**WHAT IS HTML**

* HTML is an acronym which stands for Hyper Text Markup Language. Let's see what is Hyper Text and what is Markup Language?
* **Hyper Text:** Hyper Text simply means "Text within Text". A text has a link within it, is a hypertext. Every time when you click on a word which brings you to a new webpage, you have clicked on a hypertext.
* **Markup language:** A markup language is a programming language that is used make text more interactive and dynamic. It can turn a text into images, tables, links etc.

**WEB DESIGN BASICS**

**Introduction to web design**

A web site design template is a pre-made website design template which can be customized to reflect your company’s branding. Website design templates can be found in various formats like Photoshop and HTML. Many times, these templates are compatible with HTML editors like GoLive, FrontPage, and Dreamweaver.

Web site templates can be very useful; they can be used by experienced web designers to ‘jump-start’ the creation of a website. They are also a way for people to put out great-looking web sites quickly with little or no knowledge of HTML and web design.

(C) Using colors like a pro

(D) The meaning of colors in web design

(E) Working with images

(F) Working with icons

(G) Spacing and layout

(H) Introduction to user experience

**DOM STRUCTURE**

* A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content.
* The way a document content is accessed and modified is called the Document Object Model, or DOM. The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.
* Window object − Top of the hierarchy. It is the outmost element of the object hierarchy.
* Document object − Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.
* Form object − Everything enclosed in the <form>...</form> tags sets the form object.
* Form control elements − The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

**HTML ELEMENTS: -**

* <! DOCTYPE> : This element defines the document type.
* <head> : This element defines information about document.
* <title> : This element is used to add title for document.
* <body> : This contains all the content of document.
* <h1>to<h6> : These elements used to add headings to the html page
* <p> : This element used to add paragraphs to html document
* <a href=” link”> : This provides a hyperlink.
* <b> : This is used to make the text bold.
* <address> : This element defines the contact information.
* <div> : This defines section in document.
* <strong> : This defines the important text.
* <br> : This is used to break the line and to start in new line.
* <style> : This defines style information of document.
* <img> : This element used to add image to the document.
* <form> : This is used to create html form to take the inputs.
* <footer> : This is used to add footer for document.
* <input> : This is used to create input fields.
* <label> : This defines a label for input element.
* <table> : This is used to create table
* <tr> : This element used to create rows in table.
* <th> : This is used to add heading in table.
* <td> : This is used to add data in the table.
* <ul> : This element defines unordered list.
* <ol> : This element defines ordered list.
* <li> : This elelment defines listitem.
* <Select> : This defines drop down list

**HTML ATTRIBUTES: -**

Attributes provide additional information about the HTML elements.

* All HTML elements can have **attributes**
* Attributes provide **additional information** about an element
* Attributes are always specified in **the start tag**
* Attributes usually come in name/value pairs like: **name="value"**

**HTML COLORS:**

We can apply colors in three ways to our template

1.color name (example..**color:black**;)

2.hexadecimal value of color(ex..**color:** **#000000;**)

3.rgb(). (Ex..**color:rgb(0,0,0);**)

**HTML PARAGRAPH: <p>** element defines a paragraph.

<P>something content in p tag</p>

**HTML HEADINGS:**

Headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

### **Example**

<h1>This is heading 1</h1>  
<h2>This is heading 2</h2>  
<h3>This is heading 3</h3>  
<h4>This is heading 4</h4>  
<h5>This is heading 5</h5>  
<h6>This is heading 6</h6>

**Output:**

# This is heading 1

## This is heading 2

### This is heading 3

#### This is heading 4

##### This is heading 5

###### This is heading 6

**HTML IMAGES:**

**<img>** tag defines images.

**Syntax:**

<img src=”image.jpg” alt=”this is an image”>

We can take image as background image by using css

**i.e:** background-image:url(‘imagename.jpg’);

**HTML LISTS:**

Html lists are used to display the content in list format. There are two types of lists

1.**unordered list**

2.**ordered list**

**Unordered list**  will display the content by default with bullets (i.e..like black dots.)

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

**Syntax:**

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

**Output:**

* Coffee
* Tea
* Milk

We can change the bullets as well. **Type** attribute is used to change the bullets to any other like square. Or we can also change by **using list-style-type** property. In css.

**Syntax:**

<ul type=”square”>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

**Output:**

* Coffee
* Tea
* Milk

Using **list-style-type**

<ul style=”list-style-type:square”>

**Output will be same.**

**Ordered list**  will display the content by default with numbers

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

**Syntax:**

<ol>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

**Output:**

1. Coffee
2. Tea
3. Milk

We can change the numbers as well. **Type** attribute is used to change the bullets to any other like roman numbers,alphabets.

**Syntax:**

<ol type=”a”>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

**Output:**

1. Coffee
2. Tea
3. Milk

**HTML TABLES:**

* A table is defined using the [<table>](http://html.com/tags/table/) element, and contains a number of [table cells ( <td>, for “table data” )](http://html.com/tags/td/) which are organized into [table rows ( <tr>)](http://html.com/tags/tr/). The markup (HTML code) for a table is always based on rows, never columns.
* Table cells which act as column headers or row headers should use the <th> (table header) element.
* Table cells can be merged using the [colspan](http://html.com/attributes/td-colspan/) and rowspan attributes.
* Tables can be broken into sections using the following elements:
  + [<thead> — Table header](http://html.com/tags/thead/)
  + [<tbody> — Table body](http://html.com/tags/tbody/)
  + [<tfoot> — Table footer](http://html.com/tags/tfoot/)
* A caption can be added to a table using the [<caption>](http://html.com/tags/caption/) element.
* You can use [<col>](http://html.com/tags/col/) and [<colgroup>](http://html.com/tags/colgroup/) to define table columns for styling. However, there are a [number of limitations with this practice](http://html.com/tags/colgroup/).

## Table Code Sample: Simple Table

<table>

<tr>

<th>Name</th>

<th>Favorite Color</th>

</tr>

<tr>

<td>Bob</td>

<td>Yellow</td>

</tr>

<tr>

<td>Michelle</td>

<td>Purple</td>

</tr>

</table>

|  |  |
| --- | --- |
| **Name** | **Favorite Color** |
| Bob | Yellow |
| Michelle | Purple |

Read more: <http://html.com/tables/#ixzz4iqh0qLK1>

* Use the HTML **<table>** element to define a table
* Use the HTML **<tr>** element to define a table row
* Use the HTML **<td>** element to define a table data
* Use the HTML **<th>** element to define a table heading
* Use the HTML **<caption>** element to define a table caption
* Use the CSS **border** property to define a border
* Use the CSS **border-collapse** property to collapse cell borders
* Use the CSS **padding** property to add padding to cells
* Use the CSS **text-align** property to align cell text
* Use the CSS **border-spacing** property to set the spacing between cells
* Use the **colspan** attribute to make a cell span many columns
* Use the **rowspan** attribute to make a cell span many rows
* Use the **id** attribute to uniquely define one table

# **HTML Form**

An **HTML form** is *a section of a document* which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.

An HTML form facilitates the user to enter data that is to be sent to the server for processing.

## **Why use HTML Form**

HTML forms are required if you want to collect some data from of the site visitor.

For example: If a user want to purchase some items on internet, he/she must fill the form such as shipping address and credit/debit card details so that item can be sent to the given address.

## **HTML Form Syntax**

1. **<form** action="server url" method="get|post"**>**
2. //input controls e.g. textfield, textarea, radiobutton, button
3. **</form>**

## **HTML Form Tags**

Let's see the list of HTML 5 form tags.

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <form> | It defines an HTML form to enter inputs by the used side. |
| <input> | It defines an input control. |
| <textarea> | It defines a multi-line input control. |
| <label> | It defines a label for an input element. |
| <fieldset> | It groups the related element in a form. |
| <legend> | It defines a caption for a <fieldset> element. |
| <select> | It defines a drop-down list. |
| <optgroup> | It defines a group of related options in a drop-down list. |
| <option> | It defines an option in a drop-down list. |
| <button> | It defines a clickable button. |
|  |  |

**HTML FORM STRUCTURE:**

**CODE:**

<p>Paragraph content.</p>

<form>

<fieldset disabled>

<legend>Contact Form</legend>

<!--These elements are discussed later in this tutorial-->

<p>Name: <input type="text" size="30"></p>

<p>Email: <input type="text" size="30"></p>

<p>Subject: <input type="text" size="30"></p>

<p>password: <input type="password"></p>

<p> <input type="button" value="button"></p>

<label>Gender</label>

<p><label>male</label><input type="radio" name="gender"></p>

<p><label>female</label><input type="radio" name="gender"></p>

<p><label>html</label><input type="chekbox" name=""></p>

<p><label>css</label><input type="chekbox" name=""></p>

<p><label>javascript</label><input type="chekbox" name=""></p>

<p>Message: <textarea>Type your message here</textarea></p>

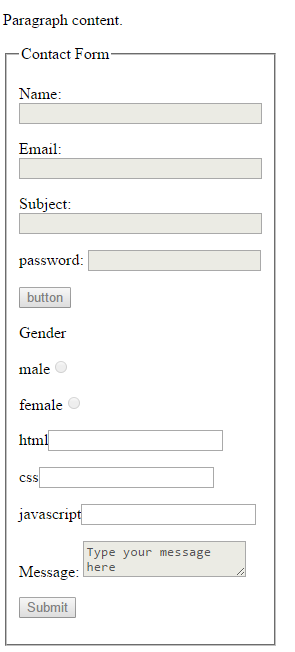
<p><input type="submit"></p>

</fieldset>

</form>

**OUTPUT:**

Output for the above code is:



**For more read:** [**http://html.com/forms/**](http://html.com/forms/)

**CSS**

**FORMATTING WITH CSS: -**

Formatting with css has a lot to do with html elements that means you can format a element with css and and apply styles as needed. Below are the css properties that you can apply on a html element

**CSS PROPERTIES: -**

* Float : This is used to move the element to right or left.
* Margin : This is used to add all margin properties to Element in single line.
* Padding : This is used to add all padding properties to Element in single line.
* Position : specifies the type of position used for element from positions relative, absolute, and fixed.
* text-decoration : defines the decoration added to the text.
* font-family : defines the font-family for text
* background-img : used to add background image to element.
* Active : used to add stylings when the element is in active state.
* Hover : used to add stylings when the element hovered.
* Visited : used to add stylings when link is in visited state.
* first-child : used to select the first child from parent element.
* last-child : used to select the last child from parent element.
* After : used to insert some content after the selected Element.
* Before : used to insert some content before the selected Element.
* box-shadow : used to add shadows to the selected element.
* Border : used to insert border to selected element.
* Clear : specifies on which sides of an element floating Elements are not allowed to float.
* Visibility : specifies whether the element to be visible or not.

**UNITS IN CSS: -**

Basically, there are four types of units in css.

* 1.ems
* 2.pixels
* 3.percentages
* 4.points (pt.)

**1. Ems:** The “em” is a scalable unit that is used in web document media. An em is equal to the current font-size, for instance, if the font-size of the document is 12pt, 1em is equal to 12pt. Ems are scalable in nature.  Ems is getting popular in web documents due to scalability and their mobile-device-friendly nature.

* 1em=12pt
* 1em=16px
* 1em=100%

**2. Pixels:** Pixels are fixed-size units that are used in screen media (i.e. to be read on the computer screen). One pixel is equal to one dot on the computer screen (the smallest division of your screen’s resolution).  Many web designers use pixel units in web documents to produce a pixel-perfect representation of their site as it is rendered in the browser.

* 1px=0.0625em
* 1px=0.75pts

**3. Percentages:** The percent unit is much like the “em” unit, save for a few fundamental differences. First and foremost, the current font-size is equal to 100% (i.e. 12pt = 100%). While using the percent unit, your text remains fully scalable for mobile devices and for accessibility.

If we want to convert pixels into percentage then the calculation is as follows (example: calculating width from pixels to percentage):

Percentage (%) = Width of child / width of parent.

**4. Points (pt.):**  Points are traditionally used in print media (anything that is to be printed on paper, etc.). One point is equal to 1/72 of an inch. Points are much like pixels; in that they are fixed-size units and cannot scale in size.

**REM:**  rem is also unit in css, rem stands for root em. This is only relative to the root element. Root element means html. If we change the font-size of the root element then rem will also effected.

**Difference between Em and Rem:**

Em is relative to its nearest element or its parent element, but rem is only relative to the root element. If we change the font-size of the parent element, then em also effected but not rem. Rem will be effected only when the root element effected.

**Difference between Em and Rem:**

Em is relative to its nearest element or its parent element, but rem is only relative to the root element. If we change the font-size of the parent element, then em also effected but not rem. Rem will be effected only when the root element effected.

## **Font Awesome Icons:**

To use the Font Awesome icons, add the following line inside the <head> section of your HTML page:

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

**Note:** No downloading or installation is required!

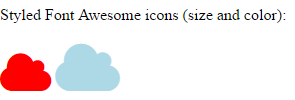
### **Example**

<head>  
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">  
</head>  
<body>  
  
<p>Styled Font Awesome icons (size and color):</p>

<i class="fa fa-cloud" style="font-size:48px;color:red;"></i>

<i class="fa fa-cloud" style="font-size:60px;color:lightblue;"></i>  
  
</body>

**Output:**



**NAVIGATION BAR**: A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense

**EXAMPLE:**

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover {

background-color: #111;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

</body>

</html>

**Output:**

C:\Users\Digitallync\Documents\navbar.PNG

**BOX MODEL:** Any HTML element can be considered a box, and so the box model applies to all HTML (and XHTML) elements. The box model is the specification that defines how a box and its attributes relate to each other. In its simplest form, the box model tells browsers that a box defined as having width 100 pixels and height 50 pixels should be drawn 100 pixels wide and 50 pixels tall.

* **Content** - The content of the box, where text and images appear
* **Padding** - Clears an area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent

EXAMPLE:

<!DOCTYPE html>

<html>

<head>

<style>

div {

border: 6px solid #949599;

height: 100px;

margin: 20px;

padding: 20px;

width: 400px;

}

</style>

</head>

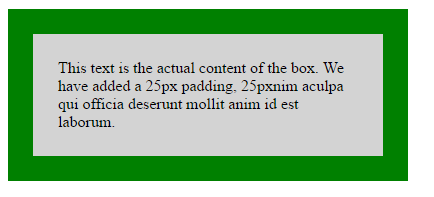
<body>

<div>.</div>

</body>

</html>

**OUTPUT:**

****

**FLOATS:**

**Floats:** The float property causes an element to be moved to one side of the parent element.

* **Float-Left** − Element is floated to the left side of its parent element's content area.
* **Float-Right** − Element is floated to the right side of its parent element's content area.
* **Float-None** − Element is not floated.

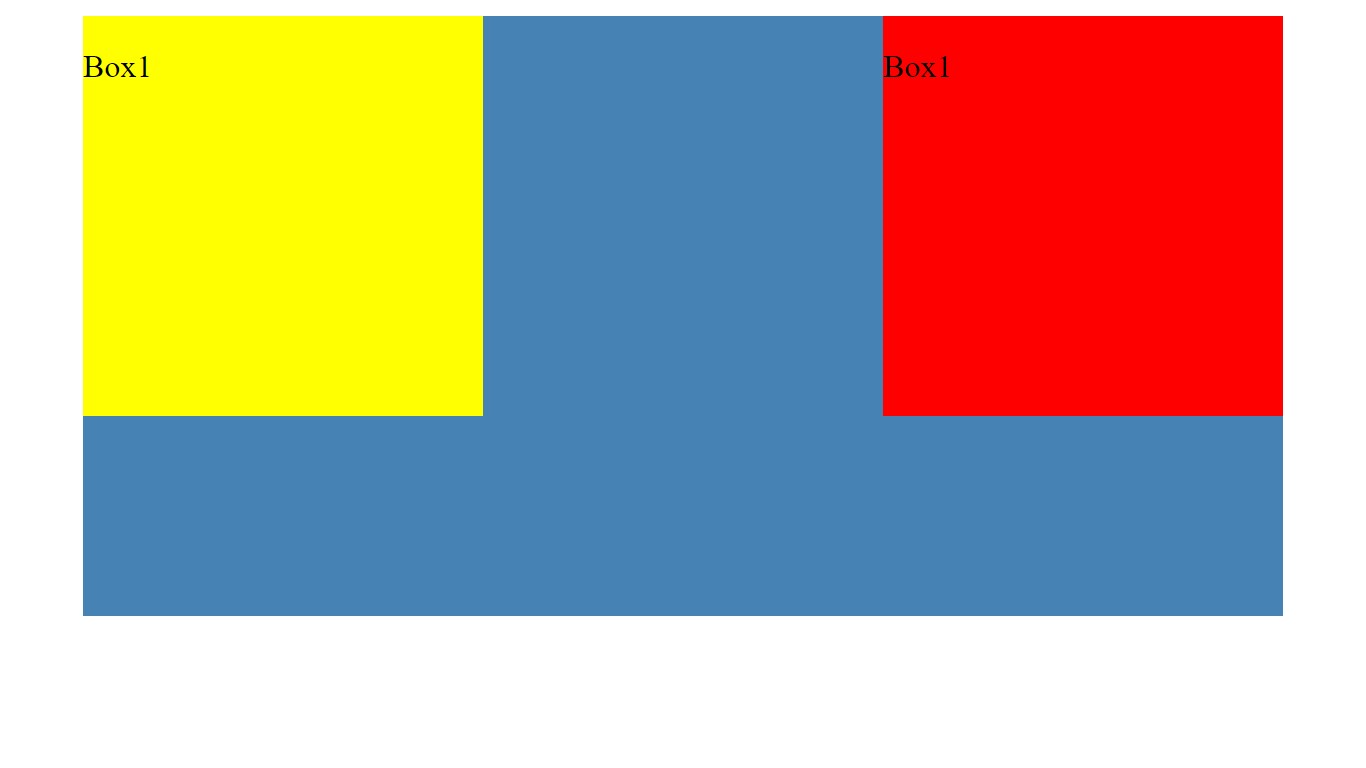
**Techniques for clearing the floats:**

* **Overflow:** To wrap the child elements into the parent element
* **Clear:** To clear the floated area
* **Clear: both** No floating elements allowed on either the left or the right side
* **Clear-fix:** The “clear fix” (which means *fixing* the *clearing* of floats) defines a clear fix class in our stylesheet that we can apply to any float-containing element. This will force the container element to expand, pushing subsequent elements beneath it.

**Code:**

|  |  |
| --- | --- |
| **HTML**  <! DOCTYPE html>  <html>  <head>  <title>Floats</title>  </head>  <body>  <div class="parent">  <div class="child1">  <p>Box1</p>  </div>  <div class="child2">  <p>Box1</p>  </div>  </div>  </body>  </html> | **CSS**  . parent {  margin: auto;  width: 600px;  height: 300px;  background-color: steelblue;  /\*applying the overflow property to clear the floats\*/  overflow: hidden;  }  . child1{  background-color: yellow;  width:200px;  height:200px;  /\*applying the float: left property to the child1\*/  float: left;  }  . child2{  background-color: red;  width:200px;  height:200px;  /\*applying the float: right property to the child\*/  float: right;  } |

**Output:**



**POSITIONS: -**

Position property is used to specify the type of the position used for the element from the following.

There are four position properties in css.

1. Static

2. Relative

3. Absolute

4. Fixed

**1. Static:** HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.it always positioned as per the normal flow of the page

**Syntax:** div {

Position: static;

}

**2. Relative:** If element is positioned relative, then element is relative to its normal position. So, if we give value as "left: 20px" it adds 20 pixels to the element's LEFT position. It means the element moves from its own boundary 20px to the right side.

**Syntax:** div

{

Position: relative;

Left: 20px;

}

**3. Absolute:** This tells the browser that whatever is going to be positioned absolute, should be removed from the normal flow of the document and will be placed in an exact location on the page. It won't affect how the elements before it or after it in the HTML are positioned on the Web page however it will be subject to its parents positioning unless you override it.

**Syntax:** div

{

Position: absolute;

Top: 30px;

Left: 50px;

}

As per the above code, div will move 30px from top 50px from left, by considering the browser border.

**Difference between relative and absolute:**

When we give ‘position: relative’ to the parent div and ‘position: absolute’ to the child div then the child div will move by considering the border of the parent div.

**Syntax:** parent div

{

Position: relative;

}

Child div

{

Position: absolute;

Top: 40px;

Right: 50px;

}

**4. Fixed:** A fixed element is positioned relative to the viewport (browser window), which means it always stays in the same place even if the page is scrolled.

**Syntax:** div

{

Position: fixed;

}

In the above syntax div was given the position as fixed so then that div element will be fixed to the browser it will not move even if the page is scrolled.

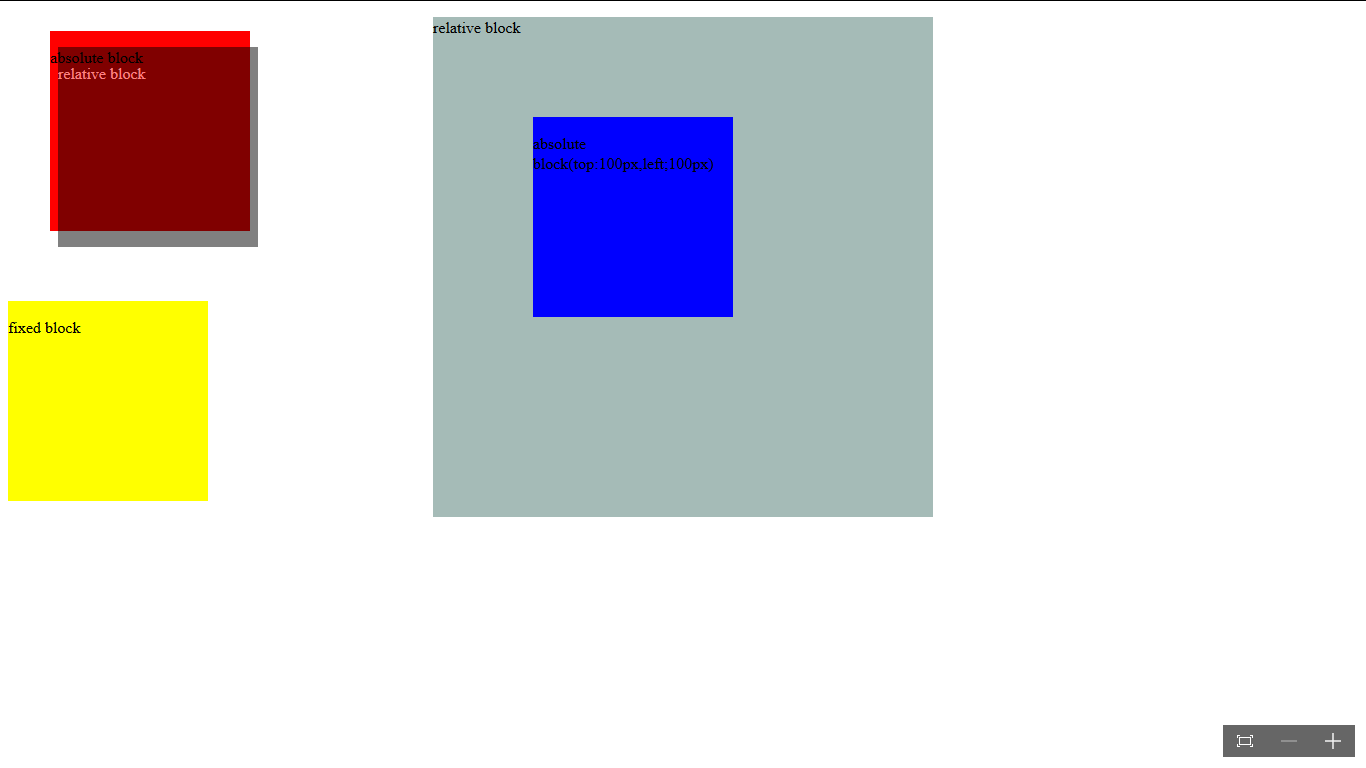
**Difference between fixed and absolute:**

When we give position as fixed to an element that will be fixed to the viewport but when we give absolute then that element is not fixed to the viewport, it will move along with the page when we scroll, it also moves if we give top/left/bottom/right values.

**Code For all Positions:**

|  |  |
| --- | --- |
| **HTML**  <body>  <div class="fixedBlock">  <p>fixed block</p>  </div>  <div class="absoluteBlock">  <p>absolute block</p>  </div>  <div class="relativeBlock">  <p>relative block</p>  </div>  <div class="parent">  <p>relative block</p>  <div class="child">  <p>absolute block (top:100px, left;100px) </p>  </div>  </div>  </body> | **CSS**  <style>  /\*fixed block: fixed to browser page\*/  .fixedBlock{  background-color: yellow;  position: fixed;  width:200px;  height: 200px;  top:300px;  clear:both;  float:left;  }  /\*here we are taking same top and left properties for absolute block and relative block but the difference we can see here is relative block is taking its own boundary whereas absolute block is taking the browser boundary for changing their respective postions\*/  . absoluteBlock{  background-color: red;  position: absolute;  width:200px;  height: 200px;  top:30px;  left:50px;  clear: both;  float: left;  }  .relativeBlock{  background-color: black;  position: relative;  width:200px;  height: 200px;  top:30px;  left:50px;  clear: both;  float: left;  color: white;  opacity: 0.5;  }  /\*differnce between the relative and absolute block\*/  . parent{  width:500px;  height: 500px;  background-color: #A5BBB7 ;  position: relative;  margin: auto;    }  .child{  width:200px;  height:200px;  position: absolute;  background-color: blue;  top:100px;  left:100px;  }  </style> |

**Output:**



**VIEWPORT: -**

Viewport is the user's visible area of a web page. Viewport Varies with the device, and will be smaller on a mobile phone than on a computer screen. Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.

**Syntax:**

|  |
| --- |
| <meta name="viewport" content="width=device-width, initial-scale=1.0"> |

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.

**PSEUDO CLASSES: -**

**Pseudo class:** A pseudo-class is used to define a special state of an element. These are following pseudo classes.

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

**: Last-child:** p: last-child Selects every <p> elements that is the last child of its parent.

**: Focus: -** input: focus selects the <input> element that has focus.

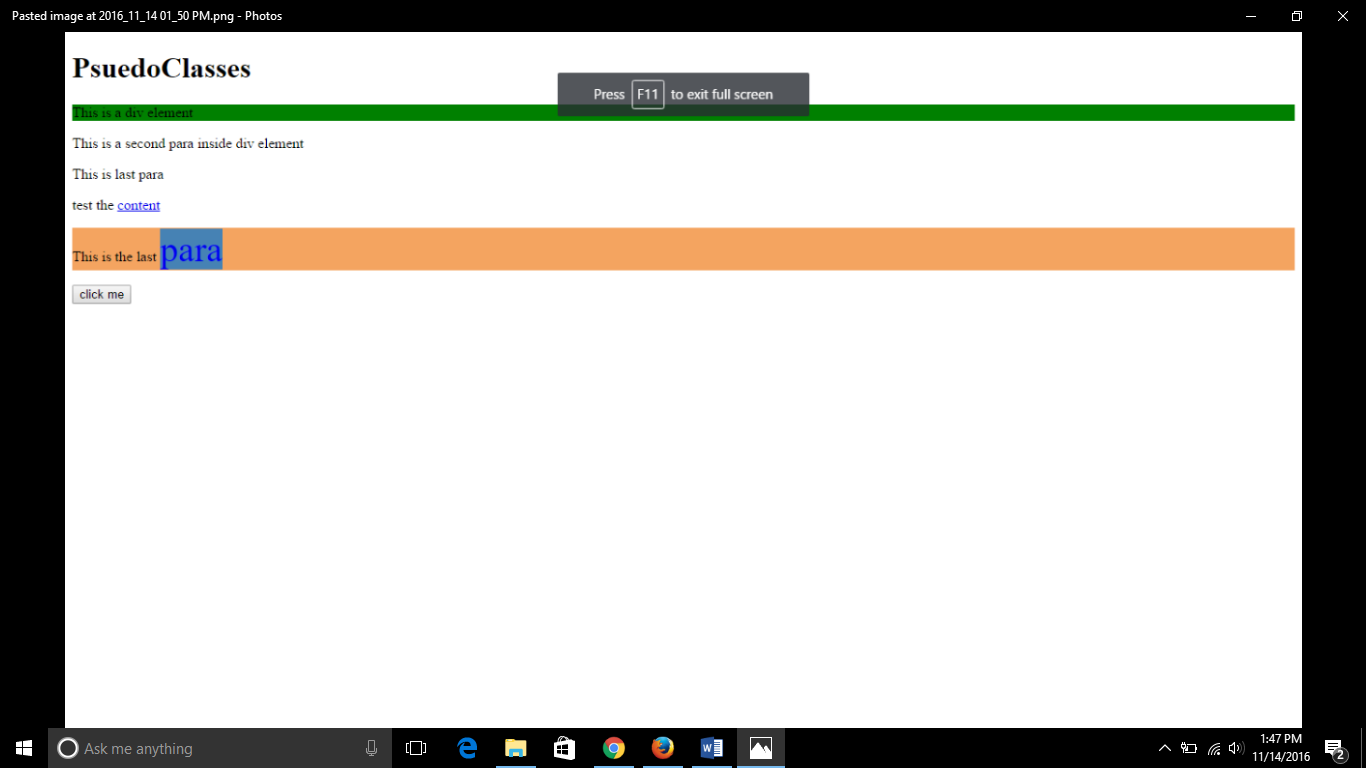
**Example: -**

* + Style an element when a user mousse over it.
  + Style visited and unvisited links differently.
  + Style an element when it gets focus.

**CODE:**

|  |  |
| --- | --- |
| **HTML**  <body>  <h1>Pseudo-events</h1>  <div>  <p>This is a div element</p>  <p>This is a second para inside div element</p>  <p>This is last para</p>  <div>  test the <a href="#">content</a>  <p>This is the last <a href="#">para </a></p>  <!-- here the anchor tag is the only child for its parent element i.e <p>tag-->  </div>  </div>  <input type="button" value="click me">  </input>  </body>  </html> | **CSS**  <style type="text/css">  input: hover {  background-color: light green;  cursor: pointer;  }  a: active {  font-size: 50px;  background-color: #f00;  color: #fff;  }  p: first-child {  background-color: green;  }  p:last-child{  background-color: sandybrown;  }  a: only-child{  color:"#999";  background-color: steelblue;  text-decoration: none;  font-size: 40px;  }  </style> |

**OUTPUT:**



**PSEUDO ELEMENTS:**

**Pseudo Elements: -** A CSS Pseudo-element is used to style specified parts of an element. Like, Style the first letter, or line, of an element

Insert content before, or after, the content of an element

**Examples: -**

**: After (like p: After): -** Insert something after the content of each <p> element

**: Before (p: before): -** Insert something before the content of each <p> element

**: First-letter (p: first-letter): -** Selects the first letter of each <p> element

**: First-line (: first-line): -** Selects the first line of each <p> element

|  |  |
| --- | --- |
| **HTML**  <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. </p> | **CSS**  p: first-line {color: red;  }  p: first-letter {  font-size: 23px;  color: black;  }  p: before {  content: "yes";  } |

**Media queries:**

A **media query** basically limits the style sheets scope by using media features, such as width, height, and color.

Media queries consist of an optional media type, as of the CSS3 specification; contain zero or more expressions, expressed as media features, which resolve to either true or false.

The following are the properties that would be used:

* width
* height
* float
* margins

**Syntax:**

@media only screen and (max-device-width: 480px) {  
/\* some specific styles come here \*/  
}

**Responsive web design:**

Responsive web design is the practice of building a website suitable to work on every device and every screen size, no matter how large or small, mobile or desktop. Responsive web design is focused to provide a gratifying experience for everyone.

**Adaptive web design:**

Adaptive web design (AWD) aims to create a website that is applicable to a Variety of different platforms and layouts, AWD has several distinct layouts for multiple screen sizes and the layout used depends on the screen size being used by the website user.

**Difference between Adaptive and Responsive design:**

Responsive and adaptive web design are closely related, and often transposed as one in the same. Responsive generally means to react quickly and positively to any change, while adaptive means to be easily modified for a new purpose or situation, such as change. With responsive design websites, continually and fluidly change based on different factors, such as viewport width, while adaptive websites are built to a group of preset factors.

Currently the most popular technique lies within responsive web design, favoring design that dynamically adapts to different browser and device viewports, changing layout and content along the way

**Min width and max width:**

**Min width:**

**Max width:** The media queries will be applied according to the given max-width when browser width is changed respective to the devices.

@media screen and (min-width: 769px) {

//css goes here

}

@media only screen and (max-device-width: 480px) {

//css goes here

}

**Example code:**

<! DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<style>

/\*media query \*/

@media only screen and (max-width: 500px) {

body {

Background-color: grey;

}

}

. form {

Margin-top: 50px;

}

. Form input {

Margin-bottom: 17px;

Width: 140px;

Height: 20px;

Margin-left: 10px;

}

</style>

</head>

<Body>

<Center>

<div class="form">

Name :< input type="text"><br>

Email :< input type="text"><br>

Mobile :< input type="text"><br>

</div>

</center>

</body>

</html>