Table of Contents

1. Introduction

1.1 What is CSS?

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

1.2 Why CSS?

Hyper Text Markup Language (HTML) is used to structure the collection of information in a document.
But the presentation of the document depends on how the markup elements within the page is styled.
A document looks elegant when styles are used appropriately.
CSS can be used to provide styles to the markup elements.

1.3 CSS Syntax

```
selectors {
  css-property: value;
}
```

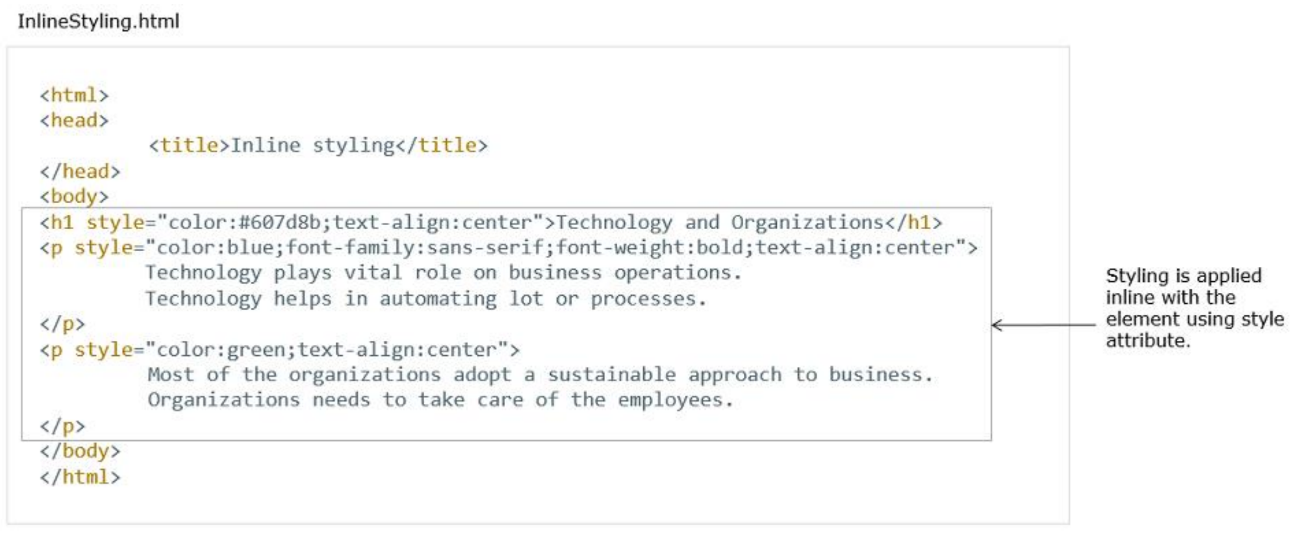
In the above syntax the code inside the curly braces is known as CSS declaration block where we provide different CSS styling to style our web pages. selectors refer to html element which requires styling.

1.4 Specifying stylesheets

There are different ways of specifying stylesheets:

1.4.1 Inline Styling

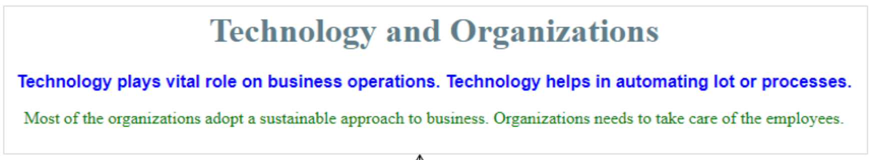
We can use the style attribute to specify the styling. This is referred to as inline styling. We can style only one element at a time using Inline styling.


InlineStyling.html

```
<html>
<head>
        <title>Inline styling</title>
</head>
<body>
<h1 style="color:#607d8b;text-align:center">Technology and Organizations</h1>
<p style="color:blue;font-family:sans-serif;font-weight:bold;text-align:center">
        Technology plays vital role on business operations.
        Technology helps in automating lot or processes.
</p>
<p style="color:green;text-align:center">
        Most of the organizations adopt a sustainable approach to business.
        Organizations needs to take care of the employees.
</p>
</body>
</html>
```

Styling is applied inline with the element using style attribute.

Output



Technology and Organizations

Technology plays vital role on business operations. Technology helps in automating lot or processes.

Most of the organizations adopt a sustainable approach to business. Organizations needs to take care of the employees.

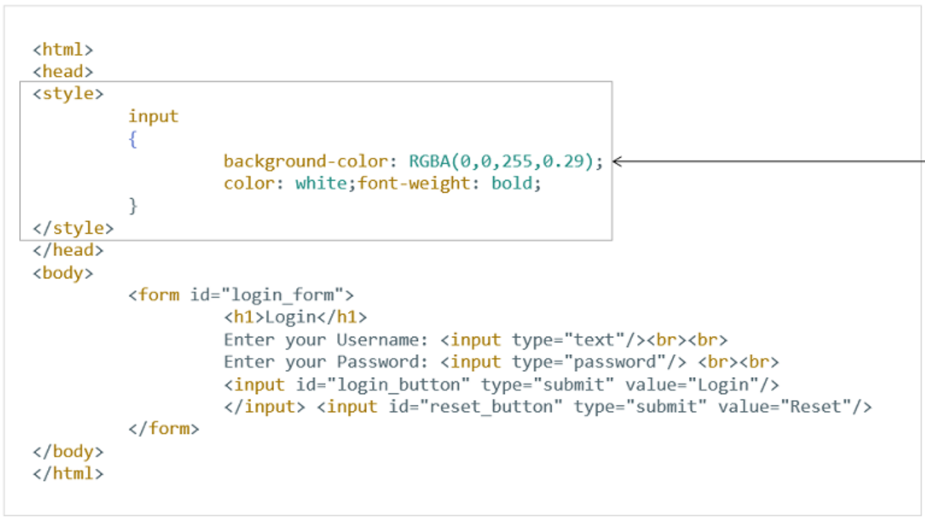
The heading and the paragraphs are rendered with their respective inline styling applied.

This approach is not a best practice because it mixes the content and presentation hence should be used only when you want to style a particular element differently.

1.4.2 Internal/Embedded Styling

Style rules can be written separately within the style tag of HTML.
This way of styling is referred to as internal/embedded styling as the styling code is embedded along with the HTML code.
The style rules are written using the internal styling effects the HTML elements of the web page in which it is written.

InternalStyling.html



```
<html>
<head>
<style>
         input
         {
                  background-color: RGBA(0,0,255,0.29);
                  color: white;font-weight: bold;
         }
</style>
</head>
<body>
         <form id="login_form">
                  <h1>Login</h1>
                  Enter your Username: <input type="text"/><br><br>
                  Enter your Password: <input type="password"/> <br><br>
                  <input id="login_button" type="submit" value="Login"/>
                  </input> <input id="reset_button" type="submit" value="Reset"/>
         </form>
</body>
</html>
```

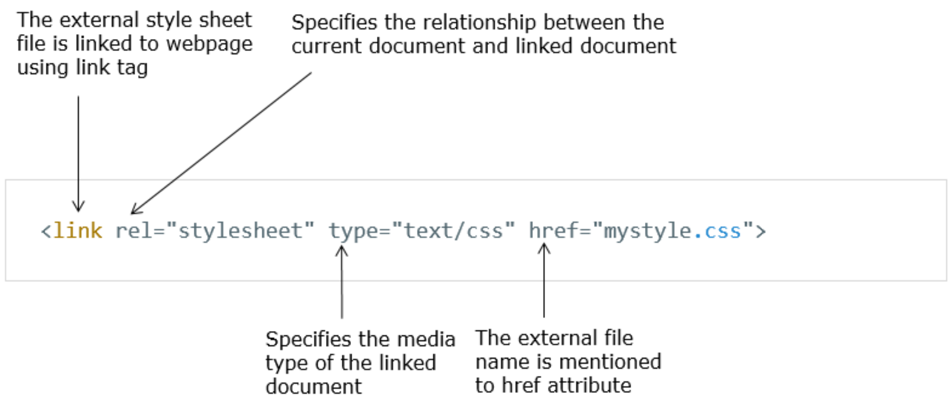
Internal styling is applied to input fields within the style tag.

Internal style sheets have below disadvantages

- Can affect only one web page.
- Can affect the page load time.

1.4.3 External Stylesheet

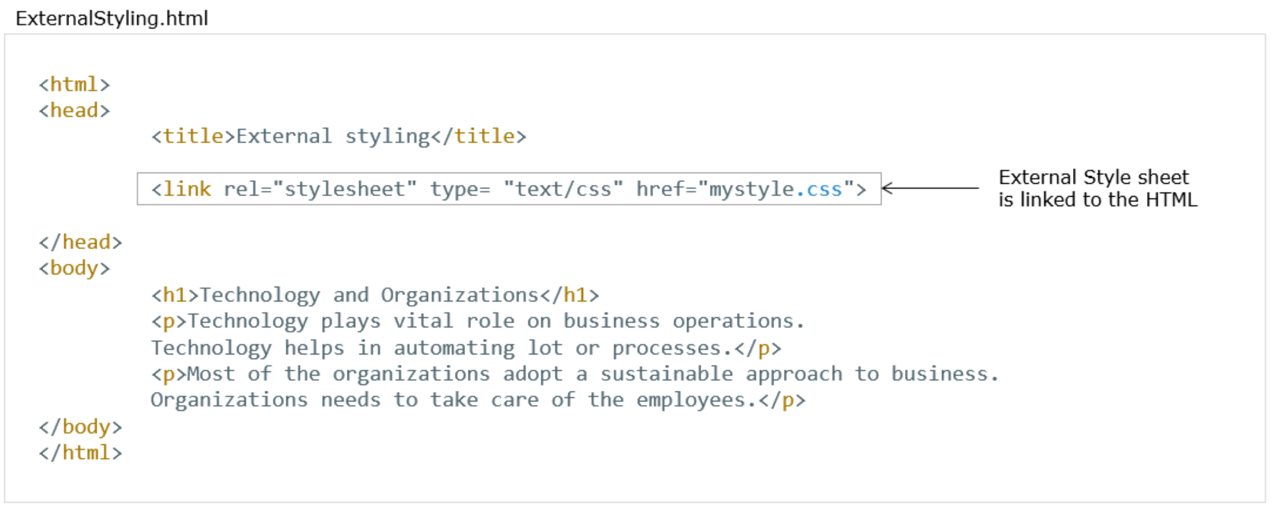
Style rules can also be written in a separate file.
This file is saved with the .css extension.
This approach of writing style rules is called external styling.
The HTML files which want to use styles written in the external file need to use link tag of HTML.



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

The external style sheet file is linked to webpage using link tag
Specifies the relationship between the current document and linked document
Specifies the media type of the linked document
The external file name is mentioned to href attribute

External style sheets can be reused and easily maintained.
When the same style rules need to be implemented in another web page, we can link the external style sheet to the web page without repeating the styling code.
When the styling rules are changed in the external style sheet the change will be reflected in all the web pages.



ExternalStyling.html

```html
<html>
<head>
        <title>External styling</title>

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

</head>
<body>
        <h1>Technology and Organizations</h1>
        <p>Technology plays vital role on business operations.
        Technology helps in automating lot or processes.</p>
        <p>Most of the organizations adopt a sustainable approach to business.
        Organizations needs to take care of the employees.</p>
</body>
</html>
```

ExternalStyling.html

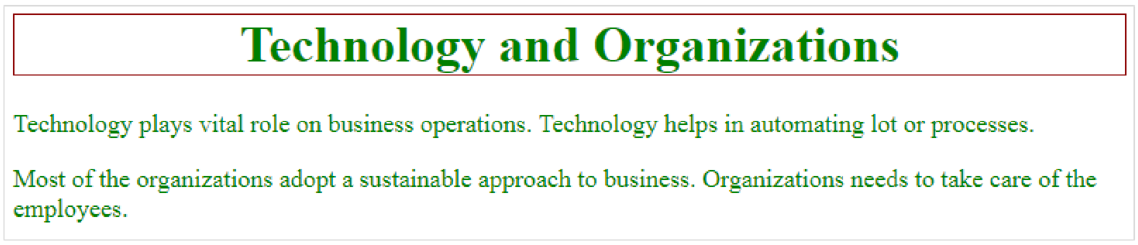
```
<html>
<head>
         <title>External styling</title>

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

</head>
<body>
         <h1>Technology and Organizations</h1>
         <p>Technology plays vital role on business operations.
         Technology helps in automating lot or processes.</p>
         <p>Most of the organizations adopt a sustainable approach to business.
         Organizations needs to take care of the employees.</p>
</body>
</html>
```

External Style sheet
is linked to the HTML

Output



Technology and Organizations

Technology plays vital role on business operations. Technology helps in automating lot or processes.

Most of the organizations adopt a sustainable approach to business. Organizations needs to take care of the employees.

1.4.4 Import Stylesheet

Sometimes multiple websites want to use the same style sheet files. This file might be available to everyone using a URL.
This is called importing style sheets.
To import style files import statement should be placed within the style tag.

The syntax of the import statement is
@import url("url of css file")

Syntax

```
<style type="text/css">
         @import url("http://www.xyz.com/style2.css")
</style>
```

It is possible to import a CSS file within another CSS file

Consider a CSS file style.css

```
body {
  background-color: lightgrey;
}
.red {
  color: red;
}
#pid {
  background-color: yellow;
  color: green;
}
```

If you need to import this style.css file within another CSS file say sample.css file, then you can import as shown below

```
@import url('style.css')
div {
  border: lightgrey dashed 3px;
  padding: 20px;
  margin: 60px;
  width: 220px;
  height: 125px;
}
```

While importing we need to provide the complete URL of the CSS file.

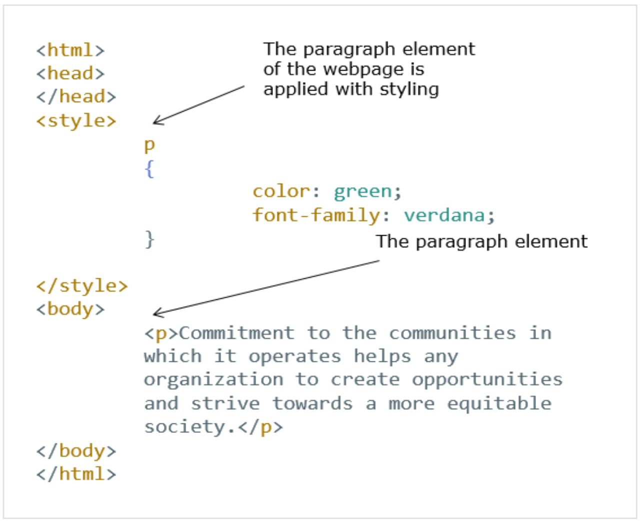
2. Selectors

2.1 Element Selector

Selects all the HTML elements that match the tag/node name.

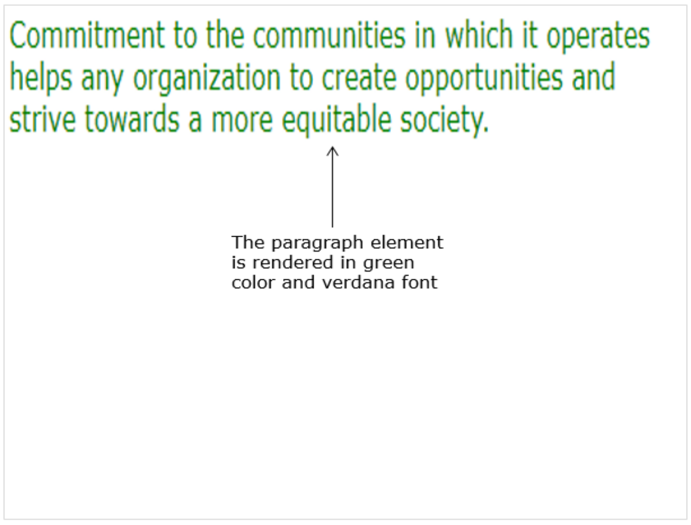
Tag_Name {property: value}

We have a requirement where we need to set the font color as green to all the paragraph(p) element. The common thing in all the elements is the tag name i.e. "p". For such purpose, we can use Element selector.



```
<html>
<head>
</head>
<style>
         p
         {
                  color: green;
                  font-family: verdana;
         }
</style>
<body>
         <p>Commitment to the communities in
         which it operates helps any
         organization to create opportunities
         and strive towards a more equitable
         society.</p>
</body>
</html>
```

The paragraph element of the webpage is applied with styling
The paragraph element



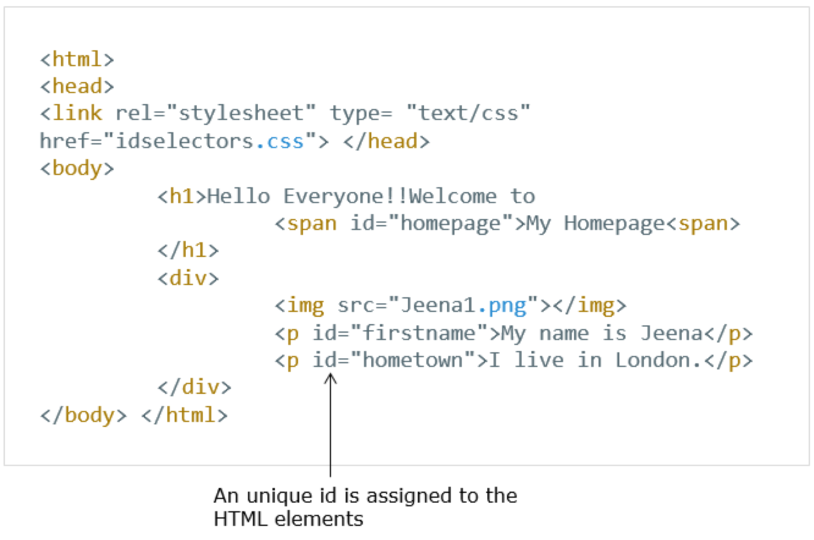
Commitment to the communities in which it operates helps any organization to create opportunities and strive towards a more equitable society.

The paragraph element is rendered in green color and verdana font

2.2 Id Selector

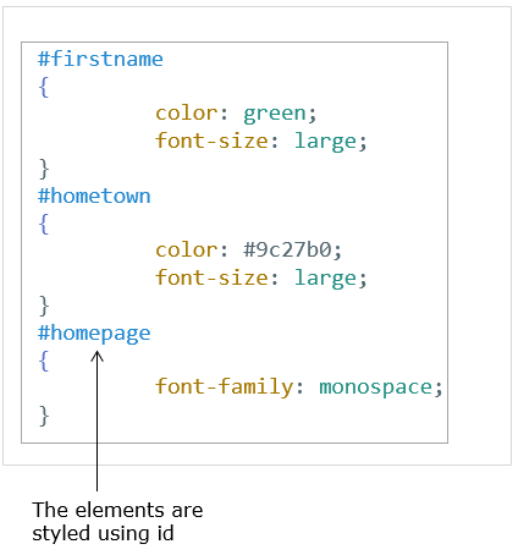
Id selectors match an element based on the element's id attribute and apply the styling accordingly.
Id must be unique for each HTML element.
The id should be prefixed with "#" while using the id in CSS file for styling.
Id selectors are used for styling an element specifically.

idSelectors.html



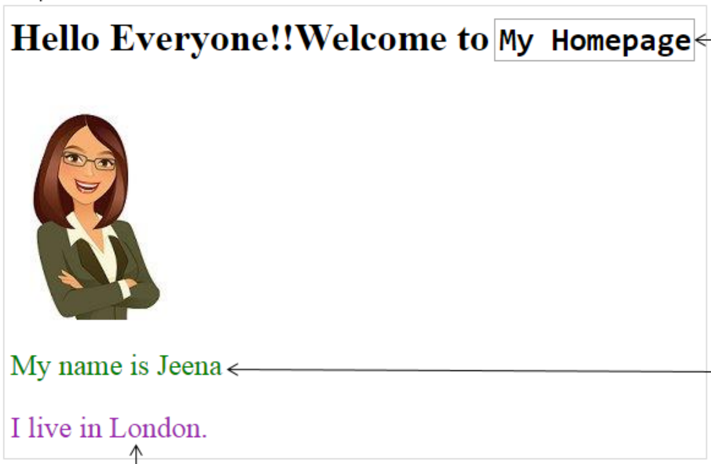
```
<html>
<head>
<link rel="stylesheet" type= "text/css"
href="idselectors.css"> </head>
<body>
         <h1>Hello Everyone!!Welcome to
                  <span id="homepage">My Homepage<span>
         </h1>
         <div>
                  <img src="Jeena1.png"></img>
                  <p id="firstname">My name is Jeena</p>
                  <p id="hometown">I live in London.</p>
         </div>
</body> </html>
```

idselectors.css



```
#firstname
{
         color: green;
         font-size: large;
}
#hometown
{
         color: #9c27b0;
         font-size: large;
}
#homepage
{
         font-family: monospace;
}
```

Output



Hello Everyone!!Welcome to My Homepage

My name is Jeena

I live in London.

Styled using #homepage id selector

Styled using #firstname id selector

Styled using #hometown id selector

2.3 Class Selector

Class selectors match an element based on the contents of the element's class attribute and apply the styling. When you want to style a set of elements with common styling you can assign a single class name to a set of elements.
All the elements with that class name will be applied with the same styling.
The class name should be prefixed with a "."

Classselectors.html

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="classselectors.css">
<title>Class selectors</title>
</head>
<body>
    <div class="intro">
       <img src="Jeena1.png">
       <p>My name is Jeena</p>
       <p>I live in London.</p>
    </div>
    <p>I like the following beverages :</p>
    <ul class="intro">
       <li> Coffee</li>
       <li>Tea</li>
    </ul>
</body>
</html>
```

Class name is assigned to the <div> and <ul> element

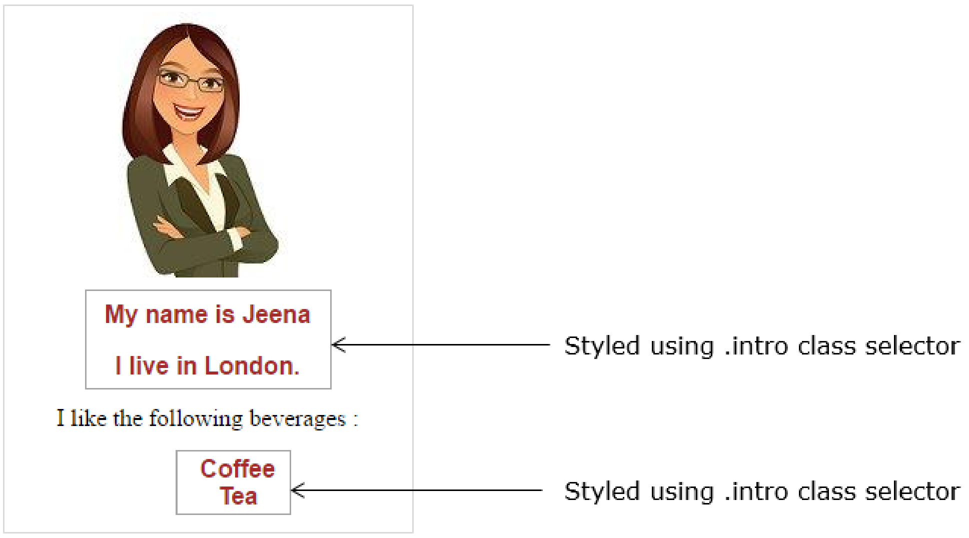
classselectors.css

```
.intro
{
    color: brown;
    font-family: sans-serif;
    font-weight: bold;

}
```

The styling is applied using class name

Output


My name is Jeena
I live in London.
I like the following beverages :
Coffee
Tea
Styled using .intro class selector
Styled using .intro class selector

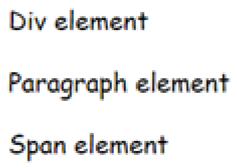
2.4 Universal Selector

Selects all the HTML elements on the webpage.
It is generally used when you need to apply the same style for all the elements of the webpage.

*{property: value}

```
<div>Div element</div>
<p id=”sad face”>Paragraph element</p>
<span class=”happy face”>Span element</span>
```

```
*{ font-family: cursive; }
```



2.5 Attribute Selector

Selects the HTML elements, if the attribute and its values of the HTML element match the selector.

If we have a requirement to style an element based on the attribute of the element or based on the value of the attribute of the element.

In this scenario, we can use **attribute selectors**.

Attribute Selector	Description
a [attribute]	Selects all A elements with the specified attribute.
a[attribute=value]	Selects all A elements with specified attribute and value.
a[attribute^=value]	Selects all A elements where the value of attribute starts with the specified value.
a[attribute$=value]	Selects all A elements where the value of attribute ends with the specified value.
a[attribute*=value]	Selects all A elements where the value of the attribute contains the specified value.

2.6 Combinator/Relational Selector

Combinators are the selectors formed by combining multiple selectors and establishing a relationship between the selectors.
The relationship between the selectors is established using the symbols >, +, ~, space (" “)
Used for selecting a specific element/node to style them.

```
Selector1 (>, +, ~, (space)) Selector2
```

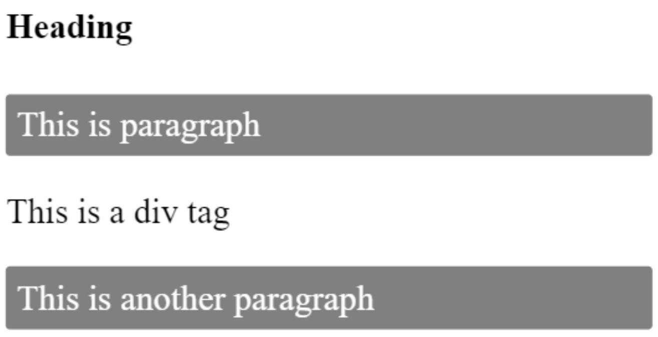
2.6.1 General Sibling Combinator

It selects all the elements that match the second selector and are siblings to the first element. It is represented by ~.

```
Selector1 ~ Selector2 {/* styling properties*/}
```

```
<h4>Heading</h4>
<p> This is a paragraph </p>
<div> This is a div tag </div>
<p> This is another paragraph </p>
```

```
h4 ~ p {
   background-color: gray;
   border-radius: 2px;
   padding: 5px;
   color: white;
}
```


Heading

This is paragraph

This is a div tag

This is another paragraph

2.6.2 Adjacent Sibling Combinator

It selects the second element which is adjacent (i.e. right next to) to the first element and both having the same parent. It is represented by +.

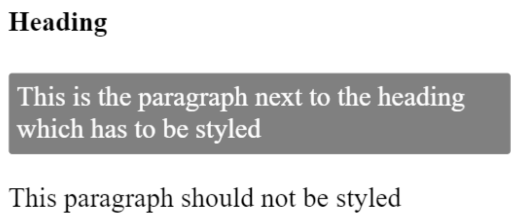
Selector1 + Selector2 {/* styling properties*/}

```
<h4>Heading</h4>

<p>This is the paragraph next to the heading which has to be styled</p>

<p>This paragraph should not be styled</p>
```

```
h4 + p {
   background-color: gray;
   border-radius: 2px;
   padding: 5px;
   color: white;
}
```


Heading

This is the paragraph next to the heading which has to be styled

This paragraph should not be styled

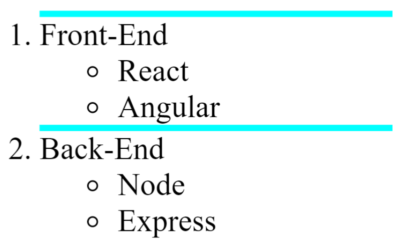
2.6.3 Child Combinator

It selects the second element which is a direct child of the first element. It is represented by >.

Selector1 > Selector2 {/* styling properties*/}

```
<ol>
   <li>
      Front-End
      <ul>
         <li>React</li>
         <li>Angular</li>
      </ul>
   </li>
   <li>
      Back-End
      <ul>
         <li>Node</li>
         <li>Express</li>
      </ul>
   </li>
</ol>
```

```
ol>li {
   border-top: 3px solid aqua;
}
```


1. Front-End
   - React
   - Angular
2. Back-End
   - Node
   - Express

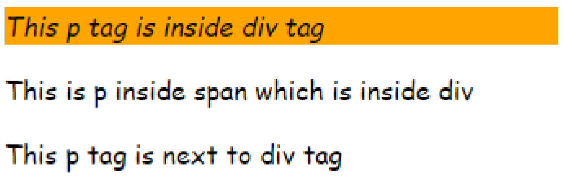
2.6.4 Descendant Combinator

It selects the second element which is a descendant (child/grandchild etc of the first element. It is represented by a space ( ).

```
Selector1 Selector2 {/* styling properties*/}
```

```
<div id=”firstDiv”>
   <p class=”p1”>This p tag is inside div tag</p> <!—Selected ->
   <span>
      <p>This is p inside span which is inside div</p><!- Selected ->
   </span>
</div>
<p class=”p1”>This p tag is next to div tag</p><!—Not selected ->
```

```
#firstDiv .p1{
        font-style: italic;
        background-color: orange;
      }
```



```
This p tag is inside div tag

This is p inside span which is inside div

This p tag is next to div tag
```

2.7 Grouping of Selectors

If we have a requirement to apply the same style to all the headings like h1, h2, etc. on the webpage.
We may not like to repeat the same style code for all headings.
We may use grouping options in such scenarios and use style code only once.
Let us now learn how to use grouping for styling.

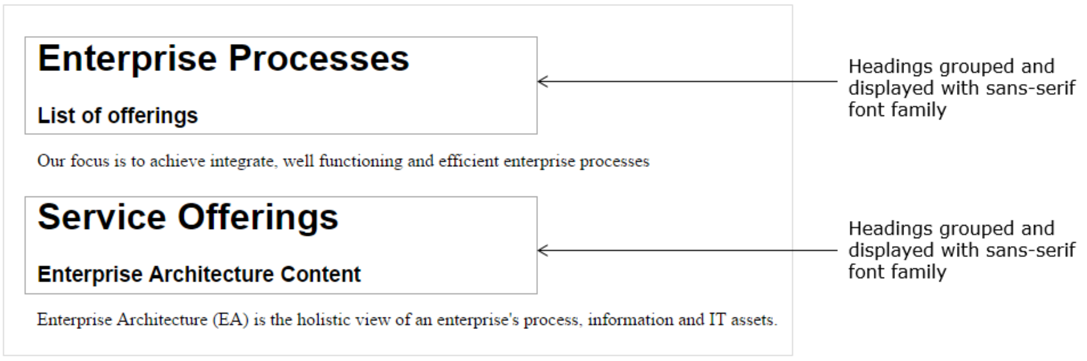
Grouping.html

```
<html>
<head>
         <title>Grouping</title>
         <body>
         <h1>Enterprise Processes</h1>
         <h3>List of offerings</h3>
         <p>Our focus is to achieve integrate,
         well functioning and efficient enterprise processes</p>
         <h1>Service Offerings</h1>
         <h3>Enterprise Architecture Content</h3>
         <p>Enterprise Architecture (EA) is the holistic
         view of an enterprise's process, information and IT assets.</p>
</body>
<html>
```

Mystyle.css

```
h1,h3 { font-family: sans-serif; }
```

Output



Enterprise Processes

List of offerings

Our focus is to achieve integrate, well functioning and efficient enterprise processes

Service Offerings

Enterprise Architecture Content

Enterprise Architecture (EA) is the holistic view of an enterprise's process, information and IT assets.

Headings grouped and displayed with sans-serif font family
Headings grouped and displayed with sans-serif font family

3. Inheritance

The child elements can inherit the style properties of the parent element but the styling is not always inherited by child elements.
Some of the CSS style properties have “inherit” value associated with them. For example, Font and Text properties are inherited but the border and margin properties are not inherited.
If the “inherit” value is associated with the property then styling is also inherited. Otherwise, styling is not inherited.
If we want the styling which is not inherited by default to be inherited then we have to specify inherit to the style attribute explicitly.

Example:

border: inherit

4. Cascading Order

We learned that CSS refers to cascading style sheets. We discussed how to use the style sheets, but you might be wondering what does this cascading mean?

Well cascading refers to the cascading order

If styling is mentioned using multiple ways ultimately they cascade to one style.

4.1 Priority order

- Inline
- Internal and/or import
- External
- Browser default: The browser adds some minimal styling to the web page. The browser styling will be overwritten when the developer includes styling to the webpage
  The styling added by the browser is not consistent across different browsers. Each browser has its own default styling.

4.2 Specificity

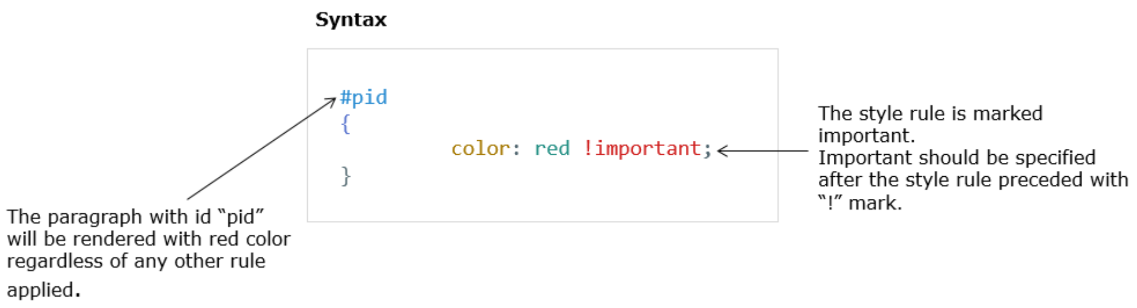
Thumb Rule: If two rules have the same weight, the latter wins.
For example, Imported style sheets and internal style sheets actually carry the same weight, but since imported styles are considered to be before any rules in the style sheet itself, the latter will win (the internal).
In addition to internal priority specificity also counts. A Specificity is a 3-digit number.
For example, if specificity is 123 then,

- 1 refers to the number of id selectors
- 2 refers to the number of class selectors
- 3 refers to the number of elements

Highest specificity always wins.

4.3 !important

If any styling is followed by !important. It is given the highest priority. !important override all previous styling and give this the highest priority.


Syntax

```
#pid
{
        color: red !important;
}
```

The style rule is marked important. Important should be specified after the style rule preceded with "!" mark.

The paragraph with id "pid" will be rendered with red color regardless of any other rule applied.

5. Pseudo Classes

A **CSS pseudo class** can be described as a *keyword that is added to a selector which defines a special state of the selected elements.* Using the same we can provide the styling to be applied when the HTML element changes its state.

For example, it can be used to:

- style an element when it gets focus, i.e. the state is "focus"
- style an element when a user hovers over it, i.e. the state is "hover"
- style visited and un-visited links differently

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

We can style our **links** based on their state using the following pseudo-classes.

Pseudo-class	Description
:link	It is used to style links which are not visited
:visited	Styles the state of the link when link is visited at least once
:hover	Styles the state of the link when mouse is placed over the link
:active	Styles the state of link when can you click on the link

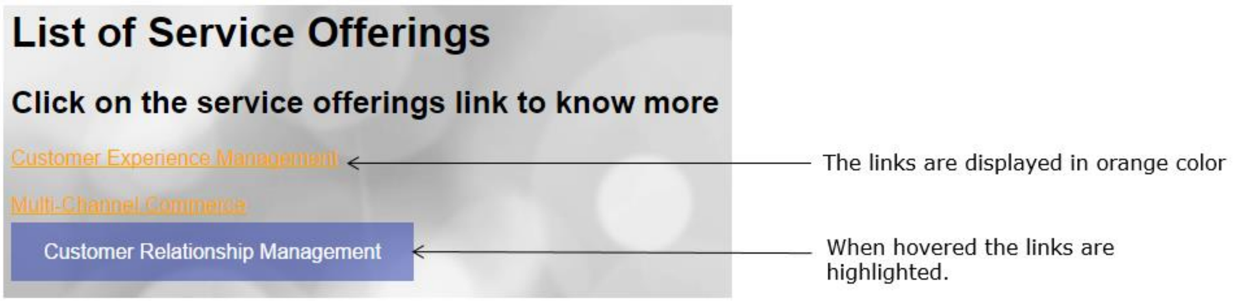
pseudoclass.html

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="link.css">
</head>
<body>
         <h1>List of Service Offerings</h1>
         <h2>Click on the service offerings link to know more</h2>
         <a href="Customer_expr_mngmt.html">Customer Experience Management</a><br><br>
         <a href="Multi_Channel_Commerce.html">Multi-Channel Commerce</a><br><br>
         <a href="Customer_Relationship_Management.html">Customer Relationship Management</a><br><br>
</body>
</html>
```

pseudoclass.html

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="link.css">
</head>
<body>
         <h1>List of Service Offerings</h1>
         <h2>Click on the service offerings link to know more</h2>
         <a href="Customer_expr_mngmt.html">Customer Experience Management</a><br><br>
         <a href="Multi_Channel_Commerce.html">Multi-Channel Commerce</a><br><br>
         <a href="Customer_Relationship_Management.html">Customer Relationship Management</a><br><br>
</body>
</html>
```

Output



List of Service Offerings

Click on the service offerings link to know more

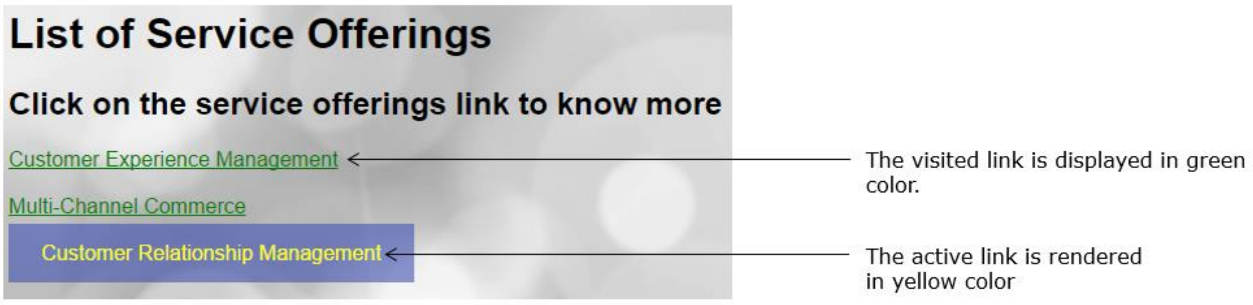
Customer Experience Management

Multi-Channel Commerce

Customer Relationship Management

The links are displayed in orange color

When hovered the links are highlighted.



List of Service Offerings

Click on the service offerings link to know more

Customer Experience Management

Multi-Channel Commerce

Customer Relationship Management

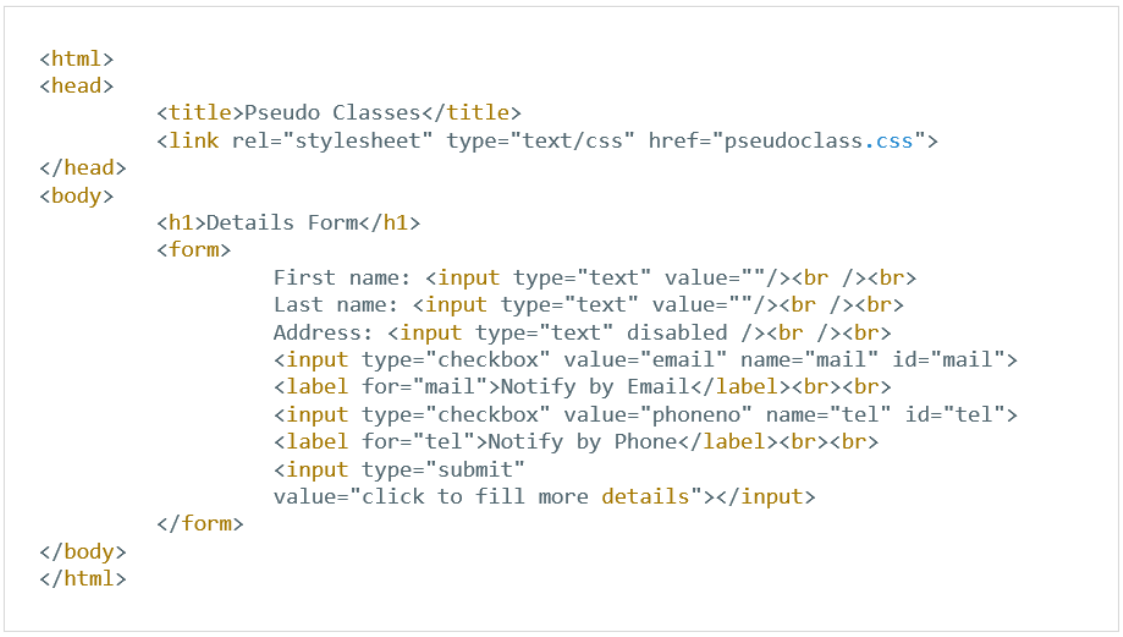
The visited link is displayed in green color.

The active link is rendered in yellow color

Input related pseudo-classes: We can style the input fields based on their state.

pseudoclass	Description
:focus	The state of the element that is currently targeted by keyboard or activated by mouse
:enabled	The state enabled is associated with input fields, text area, and dropdowns. By default, all the input elements will be enabled
:disabled	The state of the element is disabled when they are disabled using the HTML attribute disabled
:checked	The state of the element when they are selected. The checked state is associated with input elements and radio buttons
:indeterminate	The third state of checkbox apart from checked and unchecked.
:required	Represents the input element with the required attribute set.

pseudoclass.html

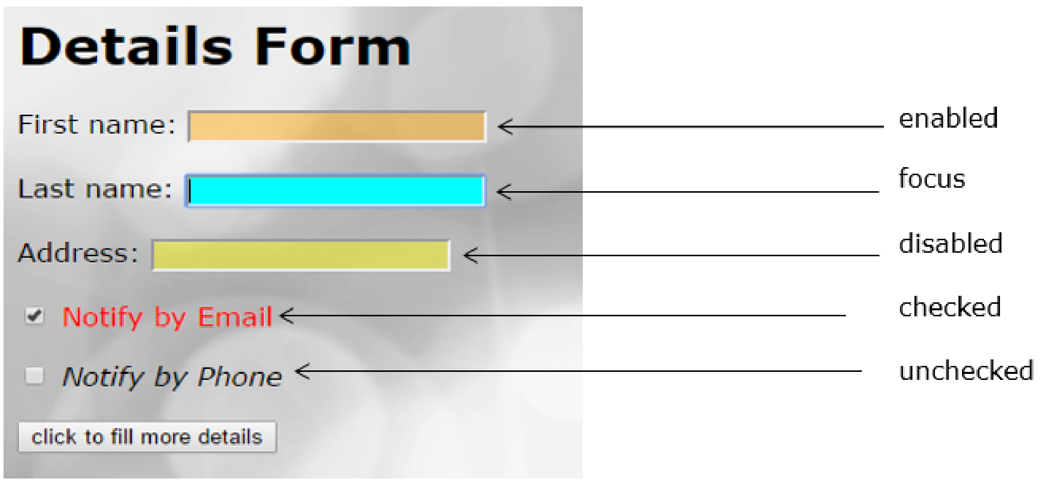


```
<html>
<head>
        <title>Pseudo Classes</title>
        <link rel="stylesheet" type="text/css" href="pseudoclass.css">
</head>
<body>
        <h1>Details Form</h1>
        <form>
                First name: <input type="text" value=""/><br /><br>
                Last name: <input type="text" value=""/><br /><br>
                Address: <input type="text" disabled /><br /><br>
                <input type="checkbox" value="email" name="mail" id="mail">
                <label for="mail">Notify by Email</label><br><br>
                <input type="checkbox" value="phoneno" name="tel" id="tel">
                <label for="tel">Notify by Phone</label><br><br>
                <input type="submit"
                value="click to fill more details"></input>
        </form>
</body>
</html>
```

pseudoclass.css

```css
input[type="text"]:enabled
{
        background:orange;
}
input[type="text"]:disabled
{
        background:yellow;
}
input[type="text"]:focus
{
        background:blue;
}
input[type=checkbox] + label
{
        color: #ccc; //grey color
        font-style: italic;
}
input[type=checkbox]:checked + label
{
        color: #f00; //red color
        font-style: normal;
}
```

Output



Position based pseudo-classes: We can style the HTML elements based on their position on the webpage.

pseudo-class	**Description**
:root	Selects the document’s root element.
A:firstchild	Selects every A element that is the first child of its parent.
A:last-child	Selects every A element that is the last child of its parent.
A B:nth-child(n)	Selects every B element that is the nth child of its parent A.

A:nth-of-type(n)	Select every A element that is nth A element of its parent.
A:first-of-type	Selects every A element that is first A element of its parent.
A:last-of-type	Selects every A element that is last A element of its parent.
A:nth-last-of-type(n)	Selects every A element that is the nth child of its parent counting from last.
A B:nth-last-child(n)	Selects every B element that is the nth child of its parent A, counting from the last child.
A:only-of-type	Selects every element A that is the only child of its type, of its parent.

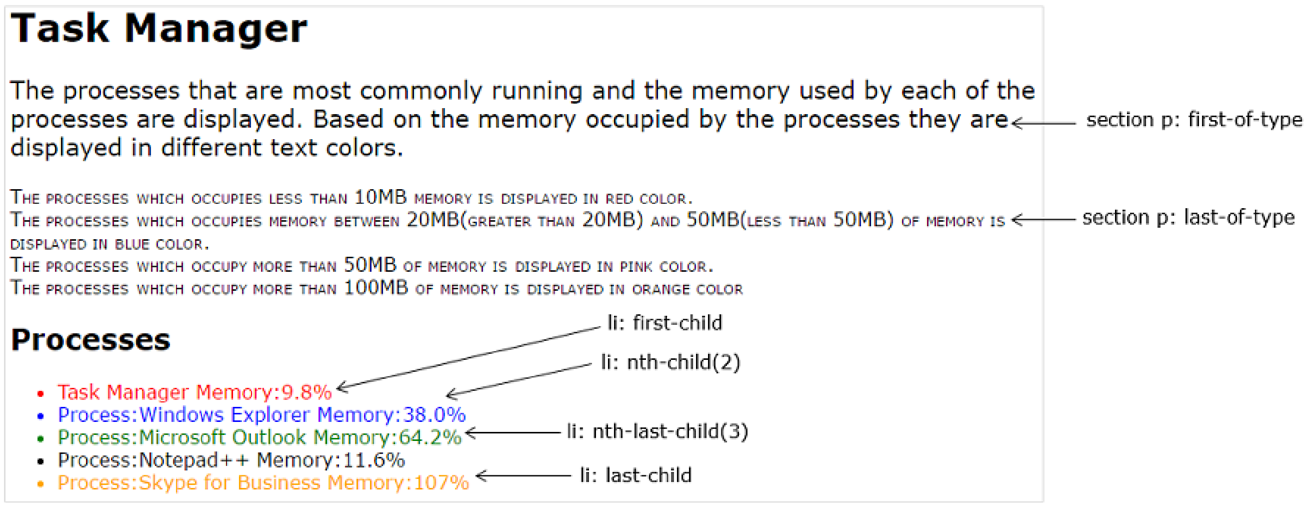
Pseudoclass.html

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="psuedoclass.css">
</head>
<body>
<h1>Task Manager</h1>
<section>
        <p>The processes that are most commonly running and the memory used by each of the  processes
        are displayed. Based on the memory occupied by the processes they are displayed in different
        text colors.</p>
        <p>The processes which occupies less than 10MB memory is displayed in red color.<br>
        The processes which occupies memory between 20MB(greater than 20MB) and 50MB(less than
        50MB) of memory is displayed in blue color.<br>
        The processes which occupy more than 50MB of memory is displayed in green color.<br>
        The processes which occupy more than 100MB of memory is displayed in orange color</p><br><br>
</section>
<h2>Processes</h2>
<ul>
        <li>Task Manager Memory:9.8%</li>
        <li>Process:Windows Explorer Memory:38.0%</li>
        <li>Process:Microsoft Outlook Memory:64.2%</li>
        <li>Process:Notepad++ Memory:11.6%</li>
        <li>Process:Skype for Business Memory:107%</li>
</ul>
</body>
</html>
```

Pseudoclasses.css

```
li:first-child { color: red; }
li:nth-child(2) { color: blue; }
li:last-child { color: orange;}
li:nth-last-child(3) {color: green; }
section p:first-of-type { font-size:20px; }
section p:last-of-type { font-variant: small-caps;}
```

Output



Task Manager

The processes that are most commonly running and the memory used by each of the processes are displayed. Based on the memory occupied by the processes they are displayed in different text colors.

THE PROCESSES WHICH OCCUPIES LESS THAN 10MB MEMORY IS DISPLAYED IN RED COLOR.
THE PROCESSES WHICH OCCUPIES MEMORY BETWEEN 20MB(GREATER THAN 20MB) AND 50MB(LESS THAN 50MB) OF MEMORY IS DISPLAYED IN BLUE COLOR.
THE PROCESSES WHICH OCCUPY MORE THAN 50MB OF MEMORY IS DISPLAYED IN PINK COLOR.
THE PROCESSES WHICH OCCUPY MORE THAN 100MB OF MEMORY IS DISPLAYED IN ORANGE COLOR

Processes

- Task Manager Memory:9.8%
- Process:Windows Explorer Memory:38.0%
- Process:Microsoft Outlook Memory:64.2%
- Process:Notepad++ Memory:11.6%
- Process:Skype for Business Memory:107%

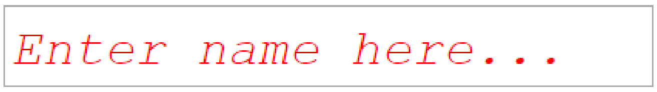
6. Pseudo Elements

A CSS **pseudo-element** can be taken as a keyword that is added to a selector which lets you to design a specific part of the selected elements.
For example, it can be *placeholder of input field* or *first line of the paragraph*. Also, we can add content before and after the text using pseudo elements.
They are usually **denoted by two colons (::) together**, unlike *single colon (:) for pseudo classes*.

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

Pseudo-element	Description
::first-line	It is used to style first line of the text
::first-letter	It is used to style first letter of the text. It is applied to only block level elements
::before	It is used to insert some content or style before the content of an element
::after	It is used to insert some content or style after the content of an element
::selection	It is used to style the portion of an element that is selected by the user

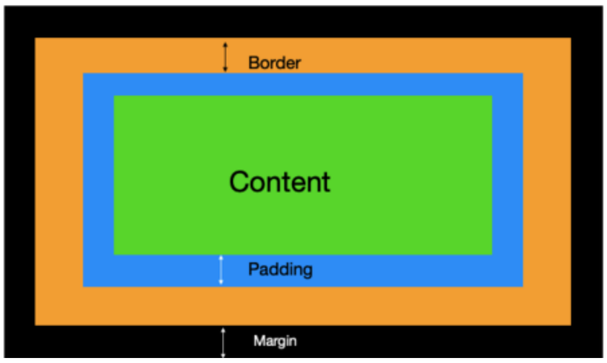
```
<head>
   <style>
      input::placeholder{
         color:red;
         font-style:italic;
         font-family: 'Courier New', Courier, monospace;
      }
   </style>
</head>
<body>
   <input type="text"  placeholder="Enter name here...">
</body>
```



```
Enter name here...
```

7. Box Model

The **CSS box model** is basically an invisible box which wraps around every HTML element. Thus, HTML elements are considered as boxes, be it a <div> tag or a <p> tag or any other tag. In CSS, the term "**Box Model**" is used when we are talking about design and layout.



Total Width and Height of an HTML element

When we specify the width and height property for any HTML element using CSS, we are actually setting the width and height of the content not the width and height of the complete element.
The complete width and height of an element include the padding, border
Total element width = width + left padding + right padding + left border + right border
Total element height = height + top padding + bottom padding + top border + bottom border

7.1 Box Sizing

Box-sizing defines how the height and the width of an element is to be calculated, whether to include the paddings and borders or not.
By giving border-box as the value of box-sizing property what we are doing is we are saying include the borders and paddings as well. Now the total width/height of the box is 100px only which includes 20px + 20px of borders as well.

There are mainly two types of box-sizing:

- **Content-box** (by default, it does not include borders and padding)
- **Border-box** (it includes border and padding as well in the size of the box)

7.2 Box-Shadow

The **box-shadow** CSS property adds shadow effects around an element's frame. You can set multiple effects separated by commas. A box shadow is described by X and Y offsets relative to the element, blur and spread radius, and color.

/* Keyword values */

box-shadow: none;

/* offset-x | offset-y | color */

box-shadow: 60px -16px teal;

```
/* offset-x | offset-y | blur-radius | color */

box-shadow: 10px 5px 5px black;


/* offset-x | offset-y | blur-radius | spread-radius | color */

box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2);


/* inset | offset-x | offset-y | color */

box-shadow: inset 5em 1em gold;


/* Any number of shadows, separated by commas */

box-shadow: 3px 3px red, -1em 0 0.4em olive;


/* Global keywords */

box-shadow: inherit;

box-shadow: initial;

box-shadow: revert;

box-shadow: unset;
```

7.3 Overflow

The CSS overflow property controls what happens to content when it is too big to fit into an area.
It specifies whether to clip the contents or to add scroll bars.

Overflow: hidden
The overflow will be clipped, and the rest of the content is hidden.

Overflow: visible
The overflowing content will not be clipped and will be rendered outside the box. By default, the overflow is visible.

Overflow: scroll
The overflowing content will be clipped but neither it is going to be lost nor it is going to be displayed outside the box but instead a scrollbar would be added to scroll inside the box.

Overflow: auto
The auto value is like scroll, but it adds scrollbars only when necessary.

8. Padding Properties

The CSS padding property can be used to generate space around the content.
The padding property is used to set some white space between the element's content and the element's border.
Padding is the shorthand property for specifying padding-top, padding-right, padding-bottom, padding-left.
The padding property can take from 1 to 4 values.

```
/* Apply to all four sides */
padding: 1em;

/* vertical | horizontal */
padding: 5% 10%;

/* top | horizontal | bottom */
padding: 1em 2em 2em;

/* top | right | bottom | left */
padding: 5px 1em 0 2em;
```

9. Margin Properties

Margin is a short hand property to set top, right, bottom and left properties.

CSS has many shorthand properties. The short hand properties in CSS can take from 1 to 4 values. The order of these values are top, right, bottom and left
If 1 value is specified then the same value is considered for all other 3 values.
If 2 values are specified then the first value is considered as top and bottom values and the second value is considered as left and right values.
If 3 values are specified then the first value is considered as the top value and the second value is considered as left and right values and the third value is considered as bottom value.

Mastering margin collapsing

The top and bottom margins of blocks are sometimes combined (collapsed) into a single margin whose size is the largest of the individual margins (or just one of them, if they are equal), a behavior known as **margin collapsing**. Note that the margins of floating and absolutely positioned elements never collapse.
Margin collapsing occurs in three basic cases:

Adjacent siblings
The margins of adjacent siblings are collapsed (except when the latter sibling needs to be cleared past floats).

No content separating parent and descendants
If there is no border, padding, inline part, block formatting context created, or *clearance* to separate the margin-top of a block from the margin-top of one or more of its descendant blocks; or no border, padding, inline content, height, or min-height to separate the margin-bottom of a block from the margin-bottom of one or more of its descendant blocks, then those margins collapse. The collapsed margin ends up outside the parent.

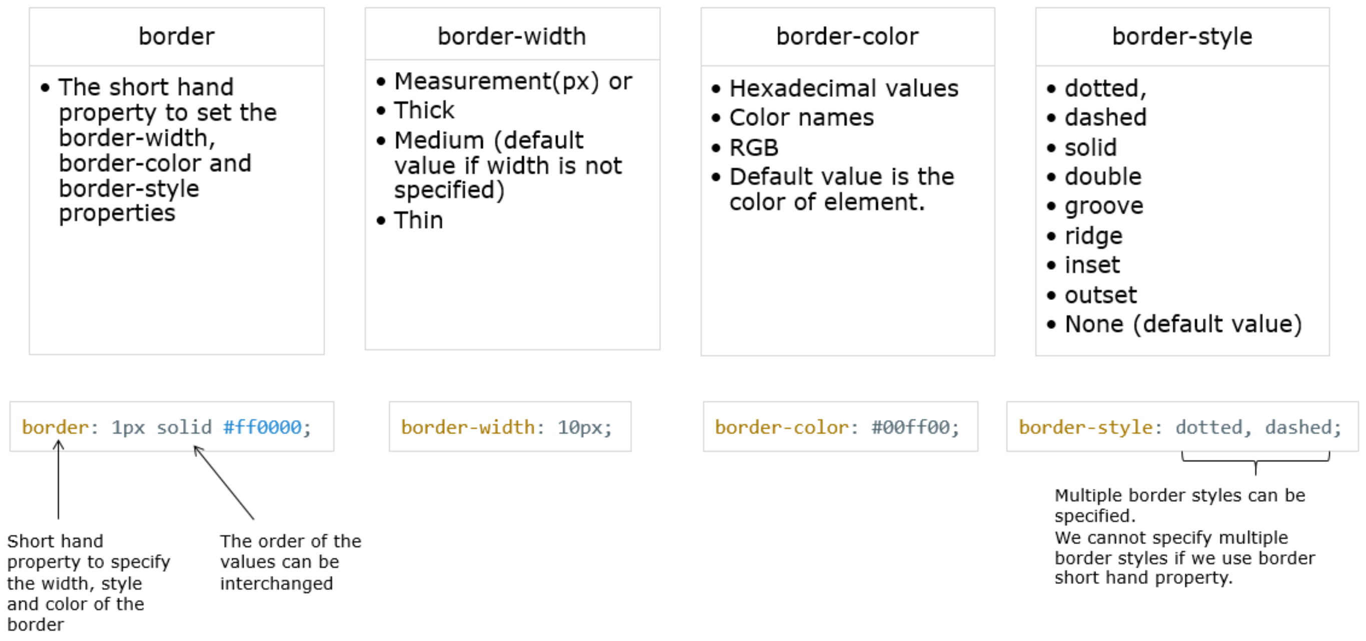
Empty blocks
If there is no border, padding, inline content, height, or min-height to separate a block's margin-top from its margin-bottom, then its top and bottom margins collapse.

Some things to note:

- More complex margin collapsing (of more than two margins) occurs when the above cases are combined.
- These rules apply even to margins that are zero, so the margin of a descendant ends up outside its parent (according to the rules above) whether or not the parent's margin is zero.
- When negative margins are involved, the size of the collapsed margin is the sum of the largest positive margin and the smallest (most negative) negative margin.
- When all margins are negative, the size of the collapsed margin is the smallest (most negative) margin. This applies to both adjacent elements and nested elements.

10. Border Properties

Applying an appropriate border to the HTML element of your page can improve the appearance of your webpage.
We have a lot of scenarios where we can apply the border to a few of our elements.



border
- The short hand property to set the border-width, border-color and border-style properties

border-width
- Measurement(px) or
- Thick
- Medium (default value if width is not specified)
- Thin

border-color
- Hexadecimal values
- Color names
- RGB
- Default value is the color of element.

border-style
- dotted,
- dashed
- solid
- double
- groove
- ridge
- inset
- outset
- None (default value)

```
border: 1px solid #ff0000;
```

```
border-width: 10px;
```

```
border-color: #00ff00;
```

```
border-style: dotted, dashed;
```

Short hand property to specify the width, style and color of the border
The order of the values can be interchanged
Multiple border styles can be specified. We cannot specify multiple border styles if we use border short hand property.

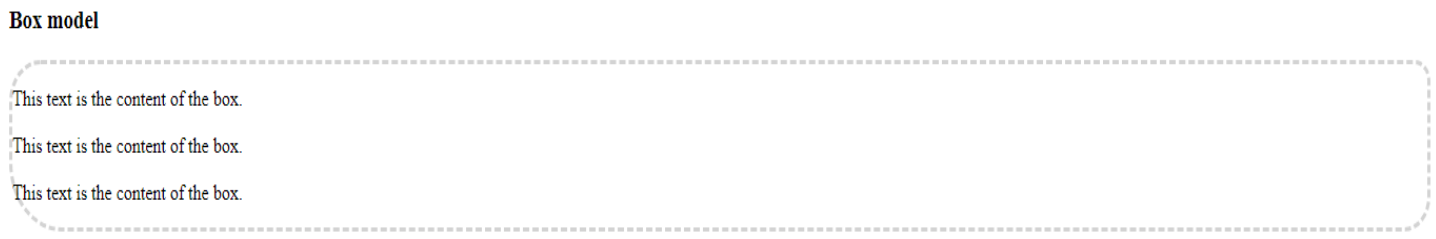
10.1 Border- radius

Border property adds a sharp border to our HTML element.
If we need rounded corners for our border, we can achieve the requirement using the border-radius property of CSS3.
The border-radius is a shorthand property. Border radius value can be expressed using px or %.
Percentage(%) is used when we want border-radius to be directly correlated with elements width.
border-radius is a combination of below four properties.

- border-top-left-radius
- border-top-right-radius
- border-bottom-right-radius
- border-bottom-left-radius

```
border: lightgrey dashed 3px;

border-radius: 30px 15px 25px 50px;
```



Box model

This text is the content of the box.

This text is the content of the box.

This text is the content of the box.

10.2 Border- image

Border image is used to specify an image as a border around the HTML element. Border image is a short-hand property to set border-source, border-image-slice, border-width, border-outset, border repeat.

border-image	**description**
border-source	Path of the image to be used for the border.
border-image-slice	Specifies the values to slice the image. Can have 1 to 4(top, right, bottom, left) values expressed as pixels or %.
border-width	Specifies the width of the border-image.
border-outset	Specifies the amount by which border image area extends beyond border-box.
border-repeat	Specifies whether the edges of the image should be stretched, repeated, rounded or scaled.

11. Classification Properties

11.1 display

The **display** CSS property sets whether an element is treated as a block or inline element and the layout used for its children, such as flow layout, grid or flex.

11.2 cursor

The **cursor** CSS property sets the type of mouse cursor, if any, to show when the mouse pointer is over an element.

11.3 visibility

The **visibility** CSS property shows or hides an element without changing the layout of a document.

11.4 vertical-align

The **vertical-align** CSS property sets vertical alignment of an inline, inline-block or table-cell box.

```
vertical-align: baseline;
vertical-align: top;
vertical-align: middle;
vertical-align: bottom;
vertical-align: sub;
vertical-align: text-top;
```

11.5 list-style-type

The **list-style-type** CSS property sets the marker (such as a disc, character, or custom counter style) of a list item element.

11.6 list-style-position

The **list-style-position** CSS property sets the position of the ::marker relative to a list item.

11.7 list-style-image

The **list-style-image** CSS property sets an image to be used as the list item marker.

11.8 list-style

The **list-style** CSS shorthand property allows you to set all the list style properties at once.

This property is a shorthand for the following CSS properties:

- list-style-image
- list-style-position
- list-style-type

```
list-style: square;
list-style: inside;
list-style: url('/media/examples/rocket.svg');
list-style: none;
list-style: georgian inside url('/media/examples/rocket.svg');
list-style: georgian outside url('/non-existent.svg');
```

12. Position Properties

The **CSS Position** property sets how an element is positioned in a webpage.
We have following properties which helps us to position our elements:

- Top
- Bottom
- Left
- Right

These properties let us decide the exact location of our element to be placed.

Types of position

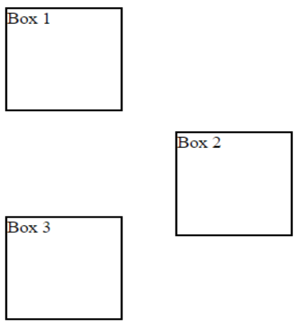
12.1 Static

All the html elements are positioned *static by default*. These are not affected by the top, right, bottom and left properties. If you scroll the page the page up, all the statically positioned elements will go up.

12.2 Relative

Just like the static position, the element's original position remains in the flow of the document but now the top, right, bottom, left and z-index properties will work. The element will get positioned relative to its original position (i.e. if we do not give any top/right/bottom/left values). Other content will not be adjusted to fit into any kind of gap left by the element.

```
<head>
  <style>
    div {
      Border: 2px solid black;
      Height: 100px;
      Width: 100px;
     }
   #box2 {
      Position: relative;
      Top: 20px;
      Left: 150px;
     }
  </style>
</head>
```



As it is evident from the output, Box 2 has shifted from its original position. **It has not shifted with respect to Box 1 or Box 3, but it has shifted with respect to its original position**. It has shifted 50px bottom with respect to its original position and has shifted 200 px right from its original position. We gave Top as 50px, this means that spacing from top will be 50px and similarly with Left, i.e. it has shifted right but the original position is now 150px Left away from it.

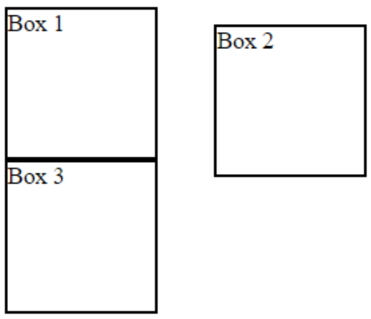
There is one more thing to notice, that is Box 1 and Box 3. *There has been no change on Box 1 and Box 3 by the position of Box 2.*

12.3 Absolute

Absolute positions elements against *containing element* (parent element).

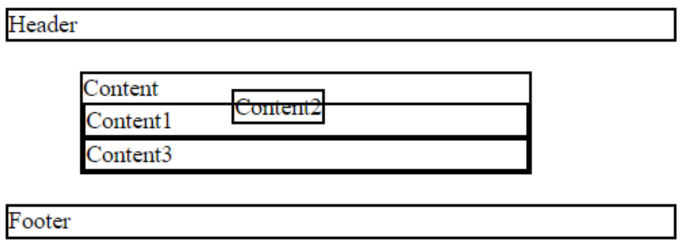
Unlike Static and Relative, Absolute behaves differently. The element is removed from the flow of the document. It is positioned relative to its closest positioned ancestor (if there is any and not static), else it is placed relative to the browser, this means if parent tag is not having explicit position but the grandparent has position , then absolute will be with respect to grandparent and not the browser or its own parent. Its Ultimate position is determined by the values of top/right/bottom/left.

```
<head>
  <style>
     div {
       Border: 2px solid black;
       Height: 100px;
       Width: 100px;
     }
     #box2 {
       Position: absolute;
       Top: 50px;
       Left: 150px;
     }
  </style>
</head>
```



Box 2 has been completely taken out of the flow of the document and has been placed at the position specifically mentioned with respect to the browser screen (because there is no containing element in this example). **One more thing to note here is that Box 3 has shifted up because it behaves as if Box 2 was never there in the first place.**

```
<html>
   <head>
      <style>
         div {
            border: 2px solid black;
         }
         #content {
            position: relative;
            left: 30px;
            margin: 20px;
            width:200px;
         }
         #content2 {
            position: absolute;
            top: 10px;
            left: 100px;
         }
      </style>
   </head>
   <body>
      <div id = "header">Header</div>
      <div id = "content">Content
         <div id = "content1">Content1</div>
         <div id = "content2">Content2</div>
         <div id = "content3">Content3</div>
      </div>
      <div id = "footer">Footer</div>
   </body>
</html>
```



The output is no doubt a bit messy, but it will help you understand the absolute position very clearly.

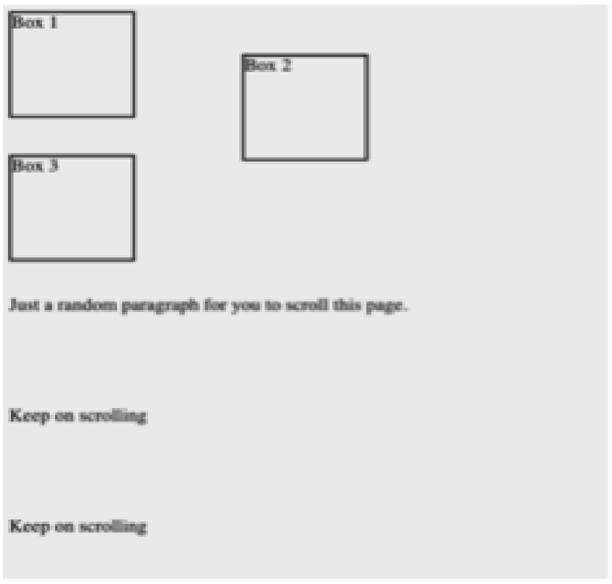
Content div is the parent element and content 1, 2 and 3 comes under it. Here we are applying position absolute to content 2 by giving top as 10px and left as 100px, now because its parent is also positioned (content is positioned relative), content 2 took the absolute position inside the content div.

If content div was given **static position** or **no position** (*static is by default*), then the output would be different. Then content 2 box would have been positioned with respect to the browser, it would be merging with the header div, you can try this out by yourself also.

12.4 Fixed

Just like Absolute, the element is removed from the flow of the document. Interesting thing about fixed is it sticks onto the browser window and even if we scroll it will not move.

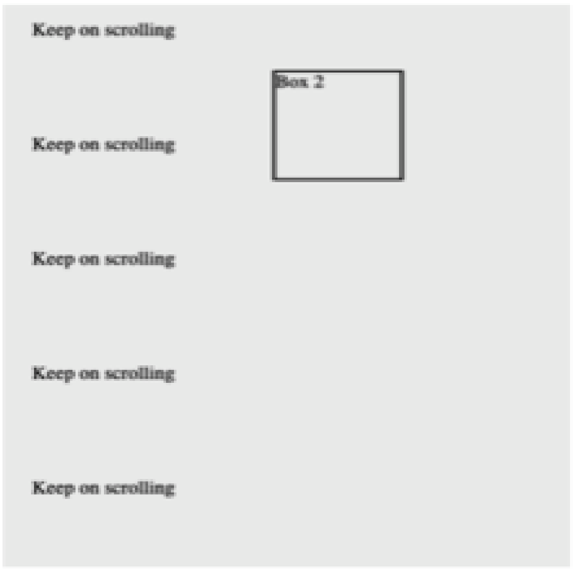
```
<head>
  <style>
    div {
      Border: 2px solid black;
      Height: 100px;
      Width: 100px;
     }
   #box2 {
      Position: fixed;
      Top: 20px;
      Left: 200px;
    }
   .p {
     Font-size: x-large;
    }
  </style>
</head>
```



Just a random paragraph for you to scroll this page.

Keep on scrolling

Keep on scrolling



Keep on scrolling

Keep on scrolling

Keep on scrolling

Keep on scrolling

Keep on scrolling

From the two outputs Box 2 has taken a spot on the browser window and cannot be moved even if we keep on scrolling. Another point worth noting is that Box 3 has shifted up as if Box 2 was never there in the first place.

12.5 Sticky

It is a combination of Static as well as Fixed. Initially it will behave as Static but as soon as it reaches the top/bottom of the screen when we scroll then it gets fixed there even if we keep on scrolling.

position property	**Description**
static	Default value. Elements render in order, as they appear in the document flow. The top, right, bottom and left values has no effect
relative	The element is positioned relatively. The top, right, bottom and right values will be applied relative to the element itself. Ex: left:20px will move the element to the left by 20px from it's current position. The 20px is not w.r.t. the viewport is w.r.t the element's current position.
absolute	The element is removed from the normal document flow and will be placed in an exact location. If there is a parent-child relationship the element is positioned relative to its first positioned (not static) ancestor element. If there is no parent element whose position is other than static then the top, right, bottom and left values are relative to the viewport (<html> element).
fixed	The element is positioned relative to viewport or browser. The viewport doesn't change when the window is scrolled.

12.6 Z-index

Stacking is a process of conceptualizing an HTML element as a three-dimensional structure (3D Structure). It gives priority to the HTML elements based on which the element's appearance changes on loading the HTML document on the view port. The order of rendering the HTML elements will be based on the stacking value.

The process of prioritizing the order of rendering HTML elements can be achieved using a CSS Property called 'z-index'. In general, whenever we deal with 3D Structure analysis, the third axis is always called Z-axis. Hence the name, 'z-index'. This property can be assigned with a number that influences the order of rendering elements on the viewport. Stacking can be used on element which is positioned (the position property should have either absolute or relative as its value).

Z-index specifies how the elements should be stacked on a web page. Element with the greatest stack order is always at the front.
Z-index works only on positioned elements other than a static position. Value of z-index can be any number including negative values.
When there is a parent-child relationship between the elements then the z-index value of the child will be relative to its parent z-index value.

zindex.html

```
<html>
<head>
         <title>Z-Index</title>
         <link rel="stylesheet" type="text/css" href="zindex.css">

</head>
<body> <br/>
<div id="div1"> <br/>
         <span class="bold">DIV #1</span> <br/>position: relative;
         <div id="div2"> <br />
             <span class="bold">DIV #2</span> <br/>position: absolute; <br/>z-index: 1;
         </div>
</div> <br />
<div id="div3"> <br/>
         <span class="bold">DIV #3</span> <br/>position: relative;
         <div id="div4"> <br/>
              <span class="bold">DIV #4</span> <br/>position: absolute; <br/>z-index: 2;
         </div>
</div>
</body>
</html>
```

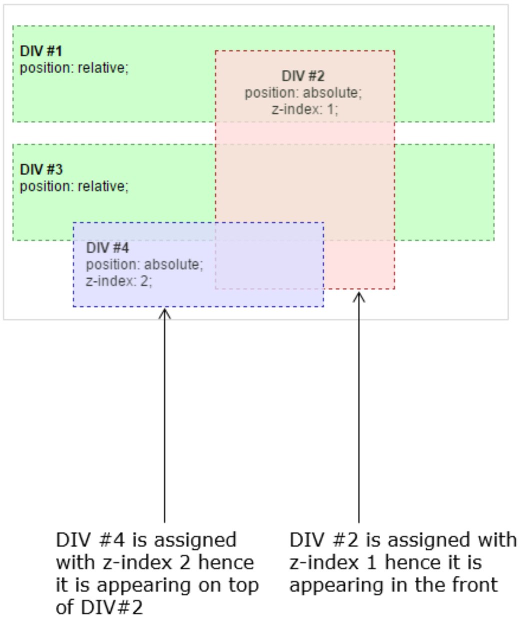
zindex.css

```
div
{
   font: 12px Arial;
}
span.bold
{
   font-weight: bold;
}
#div1,#div3
{
   background-color: #ccffcc;
   border: 1px dashed #669966;
   padding-left: 5px;
   position: relative;
   height: 80px;
}
```

```
#div2
{
   background-color: #ffdddd;
   border: 1px dashed #990000;
   height: 200px;
   left: 170px;
   opacity: 0.8;
   position: absolute;
   text-align: center;
   top: 20px;
   width: 150px;
   z-index: 1;
}
```

```
#div4
{
   background-color: #ddddff;
   border: 1px dashed #000099;
   left: 50px;
   opacity: 0.8;
   padding-left: 10px;
   position: absolute;
   text-align: left;
   top: 65px;
   width: 200px; height: 70px;
   z-index: 2;
}
```

output with z-index



DIV #1
position: relative;
DIV #2
position: absolute;
z-index: 1;
DIV #3
position: relative;
DIV #4
position: absolute;
z-index: 2;
DIV #4 is assigned with z-index 2 hence it is appearing on top of DIV#2
DIV #2 is assigned with z-index 1 hence it is appearing in the front

12.7 Float and Clear

Float property specifies whether an element should float or not, where text and inline elements will wrap around it.

The property can take 3 values: left, right and none.

float property	Description
left	Indicates that element should float on the left of its containing block
right	Indicates that element should float on the right of its containing block
none	Indicates that the element will not float. Default value

Clear property specifies that no elements should float on a particular element.

clear property	Description
left	Indicates that floating elements are not allowed on the left side
right	Indicates that floating elements are not allowed on the right side
both	Indicates that floating elements are not allowed on both the sides
none	Indicates floating elements are allowed on both sides. Default value

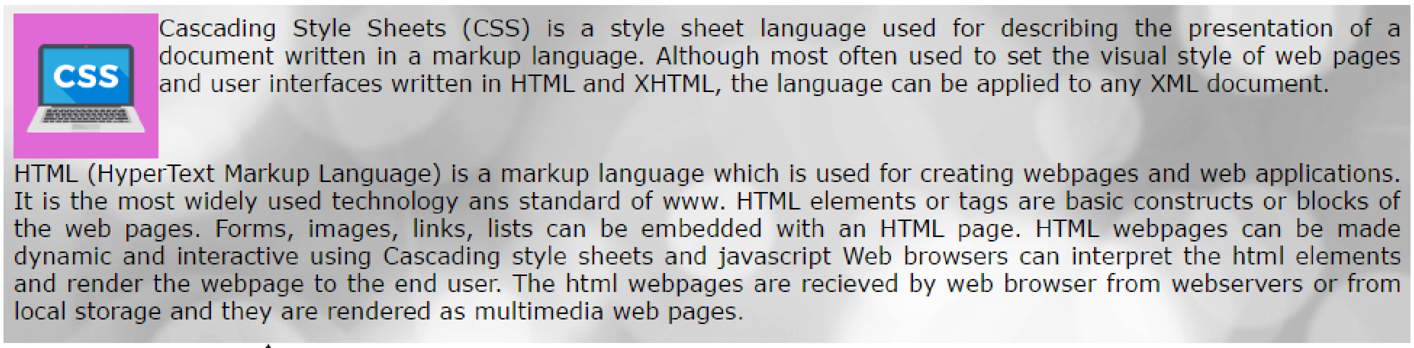
float.html

```
<html>
<head> <title>CSS Float Property</title>
<link rel="stylesheet" type="text/css" href="clear.css"> </head>
<body>
<img id="cssimage" src="images/css.png" alt="image cannot be loaded" width="100px" height="100px"></img>
<p> Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document
written in a markup language. Although most often used to set the visual style of web pages and user interfaces
written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and
is applicable to rendering in speech, or on other media.
Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging
webpages, user interfaces for web applications, and user interfaces for many mobile applications.
</p>
<p id="clearfloat"> Float and clear The float property may have one of three values.
Absolutely positioned or fixed items cannot be floated. Other elements normally flow around floated items, unless
they are prevented from doing so by their clear property.
left: The item floats to the left of the line that it would have appeared in; other items may flow around its
right side.
right: The item floats to the right of the line that it would have appeared in; other items may flow around its
left side.
clear: Forces the element to appear underneath ('clear') floated elements to the left (clear:left), right
(clear:right) or both sides (clear:both).
</p>
</body>
</html>
```

clear.css

```
body{font-family: Verdana;}
p{text-align: justify;}
#cssimage {float: left;}
#clearfloat{clear: both;}
```

Output with clear



Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document.

HTML (HyperText Markup Language) is a markup language which is used for creating webpages and web applications. It is the most widely used technology ans standard of www. HTML elements or tags are basic constructs or blocks of the web pages. Forms, images, links, lists can be embedded with an HTML page. HTML webpages can be made dynamic and interactive using Cascading style sheets and javascript Web browsers can interpret the html elements and render the webpage to the end user. The html webpages are recieved by web browser from webservers or from local storage and they are rendered as multimedia web pages.

The second paragraph is specified with clear property.
So the second paragraph doesn’t wrap around the
image. Hence it starts in the new line.

13. Background Properties

13.1 background-image

URL	Specifies the source of the background-image
none	Specifies that there is no background image. Default value

13.2 background-color

The **background-color** CSS property sets the background color of an element.

13.3 background-size

The **background-size** CSS property sets the size of the element's background image. The image can be left to its natural size, stretched, or constrained to fit the available space.

Values

contain

Scales the image as large as possible within its container without cropping or stretching the image. If the container is larger than the image, this will result in image tiling, unless the background-repeat property is set to no-repeat.

cover

Scales the image as large as possible to fill the container, stretching the image if necessary. If the proportions of the image differ from the element, it is cropped either vertically or horizontally so that no empty space remains.

auto

Scales the background image in the corresponding direction such that its intrinsic proportions are maintained.

<length>

Stretches the image in the corresponding dimension to the specified length. Negative values are not allowed.

<percentage>

Stretches the image in the corresponding dimension to the specified percentage of the background positioning area. The background positioning area is determined by the value of background-origin (by default, the padding box). However, if the background's background-attachment value is fixed, the positioning area is instead the entire viewport. Negative values are not allowed.

13.4 background-clip

The **background-clip** CSS property sets whether an element's background extends underneath its border box, padding box, or content box.

13.5 background-origin

The background-origin CSS property sets the background's origin: from the border start, inside the border, or inside the padding.

13.6 background-attachment

The background-attachment CSS property sets whether a background image's position is fixed within the viewport, or scrolls with its containing block.

scroll

When the webpage is scrolled the background-image also scrolls. Default value

fixed

When the webpage has scrolled the position of the image remains fixed

13.7 background-repeat

The background-repeat CSS property sets how background images are repeated. A background image can be repeated along the horizontal and vertical axes, or not repeated at all.

repeat	Repeat will repeat the image across x and y-axis. Default Value
repeat-x	Repeat-x will repeat the image across the x-axis.
repeat-y	Repeat-y will repeat the image across the y-axis.
no-repeat	No-repeat will not repeat the image.

13.8 background-position

The background-position CSS property sets the initial position for each background image. The position is relative to the position layer set by background-origin.

Values

<position>

A <position>. A position defines an x/y coordinate, to place an item relative to the edges of an element's box. It can be defined using one to four values. If two non-keyword values are used, the first value represents the horizontal position and the second represents the vertical position. If only one value is specified, the second value is assumed to be center. If three or four values are used, the length-percentage values are offsets for the preceding keyword value(s).

1-value syntax: the value may be:

- The keyword value center, which centers the image.
- One of the keyword values top, left, bottom, right. This specifies an edge against which to place the item. The other dimension is then set to 50%, so the item is placed in the middle of the edge specified.
- A <length> or <percentage>. This specifies the X coordinate relative to the left edge, with the Y coordinate set to 50%.

2-value syntax: one value defines X and the other defines Y. Each value may be:

- One of the keyword values top, left, bottom, right. If left or right are given here, then this defines X and the other given value defines Y. If top or bottom are given, then this defines Y and the other value defines X.
- A <length> or <percentage>. If the other value is left or right, then this value defines Y, relative to the top edge. If the other value is top or bottom, then this value defines X, relative to the left edge. If both values are <length> or <percentage> values, then the first defines X and the second Y.
- Note that: If one value is top or bottom, then the other value may not be top or bottom. If one value is left or right, then the other value may not be left or right. This means, e.g., that top top and left right are not valid.
- The default value is top left or 0% 0%.

3-value syntax: Two values are keyword values, and the third is the offset for the preceding value:

- The first value is one of the keyword values top, left, bottom, right, or center. If left or right are given here, then this defines X. If top or bottom are given, then this defines Y and the other keyword value defines X.
- The <length> or <percentage> value, if it is the second value, is the offset for the first value. If it is the third value, it is the offset for the second value.
- The single length or percentage value is an offset for the keyword value that precedes it. The combination of one keyword with two <length> or <percentage> values is not valid.

4-value syntax: The first and third values are keyword value defining X and Y. The second and fourth values are offsets for the preceding X and Y keyword values:

- The first value and third values one of the keyword values top, left, bottom, right. If left or right are given here, then this defines X. If top or bottom are given, then this defines Y and the other keyword value defines X.
- The second and fourth values are <length> or <percentage> values. The second value is the offset for the first keyword. The fourth value is the offset for the second keyword.

13.9 background

The background shorthand CSS property sets all background style properties at once, such as color, image, origin and size, or repeat method.

- The <bg-size> value may only be included immediately after <position>, separated with the '/' character, like this: "center/80%".
- The <box> value may be included zero, one, or two times. If included once, it sets both background-origin and background-clip. If it is included twice, the first occurrence sets background-origin, and the second sets background-clip.

```
/* Using a <background-color> */

background: green;

/* Using a <bg-image> and <repeat-style> */

background: url("test.jpg") repeat-y;

/* Using a <box> and <background-color> */

background: border-box red;

/* A single image, centered and scaled */

background: no-repeat center/80% url("../img/image.png");

/* Global values */

background: inherit;

background: initial;

background: revert;

background: unset;
```

14. CSS Units

14.1 Absolute length units

These are the units which do not depend on the viewport. They remain same in all possible screen dimensions; i.e., the elements will not dynamically change its dimension based on the screen size.

Name	Unit
Centimeter	Cm
Inch	In
Millimeter	Mm
Pixel	Px

Most widely used absolute length is pixels since it has a crystal-clear control over the elements. No other unit is as accurate as pixel on a computer screen.

Note: Viewport is the webpage's area of screen which is visible for the user.

14.2 Relative units

These are the units which either depend on the viewport size or the properties of the parent elements in which the current element is wrapped up.
These units change the element's size relatively with respect to changes of all other elements.
Different relative units are:

14.2.1 vw and vh

These units provide dimension for elements which are related to the width and height of the viewport.

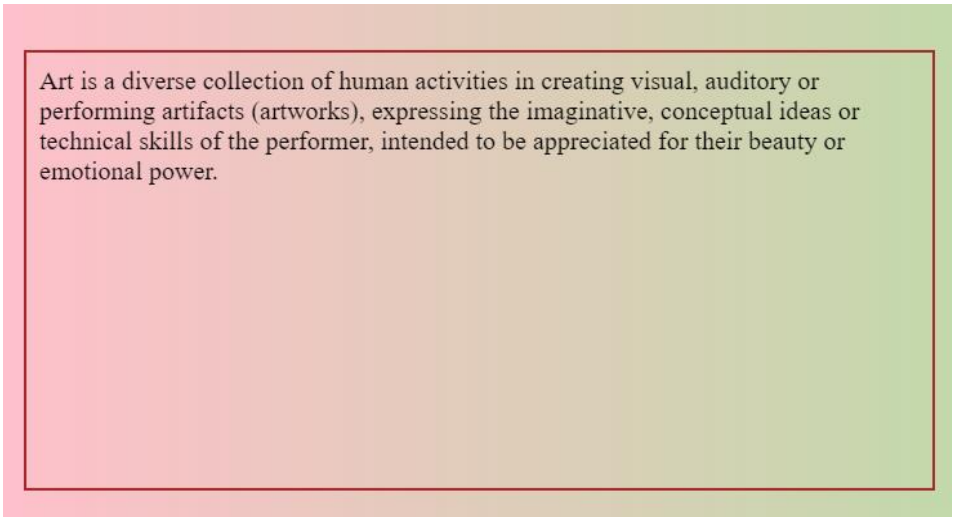
```
<div class="vwUnit">
   Art is a diverse collection of human activities in
   creating visual, auditory or performing artifacts
   (artworks), expressing the imaginative,
   conceptual ideas or technical skills of the performer,
    intended to be appreciated for their beauty or emotional power.
</div>
```

```
.vwUnit{
   border : 2px solid brown;
   width : 50vw;
   height: 50vh;
}
```

Note: vw and vh doesn't consider padding and margins while computing the width and height relatively based on the viewport size, whereas, percentage units consider padding and margins.

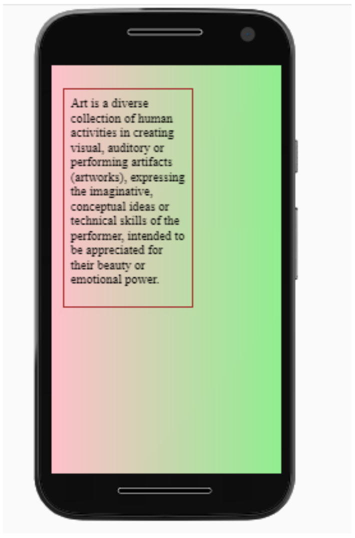
The following view represents the paragraph when different screen sizes are used with vw and vh units.

For a normal laptop or desktop screen,



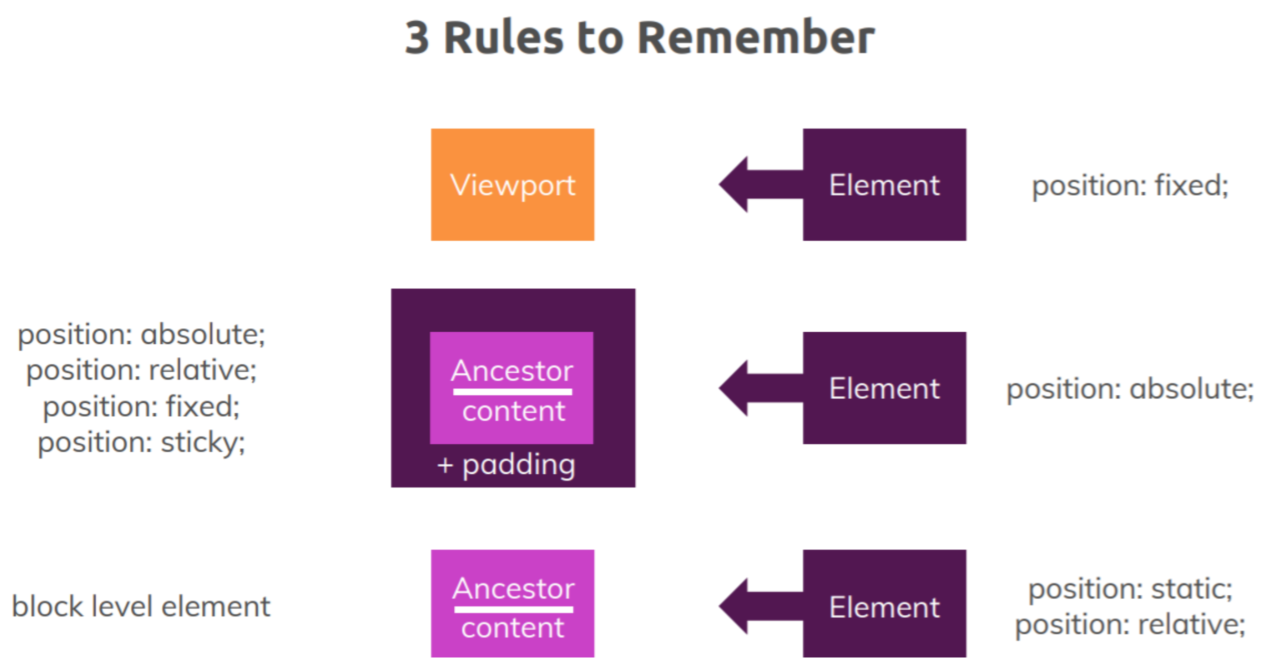
Art is a diverse collection of human activities in creating visual, auditory or performing artifacts (artworks), expressing the imaginative, conceptual ideas or technical skills of the performer, intended to be appreciated for their beauty or emotional power.

For a mobile screen,



Art is a diverse collection of human activities in creating visual, auditory or performing artifacts (artworks), expressing the imaginative, conceptual ideas or technical skills of the performer, intended to be appreciated for their beauty or emotional power.

14.2.2 percent (%)



14.2.3 rem

This unit styles the element relative to the font-size of the root element. Except that, it works exactly the way how em units work.

the font-size of the content in box is 2rem i.e.,

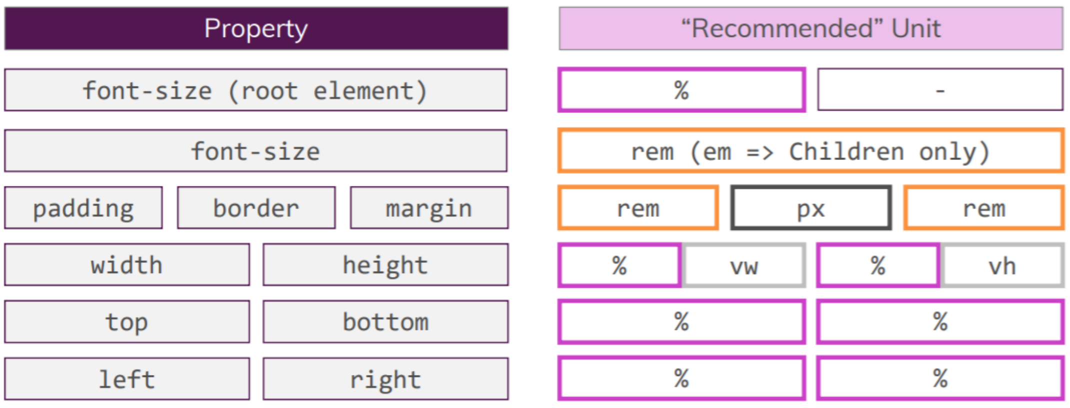
font-size = 2*root font-size

14.2.4 em

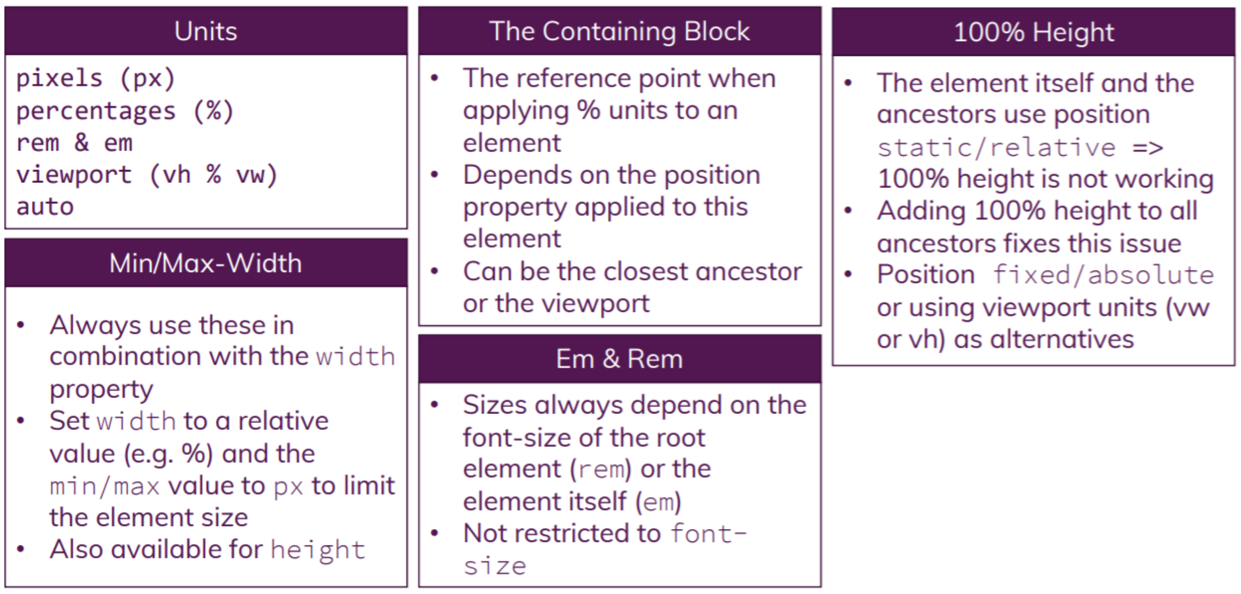
This unit styles the element relative to the font-size of the parent element when typographical properties are used. Else, it styles the element relative to the font-size of itself when other properties are used.

font-size of the content = font-size in the parent element * font-size in the emUnit class

14.2.5 Recommended Units



Property			"Recommended" Unit		
font-size (root element)			%	-	
font-size			rem (em => Children only)		
padding	border	margin	rem	px	rem
width	height		% / vw	% / vh	
top	bottom		%	%	
left	right		%	%	



Units

```
pixels (px)
percentages (%)
rem & em
viewport (vh % vw)
auto
```

Min/Max-Width

- Always use these in combination with the `width` property
- Set `width` to a relative value (e.g. %) and the `min/max` value to `px` to limit the element size
- Also available for `height`

The Containing Block

- The reference point when applying % units to an element
- Depends on the position property applied to this element
- Can be the closest ancestor or the viewport

Em & Rem

- Sizes always depend on the font-size of the root element (`rem`) or the element itself (`em`)
- Not restricted to `font-size`

100% Height

- The element itself and the ancestors use position `static/relative =>` 100% height is not working
- Adding 100% height to all ancestors fixes this issue
- Position `fixed/absolute` or using viewport units (vw or vh) as alternatives

15. Responsive Web Design

A Responsive page is a web page that renders well in all the devices like desktop, laptop, tablet, and mobiles, which have different screen sizes.

We can make a web page's layout change according to the screen size using the below techniques

Flexible Images

The image grows or shrinks according to the container dimensions. We need to specify the width of the image using the percentage of its container width.

Setting the width of foreground images

We will apply relative dimensions to the <img> element
Problem: Consider we have an image that needs to be displayed at 150X150 size inside a 720 px wide element and we want it to scale it as the parent element scales.
Solution: For the above problem, we can have a container exclusively for the image, set its relative size on the container, and let the image fill container width.
i.e., 150/720 = 20.8333%
and to scale the image when the parent scales but limit the extent to which it scales up/down we can set max-width/min-width on the parent.

Setting the width of the background images

Problem: Consider we want the background image to scale when the element for which background is set to scales.
Solution: For the above problem, we can set background-size: 100%

Fluid Layouts

Layout grows and shrinks according to the viewport dimensions. We need to use relative units such as % instead of absolute units (px).
In order to fit the layout to viewport width, we will replace absolute units for dimensions of elements(px) with relative units (%)
The width of the topmost element (<html>) is browser width (due to viewport tag)
Thus specifying the width of all elements using % will scale the layout to the viewport width. Such a layout is called fluid layout
We need to represent all px widths as % of the corresponding parent width.

Media queries

Media queries apply CSS styling based on the device form factors. Hence the layout can differ on a desktop, tablet, and mobile.

```
@media screen and (max-width: 520px) {
  .body {
    background: none;
  }
}
```

This media query hides a background image for the element with the class body when the page is viewed on a screen and browser viewport width is lesser than 520px.

Media Types

The important media types are

Media Type	Description
all	Applies to all devices (default value)
handheld	Applies to handheld devices
print	Applies to printer
screen	Applies to any type of screen
tv	Applies to TV screens

Media Features

The important media types are

Feature	Values	min/max	Description
device-height, device-width	number	Yes	Height/Width of the output device (specified in px, em, etc.)
height, width	number	Yes	Height/Width of the viewport
orientation	landscape | portrait	No	The orientation of the device
resolution	number	Yes	Resolution(pixel density) of the device (in dpi, dpcm, etc.)
device	number	Yes	The number of device pixels per CSS pixel. Used to detect high-resolution screens like iPad retina

Viewport meta tag

It is observed that sometimes mobile devices display a full desktop site. In such cases, the computed page width is much higher than the expected and hence the appropriate media queries will not be applied. It also results in pages being zoomed out on mobile devices.
To overcome the above problem we can use the viewport meta tag

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scale=0">
```

When we set width=device-width the page will be as wide as the device which prevents default zooming out actions on some mobiles
When we set initial-scale=1.0 the page is unscaled initially and cannot be further scaled by the user
The viewport is the area on a browser where the page is rendered.

Viewport width > Browser width > page is wider than browser

16. Font Properties

The font-family CSS property specifies a prioritized list of one or more font family names and/or generic family names for the selected element.

16.1 font-family

The font-family CSS property specifies a prioritized list of one or more font family names and/or generic family names for the selected element.

Values are separated by commas to indicate that they are alternatives. The browser will select the first font in the list that is installed or that can be downloaded using a @font-face at-rule.

It is often convenient to use the shorthand property font to set font-size and other font related properties all at once.

You should always include at least one generic family name in a font-family list, since there's no guarantee that any given font is available. This lets the browser select an acceptable fallback font when necessary.

The font-family property specifies a list of fonts, from highest priority to lowest. Font selection does not stop at the first font in the list that is on the user's system. Rather, font selection is done one character at a time, so that if an available font does not have a glyph for a needed character, the latter fonts are tried. When a font is only available in some styles, variants, or sizes, those properties may also influence which font family is chosen.

font-family: "specific family", generic family;

font-family: "comic sans ms", Arial;

16.2 font-style

Specifies the style of the font. Values can be normal (default), italic, oblique

font-style: italic;

16.3 font-variant

Specifies the variant of the font. Values can be normal (default) and small-caps

font-variant: small-caps;

16.4 font-size

Specifies the size of the font. Values can be x-small, small, medium, large, and can be a specific number with pixels unit.

font-size: x-small;
font-size: 20px;

16.5 font-weight

Specifies the weight of the font. Values can be normal, bold, bolder, light, x-large.

font-weight: bold;

16.6 line-height

Specifies the height of the line.

line-height: 10px;
line-height: 1cm;
line-height: 0.5;

When line-height is specified without unit, then the number will be multiplied with font-size.

16.7 custom font

We can specify a custom font using the @font-face rule
Web fonts allow developers/designers to include fonts that are not installed on the user's computer.
The @font-face rule helps in loading custom fonts on a webpage. It instructs the browser to download the custom font specified in the style sheet from where it is hosted and display it.
We can create a custom font file using online tools or download the available fonts from the internet, and store it on the server. Whenever needed the font file will be downloaded.
Assign a name to our custom font and specify the source of our font file.

```
@font-face {
  font-family: name;
  src: url(“pathofthecustomfontfile.”);
}
```

16.8 font

The font CSS shorthand property sets all the different properties of an element's font. Alternatively, it sets an element's font to a system font.

font: font-style font-variant font-weight font-size/line-height font-family;

17. Text Properties

17.1 text-align

Specifies the horizontal alignment of the text.

Values can be left, right, center, justify.

text-align: right;

17.2 text-decoration

Specifies whether the text should be underlined or overlined.

Values can be none (default), underline, overline, line-through

text-decoration: underline;

17.3 text-transform

Specifies whether the text should be rendered in uppercase, lowercase, or camelcase. Values can be capitalized, uppercase, lowercase.

text-transform: capitalize;

17.4 text-indent

The text-indent CSS property sets the length of empty space (indentation) that is put before lines of text in a block.

text-indent: 90px;

17.5 letter-spacing

Specifies the spacing between the letters.

letter-spacing: 90px;

17.6 word-spacing

Specifies the spacing between the words.

word-spacing: 90px;

17.7 text-shadow

The text-shadow property adds a shadow to the text of any HTML element

text-shadow: 10px 6px 2px black;

The first value specifies the horizontal offset. Positive value creates a shadow to the right. Negative value creates the shadow to the left.

The second value specifies the vertical offset. Positive value creates a shadow to the left. Negative value creates the shadow to the right.

The third value specifies the blur distance. An optional value that decides the percentage of blur effect and takes a positive value.

The fourth value specifies the shadow color.

18. Flexbox

Flexbox is one of the most widely used CSS properties that help us to align elements in the given available space of the HTML document.

It is a one-dimensional layout structure that has good and strong capabilities of aligning items dynamically within a given container by maintaining proper spaces between each item under that container. Since it supports dynamic change in appearance, it is named 'flex'.

It can accommodate both expansion of items and shrinkage of items according to the given situation in the given available space.

18.1 Flex Container

A flexbox requires a container within which all elements can be aligned. Hence, a flex container will have the ability to alter all the elements that comes within that container but not any other element. (display: flex)

Properties of a flex container:

18.1.1 flex-direction

The **flex-direction** CSS property sets how flex items are placed in the flex container defining the main axis and the direction (normal or reversed).

The following values are accepted:

row

The flex container's main-axis is defined to be the same as the text direction. The main-start and main-end points are the same as the content direction.

row-reverse

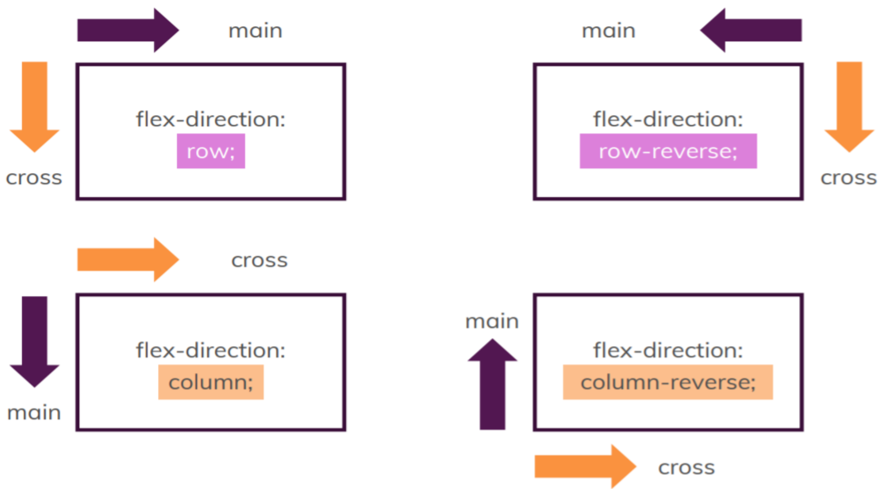
Behaves the same as row but the main-start and main-end points are opposite to the content direction.

column

The flex container's main-axis is the same as the block-axis. The main-start and main-end points are the same as the before and after points of the writing-mode.

column-reverse

Behaves the same as column but the main-start and main-end are opposite to the content direction.



18.1.2 flex-wrap

The flex-wrap CSS property sets whether flex items are forced onto one line or can wrap onto multiple lines. If wrapping is allowed, it sets the direction that lines are stacked.

The following values are accepted:

nowrap

The flex items are laid out in a single line which may cause the flex container to overflow. The cross-start is either equivalent to start or before depending on the flex-direction value. This is the default value.

wrap

The flex items break into multiple lines. The cross-start is either equivalent to start or before depending flex-direction value and the cross-end is the opposite of the specified cross-start.

wrap-reverse

Behaves the same as wrap but cross-start and cross-end are permuted.

18.1.3 flex-flow

The flex-flow CSS shorthand property specifies the direction of a flex container, as well as its wrapping behavior.

flex-flow: flex-direction flex-wrap

18.1.4 align-items

The CSS align-items property sets the align-self value on all direct children as a group. In Flexbox, it controls the alignment of items on the Cross Axis.

Values:

stretch

Flex items are stretched such that the cross-size of the item's margin box is the same as the line while respecting width and height constraints.

flex-start

The cross-start margin edges of the flex items are flushed with the cross-start edge of the line.

flex-end

The cross-end margin edges of the flex items are flushed with the cross-end edge of the line.

center

The flex items' margin boxes are centered within the line on the cross-axis. If the cross-size of an item is larger than the flex container, it will overflow equally in both directions.

18.1.5 justify-content

The CSS justify-content property defines how the browser distributes space between and around content items along the main-axis of a flex container.

flex-start

The items are packed flush to each other toward the edge of the alignment container depending on the flex container's main-start side. This only applies to flex layout items. For items that are not children of a flex container, this value is treated like start.

flex-end

The items are packed flush to each other toward the edge of the alignment container depending on the flex container's main-end side. This only applies to flex layout items. For items that are not children of a flex container, this value is treated like end.

center

The items are packed flush to each other toward the center of the alignment container along the main axis.

space-between

The items are evenly distributed within the alignment container along the main axis. The spacing between each pair of adjacent items is the same. The first item is flush with the main-start edge, and the last item is flush with the main-end edge.

space-around

The items are evenly distributed within the alignment container along the main axis. The spacing between each pair of adjacent items is the same. The empty space before the first and after the last item equals half of the space between each pair of adjacent items.

18.1.6 align-content

The CSS align-content property sets the distribution of space between and around content items along a flexbox's cross-axis.

flex-start

The items are packed flush to each other against the edge of the alignment container depending on the flex container's cross-start side. This only applies to flex layout items. For items that are not children of a flex container, this value is treated like start.

flex-end

The items are packed flush to each other against the edge of the alignment container depending on the flex container's cross-end side. This only applies to flex layout items. For items that are not children of a flex container, this value is treated like end.

center

The items are packed flush to each other in the center of the alignment container along the cross axis.

space-between

The items are evenly distributed within the alignment container along the cross axis. The spacing between each pair of adjacent items is the same. The first item is flush with the start edge of the alignment container in the cross axis, and the last item is flush with the end edge of the alignment container in the cross axis.

space-around

The items are evenly distributed within the alignment container along the cross axis. The spacing between each pair of adjacent items is the same. The empty space before the first and after the last item equals half of the space between each pair of adjacent items.

18.2 Flex Items

Children of flex container are flex items.

Properties of a flex item:

18.2.1 order

The order CSS property sets the order to lay out an item in a flex container. Items in a container are sorted by ascending order value and then by their source code order.

Takes integer value.

18.2.2 align-self

The align-self CSS property overrides a flex item's align-items value. In Flexbox, it aligns the item on the cross axis.

18.2.3 flex-grow

The flex-grow CSS property sets the flex grow factor of a flex item's main size

Negative values are invalid. Defaults to 0. (takes a whole number)

This property specifies how much of the remaining space in the flex container should be assigned to the item (the flex grow factor).

18.2.4 flex-shrink

The flex-shrink CSS property sets the flex shrink factor of a flex item. If the size of all flex items is larger than the flex container, items shrink to fit according to flex-shrink.

Negative values are invalid. Defaults to 1. (takes a whole number)

18.2.5 flex-basis

The flex-basis CSS property sets the initial main size of a flex item.

18.2.6 flex

The flex CSS shorthand property sets how a flex item will grow or shrink to fit the space available in its flex container.

The flex property may be specified using one, two, or three values.

One-value syntax: the value must be one of:

a <number>: In this case it is interpreted as flex: <number> 1 0; the <flex-shrink> value is assumed to be 1 and the <flex-basis> value is assumed to be 0.

one of the keywords: none, auto, or initial.

Two-value syntax: The first value must be:

a <number> and it is interpreted as <flex-grow>.

The second value must be one of:

a <number>: then it is interpreted as <flex-shrink>.

a valid value for width: then it is interpreted as <flex-basis>.

Three-value syntax: the values must be in the following order:

a <number> for <flex-grow>.

a <number> for <flex-shrink>.

a valid value for width for <flex-basis>.

19. Grid

CSS Grid is the latest way to create layouts. CSS Grid allows to create rows and columns.(display: grid)

19.1 grid-template-rows

The grid-template-rows CSS property defines the line names and track sizing functions of the grid rows.

grid-template-rows: [linename1] measure [linename2] measure

19.2 grid-template-columns

The grid-template-columns CSS property defines the line names and track sizing functions of the grid columns.

grid-template-columns: [linename1] measure [linename2] measure

19.3 grid-template-areas

The grid-template-areas CSS property specifies named grid areas, establishing the cells in the grid and assigning them names.

```
grid-template-areas:
        "b b a"
        "b b c"
        "b b c";
```

19.4 grid-row-start

The grid-row-start CSS property specifies a grid item’s start position within the grid row by contributing a line, a span, or nothing (automatic) to its grid placement, thereby specifying the inline-start edge of its grid area.

19.5 grid-row-end

The grid-row-end CSS property specifies a grid item’s end position within the grid row by contributing a line, a span, or nothing (automatic) to its grid placement, thereby specifying the inline-end edge of its grid area.

19.6 grid-row

The grid-row CSS shorthand property specifies a grid item’s size and location within the grid row by contributing a line, a span, or nothing (automatic) to its grid placement, thereby specifying the inline-start and inline-end edge of its grid area.

grid-row: grid-row-start/grid-row-end

19.7 grid-column-start

The grid-column-start CSS property specifies a grid item’s start position within the grid column by contributing a line, a span, or nothing (automatic) to its grid placement. This start position defines the block-start edge of the grid area.

19.8 grid-column-end

The grid-column-end CSS property specifies a grid item’s end position within the grid column by contributing a line, a span, or nothing (automatic) to its grid placement, thereby specifying the block-end edge of its grid area.

19.9 grid-column

The grid-column CSS shorthand property specifies a grid item's size and location within a grid column by contributing a line, a span, or nothing (automatic) to its grid placement, thereby specifying the inline-start and inline-end edge of its grid area.

grid-row: grid-column-start/grid-column-end

19.10 grid-row-gap

The row-gap CSS property sets the size of the gap (gutter) between an element's grid rows.

19.11 grid-column-gap

The column-gap CSS property sets the size of the gap (gutter) between an element's columns.

19.12 grid-gap

The gap CSS property sets the gaps (gutters) between rows and columns. It is a shorthand for row-gap and column-gap.

20. Transformation

A transformation is an effect that lets an element to change its shape, size, and position.

Using the CSS property ‘transform’ we can move, tilt, scale any element of our web page.

The following are the few CSS transform functions:

20.1 translate

CSS3 translate() function is used to move an element from one position to another position.

An element can be moved to the right, left, top, bottom.

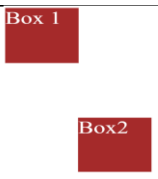
Syntax: translate (x,y) -> moves the element to right by x-value, and y-value down from its current position.

An element can be moved in only one direction i.e. horizontal/vertical using translateX and translateY functions.

Example:

translate(20px, 30px) -> moves the element to right by 20px and 30px down from the current position.
translate(-40px, 20px) -> moves the element to left by 40px and 20px down from the current position.
translate(-30px, -25px) -> moves the element to left by 30px and 25px up from the current position.

```
<html>
<head>
   <title> Transformation - Translate</title>
   <style>
      .box{
         height: 50px;
         width: 50px;
         background-color: brown;
         color: white;
      }
      #move{
         position: relative;
         transform: translate(50px, 50px); /* Moves the element right and down by 50px */
      }
   </style>
</head>
<body>
   <div class='box'>Box 1</div>
   <div class='box' id='move'>Box2</div>
</body>
</html>
```



20.2 scale

CSS3 scale() function is used to change(increase/decrease) the size of any element.

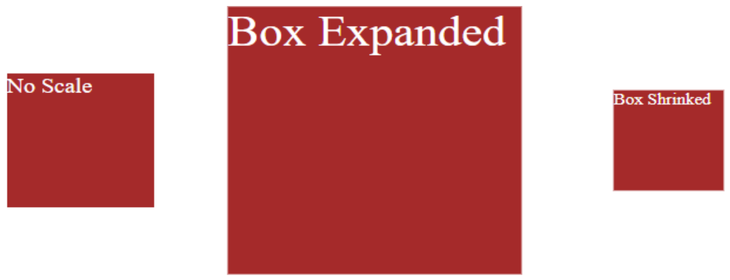
Syntax: scale(x,y) -> scales the element width(x-value) & height(y value)

If only one value is supplied to the scale(value) function, then that value will be considered as both x & y parameters

Example:

scale(2,1) -> The width of the element will be doubled and height remains the same.
scale(0.5) -> will decrease the element size by half
scale(1)-> element size will remain same
scale(2)-> doubles the element size

```
<html>
<head>
    <title>Transformation - Scale</title>
    <style>
        .box{
            float: left;
            margin: 50px;
            height: 100px;
            width: 100px;
            background-color: brown;
            color: white;
        }
        #expand{
            transform: scale(2);
        }
        #shrik{
            transform: scale(0.75);
        }
    </style>
</head>
<body>
    <div class='box'>No Scale </div>
    <div class='box' id='expand'>Box Expanded</div>
    <div class='box' id='shrik'>Box Shrinked</div>
</body>
</html>
```



20.3 rotate

CSS3 rotate() function is used to rotate an element from its origin point up to a specified angle. The angle is specified in degrees.

Syntax: rotate(x) -> Rotates the element.

If the angle specified is a positive value then the element rotates clockwise. if the angle specified is a negative value then the element rotates anti-clockwise.

Example:

rotate(20deg) -> Rotates the element by 20 degrees in clockwise direction.
rotate(-50deg) -> Rotates the element by 50 degrees in anti-clockwise direction.

20.4 skew

CSS3 skew() function is used to turn/skew any element to any angle.

The turning/skewing is dependent on the parameters specified for the vertical & horizontal axis.

Syntax: skew(x,y) -> skews the element from x degrees from the x-axis & y- degrees from the y-axis.

if the y-axis value is not specified then it has a zero value.

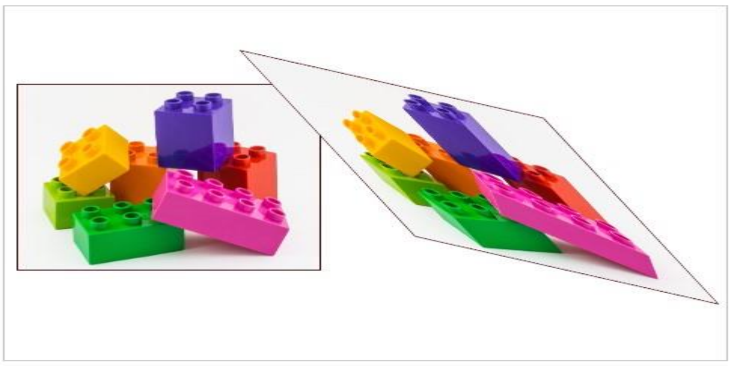
To skew an element across a single axis, we can use skewX() and skewY() functions.

Example:

skew(20deg) -> skew an element by 20 degrees across x-axis.
skew(30deg, 10deg) -> skews an element by 30 degrees across x-axis and 10 degrees across y-axis.
skewX(15deg) -> skews an element by 15 degrees across x-axis.
skewY(45deg) -> skews an element by 45 degrees across y-axis.

```
<html>
<head>
   <title>Rotation Demo</title>
         <link rel="stylesheet" type="text/css" href="skew.css">
</head>
<body>
<br />
<img src="images/Toy.jpg" alt="Toy"></img>
<img id="skew" src="images/Toy.jpg" alt="ToySkew"></img>
</body>
</html>
```

```
img
{
width:200px;
height:200px;
border:1px solid #440000;
}
img#skew
{
width:200px;
height:200px;
border:1px solid #440000;
margin-top:70px;
transform:skew(30deg,20deg);
}
```



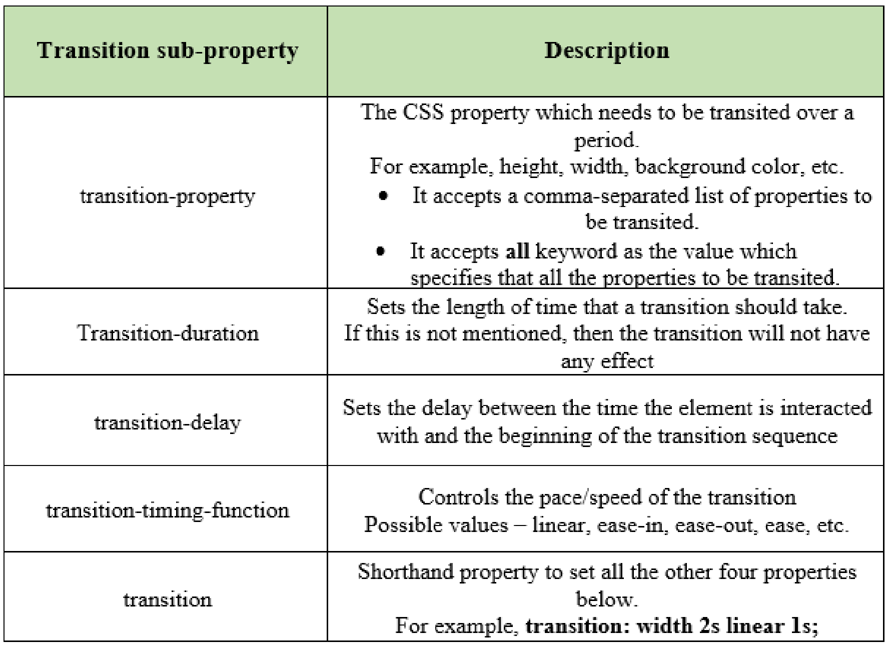
21. Transition

Transitions are gradual changes in the properties of an element when it changes from one style to another.

Using CSS3 transition effects can be added to elements without the need for any scripting or additional software like Flash.

To add a transition, two things must be specified:

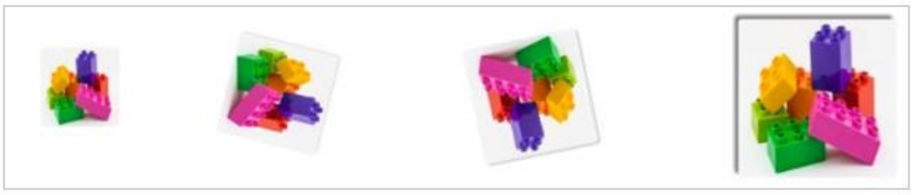
- The CSS property to which the effect should be added.
- The duration of the effect.



Transition sub-property	**Description**
transition-property	The CSS property which needs to be transited over a period. For example, height, width, background color, etc. • It accepts a comma-separated list of properties to be transited. • It accepts **all** keyword as the value which specifies that all the properties to be transited.
Transition-duration	Sets the length of time that a transition should take. If this is not mentioned, then the transition will not have any effect
transition-delay	Sets the delay between the time the element is interacted with and the beginning of the transition sequence
transition-timing-function	Controls the pace/speed of the transition Possible values – linear, ease-in, ease-out, ease, etc.
transition	Shorthand property to set all the other four properties below. For example, **transition: width 2s linear 1s;**

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="transition.css">
</head>
<body>
<img src="images/Toy.jpg" alt="ToyImage"></img>
</body>
</html>
```

```
img{
width: 100px;
height: 100px;
border-radius: 5px;
margin-left: 50px;
margin-top:50px;
transition: 3s linear;
}
img:hover {
width: 200px;
height: 200px;
margin-left: 400px;
transform:rotate(360deg);
box-shadow: -5px -5px 5px #888;
}
```



22. Animations

CSS3 Animations can be used to replace animated images, scripts, and flash animations in most of the web pages.

- Animations in CSS are done using key-frames.
- The @keyframes style rule controls the intermediate steps in a CSS animation sequence.
- Within each key-frame, we define styles for the start and the end of smooth animation. We can also define styles for keyframes in the intermediate steps along the animation sequence.
- We can specify when the change will happen using from and to keywords or give values in percentages. 0% marks the beginning of the animation and 100% is when the animation is complete. This gives more control over the intermediate steps of the animation sequence.

Animation is a short-hand property to specify the properties mentioned below

animation property	**Description**
animation-name	Every animation is given a name using which it is attached to an element.
animation-duration	The time span for an animation to complete one iteration.
animation-timing	Controls the pace/ speed of the animation. The key terms used are linear, ease, ease-in, ease-out, ease-in-out.
animation-delay	Specifies the delay before starting the animation.
animation-iteration-count	Specifies the number of iterations the animation will play. The Default value is 1.
animation-direction	The default value is normal which plays the animation in the forward direction. Takes the value reverse which plays the animation in backward direction Takes the value alternate which plays the animation in the forward direction every odd time and backward direction even time Takes value alternate-reverse which plays the animation in reverse direction every odd time and forward direction even time
play-state	Explains the state of animation. Can take values ‘paused’ or ‘running’. Paused will pause the animation. Running will resume the animation from where it was paused.
fill-mode	The property is used to describe what occurs before the animations start & after the animation ends.

fill-mode property	**Description**
none	Default value. The properties remain in their original until the animation starts and once the animation ends the properties return to their original state.
forwards	Once the animation ends (determined by animation-iteration-count), the properties retain the final keyframe values.
backwards	The animation will apply the property values defined in the keyframe that will start the first iteration of the animation, during the period defined by animation-delay. These are either the values of the from keyframe (when animation-direction is "normal" or "alternate") or those of the to keyframe (when animation-direction is "reverse" or "alternate-reverse")
both	The animation will follow the rules of both forwards and backwards properties. That is, it will extend the animation properties in both directions

```
/* @keyframes duration | easing-function | delay |
iteration-count | direction | fill-mode | play-state | name */
animation: 3s ease-in 1s 2 reverse both paused slidein;

/* @keyframes name | duration | easing-function | delay */
animation: slidein 3s linear 1s;

/* @keyframes name | duration */
animation: slidein 3s;
```

23. CSS variables

CSS variables are custom properties that can be used to change the appearance of the webpage.

The process of using including CSS Variables includes two steps. They are:

23.1 Declaring the variable

While declaring a variable, the custom property name must start with -- (2 hyphens) followed by the property name. A variable can be declared as shown below:

--text-color: brown;

23.2 Using the declared variable

The variable which is declared can be used with a function called var. This function is helpful in inserting the value of a custom property instead of a pre- defined value.

The variable can be assigned to any css property that accepts the declared value.

For example, if the variable declared is

  --text-color: brown;

then, the variable can be used for any CSS property that accepts color; i.e., color, background-color, color in border property and so on.

Note: var function can take two parameters

The first parameter is a mandatory parameter i.e., the declared variable. The second parameter is an optional parameter. This parameter is used when the declared variable isn't set or an invalid value is used for the first parameter.

 color: var(--text-color,red);

24. SASS/SCSS

Problems faced by developers while implementing new requirements:

- Maintainability - Maintaining the existing CSS code was getting tedious as the code was poorly organized, and had separate styles for each requirement
- Reusability - Existing CSS code wasn't modular and they were not able to reuse much of it

Cons of CSS:

- Inheritance, function building, definition reuse is difficult in CSS.
- Maintenance is considered as major problem when managing the large or complex project systems.

Increased expectations from CSS :

- In the evolving web environment, HTML and CSS are introduced with new specifications very often.
- These specifications are applied by browsers when they are in state of proposal along with their vendor prefixes. Writing the code with vendor specifications is an issue as all different versions from vendor to be added to achieve a single result.

CSS on its own can be fun, but stylesheets are getting larger, more complex, and harder to maintain.
With their advanced features, pre-processors help in achieving to write CSS codes that can be reused, easily maintainable and inheritable code. Sass provides you features such as variables, mixins, inheritance, nested rules ,built in functions and other different stuff that are difficult to do with CSS
With CSS pre-processing, we can easily increase our productivity, and decrease the amount of CSS code we are writing in a project.
There are various pre-processors available, each with their own pros and cons. Most of them have similar features, more or less. Developers can pick one according to their coding familiarity and start using it in their next project.

Syntactically Awesome Style Sheets(SASS) is considered as a CSS preprocessor and is a stylesheet language .

- Hampton Catlin is the author of SASS and Natalie Weizenbaum established SASS in 2007
- It is compatible with all versions of CSS
- It saves time by reducing repetition of CSS
- It help us to keep large stylesheet well-organized
- It will be compiled into CSS code
- It is open source and is therefore free to download and use

Two SASS syntaxes

- SCSS: SCSS or 'Sassy CSS' is the most commonly used syntax and is a superset of CSS3's syntax i.e., every valid CSS3 stylesheet is a valid SCSS stylesheet. This uses curly braces and semicolons similar to that of CSS.

- SASS: SASS is the older version known as the indented syntax. It was inspired by the neat conciseness of the language, Haml. It uses indentations of line instead of semicolons and brackets to specify blocks.

24.1 How to use SASS?

Step 1: Install SASS globally in your application using online store i.e. npm

```
npm install -g sass
```

The above command will install Sass in your machine,

Now run below command which returns the version of Sass if installed correctly

```
sass –version
```

Step 2: Transpiling Sass code into CSS code:

To transpile the code we need to give below command:

sass <<sass_input_file_name.scss>> <<css_output_file_name.css>>

24.2 SASS variables

Sass provides us the concepts of variables where we can store information like color,font-style,background-color or any CSS properties that we can think off or want to reuse in our application later.

Syntax for defining Sass variables:

```
$variablename : value;
```

```
// Creating variables and provding values to it
$bgColor:red;
$textDecoration: underline;
$fontColor:white;
// Using the variables for Web page
body{
   background-color: $bgColor;
   color:$fontColor;
   text-decoration: $textDecoration;
}
```

Equivalent CSS Code:

```
body {
  background-color: red;
  color: white;
  text-decoration: underline;
}
```

24.3 Nesting

```
.navbar{
   background-color: rgb(87, 156, 156);
   ul{
      display: inline;
      margin: 0;
      padding: 0;
      list-style: none;
   }
   li{
      display: inline-block;
      a{
         display: block;
         padding: 6px 12px;
         text-decoration: none;
         color: whitesmoke
      }
      a:hover{
         color:gold;
      }
   }
}
```

Please note that li and ul selectors are inside under navbar while a is deep nested and is inside li tag
Equivalent compiled CSS code:
Rules in css are not nested, hence they are defined one by one:

```
.navbar {
  background-color: #579c9c;
}
.navbar ul {
  display: inline;
  margin: 0;
  padding: 0;
  list-style: none;
}
.navbar li {
  display: inline-block;
}
.navbar li a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
  color: whitesmoke;
}
.navbar li a:hover {
  color: gold;
}
```