freelance_Project available to buy contact on 8007592194

SR.NO	Project NAME	Technology
1	Online E-Learning Platform Hub	React+Springboot+MySql
2	PG Mates / RoomSharing / Flat Mates	React+Springboot+MySql
3	Tour and Travel management System	React+Springboot+MySql
4	Election commition of India (online Voting System)	React+Springboot+MySql
5	HomeRental Booking System	React+Springboot+MySql
6	Event Management System	React+Springboot+MySql
7	Hotel Management System	React+Springboot+MySql
8	Agriculture web Project	React+Springboot+MySql
9	AirLine Reservation System / Flight booking System	React+Springboot+MySql
10	E-commerce web Project	React+Springboot+MySql
11	Hospital Management System	React+Springboot+MySql
12	E-RTO Driving licence portal	React+Springboot+MySql
13	Transpotation Services portal	React+Springboot+MySql
14	Courier Services Portal / Courier Management System	React+Springboot+MySql
15	Online Food Delivery Portal	React+Springboot+MySql
16	Muncipal Corporation Management	React+Springboot+MySql
17	Gym Management System	React+Springboot+MySql
18	Bike/Car ental System Portal	React+Springboot+MySql
19	CharityDonation web project	React+Springboot+MySql
20	Movie Booking System	React+Springboot+MySql

freelance_Project available to buy contact on 8007592194

21	Job Portal web project	React+Springboot+MySql
22	LIC Insurance Portal	React+Springboot+MySql
23	Employee Management System	React+Springboot+MySql
24	Payroll Management System	React+Springboot+MySql
25	RealEstate Property Project	React+Springboot+MySql
26	Marriage Hall Booking Project	React+Springboot+MySql
27	Online Student Management portal	React+Springboot+MySql
28	Resturant management System	React+Springboot+MySql
29	Solar Management Project	React+Springboot+MySql
30	OneStepService LinkLabourContractor	React+Springboot+MySql
31	Vehical Service Center Portal	React+Springboot+MySql
32	E-wallet Banking Project	React+Springboot+MySql
33	Blogg Application Project	React+Springboot+MySql
34	Car Parking booking Project	React+Springboot+MySql
35	OLA Cab Booking Portal	React+Springboot+MySql
36	Society management Portal	React+Springboot+MySql
37	E-College Portal	React+Springboot+MySql
38	FoodWaste Management Donate System	React+Springboot+MySql
39	Sports Ground Booking	React+Springboot+MySql
40	BloodBank mangement System	React+Springboot+MySql
41	Bus Tickit Booking Project	React+Springboot+MySql
42	Fruite Delivery Project	React+Springboot+MySql
43	Woodworks Bed Shop	React+Springboot+MySql
44	Online Dairy Product sell Project	React+Springboot+MySql
45	Online E-Pharma medicine sell Project	React+Springboot+MySql
46	FarmerMarketplace Web Project	React+Springboot+MySql
47	Online Cloth Store Project	React+Springboot+MySql
48		React+Springboot+MySql
49		React+Springboot+MySql
50		React+Springboot+MySql

https://www.youtube.com/@codewitharrays

https://www.instagram.com/codewitharrays/

https://t.me/codewitharrays Group Link: https://t.me/cceesept2023

+91 8007592194 +91 9284926333

codewitharrays@gmail.com

https://codewitharrays.in/project

Hyper Text Markup Language

(HTML)

INDEX

Topics **Page No**

What is HTML?

HTML is a language for describing web pages.

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is not a programming language, it is a **markup language**
- A markup language is a set of **markup tags**
- HTML uses **markup tags** to describe web pages

BASIC HTML Page Structure

```
<html>
<head>
<title>Page Title</title>
</head>
<body>
<p>
            Page Content
</p>
</body>
</html>
```

HTML Head

The Head Element

The head element contains general information, also called meta-information, about a document. Meta means "information about".

You can say that meta-data means information about data, or meta-information means information about information.

Information Inside the Head Element

The elements inside the head element should not be displayed by a browser.

According to the HTML standard, only a few tags are legal inside the head section. These are: <base>, <link>, <meta>, <title>, <style>, and <script>.

Look at the following illegal construct:

```
<head>
<p>This is some text</p>
</head>
```

In this case the browser has two options:

- Display the text because it is inside a paragraph element
- Hide the text because it is inside a head element

If you put an HTML element like <h1> or <p> inside a head element like this, most browsers will display it, even if it is illegal.

Should browsers forgive you for errors like this? We don't think so. Others do.

Head Tags

Tag	Description
<head>	Defines information about the document
<title>	Defines the document title
s<link>	Defines a resource reference
<meta>	Defines meta information

Tag	Description
<!DOCTYPE>	Defines the document type. This tag goes before the <html> start tag.

HTML Meta

The head element contains general information (meta-information) about a document.

HTML also includes a meta element that goes inside the head element. The purpose of the meta element is to provide meta-information about the document.

Most often the meta element is used to provide information that is relevant to browsers or search engines like describing the content of your document.

Keywords for Search Engines

Some search engines on the WWW will use the name and content attributes of the meta tag to index your pages.

This meta element defines a description of your page:

```
<meta name="description" content="Free Web tutorials on HTML, CSS, XML,
and XHTML" />
```

This meta element defines keywords for your page:

```
<meta name="keywords" content="HTML, DHTML, CSS, XML, XHTML, JavaScript"
/>
```

The intention of the name and content attributes is to describe the content of a page.

HTML <!DOCTYPE> Declaration

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Title of the document</title>
</head>
<body>
The content of the document......
</body>
</html>
```

Definition and Usage

The doctype declaration should be the very first thing in an HTML document, before the <html> tag.

The doctype declaration is not an HTML tag; it is an instruction to the web browser about what version of the markup language the page is written in.

The doctype declaration refers to a Document Type Definition (DTD). The DTD specifies the rules for the markup language, so that the browsers can render the content correctly.

Doctypes Available in the W3C Recommendations

HTML 4.01 Strict

This DTD contains all HTML elements and attributes, but does not include presentational or deprecated elements (like font). Framesets are not allowed.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

HTML 4.01 Transitional

This DTD contains all HTML elements and attributes, including presentational and deprecated elements (like font). Framesets are not allowed.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

HTML 4.01 Frameset

This DTD is equal to HTML 4.01 Transitional, but allows the use of frameset content.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

XHTML 1.0 Strict

This DTD contains all HTML elements and attributes, but does not include presentational or deprecated elements (like font). Framesets are not allowed. The markup must also be written as well-formed XML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitional

This DTD contains all HTML elements and attributes, including presentational and deprecated elements (like font). Framesets are not allowed. The markup must also be written as well-formed XML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML 1.0 Frameset

This DTD is equal to XHTML 1.0 Transitional, but allows the use of frameset content.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

XHTML 1.1

This DTD is equal to XHTML 1.0 Strict, but allows you to add modules (for example to provide ruby support for East-Asian languages).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

HTML <link> tag

The <link> tag defines the relationship between a document and an external resource.

The <link> tag is most used to link to style sheets.

HTML <title> tag

Definition and Usage

The <title> tag defines the title of the document.

The title element is required in all HTML/XHTML documents.

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

HTML Tags

HTML markup tags are usually called HTML tags

- HTML tags are keywords surrounded by **angle brackets** like <html>
- HTML tags normally **come in pairs** like <b> and </b>
- The first tag in a pair is the **start tag**, the second tag is the **end tag**
- Start and end tags are also called **opening tags** and **closing tags**.

HTM or HTML Extension?

When you save an HTML file, you can use either the .htm or the .html extension.

HTML Headings

HTML headings are defined with the <h1> to <h6> tags.

Example:

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

HTML Paragraphs

HTML paragraphs are defined with the <p> tag.

Example:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

HTML Links

HTML links are defined with the <a> tag.

Example:

```
<a href="http://www.infinitech.in">This is a link</a>
```

HTML Images

HTML images are defined with the <img> tag.

Example:

```
<img src="test.jpg" width="104" height="142" />
```

HTML Comments

Comments can be inserted in the HTML code to make it more readable and understandable. Comments are ignored by the browser and are not displayed.

Comments are written like this:

Example:- **<!-- This is a comment -->**
Note: There is an exclamation point after the opening bracket, but not before the closing bracket.

HTML Tip - How to View HTML Source

Have you ever seen a Web page and wondered "Hey! How did they do that?"

To find out, click the VIEW option in your browser's toolbar and select SOURCE or PAGE SOURCE. This will open a window that shows you the HTML code of the page.

HTML Line Breaks

Use the <br /> tag if you want a line break (a new line) without starting a new paragraph:

Example:-

```
<p>This is<br />a para<br />graph with line breaks</p>
```

The <br /> element is an empty HTML element. It has no end tag.

<br> or <br />

In XHTML, XML, and future versions of HTML, HTML elements with no end tag (closing tag) are not allowed.

Even if <br> works in all browsers, writing <br /> instead is more **future proof**.

HTML Text Formatting

This text is bold

This text is big

This text is italic

This is computer output

This is $_{\text{subscript}}$ and $^{\text{superscript}}$

HTML Formatting Tags

HTML uses tags like <b> and <i> for formatting output, like **bold** or *italic* text.

These HTML tags are called formatting tags.

Text Formatting Tags

Tag	Description
<b>	Defines bold text
<big>	Defines big text
<em>	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
<strong>	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<s>	Deprecated. Use <del> instead
<strike>	Deprecated. Use <del> instead
<u>	Deprecated. Use styles instead

HTML Links

Hyperlinks, Anchors, and Links

In web terms, a hyperlink is a reference (an address) to a resource on the web.

Hyperlinks can point to any resource on the web: an HTML page, an image, a sound file, a movie, etc.

An anchor is a term used to define a hyperlink destination inside a document.

The HTML anchor element <a> is used to define both hyperlinks and anchors.

We will use the term HTML link when the <a> element points to a resource, and the term HTML anchor when the <a> elements defines an address inside a document..

An HTML Link

Link syntax:

```
<a href="url">Link text</a>
```

The start tag contains attributes about the link.

The element content (Link text) defines the part to be displayed.

Note: The element content doesn't have to be text. You can link from an image or any other HTML element.

The href Attribute

The **href attribute** defines the link "address".

This <a> element defines a link to Infinitech:

```
<a href="http://www.infinitech.in/">Visit Infinitech Web Site!</a>
```

The target Attribute

The **target attribute** defines **where** the linked document will be opened.

The code below will open the document in a new browser window:

Example

```
<a href="http://www.infinitech.in/"
target="_blank">Visit Infinitech Website!</a>
```

The name Attribute

When the **name attribute** is used, the <a> element defines a named anchor inside a HTML document.

Named anchor are not displayed in any special way. They are invisible to the reader.

Named anchor syntax:

```
<a name="label">Any content</a>
```

The link syntax to a named anchor:

```
<a href="#label">Any content</a>
```

The # in the href attribute defines a link to a named anchor.

Example:

A named anchor inside an HTML document:

```
<a name="tips">Useful Tips Section</a>
```

A link to the Useful Tips Section from the same document:

```
<a href="#tips">
Jump to the Useful Tips Section</a>
```

A link to the Useful Tips Section from another document:

```
<a href="http://www.infinitech.in/html_tutorial.htm#tips">
Jump to the Useful Tips Section</a>
```

HTML IMAGES :

Inserting Images:

```
<html>
<body>

<p>
An image:
<img src="constr4.gif"
```

```
width="144" height="50">
</p>

<p>
A moving image:
<img src="hackanm.gif"
width="48" height="48">
</p>

<p>
Note that the syntax of inserting a moving image is no different from
that of a non-moving image.
</p>

</body>
</html>
```

Insert Images From Different Locations:

```
<html>
<body>

<p>An image from SMGSM:</p>
<img src="http://smgsm.com/images/logo.jpg">

</body>
</html>
```

The Image Tag and the Src Attribute

In HTML, images are defined with the <img> tag.

The <img> tag is empty, which means that it contains attributes only and it has no closing tag.

To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display on your page.

The syntax of defining an image:

```
<img src="url" />
```

The URL points to the location where the image is stored. An image named "boat.gif" located in the directory "images" on "www.infinitech.in" has the URL: http://www.infinitech.in/images/boat.gif.

The browser puts the image where the image tag occurs in the document. If you put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

The Alt Attribute

The alt attribute is used to define an "alternate text" for an image. The value of the alt attribute is an author-defined text:

```
<img src="boat.gif" alt="Big Boat" />
```

The "alt" attribute tells the reader what he or she is missing on a page if the browser can't load images. The browser will then display the alternate text instead of the image. It is a good practice to include the "alt" attribute for each image on a page, to improve the display and usefulness of your document for people who have text-only browsers.

Basic Notes - Useful Tips

If an HTML file contains ten images - eleven files are required to display the page right. Loading images take time, so my best advice is: Use images carefully.

Background Images:

```
<html>
<body background="background.jpg">
<h3>Look: A background image!</h3>
<p>If the image is smaller than the page, the image will repeat
itself.</p>
</body>
</html>
```

Alignment of Image:

```
<html>
<body>

<p>
An image
<img src="hackanm.gif"
align="bottom" width="48" height="48">
in the text
</p>
```

```
<p>
An image
<img src ="hackanm.gif"
align="middle" width="48" height="48">
in the text
</p>

<p>
An image
<img src ="hackanm.gif"
align="top" width="48" height="48">
in the text
</p>

<p>Note that bottom alignment is the default alignment</p>

<p>
An image
<img src ="hackanm.gif"
width="48" height="48">
in the text
</p>

<p>
<img src ="hackanm.gif"
width="48" height="48">
An image before the text
</p>

<p>
An image after the text
<img src ="hackanm.gif"
width="48" height="48">
</p>

</body>
</html>
```

Image Float , Size, Hyperlink of Image and ALT tag:

```
<html>
<body>

<p>
<img src ="hackanm.gif"
align ="left" width="48" height="48" alt=”An Image”>
A paragraph with an image. The align attribute of the image is set to
"left". The image will float to the left of this text.
</p>

<p>
<img src ="hackanm.gif"
```

```
align ="right" width="58" height="58" alt=”An Image”>
A paragraph with an image. The align attribute of the image is set to
"right". The image will float to the right of this text.
</p>

<p>
You can also use an image as a link:
<a href="lastpage.htm">
<img border="0" src="buttonnext.gif" width="65" height="38">
</a>
</p>

</body>
</html>
```

Note: You can make a picture larger or smaller changing the values in the "height" and "width" attributes of the img tag.

ALT Tag: The "alt" attribute tells the reader what he or she is missing on a page if the browser can't load images. It is a good practice to include the "alt" attribute for each image on a page.

HTML TABLES

Table and Border:

Tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). The letters td stands for "table data," which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

If you do not specify a border attribute the table will be displayed without any borders. Sometimes this can be useful, but most of the time, you want the borders to show.

To display a table with borders, you will have to use the border attribute:

```
<html>
<body>

<p>
Each table starts with a table tag.
Each table row starts with a tr tag.
Each table data starts with a td tag.
</p>

<h4>One column:</h4>
<table border="1">
<tr>
```

```
  <td>100</td>
</tr>
</table>

<h4>One row and three columns:</h4>
<table border="4">
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
</table>

<h4>Two rows and three columns:</h4>
<table border="8">
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>

</body>
</html>
```

Headings in a Table

Headings in a table are defined with the <th> tag.

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How it looks in a browser:

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Empty Cells in a Table

Table cells with no content are not displayed very well in most browsers.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td></td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Note that the borders around the empty table cell are missing (NB! Mozilla Firefox displays the border).

To avoid this, add a non-breaking space (&nbsp;) to empty data cells, to make the borders visible:

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>&nbsp;</td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Basic Notes - Useful Tips

The <thead>,<tbody> and <tfoot> elements are seldom used, because of bad browser support. Expect this to change in future versions of XHTML.

Table with no border

Make border = “0”

Headings in a table

```
<html>
<body>

<h4>Table headers:</h4>
<table border="1">
<tr>
  <th>Name</th>
</tr>
<tr>
  <td>Bill Gates</td>
</tr>
</table>

</body>
</html>
```

Empty cells

To avoid empty cells insert &nbsp; in a cell

Table with a caption

```
<html>
<body>
<table border="6">
<caption>My Caption</caption>
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>
</body>
</html>
```

Output:

My Caption

100	200	300
400	500	600

Table cells that span more than one row/column

```
<table border="1">
<tr>
  <th>Name</th>
  <th colspan="2">Telephone</th>
</tr>
<tr>
  <td>Bill Gates</td>
  <td>555 77 854</td>
  <td>555 77 855</td>
</tr>
</table>
```

Output:

Name	Telephone	
Bill Gates	555 77 854	555 77 855

```
<table border="1">
<tr>
  <th>First Name:</th>
  <td>Bill Gates</td>
</tr>
<tr>
  <th rowspan="2">Telephone:</th>
  <td>555 77 854</td>
</tr>
<tr>
  <td>555 77 855</td>
</tr>
</table>
```

First Name:	Bill Gates
Telephone:	555 77 854
	555 77 855

Tags inside a table

```
<table border="1">
<tr>
  <td>
   <p>This is a paragraph</p>
   <p>This is another paragraph</p>
  </td>
  <td>This cell contains a table:
   <table border="1">
   <tr>
     <td>A</td>
     <td>B</td>
   </tr>
   <tr>
     <td>C</td>
     <td>D</td>
   </tr>
   </table>
  </td>
</tr>
<tr>
  <td>This cell contains a list
   <ul>
    <li>apples</li>
    <li>bananas</li>
    <li>pineapples</li>
   </ul>
  </td>
  <td>HELLO</td>
</tr>
</table>
```

Cell padding

```
<table border="1">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

<h4>With cellpadding:</h4>
<table border="1"
cellpadding="10">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>
```

Without cellpadding:

First	Row
Second	Row

With cellpadding:

First	Row
Second	Row

Cell spacing

```
<h4>Without cellspacing:</h4>
<table border="1">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

<h4>With cellspacing:</h4>
<table border="1"
cellspacing="10">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>
```

Without cellspacing:

First	Row
Second	Row

With cellspacing:

First	Row
Second	Row

Add a background color or a background image to a table

```
<h4>A background color:</h4>
<table border="1"
bgcolor="red">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>
<h4>A background image:</h4>
<table border="1"
background="bgdesert.jpg">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>
```

A background color:

First	Row
Second	Row

Add a background color or a background image to a table cell

```
<table border="1">
<tr>
  <td bgcolor="red">First</td>
  <td>Row</td>
</tr>
<tr>
  <td
  background="bgdesert.jpg">
  Second</td>
  <td>Row</td>
</tr>
</table>
```

Cell backgrounds:

First	Row
Second	Row

Special Characters

Code	Symbol	Description
&trade;	™	Trademark
&amp;	&	Ampersand
&reg;	®	Registered trademark
&copy;	©	Copyright
&dagger;	†	Dagger
&raquo;	»	Right pointing double angle quotation mark
&laquo;	«	Left pointing double angle quotation mark
&#151;	—	Em-dash
&deg;	30°	Degree
&frac14;	¼	Quarter
&frac12;	½	Half
&frac34;	¾	Three quarters
&middot;	·	Middle dot
&iexcl;	¡	Inverted exclamation mark

Ordered and Unordered List in HTML

Unordered Lists

An unordered list is a list of items. The list items are marked with bullets (typically small black circles).

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

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

Here is how it looks in a browser:

- Coffee

- Milk

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

Ordered Lists

An ordered list is also a list of items. The list items are marked with numbers.

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

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

Here is how it looks in a browser:

1. Coffee
2. Milk

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

HTML Code for Forms and Input Fields

Forms

A form is an area that can contain form elements.

Form elements are elements that allow the user to enter information (like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.) in a form.

A form is defined with the <form> tag.

```
<form>
.
input elements
.
</form>
```

Input

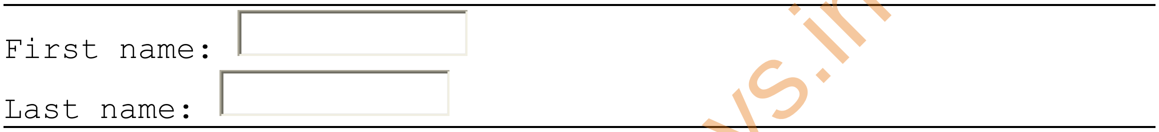
The most used form tag is the <input> tag. The type of input is specified with the type attribute. The most commonly used input types are explained below.

Text Fields

Text fields are used when you want the user to type letters, numbers, etc. in a form.

```
<form>
First name:
<input type="text" name="firstname" />
<br />
Last name:
<input type="text" name="lastname" />
</form>
```

How it looks in a browser:



First name:

Last name:

Note that the form itself is not visible. Also note that in most browsers, the width of the text field is 20 characters by default.

Radio Buttons

Radio Buttons are used when you want the user to select one of a limited number of choices.

```
<form>
<input type="radio" name="sex" value="male" /> Male
<br />
<input type="radio" name="sex" value="female" /> Female
</form>
```

How it looks in a browser:

○ Male

○ Female

Note that only one option can be chosen.

Checkboxes

Checkboxes are used when you want the user to select one or more options of a limited number of choices.

```
<form>
I have a bike:
<input type="checkbox" name="vehicle" value="Bike" />
<br />
I have a car:
<input type="checkbox" name="vehicle" value="Car" />
<br />
I have an airplane:
<input type="checkbox" name="vehicle" value="Airplane" />
</form>
```

How it looks in a browser:

I have a bike: □

I have a car: □

I have an airplane: □

The Form's Action Attribute and the Submit Button

When the user clicks on the "Submit" button, the content of the form is sent to the server. The form's action attribute defines the name of the file to send the content to. The file defined in the action attribute usually does something with the received input.

```
<form name="input" action="html_form_submit.asp"
method="get">
```

```
Username:
<input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

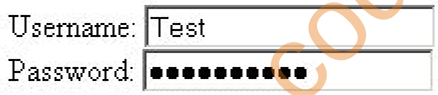
Input Type Password in HTML

Note that when you type characters in a password field, the browser displays asterisks or bullets instead of the characters.

Example:

```
<html>

<body>

<form action="">

Username:

<input type="text" name="user">

<br>

Password:

<input type="password" name="password">

</form>

</body>

</html>
```

Output:



Username: Test
Password: ●●●●●●●●●●●

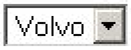
Simple Dropdown List in HTML

Example:

```
<select name="cars">
```

```
<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

<option value="fiat">Fiat</option>

<option value="audi">Audi</option>

</select>

</form>

</body>

</html>
```

Output:



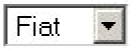
Volvo

Note : If in option tag selected =”selected” is specified then specific value will be selected by default.

Example:

```
<option value="fiat" selected="selected">Fiat</option>
```

Output:



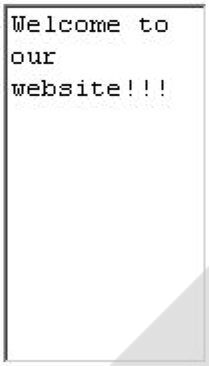
Fiat

Textarea in HTML

Example:

```
<textarea rows=”10” cols=”5”>

Welcome to our Website

</textarea>
```

Output:



```
Welcome to
our
website!!!
```

Cascading Style Sheet

(CSS)

INDEX

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles are normally stored in **Style Sheets**
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**
- Multiple style definitions will **cascade** into one

Why Css?

The original HTML was never intended to contain tags for formatting a document. HTML tags were intended to define the content of a document, like:

```
<p>This is a paragraph.</p>

<h1>This is a heading</h1>
```

When tags like <font> and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large web sites where fonts and color information had to be added to every single Web page, became a long, expensive and unduly painful process.

To solve this problem, the World Wide Web Consortium (W3C) - responsible for standardizing HTML - created CSS in addition to HTML 4.0.

With HTML 4.0, all formatting can be removed from the HTML document and stored in a separate CSS file.

All browsers support CSS today.

CSS save a lot of work

Styles sheets define HOW HTML elements are to be displayed.

Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in a Web site, just by editing one single CSS document!

CSS types and priority

Style sheets allow style information to be specified in many ways.

Styles can be specified:

- inside an HTML element
- inside the head section of an HTML page
- in an external CSS file

Tip: Even multiple external style sheets can be referenced inside a single HTML document.

Cascading order - What style will be used when there is more than one style specified for an HTML element?

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

1. Browser default
2. External style sheet
3. Internal style sheet (in the head section)
4. Inline style (inside an HTML element)

So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or in a browser (a default value).

If the link to the external style sheet is placed after the internal style sheet in HTML <head>, the external style sheet will override the internal style sheet!

CSS syntax

The CSS syntax is made up of three parts: a selector, a property and a value:

```
selector {property:value}
```

The selector is normally the HTML element/tag you wish to define, the property is the attribute you wish to change, and each property can take a value. The property and value are separated by a colon, and surrounded by curly braces:

```
body {color:black}
```

Note: If the value is multiple words, put quotes around the value:

```
p {font-family:"sans serif"}
```

Note: If you want to specify more than one property, you must separate each property with a semicolon. The example below shows how to define a center aligned paragraph, with a red text color:

```
p {text-align:center;color:red}
```

To make the style definitions more readable, you can describe one property on each line, like this:

```
p
{
text-align:center;
color:black;
font-family:arial
}
```

Grouping

You can group selectors. Separate each selector with a comma. In the example below we have grouped all the header elements. All header elements will be displayed in green text color:

```
h1,h2,h3,h4,h5,h6
{
color:green
}
```

The class Selector

With the class selector you can define different styles for the same type of HTML element.

Say that you would like to have two types of paragraphs in your document: one right-aligned paragraph, and one center-aligned paragraph. Here is how you can do it with styles:

```
p.right {text-align:right}
p.center {text-align:center}
```

You have to use the class attribute in your HTML document:

```
<p class="right">This paragraph will be right-aligned.</p>
<p class="center">This paragraph will be center-aligned.</p>
```

Note: To apply more than one class per given element, the syntax is:

```
<p class="center bold">This is a paragraph.</p>
```

The paragraph above will be styled by the class "center" AND the class "bold".

You can also omit the tag name in the selector to define a style that will be used by all HTML elements that have a certain class. In the example below, all HTML elements with class="center" will be center-aligned:

```
.center {text-align:center}
```

In the code below both the h1 element and the p element have class="center". This means that both elements will follow the rules in the ".center" selector:

```
<h1 class="center">This heading will be center-aligned</h1>
<p class="center">This paragraph will also be center-aligned.</p>
```

Do **NOT** start a class name with a number! It will not work in Mozilla/Firefox.

Add Styles to Elements with Particular Attributes

You can also apply styles to HTML elements with particular attributes.

The style rule below will match all input elements that have a type attribute with a value of "text":

```
input[type="text"] {background-color:blue}
```

The id Selector

You can also define styles for HTML elements with the id selector. The id selector is defined as #.

The style rule below will match the element that has an id attribute with a value of "green":

```
#green {color:green}
```

The style rule below will match the p element that has an id with a value of "para1":

```
p#para1
{
text-align:center;
color:red
}
```

Do **NOT** start an ID name with a number! It will not work in Mozilla/Firefox.

CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. A comment will be ignored by browsers. A CSS comment begins with "/*", and ends with "*/", like this:

```
/*This is a comment*/
p
{
text-align:center;
/*This is another comment*/
color:black;
font-family:arial
}
```

CSS Types

How to Insert a Style Sheet

When a browser reads a style sheet, it will format the document according to it. There are three ways of inserting a style sheet:

External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

The browser will read the style definitions from the file mystyle.css, and format the document according to it.

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a .css extension. An example of a style sheet file is shown below:

```
hr {color:sienna}
p {margin-left:20px}
body {background-image:url("images/back40.gif")}
```

Do not leave spaces between the property value and the units! "margin-left:20 px" (instead of "margin-left:20px") will only work in IE6, but it will not work in Firefox or Opera.

Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section by using the <style> tag, like this:

```
<head>
<style type="text/css">
hr {color:sienna}
p {margin-left:20px}
body {background-image:url("images/back40.gif")}
</style>
</head>
```

The browser will now read the style definitions, and format the document according to it.

Note: A browser normally ignores unknown tags. This means that an old browser that does not support styles, will ignore the <style> tag, but the content of the <style> tag will be displayed on the page. It is possible to prevent an old browser from displaying the content by hiding it in the HTML comment element:

```
<head>
<style type="text/css">
<!--
hr {color:sienna}
p {margin-left:20px}
body {background-image:url("images/back40.gif")}
-->
</style>
</head>
```

Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly, such as when a style is to be applied to a single occurrence of an element.

To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

For example, an external style sheet has these properties for the h3 selector:

```
h3
{
color:red;
text-align:left;
font-size:8pt
}
```

And an internal style sheet has these properties for the h3 selector:

```
h3
{
text-align:right;
font-size:20pt
}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

```
color:red;
text-align:right;
font-size:20pt
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

CSS Background

1. **Color as a background**

```
<style type="text/css">

body

{

background-color:yellow;

}

h1

{

background-color:#00ff00;

}

p

{

background-color:rgb(255,0,255);

}

</style>
```

Note: Default value is transparent.

2. **Image as a background**

```
<style type="text/css">
body {background-image:url('paper.gif')}
</style>
```

3. **How to repeat background Image?**

Note: By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally.

Background Image Vertical Repeat

```
<style type="text/css">
body
{
background-image:url('paper.gif');
background-repeat:repeat-x;
}
</style>
```

Background Image Horizontal Repeat

```
<style type="text/css">
body
{
background-image:url('paper.gif');
background-repeat:repeat-y;
}
</style>
```

Display a Background Image only one time

```
<style type="text/css">
body
{
background-image:url('paper.gif');
background-repeat:no-repeat;
}
</style>
```

4. **Image as a background**

```
<style type="text/css">
body
{
background-image:url('smiley.gif');
background-repeat:no-repeat;
background-attachment:fixed;
background-position:center;
}
</style>
```

Note: For this to work in Firefox and Opera, the background-attachment property must be set to "fixed". Background will not move scroll.

Background-attachment: Sets whether a background image is fixed or scrolls with the rest of the pages. **Default value is Scroll.**

5. **Background Image Position**

```
<style type="text/css">
body {background-image:url('paper.gif')}
</style>
```

Property Values

Value	**Description**
top left top center top right center left center center center right bottom left bottom center bottom right	If you only specify one keyword, the second value will be "center". Default value is: 0% 0%
x% y%	The first value is the horizontal position and the second value is the vertical. The top left corner is 0% 0%. The right bottom corner is 100% 100%. If you only specify one value, the other value will be 50%.
xpos ypos	The first value is the horizontal position and the second value is the vertical. The top left corner is 0 0. Units can be pixels (0px 0px) or any other CSS units. If you only specify one value, the other value will be 50%. You can mix % and positions
inherit	Specifies that the setting of the background-position property should be inherited from the parent element

Example:

```
<style type="text/css">
body
{
background-image: url('smiley.gif');
background-repeat: no-repeat;
background-attachment:fixed;
background-position: 0% 20%;
background-position: 50px 100px;
}
</style>
```

CSS Text

1. **Text Color**

The color property is used to set the color of the text. The color can be set by:

- name - specify a color name, like "red"
- RGB - specify an RGB value, like "rgb(255,0,0)"
- Hex - specify a hex value, like "#ff0000"

The default color for a page is defined in the body selector.

Example

```
body {color:blue}
h1 {color:#00ff00}
h2 {color:rgb(255,0,0)}
```

2. **Text Alignment**

The text-align property is used to set the horizontal alignment of a text.

Text can be centered, or aligned to the left or right, or justified.

When text-align is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

Example:

```
<html>

<head>

<style type="text/css">

h1 {text-align:center}

p.date {text-align:right}

p.main {text-align:justify}

</style>

</head>

<body>

<h1>CSS text-align Example</h1>
```

```
<p class="date">May, 2009</p>

<p class="main">In my younger and more vulnerable years my father gave me
some advice that I've been turning over in my mind ever since. 'Whenever
you feel like criticizing anyone,' he told me, just remember that all the
people in this world haven't had the advantages that you've had.'</p>

</body>

</html>
```

Note: Try to resize the browser window to see how justify works.

3. Text Decoration

The text-decoration property is used to set or remove decorations from text.

The text-decoration property is mostly used to remove underlines from links for design purposes:

Example:

```
<style type="text/css">

h1 {text-decoration:overline}

h2 {text-decoration:line-through}

h3 {text-decoration:underline}

h4 {text-decoration:blink}

</style>
```

Note: The "blink" value is not supported in IE, Chrome, or Safari.

4. Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

Example:

```
<html>
<head>
<style type="text/css">
```

```
p.uppercase {text-transform:uppercase}
p.lowercase {text-transform:lowercase}
p.capitalize {text-transform:capitalize}
</style>
</head>

<body>
<p class="uppercase">This is some text in a paragraph</p>
<p class="lowercase">This is some text in a paragraph</p>
<p class="capitalize">This is some text in a paragraph</p>
</body>
</html>
```

5. **Text Indentation**

The text-indentation property is used to specify the indentation of the first line of a text.

Example:

```
<html>

<head>
<style type="text/css">
      p {text-indent:50px}
</style>
</head>

<body>
<p>In my younger and more vulnerable years my father gave me some advice
that I've been turning over in my mind ever since.</p>
</body>
</html>
```

6. **Specify the space between characters**

This property is used to specify the space between characters. Negative value will decrease the space while positive value will increase the space.

Example:

```
<html>

<head>
<style type="text/css">
h1 {letter-spacing: -3px}
h4 {letter-spacing: 0.5cm}
</style>
</head>

<body>
<h1>This is header 1</h1>
```

```
<h4>This is header 4</h4>
</body>
</html>
```

7. **Specify the space between lines**

This property is used to specify the space between lines. Negative value will decrease the space while positive value will increase the space.

Example:

```
<html>
<head>
<style type="text/css">
p.small {line-height: 90%}
p.big {line-height: 20px}
</style>
</head>

<body>
<p>
This is a paragraph with a standard line-height.
This is a paragraph with a standard line-height.
This is a paragraph with a standard line-height.
</p>

<p class="small">
This is a paragraph with a smaller line-height.
This is a paragraph with a smaller line-height.
This is a paragraph with a smaller line-height.
This is a paragraph with a smaller line-height.
</p>

<p class="big">
This is a paragraph with a bigger line-height.
This is a paragraph with a bigger line-height.
This is a paragraph with a bigger line-height.
This is a paragraph with a bigger line-height.
</p>

</body>
</html>
```

Note: **line-height** can be given in % and pixel. The default line height in most browsers is about 110% to 120%.

8. **Text direction of an element**

This property is used to specify the direction of an element.

ltr: Text will start from left side.

rtl: Text will start from left side.

Example:

```
<html>
<head>
<style type="text/css">
div.one
{
direction:rtl;
}
div.two
{
direction:ltr;
}
</style>
</head>

<body>
<div class="one">Some text. Right-to-left direction.</div>
<div class="two">Some text. Left-to-right direction.</div>
</body>
</html>
```

9. Increase the white space between words

This property is used to specify the white space between words. Positive value increases the space while negative value decreases the space.

Example:

```
<html>
<head>
<style type="text/css">
p
{
word-spacing:30px;
}
</style>
</head>
<body>

<p>
This is some text. This is some text.
</p>

</body>
</html>
```

10. Text wrapping

White-space: nowrap – text will not wrap horizontal scroll will be visible.

White-space: wrap – text will wrap horizontal scroll will not be visible.

Example:

```
<html>
<head>
<style type="text/css">
p
{
white-space:nowrap;
}
</style>
</head>
<body>

<p>
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>

</body>
</html>
<style type="text/css">
p
{
white-space:nowrap;
}
</style>
```

CSS Font

1. **Font Family**

The font family of a text is set with the font-family property.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like font-family: "Times New Roman".

More than one font family is specified in a comma-separated list:

Example:

```
p{font-family:"Times New Roman",Georgia,Serif}
```

2. **Font Style**

The font-style property is mostly used to specify italic text.

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example:

```
<html>

<head>

<style type="text/css">

p.normal {font-style:normal}

p.italic {font-style:italic}

p.oblique {font-style:oblique}

</style>

</head>

<body>

<p class="normal">This is a paragraph, normal.</p>

<p class="italic">This is a paragraph, italic.</p>

<p class="oblique">This is a paragraph, oblique.</p>

</body>

</html>
```

3. **Font Size**

The font-size property sets the size of the text.

The font-size value can be an absolute, or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px **(16px=1em)**.

Example:

```
<html>

<head>

<style>

h1 {font-size:40px}

h2 {font-size:30px}

p {font-size:14px}

</style>

</head>

<body>

<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

<p>This is a paragraph.</p>

</body>

</html>
```

4. **Why font Size in em ?**

To avoid the resizing problem with Internet Explorer, many developers use em instead of pixels.

The em size unit is recommended by the W3C.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: *pixels/16=em*

Example:

```
h1 {font-size:2.5em} /* 40px/16=2.5em */
```

5. **Font Weight**

Sets the weight of a font.

Values: normal,bold,bolder,lighter,100,200,300,400,500,600,700,800,900

Example:

```
<html>

<head>

<style type="text/css">

p.normal {font-weight:normal}

p.thick {font-weight:bold}

p.thicker {font-weight:900}

</style>

</head>

<body>

<p class="normal">This is a paragraph.</p>

<p class="thick">This is a paragraph.</p>

<p class="thicker">This is a paragraph.</p>
```

```
</body>

</html>
```

6. **Font Variant**

Displays text in a small-caps font or a normal font.

Example:

```
<html>

<head>

<style type="text/css">

p.normal {font-variant:normal}

p.small {font-variant:small-caps}

</style>

</head>

<body>

<p class="normal">This is a paragraph.</p>

<p class="small">This is a paragraph.</p>

</body>

</html>
```

CSS Units

Measurements

Unit	Description
%	percentage
in	inch
cm	centimeter
mm	millimeter
em	1em is equal to the current font size. 2em means 2 times the size of the current font. E.g., if an element is displayed with a font of 12 pt, then '2em' is 24 pt. The 'em' is a very useful unit in CSS, since it can adapt automatically to the font that the reader uses
ex	one ex is the x-height of a font (x-height is usually about half the font-size)

pt	point (1 pt is the same as 1/72 inch)
pc	pica (1 pc is the same as 12 points)
px	pixels (a dot on the computer screen)

Colors

Unit	Description
color name	A color name (e.g. red)
rgb(x,x,x)	An RGB value (e.g. rgb(255,0,0))
rgb(x%, x%, x%)	An RGB percentage value (e.g. rgb(100%,0%,0%))
#rrggbb	A HEX number (e.g. #ff0000)

CSS Tables

1. **Set the layout of a table**

Example:

```
<html>

<head>

<style type="text/css">

table.ex1 {table-layout:auto}

table.ex2 {table-layout:fixed}

</style>

</head>

<body>

<table class="ex1" border="1" width="100%">

<tr>

<td width="5%">1000000000000000000000000000</td>

<td width="95%">10000000</td>

</tr>

</table>

<br />

<table class="ex2" border="1" width="100%">
```

```
<tr>

<td width="5%">100000000000000000000000000000</td>

<td width="95%">10000000</td>

</tr>

</table>

</body>

</html>
```

2. **Collapse a table border**

Sets whether the table borders are collapsed into a single border or detached as in standard HTML

Example:

```
<html>

<head>

<style type="text/css">

table {border-collapse:collapse}

</style>

</head>

<body>

<table border="1">

<tr>

<td>Peter</td>

<td>Griffin</td>

</tr>

<tr>

<td>Lois</td>
```

```
<td>Griffin</td>

</tr>

</table>

</body>

</html>
```

3. **Empty cells in a table border**

Sets whether or not to show empty cells in a table (only for the "separated borders" model)

Example:

```
<html>

<head>

<style type="text/css">

table

{

border-collapse:separate;

empty-cells:hide;

}

</style>

</head>

<body>

<table border="1">

<tr>

<td>Peter</td>

<td>Griffin</td>

</tr>
```

```
<tr>

<td>Lois</td>

<td></td>

</tr>

</table>

</body>

</html>
```

CSS Image Opacity

Note: This is not yet a CSS standard. However, it works in all modern browsers, and is a part of the W3C CSS 3 recommendation.

1. **Transparent Image**

<img src="klematis.jpg" width="150" height="113" alt="klematis" style="opacity:0.4;filter:alpha(opacity=40)" />

Firefox uses the property **opacity:x** for transparency, while IE uses **filter:alpha(opacity=x)**.

Tip: The CSS3 syntax for transparency is **opacity:x**.

In Firefox (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.

2. **Transparent Image - Mouseover effect**

<img src="klematis.jpg" style="opacity:0.4;filter:alpha(opacity=40)" onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100" onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />

The syntax for this in Firefox is: **this.style.opacity=1** and the syntax in IE is: **this.filters.alpha.opacity=100**.

When the mouse pointer moves away from the image, we want the image to be transparent again. This is done in the onmouseout attribute.

CSS Cursor

This property displays different types of cursor by moving on the text.

Example:

```
<html>

<body>

<p>Mouse over the words to change the cursor.</p>

<span style="cursor:auto">auto</span><br />

<span style="cursor:crosshair">crosshair</span><br />

<span style="cursor:default">default</span><br />

<span style="cursor:e-resize">e-resize</span><br />

<span style="cursor:help">help</span><br />

<span style="cursor:move">move</span><br />

<span style="cursor:n-resize">n-resize</span><br />

<span style="cursor:ne-resize">ne-resize</span><br />

<span style="cursor:nw-resize">nw-resize</span><br />

<span style="cursor:pointer">pointer</span><br />

<span style="cursor:progress">progress</span><br />

<span style="cursor:s-resize">s-resize</span><br />

<span style="cursor:se-resize">se-resize</span><br />

<span style="cursor:sw-resize">sw-resize</span><br />

<span style="cursor:text">text</span><br />

<span style="cursor:w-resize">w-resize</span><br />

<span style="cursor:wait">wait</span><br />

</body>

</html>
```

CSS Invisible Tags

Notice that the invisible heading still takes up space. Marked(**) statement will not be dispalyed

Example:

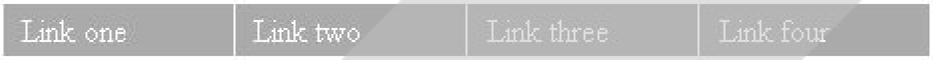
```
<style type="text/css">

h1.visible {visibility:visible}

h1.hidden {visibility:hidden}

</style>

<body>

<h1 class="visible">This is a visible heading</h1>

**<h1 class="hidden">This is an invisible heading</h1>

</body>
```

CSS Horizontal Menu

Example:

```
<html>

<head>

<style type="text/css">

ul

{

float:left;

width:100%;

padding:0;

margin:0;

list-style-type:none;
}
a
{
float:left;
width:6em;
text-decoration:none;
color:white;
background-color:#aaaaaa;
padding:0.2em 0.6em;
border-right:1px solid white;
}
a:hover {background-color:#ff3300}
li {display:inline}
</style>
</head>
<body>
<ul>
<li><a href="#">Link one</a></li>
<li><a href="#">Link two</a></li>
<li><a href="#">Link three</a></li>
<li><a href="#">Link four</a></li>
</ul>
</body>
</html>
```

Output:



CSS Margin

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent.

The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

1.Possible Values

Value	Description
auto	The browser sets the margin. The result of this is dependant of the browser
length	Defines a fixed margin (in pixels, pt, em, etc.)
%	Defines a margin in % of the containing element

It is possible to use negative values, to overlap content.

In CSS, it is possible to specify different margins for different sides:

Example:

```
margin-top:100px;
margin-bottom:100px;
margin-right:50px;
margin-left:50px;
```

1.Margin Shorthand Property

To shorten the code, it is possible to specify all the margin properties in one property. This is called a shorthand property.

The shorthand property for all the margin properties is "margin":

The margin property can have from one to four values.

- **margin:25px 50px 75px 100px;**
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px

- **margin:25px 50px 75px;**
  - top margin is 25px

  - right and left margins are 50px
  - bottom margin is 75px

- **margin:25px 50px;**
  - top and bottom margins are 25px
  - right and left margins are 50px

- **margin:25px;**
  - all four margins are 25px

Note: Margin can be set in cm and %.

CSS Border

1. Border Style

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

None of the other border properties will have any effect unless border-style is set.

border-style Values

none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

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

Example:

```
<style type="text/css">

p.none {border-style:none}

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.hidden {border-style:hidden}

</style>
```

2. <u>Border width</u>

The border-width property is used to set the width of the border.

The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example:

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

3. Border color

The border-color property is used to set the color of the border. The color can be set by:

- name - specify a color name, like "red"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- Hex - specify a hex value, like "#ff0000"

You can also set the border color to "transparent".

Note: The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example:

```
p.one
{
border-style:solid;
border-color:red;
}
p.two
{
border-style:solid;
border-color:#98bf21;
}
```

3. Border Individual Side

In CSS it is possible to specify different borders for different sides:

Example:

```
p
{
border-top-style:dotted;
border-right-style:solid;
border-bottom-style:dotted;
border-left-style:solid;
}
```

The border-style property can have from one to four values.

- **border-style:dotted solid double dashed;**

  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed

- **border-style:dotted solid double;**
  - top border is dotted
  - right and left borders are solid
  - bottom border is double

- **border-style:dotted solid;**
  - top and bottom borders are dotted
  - right and left borders are solid

- **border-style:dotted;**
  - all four borders are dotted

The border-style property is used in the example above. However, it also works with border-width and border-color.

Note: When using the border property, the orders of the values are:

- border-width
- border-style
- border-color

It does not matter if one of the values above are missing (although, border-style is required), as long as the rest are in the specified order.

All CSS Border Properties

The number in the "CSS" column indicates in which CSS version the property is defined (CSS1 or CSS2).

Property	**Description**	**Values**	**CSS**
border	Sets all the border properties in one declaration	*border-width* *border-style* *border-color*	1
border-bottom	Sets all the bottom border properties in one declaration	*border-bottom-width* *border-bottom-style* *border-bottom-color*	1
border-bottom-color	Sets the color of the bottom border	*border-color*	2
border-bottom-style	Sets the style of the bottom border	*border-style*	2

border-bottom-width	Sets the width of the bottom border	*border-width*	1
border-color	Sets the color of the four borders	*color_name* *hex_number* *rgb_number* transparent inherit	1
border-left	Sets all the left border properties in one declaration	*border-left-width* *border-left-style* *border-left-color*	1
border-left-color	Sets the color of the left border	*border-color*	2
border-left-style	Sets the style of the left border	*border-style*	2
border-left-width	Sets the width of the left border	*border-width*	1
border-right	Sets all the right border properties in one declaration	*border-right-width* *border-right-style* *border-right-color*	1
border-right-color	Sets the color of the right border	*border-color*	2
border-right-style	Sets the style of the right border	*border-style*	2
border-right-width	Sets the width of the right border	*border-width*	1
border-style	Sets the style of the four borders	none hidden dotted dashed solid double groove ridge inset outset inherit	1
border-top	Sets all the top border properties in one declaration	*border-top-width* *border-top-style* *border-top-color*	1
border-top-color	Sets the color of the top border	*border-color*	2
border-top-style	Sets the style of the top border	*border-style*	2
border-top-width	Sets the width of the top border	*border-width*	1
border-width	Sets the width of the four borders	thin medium thick *length* inherit	1

CSS Padding

The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element.

The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all paddings at once.

Possible Values

Value	Description
length	Defines a fixed padding (in pixels, pt, em, etc.)
%	Defines a padding in % of the containing element

Padding - Individual sides

In CSS, it is possible to specify different padding for different sides:

Example

```
padding-top:25px;
padding-bottom:25px;
padding-right:50px;
padding-left:50px;
```

Padding - Shorthand property

To shorten the code, it is possible to specify all the padding properties in one property. This is called a shorthand property.

The shorthand property for all the padding properties is "padding":

Example

```
padding:25px 50px;
```

The margin property can have from one to four values.

- **padding:25px 50px 75px 100px;**
  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px
  - left padding is 100px
- **padding:25px 50px 75px;**
  - top padding is 25px
  - right and left paddings are 50px
  - bottom padding is 75px
- **padding:25px 50px;**

   - top and bottom paddings are 25px
   - right and left paddings are 50px

- **padding:25px;**
   - all four paddings are 25px

CSS Pseudo-classes

CSS pseudo-classes are used to add special effects to some selectors.

Syntax

The syntax of pseudo-classes:

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

CSS classes can also be used with pseudo-classes:

```
selector.class:pseudo-class {property:value}
```

Anchor Pseudo-classes

Links can be displayed in different ways in a CSS-supporting browser:

```
a:link {color:#FF0000} /* unvisited link */
a:visited {color:#00FF00} /* visited link */
a:hover {color:#FF00FF} /* mouse over link */
a:active {color:#0000FF} /* selected link */
```

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

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

Note: Pseudo-class names are not case-sensitive.

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

```
a.red:visited {color:#FF0000}

<a class="red" href="css_syntax.asp">CSS Syntax</a>
```

If the link in the example above has been visited, it will be displayed in red.

CSS - The :first-child Pseudo-class

The :first-child pseudo-class matches a specified element that is the first child of another element.

Note: For :first-child to work in IE a <!DOCTYPE> must be declared.

Match the first <p> element

In the following example, the selector matches any <p> element that is the first child of any element:

Example

```
<html>
<head>
<style type="text/css">
p:first-child
{
color:blue
}
</style>
</head>

<body>
<p>I am a strong man.</p>
<p>I am a strong man.</p>
</body>
</html>
```

Match the first <i> element in all <p> elements

In the following example, the selector matches the first <i> element in all <p> elements:

Example

```
<html>
<head>
<style type="text/css">
p > i:first-child
{
font-weight:bold
}
</style>
</head>

<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
</body>
</html>
```

Match all <i> elements in all first child <p> elements

In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element:

Example

```
<html>
<head>
<style type="text/css">
p:first-child i
{
color:blue
}
</style>
</head>

<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
</body>
</html>
```

CSS - The :lang Pseudo-class

The :lang pseudo-class allows you to define special rules for different languages. In the example below, the :lang class defines the type of quotation marks for q elements with a lang attribute with a value of "no":

```
<html>

<head>
<style type="text/css">
q:lang(no)
{
quotes:"~" "~"
}
</style>
</head>

<body>
<p>Some text <q lang="no">A quote in a paragraph</q>
Some text.</p>
</body>

</html>
```

CSS Pseudo-elements

CSS pseudo-elements are used to add special effects to some selectors.

Syntax

The syntax of pseudo-elements:

```
selector:pseudo-element {property:value}
```

CSS classes can also be used with pseudo-elements:

```
selector.class:pseudo-element {property:value}
```

The :first-line Pseudo-element

The "first-line" pseudo-element is used to add special styles to the first line of the text in a selector:

```
p:first-line {color:#0000ff;font-variant:small-caps}

<p>Some text that ends up on two or more lines</p>
```

The output could be something like this:

```
Some text that ends
up on two or more lines
```

In the example above the browser displays the first line formatted according to the "first-line" pseudo element. **Where** the browser breaks the line depends on the size of the browser window.

Note: The "first-line" pseudo-element can only be used with block-level elements.

Note: The following properties apply to the "first-line" pseudo-element:;

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

The :first-letter Pseudo-element

The "first-letter" pseudo-element is used to add special style to the first letter of the text in a selector:

```
p:first-letter {color:#ff0000;font-size:xx-large}

<p>The first words of an article...</p>
```

The output could be something like this:

```
he first words of an article...
```

Note: The "first-letter" pseudo-element can only be used with block-level elements.

Note: The following properties apply to the "first-letter" pseudo-element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes:

```
p.article:first-letter {color:#ff0000}

<p class="article">A paragraph in an article</p>
```

The example above will make the first letter of all paragraphs with class="article" red.

Multiple Pseudo-elements

Several pseudo-elements can be combined:

```
p:first-letter {color:#ff0000;font-size:xx-large}
p:first-line {color:#0000ff}

<p>The first words of an article...</p>
```

The output could be something like this:

```
he first
words of an
article...
```

In the example above the first letter of the paragraph will be red with a font size of 24pt. The rest of the first line would be blue while the rest of the paragraph would be the default color.

CSS - The :before Pseudo-element

The ":before" pseudo-element can be used to insert some content before the content of an element.

The style below will play a sound before each occurrence of an <h1> element:

```
h1:before
{
content:url(beep.wav)
}
```

CSS - The :after Pseudo-element

The ":after" pseudo-element can be used to insert some content after the content of an element.

The style below will play a sound after each occurrence of an <h1> element:

```
h1:after
{
content:url(beep.wav)
}
```

JAVASCRIPT

INDEX

Topics **Page No**

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Are Java and JavaScript the same?

NO!

Java and JavaScript are two completely different languages in both concept and design!

Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

JavaScript was created by Netscape to extend the functions a browser can perform.

What can a JavaScript do?

- **JavaScript gives HTML designers a programming tool** - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- **JavaScript can put dynamic text into an HTML page** - A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page
- **JavaScript can react to events** - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements** - A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data** - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and -

depending on the browser - load another page specifically designed for that browser

- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

The HTML <script> tag is used to insert a JavaScript into an HTML page.

The example below shows how to use JavaSript to write text on a web page:

Example

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!");
</script>
</body>
</html>
```

Where To Implement JavaScript

JavaScripts in the body section will be executed WHILE the page loads.

JavaScripts in the head section will be executed when CALLED.

Where to Put the JavaScript

JavaScripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

Scripts in <head>

Scripts to be executed when they are called, or when an event is triggered, go in the head section.

If you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

Example

```
<html>
<head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload event");
}
</script>
</head>
```

```
<body onload="message()">
</body>
</html>
```

Scripts in <body>

Scripts to be executed when the page loads go in the body section.

If you place a script in the body section, it generates the content of a page.

Example

```
<html>
<head>
</head>

<body>
<script type="text/javascript">
document.write("This message is written by JavaScript");
</script>
</body>

</html>
```

Scripts in <head> and <body>

You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
<body>
<script type="text/javascript">
....
</script>
</body>
```

Using an External JavaScript

If you want to run the same JavaScript on several pages, without having to write the same script on every page, you can write a JavaScript in an external file.

Save the external JavaScript file with a .js file extension.

Note: The external script cannot contain the <script> tag!

To use the external script, point to the .js file in the "src" attribute of the <script> tag:

Example

```
<html>
<head>
<script type="text/javascript" src="xxx.js"></script>
</head>
<body>
</body>
</html>
```

Comments In Javascript

JavaScript comments can be used to make the code more readable.

Comments can be added to explain the JavaScript, or to make the code more readable.

Single line comments start with //

The following example uses single line comments to explain the code:

Example

```
<script type="text/javascript">
// Write a heading
document.write("<h1>This is a heading</h1>");
// Write two paragraphs:
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Multi-Line Comments

Multi line comments start with /* and end with */.

The following example uses a multi line comment to explain the code:

Example

```
<script type="text/javascript">
/*
The code below will write
one heading and two paragraphs
*/
```

```
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

Using Comments to Prevent Execution

In the following example the comment is used to prevent the execution of a single code line (can be suitable for debugging):

Example

```
<script type="text/javascript">
//document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

In the following example the comment is used to prevent the execution of a code block (can be suitable for debugging):

Example

```
<script type="text/javascript">
/*
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
*/
</script>
```

Using Comments at the End of a Line

In the following example the comment is placed at the end of a code line:

Example

```
<script type="text/javascript">
document.write("Hello"); // Write "Hello"
document.write(" Dolly!"); // Write " Dolly!"
</script>
```

JavaScript Variables

Variables are "containers" for storing information.

Do you remember algebra from school?

x=5, y=6, z=x+y

Do you remember that a letter (like x) could be used to hold a value (like 5), and that you could use the information above to calculate the value of z to be 11?

These letters are called **variables**, and variables can be used to hold values (x=5) or expressions (z=x+y).

JavaScript Variables

As with algebra, JavaScript variables are used to hold values or expressions.

A variable can have a short name, like x, or a more descriptive name, like carname.

Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter or the underscore character

Note: Because JavaScript is case-sensitive, variable names are case-sensitive.

Example

A variable's value can change during the execution of a script. You can refer to a variable by its name to display or change its value.

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the **var statement:**

```
var x;
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).

However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Volvo";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname**will hold the value **Volvo**.

Note: When you assign a text value to a variable, use quotes around the value.

Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that have not yet been declared, the variables will automatically be declared.

These statements:

```
x=5;
carname="Volvo";
```

have the same effect as:

```
var x=5;
var carname="Volvo";
```

Redeclaring JavaScript Variables

If you redeclare a JavaScript variable, it will not lose its original value.

```
var x=5;
var x;
```

After the execution of the statements above, the variable x will still have the value of 5. The value of x is not reset (or cleared) when you redeclare it.

JavaScript Operators

Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6

--	Decrement	x=--y	x=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".

To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

Adding Strings and Numbers

The rule is: **If you add a number and a string, the result will be a string!**

Example

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for true or false.

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

How can it be used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age<18) document.write("Too young");
```

Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true

||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

```
variablename=(condition)?value1:value2
```

Example

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

If the variable **visitor** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President " else it will be assigned "Dear".

Conditional Statements

Conditional statements are used to perform different actions based on different conditions.

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement to select one of many blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

If Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

```
if (condition)
  {
  code to be executed if condition is true
  }
```

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

Example

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10

var d=new Date();
var time=d.getHours();

if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
</script>
```

Notice that there is no ..else.. in this syntax. You tell the browser to execute some code only if the specified condition is true.

If...else Statement

Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
  {
  code to be executed if condition is true
  }
else
  {
  code to be executed if condition is not true
  }
```

Example

```
<script type="text/javascript">
//If the time is less than 10, you will get a "Good morning" greeting.
//Otherwise you will get a "Good day" greeting.

var d = new Date();
var time = d.getHours();

if (time < 10)
  {
  document.write("Good morning!");
  }
else
  {
  document.write("Good day!");
  }
</script>
```

If...else if...else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

Syntax

```
if (condition1)
  {
  code to be executed if condition1 is true
  }
else if (condition2)
  {
  code to be executed if condition2 is true
  }
else
  {
  code to be executed if condition1 and condition2 are not true
  }
```

Example

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
else if (time>10 && time<16)
  {
  document.write("<b>Good day</b>");
  }
else
  {
  document.write("<b>Hello World!</b>");
  }
</script>
```

JavaScript Switch Statement

Conditional statements are used to perform different actions based on different conditions.

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is different from case 1 and 2
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use**break** to prevent the code from running into the next case automatically.

Example

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.

var d=new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("Finally Friday");
  break;
case 6:
  document.write("Super Saturday");
  break;
case 0:
  document.write("Sleepy Sunday");
  break;
default:
  document.write("I'm looking forward to this weekend!");
}
</script>
```

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
alert("sometext");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_alert()
```

```
{
alert("I am an alert box!");
}
</script>
</head>
<body>

<input type="button" onclick="show_alert()" value="Show alert box" />

</body>
</html>
```

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
confirm("sometext");
```

Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
prompt("sometext","defaultvalue");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
```

```
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
  document.write("Hello " + name + "! How are you today?");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_prompt()" value="Show prompt box" />

</body>
</html>
```

JavaScript Functions

A function will be executed by an event or by a call to the function.

To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed by an event or by a call to the function.

You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax

```
function functionname(var1,var2,...,varX)
{
some code
}
```

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a

JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

JavaScript Function Example

Example

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");
}
</script>
</head>

<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```

If the line: alert("Hello world!!") in the example above had not been put within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before a user hits the input button. The function displaymessage() will be executed if the input button is clicked.

The return Statement

The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

The example below returns the product of two numbers (a and b):

Example

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>

<body>
<script type="text/javascript">
```

```
document.write(product(4,3));
</script>

</body>
</html>
```

The Lifetime of JavaScript Variables

If you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

JavaScript Loop

Loops execute a block of code a specified number of times, or while a specified condition is true.

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
code to be executed
}
```

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as **i**is less than, or equal to 5. **i** will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

JavaScript While Loop

Loops execute a block of code a specified number of times, or while a specified condition is true.

The while Loop

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

Syntax

```
while (var<=endvalue)
  {
  code to be executed
  }
```

Note: The <= could be any comparing statement.

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as **i**is less than, or equal to 5. **i** will increase by 1 each time the loop runs:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
</script>
</body>
</html>
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
do
  {
  code to be executed
  }
while (var<=endvalue);
```

Example

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
while (i<=5);
</script>
</body>
</html>
```

The break Statement

The break statement will break the loop and continue executing the code that follows after the loop (if any).

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    break;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
</script>
</body>
</html>
```

The continue Statement

The continue statement will break the current loop and continue with the next value.

Example

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    continue;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
</script>
</body>
</html>
```

JavaScript For...In Statement

The for...in statement loops through the elements of an array or through the properties of an object.

Syntax

```
for (variable in object)
  {
  code to be executed
  }
```

Note: The code in the body of the for...in loop is executed once for each element/property.

Note: The variable argument can be a named variable, an array element, or a property of an object.

Example

Use the for...in statement to loop through an array:

Example

```
<html>
<body>

<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";

for (x in mycars)
  {
  document.write(mycars[x] + "<br />");
  }
</script>

</body>
</html>
```

JavaScript Events

Events are actions that can be detected by JavaScript.

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

onLoad and onUnload

The onLoad and onUnload events are triggered when the user enters or leaves the page.

The onLoad event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

Both the onLoad and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

onFocus,onClick onBlur and onChange

The onFocus, onBlur and onChange events are often used in combination with validation of form fields.

Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```

onSubmit

The onSubmit event is used to validate ALL form fields before submitting it.

Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post" action="xxx.htm" onsubmit="return checkForm()">
```

onMouseOver and onMouseOut

onMouseOver and onMouseOut are often used to create "animated" buttons.

Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com" onmouseover="alert('An onMouseOver
event');return false"><img src="w3s.gif" alt="W3Schools" /></a>
```

JavaScript Special Characters

In JavaScript you can add special characters to a text string by using the backslash sign.

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

Look at the following JavaScript code:

```
var txt="We are the so-called "Vikings" from the north.";
document.write(txt);
```

In JavaScript, a string is started and stopped with either single or double quotes. This means that the string above will be chopped to: We are the so-called

To solve this problem, you must place a backslash (\) before each double quote in "Viking". This turns each double quote into a string literal:

```
var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);
```

JavaScript will now output the proper text string: We are the so-called "Vikings" from the north.

Here is another example:

```
document.write ("You \& I are singing!");
```

The example above will produce the following output:

```
You & I are singing!
```

The table below lists other special characters that can be added to a text string with the backslash sign:

Code	Outputs
\'	single quote
\"	double quote
\&	ampersand
\\	backslash
\n	new line
\r	carriage return
\t	tab
\b	backspace
\f	form feed

JavaScript String Object

The String object is used to manipulate a stored piece of text.

The String object is used to manipulate a stored piece of text.

Examples of use:

The following example uses the length property of the String object to find the length of a string:

```
var txt="Hello world!";
document.write(txt.length);
```

The code above will result in the following output:

```
12
```

The following example uses the toUpperCase() method of the String object to convert a string to uppercase letters:

```
var txt="Hello world!";
document.write(txt.toUpperCase());
```

The code above will result in the following output:

```
HELLO WORLD!
```

JavaScript Array Object

What is an Array?

An array is a special variable, which can hold more than one value, at a time.

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

```
$cars1="Saab";
$cars2="Volvo";
$cars3="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 best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Each element in the array has its own ID so that it can be easily accessed.

Create an Array

The following code creates an Array object called myCars:

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

There are two ways of adding values to an array (you can add as many values as you need to define as many variables you require).

1:

```
var myCars=new Array();
myCars[0]="Saab";
myCars[1]="Volvo";
myCars[2]="BMW";
```

You could also pass an integer argument to control the array's size:

```
var myCars=new Array(3);
myCars[0]="Saab";
myCars[1]="Volvo";
myCars[2]="BMW";
```

2:

```
var myCars=new Array("Saab","Volvo","BMW");
```

Note: If you specify numbers or true/false values inside the array then the type of variables will be numeric or Boolean instead of string.

Access an Array

You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.

The following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Saab
```

Modify Values in an Array

To modify a value in an existing array, just add a new value to the array with a specified index number:

```
myCars[0]="Opel";
```

Now, the following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Opel
```

JavaScript Timing Events

With JavaScript, it is possible to execute some code after a specified time-interval. This is called timing events.

It's very easy to time events in JavaScript. The two key methods that are used are:

- setTimeout() - executes a code some time in the future
- clearTimeout() - cancels the setTimeout()

Note: The setTimeout() and clearTimeout() are both methods of the HTML DOM Window object.

The setTimeout() Method

Syntax

```
var t=setTimeout("javascript statement",milliseconds);
```

The setTimeout() method returns a value - In the statement above, the value is stored in a variable called t. If you want to cancel this setTimeout(), you can refer to it using the variable name.

The first parameter of setTimeout() is a string that contains a JavaScript statement. This statement could be a statement like "alert('5 seconds!')" or a call to a function, like "alertMsg()".

The second parameter indicates how many milliseconds from now you want to execute the first parameter.

Note: There are 1000 milliseconds in one second.

Example

When the button is clicked in the example below, an alert box will be displayed after 5 seconds.

Example

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
{
var t=setTimeout("alert('5 seconds!')",5000);
}
</script>
</head>

<body>
<form>
<input type="button" value="Display timed alertbox!"
onClick="timedMsg()" />
</form>
```

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

Example - Infinite Loop

To get a timer to work in an infinite loop, we must write a function that calls itself. In the example below, when the button is clicked, the input field will start to count (for ever), starting at 0:

Example

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c;
c=c+1;
t=setTimeout("timedCount()",1000);
}
</script>
</head>

<body>
<form>
<input type="button" value="Start count!" onClick="timedCount()" />
<input type="text" id="txt" />
</form>
</body>

</html>
```

The clearTimeout() Method

Syntax

```
clearTimeout(setTimeout_variable)
```

Example

The example below is the same as the "Infinite Loop" example above. The only difference is that we have now added a "Stop Count!" button that stops the timer:

Example

```
<html>
<head>
<script type="text/javascript">
var c=0
var t

function timedCount()
{
document.getElementById('txt').value=c;
c=c+1;
t=setTimeout("timedCount()",1000);
}

function stopCount()
{
clearTimeout(t);
}
</script>
</head>

<body>
<form>
<input type="button" value="Start count!" onClick="timedCount()" />
<input type="text" id="txt" />
<input type="button" value="Stop count!" onClick="stopCount()" />
</form>
</body>

</html>
```

JavaScript Form Validation

JavaScript can be used to validate data in HTML forms before sending off the content to a server.

Form data that typically are checked by a JavaScript could be:

- has the user left required fields empty?
- has the user entered a valid e-mail address?
- has the user entered a valid date?
- has the user entered text in a numeric field?

To check a empty field

```
var field_val=document.getElementById('fieldid').value;

if(field_val=="")
{
      alert("The Field is Empty");
}
else
{
      alert("The Field is not Empty");
}
```

freelance_Project available to buy contact on 8007592194

SR.NO	Project NAME	Technology
1	Online E-Learning Platform Hub	React+Springboot+MySql
2	PG Mates / RoomSharing / Flat Mates	React+Springboot+MySql
3	Tour and Travel management System	React+Springboot+MySql
4	Election commition of India (online Voting System)	React+Springboot+MySql
5	HomeRental Booking System	React+Springboot+MySql
6	Event Management System	React+Springboot+MySql
7	Hotel Management System	React+Springboot+MySql
8	Agriculture web Project	React+Springboot+MySql
9	AirLine Reservation System / Flight booking System	React+Springboot+MySql
10	E-commerce web Project	React+Springboot+MySql
11	Hospital Management System	React+Springboot+MySql
12	E-RTO Driving licence portal	React+Springboot+MySql
13	Transpotation Services portal	React+Springboot+MySql
14	Courier Services Portal / Courier Management System	React+Springboot+MySql
15	Online Food Delivery Portal	React+Springboot+MySql
16	Muncipal Corporation Management	React+Springboot+MySql
17	Gym Management System	React+Springboot+MySql
18	Bike/Car ental System Portal	React+Springboot+MySql
19	CharityDonation web project	React+Springboot+MySql
20	Movie Booking System	React+Springboot+MySql

freelance_Project available to buy contact on 8007592194

21	Job Portal web project	React+Springboot+MySql
22	LIC Insurance Portal	React+Springboot+MySql
23	Employee Management System	React+Springboot+MySql
24	Payroll Management System	React+Springboot+MySql
25	RealEstate Property Project	React+Springboot+MySql
26	Marriage Hall Booking Project	React+Springboot+MySql
27	Online Student Management portal	React+Springboot+MySql
28	Resturant management System	React+Springboot+MySql
29	Solar Management Project	React+Springboot+MySql
30	OneStepService LinkLabourContractor	React+Springboot+MySql
31	Vehical Service Center Portal	React+Springboot+MySql
32	E-wallet Banking Project	React+Springboot+MySql
33	Blogg Application Project	React+Springboot+MySql
34	Car Parking booking Project	React+Springboot+MySql
35	OLA Cab Booking Portal	React+Springboot+MySql
36	Society management Portal	React+Springboot+MySql
37	E-College Portal	React+Springboot+MySql
38	FoodWaste Management Donate System	React+Springboot+MySql
39	Sports Ground Booking	React+Springboot+MySql
40	BloodBank mangement System	React+Springboot+MySql
41	Bus Tickit Booking Project	React+Springboot+MySql
42	Fruite Delivery Project	React+Springboot+MySql
43	Woodworks Bed Shop	React+Springboot+MySql
44	Online Dairy Product sell Project	React+Springboot+MySql
45	Online E-Pharma medicine sell Project	React+Springboot+MySql
46	FarmerMarketplace Web Project	React+Springboot+MySql
47	Online Cloth Store Project	React+Springboot+MySql
48		React+Springboot+MySql
49		React+Springboot+MySql
50		React+Springboot+MySql

https://www.youtube.com/@codewitharrays

https://www.instagram.com/codewitharrays/

https://t.me/codewitharrays Group Link: https://t.me/cceesept2023

+91 8007592194 +91 9284926333

codewitharrays@gmail.com

https://codewitharrays.in/project