size:20px;">This font size is 20 pixels</p>
  <p style="font-size:small;">This font size is small</p>
  <p style="font-size:large;">This font size is large</p>
</body>
```

## Set the Font Weight

The following example demonstrates how to set the font weight of an element. The font-weight property provides the functionality to specify how bold a font is. Possible values could be normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900.

```
<body>
  <p style="font-weight:bold;">This font is bold.</p>
  <p style="font-weight:bolder;">This font is bolder.</p>
  <p style="font-weight:500;">This font is 500 weight.</p>
</body>
```

## Set the Font Stretch

Possible values could be *normal*, *wider*, *narrower*, *ultra-condensed*, *extra-condensed*, *condensed*, *semi-condensed*, *semi-expanded*, *expanded*, *extra-expanded*, *ultra-expanded*.

```
<body>
```

```
  <p style="font-stretch:ultra-expanded;">
```

If this doesn't appear to work, it is likely that your computer doesn't have a condensed or expanded version of the font being used.

```
  </p>
```

```
</body>
```

Set the Text Color

```
<p style="color:red;">
```

Set the Text Direction

```
<p style="direction:rtl;">
```

Set the Space between Characters

```
<p style="letter-spacing:5px;">
```

Set the Space between Words

```
<p style="word-spacing:5px;">
```

Set the Text Indent

```
<p style="text-indent:1cm;">
```

## Set the Text Alignment

```
<body>
```

```
  <p style="text-align:right;">
```

```
    This will be right aligned.
```

```
  </p>
```

```
  <p style="text-align:center;">
```

```
    This will be center aligned.
```

```
  </p>
```

```
  <p style="text-align:left;">
```

```
    This will be left aligned.
```

```
  </p>
```

```
</body>
```

## Decorating the Text

```
<body>
  <p style="text-decoration:underline;">
    This will be underlined
  </p>
```

```
<p style="text-decoration:line-through;">
  This will be striked through.
</p>
```

```
<p style="text-decoration:overline;">
  This will have a over line.
</p>
```

```
<p style="text-decoration:blink;">
  This text will have blinking effect
</p>
</body>
```

## Set the Text Cases

```
<body>
  <p style="text-transform:capitalize;">
    This will be capitalized
  </p>
```

```
<p style="text-transform:uppercase;">
  This will be in uppercase
</p>
```

```
<p style="text-transform:lowercase;">
  This will be in lowercase
</p>
</body>
```

## Set the Text Shadow

```
<body>
  <p style="text-shadow:4px 4px 8px blue;">
    If your browser supports the CSS text-shadow property, this text will have
    a blue shadow.
  </p>
</body>
```

# The Image Border Property

```
<body>
  
  <br />
  
</body>
```

# The Image Height Property

```
<body>
  
  <br />
  
</body>
```

## The Image Width Property

```
<body>
  
  <br />
  
</body>
```

## The opacity Property

In Mozilla (-moz-opacity:x) x can be a value from 0.0 - 1.0. A lower value makes the element more transparent.

In IE (filter:alpha(opacity=x)) x can be a value from 0 - 100. A lower value makes the element more transparent.

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

`#id1{ property:value; }-----→ Id selector`

`.mycolour{ property:value;}-----→class selector`

`<p id="id1">tour data</p>`

`<h1 id="id1">your data</h1>`

`<p class="mycolour">your data<p>`



# 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

```
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;}
```

# Border Width

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

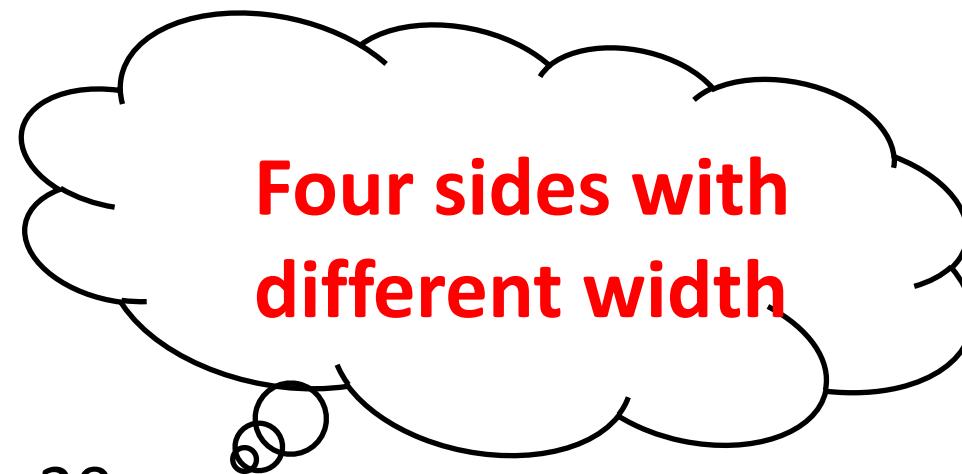
The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: [thin](#), [medium](#), or [thick](#).

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

```
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;  
}
```



## Border Color

The border-color property is used to set the color of the four borders.

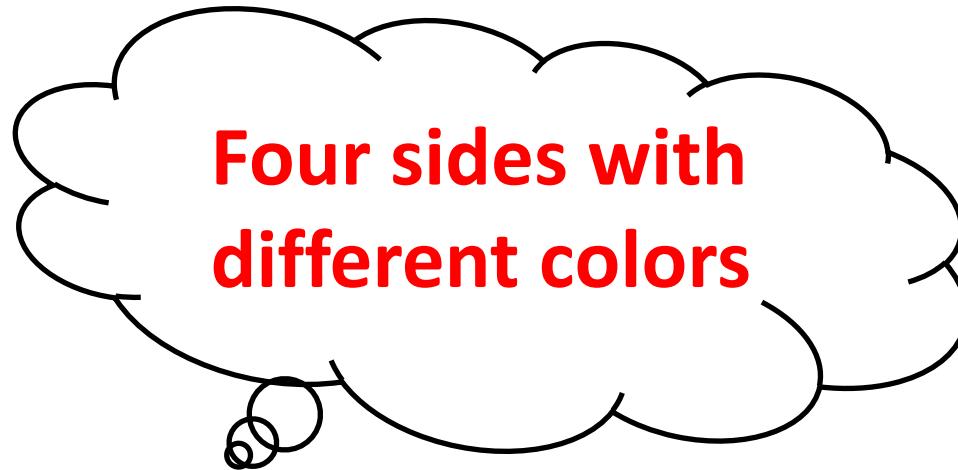
The color can be set by:

- name - specify a color name, like "red"
- Hex - specify a hex value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"

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

If border-color is not set, it inherits the color of the element.

```
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;  
}
```



## Border - Individual Sides

From the examples above you have seen that it is possible to specify a different border for each side.

In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left):

```
p {  
    border-top-style: dotted;  
    border-right-style: solid;  
    border-bottom-style: dotted;  
    border-left-style: solid;  
}
```

```
p {  
    border-left: 6px solid red;  
    background-color: lightgrey;  
}
```

# 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).

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

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

```
margin: 25px 50px 75px 100px;
```

top margin is 25px

right margin is 50px

bottom margin is 75px

left margin is 100px

```
p {  
    margin: 25px 50px 75px  
        100px;  
}
```

# The auto Value

You can set the margin property to auto to horizontally center the element within its container.

The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:

```
div {  
    width: 300px;  
    margin: auto;  
    border: 1px solid red;  
}
```

# CSS Padding

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

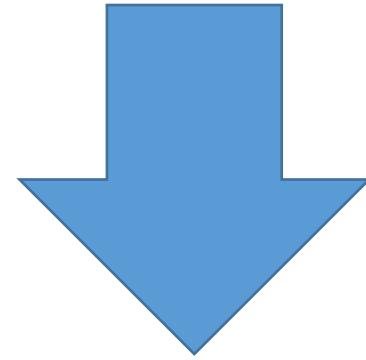
With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

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

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

```
div {  
padding-top: 50px;  
padding-right: 30px;  
padding-bottom: 50px;  
padding-left: 80px;  
}
```

```
div {  
padding: 25px 50px 75px 100px;  
}
```



top padding is 25px  
right padding is 50px  
bottom padding is 75px  
left padding is 100px

## Padding and Element Width

The CSS width property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element (the box model).

```
div {  
    width: 300px;  
    padding: 25px;  
}
```

## **Setting height and width**

The height and width properties are used to set the height and width of an element.

The height and width can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in length values, like px, cm, etc., or in percent (%) of the containing block.

```
div {  
    height: 200px;  
    width: 50%;  
    background-color: powderblue;  
}
```

```
div {  
    height: 100px;  
    width: 500px;  
    background-color: powderblue;  
}
```

# The CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

Explanation of the different parts:

- Content - The content of the box, where text and images appear
- Padding - Clears an area around the content. The padding is transparent
- Border - A border that goes around the padding and content
- Margin - Clears an area outside the border. The margin is transparent

```
div {  
    width: 300px;  
    border: 25px solid green;  
    padding: 25px;  
    margin: 25px;  
}
```

# The display Property

The display property specifies if/how an element is displayed.

```
li {  
    display: inline;  
}  
  
span {  
    display: block;  
}  
  
a {  
    display: block;  
}
```

```
span.a {  
    display: inline; /* the default for span */  
    width: 100px;  
    height: 100px;  
    padding: 5px;  
    border: 1px solid blue;  
    background-color: yellow;  
}
```

```
span.b {  
    display: inline-block;  
    width: 100px;  
    height: 100px;  
    padding: 5px;  
    border: 1px solid blue;  
    background-color: yellow;  
}
```

```
span.c {  
    display: block;  
    width: 100px;  
    height: 100px;  
    padding: 5px;  
    border: 1px solid blue;  
    background-color: yellow;  
}
```

# CSS Links

With CSS, links can be styled in different ways.

Text Link

Text Link

Link Button

Link Button

In addition, links can be styled differently depending on what state they are in.

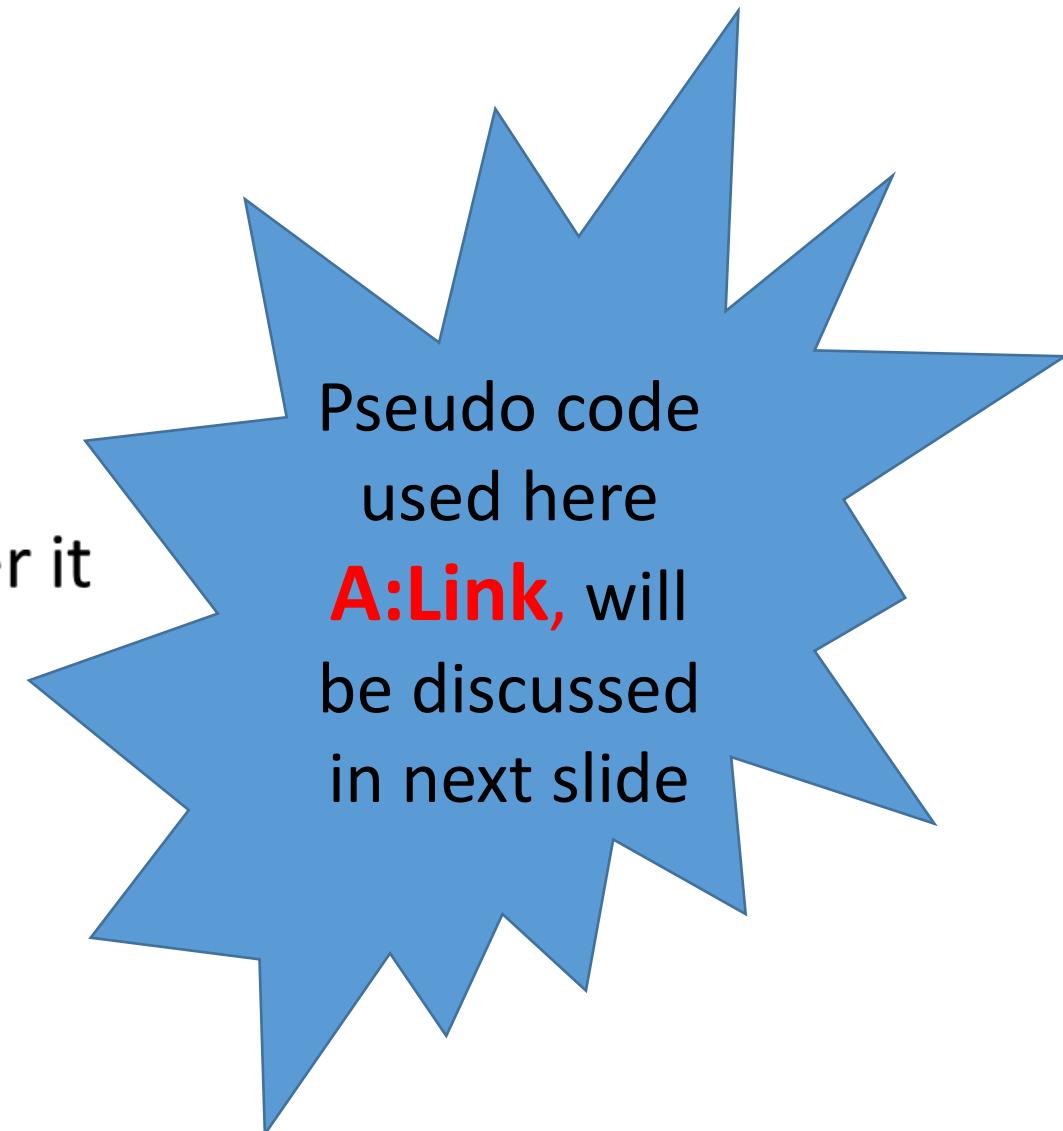
The four links states are:

a:link - a normal, unvisited link

a:visited - a link the user has visited

a:hover - a link when the user mouses over it

a:active - a link the moment it is clicked



```
/* unvisited link */  
a:link {  
    color: red;  
}  
  
/* visited link */  
a:visited {  
    color: green;  
}  
  
/* mouse over link */  
a:hover {  
    color: hotpink;  
}  
  
/* selected link */  
a:active {  
    color: blue;  
}
```

```
a:link {  
    text-decoration: none;  
}  
  
a:visited {  
    text-decoration: none;  
}  
  
a:hover {  
    text-decoration: underline;  
}  
  
a:active {  
    text-decoration: underline;  
}
```

```
a:link {  
    background-color: yellow;  
}  
  
a:visited {  
    background-color: cyan;  
}
```

```
a:hover {  
    background-color: lightgreen;  
}  
  
a:active {  
    background-color: hotpink;  
}
```

```
a:link, a:visited {  
    background-color: #f44336;  
    color: white;  
    padding: 14px 25px;  
    text-align: center;  
    text-decoration: none;  
    display: inline-block;  
}  
  
a:hover, a:active {  
    background-color: red;  
}
```

Task:

Create a navigation link as below

Home

Contact

About us

## CSS 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

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

```
a.highlight:hover {  
    color: #ff0000;  
}
```

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

```
div:hover {
    background-color: blue;
}
</style>
```

```
<style>
p {
    display: none;
    background-color: yellow;
    padding: 20px;
}

```

```
div:hover p {
    display: block;
}
</style>
```

# CSS Pseudo-elements

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

Style the first letter, or line, of an element

Insert content before, or after, the content of an element

```
selector::pseudo-element {  
    property:value;  
}
```

```
p::first-line {  
    color: #ff0000;  
    font-variant: small-caps;  
}
```

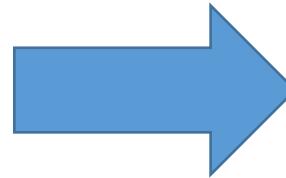
```
p::first-letter {  
    color: #ff0000;  
    font-size: xx-large;  
}
```

## Multiple Pseudo-elements

```
p::first-letter {  
    color: #ff0000;  
    font-size: xx-large;  
}
```

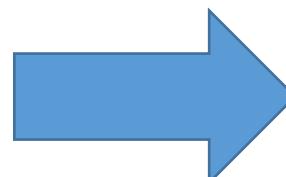
```
p::first-line {  
    color: #0000ff;  
    font-variant: small-caps;  
}
```

```
h1::before {  
    content: url(smiley.gif);  
}
```



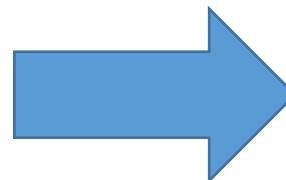
insert some content  
before the content of  
an element.

```
h1::after {  
    content: url(smiley.gif);  
}
```



insert some content  
after the content of an  
element.

```
::selection {  
    color: red;  
    background: yellow;  
}
```



Highlight the portion  
of an element that is  
selected by a user.

# CSS Opacity / Transparency

```
img {  
    opacity: 0.5;  
    filter: alpha(opacity=50);  
}
```

```
img {  
    opacity: 0.5;  
    filter: alpha(opacity=50); /* For IE8 and earlier */  
}
```

```
img:hover {  
    opacity: 1.0;  
    filter: alpha(opacity=100); /* For IE8 and earlier */  
}
```

## Transparent Box

When using the opacity property to add transparency to the background of an element, all of its child elements inherit the same transparency.

```
<style>
div {
    background-color: #4CAF50;
    padding: 10px;
}

div.first {
    opacity: 0.1;
    filter: alpha(opacity=10); /* For IE8 and earlier
*/
}
```

```
div.second {  
    opacity: 0.3;  
    filter: alpha(opacity=30);}
```

```
div.third {  
    opacity: 0.6;  
    filter: alpha(opacity=60);}  
</style>
```

```
<div class="first"><p>opacity 0.1</p></div>  
<div class="second"><p>opacity 0.3</p></div>  
<div class="third"><p>opacity 0.6</p></div>  
<div><p>opacity 1 (default)</p></div>
```

```
<style>
div {
    background: rgb(76, 175, 80);
    padding: 10px;
}

div.first {
    background: rgba(76, 175, 80, 0.1);
}

div.second {
    background: rgba(76, 175, 80, 0.3);
}

div.third {
    background: rgba(76, 175, 80, 0.6);
}
</style>
```

Task 2: Create a similar **div** in your web page



```
<style>
div.background {
    background: url(klematis.jpg) repeat;
    border: 2px solid black;
}

div.transbox {
    margin: 30px;
    background-color: #ffffff;
    border: 1px solid black;
    opacity: 0.6;
    filter: alpha(opacity=60); /* For IE8 and earlier */
}

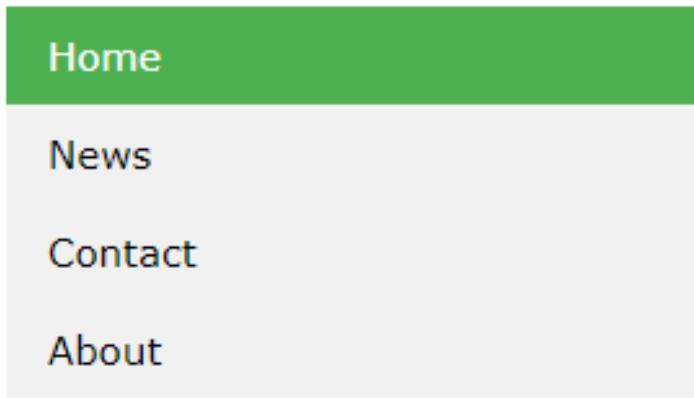
div.transbox p {
    margin: 5%;
    font-weight: bold;
    color: #000000;
}
</style>
</head>
```

```
<body>

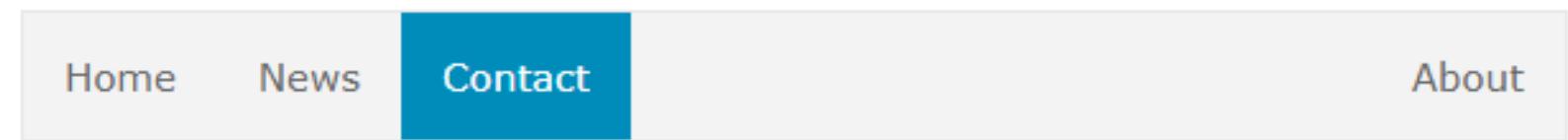
<div class="background">
    <div class="transbox">
        <p>This is some text that is placed in the transparent
        box.</p>
    </div>
</div>
```

# CSS Navigation Bar

Vertical



Horizontal



```
<ul>  
  <li><a href="default.asp">Home</a></li>  
  <li><a href="news.asp">News</a></li>  
  <li><a href="contact.asp">Contact</a></li>  
  <li><a href="about.asp">About</a></li>  
</ul>
```



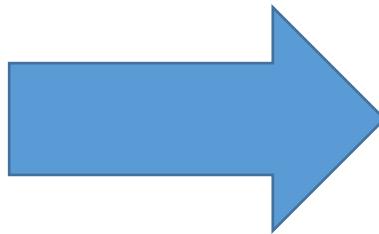
- [Home](#)
- [News](#)
- [Contact](#)
- [About](#)

```
ul {  
  list-style-type: none;  
  margin: 0;  
  padding: 0;  
}
```



[Home](#)  
[News](#)  
[Contact](#)  
[About](#)

```
<style>
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}
li a {
    display: block;
    width: 60px;
    background-color: #dddddd;
}
</style>
```



Home  
News  
Contact  
About

A screenshot of a website's header area. It features a light gray rectangular background with a thin white border. Inside, there are four blue hyperlinks arranged vertically: "Home", "News", "Contact", and "About", each with a blue underline.

```
<style>
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    width: 200px;
    background-color: #f1f1f1;
}
li a {
    display: block;
    color: #000;
    padding: 8px 16px;
    text-decoration: none;
}
li a:hover {
    background-color: #555;
    color: white;
}
</style>
```



Home  
News  
Contact  
About

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    overflow: hidden;  
    background-color: #333;  
}  
  
li {  
    float: left;  
}  
  
li a {  
    display: block;  
    color: white;  
    text-align: center;  
    padding: 14px 16px;  
    text-decoration: none;  
}  
  
li a:hover {  
    /* Change the link color to #111 (black)  
       on hover */  
    background-color: #111;  
}
```

[Home](#)   [News](#)   [Contact](#)   [About](#)

```
<!DOCTYPE html>
<html>
<head>
<style>
div.gallery {
    margin: 5px;
    border: 1px solid #ccc;
    float: left;
    width: 180px;
}
div.gallery:hover {
    border: 1px solid #777;
}
div.gallery img {
    width: 100%;
    height: auto;
}
```

```
div.desc {  
padding: 15px;  
text-align: center;  
}  
</style>  
</head>  
<body>  
  
<div class="gallery">  
 <a target="_blank" href="img_forest.jpg">  
     
 </a>  
 <div class="desc">Add a description of the  
image here</div>  
</div>  
  
<div class="gallery">  
 <a target="_blank" href="img_5terre.jpg">  
     
 </a>  
 <div class="desc">Add a description of the image  
here</div>  
</div>
```

```
<div class="gallery">
  <a target="_blank" href="img_lights.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
```

```
<div class="gallery">
  <a target="_blank" href="img_mountains.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
```

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

# Linking to a Page Section

```
<div>HTML Text Links <a name = "top"></a></div>
```

```
<a href = "/html/html_text_links.htm#top">Go to the Top</a>
```

## Download Links

```
<body>
  <a href = “handout.pdf”>Download PDF File</a>
</body>
```

## CSS Gradients

Gradients are a type of image that gradually transitions from one color to the next horizontally, vertically or diagonally

```
#grad { background-image: linear-gradient(red, yellow); }  
#grad { background-image: linear-gradient(to right, red ,  
yellow); }  
#grad { background-image: linear-gradient(to bottom right,  
red, yellow); }  
#grad { background-image: linear-gradient(-90deg, red,  
yellow); }
```

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

```
#grad { background-image:linear-gradient(to right, red,  
orange, yellow, green, blue, indigo, violet); }
```

```
#grad { background-image: linear-gradient(to right,  
rgba(255,0,0,0), rgba(255,0,0,1)); }
```

```
#grad { background-image: repeating-linear-gradient(red,  
yellow 10%, green 20%); }
```

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

```
#grad { background-image: radial-gradient(red 5%, yellow  
15%, green 60%); }
```

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

```
#grad { background-image: repeating-radial-gradient(red,  
yellow 10%, green 15%); }
```

## CSS Transitions

```
<style>
div {
    width: 100px;
    height: 100px;
    background: red;
    transition: width 2s, height 4s;
}
div:hover {
    width: 300px;
    height: 300px;
}
</style>
```

# Transition + Transformation

```
<style>
div {
    width: 100px;
    height: 100px;
    background: red;
    transition: width 2s, height 2s, transform 2s;
}
div:hover {
    width: 300px;
    height: 300px;
    transform: rotate(180deg);
}
```

# CSS Animations

```
<style>
div {
    width: 100px;
    height: 100px;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
}
@keyframes example {
    from {background-color: red;}
    to {background-color: yellow;}
}
</style>
```