Cascading Style Sheet(css)

Cascading Style Sheet

- CSS stands for cascading style sheet
- Css defines how html elements are to be displayed in the browser
- Css can control the layout of multiple web pages at once
- Syntax
  - Selector{Property:Value; Property:Value….}
  - Selector points to html element we want to style
  - Declaration block contains one or more declarations separated by semicolons.
  - Each declaration includes a css property and a value separated by a colon.
  - Declaration blocks are surrounded by curly braces.

Example

- h1 { color: red; list-style:none}
- h1 is a selector that points to html element that takes the effect defined
- color and list-style is the property and red and none are the values

Inserting CSS

- There are three ways of inserting a styles:
  - Inline CSS
  - Internal CSS
  - Extenal CSS

Inline CSS

- Inline css is used to apply the style for a single element.
- Inline css is added using the style attribute in the html element
- Style attribute can contain any css property.
- An inline css loses many of the advantages of a style sheet. So, it is rarely used.
- Example:
  - <h1 style="color:red; text-align:center"> This is heading 1</h1>
  - style="color:green;"> This is a paragraph

Internal CSS

- An internal style is applicable only for a single html page
- Internal style is defined in <style> element inside the <head> section.
- Inline and Internal css are used within the html file.
- Example:

```
<html><head>
<style>
h1{ color:blue; text-align:center;}
p{color:red;}
</style>
<body>
        <h1> This is heading</h1>
        <p> This is a paragraph</p>
</body>
</html>
```

External CSS

- All the styles are defined in the separate css file having file extension .css file.
- Html page should reference the css file to get the effect of style defined
- We can change the look and feel of an entire website by changing just the one file
- External css are defined using the <link> element inside the <head> section of a html page.

Example:

```
<html>
 <head>
   <link rel=“stylesheet” href=“global.css”>
 </head>
 <body>
  <h1> This is a heading</h1>
  <p> This is a paragraph</p>
 </body>
</html>
```

CSS Selectors

- CSS selectors are used to find or select the HTML elements we want to style
- There are mainly five ways to select the html elements
  - Element Selector
  - Id Selector
  - Class Selector
  - Universal selector
  - Group selector

CSS Element Selector

- Html elements are selected by the element name
- Example:

```
p{ text-align:center; color:red;}
```

- Here, the style is applied to all the paragraph element in the page.
- It will align the text to center and color to red to all the paragraph elements.

CSS ID selector

- ID selector uses the id attribute of an HTML element to select a specific element
- ID of an element should be unique in the page, So, Id selector is used to select one unique element.
- We use `#` character followed by the id of the element to select the html element.

Example

- #mypara{ text-align:center; color:red;}
- This is my paragraph
- The above css rule will be applied to all the html element that have id="mypara"

CSS class selector

- The class selector uses the class attribute of an html element to select a specific element.
- The class of an element is not unique within a page.
- Writing the same class name to many elements will apply the same css to all those elements at once.
- We use period `.` character followed by the class name to define the styles

Example

- .red-center-para{text-align:center; color:red;}
- This is paragraph with text align to center with red color

CSS Universal Selector

- Universal selector is a special type selector that matches any type.
- This is useful when dealing with documents containing multiple namespaces such as HTML within inline SVG or MathML or XML that mixes multiple vocabularies.
- It is used or resetting the css styles
- It is also used for inspecting the html elements
- *{text-align:center; color:red}
  - It applies the style to all the html elements on the page.
- *{ outline: 1px solid blue}
- *{ padding:0px}

CSS Grouping Selector

- When we require to apply the same style to many html elements or class or id, we use this selector.
- It prevents from writing same style for each of the elements
- We use comma `,` characters between multiple selectors to group them.
- Example:
- p, .mypara1, #myheading {

  color:green

  }

It applies the text color to green for all <p> elements, class with mypara1 elements and id with myheading elements.

CSS Colors

- Css colors are used to define color properties of various elements
- They allow developers to specify the visual appearance of text, backgrounds, borders of htmle elements.
- There are many ways to define the css color values like:
  - Named colors
  - Hexadecimal values
  - RGB (Red Green Blue)
  - RGBA (Red Green Blue Alpha)
  - HSL (Hue Saturation Lightness)

Named Color

- Css provides a set of predefined color names that can be used directly like red, green blue, yellow etc.
  - Example :
  - <h1 style=“color:green”>Heading</h1>

RGB

- RGB colors are defined using the intensity values of red green and blue components
- Each value ranges from 0 to 255
- Example:
  - <h1 style=“color:rgb(255,0,0)”>Red Heading</h1>
  - <h1 style=“color:rgb(0,255,0)”>Green Heading</h1>
  - <h1 style=“color:rgb(0,0,255)”>Blue Heading</h1>

RGBA

- RGBA colors are similar to RGB but includes an additional component for transparency(alpha)
- The alpha values takes the decimal value ranging from 0 to 1.
- Example:
- rgba(255,0,0,0.5) which represents semi transparent red.

HEX

- HEX colors are represented as a six digit combination of numbers and letters
- Color should start with # character followed by hexadecimal values
- First two digits represent the red componenet
- Second two digit represents the green component
- Third two digit represents the blue component
- Example:
  - #FF0000 for red color
  - #00FF00 for green color
  - #0000FF for blue color

HSL

- HSL colors are defined using three values of Hue, Saturation, Lightness.
- Hue can be defined as a degree on the color wheel ranging between 0 and 360 where 0 is represented by red, 120 by green and 240 by blue
  - Hue(0-360 degrees)
- The value of saturation is in percentage. 100% means the color will be fully saturated.
  - Saturation (0-100%)
- Lightness is also a percentage value where 0% is black and 100% is white
  - Lightness(0-100%)

  - Example: hsl(0,100%,50%) is red.

CSS Backgrounds

- The css background properties are used to add background effect for html elements.
- CSS background can be any color, image or any other properties like
  - CSS background-color
  - CSS background-image
    - Background-repeat
    - Background-attachment

CSS background-color

- The background-color property specifies the background color of an element
- Example:

```
body{
    background-color:blue;
}
```

CSS background-image

- background-image property specifies an image to use as the background of an element
- By default, the image is repeated so it covers the entire element
- Example:

```
body{
background-image:url(‘happy.jpg’);
}
```

CSS background-repeat

- By default, the background-image property repeats an image both horizontally and vertically
- If we need to repeat image vertically then we can set

  background-repeat:repeat-y;
- If we need to repeat image horizontall then we can set

  background-repeat:repeat-x;
- If we do not want to repeat image, we can set

  background-repeat:no-repeat;

CSS background-attachment

- The background-attachment property specifies whether the background image should scroll or be fixed.
- If we want to scroll then we can set its value ‘scroll’ otherwise we can set as ‘fixed’. By default its value is ‘scroll’
- Example:

```
Body{
background-image:url(‘happy.jpg’);
background-repeat: no-repeat;
background-attachment:fixed;
}
```

CSS Borders

- The css border properties allow us to specify the style, width and color of an element’s border.
- Example:

  div{ border: 1px solid green;}
- The border-style property specifies what kind of border to apply. Its values are:
  - Dotted□defines a dotted border
  - Dashed□defines a dashed border
  - Solid□ defines a solid border
  - Double□defines a double border
  - Inset□defines a 3d inset border
  - outset□defines a 3d outset border
  - none□No border
  - Hidden□defines a hidden border

CSS Text

- CSS has a lot of properties for formatting text
- Text Color:
  - The color property is used to set the color of the text.
  - Color value can be a named color, HEX value or RGB
  - Example:

```
body{ color:blue}
h1{ color:green}
```

This set the default color for a page to blue but for every h1 element it sets the green color text.

Text Alignment

- The text-align property is used to set the horizontal alignment of a text. A text can be left or right aligned or centered or justified.
- Example

```
p{ text-align:center;}
```

Text Decoration

- The text-decoration property is used to set or remove decorations from the text.
- The value text-decoration:none; is often used to remove underlines from links.
- Example
  - a{ text-decoration:none;}

Text Transformation

- The text-transformation property is used to specify uppercase or lowercase letters in text.
- This property can be used to turn every characters into uppercase or lowercase letters or capitalize the first letter of every words.
- Example:

```
p{ text-transform:uppercase;}
p{ text-transform:lowercase;}
p{ text-transform: capitalize;}
```

Text Spacing

- The text-indent property is used to specify the indentation of the first line of a text.

  Example: p{ text-indent:50px;}
- The letter-spacing propery is ued to specify the space between each characters in a text

  Example: p{letter-spacing:1px;}
- The line height property is used to specify space between lines.

  Example: p{ line-height:0.8;}
- The word-spacing propery is used to specify the space between the words in the text.

  Example: p{word-spacing:2px;}

Text Shadow

- The text-shadow property is used for adding shadow to text.
- Example:

  p{text-shadow: 2px 3px 5px red;}

  Here, it specifies

  2px horizontal shadow,

  3px vertical shadow,

  5px blur and

  red as the shadow color

CSS Fonts

- Choosing the right font has a huge impact on how the reader experience a website. So, choosing the right font is important.
- The font adds values to our text.
- It is also important to choose the correct color and size of font in the text.
- Some font properties are:
  - Font-family
  - Font-size
  - Font-weight
  - Font-style

Font-family:

Specifies he preferred font family or a list of font families for the text. If the first font is not available, browser will try the next one.

Example:

p{font-family: “Helvetica”, Arial, sans-serif;}

Font-size:

Sets the size of the font. It can be defined in different units like pixels (‘px’), ems (‘em’) or pecentages (‘%’).

Example:

p{font-size: 16px;}

Font-weight:

Determines the thickness or boldness of the font. Common values include ‘normal’, ‘bold’, ‘lighter’, ‘bolder’ and numeric values like ‘400’ or ‘700’

Example:

```
p{ font-weight:bold; }
```

Font-style: Defines whether the font should be displayed in a normal style or italic Common values are ‘normal’ and ‘italic’

Example:

```
p{ font-style: italic;}
```

CSS Lists

- In html, there are two types of lists. Ordered and Unordered list
- The css list propery allow us to use different item markers.
  - Circle
  - Square
  - Upper-roman
  - Lower-alpha etc
- Example:

ul.a{ list-style-type:circle’}

ul{ list-style-image:url(‘marker.gif’);} set image as list item marker

ul {background:#ff0000;} Sets the background color to list and list items

CSS Tables

- Different css properties can be used to make the table more readable and attractive.
- We can set following properties for table
  - Table borders
  - Table size
  - Table alignment
  - Table style
  - Responsive table

Table Borders

- To specify table borders in css we use border property
- Example:

```
Table, th, td {
border: 1px solid black;
border-collapse:collapse;
}
```

It removes the spaces between the cells and adds the 1px solid black border

Table Size

- The table size is defined by the width and height properties.
- Example:
  - table { width:100%; }
  - th, td{ height: 25px; }

Table takes the 100% of its width and set the cell height to 25px.

Table Alignment

- The text-align property sets the horizontal alignment.
- The text-align property has ‘center’, ‘left’, ‘right’ values
- Example:
  - th{ text-align:center;}

- The vertical-align property sets the vertical alignment.
- The vertical-align takes ‘top’, ‘middle’, ‘bottom’ values
- Example:
  - Td{ vertical-align:middle;}

Table Style

- To control the spaces between the border and content in the table, we use padding propery on <td> and <th> elements
  - Example: th, td{padding:10px;}
- We add the border-bottom property to <td> and <th> for horizontal dividers
  - Example: th, td { border-bottom: 1px solid red; }
- We use :hover selector on <tr> to style the row on hover over the row.
- Example: tr:hover {background-color: #ddd;}

Table Style

- For alternating rows (zebra-striped) , we can use nth-child() selector and add a background-color to all the even or odd table rows
  - Example: tr:nth-child(even){ background-color: #ddd;}
- We can set text color and background color of <th> and <td> elements as well.

  Example: th{

  background-color: #fefefe;

  color:#454545;

  }

Responsive Table

- A responsive table will display a horizontal scroll bar if the screen is too small to display the full content.
- For this, we add a table inside a container element (eg: div) and set the css for container with overflow-x:auto;
- Example:

<table>

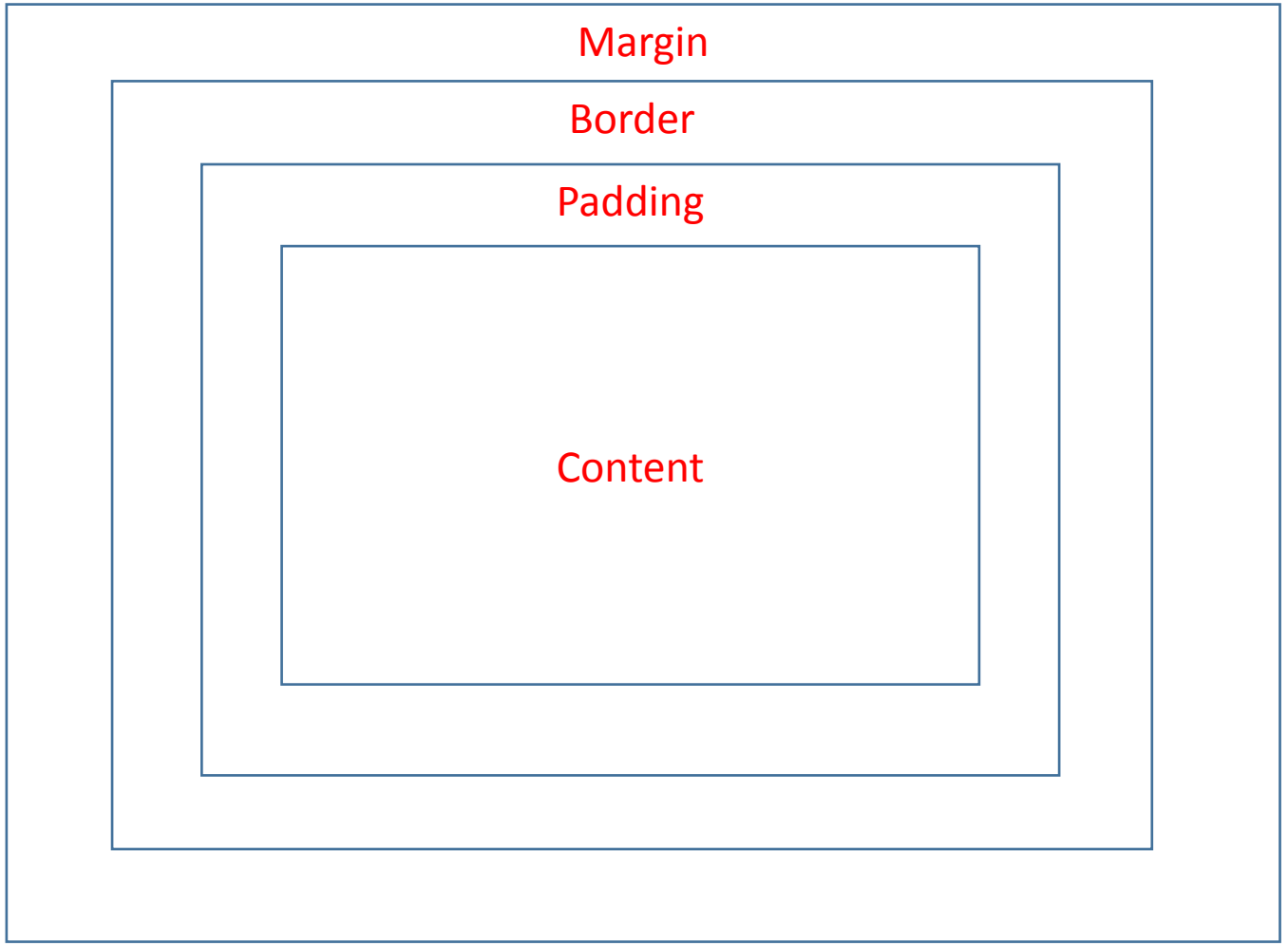
…..

</table>

CSS Box Model

- Html elements can be considered as a boxes.
- In css the term ‘box model’ is used when talking about design and layouts.
- It consists of several layers that surround the html element like content, padding, border, and margin.
- These layers determine the overall size and spacing of an element within the layout of a webpage.

Box Model



Box Model

- Content: the content of the box where text and images appear
- Padding: Transparent area or space between the border and content. It is the space inside the border.
- Border: It is an area that goes around padding and content.
- Margin: It is an transparent area or space outside the border

Box Layout

The CSS box layout refers to the different models used to layout and position these boxes on the web page. There are several box layout models in CSS, each designed to handle different types of layouts:

1. **Block Layout**: Used for block-level elements like `<div>`, `<p>`, and `<h1>`. These elements take up the full width available and stack vertically.
2. **Inline Layout**: Used for inline elements like `<span>`, `<a>`, and `<em>`. These elements flow horizontally within a line of text.
3. **Flexbox Layout**: A layout model designed to arrange elements in a flexible way, allowing them to grow, shrink, and align based on available space. Useful for creating complex layouts with a simple and predictable structure.
4. **Grid Layout**: A powerful layout model that allows for the creation of complex, responsive grid-based designs. It divides the space into rows and columns, enabling precise placement of elements.
5. **Table Layout**: Used for `<table>` elements and their associated `<tr>`, `<td>`, etc., organizing content in a grid of rows and columns.
6. **Positioning Layout**: Involves using the `position` property (static, relative, absolute, fixed, sticky) to position elements in a specific manner within the document.

Display Property

- The display property specifies how an element is displayed.
- Every element has a default display value depending on what type of element it is.
- The default display value for most elements is ‘block’ or inline.
- Values for the display properties are:
  - block
  - inline
  - none

Block level elements

- Block level elements:
  - A block-level element always starts on a new line and takes up the full width available.
  - ,<h1> to <h6>,<p>,<form>,<header>,<footer>,<section> are som examples
  - When display:block is style is set for inline elements, it will make them display as block level elements.

  - Example: #myImage:{ display:block; }
  - <img id=“myImage src=“happy.jpg” alt=“happy face”>

Inline Elements

- Inline Elements:
  - An inline element does not start on a new line and only takes as much width as necessary
  - ,<a>,<img> ar esome examples of inline elements.
  - When display:inline is set for block level elements, it can make the block level elements to be displayed as the inline elements.

  - Example: #mydiv:{ display:block; }
  - my div content goes here…..

Display:none

- The display:none is commonly used with javascript to hide and show elements without deleting or recreating them.
  - Example: #mydiv:{ display:none; }
  - my div content goes here…..
- In the above example, div will not be visible in the browser. But still be available in the DOM.

CSS Padding

- The css ‘padding’ property are used to generate space around an elements content inside the defined borders.
- In css we have setting to set the padding on each side of an element. i.e top, right, bottom, left
- For padding to set in individual side, we have properties like
  - padding-left
  - padding-top
  - padding-right
  - padding-bottom

Example

- div{
  padding-top:50px;
  padding-right:30px;
  padding-bottom:50px;
  padding-left:30px;
  }

Padding shorthand property

- To shorten the code, it is possible to specify all the padding properties in one.
- We use padding property for shorthand property.
- Syntax for short hand is:
  - padding: top right bottom left
- Example:
  - div{
    padding: 50px 30px 50px 30px;
    }
  - div{
    padding: 50px 30px 30px; //top, right/left, bottom
    }
  - div{
    padding: 50px 30px; // top/bottom, right/left
    }
  - div{
    padding: 50px; //all sides
    }

CSS Margin

- The margin property are used to create space around the elements outside of the border.
- In css we have setting to set the margin on each side of an element. i.e top, right, bottom, left
- For padding to set in individual side, we have properties like
  - margin-left
  - margin-top
  - margin-right
  - margin-bottom

Example

- div{
  margin-top:50px;
  margin-right:30px;
  margin-bottom:50px;
  margin-left:30px;
  }

Margin shorthand property

- To shorten the code, it is possible to specify all the margin properties in one.
- We use margin property for shorthand property.
- Syntax for short hand is:
  - margin: top right bottom left
- Example:
  - div{
    margin: 50px 30px 50px 30px;
    }

CSS Positioning

- The position property specifies the type of positioning method used for an element
- There are five different position values:
  - Static,
  - Relative
  - Fixed
  - Absolute
  - Sticky
- Elements are then positioned using the top, bottom left and right properties. However these properties will not work unless the position property is set.
- Html elements are positioned ‘static’ by default which is not affected by top, bottom, left and right properties and positioned according to normal flow of page

Position:relative

- An element with postion:relative is positioned relative to its normal position.
- Setting the top, right bottom and left properties will cause the element to be adjusted away from its normal position.
- Example

```
div{
position:relative;
left: 50%;
top:50%;
}
```

Position: fixed

- An element with position:fixed is positioned such that it always stays in the same place even if the page is scrolled.
- The top, right, bottom and left properties are used to position the element.
- Example:
- div{

```
    position:fixed;
    top:0px;
    left:0px;
}
```

Position: absolute

- An element is position: absolute is positioned relative to the nearest positioned ancestor.
- If it has no positioned ancestors it used the document body and moves along with page scrolling.
- Absolute positioned elements are removed from the normal flow and can overlap elements
- Example:

```
div.relative{position:relative;}
div.absolute{postion:absolute; top:80px; right:0px;}
<div class=“relative”>this is relatively positioned div</div>
<div class=“absolute”> this is absolutely positioned div </div>
```

Example

```
.relative{
        position:relative;
        height:300px;
        width:500px;
        border: 1px solid green;
}
.absolute{
        position:absolute;
        left:50px;
        right:50px;
        top:-10px;
        border:1px solid red;
}
```

```
<div class="relative">
    <span>this is relative div</span>
    <div class="absolute">
    <span> this is absolute div</span>
    </div>
</div>
```

Position: sticky;

- The element with position: sticky; is positioned based on the user’s scroll position.
- A sticky element toggles between relative and fixed depending on the scroll position.
- It is positioned relative until a given offset position is met in the viewport, then it sticks in the place like position:fixed
- Additionally, at least one of the following properties: top, bottom, left or right must be provided.
- As the user scrolls down the page, the sticky element behaves like position: relative until it reaches its sticky position relative to parent container

Example

```
.container {
    position: relative;
    height: 400px;
    overflow-y: scroll;
}

.sticky-element {
    position: sticky;
    top: 20px;
    background-color: #f2f2f2;
    padding: 10px;
}

.content {
    height: 800px;
}
```

```
<div class="container">
    <div class="sticky-element">Sticky Element</div>
    <div class="content">
        <!-- Content goes here -->
    </div>
</div>
```

Box-Shadow Property

- The css box-shadow property applies shadow to elements.
- It is works similar to text shadow.
- It is used to specify the horizontal shadow, vertical shadow and color
- Example:

```
div.card{
box-shadow: 5px 10px 15px grey
}
```

Here, 5px is horizontal shadow

10px is vertical shadow

15px is blur

Grey is the shadow color

Text effects

- Some of the text effects that we use are:
  - CSS text overflow
  - CSS word wrapping
  - CSS word breaking
  - CSs writing mode

CSS Text Overflow

- The css text-overflow property specifies how should the overflowed content be viewed
- It can be have two behavior: clip or ellipsis
- When it is clip, overflow text are hidden
- When it is ellipsis, it shows three dots (…)and overflow text are hidden
- Example:

```
p{
width:150px;
text-overflow:ellipsis;
white-space:nowrap;
overflow:hidden;
}
```

CSS Word Wrapping

- The css word-wrap property allows long words to be able to be broken and wrap into the next line.
- If a word is too long to fit within an area, it expands outside and may break the layout. In such case we use word-wrap property
- Example

```
p{
width:200px;
word-wrap:break-word;
}
```

-
```
<p>
    ThisIsALongWordThatMightBreakTheLayoutWithoutWordWrap
</p>
```

CSS Word Breaking

- The css word-break property is used to control how words should be broken and wrapped within an element’s content when they exceeds the available space.
- Its particularly useful when dealing with languages that don’t use spaces between words such as Chinese, Japanes or Korean
- It can have three values: break-all, break-word, keep-all
- Break-all can break the word any characters
- Break-word will break the word only at hypenation points
- Keep-all will not break the words.

CSS Writing Mode

- The writing-mode property in css controls the writing and layout of content within an element.
- It specifies whether text should be laid out horizontally or vertically and whether it should be displayed from left to right or right to left.
- It is particularly useful for supporting languages and scripts that are written vertically or right to left.
- The writing-mode property has several values that determine the orientation and directions of text

Writing Mode Values

- Horizontal –tb (default) text are laid out horizontally from left to right with lines stacking from top to bottom.
- Vertial-rl: text are laid out vertically and lines are stacked from right to left
- vertical-lr text are laid out vertically and lines are stacked from left to right.

Responsive Web Design

- Web pages can be viewed using many different devices like desktops, tablets and phones
- Web page should look good and easy to use regardless of the device
- Reponsive web design make the web pages look good on all devices.
- We call it responsive web design when we use css and html to resize, hide, shrink, enlarge or move the content to make it look good on any screen.
- Principles of Responsive Web Design are:
  - Viewport
  - Media Queries
  - Flexible Images
  - Fluid Grids

Viewport

- The user’s visible area of a webpage is called viewport
- The viewport varies with devices and will be small on mobile phones than a computer screen
- Before tablets and mobile phones, web pages were designed only for computer screens and it was common for web pages to have static design and a fixed size.
- Now web pages need to be changes according to viewport of the device
- HTML5 introduced the way to control the viewport using <meta> tag

- <meta name=“viewport” content=“width=device-width, initial-scale=1.0”>
- This gives the browser instructions on how to control the page dimensions and scaling.
- The “width=device-width” sets the width of the page to follow the screen width of the device.
- The “initial-scale=1.0” sets the initial zoom level when the page is first loaded in the browser
- <meta> tag only doesn’t make the page responsive but it is a quick fix method.

Media Queries

- Media query is a css technique introduced in css3.
- It uses the @media rule to include a block of css properties only when some conditions is true to make the webpage responsive.

Example: @media only screen and (max-width:600px){

div{ width:360px;}

}

- The example above sets the div width to 360px if the browser window is 600px or small.

There are tons of screens and devices with different heights and widths. So, it is hard to create an exact breakpoints for each device.
We can target mainly five groups

@media only screen and (max-width:600px) for extra small devices (phones, 600px and down)
@media only screen and (min-width:600px) Small devices (portrait tabs and large phones)
@media only screen and (min-width:768px) Medium devices(landscape tabs, 768px and up)
@media only screen and (min-width:992px) Large devices(laptops/desktops, 992px and up)
@media only screen and (min-width:1200px) Extra large devices(large laptops and desktops, 1200px and up)

Media queries can also be used to change layout of a page depending on the orientation of the browser:
Example: @media only screen and (orientation:landscape){

```
    body{
        background-color:lightblue;
    }
}
```

Flexible Images

- Image are made responsive by setting the maximum width to 100%
- This ensures that images automatically scale down to fit smaller screens without overflowing

Fluid Grids

- Instead of using the fixed pixel width, responsive designs use fluid grids that adjust based on the screen size.
- This allows content to proportionally scale up or down as the screen size changes.
- It is done using the flexbox.

Introduction to Bootstrap

- Bootstrap is a free and open-source most popular css framework for developing responsive web pages.
- Bootstrap 5 is the newest version of bootstrap
- Bootstrap provides a collection of pre-designed HTML, CSS and Javascript components that can be easily customized and integrated to projects.
- It can be added into the project either downloading the bootstrap bundle or directly using the bootstrap CDN

Key features of bootstrap

- **Responsive Grid System:** Bootstrap offers a responsive grid system that allows you to create flexible layouts that adapt to different screen sizes. The grid system is based on a 12-column layout, making it easy to arrange content across various devices.
- **Ready-to-Use Components:** Bootstrap includes a wide range of components such as navigation bars, buttons, forms, modals, carousels, and more. These components are styled and designed to ensure consistency and a polished appearance.
- **Typography:** Bootstrap provides a well-designed set of typographic styles, including headings, paragraphs, lists, and other text formatting options.

- **JavaScript Components:** Bootstrap includes JavaScript components that enhance user interactions and functionality, such as dropdowns, modals, tooltips, and more.
- **Extensive Documentation:** Bootstrap comes with comprehensive documentation that provides guidance on using its components, classes, and features. The documentation also includes examples and usage instructions.