Internet programming and web technologies (20A35502T)

**UNIT -2 HTml5**

**HTML5 – Text tags:**

HTML text content elements to organize blocks or sections of content placed between the opening <body>  and closing </body> tags.

| Element | Description |
| --- | --- |
| < blockquote> | The <blockquote> is used to distinguish quoted text from the rest of the content.  The cite attribute in the blockquote tag in HTML can be used to refer to the source of the text or content. It is an optional attribute. |
| <dl> | <dl> tag defines the description list. |
| <dt> | <dt> tag defines data term |
| <dd> | The <dd>  tag defines data definition |
| <ul> | An <ul>tag is suitable for elements that do not need to be presented numerically and which can be rearranged without changing the meaning of the list.   * type attribute specifies the type of numbering to provide for <ul> tag. |
| <ol> | The <ol> tag is for an ordered list, an ordered list can be numerical or alphabetical. Inside the <ol>tag you have to make a list<li>of items that will follow the order.   * type attribute specifies the type of numbering to provide for <ol> tag. * The start attribute is used with <ol> tag to specify from where to start the list items. |
| <div> | The <div> tags are generic containers that can hold flow content, which does not have a default rendering.It has no effect on the content or layout until styled in some way using CSS. |
| <figure> | The <figure>  element represents self-contained content, potentially with an optional caption, which is specified using the figcaption element. The figure, its caption, and its contents are referenced as a single unit. |
| <figcaption> | The <figcaption>  element represents a caption or legend describing the rest of the contents of its parent figure element. |
| <hr> | The <hr>  element represents a thematic break between paragraph-level elements: for example, a change of scene in a story, or a shift of topic within a section. |
| <p> | The <p>  element represents a paragraph. |
| <pre> | The <pre> element represents preformatted text which is to be presented exactly as written in the HTML file. |

**Examples:**

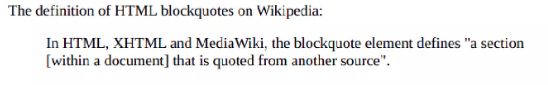
1.

<p>The definition of HTML blockquotes on Wikipedia:</p>

<blockquote cite="https://en.wikipedia.org/wiki/Blockquote\_element">

In HTML, XHTML and MediaWiki, the blockquote element defines "a section [within a document] that is quoted from another source".

</blockquote>

****

2. <dl>

<dt><b>Apple</b></dt>

<dd>A red colored fruit</dd>

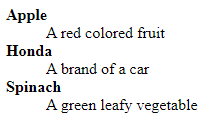
<dt><b>Honda</b></dt>

<dd>A brand of a car</dd>

<dt><b>Spinach</b></dt>

<dd>A green leafy vegetable</dd>

</dl>



3.<ol>

<li>Apple</li>

<li>Mango</li>

<li>Banana</li>

<li>Grapes</li>

<li>Orange</li>

</ol>

****

4**.** <ul>

<li>Apple</li>

<li>Mango</li>

<li>Banana</li>

<li>Grapes</li>

<li>Orange</li>

</ul>

****

**Inline text semantic**

Use the HTML inline text semantic to define the meaning, structure, or style of a word, line, or any arbitrary piece of text.

| **Formatting tag** | **Use** |
| --- | --- |
| <b> | It is used to write in bold. |
| <strong> | It is used to write in bold font with emphasizing the text. |
| <i> | It is used to write in italic font. |
| <em> | It is used to write in italic font with emphasizing the text. |
| <u> | It is used to underline a word. |
| <mark> | It is used to highlight a word with a yellow background. |
| <strike> | It is used to add a horizontal line through a word. |
| <tt> | It is used to write in monospace font. |
| <sub> | It is used to write subscripts of a word. |
| <sup> | It is used to write superscript of a word. |
| <del> | This tag is used to remove a word by adding a horizontal line through it. It is used with an insert tag. |
| <ins> | It is used to insert a word.HTML adds an underline to the added word it is used with the delete tag. |
| <big> | It is used to make the font one size bigger than the default size. |
| <small> | It is used to make the font one size smaller than the default size. |

Examples:

1. <p>Bold text = <b>Formatting tags</b> are very <strong>Helpful</strong></p>



2. <p>Italic text = <i>Formatting tags</i> are <em>BEST!</em></p>



3. <p>Underlined text = <u>Formatting tags</u></p>



4. <p>Highlighted text = <mark>Formatting tags</mark></p>



5. <p>Striked text = <s>Formatting tags</s></p>



6. <p>Monospace text = <tt>Formatting tags</tt></p>



7. <p>Subscript text = H<sub>2</sub>O



8. <p>Superscript text = x<sup>2</sup>+ y<sup>2</sup></p>



9. <p><del>HTML4</del> <ins>HTML5</ins> is the latest version of HTML!</p>



10. <p>This is an example of <small>small tag</small> and<big> big tag</big></p>



# Forms in HTML

HTML Forms are used to collect information submitted by the user, such as a user's name, email address, or any other essential information, and are sent to the server for processing the data. Forms in HTML contain a checkbox, radio button, input text fields, password fields etc., which can be seen on any website registration or Login/signup pages.

Forms in HTML help the user to enter certain details asked and sent them further to the server for processing.

Syntax:

# <form> . *form elements* . </form>

**Form attributes:**

1. action

2. method

3. autocomplete

4. name

5. target

#### action

After the submission of the form, the *action* attribute specifies the action to be taken, it specifies where the information should proceed. For example, it can be any URL that you prefer, like .php, .asp, etc. If it is left empty, then by default, it will be processed on the same page.

**Syntax**

**<form action = “file.html” method = “post”>**

#### method

The method attribute in the form instructs the browser on how to send data from the form to a web server. That is possible in two ways:

1. GET method
2. POST method

GET Method It is not secure as data is displayed in the URL. If the method is not specified, the value is set to "get" as default.

**Syntax**

<form action="file.html" method="get">

POST Method POST Methos is used to process sensitive data as the data is not displayed in the URL. Form data is sent as per the HTTP post-transaction.

**Syntax**

**<form action="file.html" method="post">**

#### autocomplete

If we want the user to enter the data manually or autocomplete the data by itself from the past entries, it can be done by Autocomplete on/off.  
**ON** - It is set as default, if not specified. If autocomplete is on, then the browser completes the data from the entries made by the user in the past.

#### name

The name attribute is used to reference form data after submission or to refer to items in JavaScript. It is also used to specify the name of the form.

#### target

The target attribute specifies where the response should be opened after the form has been submitted.  
It can be opened in two forms -

**\_self** - The current page will display the response.

**\_blank** - A new page will display the response.

**\_parent** - The parent frame will display the response.

**\_top** - The full page will display the response.

If there is no parent in \_parent and \_top they behave the same as \_self.

## HTML Form Controls

There are different types of form controls that you can use to collect data using HTML form –

* label
* Text Input Controls
* Checkboxes Controls
* Radio Box Controls
* Select Box Controls
* File Select boxes
* Hidden Controls
* Clickable Buttons
* Submit and Reset Button
* **Fieldset**
* **legend**

**<label>:**

It defines label for <form> elements.

## Text input controls:

There are three types of text input used on forms −

* Single-line text input controls − This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag.
* Password input controls − This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl <input> tag.
* Multi-line text input controls − This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

## Single-line text input controls

This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag.

### Example

## <form >

## <label> First name: <input type = "text" name = "first\_name" /></label>

## <br>

## Last name: <input type = "text" name = "last\_name" />

## </form>

## 

## Attributes

Following is the list of attributes for <input> tag for creating text field.

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **type**  Indicates the type of input control and for text input control it will be set to text. |
| 2 | **name**  Used to give a name to the control which is sent to the server to be recognized and get the value. |
| 3 | **value**  This can be used to provide an initial value inside the control. |
| 4 | **size**  Allows to specify the width of the text-input control in terms of characters. |
| 5 | **maxlength**  Allows to specify the maximum number of characters a user can enter into the text box. |

## Password input controls

This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input>tag but type attribute is set to password.

### Example

## <form >

## User ID : <input type = "text" name = "user\_id" />

## <br>

## Password: <input type = "password" name = "password" />

## </form>

## 

## Attributes

Following is the list of attributes for <input> tag for creating password field.

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **type**  Indicates the type of input control and for password input control it will be set to password. |
| 2 | **name**  Used to give a name to the control which is sent to the server to be recognized and get the value. |
| 3 | **value**  This can be used to provide an initial value inside the control. |
| 4 | **size**  Allows to specify the width of the text-input control in terms of characters. |
| 5 | **maxlength**  Allows to specify the maximum number of characters a user can enter into the text box. |

## Multiple-Line Text Input Controls

This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

### Example

## <form>

## Description : <br />

## <textarea rows = "5" cols = "50" name = "description">

## Enter description here...

## </textarea>

## </form>

## 

## Attributes

Following is the list of attributes for <textarea> tag.

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **name**  Used to give a name to the control which is sent to the server to be recognized and get the value. |
| 2 | **rows**  Indicates the number of rows of text area box. |
| 3 | **cols**  Indicates the number of columns of text area box |

## Checkbox Control:

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to checkbox**.**.

### Example

## <form>

## <input type = "checkbox" name = "maths" value = "on"> Maths

## <input type = "checkbox" name = "physics" value = "on"> Physics

## </form>

## 

## Attributes

Following is the list of attributes for <checkbox> tag.

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **type**  Indicates the type of input control and for checkbox input control it will be set to checkbox**.**. |
| 2 | **name**  Used to give a name to the control which is sent to the server to be recognized and get the value. |
| 3 | **value**  The value that will be used if the checkbox is selected. |
| 4 | **checked**  Set to checked if you want to select it by default. |

## Radio Button Control:

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to radio.

### Example

## <form>

## <input type = "radio" name = "subject" value = "maths"> Maths

## <input type = "radio" name = "subject" value = "physics"> Physics

## </form>

## 

## Attributes

Following is the list of attributes for radio button.

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **type**  Indicates the type of input control and for checkbox input control it will be set to radio. |
| 2 | **name**  Used to give a name to the control which is sent to the server to be recognized and get the value. |
| 3 | **value**  The value that will be used if the radio box is selected. |
| 4 | **checked**  Set to checked if you want to select it by default. |

## Select Box Control:

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

### Example

## <form>

## <select name = "dropdown">

## <option value = "Maths" selected>Maths</option>

## <option value = "Physics">Physics</option>

## </select>

## </form>

## 

**Attributes**

Following is the list of important attributes of <select> tag −

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **name**  Used to give a name to the control which is sent to the server to be recognized and get the value. |
| 2 | **size**  This can be used to present a scrolling list box. |
| 3 | **multiple**  If set to "multiple" then allows a user to select multiple items from the menu. |

Following is the list of important attributes of <option> tag −

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **value**  The value that will be used if an option in the select box box is selected. |
| 2 | **selected**  Specifies that this option should be the initially selected value when the page loads. |
| 3 | **label**  An alternative way of labeling options |

## File Upload Box:

### It allows a user to upload a file to a web site. It also known as a file select box. This is also created using the <input> element but type attribute is set to file.

### Example

## <form>

## <input type = "file" name = "fileupload" accept = "image/\*" />

## </form>

## 

## Attributes

Following is the list of important attributes of file upload box −

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **name**  Used to give a name to the control which is sent to the server to be recognized and get the value. |
| 2 | **accept**  Specifies the types of files that the server accepts. |

## Button Controls:

<input>tag is used to create a clickable button by setting its type attribute to button .

The type attribute can take the following values −

|  |  |
| --- | --- |
| **Sr.No** | **Type & Description** |
| 1 | **submit**  This creates a button that automatically submits a form. |
| 2 | **reset**  This creates a button that automatically resets form controls to their initial values. |
| 3 | **button**  This creates a button that is used to trigger a client-side script when the user clicks that button. |
| 4 | **image**  This creates a clickable button but we can use an image as background of the button. |

### Example

## <form>

## <input type = "submit" name = "submit" value = "Submit" />

## <input type = "reset" name = "reset" value = "Reset" />

## <input type = "button" name = "ok" value = "OK" />

## </form>

## 

## Hidden Form Controls:

Hidden form controls are used to hide data inside the page which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page. For example, following hidden form is being used to keep current page number. When a user will click next page then the value of hidden control will be sent to the web server and there it will decide which page will be displayed next based on the passed current page.

### Example

## <form>

## <p>This is page 10</p>

## <input type = "hidden" name = "pagename" value = "10" />

## <input type = "submit" name = "submit" value = "Submit" />

## <input type = "reset" name = "reset" value = "Reset" />

## </form>

## 

* **<fieldset>:**It is used to draw a box around other form elements and group the related data.
* **<legend>:**It defines caption for fieldset elements.

## Example:

## <fieldset>

## <legend>User Information:</legend>

## <label for="name">Enter name</label><br>

## <input type="text" id="name" name="name"><br>

## <label for="pass">Enter Password</label><br>

## <input type="Password" id="pass" name="pass"><br>

## <input type="submit" value="submit">

## </fieldset>

## 

## HTML Form Example:

## <form id="survey-form">

## <h2> Survey Form</h2>

## <div>Thank you for taking the time to help us improve the platform. </div>

## <div class="form-group">

## <label id="name-label" for="name">Name</label>

## <input type="text" name="name" id="name" class="form-control" placeholder="Enter your name" required/>

## <label id="email-label" for="email">Email</label>

## <input type="email" name="email" id="email" class="form-control" placeholder="Enter your Email" required/>

## <label id="number-label" for="number">Age<span class="clue">(optional)</span></label>

## <input type="number" name="age" id="number" min="10" max="99" class="form-control" placeholder="Age"/>

## <p>Which option best describes your current role?</p>

## <select id="dropdown" name="role" class="form-control" required>

## <option disabled selected value>Select current role</option>

## <option value="student">Student</option>

## <option value="job">Full time Job</option>

## <option value="learner">Full time Learner</option>

## <option value="preferNo">Prefer not to say</option>

## <option value="other">Other</option>

## </select>

## <button type="submit" id="submit" class="submit-button">Submit</button>

## </label>

## </form>

## 

**New Input Types in HTML5:**

HTML5 introduces several new <input> types like email, date, time, color, range, and so on. to improve the user experience and to make the forms more interactive.

If a browser failed to recognize these new input types, it will treat them like a normal text box.

* Color
* Date
* Datetime-local
* Email
* Month
* Number
* Range
* Search
* Tel
* Time
* URL
* week

**Input Type Color**

The color input type allows the user to select a color from a color picker and returns the color value in hexadecimal format (#rrggbb).

the default is #000000, which is black.

**Example:**

<form>

<label for="mycolor">Select Color:</label>

<input type="color" value="#00ff00" id="mycolor">

</form>



**Input Type Date**

The date input type allows the user to select a date from a drop-down calendar.

The date value includes the year, month, and day, but not the time.

<form>

<label for="mydate">Select Date:</label>

<input type="date" value="2019-04-15" id="mydate">

</form>



**Input Type Datetime-local**

The datetime-local input type allows the user to select both local date and time, including the year, month, and day as well as the time in hours and minutes.

<form>

<label for="mydatetime">Choose Date and Time:</label>

<input type="datetime-local" id="mydatetime">

</form>



**Input Type Email**

The email input type allows the user to enter e-mail address. It is very similar to a standard text input type, but if it is used in combination with the required attribute, the browser may look for the patterns to ensure a properly-formatted e-mail address should be entered.

<form>

<label for="myemail">Enter Email Address:</label>

<input type="email" id="myemail" required>

</form>

## 

## Input Type Month

The month input type allows the user to select a month and year from a drop-down calendar.

The value is a string in the format "YYYY-MM", where YYYY is the four-digit year and MM is the month number.

<form>

<label for="mymonth">Select Month:</label>

<input type="month" id="mymonth">

</form>



**Input Type Number**

The number input type can be used for entering a numerical value. Restriction can be provided to enter only acceptable values using the additional attributes min, max, and step.

<form>

<label for="mynumber">Enter a Number:</label>

<input type="number" min="1" max="10" step="0.5" id="mynumber">

</form>



**Input Type Range**

The range input type can be used for entering a numerical value within a specified range. It works very similar to number input, but it offers a simpler control for entering a number.

<form>

<label for="mynumber">Select a Number:</label>

<input type="range" min="1" max="10" step="0.5" id="mynumber">

</form>



**Input Type Search**

The search input type can be used for creating search input fields.

A search field typically behaves like a regular text field, but in some browsers like Chrome and Safari as soon as you start typing in the search box a small cross appears on the right side of the field that lets quickly clear the search field.



## Input Type Tel

The tel input type can be used for entering a telephone number.

Browsers don't support tel input validation natively. However, the placeholder attribute to help users in entering the correct format for a phone number, or specify a regular Expression to validate the user input using the pattern attribute.

<form>

<label for="myphone">Telephone Number:</label>

<input type="tel" id="myphone" placeholder="xx-xxxx-xxxx" required>

</form>



**Input Type Time**

The time input type can be used for entering a time (hours and minutes).

<form>

<label for="mytime">Select Time:</label>

<input type="time" id="mytime">

</form>



**Input Type URL**

The url input type can be used for entering URL's or web addresses.

multiple attribute to enter more than one URL. Also, if required attribute is specified browser will automatically carry out validation to ensure that only text that matches the standard format for URLs is entered into the input box.

<form>

<label for="myurl">Enter Website URL:</label>

<input type="url" id="myurl" required>

</form>



**Input Type Week**

The week input type allows the user to select a week and year from a drop-down calendar.

<form>

<label for="myweek">Select Week:</label>

<input type="week" id="myweek">

</form>



## semantic tags:

* A Semantic element clearly describes its meaning to both the browser and the developer.
* Semantic tags  add additional information about that document.

| **Semantic tag** | **Use** |
| --- | --- |
| <article> | The HTML <article> tag is used for independent self-contained content. |
| <aside> | Display some content aside from the content it is placed in |
| <details> | It creates a disclosure widget and contains additional details, that the user can open and view. |
| <figcaption> | This tag is used to give caption to a image |
| <figure> | Figure tag is used to add image, diagrams, code, etc in the article. |
| <mark> | It is used to highlight a word with a yellow background. |
| <footer> | It defines a footer for the web page |
| <header> | It defines a header for the webpage |
| <main> | It is used to write main content of the article . |
| <nav> | This tag is used to create navigation bar of webpage. |
| <section> | This tag is used to create different sections in webpage |

## Examples:

<article>

<h2>The article element</h2>

<p> The article element have independent, self-contained content</p>

</article>

## 

<aside>

<h5>Note:</h5>

<p>Some information about main content.</p>

</aside>

## 

<details>

<p>This is a detail tag.</p>

</details>

## 

<figure>

<img src="HTML.jpg" style="width:100%">

<figcaption>HTML logo.</figcaption>

</figure>

## 

<header>

<h2>Scaler tips.com</h2>

<p> An educational website</p>

</header>

## 

<footer>

<p><i>Written by:</i>XYZ</p>

<p><i>Email id:</i>XYZ@gmail.com</p>

</footer>

## 

<main>

<h1>Title</h1>

<p>Content related to title.</p>

</main>

## 

<p>Highlighted text = <mark>Scaler tips</mark></p>

## 

<nav>

<h1>nav tag</h1>

<a href=" ">Home</a>

<a href=" ">Contacts</a>

<a href=" ">index</a>

</nav>

## 

<section>

<h2>loremipsumtext*</*h2*>*

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi elementum finibus lectus, vel sollicitudin

tortor aliquam ac. Fusce molestie fringilla magna, et posuere tortor</p>

</section>

## 

# HTML Graphics

There are two modern web technologies for creating rich drawn graphics within the browser:

**HTML5 Canvas** and **Scalable Vector Graphics(SVG)**

* Canvas
* SVG

# HTML Canvas

# A canvas in HTML is an area where graphics can be drawn with JavaScript

# The canvas tag creates the graphic container.

# JavaScript then draws paths, shapes, text, images, etc. inside this container.

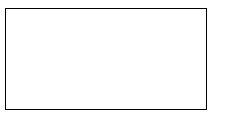
# Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

Basic Canvas Example:

<canvas id="myCanvas" width="200" height="100"style="border:1px solid #000000;">

</canvas>



# HTML Canvas Drawing

Drawing on a canvas must be done through JavaScript and requires the canvas element's graphics context.The graphics context is the object for which the various drawing methods are defined.

To draw on a canvas follow the following steps :

1. Find the canvas element using the HTML DOM method getElementById
2. Get the graphics context for the canvas element using getContext
3. Draw on the canvas using one of a number of defined graphics methods

These must be done in the context of a <script> element.

Example:

## <canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;">

## </canvas>

## <script>

## var canvas = document.getElementById("myCanvas");

## var ctx = canvas.getContext("2d");

## ctx.fillStyle = "#FF0000";

## ctx.fillRect(0,0,150,75);

## </script>

## 

# HTML Canvas Coordinates

The HTML canvas is a two-dimensional grid.

The upper-left corner of the canvas has the coordinates (0,0)

In the previous chapter, you saw this method used: fillRect(0,0,150,75).

This means: Start at the upper-left corner (0,0) and draw a 150x75 pixels rectangle.

Draw a Line

To draw a straight line on a canvas, use the following methods:

* moveTo(*x, y*) - defines the starting point of the line
* lineTo(*x, y*) - defines the ending point of the line

To actually draw the line, you must use one of the "ink" methods, like stroke().

# <canvas id="myCanvas" width="200" height="100"

# style="border:1px solid #d3d3d3;">

# Your browser does not support the canvas element.

# </canvas>

# <script>

# var canvas = document.getElementById("myCanvas");

# var ctx = canvas.getContext("2d");

# ctx.moveTo(0,0);

# ctx.lineTo(200,100);

# ctx.stroke();

# </script>

# 

# HTML Canvas Images

Canvas - Images

To draw an image on a canvas, use the following method:

* drawImage(*image,x,y*)

# <body>

# <img id="scream" width="220" height="277"

# src="idea-04.jpg" alt="The Scream">

# <p>Canvas:</p>

# <canvas id="myCanvas" width="240" height="297"

# style="border:1px solid #d3d3d3;">

# Your browser does not support the HTML5 canvas tag.

# </canvas>

# <script>

# window.onload = function() {

# var canvas = document.getElementById("myCanvas");

# var ctx = canvas.getContext("2d");

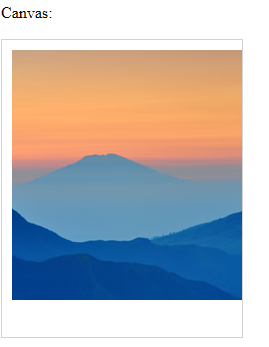
# var img = document.getElementById("scream");

# ctx.drawImage(img, 10, 10);

# };

# </script>

# </body>



# HTML Canvas Text

Drawing Text on the Canvas

To draw text on a canvas, the most important property and methods are:

* font - defines the font properties for the text
* fillText(*text,x,y*) - draws "filled" text on the canvas
* strokeText(*text,x,y*) - draws text on the canvas (no fill)

## <canvas id="myCanvas" width="200" height="100"

## style="border:1px solid #d3d3d3;">

## Your browser does not support the canvas element.

## </canvas>

## <script>

## var canvas = document.getElementById("myCanvas");

## var ctx = canvas.getContext("2d");

## ctx.font = "30px Arial";

## ctx.fillText("Hello World",10,50);

## </script>

## 

# SVG

* SVG stands for Scalable Vector Graphics
* SVG is used to define vector-based graphics for the Web
* SVG defines the graphics in XML format
* Every element and every attribute in SVG files can be animated
* SVG is a W3C recommendation
* SVG integrates with other W3C standards such as the DOM and XSL.
* SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

SVG Advantages

Advantages of using SVG over other image formats (like JPEG and GIF) are:

* SVG images can be created and edited with any text editor
* SVG images can be searched, indexed, scripted, and compressed
* SVG images are scalable
* SVG images can be printed with high quality at any resolution
* SVG images are zoomable
* SVG graphics do NOT lose any quality if they are zoomed or resized

SVG Shapes

SVG has some predefined shape elements that can be used by developers:

* Circle <circle>
* Rectangle <rect>
* Ellipse <ellipse>
* Line <line>
* Polyline <polyline>
* Polygon <polygon>
* Path <path>

## Example 1:

## <svg width="100" height="100">

## <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />

## 

**SVG Code explanation:**

* An SVG image begins with an <svg> element
* The width and height attributes of the <svg> element define the width and height of the SVG image
* The <circle> element is used to draw a circle
* The cx and cy attributes define the x and y coordinates of the center of the circle. If cx and cy are not set, the circle's center is set to (0, 0)
* The r attribute defines the radius of the circle
* The stroke and stroke-width attributes control how the outline of a shape appears. We set the outline of the circle to a 4px green "border"
* The fill attribute refers to the color inside the circle. We set the fill color to yellow
* The closing </svg> tag closes the SVG image

## Example 2:

## <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />

## Sorry, your browser does not support inline SVG.

## </svg>

## 

## Example3:

## <svg height="140" width="500">

## <ellipse cx="200" cy="80" rx="100" ry="50" style="fill:yellow;stroke:purple;stroke-width:2" />

## Sorry, your browser does not support inline SVG.

## </svg>

## 

## Example4:

## <svg height="210" width="500">

## <line x1="0" y1="0" x2="200" y2="200" style="stroke:rgb(255,0,0);stroke-width:2" />

## Sorry, your browser does not support inline SVG.

## </svg>

## 

## Differences Between SVG and Canvas

## SVG is a language for describing 2D graphics in XML.

## Canvas draws 2D graphics, on the fly (with a JavaScript).

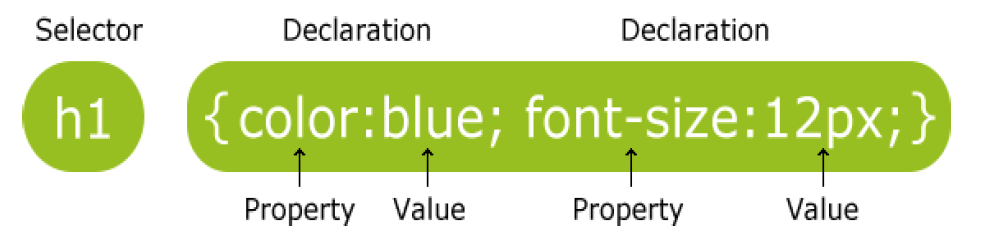
## 

## CSS:

## The abbreviation of CSS is Cascading Style Sheet. It helps in the styling of the web pages and helps in designing of the pages.

**CSS Syntax**

A CSS rule has three main parts: a selector, a property and a value.



The selector is normally the HTML element that style applied to.

The property is the style attribute to change.

Value of a style.

Types of CSS:

There are 3 types of CSS. Those are:

* Inline style
* Internal/Embedded style sheet
* External style sheet

**Inline style**

Inline CSS is used to add styles to tags.

o Inline CSS has the highest priority out of external, internal, and inline CSS.

o To add a style inside an HTML element all have to do is specify the desired CSS properties with the style HTML attribute.

**Syntax**:

<tagname style=”Property:value;Property:value;”>

Example:

<h1 style="color: blue; margin-left: 30px;">This is a heading.</h1>



**Internal Style Sheet:**

An internal style sheet should be used when a single document has a unique style.

Internal style should be specified in the head section of an HTML page using <style> tag.

<head>

**Syntax**:

<head>

<style type=”text/css”>

tagname

{

Property: value;

}

tagname

{

Property: value;

}

</style>

## </head>

## Example:

## <head>

## <style>

## body {

## background-color: yellow;

## }

## h1 {

## color: maroon; margin-left: 40px;

## }

## </style>

## </head>

## <body>

## <h1 >This is a heading.</h1>

## </body>

## 

## External Style Sheet:

## External Style Sheets are used to apply styles to all html web pages.

## Create a separate file with the extension .css and write css code in that file.

## Link both HTML and CSS file by using link tag in head part of HTML file.

## Syntax:

<head>

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

</head>

Example:

Mystyle.css

Body{

background-color: lightpink;

}

h1 {

color: navy; margin-left: 20px;

HtmlFile:

<html>

<head>

<link rel=”stylesheet” href=” myStyle.css”>

</head>

<body>

<h1>Hi I am from External Sheet</h1>

</body>

</html>



## CSS Selectors:

## As Selectors not only tags the following can be used. Those are:

## Simple Selectors(tag,id,class)

## Combinator Selectors(Space,>,+,~)

## Pseudo –class Selectors(:Special Starter)

## Pseudo-Element Selectors(::)

## Attribute Selectors([])

## 1. Simple Selectors:

## Simple Selectors can be tag or id or class

## Tagbased:

## Syntax:

## TagName{

## Property:Value;

## Property:value;

## }

## Example:

<head>

<Style type=”text\css”>

h1

{

color: blue;

margin-left: 40px;

}

</style >

</head>

<body>

<h1> Welcome</h1>

</body> ****

Example:

<head>

#first{

Color:red;

}

#second{

Color:blue;

}

</head>

<body>

<b id=”first”> Hello Welcome</b></br>

<b id=”second”>This is CSS selector class</b>

</body>

**ID based**:

Syntax:

#idname{

Property:Value;

Property:value;

}

Output

****

ClassName:

Syntax:

.classname{

Property:Value;

Property:value;

}

## 

## 

Example:

<head>

<style type="text/css">

.first{

Color:red;

}

.second{

Color:blue;

}</style > </head>

<body>

<b class="first"> Hello Welcome</b></br>

<b class="second"> cse</b></br>

<p class="first"> Hi</b>

<p class="second"> vits & vec</b>

</body>

<b id=”second”>This is CSS selector class</b>

## 

## 2. Combinator Selector:

## There are 4 types. Those are:

## Descendent Selector

## Child Selector

## Adjacent sibling selector

## General sibbling selector

## I. Descendent Selector:

## These tags applies styles which are available in another tag.

<body>

<div>

<p> Hi</p>

<h1> Hello</h1>

<p>Welcome</p>

</div>

<p>HTML</p></body>

<style type=”text/css”>

div p{

color:red;

}

## Output:

## In Html where ever p tag is available in div tag all that content will be displayed in red.

## 

## II. Child Selector:

## Child Selector(>) will apply style only on direct child but not on sub childs.

<style type=”text/css”>

div > p{

color:red;

}

<body>

<div>

<p> Hi</p>

<h1> Hello</h1>

<p>Welcome</p>

<h2>

<p> Inside descendant h2</p>

</h2>

<h1><h2>

<p> Inside descendant h2</p></h2></h1>

</div>

<p> Good Bye</p>

</body>

## Output:

Hi

Hello

Welcome will be in red color.

## 

## III. Adjacent sibling selector

## Adjacent sibling selector(+) will apply styles after the tag is closed

<head>

<style type="text/css">

div +p{

color:red;

}

</style>

</head>

## Output:

## Good Bye

will be in red color.

<body>

<div>

<p> Hi</p>

<h1> Hello</h1>

<p>Welcome</p>

<h2>

<p> Inside descendant h2</p>

</h2>

<h1><h2>

<p> Inside descendant h2</p></h2></h1>

</div>

<p> Good Bye</p>

<p> to </p>

<p> css</p>

</body>

</body>

## 

## IV. General sibbling selector

## These tags applies styles to all tags that are available after another tag.

<body>

<div>

<p> Hi</p>

<h1> Hello</h1>

<p>Welcome</p>

<h2>

<p> Inside descendant h2</p>

</h2>

<h1><h2>

<p> Inside descendant h2</p></h2></h1>

</div>

<p> Good Bye</p>

<p> to </p>

<p> css</p>

</body>

<style type=”text/css”>

div ~ p{

color:red;

}

## Output:

Good Bye

To

css

will be in red color.

## 

## 3. Pseudo class Selector(:)

## It is used to represent special states of an element. These are generally used in hyperlinks.

<style type=”text/css”>

a:link{

color:green;

}

a:hover{

color:red;

}

a:visited{

color:pink;

}

a:active{

color:violet;

}

<body>

<a href=”#”>Download Couse1 Material</a>

<a href=”#”>Download Couse2 Material</a>

<a href=”#”>Download Couse3 Material</a>

<a href=”#”>Download Couse4 Material</a>

</body>

## 

## 4. Pseudo-Element Selectors(::)

## It is used to apply style to a particular part of on an element.

## Example:

## 

<style type=”text/css”>

H1::first-letter{

Color:red;

}

<body>

<H1>Telugu</H1>

</body>

## Output:

## T will be in red color in the word Telugu.

## 

## 5. Attribute Selector([ ]):

## It is used to apply style to an attribute.

## Output:

<style type=”text/css”>

H1[id]{

Color:red;

}

<body>

<H1 id=”one”>First</H1>

<H1 >Second</H1>

<H1 id=”third”>Three</H1>

</body>

## First,Third will be in red color.

## 

**CSS Background with all its attributes:**

CSS background properties are used to define the background effects of an element.

**1. CSS Background Color**

o The background-color property specifies the background color of an element.

o The background color of a page is defined in the body selector:

o Below is example of CSS backgrounds

Example:

<head>

<style>

h2,p{

background-color: green;

}

</style>

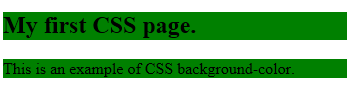
</head>

<body>

<h2>My first CSS page.</h2>

<p> This is an example of CSS background-color.</p>

</body>



**2. CSS Background Image**

o The background-image property specifies an image to use as the background of an element.

Example:

<head>

<style>

body {

background-image: url("idea-04.jpg");

margin-left:100px;

}

</style>

</head>

<body>

<h1>Hello m</h1>

</body>

****

**3. Background Image Repeat**

the background-image property repeats the background image horizontally and vertically. Some images are repeated only horizontally or vertically.

The background looks better if the image repeated horizontally only.

Example:

<head>

<style>

<body>background-image: url("idea-04.jpg");

margin-left:100px;

background-repeat: repeat-y;

}

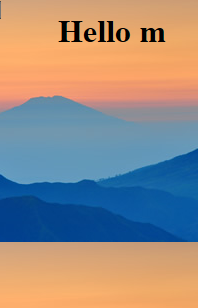
</style>

</head>

<body>

<h1>Hello m</h1>

</body>

**

**4. CSS Background Attachment**

o The background-attachment property specifies whether the background image should scroll or be fixed

**5. CSS Background Image Positioning :**

The background-position property is used to define the initial position of the background image. By default, the background image is placed on the top-left of the webpage.

The following positions can be specified.

1. center
2. top
3. bottom
4. left
5. right

Example:

<style>

body {

background-image: url("idea-04.jpg");

background-repeat: no-repeat;

background-position: right top;

margin-right: 200px;

background-attachment: fixed;

}

</style>

<h1>The background-attachment Property</h1>

<p>The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page).</p>

<p><strong>Tip:</strong> If you do not see any scrollbars, try to resize the browser window.</p>

<p>The background-image is fixed. Try to scroll down the page.</p>

<p>The background-image is fixed. Try to scroll down the page.</p>

<p>The background-image is fixed. Try to scroll down the page.</p>

<p>The background-image is fixed. Try to scroll down the page.</p>

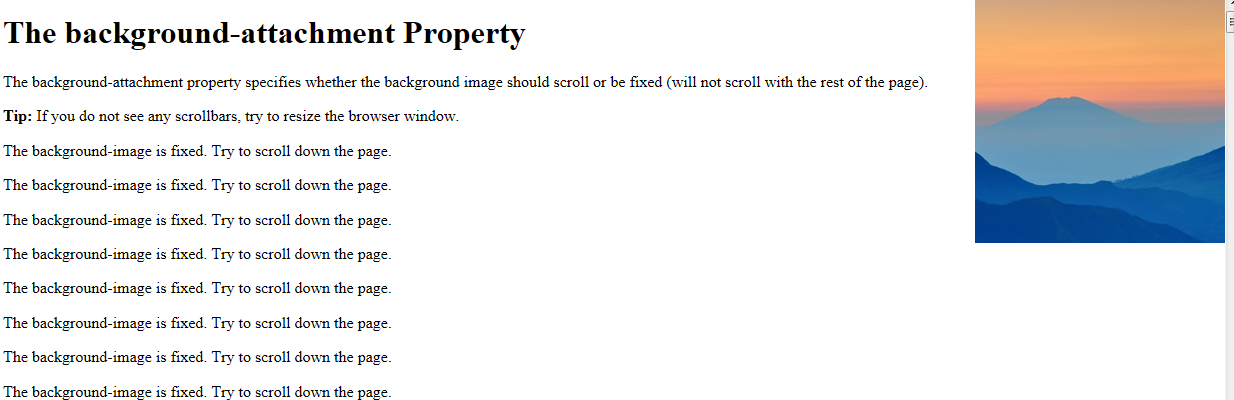
<p>The background-image is fixed. Try to scroll down the page.</p>

<p>The background-image is fixed. Try to scroll down the page.</p>

<p>The background-image is fixed. Try to scroll down the page.</p>

<p>The background-image is fixed. Try to scroll down the page.</p>

<p>The background-image is fixed. Try to scroll down the page.</p>



**CSS Borders:**The CSS border properties are use to specify the style, color and size of the border of an element. The CSS border properties are given below

* border-style
* border-color
* border-width
* border-radius

**1)CSS border-style**

* The Border style property is used to specify the border type which we want to display on the web page.
* There are some border style values which are used with border-style property to define a border.

|  |  |
| --- | --- |
| **Value** | **Description** |
| none | It does not define any border. |
| dotted | It is used to define a dotted border. |
| dashed | It is used to define a dashed border. |
| solid | It is used to define a solid border. |
| double | It defines two borders with the same border-width value. |
| groove | It defines a 3d grooved border. effect is generated according to border-color value. |
| ridge | It defines a 3d ridged border. effect is generated according to border-color value. |
| inset | It defines a 3d inset border. effect is generated according to border-color value. |
| outset | It defines a 3d outset border. effect is generated according to border-color value. |

Example:

<head>

<style>

p.dotted {border-style: dotted;}

p.dashed {border-style: dashed;}

p.solid {border-style: solid;}

p.double {border-style: double;}

</style>

</head>

<body>

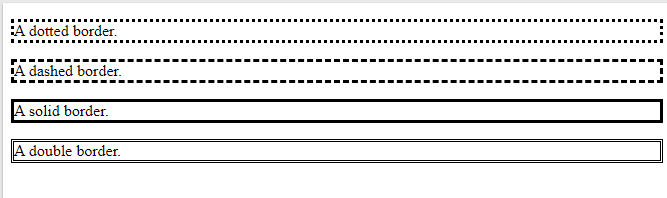
<p class="dotted">A dotted border.</p>

<p class="dashed">A dashed border.</p>

<p class="solid">A solid border.</p>

<p class="double">A double border.</p>

</body>

****

**2) CSS border-width**

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

<head>

<style>

p.one {

border-style: solid;

border-width: 5px;

}

p.two {

border-style: solid;

border-width: medium;

}

p.three {

border-style: dotted;

border-width: 2px;

}

p.four {

border-style: dotted;

border-width: thick;

}

p.five {

border-style: double;

border-width: 15px;

}

p.six {

border-style: double;

border-width: thick;

}

</style>

</head>

<body>

<p class="one">Some text.</p>

<p class="two">Some text.</p>

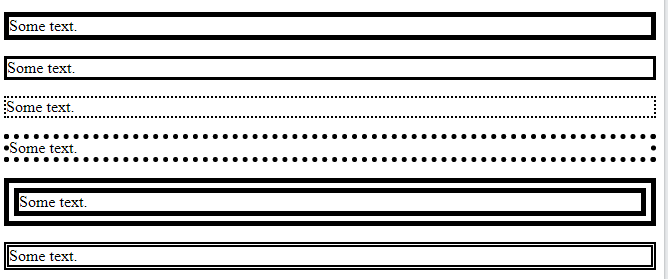
<p class="three">Some text.</p>

<p class="four">Some text.</p>

<p class="five">Some text.</p>

<p class="six">Some text.</p>

</body>

****

**3) CSS border-color**

The border-color property is used to set the color for the borders.

The color can be set by:

* name - specify a color name, like "red"
* HEX - specify a HEX value, like "#ff0000"
* RGB - specify a RGB value, like "rgb(255,0,0)"
* HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
* transparent

**Note:** If border-color is not set, it inherits the color of the element.

<head>

<style>

p.one{

border-style: solid;

border-color: red;

}

p.two {

border-style: solid;

border-color: green;

}

p.three {

border-style: dotted;

border-color: blue;

}

</style>

</head>

<body>

<p class="one">A solid red border</p>

<p class="two">A solid green border</p>

<p class="three">A dotted blue border</p>

</body>

****

# 4)CSS border-radius property:

This CSS property sets the rounded borders and provides the rounded corners around an element, tags, or div. It defines the radius of the corners of an element.

It is shorthand for **border top-left-radius, border-top-right-radius, border-bottom-right-radius** and **border-bottom-left-radius**.

|  |  |
| --- | --- |
| **Property** | **Description** |
| **border-top-left-radius** | It is used to set the border-radius for the top-left corner |
| **border-top-right-radius** | It is used to set the border-radius for the top-right corner |
| **border-bottom-right-radius** | It is used to set the border-radius for the bottom-right corner |
| **border-bottom-left-radius** | It is used to set the border-radius for the bottom-left corner |

Example

<style>

p.normal {

border: 2px solid red;

padding: 5px;

}

p.round1 {

border: 2px solid red;

border-radius: 5px;

padding: 5px;

}

p.round2 {

border: 2px solid red;

border-radius: 8px;

padding: 5px;

}

p.round3 {

border: 2px solid red;

border-radius: 12px;

padding: 5px;

}

</style>

</head>

<body>

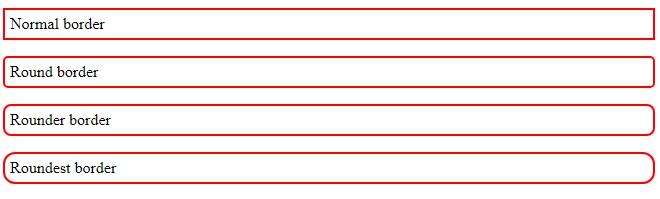
<p class="normal">Normal border</p>

<p class="round1">Round border</p>

<p class="round2">Rounder border</p>

<p class="round3">Roundest border</p>

</body>



# CSS – Text:

CSS Text allows to control the spacing, decoration, and alignment of text.

text properties of an element are:

1. color property
2. direction property
3. letter-spacing property
4. word-spacing property
5. text-indent property
6. text-align property
7. text-decoration property
8. text-transform property
9. white-space property
10. text-shadow property

1.color property :

The color property is used to set the color of a text.

<p style = "color: red;">

This text will be written in red.

</p>

****

2.direction property :

The direction property is used to set the text direction.

<p style = "direction:rtl;">

This text will be rendered from right to left </p>



3.letter-spacing property :

The letter-spacing property is used to add or subtract space between the letters that make up a word.

<p style = "letter-spacing:5px;">

This text is having space between letters.</p>



4.word-spacing property :

The word-spacing property is used to add or subtract space between the words of a sentence.

<p style = "word-spacing:5px;">

This text is having space between words.

</p>



5.text-indent property **:**

The text-indent property is used to indent the text of a paragraph.

<p style = "text-indent:1cm;">

This text will have first line indented by 1cm and this line will remain at

its actual position this is done by CSS text-indent property.

</p>



6.text-align property :

The text-align property is used to align the text of a document.

<p style = "text-align:right;">

This will be right aligned.

</p>

<p style = "text-align:center;">

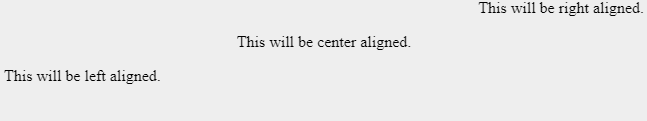
This will be center aligned.

</p>

<p style = "text-align:left;">

This will be left aligned.

</p>



7.text-decoration property:

The text-decoration property is used to underline, overline, and strikethrough text.

<p style = "text-decoration:underline;">

This will be underlined

</p>

<p style = "text-decoration:line-through;">

This will be striked through.

</p>

<p style = "text-decoration:overline;">

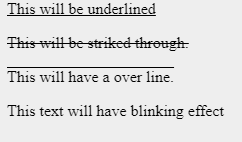
This will have a over line.

</p>

<p style = "text-decoration:blink;">

This text will have blinking effect

</p>



8.text-transform property :

The text-transform property is used to capitalize text or convert text to uppercase or lowercase letters.

<p style = "text-transform:capitalize;">

This will be capitalized

</p>

<p style = "text-transform:uppercase;">

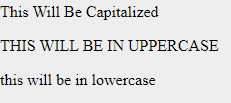
This will be in uppercase

</p>

<p style = "text-transform:lowercase;">

This will be in lowercase

</p>



9.white-space property :

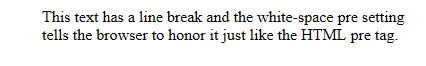
The white-space property is used to control the flow and formatting of text.

<p style = "white-space:pre;">

This text has a line break and the white-space pre setting

tells the browser to honor it just like the HTML pre tag.

</p>



10.text-shadow property :

The text-shadow property is used to set the text shadow around a text.

<p style = "text-shadow:4px 4px 8px blue;">

If your browser supports the CSS text-shadow property,

this text will have a blue shadow.

</p>

****

**BOX MODEL:**

In web development, the CSS box model refers to how HTML elements are modeled in browser engines and how the dimensions of those HTML elements are derived from CSS properties. It is a fundamental concept for the composition of HTML webpages.

The CSS box model is essentially a box that wraps around every HTML element. It consists

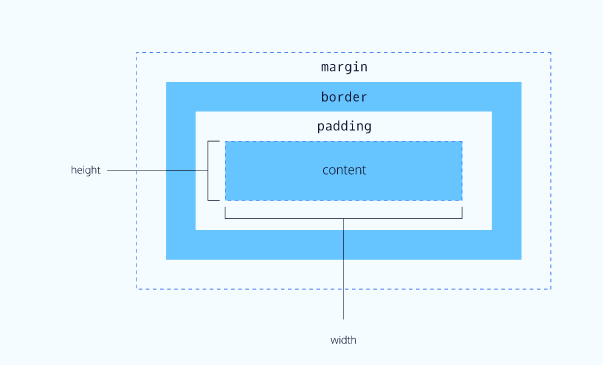
of: margins, borders, padding, and the actual content. These are also called as properties of css box.

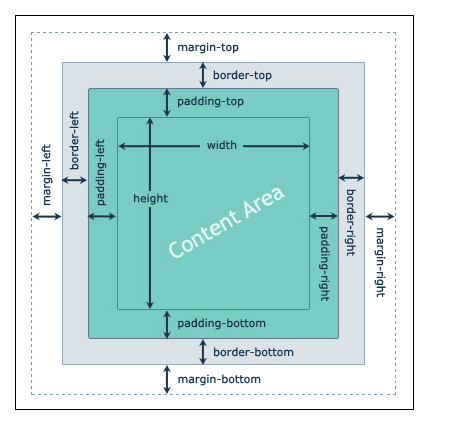
Content: The content area consists of content like image, text, or other forms of media content. The height and width properties help to modify the box dimensions.

Padding: The padding area is the space around the content area and within the border-box. It can be applied to all sides of the box or to the specific, selected side(s) - top, right, bottom, and/or left.

Border**:** The border area surrounds the padding and the content, and can be applied to all the sides of the box or to selected side(s) - top, right, bottom, and/or left.

Margin: The amount of space between the border and the outside edge of the element.(or)The margin area consists of space between the border and the margin. The margin doesn’t possess its own background color and is completely transparent. It shows the background color of the element, like the body element.





Example:

<head>

<style type="text/css">

div {

background-color: lightgrey;

width: 300px;

border: 15px solid green;

padding: 50px;

margin: 20px;

}

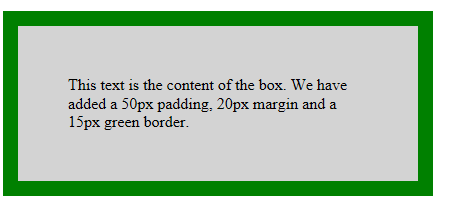
</style>

</head>

<body>

<div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.</div>

</body>



**Cascading and inheritance of style properties:**

Inheritance and the cascade are two fundamental concepts in CSS.

**Inheritance:**

Inheritance is a process of receiving values of properties by a child element from its parent element.

Inheritance can be used to specify the font properties for the html or body elements and the styles will be inherited by all other elements. For a specify container background and text colors elements can be applied .Text color will automatically be any child elements in that container.The background color is not inherited, but the initial value for background-color is transparent, which means a parent’s background will shine through. The effect is similar to the page’s appearance if background colors were inherited.

Example:

**One.css**

.up {

background-color: maroon;

color: blue;

font-weight: bold;

}

.down {

background-color: inherit;

color: inherit;

font-weight: normal;

}

inhertExample.html

<!doctype html>

<html lang="en">

<head>

<link href="one.css" rel="stylesheet" type="text/css" />

</head>

<body>

<div class="up">w3resource<br/>

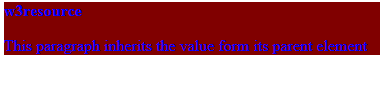
<p class="down">This paragraph inherits the value form its

parent element</p>

</div>

</body>

</html>



**The Cascade:**

It is the mechanism that controls the end result when multiple, conflicting CSS declarations apply to the same element. There are four main concepts that control the order in which CSS declarations are applied:

1. Origin & Importance
2. Selector Specificity
3. Order of Appearance
4. Initial & Inherited Properties (default values)

#### Origin & importance:

The highest weighted attribute that the cascade checks is a combination of the importance and the origin of a given rule.

As far as the origin of a CSS rule goes, there are three places that a rule can come from.

1. ***User-Agent***: These are the default styles provided for the element by the browser. This is why inputs can look slightly different on different browsers, and it’s also one of the reasons that people like to use CSS resets, to make sure that user-agent styles are overridden.
2. ***User***: These are defined and controlled by the user of the browser. Not everyone will have one, but when people do add one, it’s usually for overriding styles & adding accessibility to websites.
3. ***Author***: This is CSS declared *by the HTML document*. When we’re writing stuff as front-end developers this is really the only origin that we have in our control.

The importanceof a CSS declaration is determined by the appropriately-named !important syntax. Adding !important to a CSS rule automatically jumps it to the front of the cascade algorithm

#### Selector specificity

The second weight in the CSS cascade is selector specificity. In this tier, the browser looks at the selectors used in the CSS declaration.

CSS selectors can belong to one of the following weighted tiers.

1. Inline styles (anything inside a style tag)
2. ID selectors
3. Classes / pseudo-selectors
4. Type selectors (for example, h1) & pseudo-elements (::before)

#### Source order

The last main tier of the CSS cascade algorithm is resolution by source order. When two selectors have the same specificity, the declaration that comes last in the source code wins.

#### Initial & inherited properties

While initial & inherited values aren’t truly part of the CSS cascade, they do determine what happens if there are no CSS declarations targeting the element.

# CSS-animations:

CSS animations make it possible to do simple animations without JavaScript at all.

An animation makes an element change gradually from one style to another.

**CSS Animation property**is used to create animation on the webpage.

Animations consist of two components, a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation's style, as well as possible intermediate waypoints.

\*CSS animations are dependent on keyframes and animation properties.

**keyframes**– keyframes are used to define the styles an element will have at various times.

**animation properties**– animation properties are used to assign @keyframes to a specific element and determine how it is animated.

There are eight animation properties:

* **animation-delay** – specifies a delay for the start of an animation.
* **animation-direction** – specifies whether an animation should play in reverse direction or alternate cycles.
* **animation-duration** – specifies how many seconds or milliseconds an animation takes to complete one cycle.
* **animation-fill-mode** – specifies a style for the element when the animation is not playing. Such as when it is finished or when it has a delay.
* **animation-iteration-count** – specifies the number of times an animation should be played.
* **animation-name** – specifies the name of the @keyframes animation.
* **animation-play-state** – specifies whether the animation is running or paused.
* **animation-timing-function** – specifies the speed curve of the animation.

## How CSS animation works:

When the animation is created in the [**@keyframe rule**](https://www.javatpoint.com/css-keyframes-rule), it must be bound with selector; otherwise the animation will have no effect.

The animation could be bound to the selector by specifying at least these two properties:

* The name of the animation
* The duration of the animation

Example:

<head>

<style>

div{

width: 100px;

height: 100px;

background-color: red;

animation-name: example;

animation-duration: 4s;

}

@keyframes example {

0% {background-color: red;}

25% {background-color: yellow;}

50% {background-color: blue;}

100% {background-color: green;}

}

</style>

</head>

<body>

<h1>CSS Animation</h1>

<div></div>

</body>



### CSS Transitions:

A CSS transition allows changing the property values of an element over a given duration provided by the user. To create a transition you must first identify which CSS property you want to add an effect to and then specify the duration of the effect. If no duration is set, the transition will not occur.

There are four transition properties:

**transition-delay**– specifies the delay, in seconds (s),we would like to assign your transition effect.

**transition-duration** – specifies the duration, in seconds (s) or milliseconds (ms), we would like to assign your transition effect.

**transition-property** – specifies the name of the CSS property our transition effect is meant for.

**transition-timing-function**– Specifies the speed curve of the transition effect. Meaning, the type of speed variation we want to select for our transition effect. There is no “fast” or “slow” options. Instead there are speed curve options that go from one speed to another. Such as “ease” which tells your effect to start slow, then go fast, then end slowly.

To create a transition you only need to change one of these properties over the duration you choose. However, it is possible to change more than one property at the same time; resulting in more dramatic transitions.

Example:

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

transition: width 2s;

}

div:hover {

width: 300px;

}

</style>

</head>

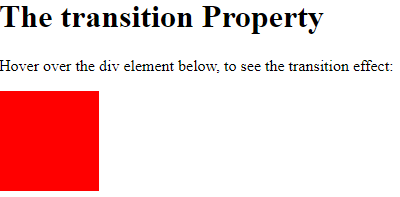
<body>

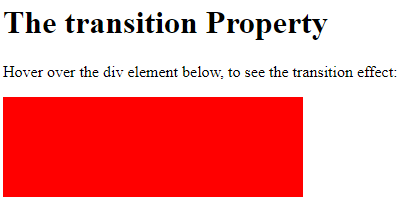
<h1>The transition Property</h1>

<p>Hover over the div element below, to see the transition effect:</p>

<div></div>

</body>





**Layout:**

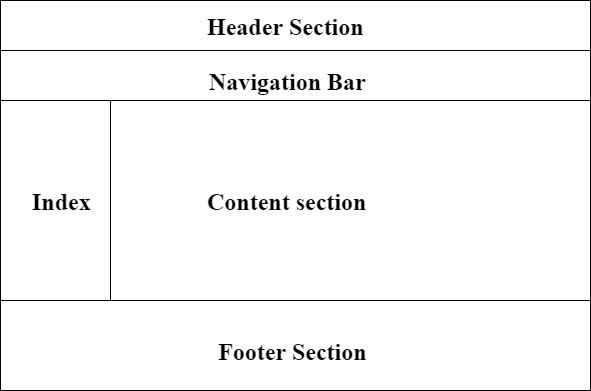
 A layout would be an arrangement of putting things on a web page in a proper way so that the web page looks aesthetic, pleasing, and uniform.

The Web page layout is the most important part to keep in mind while creating the website or web applications so that our website will look professional with great design artifacts.

Layout Elements

The elements which help create layouts are :

* header
* nav
* section
* article
* aside
* footer
* details
* summary



**header**:

Used to include header content in the web page, like information about the page, summary, login/register links. shopping cart details etc.

**nav**:

Used to provide navigational links.

Example: menus for routing through across the application pages.

**section**:

Used to represent a section in the web page and it can be anything such as quick news section, headline, etc.

**article**:

it’s a separate section used to include a blog, a forum post, a magazine article, etc.

**aside**:

it's used to include some extra information regarding the main content.

**footer**:

Used to include footer content in the web page, like licensing information, copyright information, some quick access links, etc.

**details**:

Creates an interactive section, which when clicked will display its information i.e. hide/show the section based on the request.

**summary** :

Used to include the information within a <details> element, click on the <summary> element would display the info.

CSS provides several page layout techniques allow controlling the positions of the elements on the page.

### Various Layout Techniques

Given below are the various layout techniques:

* Normal layout
* CSS display property
* Flexbox
* Floats
* Grid layout
* Positioning elements
* Table layout
* Multi-column layout

**Normal Flow Layout:**

By default the HTML document will be displayed by the browser in normal flow layout. Elements are displayed in the order they occur in the HTML file .

Block elements display vertically from top to bottom.

Examples: <p>,<div>,<h2>,<ul>,and<table>

Inline elements display horizontally from left to right.

Examples: <em>,<a>,<img>,and <span>

* In normal flow processing, each displayed element has a corresponding box
  + html element box is called initial containing block and corresponds to entire document
  + Boxes of child elements are contained in boxes of parent
  + Sibling block elements are laid out one on top of the other
  + Sibling inline elements are one after the other

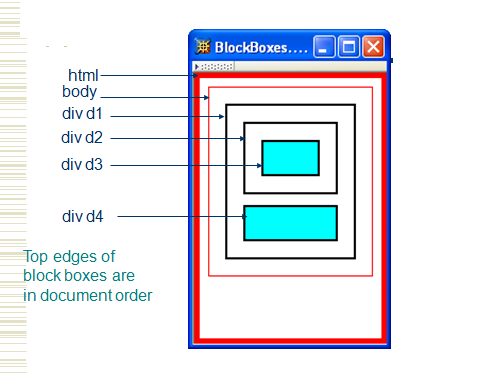
**Example**:

CSS File:



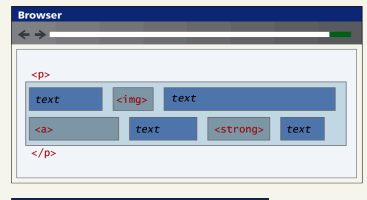
HTML File:





Example 2:





Example 3:

<!doctype html>

<html lang="en">

<head>

<title>NormalFlow</title>

</head>

<body>

<h1>This is <span style="color:blue;font-weight:bold">NormalFlow Layout</span> example.</h1>

<p>

the image <img src="idea-04.jpg"/>contains<br/>

<a href="">click</a> the <strong>Link</strong>

</p>

<div>

<h2> In Div Block</h2>

<p>

div is an example of block level

</p>

<p>

Other examples are<br/>

h2 tag<br/>

ul tag<br/>

</p>

</div>

<ul>

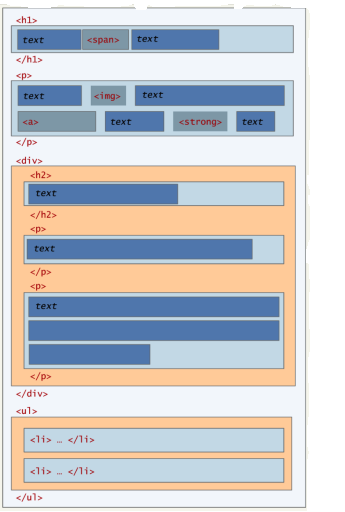
<li>one</li>

<li>two</li>

</ul>

</body>

</html>



It is possible to change whether an element is block level or inline via the CSS display property. Consider the following two CSS rules:

span { display: block; }

li { display: inline; }

These two rules will make all elements behave like block-level elements and all

<li> elements like inline (that is, each list item will be displayed on the same line).

# CSS display Property:

The display property specifies the display behavior (the type of rendering box) of an element.

**Syntax**

element {

display: value;

}

## Property Values:

|  |  |  |
| --- | --- | --- |
| **Value** | **Description** |  |
| inline | Displays an element as an inline element (like <span>). Any height and width properties will have no effect |  |
| block | Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width | [Demo ❯](https://www.w3schools.com/cssref/playdemo.asp?filename=playcss_display&preval=block) |
| contents | Makes the container disappear, making the child elements children of the element the next level up in the DOM |  |
| flex | Displays an element as a block-level flex container |  |
| grid | Displays an element as a block-level grid container |  |
| inline-block | Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values |  |
| inline-flex | Displays an element as an inline-level flex container |  |
| inline-grid | Displays an element as an inline-level grid container |  |
| inline-table | The element is displayed as an inline-level table |  |
| list-item | Let the element behave like a <li> element | [Demo ❯](https://www.w3schools.com/cssref/playdemo.asp?filename=playcss_display&preval=list-item) |
| run-in | Displays an element as either block or inline, depending on context |  |
| table | Let the element behave like a <table> element |  |
| table-caption | Let the element behave like a <caption> element |  |
| table-column-group | Let the element behave like a <colgroup> element |  |
| table-header-group | Let the element behave like a <thead> element |  |
| table-footer-group | Let the element behave like a <tfoot> element |  |
| table-row-group | Let the element behave like a <tbody> element |  |
| table-cell | Let the element behave like a <td> element |  |
| table-column | Let the element behave like a <col> element |  |
| table-row | Let the element behave like a <tr> element |  |
| none | The element is completely removed |  |
| initial | Sets this property to its default value . |  |
| inherit | Inherits this property from its parent element. |  |

Example:

<!doctype html>

<html lang="en">

<head>

<title> CSS Display Property Example </title>

<style>

.main {

margin:30px;

text-align:center;

}

#para1{

height: 120px;

width: 180px;

background: darksalmon;

display: block;

}

#para2{

height: 120px;

width: 180px;

background: olive;

display: block;

}

#para3{

height: 120px;

width: 180px;

background: fuchsia;

display: block;

}

</style>

</head>

<body>

<h2> Display Block Example </h2>

<br>

<div class = "main">

<div id="para1"> Java is a highly object-oriented, platform-independent and secure programming language. </div>

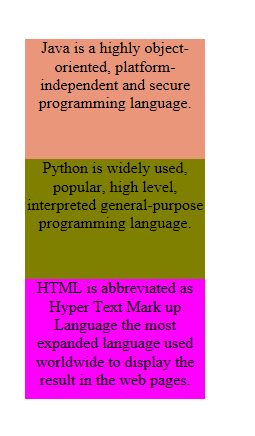
<div id="para2"> Python is widely used, popular, high level, interpreted general-purpose programming language. </div>

<div id="para3"> HTML is abbreviated as Hyper Text Mark up Language the most expanded language used worldwide to display the result in the web pages.</div>

</div>

</body>

</html>



**Beyond the Normal Flow:**

Normal Flow can be altered using CSS Position & CSS Floats properties.

Positions in CSS: Positions are used to change the order of an element.

There are 5 position properties in CSS.

Those are:

1. static
2. Absolute
3. Relative
4. Fixed
5. Stick

Syntax:

selector{

position: value;

}

value = static, relative, absolute, fixed, sticky

The following attributes can be applied for position property:

1. Top
2. Left
3. Right
4. Bottom

### static

Static is the default position for HTML elements. Elements with **position: static** are positioned based on the normal flow of the page, as you would expect them to be without any CSS styling. They are not affected by the top, right, bottom, or left properties.

“Z-Index” also does not apply to static elements.

Example:

Css File: body

{

background:url("Night.jpg");

}

#Astro\_Girl

{

position:static;

margin-bottom:10px;

}

#Astro\_Cartoon

{

position:static;

margin-bottom:10px;

}

#Astro\_Boy

{

position:static;

margin-bottom:10px;

}

HTML File:

<!doctype html>

<html lang="en">

<head>

<title>Understanding static Position</title>

<link rel="stylesheet" href="NoPositions.css">

</head>

<body>

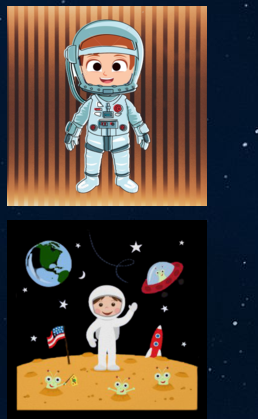
<div id="Astro\_Girl"><img src="Astro\_Girl.png"></img></div>

<div id="Astro\_Cartoon"><img src="Astro\_Cartoon.png"></img></div>

<div id="Astro\_Boy"><img src="Astro\_Boy.png"></img></div>

</body>

</html>



**relative**

 position: relative sets the new position of an element relative to the normal position.

To modify the position apply the top, bottom, right, and left properties

Example:

CSS File:

body

{

background:url("Night.jpg");

}

#Astro\_Girl

{

position:relative;

left:200px;

width:200px;

height:200px;

}

#Astro\_Cartoon

{

position:relative;

left:400px;

width:200px;

height: 200px;

}

#Astro\_Boy

{

position:relative;

left:200px;

width:200px;

height:200px;

}

HTML File

<!doctype html>

<html lang="en">

<head>

<title>Understanding relative Position</title>

<link rel="stylesheet" href="relativecss.css">

</head>

<body>

<div id="Astro\_Girl"><img src="Astro\_Girl.png"></img></div>

<div id="Astro\_Cartoon"><img src="Astro\_Cartoon.png"></img></div>

<div id="Astro\_Boy"><img src="Astro\_Boy.png"></img></div>

</body>

</html>



**Absolute**

Depends up on the parent position which side it has to move will be decided. If parent position is not specified then <body> tag position based is considered.

body

{

background:url("Night.jpg");

}

#Astro\_Girl

{

position:absolute;

}

#Astro\_Cartoon

{

position:static;

}

#Astro\_Boy

{

position:relative;

left:420px;

width:200px;

height:200px;

}

HTML File:

<!doctype html>

<html lang="en">

<head>

<title>Understanding absolute Position</title>

<link rel="stylesheet" href="absoluteposition.css">

</head>

<body>

<div id="Astro\_Girl"><img src="Astro\_Girl.png"></img></div>

<div id="Astro\_Cartoon"><img src="Astro\_Cartoon.png"></img></div>

<div id="Astro\_Boy"><img src="Astro\_Boy.png"></img></div>

</body>

</html>



**Fixed:**

An element under position: fixed is also removed from the normal document flow. It is positioned relative to the *viewport*. The top, right, bottom, and left properties are used for these values.

There are two main differences between fixed and absolute:

* In position:fixed, all the elements are placed relative to the <html> document even if it has a parent class.
* The elements are not affected by scrolling. They stay in their exact position even if we scroll the page.

CSS File:

body

{

background:url("Night.jpg");

}

#Astro\_Girl

{

position:fixed;

top:50px;

left:100px;

}

#Astro\_Cartoon

{

position:fixed;

top:250px;

left:300px;

}

#Astro\_Boy

{

position:absolute;

top:50px;

left:500px;

}

HTML FILE:

<!doctype html>

<html lang="en">

<head>

<title>Understanding fixed Position</title>

<link rel="stylesheet" href="fixedposition.css">

</head>

<body>

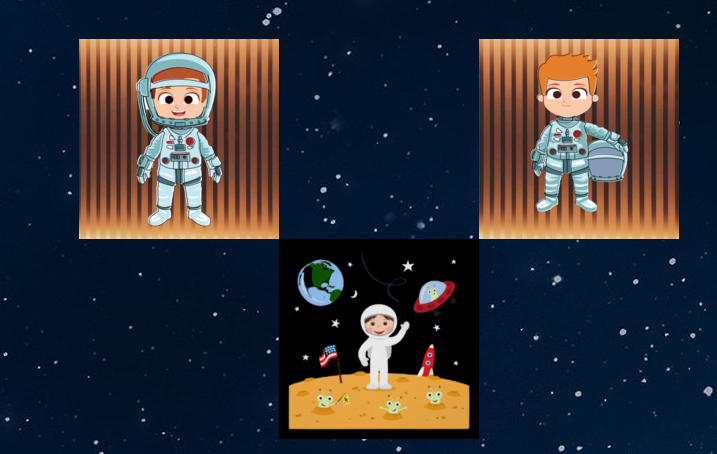
<div id="Astro\_Girl"><img src="Astro\_Girl.png"></img></div>

<div id="Astro\_Cartoon"><img src="Astro\_Cartoon.png"></img></div>

<div id="Astro\_Boy"><img src="Astro\_Boy.png"></img></div>

</body>

</html>



**Sticky:**

 Position:sticky can be explained as a mixture of  position:relative  and  position:fixed. At declaration, it acts like position:relative,but when scrolling, it acts like position:fixed.

Sticky position is used to stop scrolling after reaching a particular point.

CSS File:

body

{

background:url("Night.jpg");

}

#Astro\_Girl

{

position:absolute;

top:0px;

left:400px;

}

#Astro\_Cartoon

{

position:sticky;

top:200px;

}

#Astro\_Boy

{

position:absolute;

top:400px;

left:400px;

}

HTML File:

<!doctype html>

<html lang="en">

<head>

<title>Understanding sticky Position</title>

<link rel="stylesheet" href="stickyposition.css">

</head>

<body>

<div id="Astro\_Girl"><img src="Astro\_Girl.png"></img></div>

<div id="Astro\_Cartoon"><img src="Astro\_Cartoon.png"></img></div>

<div id="Astro\_Boy"><img src="Astro\_Boy.png"></img></div>

</body>

</html>



**Z-index**

 z-index to specify which element should be on top of the others. In other words, the z-index specifies the **stack order** of our elements. This is common for positioning text over an image.It can create “layers” between HTML elements

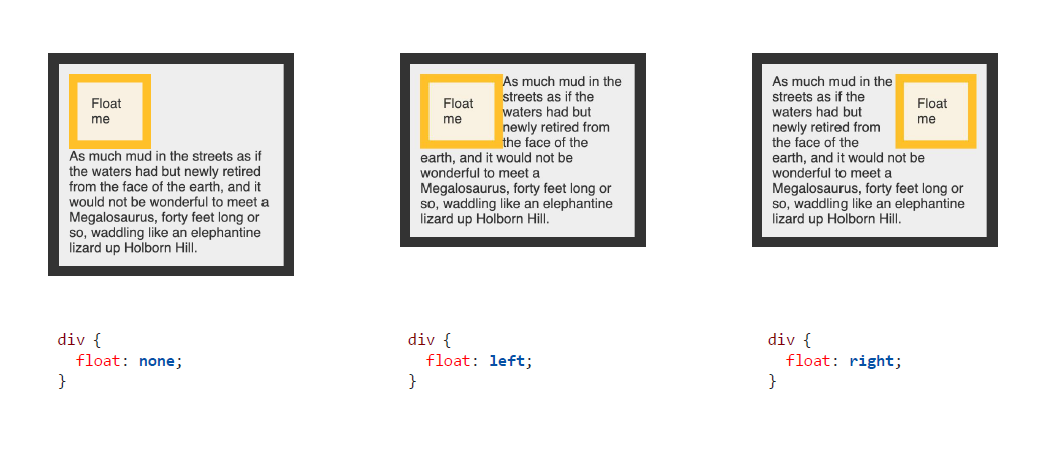
**CSS float:**

**CSS Float**was mainly used to float images and wrap text around it.

The possible float property value is given below.

* 1. **Left –** The element placed on the left side of the container.
  2. **Right –** The element placed on the right side of the container.
  3. **None –** Default float property, does not apply any float values to the property.

Example:



**CSS Float Property** can also be used to build layouts using div tag.

**MEDIA TYPE:**

Specifying Media Type:

These are used to format the document to be presented correctly on various type of media such as screen, print, speech ,browser,…etc.

**Media Type:**

all: for all media type devices.

braille: for feedback devices.

embossed: for paged braille printers. Braille characters are small rectangular blocks called cells that contain tiny palpable bumps called raised dots.

handheld: for hand-held type devices.

print: for printed version and display print preview on the screen.

projection: for projected presentations such as projectors.

screen: for computer screens. view on browser window.

tty: for media using fixed-pitch such as teletypes.

tv: for display television devices.

Generally used media types are: print, screen, speech, all.

**Creating Media Dependent Style Sheets:**

Media dependent stylesheets are basic stylesheets only but apply to html document only when mediatype matches device type on which document is visible.

Three methods are used.

1. Using the @media at rules:

It is used to define style rules for different media types in a single style sheet.

Example:

<style>

@media screen{

body {

color: #32cd32;

font-family: Arial, sans-serif;

font-size: 14px;

}

}

@media print {

body {

color: #ff6347;

font-family: Times, serif;

font-size: 12pt;

}

}

@media screen, print {

body {

line-height: 1.2;

}

}

</style>

</head>

<body>

<h1>CSS Media Types</h1>

<p><strong>Note:</strong> If you print (or print preview) this page the output of HTML code appears differently than it appears on the screen.</p>

</body>

</html>

# Output:

# CSS Media Types

**Note:** If you print (or print preview) this page the output of HTML code appears differently than it appears on the screen.

# 2. Usage of CSS @import: rule

# The @import: rule imports another stylesheet into the current style sheet. It should appear right at the start of the style sheet before any of the rules, and its value is a URL.

<style>

   <!--

      @import "mystyle.css";

      or

      @import url("mystyle.css");

      .......other CSS rules .....

   -->

</style>

Example 2:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of CSS @import rule</title>

<style>

@import url("/examples/css/screen.css") screen;

@import url("/examples/css/print.css") print;

body {

background: #f6f6f6;

line-height: 1.2;

}

</style>

</head>

<body>

<h1>Media Dependent Style Sheets Using the @import At-rules</h1>

<p>If you print (or print preview) this page the output of HTML code appears differently than it appears on the screen. Because different style sheets are used for different media types i.e. screen.css for computer screen while print.css for print media.</p>

</body>

</html>

**Ouput**:

# Media Dependent Style Sheets Using the @import At-rules

If you print (or print preview) this page the output of HTML code appears differently than it appears on the screen. Because different style sheets are used for different media types i.e. screen.css for computer screen while print.css for print media.

**3. Using the link element:**

Link element is used to specify the target media for an external style sheet.

Example:

<link rel=”stylesheet” media=”all” href=”URL”>

URL=> address of Css file.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of HTML media attribute</title>

<link rel="stylesheet" media="all" href="/examples/css/common.css">

<link rel="stylesheet" media="screen" href="/examples/css/screen.css">

<link rel="stylesheet" media="print" href="/examples/css/print.css">

</head>

<body>

<h1>Media Dependent Style Sheets Using the <code>&lt;link&gt;</code> Element</h1>

<p>If you print (or print preview) this page the output of HTML code appears differently than it appears on the screen. Because different style sheets are used for different media types i.e. screen.css for computer screen while print.css for print media.</p>

</body>

</html>

# Media Dependent Style Sheets Using the <link> Element

If you print (or print preview) this page the output of HTML code appears differently than it appears on the screen. Because different style sheets are used for different media types i.e. screen.css for computer screen while print.css for print media.

**Introduction to responsive design:**

Responsive web design is about creating websites(web applications ) which automatically adjust themselves to look good on all devices from small phones to large desktop.

**Responsive structure**



**VIEW PORT:**

* The viewport is the user's visible area of a web page.
* The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.
* HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.
* <meta name="viewport" content="width=device-width, initial-scale=1.0">
* This gives the browser instructions on how to control the page's dimensions and scaling.
* The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
* The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

**Responsive design has three core principles:**

1. **Fluid Grid System**

A fluid grid has fluid-width columns, fixed gutters and fixed side margins. The fluid grid has a flexible content width that goes edge to edge as per the screen size. In a fluid grid, columns either grow or shrink to adapt to the available space.

1. **Fluid Image Use**

The easiest way to handle fluid images (images that scale to fit their container) is using the CSS command:

**img { max-width: 100%;}**

This tells the browser that the image should be a maximum 100% of its pixel value and that it should scale according to its container. The idea is that this prevents an image from being stretched when the container becomes larger than the image – and thus eliminates degradation of the image – and ensures that it will shrink to the container when needed (this also maintains the original aspect ratio of the image).

1. **Media Queries**

A Media query tells the browser to display the content in a particular way on a particular screen.

It uses the @media rule to include a block of CSS properties only if a certain condition is true.

<style>

         @media screen and (max-width: 420px) {

            body {

               background-color: lightblue;

            }

         }

      </style>

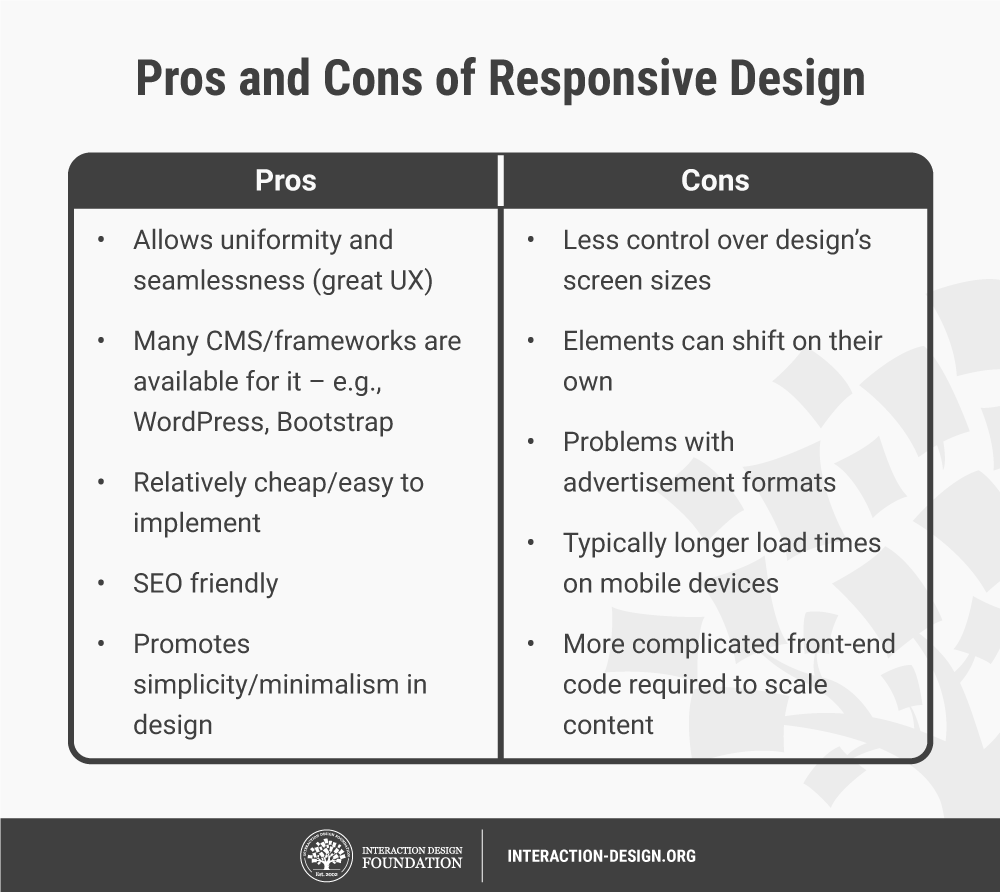
**Note:**

**<768px=phones**

**>768=Tablet**

**<=992=desktop**

**>1200=large desk top.**



# Bootstrap:

Bootstrap was developed by Mark Otto and Jacob Thornton

 Bootstrap is a free, open source front-end development framework for the creation of websites and web apps. Designed to enable responsive development of mobile- first websites, Bootstrap provides a collection of syntax for template designs.

Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins

**Where to Get Bootstrap**

There are two ways to start using Bootstrap on your own web site

Method 1:

Bootstrap needs atleast 3 files for its operation.

bootstrap.css :

This file contains various CSS for bootstrap.

bootstrap.js:

This file contains various JavaScript functionalities e.g. dropdown and alerts etc.

jQuery.js :

This file is the jQuery library which can be downloaded from the ‘jQuery’ website. It is required for proper working of ‘bootstrap.js’.

Download these files from getbootstrap.com.

Method 2:

Include Bootstrap from a CDN:

In this method no need not to download the files, but provide the links

to the files. In this case the code will not work in offline mode.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

</head>

<body>

<div class="jumbotron text-center">

<h1>My First Bootstrap Page</h1>

<p>Resize this responsive page to see the effect!</p>

</div>

<div class="container">

<div class="row">

<div class="col-sm-4">

<h3>First Year Subjects</h3>

<p>M1.</p>

<p>CPDS</p>

<p>Chemistry</p>

</div>

<div class="col-sm-4">

<h3>Second Year Subjects</h3>

<p>ADS</p>

<p>Java</p>

<p>DBMS</p>

</div>

<div class="col-sm-4">

<h3>Third Year Subjects</h3>

<p>IP&WT</p>

<p>CN</p>

<p>Embedded</p>

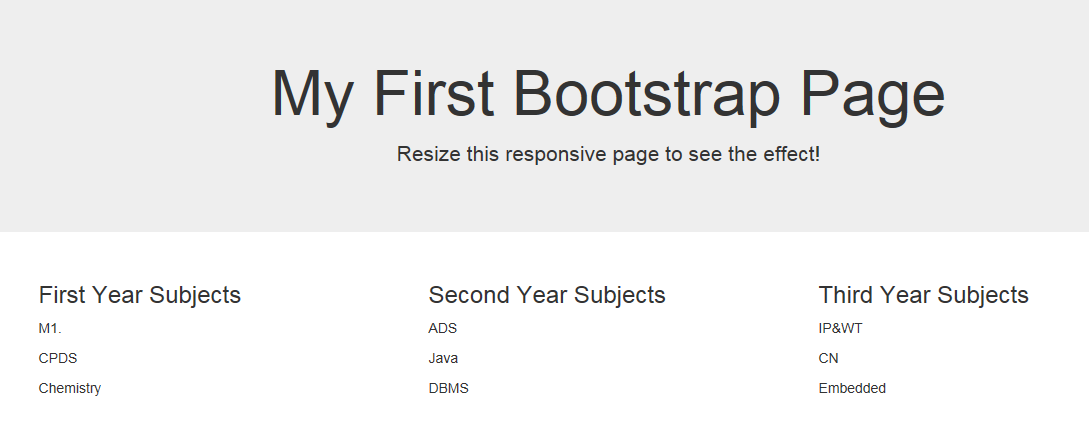
</div>

</div>

</div>

</body>

</html>



**Bootstrap package contains:**

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

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

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

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

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

**HTML – Entities:**

Some characters are reserved in HTML and they have special meaning when used in HTML document. For example, you cannot use the greater than and less than signs or angle brackets within your HTML text because the browser will treat them differently and will try to draw a meaning related to HTML tag.

A character entity looks like this:

&*entity\_name*;

OR

&#*entity\_number*;

## Some Useful HTML Character Entities:

## image map:

An imagemap is a User Interface technique in which a user can click anywhere inside of an image, and the location of the click affects the outcome of the click. The idea is that clicking on different sections of the image creates different effects. For example, the image might be a map of the United States, and clicking within the borders of a particular state leads to a page about that state.

There are two kinds of image maps:

* [server-side image maps](https://html.com/attributes/img-ismap/)
* client-side image maps

The usemap attribute specifies an image as a client-side image map.

An image map is an image with clickable areas.

The usemap attribute is associated with a <map>element's name attribute, and creates a relationship between the <img>and the<map>1.

Example:

<!doctype html>

<html lang="en">

<head> <title>USe Map Example</title> </head>

<body>

<img src="idea-04.jpg" alt="Workplace" usemap="#workmap" width="400" height="379">

<map name="workmap">

<area shape="rect" coords="34,44,270,350" alt="Computer" href="Astro\_Boy.png">

<area shape="rect" coords="290,172,333,250" alt="Phone" href="Astro\_Cartoon.png">

<area shape="circle" coords="337,300,44" alt="Cup of coffee" href="Astro\_Girl.png">

</map>

</body>

</html>