CSS

- CSS is the language we use to style and align our HTML elements.
- CSS was introduced in 1994 by Haykon wium lie.
- First version of CSS was introduced in 1996(CSS1)
- CSS2-1998
- Current version of CSS is CSS3-1999
- CSS stands for cascading style sheet. CSS is optional but it converts an off looking HTML page into a beautiful & responsive website.

Syntax of CSS

- **CSS syntax:-**

```
Selector{
    Property: value ;
}

P{
    color: blue
}
```

Types of CSS

- It is style sheet language which is used to describe the look and formatting of documents written in markup language.
- It is generally used with html to change the style of web pages and UI.
- Commenting in CSS :-- /*-----*/

Three ways to add CSS file

1. **Inline style sheet : Style attribute**

```
<p style : ” color: red”>Hello CSS</p>
```

2. **Internal style sheet : <style >tag**

Example:

```
<style>
    p{
        color : value
        }
</style>
```

External CSS Style

- External style sheet==<link>tag
- External CSS is used to apply CSS on multiple pages .

- Extension must be .CSS for CSS files.

Creating First CSS website

What is DOM ?

DOM stands for document object model. When a page is loaded, the browser creates a DOM of the page which is constructed as a tree of objects.

Selectors

A CSS selector selects the HTML element(s) you want to style.

```
Selector{
    property: value;
    property: value;
}
```

Types of CSS Selectors

- Simple Selector
- Combinator Selector
- Attribute Selector
- Pseudo class Selector
- Pseudo Element Selector

Simple Selector

- Id selector (#)
- Class selector (.)
- Universal Selector (*)
- Element Selector (tag)
- Grouping Selector(,)

1. **Id selector (#)**
   - Unique id attribute within the page is selected
   - Core attribute selector
   - Selected by symbol "#" followed by id name
   - SYNTAX:

```
 #id_name{

    CSS  properties
}
```

2. **Class selector (.)**
   - Core attribute selector ,with specific class.
   - The character to access in CSS file " . "
   - Multiple class name can be called using comma
   - SYNTAX:

```
.class_name{
    /* css  properties*/
}
```

- NOTE: PRIORITY ORDER =====

  ID > CLASS > TYPE/ELEMENT >UNIVERSAL

3. **Universal Selector (*)**
   - Select any and all types of elements in html page.
   - All the elements within a body tag.
   - Symbol of selector: “ * “
   - SYNTAX:

```
*{
    properties
}
```

4. **Element Selector (tag)**
- Selects particular element.
- Call by type of tag.
  - SYNTAX:

```

element name{
    properties
}
```

Combinator Selector

A combinator is something that explains the relationship between the selectors.

<u>Types:</u>

- Descendent selector(space)
- Child selector(>)
- Adjacent sibling selector(+)
- General sibling selector(~)

1. **Descendent selector(space)**

   - The descendant selector matches all elements that are descendants of a specified element.
   - (parent, parent’s parent, parent’s parents’ parent)
   - Syntax:

```
Selector1 Selector2{
        property : declaration
        }
```

```
Div p{
        prop: Val;
    }
```

2. **Child selector(>)**

- The child selector selects all the elements that are the children of a specified element
- It is placed between two CSS selectors, matches only those elements matched by second selector and direct child.
- **Syntax:**

  selector 1 > selector 2 { properties }
- **Ex:**

```
Div>p{
        prop : val
    }
```

3. **Adjacent sibling selector(+)**
- The adjacent sibling selector is used to select an element that is directly after another specific element.
- Sibling elements must have the same parent element, and "adjacent" means "immediately following".
- Syntax :

```
    former_element + target_element {
        style : properties
    }
```

4. **General sibling selector(~)**
- The general sibling selector selects all elements that are next sibling of a specified element.
- The general sibling combinator (~) separates two selectors.

- **Syntax :**

```
    former_element ~ target_element {
        style : properties
    }
```

Attribute Selector

- Attribute provides extra information to the tag.
- In this attribute selector we are targeting the elements based on attributes

```
Selector[attribute]{
Property: value
}
```

- EX:

- [attr=value]:Represents element with an attribute name of attr whose value is exactly value.

Pseudo classes

A CSS pseudo-class is a keyword added to a selector that specify a special state of the selected element.

For Example: It can be used to

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

```
    Selector: pseudo-class{
    property: value;
}
```

- Dynamic pseudo classes
    - **Anchor Pseudo-classes**
        - Link
        - Visited
        - Active
        - Focus
        - Hover
    - Structural pseudo classes
        - First-child
        - Last-child
        - Nth-child()

Pseudo Class Selector

- a:hover MUST come after we mouse hover on it
- a:link and a:visited in the CSS definition in order to be effective.
- a:active MUST come after

- The :first-child pseudo-class matches a specified element that is the first child of another element.
- The :last-child pseudo-class matches a specified element that is the last child of another element.
- The :nth-child(*n*) selector matches every element that is the *n*th child of its parent.
- *n* can be a number, a keyword (odd or even), or a formula (like *an* + *b*).

Pseudo Elements

- A CSS pseudo-element is used to style specified parts of an element.
- For example, it can be used to:
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element

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

:: first-line

:: first-letter

::before

::after

::marker

::selection

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::marker	::marker	Selects the markers of list items
::selection	p::selection	Selects the portion of an element that is selected by a user

Pseudo class v/s pseudo element

PSEUDO CLASS	**PSEUDO ELEMENT**
selectors are selected by " : "	selectors are selected by " :: "
Pseudo-classes are used to target state.	Pseudo-elements are used to target specific parts of an element.

Text Property

- Text formatting
- Color
- Text-align
- Text-transform
- Text-shadow
- Text-decoration
- Letter-spacing
- Word-spacing
- Text-indention

Background Property

- Background-image : url (“image.jpg”)
- Background-repeat : no-repeat/repeat-x/Y
- Background-size : cover/100%
- Background-Position : right/left/center
- Background-attachment : scroll/fixed
- Background-color :

  The CSS background-color property specifies the background color of a container

Font Property

- Font-size: large/small/medium
- Font-weight: bold/bolder/lighter/normal
- Font-style: italic
- Font-family: font styles

Color Property

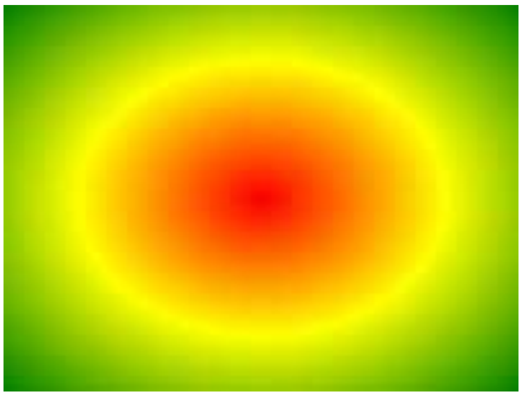
The CSS color property can be used to set text color inside an element.

Similarly, we can set color for different elements

Types of color values:

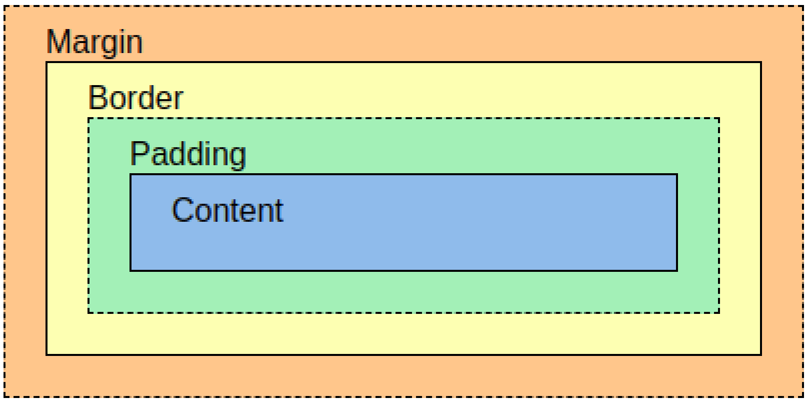
- **RGB :** Specify color using Red, Green , Blue values. EX. **rgb (255,255,0)**
- **HEX Code :** Specify color using hex code. EX.**Color: #efefef;**
- **HSL : Specify** The color using HSL values(Hue, Saturation, Lightness)

Gradient



- Linear-gradient(direction,color-stop1,color-stop2);
- Radial-gradient(shape size at position, start-color,…,last-color);

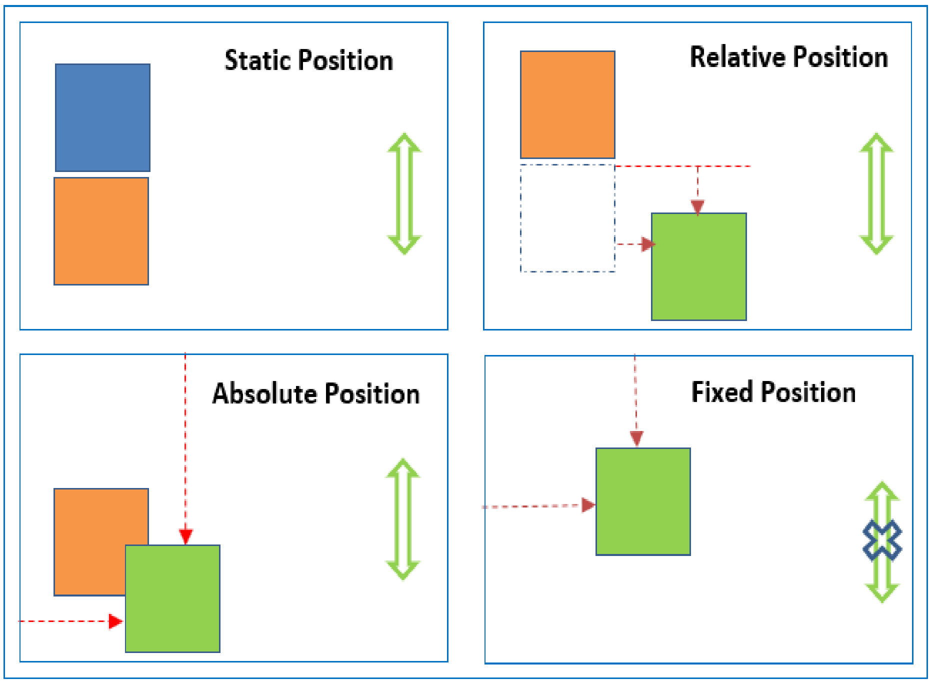
Box Model



- The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image illustrates the box model:
- **Content -** The content of the box, where text and images appear
- **Padding -** Clears an area around the content. The padding is transparent
- **Border -** A border that goes around the padding and content
- **Margin -** Clears an area outside the border. The margin is transparent

Properties

- Margin-style
  - Margin-top
  - Margin-bottom
  - Margin-left
  - Margin-right
- Padding-style
  - Padding-left
  - Padding-right
  - Padding-top
  - Padding-bottom
- Border-style
  - Border-color
  - Border-width
  - Border-radius



- Margin Collapse

  When two margin from different elements overlap, the equivalent margin is the greater of the two. This is called margin collapse.

- Box Sizing

  Determines what out of padding and border is included in elements width and height. It can be **content-box** (Include only content in width/height) or **border-box(** The content width and height include **content + padding + border**)

Font and Display

Display properties

The CSS display property is used to determine whether an element is treated as a block/ inline element & the layout sed for its children. (

- **Inline :** Displays an element as an inline element (like <span>). Any height and width properties will have no effect
- **block** : Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width
- **Inline-block :** Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values
- **display : none vs visibility : hidden** :

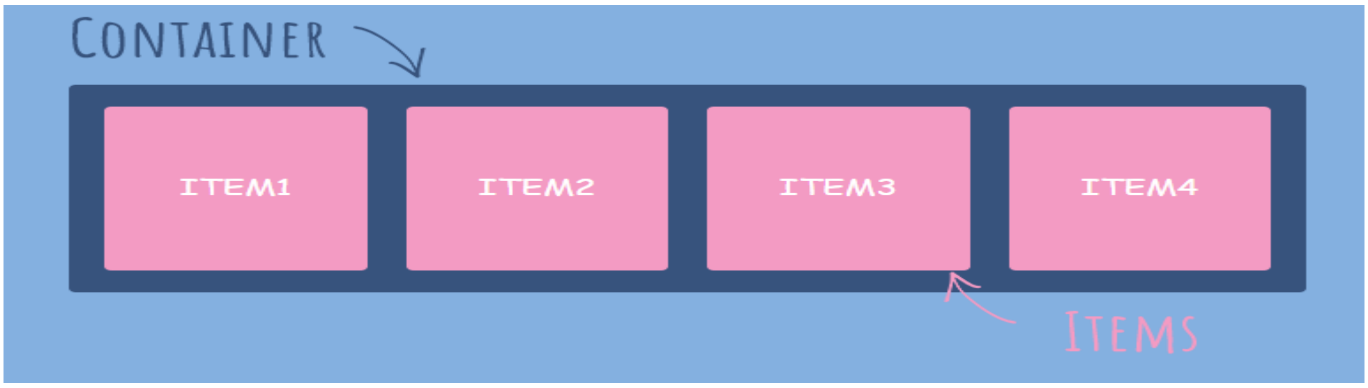
  With display : none, the element is removed from the document. Its space is not blocked.

  With visibility : hidden, the element is hidden but its space is reserved.
- **Flex :** Displays an element as a block-level flex container

Flex Properties

Flex-box properties

- **Display :** flex
- **Flex-direction:** row/column
- **Flex-wrap**: wrap/nowrap/wrap-reverse
- **Justify-content**: flex-start/flex-end/center/space-around/space-between/baseline
- **Align-items:** flex-start/flex-end/center



Flex-items properties

- Flex-basis:<length>
- Flex-grow: <number>

Size, Position and Lists

There are more units for describing size other than 'px'.

They are rem, cm, vw, vh, percentage, etc.

What's wrong with pixels ?

Pixels are relative to the viewing device. For a device with the size 1920 X 1080, 1 px is 1 unit of 1080/1920.

Relative length

These units are relative to the other length property. Following is some of the most commonly used relative length.

- **Em :** Unit relative to the parent font size.
- **Rem :** unit relative to the root font size (<html> tag)
- **Vw :** unit relative to 1 % viewport width.
- **Vh** : unit relative to 1 % of viewport height.
- **% :** unit relative to the parent element.

Min/max-height/width property

CSS has amin-height , max-height , min-width, max-width property.

If the content is smaller than the minimum height, minimum height will be applied.

Similar is the case for other related property.

The position property

Used to manipulate the location if an element.

Following are the possible values:

- **Static :** The default position.
- **Relative :** The top/bottom/ left/right/z-index will work. Otherwise, the element is in the flow of document like static
- **Absolute:** The element is removed from the flow and is relatively positioned to its first non-static ancestor – top/bottom will work.
- **Fixed :** Lust like absolute except the element is positioned relative to the browser window
- **Sticky:** The element is positioned based on user's scroll position.

List-style property

The list style property is a shorthand for type, position and image.

z-index property

The z-index property specifies the stack order of an element.

It defines which layer will be above which is case of overlapping elements.

Flexbox

Before we look into the SCC flexbox, we will look into the float and clear property.

The float property

Float property is simple. It first flows the element towards left/ right.

The Clear property

Used to clear the float. It specifies what element can float beside a given element.

The CSS flexbox

Aims at providing a better way to layout, align and distribute space among items in a container.

Flex-direction property

Define the direction forward which items are laid. Can be row, row-reverse, column, column-reverse.

Flex-property for parent(flex container)

Following are the properties for the flex parent

Flex-wrap :	Can be wrap, nowrap, wrap-reverse. Wrap items as needed with this property.
Justify-content:	defines alignment along main axis.
Align-items:	Define alignment along cross axis.
Align-content:	Align a flex container's lines when there is extra in the cross axis.

Flex-property for children(flex items)

Following are the properties for the flex children's.

Order:	controls the order in which the items appear in the flex container.
Align-self:	allows default alignment to be overridden for the individual flex items.
Flex-grow:	defines the ability for a flex item to grow.
Flex-shrink:	specifies how much a flex item will shrink relative to the rest of the flex items.

CSS Grid and media queries

A CSS grid can be initialized using :

```
container{

  display: grid;
}
```

All direct children automatically become grid items

The Grid-column-gap property

Used to adjust the space between the columns of a CSS grid.

The **Grid-row -gap property**

Used to adjust he space between the rows of a CSS grid

The Grid-gap property

Shorthand property for grid-row-gap and grid-column-gap

```
container{
    display: grid;
    grid-gap: 40px    100px;
}             row          column
```

Note: For a single value of grid-gap, both row and column gaps can be set in one value.

Following are the properties for grid container:

1. The grid-template-columns property can be used to specify the width of columns.

```
container{
    display: grid;
    grid-template-columns: 80px 120px auto;
}
```

2. The grid-template-rows property can be used to specify the height of each row.

```
container{
    display: grid;
    grid-template-rows: 70px 150px;
}
```

3. The justify-content property is used to align the whole grid inside the container.
4. The align-content property is used to vertically align the whole grid inside the container.

Following are the properties for grid item:

1. The grid-column property defines how many columns an item span.

```
    .grid-item{
        grid-column: 1/5;
    }
```

2. The grid-row property defines how many rows an item will span
3. We can make an item to start on column 1 and span 3 columns like this:

```
    .item{
        grid-column: 1/span 3;
    }
```

CSS Media Queries

Used to apply CSS only when a certain condition is true:

Syntax:

```
          @media only screen and (max-width: 800 px){

        body{
            background: red;
        }
    }
```

Transforms, Transitions & Animations

Transforms are used to rotate, move, skew or scale elements. Theya re used to create a 3-D effects.

The transform Property

Used to apply a 2D or 3D transformation to an element

The transform-origin Property

Allows to change position if transformed elements.

2D transforms: can change x & y axis.

3D transforms: can change z axis as well.

CSS 2D transform methods

You can use the following 2D transform in CSS.

- Translate()
- Rotate()
- scaleX()
- scaleY()
- skew()
- matrix()
- scale()

CSS 3D transform methods

- rotateX()
- rotateY()
- rotateZ()

2D and 3D Transforms

Transforms allow you to move, rotate, scale, skew, element.

Transform property

- The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.

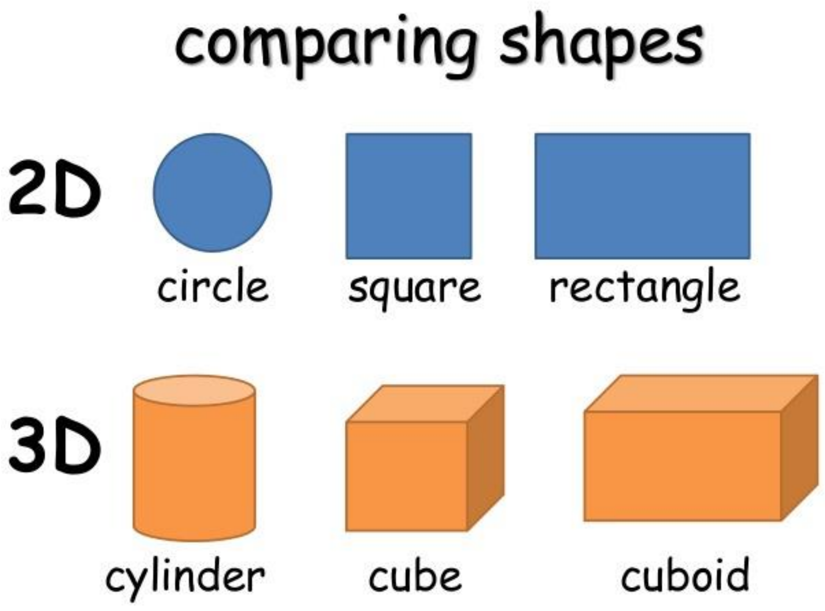
2D transform :

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()

3D transform :

- rotateX()
- rotateY()
- rotateZ()

Difference



2D Transforms

1. **translate()-** moves an element from its current position.
2. **rotate()-** rotate an element clockwise/counter-clockwise according to degree.

Using negative values will rotate the element counter-clockwise.

3. **scale()-** increases/decreases the size of an element.
4. **scaleX()-** increases or decreases the width of an element.
5. **scaleY()-** increases or decreases the height of an element.
6. **skew()-** skews an element along the X and Y-axis by the given angles.
7. **skewX()-** skews an element along the X-axis by the given angle.[degree 0-360]
8. **skewY()-** skews an element along the Y-axis by the given angle.
9. **matrix()-** 6 parameters
10. mathematical function

matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

transform: matrix(1, -0.3, 0, 1, 0, 0);

3D Transforms

- **rotateX()-** rotates an element around its X-axis.
- **rotateY()-** rotates an element around its Y-axis.
- **rotateZ()-** rotates an element around its Y-axis.

CSS Transitions

Used to change property values smoothly, over a given duration.

Transition Property

The transition property is used to add transition in CSS.

- **Transition-property**: The CSS property you want to add an effect
- **Transition-duration:** The duration of the effect
- **Transition-timing-function:** How you want the property to transition.
- **Transition-delay** : The transition-delay property specifies a delay (in seconds) for the transition effect.
- **Transition(shorthand property)**
  - transition: width 2s linear 1s

Transition-timing-function

- **ease** - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- **linear** - specifies a transition effect with the same speed from start to end
- **ease-in** - specifies a transition effect with a slow start
- **ease-out** - specifies a transition effect with a slow end
- **ease-in-out** - specifies a transition effect with a slow start and end

Animations

- CSS allows animation of HTML elements without using JavaScript or Flash!
- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times as you want.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.

Properties to add animation

Following are the properties used to set animation in CSS.

1.	**Animation-name :**	Name of the animation.
2.	**Animation-duration:**	How long does the animation run ?
3.	**Animation-timing-function:**	Determines speed curve of the animation.
4.	**Animation-delay :**	Delay for the start of an animation.
5.	**Animation-iteration-count:**	number of items an animation should run.
6.	**Animation-direction :**	Specifies the direction of the animation