**UNIT 1**

**INTERNET BASICS**:

**What is Internet?**

The Internet is essentially a global network of computing resources. You can think of the Internet as a physical collection of routers and circuits as a set of shared resources.

Some common definitions given in the past include −

* A network of networks based on the TCP/IP communications protocol.
* A community of people who use and develop those networks.

**Internet-Based Services**

Some of the basic services available to Internet users are −

* **Email** − A fast, easy, and inexpensive way to communicate with other Internet users around the world.
* **Telnet** − Allows a user to log into a remote computer as though it were a local system.
* **FTP** − Allows a user to transfer virtually every kind of file that can be stored on a computer from one Internet-connected computer to another.
* **UseNet news** − A distributed bulletin board that offers a combination news and discussion service on thousands of topics.
* **World Wide Web (WWW)** − A hypertext interface to Internet information resources.

## What is WWW?

WWW stands for **W**orld **W**ide **W**eb. A technical definition of the World Wide Web is − All the resources and users on the Internet that are using the Hypertext Transfer Protocol (HTTP).

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

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

## What is HTTP?

HTTP stands for **H**ypertext **T**ransfer **P**rotocol. This is the protocol being used to transfer hypertext documents that makes the World Wide Web possible.

A standard web address such as [Yahoo.com](http://www.yahoo.com/) is called a URL and here the prefix **http** indicates its protocol.

**What is URL?**

URL stands for **U**niform **R**esource **L**ocator, and is used to specify addresses on the World Wide Web. A URL is the fundamental network identification for any resource connected to the web (e.g., hypertext pages, images, and sound files).

A URL will have the following format −

protocol://hostname/other\_information

The protocol specifies how information is transferred from a link. The protocol used for web resources is HyperText Transfer Protocol (HTTP). Other protocols compatible with most web browsers include FTP, telnet, newsgroups, and Gopher.

The protocol is followed by a colon, two slashes, and then the domain name. The domain name is the computer on which the resource is located.

Links to particular files or subdirectories may be further specified after the domain name. The directory names are separated by single forward slashes.

## What is Website?

## Each page available on the website is called a *web page* and first page of any website is called *home page* for that site. there are millions of websites available on the web.

## What is Web Server?

Every Website sits on a computer known as a Web server. This server is always connected to the internet. Every Web server that is connected to the Internet is given a unique address made up of a series of four numbers between 0 and 256 separated by periods. For example, 68.178.157.132 or 68.122.35.127.

When you register a Web address, also known as a domain name, such as tutorialspoint.com you have to specify the IP address of the Web server that will host the site.

## What is Web Browser?

Web Browsers are software installed on your PC. To access the Web you need a web browsers, such as Netscape Navigator, Microsoft Internet Explorer or Mozilla Firefox.

Currently you must be using any sort of Web browser while you are navigating through my site tutorialspoint.com. On the Web, when you navigate through pages of information this is commonly known as *browsing or surfing*.

## What is SMTP Server?

SMTP stands for **S**imple **M**ail **T**ransfer **P**rotocol Server. This server takes care of delivering emails from one server to another server. When you send an email to an email address, it is delivered to its recipient by a SMTP Server.

## What is ISP?

ISP stands for **I**nternet **S**ervice **P**rovider. They are the companies who provide you service in terms of internet connection to connect to the internet.

You will buy space on a Web Server from any Internet Service Provider. This space will be used to host your Website.

## What is HTML?

HTML stands for **H**yper **T**ext **M**arkup **L**anguage. This is the language in which we write web pages for any Website. Even the page you are reading right now is written in HTML.

This is a subset of Standard Generalized Mark-Up Language (SGML) for electronic publishing, the specific standard used for the World Wide Web.

## What is Hyperlink?

A hyperlink or simply a link is a selectable element in an electronic document that serves as an access point to other electronic resources. Typically, you click the hyperlink to access the linked resource. Familiar hyperlinks include buttons, icons, image maps, and clickable text links.

## What is DNS?

DNS stands for **D**omain **N**ame **S**ystem. When someone types in your domain name, www.example.com, your browser will ask the Domain Name System to find the IP that hosts your site. When you register your domain name, your IP address should be put in a DNS along with your domain name. Without doing it your domain name will not be functioning properly.

## What is W3C?

W3C stands for **W**orld **W**ide **W**eb Consortium which is an international consortium of companies involved with the Internet and the Web.

The W3C was founded in 1994 by Tim Berners-Lee, the original architect of the World Wide Web. The organization's purpose is to develop open standards so that the Web evolves in a single direction rather than being splintered among competing factions. The W3C is the chief standards body for HTTP and HTML.

# Computer Concepts - Communication on Internet

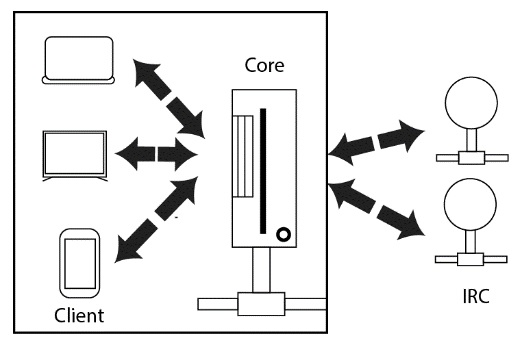
## Email

Electronic mail or email is one of the key parts of e-revolution which is specially designed for communication purpose. Once you have an email account, you can start sending electronic messages to anyone if you have the recipient’s email address. The format of an email address is "**username**, @ symbol, **domain name (yahoo.com, gmail.com, etc.**)". For example: **name@yahoo.com**.

## Internet Relay Chat (IRC)

IRC is a form of real-time internet chat or synchronous conferencing. It is mainly used for group communication in discussion forums called channels, also allows one-to-one communication via private message, and both chats and data transfers via Direct Client-to-Client protocol.

IRC client software is available for every operating system.

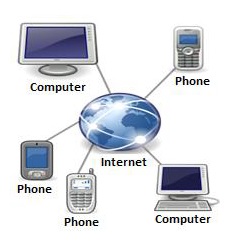


## Video Conference

A video conference is a telecommunication technology, which permits two or more people in different locations to interact via mutual video or audio transmission simultaneously. It is also called visual collaboration and is a type of groupware. Video conferencing uses telecommunication technology to bring people at different sites together for a meeting/conversation. This can be as simple as a conversation between two people in private offices, or involves several sites with more than two people. It can also be used to share documents, computer displayed information, whiteboards, etc.

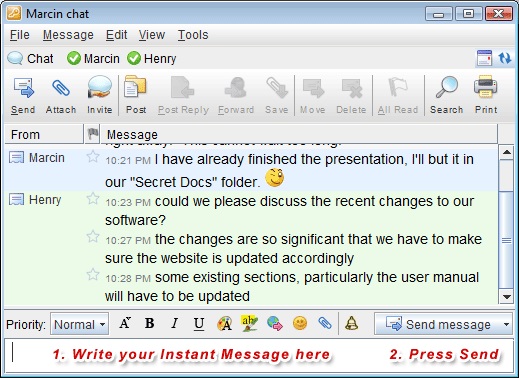
## Voice Over Telephony (VOIP)

VOIP stands for voice over IP, where IP refers to the Internet Protocol which is a base for all internet communications. This phenomenon began as an optional mutual voice extension to some of the instant messaging systems that took off around the year 2000. In recent years, VOIP systems are easy to use and as convenient as a traditional telephone. Voice quality can still differ from call to call but is often equal to and can even beat the traditional calls.



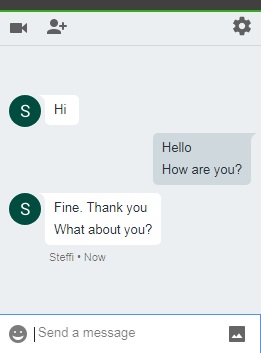
## Instant Messaging

Instant Messaging (IM) is a process of transferring real-time messages between users. It facilitates private chat room atmosphere. IM alerts users when some of their peers are online, so the users can start chatting with them.



**Chat**

Online conversations in which you are immediately able to send messages back and forth to one another is called "chat".



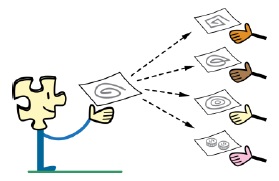
**Remote access**

Internet permits computer users to connect to other computers across the world and to store information effortlessly. This can be done with or without any security, authentication and encryption technologies depending on the requirements which encourages work from home culture.



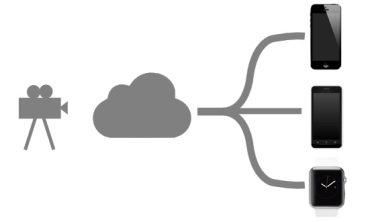
**Collaboration**

Low cost and rapid sharing of ideas, knowledge and skills has made collaborative work easy. It is a convenient way to stay in touch with colleagues through group chat or private messaging. Email is also an easy way to communicate. Collaboration helps voice and video chat between team members and to work with shared set of documents.



**File Sharing**

* A file or document can be e-mailed to anyone as an attachment.
* It can be uploaded to a website or FTP server, for users to download content easily.
* It can be put into a shared location or onto a cloud for instant use by colleagues.

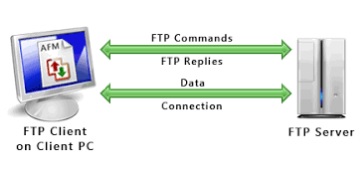


**Streaming media**

Many existing radio and television presenters provide internet "feeds" of their live audio and video streams. An internet-connected device, such as a computer or something more specific, can be used to access on-line media in much the same way, as television or radio receiver.

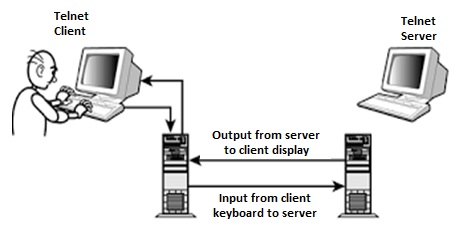
**File Transfer Protocol**

File Transfer Protocol is a network protocol which is used to transfer data from one computer to another over any TCP/IP based network. It helps to manipulate files on another computer regardless of the operating system involved.



**Telnet**

TELNET means TELecommunication NETwork. It is a network protocol used on the internet or LAN connections. It helps to provide text oriented mutual communication using virtual terminal connection.



# Domain name

A **domain name** is a website's address on the [Internet](https://developer.mozilla.org/en-US/docs/Glossary/Internet). Domain names are used in [URLs](https://developer.mozilla.org/en-US/docs/Glossary/URL) to identify which server a specific webpage belongs to. The domain name consists of a hierarchical sequence of names (labels) separated by periods (dots) and ending with an [extension](https://developer.mozilla.org/en-US/docs/Glossary/TLD).

# What is an Internet domain?

The Internet domain name1 is the basic element of the addressing structure of the Internet.

The Internet is a huge network of hundreds of millions of connected computers all over the world. In order for them all to be able to communicate with one another, they need to know which device is located where on the Internet. Each device on the Internet has its own IP address, defined in accordance with the Internet protocol, and each such address is unique at the global level. These addresses are complicated numbers, very difficult to remember, and so Internet domain names were introduced. These are paired up with IP addresses and are much simpler for humans to use. These names are also unique at the global level and comprise the Domain Name System (DNS), which is the Internet’s addressing system.

An Internet domain name consists of at least two segments separated by a dot (e.g. **domen.rs**). There are also names with three segments (e.g. **domen.co.rs**), where the dot and last segment are the domain extension, in this case our national .RS domain designation.

A segment can contain numbers (0-9), letters from various alphabets and hyphens (-), and the length of a segment cannot be shorter than two or longer than 63 characters. A segment cannot contain a hyphen at the start or the end, nor two consecutive hyphens as the third and fourth characters.

When a domain name is registered, it is entered into a special data base and is supported by an Internet infrastructure which enables it to function on the Internet. This database and infrastructure are managed by an Internet domain registry. The national registry for the .RS and .СРБ domains is the Serbian National Internet Domain Name Registry Foundation (RNIDS).

Originally, domain names only supported the letters of the English alphabet (ASCII code), but with the development of the IDN standard it became possible for domain names to use the scripts of other languages of the world. These scripts include Cyrillic, Greek, Chinese, Arabic and Indian, as well as Latin scripts containing letters which do not exist in the English alphabet (such as đ, š, ž, ä, ß, ů, to name a few).

Your Internet domain name defines both the web address of your website and your email address, and that is why it is an important part of your Internet identity or that of your company. Choosing and registering a distinctive Internet domain name are the first steps in creating an Internet identity and developing an online business.

**Web server**

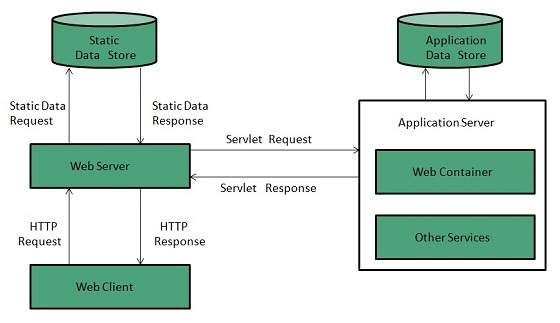
**Web server** is a computer where the web content is stored. Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.

Web site is collection of web pages whileweb server is a software that respond to the request for web resources.

**Web Server Working**

Web server respond to the client request in either of the following two ways:

* Sending the file to the client associated with the requested URL.
* Generating response by invoking a script and communicating with database



**Key Points**

* When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response.
* If the requested web page is not found, web server will the send an **HTTP response:Error 404 Not found.**
* If client has requested for some other resources then the web server will contact to the application server and data store to construct the HTTP response.

**HTML Introduction:**

HTML is the standard markup language for creating Web pages.

## What is HTML?

* HTML stands for Hyper Text Markup Language
* HTML is the standard markup language for creating Web pages
* HTML describes the structure of a Web page
* HTML consists of a series of elements
* HTML elements tell the browser how to display the content
* HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

## A Simple HTML Document

### Example

<!DOCTYPE html>  
<html>  
<head>  
<title>Page Title</title>  
</head>  
<body>  
  
<h1>My First Heading</h1>  
<p>My first paragraph.</p>  
  
</body>  
</html>

# HTML Tags

HTML tags are like keywords which defines that how web browser will format and display the content. With the help of tags, a web browser can distinguish between an HTML content and a simple content. HTML tags contain three main parts: opening tag, content and closing tag. But some HTML tags are unclosed tags.

When a web browser reads an HTML document, browser reads it from top to bottom and left to right. HTML tags are used to create HTML documents and render their properties. Each HTML tags have different properties.

An HTML file must have some essential tags so that web browser can differentiate between a simple text and HTML text. You can use as many tags you want as per your code requirement.

* All HTML tags must enclosed within < > these brackets.
* Every tag in HTML perform different tasks.
* If you have used an open tag <tag>, then you must use a close tag </tag> (except some tags)

## Syntax

<tag> content </tag>

## Types of tags in HTML- HTML tags list download

The types of tags in HTML are categorized on the basis of their appearance. Some tags comes in pairs and others are single. You can also [download HTML tags list pdf](https://www.coderepublics.com/HTML/images/HTML%20Tags%20List.pdf) or can look at the table at the page end.

**There are two types of tags in HTML that are used by the**[**Website Designers**](https://thewebseeker.com/full-stack-developer/)**:**

1. Paired Tags (Opening and Closing Tags)
2. Unpaired Tags (Singular Tag)

### Paired Tags - Opening and Closing Tags

Paired tags are a set of two tags with the same name. In each Paired tag set, one is an opening tag, and the other one is the closing tag. The closing tag has a / slash, it means that the tag is closed now.

It is necessary to close a paired tag; otherwise, it can result in the malfunctioning of the website. When the content is written within paired tags, then it ensures that the effect of those tags would be limited to only the content between them.

Look at the list of some paired tags in [HTML](https://www.coderepublics.com/HTML/html-tutorial.php) below. Notice that each tag has a closing tag with a slash(/) before the name of the tag.

Syntax: <tag> Content </tag>

### List of some paired tags in HTML:

| **Open Tag** | **Close Tag** |
| --- | --- |
| <html> | </html> |
| <table> | </table> |
| <form> | </form> |
| <span> | </span> |
| <ul> | </ul> |
| <p> | </p> |
| <head> | </head> |
| <div> | </div> |

### Unpaired Tags - Singular Tags

Unpaired tags are single tags with no closing tag. These tags are also called **Singular Tags**. These are also called **non-container tags** because they do not contain any content.

It is recommended to close the unpaired/singular tags also. But unfortunately, we do not have the closing tag for those. So, an unpaired tag is closed after adding a slash(/) just before the greater than > sign. For example: <br />.

Look below the list of some Unpaired Tags in HTML. Notice the use of slash(/) in the tags, to close them.

#### Some Unpaired Tags are:

| **Open Tag** |
| --- |
| <br> |
| <hr> |
| <meta> |
| <input> |

## HTML Heading Tags - H1 tag to H6 tag

Heading tag is used to give headings of different sizes in a document. There are six different HTML heading tags, which gives different heading sizes and are defined by <h1> to <h6> tags. <h1> gives the largest heading and <h6> gives the smallest one. So <h1> can be used for most important headings and <h6> can be used for a least important one.

Example:

<!DOCTYPE html>   
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<title> HTML Heading Tag </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**

## HTML p tag - Paragraph tag

The <p> tag is used to define a paragraph in a document. [HTML paragraph](https://www.coderepublics.com/HTML/html-paragraph-tag.php) or HTML <p> tag gives the text inside it, a paragraph like finishing. It is a notable point that a browser itself add a line break before and after a paragraph.

Let's take a simple example to see how it works.

Example

<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<title> HTML Paragraph Tag </title>  
</head>   
<body>   
<p> This is First Paragraph </p>  
<p> This is Second Paragraph </p>   
<p> This is Third Paragraph </p>   
</body>   
</html>

#### Output

This is First Paragraph

This is Second Paragraph

This is Third Paragraph

## HTML a tag - Anchor Tag

HTML Hyperlink is defined with the <a> tag (Anchor tag). It is used to give a link to any file, webpage, image etc.

This tag is called [anchor tag](https://www.coderepublics.com/HTML/html-anchor-tag.php) and anything between the opening <a> tag and the closing </a> tag becomes part of the link, and a user can click that part to reach to the linked document.

Following is the simple syntax to use <a> tag.

Example:

<!DOCTYPE html>   
<HTML lang="en">   
<head>   
<meta charset="UTF-8">  
<title> HTML Anchor Tag </title>   
</head>   
<body>   
<a target="\_blank" href="https://www.coderepublics.com"> This is a link </a>   
</body>   
</html>

#### Output

[This is a link](https://www.coderepublics.com/)

**Note :**Use 'target = \_blank' as an attribute in <a> tag to open the link in a new tab.

### HTML img tag - Image Tag

The Image Tag is used to add Images in HTML documents. The [HTML img tag](https://www.coderepublics.com/HTML/html-image-tags.php) is used to add image in a document. The 'src' attribute is used to give source(address) of the image. The height and width of the image can be controlled by the attributes - height="px" and width="px".

The alt attribute is used as an alternative in a case if the image is not shown. Anything written as a value of this attribute will be displayed. It will give information about the image.

Example

<!DOCTYPE html>   
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<title> HTML Image Tag </title>  
</head>   
<body>   
<img src="HTML-Image.png" width="400px" height="200px">  
</body>   
</html>

#### Output



Commonly used tags:

|  |  |
| --- | --- |
| [<h1> to <h6>](https://www.coderepublics.com/HTML/html-heading-tags.php) | It defines headings' sizes from level 1 to 6 |
| [<head>](https://www.coderepublics.com/HTML/html-tags.php) | It defines the head section of an HTML document |
| [<html>](https://www.coderepublics.com/HTML/html-tags.php) | It is the root element of HTML document. |
| [<form>](https://www.coderepublics.com/HTML/html-forms.php) | It defines a form. |
| <[footer>](https://www.coderepublics.com/HTML/html-footer-tag.php) | It defines footer of a webpage |
| [<em>](https://www.coderepublics.com/HTML/html-tags.php) | It emphasizes text content |
| [<div>](https://www.coderepublics.com/HTML/html-tags.php) | It defines a division within HTML document. |
| [<dl>](https://www.coderepublics.com/HTML/html-description-list.php) | It defines a description list. |
| [<dd>](https://www.coderepublics.com/HTML/html-description-list.php) | It is used to provide definition/description of a term in description list. |
| [<del>](https://www.coderepublics.com/HTML/html-tags.php) | It defines a deleted text. |
| [<body>](https://www.coderepublics.com/HTML/html-tags.php) | It defines the body section of an HTML document. |
| [<br>](https://www.coderepublics.com/HTML/html-br-tag.php) | It creates a single line break |
| [<a>](https://www.coderepublics.com/HTML/html-anchor-tag.php) | Creates hyperlink |

**Title and Footer tag**

Title tag:

Definition and Usage

The <title> tag defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.

The <title> tag is required in HTML documents!

The contents of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.

The <title> element:

* defines a title in the browser toolbar
* provides a title for the page when it is added to favorites
* displays a title for the page in search-engine results

Here are some tips for creating good titles:

* Go for a longer, descriptive title (avoid one- or two-word titles)
* Search engines will display about 50-60 characters of the title, so try not to have titles longer than that
* Do not use just a list of words as the title (this may reduce the page's position in search results)

### Example

Define a title for your HTML document:

<!DOCTYPE html>  
<html>  
<head>  
  <title>HTML Elements Reference</title>  
</head>  
<body>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>

**Footer Tag**

Definition and Usage

The <footer> tag defines a footer for a document or section.

A <footer> element typically contains:

* authorship information
* copyright information
* contact information
* sitemap
* back to top links
* related documents

You can have several <footer> elements in one document.

Example:

<!DOCTYPE html>

<html>

<body>

<h1>The footer element</h1>

<footer>

<p>Author: Hege Refsnes<br>

<a href="mailto:hege@example.com">hege@example.com</a></p>

</footer>

</body>

</html>

**HTML LIST:**

HTML lists allow web developers to group a set of related items in lists.

### Example

An unordered HTML list:

* Item
* Item
* Item
* Item

An ordered HTML list:

1. First item
2. Second item
3. Third item
4. Fourth item

**Types of Lists:**

## 1.)Unordered HTML List

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

The list items will be marked with bullets (small black circles) by default:

### Example

<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>

## 2.)Ordered HTML List

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

The list items will be marked with numbers by default:

### Example

<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>

## 3.)HTML Description Lists

HTML also supports description lists.

A description list is a list of terms, with a description of each term.

The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

### Example

<dl>  
  <dt>Coffee</dt>  
  <dd>- black hot drink</dd>  
  <dt>Milk</dt>  
  <dd>- white cold drink</dd>  
</dl>

**Linking Documents**

A link is specified using HTML tag <a>. This tag is called **anchor tag** and anything between the opening <a> tag and the closing </a> tag becomes part of the link and a user can click that part to reach to the linked document. Following is the simple syntax to use <a> tag.

<a href = "Document URL" ... attributes-list>Link Text</a>

Example:

<!DOCTYPE html>

<html>

<head>

<title>Hyperlink Example</title>

</head>

<body>

<p>Click following link</p>

<a href = "https://www.tutorialspoint.com" target = "\_self">Tutorials Point</a>

</body>

</html>

OUTPUT:

Click following link

[Tutorials Point](https://www.tutorialspoint.com/)

## What are external links?

External Links are hyperlinks that point at (target) any domain other than the domain the link exists on (source). In layman's terms, if another website links to you, this is considered an external link to your site. Similarly, if you link out to another website, this is also considered an external link.

##### **Code Sample**

<a href="http://www.external-domain.com/">Link Anchor Text</a>

**HTML FRAMES:**

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

**Disadvantages of Frames**

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages −

* Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
* Sometimes your page will be displayed differently on different computers due to different screen resolution.
* The browser's *back* button might not work as the user hopes.
* There are still few browsers that do not support frame technology.

**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. Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

**Note** − The <frame> tag deprecated in HTML5. Do not use this element.

### Example

Following is the example to create three horizontal frames −

<!DOCTYPE html>

<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>

The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag −

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **cols**  Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways −  Absolute values in pixels. For example, to create three vertical frames, use *cols = "100, 500, 100"*.  A percentage of the browser window. For example, to create three vertical frames, use *cols = "10%, 80%, 10%"*.  Using a wildcard symbol. For example, to create three vertical frames, use *cols = "10%, \*, 10%"*. In this case wildcard takes remainder of the window.  As relative widths of the browser window. For example, to create three vertical frames, use *cols = "3\*, 2\*, 1\*"*. This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth. |
| 2 | **rows**  This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use *rows = "10%, 90%"*. You can specify the height of each row in the same way as explained above for columns. |
| 3 | **border**  This attribute specifies the width of the border of each frame in pixels. For example, border = "5". A value of zero means no border. |
| 4 | **frameborder**  This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder = "0" specifies no border. |
| 5 | **framespacing**  This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing = "10" means there should be 10 pixels spacing between each frames. |

The <frame> Tag Attributes

Following are the important attributes of <frame> tag −

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **src**  This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top\_frame.htm" will load an HTML file available in html directory. |
| 2 | **name**  This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link. |
| 3 | frameborder  This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no). |
| 4 | marginwidth  This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth = "10". |
| 5 | marginheight  This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight = "10". |
| 6 | noresize  By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize = "noresize". |
| 7 | scrolling  This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars. |
| 8 | longdesc  This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc = "framedescription.htm" |

# HTML frameset Tag

The <frameset> tag in HTML is used to define the frameset. The <frameset> element contains one or more frame elements. It is used to specify the number of rows and columns in frameset with their pixel of spaces. Each element can hold a separate document.

**Note:** The <frameset> tag is not supported in HTML5.

**Syntax:**

<frameset cols = "pixels|%|\*">

**Attributes:** The list of frameset attributes are given below:

* [**cols**](https://www.geeksforgeeks.org/html-cols-attribute/)**:** The cols attribute is used to create vertical frames in a web browser. This attribute is basically used to define the no. of columns and their size inside the frameset tag.
* [**rows**](https://www.geeksforgeeks.org/html-rows-attribute/)**:** The rows attribute is used to create horizontal frames in the web browser. This attribute is used to define the no. of rows and their size inside the frameset tag.
* [**border**](https://www.geeksforgeeks.org/html-border-attribute/)**:** This attribute of frameset tag defines the width of the border of each frame in pixels. Zero value is used for no border.
* [**frameborder**](https://www.geeksforgeeks.org/html-frame-frameborder-attribute/)**:** This attribute of frameset tag is used to specify whether a three-dimensional border should be displayed between the frames or not for this use two values 0 and 1, where 0 defines no border and value 1 signifies for yes there will be a border.
* [**framespacing**](https://www.geeksforgeeks.org/html5-mathml-framespacing-attribute/)**:** This attribute of frameset tag is used to specify the amount of spacing between the frames in a frameset. This can take any integer value as a parameter which basically denotes the value in pixel.
* <!DOCTYPE html>
* <html>
* <head>
* <title>frameset attribute</title>
* </head>
* <!-- frameset attribute starts here -->
* <frameset rows = "20%, 60%, 20%">
* <frame name = "top" src = "attr1.png" />
* <frame name = "main" src = "gradient3.png" />
* <frame name = "bottom" src = "col\_last.png" />
* <noframes>
* <body>The browser you are working does not
* support frames.</body>
* </noframes>
* </frameset>
* <!-- frameset attribute ends here -->
* </html>
* **Output:**   
  The above example basically used to create three horizontal frames: top, middle, and bottom using row attribute of frameset tag, and the noframe tag is used for that browser that doesn’t support noframe.



**Dynamic HTML**

**Dynamic HTML**, or **DHTML**, is a term which was used by some browser vendors to describe the combination of [HTML](https://en.wikipedia.org/wiki/HTML), [style sheets](https://en.wikipedia.org/wiki/Style_sheet_(web_development)) and [client-side scripts](https://en.wikipedia.org/wiki/Dynamic_web_page#Client-side_scripting) (JavaScript, VBScript, or any other supported scripts) that enabled the creation of interactive and animated documents.[[1]](https://en.wikipedia.org/wiki/Dynamic_HTML#cite_note-1)[[2]](https://en.wikipedia.org/wiki/Dynamic_HTML#cite_note-2) The application of DHTML was introduced by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) with the release of [Internet Explorer 4](https://en.wikipedia.org/wiki/Internet_Explorer_4) in 1997.

DHTML allows scripting languages to change [variables](https://en.wikipedia.org/wiki/Variable_(programming)) in a web page's definition language, which in turn affects the look and function of otherwise "static" HTML page content after the page has been fully loaded and during the viewing process. Thus the dynamic characteristic of DHTML is the way it functions while a page is viewed, not in its ability to generate a unique page with each page load.

**Uses**

DHTML allows authors to add effects to their pages that are otherwise difficult to achieve, by changing the [Document Object Model](https://en.wikipedia.org/wiki/Document_Object_Model) (DOM) and page style. The combination of HTML, CSS, and JavaScript offers ways to:

* Animate text and images in their document.
* Embed a ticker or other dynamic display that automatically refreshes its content with the latest news, stock quotes, or other data.
* Use a form to capture user input, and then process, verify and respond to that data without having to send data back to the server.
* Include rollover buttons or drop-down menus.

**Document Object Model**

DHTML is not a technology in and of itself; rather, it is the product of three related and complementary technologies: HTML, Cascading Style Sheets (CSS), and [JavaScript](https://en.wikipedia.org/wiki/JavaScript). To allow scripts and components to access features of HTML and CSS, the contents of the document are represented as objects in a programming model known as the [Document Object Model](https://en.wikipedia.org/wiki/Document_Object_Model) (DOM).

The DOM API is the foundation of DHTML, providing a structured interface that allows access and manipulation of virtually anything in the document. The HTML elements in the document are available as a hierarchical [tree](https://en.wikipedia.org/wiki/Tree_(data_structure)) of individual objects, making it possible to examine and modify an element and its attributes by reading and setting properties and by calling methods. The text between elements is also available through DOM properties and methods.

**Dynamic styles**

Dynamic styles are a key feature of DHTML. By using CSS, one can quickly change the appearance and formatting of elements in a document without adding or removing elements. This helps keep documents small and the scripts that manipulate the document fast.

The object model provides programmatic access to styles. This means you can change inline styles on individual elements and change style rules using simple JavaScript programming.

Inline styles are CSS style assignments that have been applied to an element using the style attribute. You can examine and set these styles by retrieving the style object for an individual element. For example, to highlight the text in a heading when the user moves the mouse pointer over it, you can use the style object to enlarge the font and change its color,

**CSS:**

What is CSS?

* CSS stands for Cascading Style Sheets
* CSS describes how HTML elements are to be displayed on screen, paper, or in other media
* CSS saves a lot of work. It can control the layout of multiple web pages all at once
* External stylesheets are stored in CSS files
* Why Use CSS?
* CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

Example of css:

body {  
  background-color: lightblue;  
}  
  
h1 {  
  color: white;  
  text-align: center;  
}  
  
p {  
  font-family: verdana;  
  font-size: 20px;  
}

# HTML Forms

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

### Example

Top of Form

First name:  
  
Last name:  
  
  


Bottom of Form

The <input> Element

The HTML <input> element is the most used form element.

An <input> element can be displayed in many ways, depending on the type attribute.Here are some examples:

|  |  |
| --- | --- |
| **Type** | **Description** |
| <input type="text"> | Displays a single-line text input field |
| <input type="radio"> | Displays a radio button (for selecting one of many choices) |
| <input type="checkbox"> | Displays a checkbox (for selecting zero or more of many choices) |
| <input type="submit"> | Displays a submit button (for submitting the form) |
| <input type="button"> | Displays a clickable button |

Text Fields

The <input type="text"> defines a single-line input field for text input.

### Example

A form with input fields for text:

<form>  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname"><br>  
  <label for="lname">Last name:</label><br>  
  <input type="text" id="lname" name="lname">  
</form>

This is how the HTML code above will be displayed in a browser:

Top of Form

First name:  
  
Last name:  


Bottom of Form

The <label> Element

Notice the use of the <label> element in the example above.

The <label> tag defines a label for many form elements.

The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focuses on the input element.

The <label> element also helps users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.

The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

Radio Buttons

The <input type="radio"> defines a radio button.

Radio buttons let a user select ONE of a limited number of choices.

### Example

A form with radio buttons:

<p>Choose your favorite Web language:</p>  
  
<form>  
  <input type="radio" id="html" name="fav\_language" value="HTML">  
  <label for="html">HTML</label><br>  
  <input type="radio" id="css" name="fav\_language" value="CSS">  
  <label for="css">CSS</label><br>  
  <input type="radio" id="javascript" name="fav\_language" value="JavaScript">  
  <label for="javascript">JavaScript</label>  
</form>

This is how the HTML code above will be displayed in a browser:

Choose your favorite Web language:

 HTML  
 CSS  
 JavaScript

Checkboxes

The <input type="checkbox"> defines a **checkbox**.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

### Example

A form with checkboxes:

<form>  
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">  
  <label for="vehicle1"> I have a bike</label><br>  
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">  
  <label for="vehicle2"> I have a car</label><br>  
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">  
  <label for="vehicle3"> I have a boat</label>  
</form>

This is how the HTML code above will be displayed in a browser:

 I have a bike  
 I have a car  
 I have a boat

The Submit Button

The <input type="submit"> defines a button for submitting the form data to a form-handler.

The form-handler is typically a file on the server with a script for processing input data.

The form-handler is specified in the form's action attribute.

### Example

A form with a submit button:

<form action="/action\_page.php">  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname" value="John"><br>  
  <label for="lname">Last name:</label><br>  
  <input type="text" id="lname" name="lname" value="Doe"><br><br>  
  <input type="submit" value="Submit">  
</form>

This is how the HTML code above will be displayed in a browser:

Top of Form

First name:  
  
Last name:  
  
  


Bottom of Form

The Name Attribute for <input>

Notice that each input field must have a name attribute to be submitted.

If the name attribute is omitted, the value of the input field will not be sent at all.

### Example

This example will not submit the value of the "First name" input field:

<form action="/action\_page.php">  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" value="John"><br><br>  
  <input type="submit" value="Submit">  
</form>

**JavaScript**

**Introduction To javaScript:**

**JavaScript** is a lightweight, cross-platform, and interpreted compiled programming language which is also known as the scripting language for webpages. It is well-known for the development of web pages, many non-browser environments also use it. JavaScript can be used for [**Client-side**](https://www.geeksforgeeks.org/server-side-client-side-programming/) developments as well as [**Server-side**](https://www.geeksforgeeks.org/server-side-client-side-programming/) developments. Javascript is both imperative and declarative type of language. JavaScript contains a standard library of objects, like [**Array**](https://www.geeksforgeeks.org/arrays-in-javascript/), [**Date**](https://www.geeksforgeeks.org/javascript-date-objects/), and [**Math**](https://www.geeksforgeeks.org/javascript-math-object/), and a core set of language elements like [**operators**](https://www.geeksforgeeks.org/javascript-operators/), **control structures**, and [**statements**](https://www.geeksforgeeks.org/javascript-statements/).

**Keypoints regarding JavaScript:**

* **Client-side:** It supplies objects to control a browser and its [Document Object Model (DOM).](https://www.geeksforgeeks.org/dom-document-object-model/) Like if client-side extensions allow an application to place elements on an HTML form and respond to user events such as **mouse clicks**, **form input**, and **page navigation**. Useful libraries for the client-side are **[AngularJS](https://www.geeksforgeeks.org/introduction-to-angularjs/)**, **[ReactJS](https://www.geeksforgeeks.org/react-js-introduction-working/)**, **[VueJS](https://www.geeksforgeeks.org/vue-js/)** and so many others.
* **Server-side:** It supplies objects relevant to running JavaScript on a server. Like if the server-side extensions allow an application to communicate with a database, and provide continuity of information from one invocation to another of the application, or perform file manipulations on a server. The useful framework which is the most famous these days is [**node.js**](https://www.geeksforgeeks.org/introduction-to-nodejs/).
* **Imperative language –**In this type of language we are mostly concern about how it is to be done . It simply control the flow of computation . The procedural programming approach , object, oriented approach comes under this like async await we are thinking what it is to be done further after async call.
* **Declarative programming –**In this type of language we are concern about how it is to be done , basically here logical computation require . Here  main goal is to describe the desired result without direct dictation on how to get it like  arrow function do .

**JavaScript can be added to your HTML file in**[**two ways**](https://www.geeksforgeeks.org/where-to-put-javascript-in-an-html-document/):

* **Internal JS:** We can add JavaScript directly to our HTML file by writing the code inside the <script> tag. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.
* [**External JS**](https://www.geeksforgeeks.org/what-is-external-javascript/)**:** We can write JavaScript code in other file having an extension.js and then link this file inside the <head> tag of the HTML file in which we want to add this code.

**Syntax:**

<script>

// JavaScript Code

</script>

**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>

        Basic Example to Describe JavaScript

    </title>

</head>

<body>

    <!-- JavaScript code can be embedded inside

        head section or body section -->

    <script>

        console.log("Welcome to GeeksforGeeks");

    </script>

</body>

</html>

**Output:** The output will display on the console.

Welcome to GeeksforGeeks

**History of JavaScript:** It was created in 1995 by Brendan Eich while he was an engineer at Netscape. It was originally going to be named LiveScript but was renamed. Unlike most programming languages, the JavaScript language has no concept of input or output. It is designed to run as a scripting language in a host environment, and it is up to the host environment to provide mechanisms for communicating with the outside world. The most common host environment is the browser.

**Features of JavaScript:** According to a recent survey conducted by **Stack Overflow**, JavaScript is the most popular language on earth.

With advances in browser technology and JavaScript having moved into the server with Node.js and other frameworks, JavaScript is capable of so much more. Here are a few things that we can do with JavaScript:

* JavaScript was created in the first place for DOM manipulation. Earlier websites were mostly static, after JS was created dynamic Web sites were made.
* Functions in JS are objects. They may have properties and methods just like another object. They can be passed as arguments in other functions.
* Can handle date and time.
* Performs Form Validation although the forms are created using HTML.
* No compiler is needed.

**Applications of JavaScript:**

* **Web Development:** Adding interactivity and behavior to static sites JavaScript was invented to do this in 1995. By using AngularJS that can be achieved so easily.
* **Web Applications:** With technology, browsers have improved to the extent that a language was required to create robust web applications. When we explore a map in Google Maps then we only need to click and drag the mouse. All detailed view is just a click away, and this is possible only because of JavaScript. It uses Application Programming Interfaces(APIs) that provide extra power to the code. The Electron and React is helpful in this department.
* **Server Applications:** With the help of Node.js, JavaScript made its way from client to server and node.js is the most powerful on the server-side.
* **Games:** Not only in websites, but JavaScript also helps in creating games for leisure. The combination of JavaScript and HTML 5 makes JavaScript popular in game development as well. It provides the EaseJS library which provides solutions for working with rich graphics.
* **Smartwatches:** JavaScript is being used in all possible devices and applications. It provides a library PebbleJS which is used in smartwatch applications. This framework works for applications that require the internet for its functioning.
* **Art:** Artists and designers can create whatever they want using JavaScript to draw on HTML 5 canvas, and make the sound more effective also can be used [**p5.js**](https://www.geeksforgeeks.org/p5-js-introduction/) library.
* **Machine Learning:** This JavaScript ml5.js library can be used in web development by using machine learning.
* **Mobile Applications:**JavaScript can also be used to build an application for non-web contexts. The features and uses of JavaScript make it a powerful tool for creating mobile applications. This is a Framework for building web and mobile apps using JavaScript. Using React Native, we can build mobile applications for different operating systems. We do not require to write code for different systems. Write once use it anywhere!

**Limitations of JavaScript:**

* **Security risks:** JavaScript can be used to fetch data using AJAX or by manipulating tags that load data such as <img>, <object>, <script>. These attacks are called cross site script attacks. They inject JS that is not the part of the site into the visitor’s browser thus fetching the details.
* **Performance:**JavaScript does not provide the same level of performance as offered by many traditional languages as a complex program written in JavaScript would be comparatively slow. But as JavaScript is used to perform simple tasks in a browser, so performance is not considered a big restriction in its use.
* **Complexity:**To master a scripting language, programmers must have a thorough knowledge of all the programming concepts, core language objects, client and server-side objects otherwise it would be difficult for them to write advanced scripts using JavaScript.
* **Weak error handling and type checking facilities:**It is weakly typed language as there is no need to specify the data type of the variable. So wrong type checking is not performed by compile.

**Why JavaScript is known as a lightweight programming language?**

JavaScript is considered as lightweight due to the fact that it has low CPU usage, is easy to implement and has a minimalist syntax. Minimalist syntax as in, it has no data types. Everything is treated here as an object. It is very easy to learn because of its syntax similar to C++ and Java.

A lightweight language does not consume much of your CPU’s resources. It doesn’t put excess strain on your CPU or RAM. JavaScript runs in the browser even though it has complex paradigms and logic which means it uses fewer resources than other languages. For example, NodeJs, a variation of JavaScript not only performs faster computations but also uses less resources than its counterparts such as Dart or Java.

Additionally, when compared with other programming languages, it has less in-built libraries or frameworks, contributing as another reason for it to be lightweight. However, this brings it a drawback that we need to incorporate external libraries and frameworks.

**Is JavaScript compiled or interpreted or both?**

JavaScript is both compiled and interpreted. In the earlier versions of JavaScript, it used only the interpreter that executed code line by line and shows the result immediately. But with time the performance became an issue as interpretation is quite slow. Therefore, in the newer versions of JS, probably after the V8, JIT compiler was also incorporated to optimize the execution and display the result more quickly. This JIT compiler generates a bytecode that is relatively easier to code. This bytecode is a set of highly optimized instructions.

The V8 engine initially uses an interpreter, to interpret the code. On further executions, the V8 engine finds patterns such as frequently executed functions, frequently used variables, and compiles them to improve performance.

# JavaScript Output

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

* Writing into an HTML element, using innerHTML.
* Writing into the HTML output using document.write().
* Writing into an alert box, using window.alert().
* Writing into the browser console, using console.log().
* Using innerHTML
* To access an HTML element, JavaScript can use the document.getElementById(id) method.
* The id attribute defines the HTML element. The innerHTML property defines the HTML content:

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My First Paragraph</p>  
  
<p id="demo"></p>  
  
<script>  
document.getElementById("demo").innerHTML = 5 + 6;  
</script>  
  
</body>  
</html>

**Output:**

**My First Web Page**

My First Paragraph.

11

**Using document.write()**

For testing purposes, it is convenient to use document.write():

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My first paragraph.</p>  
  
<script>  
document.write(5 + 6);  
</script>  
</body>  
</html>

**OUTPUT:**

**My First Web Page**

My first paragraph.

Never call document.write after the document has finished loading. It will overwrite the whole document.

11

# JavaScript Syntax

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

// How to create variables:  
var x;  
let y;  
  
// How to use variables:  
x = 5;  
y = 6;  
let z = x + y;

JavaScript Values

The JavaScript syntax defines two types of values:

* Fixed values
* Variable values

Fixed values are called **Literals**.

Variable values are called **Variables**.

**JavaScript Literals**

The two most important syntax rules for fixed values are:

1. **Numbers** are written with or without decimals:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Numbers</h2>

<p>Number can be written with or without decimals.</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = 10.50;

</script>

</body>

</html>

**OUTPUT:**

JavaScript Numbers

Number can be written with or without decimals.

10.5

1. **Strings** are text, written within double or single quotes:

Eg:"John Doe"  
  
'John Doe'

**JavaScript Variables**

In a programming language, **variables** are used to **store** data values.

JavaScript uses the keywords var, let and const to **declare** variables.

An **equal sign** is used to **assign values** to variables.

In this example, x is defined as a variable. Then, x is assigned (given) the value 6:

EG: let x;  
x = 6;

JavaScript Operators

JavaScript uses **arithmetic operators** ( + - \* / ) to **compute** values:

EG:(5 + 6) \* 10

JavaScript uses an **assignment operator** ( = ) to **assign** values to variables:

Eg:let x, y;  
x = 5;  
y = 6;

**JavaScript Expressions**

An expression is a combination of values, variables, and operators, which computes to a value.

The computation is called an evaluation.

For example, 5 \* 10 evaluates to 50:

Eg: 5 \* 10

Expressions can also contain variable values:

Eg: x \* 10

The values can be of various types, such as numbers and strings.

For example, "John" + " " + "Doe", evaluates to "John Doe":

"John" + " " + "Doe"

JavaScript Keywords

JavaScript **keywords** are used to identify actions to be performed.

The let keyword tells the browser to create variables:

Eg: let x, y;  
x = 5 + 6;  
y = x \* 10;

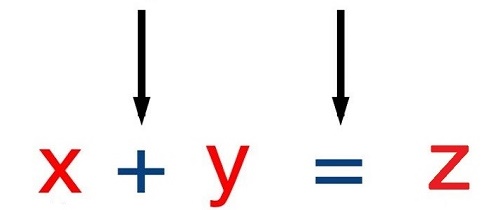
The var keyword also tells the browser to create variables:

Eg :var x, y;  
x = 5 + 6;  
y = x \* 10;

# JavaScript Operators

The **Addition Operator** **+** adds numbers:

The **Assignment Operator** **=** assigns a value to a variable.



JavaScript Assignment

The **Assignment Operator** (=) assigns a value to a variable:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Operators</h1>

<h2>The = Operator</h2>

<p id="demo"></p>

<script>

let x = 10;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

Output:

JavaScript Operators

## The = Operator

10

// Assign the value 5 to x  
let x = 5;  
// Assign the value 2 to y  
let y = 2;  
// Assign the value x + y to z:  
let z = x + y;

Types of JavaScript Operators

There are different types of JavaScript operators:

* Arithmetic Operators
* Assignment Operators
* Comparison Operators
* String Operators
* Logical Operators
* Bitwise Operators
* Ternary Operators
* Type Operators

1.)JavaScript Arithmetic Operators:

* **Arithmetic Operators** are used to perform arithmetic on numbers:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Arithmetic</h1>

<h2>Arithmetic Operations</h2>

<p>A typical arithmetic operation takes two numbers (or expressions) and produces a new number.</p>

<p id="demo"></p>

<script>

let a = 3;

let x = (100 + 50) \* a;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

Output:

# JavaScript Arithmetic

## Arithmetic Operations

A typical arithmetic operation takes two numbers (or expressions) and produces a new number.

450

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| \*\* | Exponentiation ([ES2016](https://www.w3schools.com/js/js_2016.asp)) |
| / | Division |
| % | Modulus (Division Remainder) |
|  |  |
| ++ | Increment |
| -- | Decrement |

2.)JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

The **Addition Assignment Operator** (+=) adds a value to a variable.

**<!DOCTYPE html>**

**<html>**

**<body>**

**<h1>JavaScript Arithmetic</h1>**

**<h2>The += Operator</h2>**

**<p id="demo"></p>**

**<script>**

**var x = 10;**

**x += 5;**

**document.getElementById("demo").innerHTML = x;**

**</script>**

**</body>**

**</html>**

**Output:**

# JavaScript Arithmetic

## The += Operator

15

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Same As** |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |
| \*\*= | x \*\*= y | x = x \*\* y |

3.)JavaScript Comparison Operators:

|  |  |
| --- | --- |
| **Operator** | **Description** |
| == | equal to |
| === | equal value and equal type |
| != | not equal |
| !== | not equal value or not equal type |
| > | greater than |
| < | less than |
| >= | greater than or equal to |
| <= | less than or equal to |
| ? | ternary operator |

4.)JavaScript String Comparison

All the comparison operators above can also be used on strings:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript String Operators</h1>

<p>All conditional operators can be used on both numbers and strings.</p>

<p id="demo"></p>

<script>

let text1 = "A";

let text2 = "B";

let result = text1 < text2;

document.getElementById("demo").innerHTML = "Is A less than B? " + result;

</script>

</body>

</html>

Output:

# JavaScript String Operators

All conditional operators can be used on both numbers and strings.

Is A less than B? true

## 5.)JavaScript String Addition

The + can also be used to add (concatenate) strings:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript String Operators</h1>

<h2>The + Operator</h2>

<p>The + operator concatenates (adds) strings.</p>

<p id="demo"></p>

<script>

let text1 = "John";

let text2 = "Doe";

let text3 = text1 + " " + text2;

document.getElementById("demo").innerHTML = text3;

</script>

</body>

</html>

Output:

# 6.)JavaScript String Operators

## The + Operator

The + operator concatenates (adds) strings.

John Doe

**<!DOCTYPE html>**

**<html>**

**<body>**

**<h1>JavaScript String Operators</h1>**

**<h2>The + Operator</h2>**

**<p>The + operator concatenates (adds) strings.</p>**

**<p id="demo"></p>**

**<script>**

**let text1 = "John";**

**let text2 = "Doe";**

**let text3 = text1 + " " + text2;**

**document.getElementById("demo").innerHTML = text3;**

**</script>**

**</body>**

**</html>**

**Output:**

# JavaScript String Operators

## The + Operator

The + operator concatenates (adds) strings.

John Doe

7.)Adding Strings and Numbers

Adding two numbers, will return the sum, but adding a number and a string will return a string:

**<!DOCTYPE html>**

**<html>**

**<body>**

**<h1>JavaScript String Operators</h1>**

**<h2>The + Operator</h2>**

**<p>Adding a number and a string, returns a string.</p>**

**<p id="demo"></p>**

**<script>**

**let x = 5 + 5;**

**let y = "5" + 5;**

**let z = "Hello" + 5;**

**document.getElementById("demo").innerHTML =**

**x + "<br>" + y + "<br>" + z;**

**</script>**

**</body>**

**</html>**

**Output:**

# JavaScript String Operators

## The + Operator

Adding a number and a string, returns a string.

10  
55  
Hello5

8.)JavaScript Logical Operators

|  |  |
| --- | --- |
| **Operator** | **Description** |
| && | logical and |
| || | logical or |
| ! | logical not |

JavaScript Type Operators

|  |  |
| --- | --- |
| **Operator** | **Description** |
| typeof | Returns the type of a variable |
| instanceof | Returns true if an object is an instance of an object type |

## 9.) JavaScript Bitwise Operators

Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Same as** | **Result** | **Decimal** |
| & | AND | 5 & 1 | 0101 & 0001 | 0001 | 1 |
| | | OR | 5 | 1 | 0101 | 0001 | 0101 | 5 |
| ~ | NOT | ~ 5 | ~0101 | 1010 | 10 |
| ^ | XOR | 5 ^ 1 | 0101 ^ 0001 | 0100 | 4 |
| << | left shift | 5 << 1 | 0101 << 1 | 1010 | 10 |
| >> | right shift | 5 >> 1 | 0101 >> 1 | 0010 | 2 |
| >>> | unsigned right shift | 5 >>> 1 | 0101 >>> 1 | 0010 | 2 |

The examples above uses 4 bits unsigned examples. But JavaScript uses 32-bit signed numbers.  
Because of this, in JavaScript, ~ 5 will not return 10. It will return -6.  
~00000000000000000000000000000101 will return 11111111111111111111111111111010

# JavaScript Data Types

### JavaScript has 8 Datatypes

1. String  
2. Number  
3. Bigint  
4. Boolean  
5. Undefined  
6. Null  
7. Symbol  
8. Object

### The Object Datatype

The object data type can contain:

1. An object  
2. An array  
3. A date

The Concept of Data Types

In programming, data types is an important concept.

To be able to operate on variables, it is important to know something about the type.

Without data types, a computer cannot safely solve this:

let x = 16 + "Volvo";

Does it make any sense to add "Volvo" to sixteen? Will it produce an error or will it produce a result?

JavaScript will treat the example above as:

let x = "16" + "Volvo";

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript</h2>

<p>When adding a string and a number, JavaScript will treat the number as a string.</p>

<p id="demo"></p>

<script>

let x = "Volvo" + 16;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

Output

**JavaScript**

When adding a string and a number, JavaScript will treat the number as a string.

Volvo16

# JavaScript Data Types

### JavaScript has 8 Datatypes

1. String  
2. Number  
3. Bigint  
4. Boolean  
5. Undefined  
6. Null  
7. Symbol  
8. Object

### The Object Datatype

The object data type can contain:

1. An object  
2. An array  
3. A date

Example:

// Numbers:  
let length = 16;  
let weight = 7.5;  
  
// Strings:  
let color = "Yellow";  
let lastName = "Johnson";  
  
// Booleans  
let x = true;  
let y = false;  
  
// Object:  
const person = {firstName:"John", lastName:"Doe"};  
  
// Array object:  
const cars = ["Saab", "Volvo", "BMW"];  
  
// Date object:  
const date = new Date("2022-03-25");

The Concept of Data Types

In programming, data types is an important concept.

To be able to operate on variables, it is important to know something about the type.

Without data types, a computer cannot safely solve this:

let x = 16 + "Volvo";

Does it make any sense to add "Volvo" to sixteen? Will it produce an error or will it produce a result?

JavaScript will treat the example above as:

let x = "16" + "Volvo";

When adding a number and a string, JavaScript will treat the number as a string.

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

let x = 16 + 4 + "Volvo";

20Volvo

let x = "Volvo" + 16 + 4;

Volvo164

JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

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

# 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).

**Example:**

// Function to compute the product of p1 and p2  
function myFunction(p1, p2) {  
  return p1 \* p2;  
}

JavaScript Function Syntax

A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:  
**(*parameter1, parameter2, ...*)**

The code to be executed, by the function, is placed inside curly brackets: **{}**

function name(parameter1, parameter2, parameter3) {  
  // code to be executed  
}

Function **parameters** are listed inside the parentheses () in the function definition.

Function **arguments** are the **values** received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variables.

Function Invocation

The code inside the function will execute when "something" **invokes** (calls) the function:

* When an event occurs (when a user clicks a button)
* When it is invoked (called) from JavaScript code
* Automatically (self invoked)

You will learn a lot more about function invocation later in this tutorial.

Function Return

When JavaScript reaches a return statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Functions often compute a **return value**. The return value is "returned" back to the "caller":

### Example

Calculate the product of two numbers, and return the result:

let x = myFunction(4, 3);   // Function is called, return value will end up in x  
  
function myFunction(a, b) {  
  return a \* b;             // Function returns the product of a and b  
}

The result in x will be:

12

Why Functions?

You can reuse code: Define the code once, and use it many times.

You can use the same code many times with different arguments, to produce different results.

### Example

Convert Fahrenheit to Celsius:

function toCelsius(fahrenheit) {  
  return (5/9) \* (fahrenheit-32);  
}  
document.getElementById("demo").innerHTML = toCelsius(77);

The () Operator Invokes the Function

Using the example above, toCelsius refers to the function object, and toCelsius() refers to the function result.

Accessing a function without () will return the function object instead of the function result.

function toCelsius(fahrenheit) {  
  return (5/9) \* (fahrenheit-32);  
}  
document.getElementById("demo").innerHTML = toCelsius;

Functions Used as Variable Values

Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.

### Example

Instead of using a variable to store the return value of a function:

let x = toCelsius(77);  
let text = "The temperature is " + x + " Celsius";

You can use the function directly, as a variable value:

let text = "The temperature is " + toCelsius(77) + " Celsius";

Local Variables

Variables declared within a JavaScript function, become **LOCAL** to the function.

Local variables can only be accessed from within the function.

// code here can NOT use carName  
  
function myFunction() {  
  let carName = "Volvo";  
  // code here CAN use carName  
}  
  
// code here can NOT use carName

# JavaScript Objects

Real Life Objects, Properties, and Methods

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** |
| https://www.w3schools.com/js/objectExplained.gif | car.name = Fiat  car.model = 500  car.weight = 850kg  car.color = white | car.start()  car.drive()  car.brake()  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**.

JavaScript Objects

You have already learned that JavaScript variables are containers for data values.

This code assigns a **simple value** (Fiat) to a **variable** named car:

let car = "Fiat";

Objects are variables too. But objects can contain many values.

This code assigns **many values** (Fiat, 500, white) to a **variable** named car:

const car = {type:"Fiat", model:"500", color:"white"};

The values are written as **name:value** pairs (name and value separated by a colon).

It is a common practice to declare objects with the const keyword.

Learn more about using const with objects in the chapter: [JS Const](https://www.w3schools.com/js/js_const.asp).

Object Definition

You define (and create) a JavaScript object with an object literal:

const 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 |

Accessing Object Properties

You can access object properties in two ways:

*objectName.propertyName*

Object Methods

Objects can also have **methods**.

Methods are **actions** that can be performed on objects.

Methods are stored in properties as **function definitions**.

|  |  |
| --- | --- |
| **Property** | **Property Value** |
| firstName | John |
| lastName | Doe |
| age | 50 |
| eyeColor | blue |
| fullName | function() {return this.firstName + " " + this.lastName;} |

# JavaScript Events

HTML events are **"things"** that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

HTML Events

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

* An HTML web page has finished loading
* An HTML input field was changed
* An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

With single quotes:

<element event=**'*some JavaScript*'**>

<element event=**"*some JavaScript*"**>

Example:

<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>

In the example above, the JavaScript code changes the content of the element with id="demo".

Common HTML Events

Here is a list of some common HTML events:

|  |  |
| --- | --- |
| **Event** | **Description** |
| onchange | An HTML element has been changed |
| onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse over an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| onload | The browser has finished loading the page |

JavaScript Event Handlers

Event handlers can be used to handle and verify user input, user actions, and browser actions:

* Things that should be done every time a page loads
* Things that should be done when the page is closed
* Action that should be performed when a user clicks a button
* Content that should be verified when a user inputs data
* And more ...

Many different methods can be used to let JavaScript work with events:

* HTML event attributes can execute JavaScript code directly
* HTML event attributes can call JavaScript functions
* You can assign your own event handler functions to HTML elements
* You can prevent events from being sent or being handled
* And more ...

# JavaScript String Methods

|  |  |
| --- | --- |
| String length String slice() String substring() String substr() String replace() String replaceAll() String toUpperCase() String toLowerCase() String concat() | String trim() String trimStart() String trimEnd() String padStart() String padEnd() String charAt() String charCodeAt() String split() |

JavaScript String Length

The length property returns the length of a string:

let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
let length = text.length;

Converting to Upper and Lower Case

A string is converted to upper case with toUpperCase():

A string is converted to lower case with toLowerCase():

JavaScript String toUpperCase()

let text1 = "Hello World!";  
let text2 = text1.toUpperCase();

JavaScript String toLowerCase()

let text1 = "Hello World!";       // String  
let text2 = text1.toLowerCase();  // text2 is text1 converted to lower

JavaScript String concat()

concat() joins two or more strings:

let text1 = "Hello";  
let text2 = "World";  
let text3 = text1.concat(" ", text2);

# JavaScript Number Methods

These **number methods** can be used on all JavaScript numbers:

|  |  |
| --- | --- |
| **Method** | **Description** |
| toString() | Returns a number as a string |
| toExponential() | Returns a number written in exponential notation |
| toFixed() | Returns a number written with a number of decimals |
| toPrecision() | Returns a number written with a specified length |
| ValueOf() | Returns a number as a number |

The toString() Method

The toString() method returns a number as a string.

All number methods can be used on any type of numbers (literals, variables, or expressions):

Example:

let x = 123;  
x.toString();  
(123).toString();  
(100 + 23).toString();

**JavaScript Number Methods**

The toString() method converts a number to a string.

123  
123  
123

# JavaScript Arrays

An array is a special variable, which can hold more than one value:

const cars = ["Saab", "Volvo", "BMW"];

Why Use Arrays?

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

let car1 = "Saab";  
let car2 = "Volvo";  
let car3 = "BMW";

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

const array\_name = [item1, item2, ...];

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Arrays</h2>

<p id="demo"></p>

<script>

const cars = ["Saab", "Volvo", "BMW"];

document.getElementById("demo").innerHTML = cars;

</script>

</body>

</html>

**Output: JavaScript Arrays**

Saab,Volvo,BMW

# JavaScript Array Methods

Converting Arrays to Strings

The JavaScript method toString() converts an array to a string of (comma separated) array values.

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo").innerHTML = fruits.toString();

output:

Banana,Orange,Apple,Mango

Popping and Pushing

When you work with arrays, it is easy to remove elements and add new elements.

This is what popping and pushing is:

Popping items **out** of an array, or pushing items **into** an array.

JavaScript Array pop()

The pop() method removes the last element from an array:

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.pop();

**JavaScript Array Methods**

**pop()**

The pop() method removes the last element from an array.

Banana,Orange,Apple,Mango

Banana,Orange,Apple

JavaScript Array push()

The push() method adds a new element to an array (at the end):

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.push("Kiwi");

# JavaScript Arrays

## The push() Method

The push() method returns the new array length:

5

Banana,Orange,Apple,Mango,Kiwi

# JavaScript Date Objects

const d = new Date();

avaScript Date Output

By default, JavaScript will use the browser's time zone and display a date as a full text string:

**Mon Apr 10 2023 10:01:27 GMT+0530 (India Standard Time)**

Creating Date Objects

Date objects are created with the new Date() constructor.

There are **9 ways** to create a new date object:

new Date()  
new Date(date string)  
  
new Date(year,month)  
new Date(year,month,day)  
new Date(year,month,day,hours)  
new Date(year,month,day,hours,minutes)  
new Date(year,month,day,hours,minutes,seconds)  
new Date(year,month,day,hours,minutes,seconds,ms)  
  
new Date(milliseconds)

JavaScript new Date()

new Date() creates a date object with the **current date and time**:

const d = new Date();

# JavaScript Get Date Method:

The new Date() Constructor

In JavaScript, date objects are created with new Date().

new Date() returns a date object with the current date and time

Date Get Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| getFullYear() | Get **year** as a four digit number (yyyy) |
| getMonth() | Get **month** as a number (0-11) |
| getDate() | Get **day** as a number (1-31) |
| getDay() | Get **weekday** as a number (0-6) |
| getHours() | Get **hour** (0-23) |
| getMinutes() | Get **minute** (0-59) |
| getSeconds() | Get **second** (0-59) |
| getMilliseconds() | Get **millisecond** (0-999) |
| getTime() | Get **time** (milliseconds since January 1, 1970) |

const d = new Date("2021-03-25");  
d.getFullYear();

# JavaScript Dates

## The getFullYear() Method

Return the full year of a date object:

2023

# JavaScript Set Date Methods

Set Date methods let you set date values (years, months, days, hours, minutes, seconds, milliseconds) for a Date Object.

Set Date Methods

Set Date methods are used for setting a part of a date:

|  |  |
| --- | --- |
| **Method** | **Description** |
| setDate() | Set the day as a number (1-31) |
| setFullYear() | Set the year (optionally month and day) |
| setHours() | Set the hour (0-23) |
| setMilliseconds() | Set the milliseconds (0-999) |
| setMinutes() | Set the minutes (0-59) |
| setMonth() | Set the month (0-11) |
| setSeconds() | Set the seconds (0-59) |
| setTime() | Set the time (milliseconds since January 1, 1970) |

The setFullYear() Method

The setFullYear() method sets the year of a date object. In this example to 2020:

const d = new Date();  
d.setFullYear(2023);

**JavaScript setFullYear()**

The setFullYear() method sets the year of a date object:

Mon Apr 10 2023 10:08:38 GMT+0530 (India Standard Time)

# JavaScript Math Object

The JavaScript Math object allows you to perform mathematical tasks on numbers.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Math.PI</h2>

<p>Math.PI returns the ratio of a circle's circumference to its diameter:</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = Math.PI;

</script>

</body>

</html>

**JavaScript Math.PI**

Math.PI returns the ratio of a circle's circumference to its diameter:

3.141592653589793

The Math Object

Unlike other objects, the Math object has no constructor.

The Math object is static.

All methods and properties can be used without creating a Math object first.

Math Properties (Constants)

The syntax for any Math property is : Math.*property*.

JavaScript provides 8 mathematical constants that can be accessed as Math properties:

Math.E        // returns Euler's number  
Math.PI       // returns PI  
Math.SQRT2    // returns the square root of 2  
Math.SQRT1\_2  // returns the square root of 1/2  
Math.LN2      // returns the natural logarithm of 2  
Math.LN10     // returns the natural logarithm of 10  
Math.LOG2E    // returns base 2 logarithm of E  
Math.LOG10E   // returns base 10 logarithm of E

# JavaScript Random

Math.random()

Math.random() returns a random number between 0 (inclusive),  and 1 (exclusive):

### Example

// Returns a random number:  
Math.random();

Math.random() always returns a number lower than 1.

JavaScript Random Integers

Math.random() used with Math.floor() can be used to return random integers.

There is no such thing as JavaScript integers.

We are talking about numbers with no decimals here.

# JavaScript Booleans

[❮ Previous](https://www.w3schools.com/js/js_random.asp)[Next ❯](https://www.w3schools.com/js/js_comparisons.asp)

A JavaScript Boolean represents one of two values: **true** or **false**.

Boolean Values

Very often, in programming, you will need a data type that can only have one of two values, like

* YES / NO
* ON / OFF
* TRUE / FALSE

For this, JavaScript has a **Boolean** data type. It can only take the values **true** or **false**.

The Boolean() Function

You can use the Boolean() function to find out if an expression (or a variable) is true:

Eg:Boolean(10 > 9)

Comparisons and Conditions

The chapter JS Comparisons gives a full overview of comparison operators.

The chapter JS Conditions gives a full overview of conditional statements.

Here are some examples:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | equal to | if (day == "Monday") |
| > | greater than | if (salary > 9000) |
| < | less than | if (age < 18) |

# JavaScript For Loop

Loops can execute a block of code a number of times.

JavaScript Loops

Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this is the case when working with arrays:

### Instead of writing:

text += cars[0] + "<br>";  
text += cars[1] + "<br>";  
text += cars[2] + "<br>";  
text += cars[3] + "<br>";  
text += cars[4] + "<br>";  
text += cars[5] + "<br>";

### You can write:

for (let i = 0; i < cars.length; i++) {  
  text += cars[i] + "<br>";  
}

Different Kinds of Loops

JavaScript supports different kinds of loops:

* for - loops through a block of code a number of times
* for/in - loops through the properties of an object
* for/of - loops through the values of an iterable object
* while - loops through a block of code while a specified condition is true
* do/while - also loops through a block of code while a specified condition is true

## The For Loop

The for statement creates a loop with 3 optional expressions:

for (*expression 1*;*expression 2*;*expression 3*) {  
  // *code block to be executed*  
}

**Expression 1** is executed (one time) before the execution of the code block.

**Expression 2** defines the condition for executing the code block.

**Expression 3** is executed (every time) after the code block has been executed.

### Example

for (let i = 0; i < 5; i++) {  
  text += "The number is " + i + "<br>";  
}

# JavaScript While Loop

Loops can execute a block of code as long as a specified condition is true.

## The While Loop

The while loop loops through a block of code as long as a specified condition is true.

### Syntax

while (condition) {  
*// code block to be executed*  
}

eg:

while (i < 10) {  
  text += "The number is " + i;  
  i++;  
}

## The Do While Loop

The do while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

### Syntax

do {  
*// code block to be executed*}  
while (condition);

do {  
  text += "The number is " + i;  
  i++;  
}  
while (i < 10);

# JavaScript typeof

In JavaScript there are 5 different data types that can contain values:

* string
* number
* boolean
* object
* function

There are 6 types of objects:

* Object
* Date
* Array
* String
* Number
* Boolean

And 2 data types that cannot contain values:

* null
* undefined

The typeof Operator

You can use the typeof operator to find the data type of a JavaScript variable.

typeof "John"                 // Returns "string"  
typeof 3.14                   // Returns "number"  
typeof NaN                    // Returns "number"  
typeof false                  // Returns "boolean"  
typeof [1,2,3,4]              // Returns "object"  
typeof {name:'John', age:34}  // Returns "object"  
typeof new Date()             // Returns "object"  
typeof function () {}         // Returns "function"  
typeof myCar                  // Returns "undefined" \*  
typeof null                   // Returns "object"

# JavaScript Type Conversion

* Converting Strings to Numbers
* Converting Numbers to Strings
* Converting Dates to Numbers
* Converting Numbers to Dates
* Converting Booleans to Numbers
* Converting Numbers to Booleans

JavaScript Type Conversion

JavaScript variables can be converted to a new variable and another data type:

* By the use of a JavaScript function
* **Automatically** by JavaScript itself

Converting Strings to Numbers

The global method Number() converts a variable (or a value) into a number.

A numeric string (like "3.14") converts to a number (like 3.14).

An empty string (like "") converts to 0.

A non numeric string (like "John") converts to NaN (Not a Number).

### Examples

These will convert:

Number("3.14")  
Number(Math.PI)  
Number(" ")  
Number("")

These will not convert:

Number("99 88")  
Number("John")

## Number Methods

In the chapter [Number Methods](https://www.w3schools.com/js/js_number_methods.asp), you will find more methods that can be used to convert strings to numbers:

|  |  |
| --- | --- |
| **Method** | **Description** |
| Number() | Returns a number, converted from its argument |
|  |  |
| parseFloat() | Parses a string and returns a floating point number |
| parseInt() | Parses a string and returns an integer |

## Converting Numbers to Strings

The global method String() can convert numbers to strings.

It can be used on any type of numbers, literals, variables, or expressions:

### Example

String(x)         // returns a string from a number variable x  
String(123)       // returns a string from a number literal 123  
String(100 + 23)  // returns a string from a number from an expression

# JavaScript Scope

Scope determines the accessibility (visibility) of variables.

JavaScript has 3 types of scope:

* Block scope
* Function scope
* Global scope