**HTML**

**HTML – H**yper **T**ext **M**arkup **L**anguage

HTML is the standard markup language for creating Web pages.

HTML elements tell the browser how to display the content.

**HTML Element:**

The HTML element usually consists of an opening tag (<element\_name>), closing tag (</element\_name>) , and the content in between: <element\_name>...content...</element\_name>

**Examples of some HTML elements:**

* <h1>Heading</h1>
* <p>paragraph</p>

**Basic HTML document Structure:**

<!DOCTYPE html>

<html>

<head>

<title> Example </title>

</head>

<body>

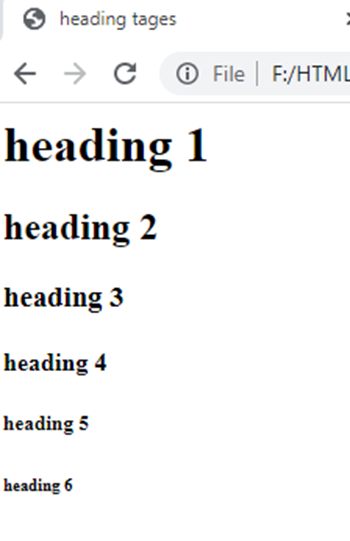
This is where you would include the content on your Web page.

</body> </html>

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <!DOCTYPE html> | Defines that this document is an HTML5 document. |
| <html> | It is the root element of an HTML page and it encloses the complete content of the HTML document. |
| <head> | Describes meta-information about the HTML page like title, CSS styles, and encoding type. |
| <title> | Specifies a title for the HTML page. |
| <body> | Defines the document body. It contains all the visible information on the web page. |

**HTML tags:**

**Headings:**

In general documents are starts with heading. To represent headings HTML have six different levels. <h1>, <h2>,<h3>,<h4>,<h5> and <h6>. <h1> refers highest and <h6> refers lowest.

<html>

    <head>

        <title>heading tages</title>

    </head>

    <body>

        <h1>heading 1</h1>

        <h2>heading 2</h2>

        <h3>heading 3</h3>

        <h4>heading 4</h4>

        <h5>heading 5</h5>

        <h6>heading 6</h6>

    </body>

</html>

**Paragraphs:**

The <p> used to organize content into different paragraphs. Each paragraph of text should be represented in between an opening <p> and a closing </p> tag as shown below in the example.

<html>

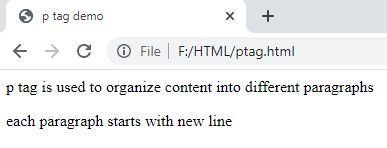
    <body>

        <p>p tag is used to organize content into different paragraphs </p>

        <p>each paragraph starts with new line</p>

    </body>

</html>



**Formatting Elements:**

**Bold Text:**

**<b>** - used to display Bold text

**<strong>** - To represent Important text. Content will display in bold only.

**<i>** - to display text in italic.

**<em>** - used to represent Important text. Content will display in italic only.

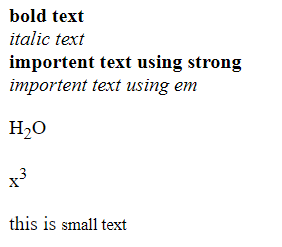
**<small>** - it display 1em smaller than actual font.

**<sub>** - Subscript text

**<sup>** - Superscript text

**<pre>** - Preformatted text.

<html>

    <body>

        <b> bold text</b><br/>

        <i>italic text</i><br/>

        <strong>importent text using strong</strong><br/>

        <em>importent text using em</em><br/>

        <p>H<sub>2</sub>O</p>

        <p>x<sup>3</sup></p>

        <p>this is <small>small text</small></p>

    </body>

</html>

**Hyperlinks:**

It is a connection from one Web resource to another.

Syntax:

<a href=”URL”>Text Link</a>

<html>

    <body>

        <a href="https://www.vectorindia.org/">VectorIndia</a><br/>

        <h1> image as a hyper link</h1>

        <a  href="https://www.vectorindia.org/"><img src="D:\html\logo.jpg" width="100px" height="100px " alt="this is image"/></a>

    </body>

</html>

  <nav> <ul>

                <li><a href="about.html">  About  </a> </li>

                <li><a href="login.html">   Login </a> </li>

                <li><a href="register.html"> Register </a></li>

                </ul>         </nav>

**Bookmarks:**

Bookmarks are useful if a web page is very long. To create bookmarks first create the bookmark, then add a link to it. When the link is clicked, the page will scroll down or up to the location with the bookmark.

<!DOCTYPE html>

<html>

<body>

<p><a href="#s4">Jump to section 4</a></p>

<p><a href="#s10">Jump to section 10</a></p>

<h2>Section 1</h2>

<p>content of section 1</p>

<h2 id="s4">Section 4</h2>

<p>content of section 4</p>

<h2 id="s10">Section 10</h2>

<p>content of section 10</p>

</body>

</html>

NOTE: add 100 to 200 lines of content to each section to see the bookmarks’ effect.

The **target** attribute specifies where to open the linked document.

The target attribute can have one of the following values:

**\_self** - Default. Opens the document in the same window/tab as it was clicked.

**\_blank** - Opens the document in a new window or tab.

**\_parent** - Opens the document in the parent frame.

**\_top** - Opens the document in the full body of the window.

**Frame name** - Opens the documents in the specified frame.

**Images:**

Images enhance the visual appearance of web pages by making them more interesting and colorful. The <img> tag is used to embed an image in a web page.

<img src="url" alt="some\_text">

The alt attribute is used to define an "alternate text" for an image.

**Image maps:**

An image map consists of an image with clickable areas, where you can click on the image, and it will open to a new or the provided destination.

<html>

    <body>

        <img src="bg.jpg" width="400px" height="400px" usemap="#im1"/>

        <map name="im1">

            <area shape="circle" coords="200,100,50" href="https://www.maketecheasier.com/laptop-screen-guide/"/>

            <area shape="circle" coords="150,250,50" href="https://en.wikipedia.org/wiki/Computer\_keyboard"/>

            <area shape="circle" coords="100,350,50" href="https://www.techtarget.com/searchmobilecomputing/definition/touch-pad"/>

        </map>

    </body>

</html>

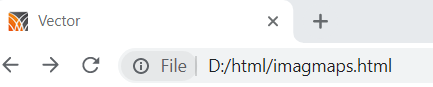
**Favicon:**

 <head>

        <link rel="icon" type="image/x-icon" href="fev.png">

        <title>Vector</title>

    </head>



**Emojis:** Emojis are characters from the UTF-8 alphabet.

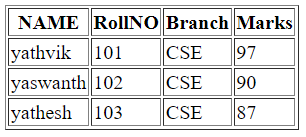
<meta charset="UTF-8">

<p> &#128512; &#128516; &#128525; &#128151; &#128517; </p>

**Tables:**

To arrange data into rows and columns in webpage HTML tables are useful. HTML tables are created using the <table> tag in which the <tr> tag is used to create table rows and <td> tag is used to create data cells.

<html>

    <body>

        <table border="1">

            <tr>

                <th>NAME</th>

                <th>RollNO</th>

                <th>Branch</th>

                <th>Marks</th>

            </tr>

            <tr>

                <td>yathvik</td>

                <td>101</td>

                <td>CSE</td>

                <td>97</td>

            </tr>

            <tr>

                <td>yaswanth</td>

                <td>102</td>

                <td>CSE</td>

                <td>90</td>

            </tr>

            <tr>

                <td>yathesh</td>

                <td>103</td>

                <td>CSE</td>

                <td>87</td>

            </tr>

        </table>

    </body>

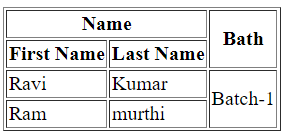
</html>

**Colspan:** To make a cell span over multiple columns, use the colspan attribute.

**rowspan:** To make a cell span over multiple rows, use the rowspan attribute.

<table border="1">

         <tr>

            <th colspan="2">Name</th>

            <th rowspan="2">Bath</th>

        </tr>

        <tr>

            <th>First Name</th>

            <th>Last Name</th>

        </tr>

        <tr>

            <td>Ravi</td>

            <td>Kumar</td>

            <td rowspan="2">Batch-1</td>

        </tr>

        <tr>

            <td>Ram</td>

            <td>murthi</td>

        </tr>

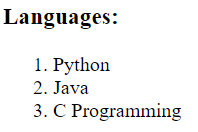
       </table>

**Nested Tables:** You can use one table inside another table. Not only tables you can use almost all the tags inside table data tag <td>.

**Lists:** HTML Lists are used to specify lists of information in a well-formed and semantic way. All lists may contain one or more list elements. There are three different types of HTML lists:

* Ordered List (ol)
* Unordered List (ul)
* Description List or Definition List (dl)

**Ordered List:**

  <h3>Languages:</h3>

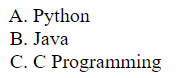
        <ol>

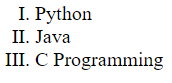
            <li>Python</li>

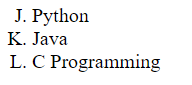
            <li>Java</li>

            <li>C Programming</li>

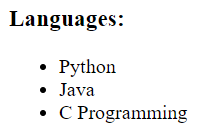
        </ol>

****<ol type="A">

 <ol type="I">

****<ol type="A" start="10">

**Unordered List:**

<h3>Languages:</h3>

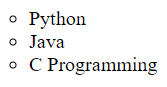
<ul>

  <li>Python</li>

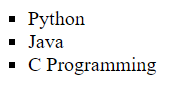
  <li>Java</li>

<li>C Programming</li>

</ul>

****

<ul type="circle">

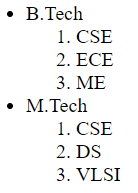
****

 <ul type="square">

**Nested List:**

  <ul>

   <li>B.Tech

      <ol>

      <li>CSE</li> <li>ECE</li>

      <li>ME</li>

</ol> </li>

   <li> M.Tech

      <ol>

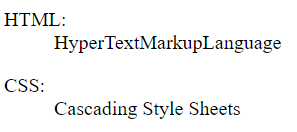
         <li>CSE</li> <li>DS</li> <li>VLSI</li>

      </ol> </li>

</ul>

**Description Lists:**

<body>

   <dl>

       <dt>HTML:</dt>

       <dd>HyperTextMarkupLanguage</dd>

   </dl>

   <dl>

    <dt>CSS:</dt>

    <dd>Cascading Style Sheets</dd>

 </dl>

</body>

**Block-level Elements:** Consume the entire width available irrespective of their sufficiency. They always start in a new line and have top and bottom margins.

**Eg:** <div>, <p>, <h1> etc.

**Inline Elements:** Occupy only enough width that is sufficient to it and allows other elements next to it which are inline. Inline elements don’t start from a new line and don’t have top and bottom margins as block elements have.

Eg: <a>, <span>, <img>, <label> etc.

**Iframe:** iframe is used to display a web page within a web page and it is used to embed documents in the webpage.

<iframe src="url" title="description"></iframe>

<htm>

    <body>

        <iframe width="500px" height="500px" src="D:\content\python.pdf"></iframe>

    </body> </htm>

**Forms:**

HTML Forms are used to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc. <form> tag is used to create an HTML form.

<form action="server url" method="get|post">

Form elements

</form>

**Input Element:** It is used to create form fields, and to take input from users.

 <form>

        <fieldset>

            <legend>input fileds</legend>

            <input type="text" placeholder="text filed"/><br/>

            <input type="text" placeholder="Name" value="to set value"/><br/>

            <input type="password" placeholder="password filed"/><br/>

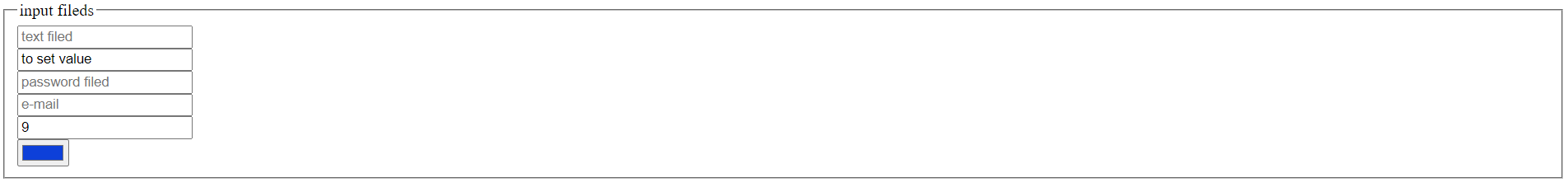
            <input type="email" placeholder="e-mail"/><br/>

            <input type="number" placeholder="Counter"/><br/>

            <input type="color"/><br/>

        </fieldset>

    </form>



 <form>

        <fieldset>

            <legend>Buttons</legend>

            <input type="radio" value="m" name="g1" checked/> Male<br/>

            <input type="radio" value="f" name="g1" /> Female<br/>

            <input type="checkbox" value="t" checked/> Telugu<br/>

            <input type="checkbox" value="h" /> Hindhi<br/>

            <input type="checkbox" value="e" checked/> English<br/>

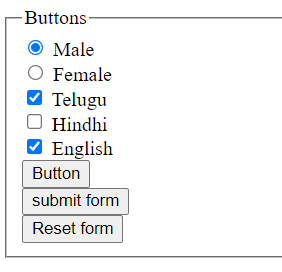
            <input type="button" value="Button"/><br/>

            <input type="submit" value="submit form" /><br/>

            <input type="reset" value="Reset form"/> <br/>

        </fieldset>

    </form>



 <form>

        <fieldset>

            <legend>Textarea and dropdown</legend>

            <textarea rows="10" cols="100">

             Address

            </textarea> <br/>

            <select>

                <option value="1">CSE</option>

                <option value="2">DS</option>

                <option value="3">AI</option>

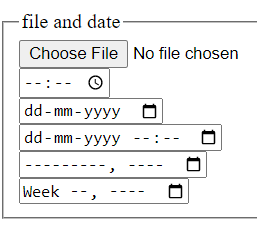
                <option value="4">ML</option>

                <option value="5">IOT</option>

            </select>

        </fieldset>

    </form>



 <form>

        <fieldset>

            <legend>file and date</legend>

            <input type="file"/><br/>

            <input type="time"/><br/>

            <input type="date"/><br/>

            <input type="datetime-local"/><br/>

            <input type="month"/><br/>

            <input type="week"/><br/>

        </fieldset>

    </form>

**Media Tags:** The HTML5 <audio> and <video> tags make it simple to add media to a website. You need to set the src attribute to identify the media source and include a controls attribute so the user can play and pause the media.

**Video:**

<!DOCTYPE html>

<html>

<body>

<video width="420" height="320" controls>

  <source src="gn.mp4" type="video/mp4">

</video>

</body>

</html>

**Auto play option:** add muted along with auto play option to work in chrome.

<video width="420" height="320" controls autoplay muted>

**Audio:**

<audio controls>

  <source src="au.mpeg" type="audio/mpeg">

</audio>

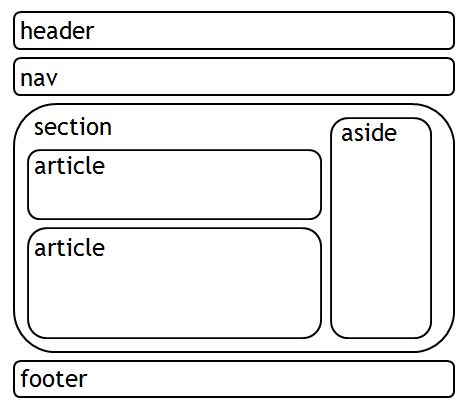
**YouTube playlist creation:**

<iframe width="420" height="345" src="https://www.youtube.com/embed/QjNVgB0S4OQ?playlist=QjNVgB0S4OQ,BirjjxUDQJA,I0vXvsoMMi4,d6fLeVcMmeQ" allowfullscreen>

    </iframe>

**Spellcheck:**

<p contenteditable="true" spellcheck="true">simple example for spellcheck</p>

****

**HTML5 document structure:**

**CSS**

CSS stands for **C**ascading **S**tyle **S**heets. CSS describes how HTML elements are to be displayed on the screen. CSS is used to define styles for web pages, including the design, layout, and variations in display for different devices and screen sizes.

**CSS syntax:**

selector {

property: value;

property: value;

property: value;

}

**Different types of CSS selectors:**

|  |  |
| --- | --- |
| element Selector | p { } |
| id Selector | #i1 { } |
| class Selector | .center { } |
| Universal Selector | \* { } |
| Grouping Selector | h1, h2, p { } |
| Child selectors | body > P { } |
| Attribute selectors | input[type=“text”]{ } |

**Adding CSS to HTML document:**

* External CSS
* Internal CSS
* Inline CSS

**External CSS:**

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

**Internal CSS:**

<style>   
body {

background-color: blue;

}

</style>

**Inline CSS:**

<h1 style="color:blue;text-align:center;">VectorIndia</h1>

**CSS properties:**

**color:** To set foreground color it is useful.

<body>

    <h1 style="color:red">color name</h1>

    <h1 style="color:rgb(234,220,110) ;">with RGB values</h1>

    <h1 style="color:rgba(234,220,110,0.8) ;">with RGB values with opacity</h1>

    <h1 style="color:#ff23a4">With hexadecimal values</h1>

</body>



**Backgrounds:**

background-color:

<body>

    <p style="background-color:blue">Hello Sekhar</p>

</body>

background-image:

<head>

    <style>

         div{

       width:300px;

       height:300px;

       background-image:url('images/logo1.jpg');

       border:2px solid red;

   }

    </style>

</head>

<body>

    <div>

    </div>

</body>

background-repeat:

  background-repeat: repeat-x; background-repeat: repeat-y;



background-repeat: no-repeat;   background-position: center center



Possible values for background-position is: left top, left center, left bottom, right top, right center, right bottom, center top, center center, center bottom, x% y%, and xpos ypos.

**background-attachment:** scroll / fixed

scroll: The background image will scroll with the page. This is the default.

Fixed: The background image will not scroll with the page.

**background-size:** auto/length/cover/contain

auto: Default value. The background image is displayed in its original size

length: Sets the width and height of the background image. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto".

Cover: Resize the background image to cover the entire container.

Contain: Resize the background image to make sure the image is fully visible.

background - Shorthand property: The order of the property values is:

background-color, background-image, background-repeat, background-attachment, background-position.

background: #0032f1 url("sam.jpg") no-repeat right top;

**background-clip:** border-box/padding-box/content-box

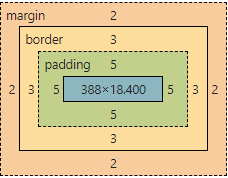
border-box: The border-box property is used to set the background color spread over the whole division.

padding-box: The padding-box property is used to set the background inside the border.

content-box: The content-box property is used to set the background color up to the content only.

**Box Model:**

According to the standard CSS basic box model, each element is surrounded by a rectangular box. It contains multiple properties including borders, margin, padding, and the content itself.



**Margins:** Margin properties are used to create space around elements

CSS has properties for specifying the margin for each side of an element: margin-top, margin-right, margin-bottom and margin-left possible values are auto or length or %.

Eg: margin: 5px 3px 7px 10px;

Top, right, bottom, left values.

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

properties for specifying the padding for each side of an element: padding-top, padding-right, padding-bottom, padding-left.

Eg: padding: 5px 6px 7px 10px;

height and width properties are used to set the height and width of an element. height and width properties do not include padding, borders, or margins. max-width property is used to set the maximum width of an element.

The border properties are used to specify how the border of the box representing an element should look. There are three properties of a border. border-color specifies the color of a border, border-style specifies whether a border should be solid, dashed line, double line, or one of the other possible values, border-width specifies the width of a border.

To specify border-color in each side border-bottom-color, border-top-color, border-left-color, border-right-color.

To specify border-style in each side border-bottom-style, border-top-style, border-left-style, border-right-style.

 p {

                border: 3px solid red;

                margin: 2px;

                padding: 5px;

            }

**Formatting text:**

<html>     <head>         <style>

           .left{

               text-align: left;

               color:blue;

               font-size: 15px;

               font-weight: 100;

               font-family: Verdana, Geneva, Tahoma, sans-serif;

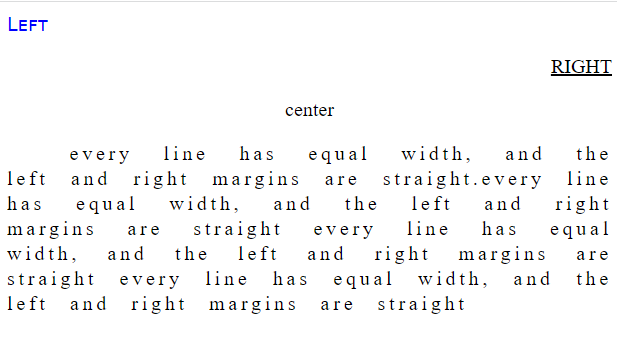
               font-variant: small-caps;

           }

           .right{

               text-align: right;

               text-decoration: underline;

               text-transform:uppercase;

           }

           .center{

               text-align: center;

           }

           .justify{

               text-align: justify;

               text-indent: 50px;

               line-height: 20px;

               letter-spacing: 3px;

               word-spacing: 10px;

           }

        </style>

    </head>

    <body>

        <p class="left">Left</p>

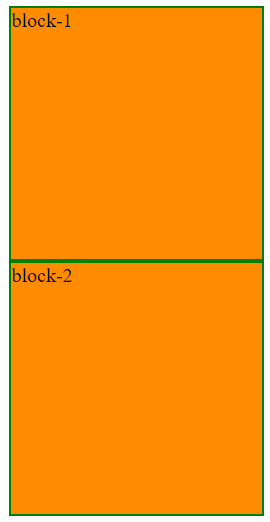
        <p class="right">right</p>

        <p class="center">center</p>

        <p class="justify">every line has equal width, and the left and right margins are straight.every line has equal width, and the left and right margins are straight every line has equal width, and the left and right margins are straight every line has equal width, and the left and right margins are straight </p>

    </body>

</html>

**display:** the display property determines how an element looks. The default display value for most of the elements is either block or inline.

**display:block**

<style>

        div{

            display:block;

            width:200px;

            height:200px;

            background-color: darkorange;

            border: 2px solid green;

            }

    </style>

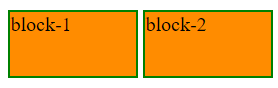
<div>block-1</div>

<div>block-2</div>

**display:inline**

inline will discard width and height of the element.

display:inline;

**display:inline-block**

 display:inline-block;

width:100px;

 height:50px;

flex: Displays an element as a block-level flex container

inline-flex: Displays an element as an inline-level flex container

none: The element is completely removed

visibility:hidden also hides an element. However, the element will still take up the same space as before.

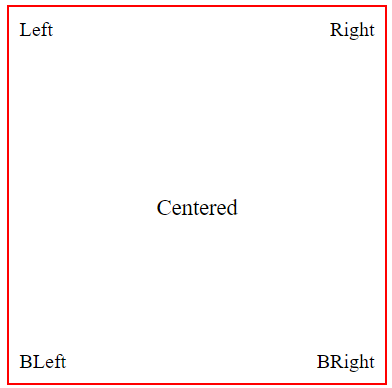
Block-level element always takes up the full width available. Setting the width of a block-level element will prevent it from stretching out to the edges of its container. to set the element horizontally center within its container set margin auto. when the browser window is smaller than the width of the element then the browser adds a horizontal scrollbar to the page. To avoid this use max-width property.

**Position Property:**

position property sets how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of positioned elements.

<html>

<head>

<style>

.container {

  position: relative;

  width:300px;

  height:300px;

  border: 2px solid red;

  margin: auto;

}

.left{

    position:absolute;

    top: 8px;

    left: 8px;

}

.right{

    position:absolute;

    top: 8px;

    right: 8px;

}

.bleft{

    position:absolute;

    bottom:8px;

    left: 8px;

}

.bright{

    position:absolute;

    bottom: 8px;

    right: 8px;

}

.center {

  position: absolute;

  top: 50%;

  width: 100%;

  text-align: center;

  font-size: 18px;

}

</style>

</head> <body>

<div class="container">

    <div class="left">Left</div>

    <div class="right">Right</div>

  <div class="center">Centered</div>

  <div class="bleft">BLeft</div>

  <div class="bright">BRight</div>

</div>

</body>

</html>

position: static

HTML elements are positioned static by default. Statically positioned elements are not affected by the top, bottom, left, and right properties.

position: relative

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.

position: fixed

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

position: absolute

An element with position: absolute; is positioned relative to the nearest positioned ancestor. if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

z-index:

The z-index property specifies the stack order of an element. z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky) and flex items.

<html>

    <style>

        div{

            width:100px; height:100px; position: absolute;

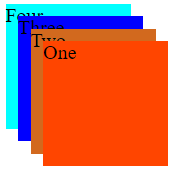
        }

        #d4{

            background-color:aqua;

            top:10px;

            left:10px;

            z-index: -1;

        }

        #d3

        {

            background-color:blue;

            top:20px;

            left:20px;

            z-index: 2;

        }

        #d2

        {

            background-color:chocolate;

            top:30px;   left:30px; z-index: 3;

        }

        #d1

        {

           background-color:orangered;

            top:40px;     left:40px; z-index: 4;

        }

    </style>

    <body>

        <div id="d1">One</div>

        <div  id="d2">Two</div>

        <div  id="d3">Three</div>

        <div  id="d4">Four</div>

    </body>

</html>

**Overflow:** overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area. The overflow property has the following values: visible, hidden, scroll, and auto.

**Links:**

* a:link - a normal, unvisited link
* a:visited - a link the user has visited
* a:hover - a link when the user mouses over it
* a:active - a link the moment it is clicked

**Lists:**

* list-style-type: circle/square/ upper-roman/lower-alpha/none
* list-style-position: outside/ inside

**Tables:**

Table Size: width and height properties are used to set table width.

To remove double borders border-collapse: collapse; property is useful.

Alignment: text-align property sets the horizontal alignment like left, right, or center of the content in <th> or <td>. vertical-align property sets the vertical alignment like top, bottom, or middle of the content in <th> or <td>.

**Forms:**

<!DOCTYPE html>

<html>

<head>

    <link rel="stylesheet" href=

"https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

    <style>

        input[type=text],

        input[type=password] {

            width: 100%;

            padding: 12px 40px;

            margin: 8px 0;

            display: inline-block;

            border: 1px solid #ccc;

            box-sizing: border-box;

        }

        button {

            background-color: #4CAF50;

            color: white;

            padding: 14px 20px;

            margin: 8px 0;

            border: none;

            cursor: pointer;

            width: 100%;

        }

        button:hover {

            opacity: 0.8;

        }

        .container {

            padding: 16px;

        }

        .fontuser {

            position: relative;

        }

        .fontuser i{

            position: absolute;

            left: 15px;

            top: 40px;

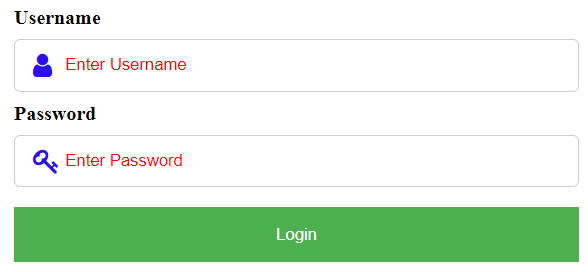
            color: rgb(44, 15, 236);

        }

        .fontpassword {

            position: relative;

        }



        .fontpassword i{

            position: absolute;

            left: 15px;

            top: 40px;

            color: rgb(44, 15, 236);

        }

        ::placeholder

        {

            color:red;

        }

  input{

            border-radius: 5px;

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="fontuser">

            <label><b>Username</b></label>

            <input type="text"

                    placeholder="Enter Username"

                    name="uname" required>

            <i class="fa fa-user fa-lg"></i>

        </div>

        <div class="fontpassword">

            <label><b>Password</b></label>

            <input type="password"

                    placeholder="Enter Password"

                    name="psw" required>

            <i class="fa fa-key fa-lg"></i>

        </div>

        <button type="submit">Login</button>

    </div>

</body>

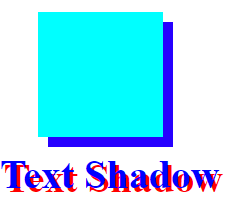
</html>

**Shadow Effects:**

<htm>

    <head>

        <style>

            div{

                width:100px;

                height:100px;

                position: relative;

                top:30px;

                left:30px;

                background-color: aqua;

                box-shadow: 8px 8px rgb(38, 0, 255);

            }

            h1{

                text-shadow: 3px 3px red;

                color:blue;

                position:relative;

                top:20px;

            }

        </style>

    </head>

    <body>

        <div>

        </div>

        <h1>Text Shadow</h1>

    </body>

</htm>

**Transitions:**

To create a transition effect the following CSS properties are important. One is duration and other one is effect.

transition: [transition-property] [transition-duration] [transition-timing-function] [transition-delay];

  div {

  width: 200px;

  height: 200px;

  background: blue;

  transition: linear 3s; }

div:hover {

  width: 400px;

}

The transition-timing-function property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

* ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
* linear - specifies a transition effect with the same speed from start to end
* ease-in - specifies a transition effect with a slow start
* ease-out - specifies a transition effect with a slow end
* ease-in-out - specifies a transition effect with a slow start and end
* cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

**2D Transforms:**

With the CSS transform property following 2D transformation methods can able to use:

* translate(): transform: translate(150px, 10px);
* rotate():  transform: rotate(-60deg);
* scaleX(): transform: scaleX(1.2);
* scaleY():  transform: scaleY(1.4);
* scale():  transform: scale(1.2, 0.8);
* skewX(): transform: skewX(10deg);
* skewY(): transform: skewY(10deg);
* skew(): transform: skew(10deg, 5deg);

**Animations:**

The animation used to change elements from one style to another. As many CSS properties and as many times can able to change. Keyframes need to specify to use of CSS animation. Keyframes hold what styles the element will have at certain times.

#sl1

{

    width: 100%;

    height:100%;

    animation: zoom  3s linear  infinite;

}

@keyframes zoom {

    0%{

        transform: scale(1.3);

    }

    15%{

        transform: scale(1);

    }

    85%{

        transform: scale(1);

    }

    100%{

        transform: scale(1.3);

    }

}

**Media queries:**

media queries to create different layouts for different screen sizes and devices. To create a two-column layout for most screen sizes, and a one-column layout for small screen sizes. Change the flex-direction from row to column at a specific breakpoint.

 .container {

  display: flex;

  flex-direction: row;

  font-size: 20px;

  text-align: center;

}

.fleft {

  background-color: #ef0eaf;

  padding: 10px;

  flex: 50%;

}

.fright {

  background-color: rgb(30, 255, 143);

  padding: 10px;

  flex: 50%;

} @media (max-width: 600px) {

  .container {

    flex-direction: column;

  } }

Flex Container:

The flex container becomes flexible by setting the display property to flex. The flex-direction property defines in which direction the container wants to stack the flex items. The column value stacks the flex items vertically. row value stacks the flex items horizontally. The flex-wrap property specifies whether the flex items should wrap or not.

**Bootstrap**

Bootstrap is the most popular HTML, CSS, and JavaScript framework for creating responsive and mobile-first websites.

CDN:

<https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css>

<https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.bundle.min.js>

Bootstrap is developed mobile-first, a strategy in which they optimize code for mobile devices first and then scale up components as necessary using CSS media queries. To ensure proper rendering and touch zooming for all devices, add the responsive viewport meta tag to your <head>.

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

**Containers:**

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

<div class="container bg-primary">

hello to all

</div>

.container-sm /md / lg / xl classes to determine when the container should be responsive. The max-width of the container will change on different screen sizes

**Grid System:**

Bootstrap’s grid system uses a series of containers, rows, and columns to layout and aligns content. It’s built with a flexbox and is fully responsive. In a grid layout, content must be placed within columns and only columns may be immediate children of rows. Column classes indicate the number of columns used out of the possible 12 per row. To specify three equal-width columns .col-4 class is useful.

Grid breakpoints are based on minimum width media queries, meaning they apply to that one breakpoint and all those above it (e.g., .col-sm-4 applies to small, medium, large, and extra-large devices, but not the first xs breakpoint).

<!DOCTYPE html>

<html lang="en">

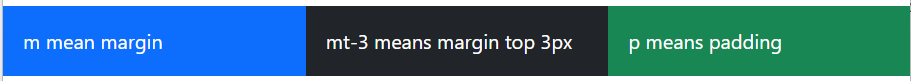
<head>

  <title>Bootstrap Example</title>

  <meta charset="utf-8">

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

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container-fluid mt-3">

  <div class="row">

    <div class="col-sm-4 bg-primary text-white p-3 ">

      m mean margin

    </div>

    <div class="col-sm-4 bg-dark text-white p-3">

        mt-3 means margin top 3px

    </div>

    <div class="col-sm-4 bg-success text-white p-3">

        p means padding

    </div>

  </div>

</div>

</body>

</html>

**Typography:**

.text-start: Indicates left-aligned text

.text-break: Prevents long text from breaking layout

.text-center : Indicates center-aligned text

.text-decoration-none: Removes the underline from a link

.text-end: Indicates right-aligned text

.text-nowrap: Indicates no wrap text

.text-lowercase: Indicates lowercased text

.text-uppercase: Indicates uppercased text

.text-capitalize: Indicates capitalized text

<ul class="list-unstyled"> Remove the default list-style and left margin on list items (immediate children only).

**Text colors:** following is the list of classes to represent predefined text colors.

.text-muted, .text-primary, .text-success, .text-info, .text-warning, .text-danger, .text-secondary, .text-white, .text-dark, and .text-light

**Background Colors:**

The classes for background colors are: .bg-primary, .bg-success, .bg-info, .bg-warning, .bg-danger, .bg-secondary, .bg-dark and .bg-light.

<div class="container-fluid mt-3 bg-danger text-info">

  Hello Sekhar

</div>

**Tables:**

Add the base class .table to any <table>, then extend with optional modifier classes or custom styles.

Use contextual classes to color table rows or individual cells.

<tr class="table-primary">for complete row</tr>

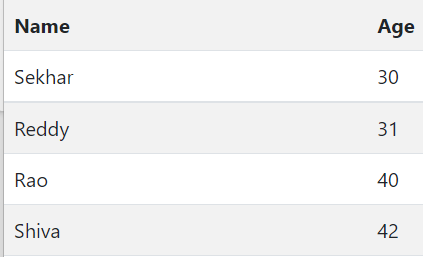
<td class="table-secondary">for single cell</td>

Use .table-striped to add zebra-striping to any table rows.

Use .table-striped-columns to add zebra-striping to any table column.

The .table-hover class adds a hover effect on table rows.

<table class="table table-striped table-hover">

    <tr><th>Name</th><th>Age</th></tr>

    <tr><td>Sekhar</td><td>30</td></tr>

    <tr><td>Reddy</td><td>31</td></tr>

    <tr><td>Rao</td><td>40</td></tr>

    <tr><td>Shiva</td><td>42</td></tr>

</table>

**Responsive images:**

Images in Bootstrap are made responsive with .img-fluid. max-width: 100%; and height: auto; are applied to the image so that it scales with the parent element. Class .img-thumbnail to give an image a rounded 1px border appearance.

<img src="..." class="rounded float-start" alt="...">

<img src="..." class="rounded float-end" alt="...">

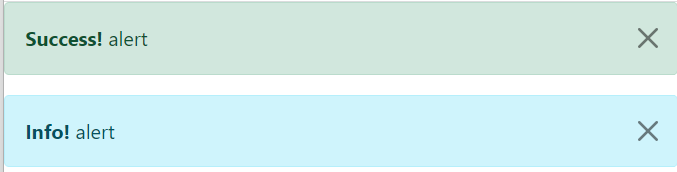
<img src="..." class="rounded mx-auto d-block" alt="...">

Float an image to the left with the .float-start class or to the right with .float-end. The .rounded class adds rounded corners to an image. The .rounded-circle class shapes the image to a circle. Center an image by adding the utility classes .mx-auto (margin:auto) and .d-block (display:block) to the image.

Eg: <img src="bg1.jpg" class="rounded-circle img-fluid"/>

**Alerts:**

The button with class="btn-close" and data-bs-dismiss="alert" is used to close the alert box. The alert-dismissible class aligns the button to the right.



<div class="alert alert-success alert-dismissible">

  <button type="button" class="btn-close" data-bs-dismiss="alert"></button>

  <strong>Success!</strong> alert

</div>

<div class="alert alert-info alert-dismissible">

  <button type="button" class="btn-close" data-bs-dismiss="alert"></button>

  <strong>Info!</strong> alert

</div>

**Buttons:**  Bootstrap includes several predefined button styles, each serving its own semantic purpose.

<button type="button" class="btn btn-primary">Primary</button>

<a class="btn btn-primary" href="#" role="button">Link</a>

<button type="button" class="btn btn-outline-primary">Primary</button>

<button type="button" class="btn btn-primary btn-lg btn-block">Block level button</button>

Wrap a series of buttons with .btn in .btn-group.

<div class="btn-group">  
  <button type="button" class="btn btn-primary">First</button>  
  <button type="button" class="btn btn-primary">Second</button>  
  <button type="button" class="btn btn-primary">Third</button>  
</div>

the class .btn-group-vertical to create a vertical button group

<div class="btn-group-vertical">  
  <button type="button" class="btn btn-primary">First</button>  
  <button type="button" class="btn btn-primary">Second</button>  
  <button type="button" class="btn btn-primary">Third</button>  
</div>

**Forms:**

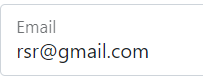
<form>

  <div class="input-group">

    <span class="input-group-text">@</span>

    <input type="email" class="form-control" placeholder="Username">

  </div> </form>

**Floating Labels:**

  <div class="form-floating mb-3 mt-3">

    <input type="text" class="form-control" id="email" placeholder="Enter email" name="email">

    <label for="email">Email</label>

  </div>

**Badges:** Badges can be used as part of links or buttons to provide a counter.

**** <button type="button" class="btn btn-primary">

    Add <span class="badge text-bg-secondary  bg-danger">5</span>

  </button>

**Progress:**

<div class="progress">

  <div class="progress-bar progress-bar-striped progress-bar-animated" role="progressbar" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100" style="width: 75%"></div>

</div>



**Spinners:**

Bootstrap spinners can be used to show the loading state in your projects.



<button class="btn btn-primary" type="button">

  <span class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span>

  Loading...

</button>

**Pagination:**

To create a basic pagination, add the .pagination class to an <ul> element. Then add the .page-item to each <li> element and a .page-link class to each link inside <li>.

<nav aria-label="Page navigation example">

  <ul class="pagination">

    <li class="page-item">

      <a class="page-link" href="#" aria-label="Previous">

        <span aria-hidden="true">&laquo;</span>

      </a>

    </li>

    <li class="page-item"><a class="page-link" href="#">1</a></li>

    <li class="page-item"><a class="page-link" href="#">2</a></li>

    <li class="page-item"><a class="page-link" href="#">3</a></li>

    <li class="page-item">

      <a class="page-link" href="#" aria-label="Next">

        <span aria-hidden="true">&raquo;</span>

      </a>

    </li>

  </ul>

</nav>

**Javascript**

JavaScript is a dynamic programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side scripts to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities. It is used to modify the HTML content.

In HTML, JavaScript code is inserted between <script> and </script> tags. You can place any number of scripts in an HTML document. Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both. The script tag provides a mechanism to allow you to store JavaScript in an external.

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

file and then include it into your HTML files.

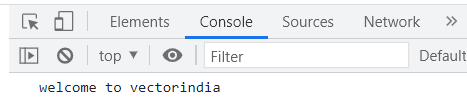
JavaScript can "display" data in different ways:

* Writing into an HTML element, using innerHTML.
* Writing into the HTML output using document.write().
* Writing into an alert box, using window.alert().
* Writing into the browser console, using console.log().

<body>

<script>

console.log("welcome to vectorindia")

</script>

</body>



document.write("hai to all");

**Using innerHTML:**

<html>

<body>

<p id="p1"></p>

<script>

id=document.getElementById("p1");

id.innerHTML="Hello Sekhar";

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

**Alert:**

window.alert("welcome to JS");

**Datatypes:**

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

**Numbers:** Any number, such as 17, -21, 3.14, or 54e7

**Strings**: "Greetings!" or "Fun" or ‘hello’

**Boolean**: Either true or false

**Null**: A special keyword for exactly that – the null value (that is, nothing)

**Arrays:** const a = ["reddy", "suman", "santhi"];

**Objects:** {firstname:”reddy”,lastname:”Sekhar”,sal:90000}

JavaScript typeof operator is used to find the type of a JavaScript variable.

**Variables:**

Four ways to declare JavaScript variables are:

* Using var

var x = 10;

var y = 20;

var z = x + y;

* Using let

let x = 10;

let y = 20;

let z = x + y;

* Using const

const x = 10;

* Using nothing

x=230

Variables defined with let cannot be redeclared. But with var it is possible. Variables defined with let have Block Scope. But if the variable defined with var can be accessible outside of block also. Variables defined with const cannot be Reassigned.

**Operators:**

Arithmetic Operators:

+, -, \*, /, %, \*\*, ++, --

**Relational operators:**

<, >, <=, >=, !=, ===(value and type), !==( not equal value and type).

**Logical Operators:**

&&, ||, !

**Assignment Operators:**

=, +=, -=, \*=

**Bitwise Operators:**

&, |, ^, <<, >>, >>>, ~

**Ternary operator:**

(cond)?exp1 :exp2

**Conditional Statements:**

**if Statement:**

if(cond)

{

//statements need to be executed when the condition is true

}

**if else:**

if (condition) {

// block of code to be executed if the condition is true

}

else {

// block of code to be executed if the condition is false

}

**Else if ladder:**

if (condition1) {

// block of code to be executed if condition1 is true

} else if (condition2) {

// block of code to be executed if the condition1 is false and condition2 is true

} else {

// block of code to be executed if the condition1 is false and condition2 is false

}

**Switch Statement:**

switch(expression) {

case x:

// code block

break;

case y:

// code block

break;

default:

// code block

}

Example:

let a=20,b=25,c=15;

if(a>b && a>c)

{

    console.log(a);

}

else if(b>c)

{

    console.log(b);

}

else

{

console.log(c);

}

c=2

switch(c)

{

    case 1:

        console.log("you selected 1");

        break;

    case 2:

        console.log("you selected 2");

        break;

    default:

        console.log("default case");

}

**Loop:**

for - loops through a block of code a number of times

for(let i=0;i<5;i++)

{

    console.log(i);

}

for-in - loops through the properties of an object

let s={"name":"sekhar","age":25,"sal":78000}

for(let i in s)

{

    console.log(s[i])

}

let a=[1,2,3,4,5,6]

for(let i in a)

{

    console.log(a[i]);

}

Foreach:

let a=[1,2,3,4,5,6]

a.forEach(element => {

    console.log(element)

});

for-of - loops through the values of an iterable object

a=[1,2,3,4,5,6]

for(i of a)

{

    console.log(i)

}

s='hello'

for(i of s)

{

    console.log(i)

}

while - loops through a block of code while a specified condition is true

let i=0

while(i<5)

{

    console.log(i)

    i++

}

do-while - also loops through a block of code while a specified condition is true.

let i=0

do

{

    console.log(i)

    i++

}while(i<5);

Functions:

Functions are used for code reusability. The most common way to define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters, and a statement block surrounded by curly braces.

Eg:

function add(a,b)

{

    console.log(a+b)

}

add(2,7)

**default arguments:**

function fun(a,b=0,c=0)

{

    console.log(a+b+c)

}

fun(3)

fun(3,4)

fun(3,4,5)

**Objects:** The object is an entity having a state and behavior. In JS objects are created in three ways

* using object literal
* By creating an instance of Object directly (using a new keyword)
* By using an object constructor (using a new keyword)

**Object by object literal:**

std={'id':101,'name':'RSR','marks':90}

console.log(std.name)

**Creating an instance of Object:**

let s=new Object();

s.name='yathvik'

s.age=30

console.log(s)

**using an Object constructor:**

function emp(name,age,sal)

{

    this.name=name

    this.age=age

    this.sal=sal

}

let e=new emp('RSR',30,100000)

console.log(e)

**Array:**

a=[1,2,3,4,5]

for(let i=0;i<a.length;i++)

{

    console.log(a[i]) }

//sorting array elements

a=[5,3,4,2,1,9]

a.sort()

console.log(a)

//adding array elements

a=[3,2,4,5,1]

a.push(10)

a[a.length]=20

console.log(a)

The join() method is used to joins all array elements into a string. console.log(a.join('-'))a=[3,4,5,6]

//deleting element

console.log(a.pop())

//deleting first element

console.log(a.shift())

console.log(a)

//adding element at start

a.unshift(20)

console.log(a)

filter() method creates a new array with all elements that pass the test implemented by the provided function.

a=[1,2,3,4,5,6,7,8]

console.log(a.filter(x=>x%2==0))

map() method creates a new array with the results of calling a provided function on every element in this array.

a=[1,2,3,4,5]

console.log(a.map(x=>x\*x))

reduce() method applies a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.

a=[1,2,3,4,5,6]

console.log(a.reduce((x,y)=>x\*y))

**Strings:** The string is zero or more characters written inside quotes.

|  |  |  |
| --- | --- | --- |
| Method | Description | example |
| length | To know the length of a string | s='hello to all'  console.log(s.length) |
| charAt() | To display character at specific index. | s='hello to all'  console.log(s.length) |
| charCodeAt() | number indicating the Unicode value of the character at the given index. | s='hello to all'  console.log(s.charCodeAt(2)) |
| indexOf() | Returns substring index position or -1 if it is not available. | s='hai how are you'  console.log(s.indexOf('how')) |
| split() | To divide a given string into tokens based on a given delimiter. | s='hai how are you'  console.log(s.split(" ")) |
| toUpperCase()  toLowerCase() | To convert given string into lower or uppercase. | s="hai To All"  console.log(s.toUpperCase())  console.log(s.toLowerCase()) |
| toString() | To convert given object into string. | a=1234  s=a.toString()  console.log(s) |
| valueOf() | To convert the given string into primitive datatype. | n=a.valueOf(s)  console.log(n) |
| replace() | replaces a specified value with another value in a string. | s="hello to all"  console.log(s.replace("to","for")) |

Sets: set doesn’t allow duplicate elements. If input contain duplicates it consider only once.

s=new Set([1,2,3,2,1,2,4,5,7,7,6])

s.add(9)

s.delete(5)

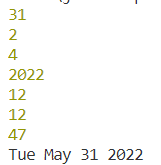
console.log(s.size)

console.log(s)

**Date:**

let d=new Date()

console.log(d.getDate())

console.log(d.getDay())

console.log(d.getMonth())

console.log(d.getFullYear())

console.log(d.getHours())

console.log(d.getMinutes())

console.log(d.getSeconds())

console.log(d.toDateString())

**Regular Expression:**

A regular expression is a sequence of characters that forms a search pattern. Regular expression defined using var pattern = /pattern/attributes;

|  |  |
| --- | --- |
| **Expression** | **Description** |
| [aeiou] | any of the characters between the brackets |
| [^aeiou] | Other than the characters between the brackets |
| [0-9] | any decimal digit from 0 through 9 |
| [a-z] | any character from lowercase a through lowercase z |
| [A-Z] | character from uppercase A through uppercase Z |
| [a-Z] | any character from lowercase a through uppercase Z |
| + | any string containing one or more occurrence |
| \* | any string containing zero or more occurrence |
| ? | At most one occurrence. |
| {N} | A string containing a sequence of N occurrences |
| {M,N} | M to N occurrences |
| {M,} | M or more occurrences |
| p$ | Ending with |
| ^p | Starting with |
| . | Single character |
| \s | whitespace character |
| \S | Non-whitespace character |
| \d | a digit (0-9) |
| \D | a non-digit |
| \w | a word character (a-z, A-Z, 0-9, \_) |
| \W | non-word character |
| (x|y|z) | matches any of the alternatives specified |

**Modifiers:**

i - Perform case-insensitive matching.

g - Performs a global match that is, find all matches rather than stopping after the first match.

**Regular Expression Methods:**

**exec():**

The exec method searches string for text that matches regexp. If it finds a match, it returns an array of results; otherwise, it returns null.

a="welcome to all of you to learn full stack"

console.log(/ll/.exec(a))

**test():** test method searches string for a text that matches regexp. If it finds a match, it returns true; otherwise, it returns false.

a="welcome to all of you to learn full stack"

console.log(/ll/.test(a))

**DOM:**

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.



**HTML DOM Methods:**

A property is a value that can be able to get or set. A method is an action that can do.

getElementById: To access an HTML element id of the element is used.

innerHTML: To change the content of HTML element innerHTML property is useful.

<html>     <body>

        <div id="i1">

        </div>

        <script>

            let r=document.getElementById("i1")

            r.innerHTML="New content for div"

        </script>

    </body> </html>

**Finding HTML Elements:**

|  |  |
| --- | --- |
| Method | Description |
| document.getElementById(id) | Find an element by element id |
| document.getElementsByTagName(name) | Find elements by tag name |
| document.getElementsByClassName(name) | Find elements by class name |

**Changing HTML Elements:**

|  |  |
| --- | --- |
| **Property** | **Description** |
| element.innerHTML =  new html content | Change the inner HTML of an element |
| element.attribute = new value | Change the attribute value of an HTML element |
| element.style.property = new style | Change the style of an HTML element |
| **Method** | **Description** |
| element.setAttribute(attribute, value) | Change the attribute value of an HTML element |

<html>

    <body>

        <img src="C:\Users\irsr5\OneDrive\Desktop\jsim.webp" with="200" height="100" onmouseover="fun(this)" onmouseleave="fun1(this)"/>

        <script>

           function fun(e)

           {

               e.src="C:\\Users\\irsr5\\OneDrive\\Desktop\\js.png"

           }

           function fun1(e)

           {

               e.src="C:\\Users\\irsr5\\OneDrive\\Desktop\\jsim.webp"

           }

****

        </script>

    </body>

</html>

**Changning the CSS style:**

****  <div onmouseenter="fun(this)"> hover to change text color</div>

        <script>

            function fun(e)

            {

                e.style.color="blue";

            }

        </script>

Events:

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page. When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc. few frequently used events are:

|  |  |
| --- | --- |
| **Event** | **Description** |
| onblur | Triggers when the window loses focus |
| onchange | Triggers when an element changes |
| onclick | Triggers on a mouse click |
| oncontextmenu | Triggers when a context menu is triggered |
| ondblclick | Triggers on a mouse double-click |
| ondrag | Triggers when an element is dragged |
| ondragend | Triggers at the end of a drag operation |
| ondragenter | Triggers when an element has been dragged to a valid drop target |
| ondragleave | Triggers when an element is being dragged over a valid drop target |
| ondragover | Triggers at the start of a drag operation |
| ondragstart | Triggers at the start of a drag operation |
| ondrop | Triggers when dragged element is being dropped |
| ondurationchange | Triggers when the length of the media is changed |
| onerror | Triggers when an error occur |
| onfocus | Triggers when the window gets focus |
| onformchange | Triggers when a form changes |
| onforminput | Triggers when a form gets user input |
| onhaschange | Triggers when the document has change |
| oninput | Triggers when an element gets user input |
| onkeydown | Triggers when a key is pressed |
| onkeypress | Triggers when a key is pressed and released |
| onkeyup | Triggers when a key is released |
| onload | Triggers when the document loads |
| onmessage | Triggers when the message is triggered |
| onmousedown | Triggers when a mouse button is pressed |
| onmousemove | Triggers when the mouse pointer moves |
| onmouseout | Triggers when the mouse pointer moves out of an element |
| onmouseover | Triggers when the mouse pointer moves over an element |
| onmouseup | Triggers when a mouse button is released |
| onmousewheel | Triggers when the mouse wheel is being rotated |
| onpause | Triggers when media data is paused |
| onplay | Triggers when media data is going to start playing |
| onplaying | Triggers when media data has start playing |
| onprogress | Triggers when the browser is fetching the media data |
| onratechange | Triggers when the media data's playing rate has changed |
| onreadystatechange | Triggers when the ready-state changes |
| onresize | Triggers when the window is resized |
| onscroll | Triggers when an element's scrollbar is being scrolled |
| onselect | Triggers when an element is selected |
| onsubmit | Triggers when a form is submitted |

 <form>

            <input type="text" value="" onchange="fun1(this)" onfocus="fun2(this)"  onblur="fun3(this)"/>

        </form>

        <script>

            function fun1(e)

            {

                e.value=e.value.toUpperCase();

            }

            function fun2(e)

            {

                e.style.background="yellow";

                e.style.color="red";

            }

            function fun3(e)

            {

                e.style.background="white";

            }

        </script>

Adding and removing events:

 <button id="b1">alert</button>

        <button onclick="fun()">add event</button>

        <button onclick="fun1()">Remove event</button>

        <script>

            let f=()=>alert("event added")

            function fun()

            {

                e=document.getElementById("b1")

                e.addEventListener("click",f)

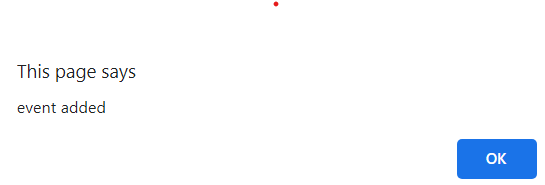
            }

            function fun1()

            {

                e=document.getElementById("b1")

                e.removeEventListener("click",f)

            }

        </script>

**Timing Events:**

The window object allows execution of code at specified time intervals.

The two key methods to use with JavaScript are:

**setTimeout(function, milliseconds)**

Executes a function, after waiting a specified number of milliseconds.

**setInterval(function, milliseconds)**

Same as setTimeout(), but repeats the execution of the function continuously.

**Digital clock using setInterval():**

 <div id="c1"></div>

        <script>

            setInterval(fun, 1000);

            function fun()

            {

                let d=new Date()

                document.getElementById("c1").innerHTML=d.toLocaleTimeString()

            }

       </script>

**JavaScript sessionStorage:**

The sessionStorage object stores data only for a session. It means that the data stored in the sessionStorage will be deleted when the browser is closed.

* setItem(name, value) – set the value for a name
* removeItem(name) – remove the name-value pair identified by name.
* getItem(name) – get the value for a given name.
* key(index) – get the name of the value in the given numeric position.
* clear() – remove all values in the sessionStorage

**localStorage:**

localStorage is a way to store data with no expiration date in the web browsers. In other words, the data stored in the browsers will persist even after closing the browser windows.

* setItem(name, value) – set the value for a name
* removeItem(name) – remove the name-value pair identified by name.
* getItem(name) – get the value for a given name.
* key(index) – get the name of the value in the given numeric position.
* clear() – remove all values.

**Templet literals:**

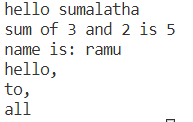
//templet literals or templet strings

//used to represent multiline strings or used to represent variables and expressions in between the strings.

//templet literals are enclosed with backticks(``)

let name='sumalatha'

console.log(`hello ${name}`)

s=`sum of 3 and 2 is ${3+2}`

console.log(s)

let std={name:'ramu',age:20,gen:'m'}

console.log(`name is: ${std.name}`)

console.log(`hello,

to,

all`)

**this usage:**

//inside object method this mean current object

let f={

    name:'ravi',

    fun:function()

    {

        console.log(this)

    }

}

f.fun()

//in regular functions this refers window object

function f1()

{

    console.log(this)

}

f1()

**Spread operator:**

// spread operator is used to unpack iterables into individuals mainly to create copy and concatenation purpose.

let a=[2,3,4,5,6,7]

console.log(...a)

let b=[1,...a,8,9,10]

console.log(...b)

let c=[...a,...b]

console.log(...c)

// spread operator is used to copy data from one object to other object

let std={name:'ravai',age:20,gen:'male',marks:90}

let nstd={...std}

console.log(std)

console.log(nstd)

console.log({...nstd})

nstd={...std,city:'HTD'}

console.log(nstd)

//spread operator with function calls

function add(a,b,c)

{

    console.log(a+b+c)

}

a=[1,2,3]

add(...a)

//extra values will be ignored if you pass more than required number of parameters

a=[3,4,5,6,7,8,9]

add(...a)

**Rest operator:**

//to combine all the arguments into a single value

function fun(...x)

{

    console.log(x)

}

fun(1,2,3,4,5,6)

//rest parameter should be the last

function fun1(a,b,...c)

{

    console.log(a)

    console.log(b)

    console.log(c)

}

fun1(2,3,4,5,6)

function sum(...a)

{

    let s=0

    for(let e of a)

    {

        s+=e

    }

    console.log(s)

}

sum(2,3,4,5,6,7,8)

//destructruing arrays

let [x,y,z]=[1,2,3]

console.log(x)

console.log(y)

console.log(z)

let [a,b, ...c]=[1,2,3,4,5,6]

console.log(a)

console.log(b)

console.log(c)

//destructuring the object

let {name,...rest}={name:'ramu',age:20,gen:'m'}

console.log(name)

console.log(rest)

**Arrow functions:**

//function declaration

function fun(a,b)

{

    console.log(a+b)

}

fun(4,8)

//function expression

let sum=function(a,b){

    console.log(a+b)

}

sum(4,5)

//arrow functions are alternate for function expressions

let f=(a,b)=>{

    console.log(a+b)

}

f(3,5)

let f1=(a,b)=>a===b

let f2=(a,b)=>{

    return a===b

}

console.log(f1(3,4))

console.log(f2(3,3))

//if only one parameter no need to use braces

let f3=name=>console.log(`hello ${name}`)

f3('yathvik')

//if their is no parameters parenthesis is empty

let f4=()=>console.log("welcome to all")

f4()

//returning the object to avoid confusion between block and object we use ()

let f5=()=>({name:'kumar',age:20})

console.log(f5())

**class:**

class X

{

    constructor(a,b)

    {

        this.a=a

        this.b=b

    }

    fun() {

        console.log(this.a+this.b)

    }

}

let o1=new X(10,20)

let o2=new X(2,4)

o1.fun()

o2.fun()

**Inheritance:**

class Animal

{

    constructor(name)

    {

        this.name=name

    }

    disp()

    {

        console.log(`${this.name} shouting`)

    }

}

class Dog extends Animal

{

    constructor(name,nm)

    {

        super(name)

        this.nm=nm

    }

    disp()

    {

        super.disp()

        console.log(`${this.nm} is shouting`)

    }

}

c1=new Dog('pupy','tomy')

c1.disp()

console.log(c1)

**Declaring multiple objects:**

//incase objects having same properties with diffrent values ex: car1,car2,car3 we can create using

//object literlas

c1={

    color:'red',

    brand:'TATA',

    maxspeed:70

}

c2={

    color:'blue',

    brand:'enova',

    maxspeed:140

}

console.log(c1)

console.log(c2)

//factory functions

//factory function is a function it will return object for every function call

function fun(color,brand,maxspeed)

{

return {

    color:color,

    brand:brand,

    maxspeed:maxspeed

}

}

c1=fun('red','TATA',80)

c2=fun('white','enova',140)

console.log(c1)

console.log(c2)

//constructor functions

//constructor function is a regular function it is going to return new object by using new operator.

function fun(name,age)

{

    this.name=name,

    this.age=age

}

s1=new fun('ravi',20)

s2=new fun('suman',21)

console.log(s1)

console.log(s2)

**Generators:**

Generator functions are a special variant of functions where the control flow can be paused and resumed, in order to produce a sequence of values.

function\* fun(n)

{

    for(let i=1;i<=n;i++)

    {

        yield i

    }

}

console.log(...fun(10))

**Exporting and importing:**

export function sum(a,b)

{

    return a+b

}

export function sub(a,b)

{

    return a-b

}

export function mul(a,b)

{

    return a\*b;

}

import \* as arth from './module.mjs'

console.log(arth.sum(4,5))

console.log(arth.sub(6,2))

console.log(arth.mul(6,4))

Default Exports: Default exports are useful to export only a single object, function, or variable. During the import, we can use any name to import.

export default function sum(a,b)

{

    return a+b

}

import add from './module.mjs'

console.log(add(4,6))

**jQuery:**

jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish and wraps them into methods that you can call with a single line of code.

**Adding jQuery to Your Web Pages:**

* Download the jQuery library from jQuery.com
* Include jQuery from a CDN, like Google

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

**jQuery Syntax:**

Basic syntax is: $(selector).action()

To prevent any jQuery code from running before the document is finished loading.

$(document).ready(function(){

// jQuery methods go here...

});

(or)

$(function(){

// jQuery methods go here...

});

**jQuery Selectors:**

Element Selector: $("div")

Id Selector: $(“#id”)

Class Selector: $(“.classname”)

All elements: $(“\*”)

Current element: $(this)

All even rows in table: $(“tr:even”)

All odd rows in table: $(“tr:odd”)

**Events:** An event represents the precise moment when something happens.

Some common DOM events are:

|  |  |  |  |
| --- | --- | --- | --- |
| **Mouse Events** | **Keyboard Events** | **Form Events** | **Document Events** |
| click | keypress | submit | load |
| dblclick | keydown | change | resize |
| hover | keyup | select | scroll |
| mousedown |  | blur | unload |
| mouseup |  | focus | ready |

In jQuery, most DOM events have an equivalent jQuery method. To assign a click event to all paragraphs on a page following statement is useful.

$(“p”).click()

**jQuery Effects:**

$(selector).hide(speed,callback);

$(selector).show(speed,callback);

$(selector).toggle(speed,callback);

<html>

    <head>

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

    </head>

    <body>

        <p>p tag 1</p>

        <p>p tag 2</p>

        <p>p tag 3</p>

        <script>

            $(document).ready(function(){

                $("p").click(function(){

                    $(this).hide()

                } )

            })

        </script>

    </body>

</html>

**Fading:**

fadeIn() method is used to fade in a hidden element.

fadeOut() method is used to fade out a visible element.

fadeToggle() method toggles between the fadeIn() and fadeOut() methods.

fadeTo() method allows fading to a given opacity (value between 0 and 1).

<body>

        <img src="C:\Users\irsr5\OneDrive\Desktop\js.png"/>

        <script>

            $(document).ready(function(){

                $("img").click(function(){

                    $(this).fadeOut(10000)

                } )

            })

        </script>

    </body>

**Sliding Elements:**

slideUp() method is used to slide up an element where as slideDown() method is used to slide down. slideToggle() methods to toggle the display state of elements between the slideUp() and slideDown() methods.

<html>

    <head>

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

        <script>

            $(document).ready(function() {

               $("#show").click(function(){

                  $("#box").slideDown(1000);

               });

               $("#hide").click(function(){

                  $("#box").slideUp(1000);

               });

         });

         </script>

         <style>

            button{cursor:pointer;}

            #box{margin-bottom:5px;padding:12px;height:100px; width:120px; background-color:#9c9cff;}

         </style>

         </head>

         <body>

            <div id="box">This is Box</div>

            <button id="hide">slideUp </button>

            <button id="show">slideDown </button>

    </body>

</html>

**Animations:**

The jQuery animate() method is used to create custom animations. $(selector).animate({params},speed,callback);

<!DOCTYPE html>

<html>

<head>

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

<script>

$(document).ready(function(){

  $("button").click(function(){

    $("div").animate({

      left: '100px',

      height: '+=10px',

      width: '+=10px'

    });

  });

});

</script>

</head>

<body>

<button>ZoomIn</button>

<div style="background:#1a0fe5;height:100px;width:100px;position:absolute;"></div>

</body>

</html>

**Chaining:** Chaining allows us to run multiple jQuery methods (on the same element) within a single statement.

Eg:

$("p").css("color", "blue").slideUp(3000).slideDown(3000);

**DOM Manipulation:**

text() - Sets or returns the text content of selected elements.

html() - Sets or returns the content of selected elements.

val() - Sets or returns the value of form fields.

<script>

$(document).ready(function(){

  $("#btn1").click(function(){

    alert("Text: " + $("#test").text());

  });

  $("#btn2").click(function(){

    alert("HTML: " + $("#test").html());

  });

});

To set the content:

$("#btn1").click(function(){

  $("#test1").text("Hai to all");

});

$("#btn2").click(function(){

  $("#test2").html("<b>Hello to all</b>");

});

$("#btn3").click(function(){

  $("#test3").val("Reddy");

});

attr() method is also used to set/change attribute values

$("a").attr("href", "https://www.google.com/")

**Add Elements:**

* append() - Inserts content at the end of the selected elements
* prepend() - Inserts content at the beginning of the selected elements
* after() - Inserts content after the selected elements
* before() - Inserts content before the selected elements

<!DOCTYPE html>

<html>

<head>

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

<script>

$(document).ready(function(){

  $("#btn2").click(function(){

    $("ol").append("<li>"+$("#ip1").val()+"</li>");

  });

});

</script>

</head>

<body>

<form>

    <input type="text" id="ip1" value="" /><br/>

</form>

<ol>

</ol>

<button id="btn2">Append list items</button>

</body>

</html>

**Manipulating CSS:**

* addClass() - Adds one or more classes to the selected elements
* removeClass() - Removes one or more classes from the selected elements
* toggleClass() - Toggles between adding/removing classes from the selected elements
* css() - Sets or returns the style attribute