CSS INTERVIEW Q&A

1. Difference between flex-direction as row and column.

This is a CSS property based on the concept of the flexbox. Flex-direction property can be applied to the parent elements and then the child elements of this parent tag will be placed accordingly.

The difference is when we apply flex-direction : row, child elements will be placed one after another in the same row (horizontally) and then the row or x-axis will be considered as the main axis, in this case the y-axis will be known as cross axis.

HTML:-

```
    <div class="container">
        <div class="flex-items"></div>
        <div class="flex-items"></div>
        <div class="flex-items"></div>
    </div>
```

CSS:-

```
.container{
    display: flex;
    flex-direction: row;
}

.flex-items{
    width: 100px;
    height: 100px;
    background-color: blueviolet;
    margin: 20px 20px;  }
```

Output:-

On the other hand if we apply flex-direction : column places the child elements in column(vertically) means every child element will be placed one after another in the same column, and then the y-axis will be considered as the main axis and x-axis will be known as cross axis.

HTML:-

```
    <div class="container">
        <div class="flex-items"></div>
        <div class="flex-items"></div>
        <div class="flex-items"></div>
    </div>
```

CSS:-

```
.container{
    display: flex;
    flex-direction: column;
}

.flex-items{
    width: 100px;
    height: 100px;
    background-color: blueviolet;
    margin: 20px 20px;
}
```

Output:-

****

2. Explain the Inline, Internal, and External Stylesheet.

In CSS, we have 3 different ways of writing styles for HTML elements:

- Inline
- Internal
- External

Inline styling gets applied directly to the element in the style attribute. This styling is having the highest priority among all. But it is not advised to use this styling much in projects.

HTML:-

```
<h1 style="color: red; text-align:center;">Heading One</h1>
<p style="color: blue;">This is a paragraph</p>
```

Output:-

Internal styling gets written in the head tag of the HTML file. It will be written in the style tag. The styling written in style tag will be applicable only in that HTML file.

HTML:-

```
<head>
        <style>
        h1 {
          color: maroon;
          margin-left: 40px;
        }
```

```
        p{
            color: maroon;
        }
        </style>
        </head>
</head>
<body>
    <h1>First Heading</h1>
    <p>This is a paragraph</p>
</body>
```

Output:-

External Styling is done by creating a different CSS file with .css extension. This file can be linked to any HTML file wherever the styling is required. The external CSS file can be attached to the HTML file in the head tag in the link tag. This styling is having the least priority but it is the most advised way of writing styling in projects.

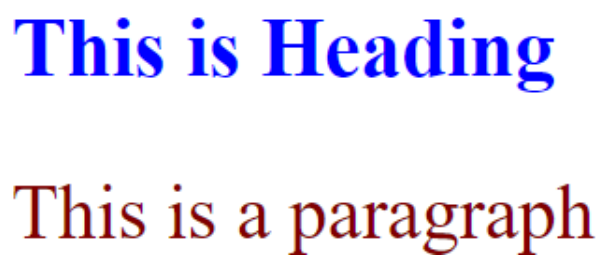
HTML:-

```
<head>
    <title>Document</title>
    <link rel="stylesheet" href="mystyle.css">
</head>
<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
</body>
```

CSS:- (file name : mystyle.css)

```
h1 {
    color: blue;
    margin-top: 40px;
  }
p{
    color: maroon;
    font-size: 30px;
}
```

Output:-



****

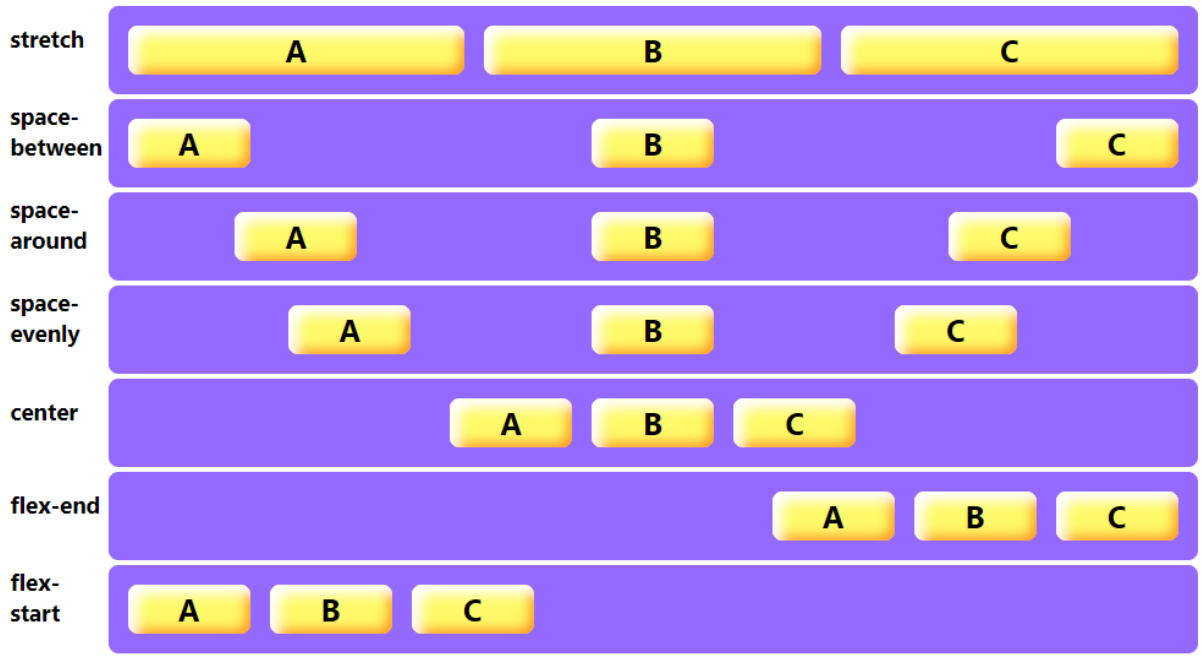
3. What does justify-content allow you to do?

The justify-content property is a sub-property of the Flex-box Module.

The justify-content property specifies how flex-items are distributed along the main axis of their parent flex-container. If the flex-direction of the parent container is row(default value) then the main axis will be the x-axis. If the flex-direction is a column then the main axis will be the y-axis. This property has 6 different possible values:

- flex-start (default)
- flex-end

- center
- space-around
- space-between
- space-evenly
- stretch



****

4. Difference between Absolute and Relative Positioning?

position: relative

- It is used when we need to position the HTML element relative to its normal position.
- We can set top, right, bottom and left properties that will cause the element to adjust away from the normal position.

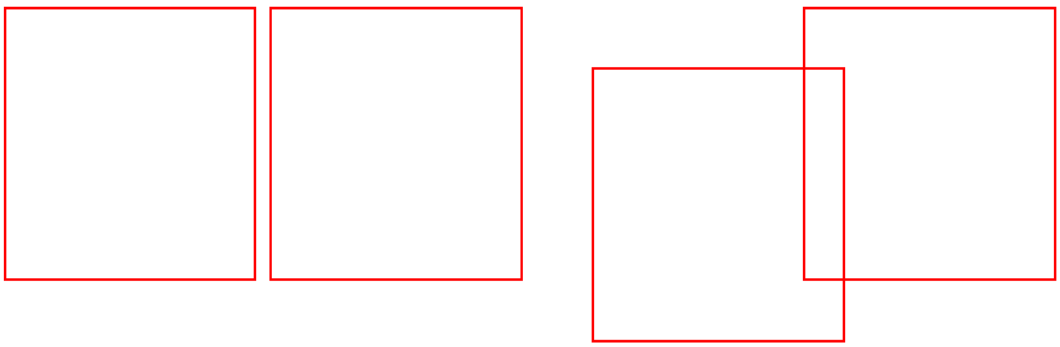
HTML:-

```
<div class="box" id="box1"></div>
<div class="box" id="box2"></div>
<div class="box" id="box3"></div>
<div class="box" id="box4"></div>
```

CSS:-

```
.box {
  border: 2px solid red;
  display: inline-block;
  width: 150px;
  height: 150px;
  margin: 2px;
 }
#box3 {
  position: relative;
  top: 34px;
  left: 34px;
}
```

Output:-



position: absolute

- An element with position absolute will move according to the position of its parent element.
- In the absence of the parent element, the HTML element will be placed according to the body tag. In this case if you want to place the element at any position based on the parent element then we need to give the position : relative to parent element and then position : absolute to child element.

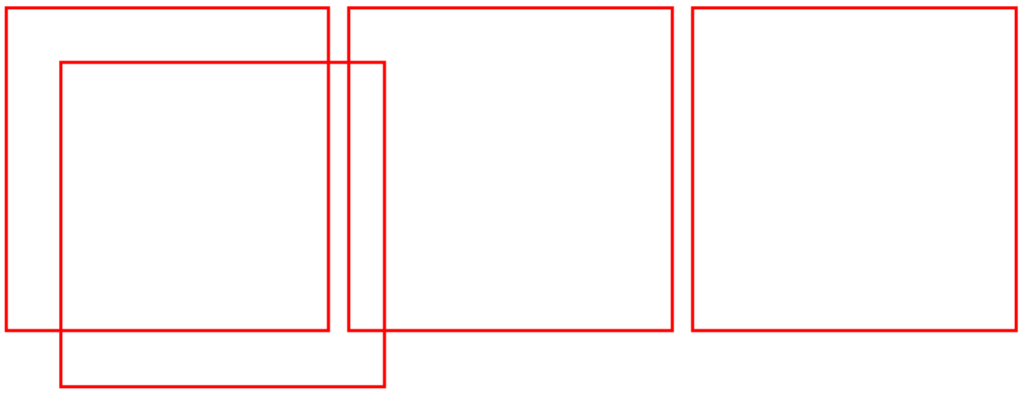
HTML:-

```
<div class="box" id="box1"></div>
<div class="box" id="box2"></div>
<div class="box" id="box3"></div>
<div class="box" id="box4"></div>
```

CSS:-

```
.box {
  border: 2px solid red;
  display: inline-block;
  width: 150px;
  height: 150px;
  margin: 2px;
 }
#box3 {
  position: absolute;
  top: 34px;
  left: 34px;
}
```

Output:-



****

5. What are grid-template-columns used for?

Grid-template-columns property defines how many columns should be there in a **grid-container**. The number of columns is determined by the number of **values** defined in the space-separated list. It takes values in px, % and fraction **(fr).**

HTML:-

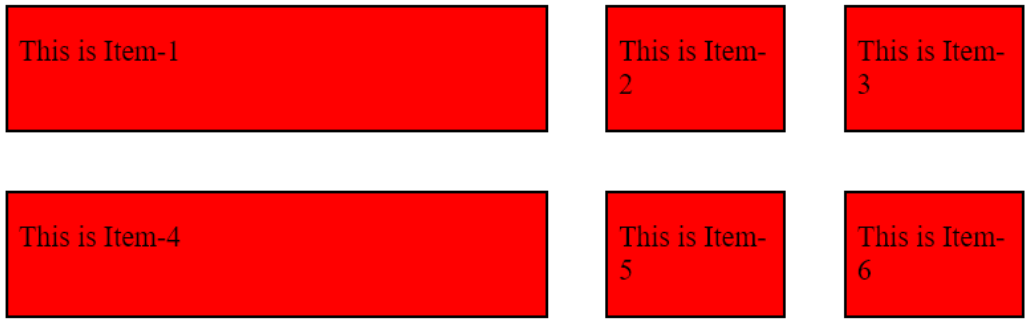
```
<div class="container">
        <div class="item">This is Item-1</div>
        <div class="item">This is Item-2</div>
        <div class="item">This is Item-3</div>
        <div class="item">This is Item-4</div>
        <div class="item">This is Item-5</div>
        <div class="item">This is Item-6</div>
</div>
```

CSS:-

```
.container{
    display: grid;
    grid-template-columns: 300px 100px 100px;     //using pixels
    grid-gap: 2rem;
}
.item{
```

```
    background-color: red;
    border: 2px solid black;
    padding: 15px 5px;
}
```

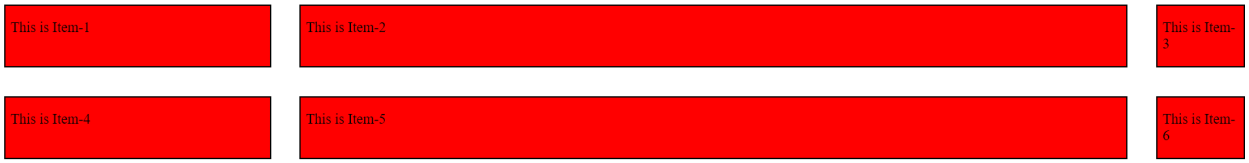
Output:-



CSS:-

```
.container{
    display: grid;
     grid-template-columns: 300px auto 100px;     //using value auto
    grid-gap: 2rem;
}
.item{
    background-color: red;
    border: 2px solid black;
    padding: 15px 5px;
}
```

Output:-



CSS:-

```
.container{
    display: grid;
     grid-template-columns: 1fr 3fr 1fr;          //using fraction
    grid-gap: 2rem;
}
.item{
    background-color: red;
    border: 2px solid black;
    padding: 15px 5px;
}
```

Output:-

CSS:-

```
.container{
    display: grid;
     grid-template-columns: repeat(3, 25%);       //using percentage and repeat
    grid-gap: 2rem;
}
.item{
    background-color: red;
    border: 2px solid black;
    padding: 15px 5px;
}
```

Output:-

****

6. What is z-index in CSS?

The z-index property determines the stack level of an HTML element. The "stack level" refers to the element's position on the *Z axis* (as opposed to the *X axis* or *Y* axis). Elements with a higher index will be placed on top of elements with a lower index. If the z-index value of any element is positive then that element will be at the top and element having negative value of z-index will be at the bottom.

Note :- z-index will only work on an element whose position property has been set to absolute, relative and fixed.

HTML:-

```
<div id="box1"></div>
<div id="box2"></div>
```

CSS:-

```
#box1 {
    width: 100px;
    height: 100px;
    top: 69px;
    position: relative;
    background-color: greenyellow;
    z-index: 1;
}
#box2 {
    width: 150px;
    height: 150px;
    top: 34px;
```

```
    left: 20px;
    position: absolute;
    background-color: rebeccapurple;
    z-index: 0;
}
```

Output:-

****

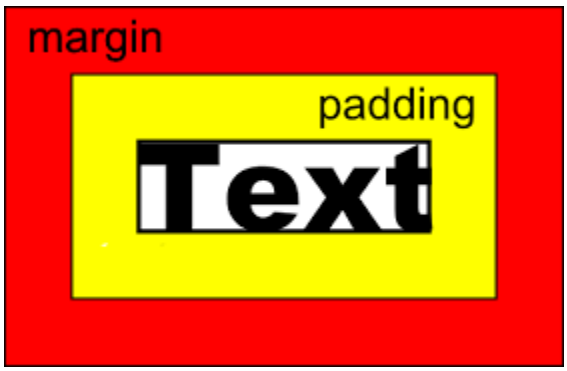
7. What is the difference between Padding and Margin.

These two properties are CSS properties that provide space or gap. Both margin and padding targets the four sides of an element.

Margin	Padding
· Margin provides the space between the border and nearby elements. It is the space outside the border.	· Padding provides the space between the border and the content of an element. It is the space inside the border

· You can set margin values to be negative if you want elements to overlap.	· Padding values can only be positive.
· When you set margin values, the element's size does not change, only the space around it.	· When you set padding values, the size of that element increases.

Illustration:-



****

8. What is box-sizing?

Box-sizing property defines how the width and height gets applied on an element.

Default value of box-sizing is content-box, in this case the actual width of the box will be equal to the border and padding and width of the box (Actual width : width + border + padding).

In case when box-sizing is set to be border-box, then the actual width will be equal to the width of the box itself. Means if we will increase the padding or border of the box then it will not increase the width of the box.

a) content-box (Default)

b) border-box

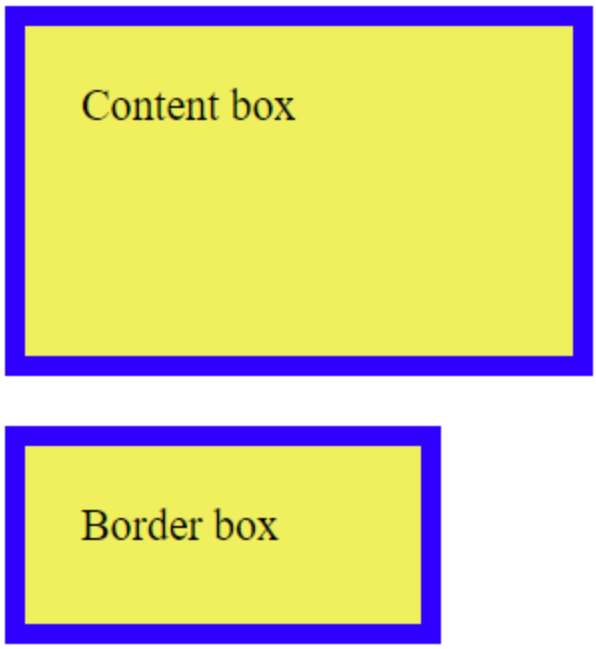
HTML:-

```
    <div class="content-box">Content box</div>
    <br>
    <div class="border-box">Border box</div>
```

CSS:-

```
div {
    width: 160px;
    height: 80px;
    padding: 20px;
    border: 8px solid rgb(47, 0, 255);
    background: rgb(240, 240, 95);
  }

  .content-box{
      box-sizing: content-box;
  }

  .border-box {
    box-sizing: border-box;
  }
```

Output:-



****

9. What is animation-delay?

It is a sub-property of CSS animation. It defines how long the animation has to wait before **starting.** Using the animation-delay property, you define the delay by specifying the number of seconds or milliseconds it should take before playing.

HTML:-

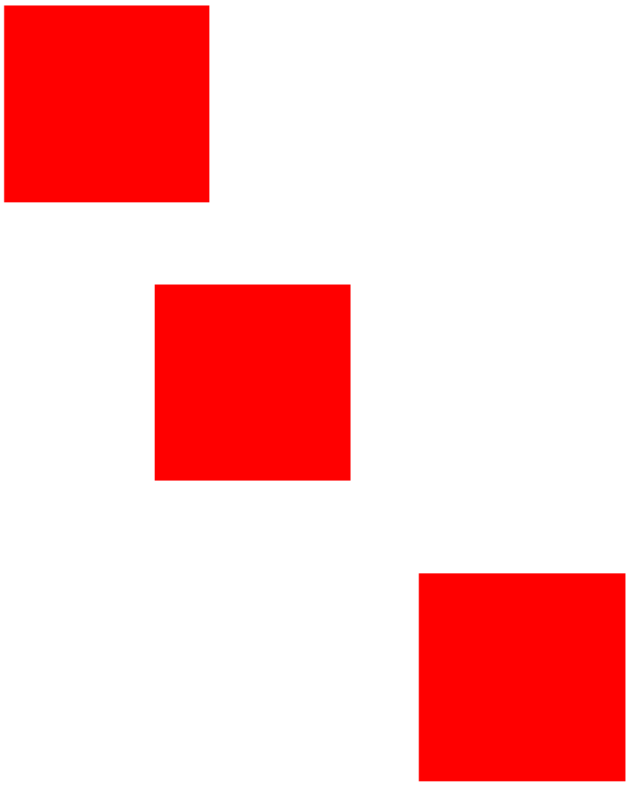
```
<div></div>
```

CSS:-

```
div {
    width: 100px;
```

```
  height: 100px;
  background: red;
  position: relative;
  animation: moving 5s infinite;
  animation-delay: 3s;
}

@keyframes moving {
  from {left: 0px;}
  to {left: 200px;}
}
```

Output:-



****

10. Which property will you use to merge cells horizontally in a grid?

grid-column-start and grid-column-end property can be used to merge cells vertically. It also has a shorthand i.e., grid-column: start/end. If you want to merge two elements in row then we will have to use grid-row property inside the child element. Remember these properties are applicable only to child elements not to parent elements.

HTML:-

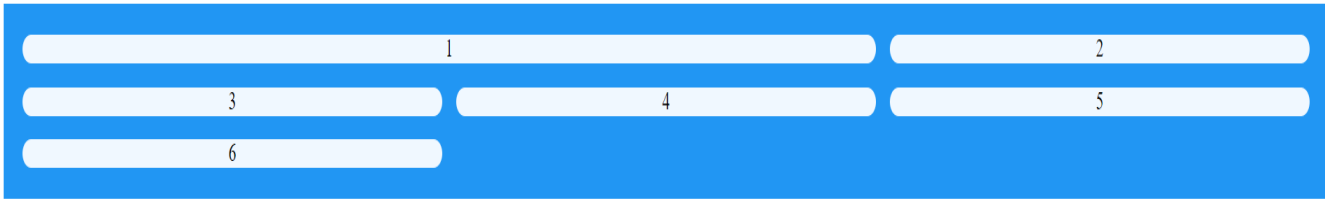
```
<div class="grid-container">
        <div class="item item1">1</div>
        <div class="item">2</div>
        <div class="item">3</div>
        <div class="item">4</div>
        <div class="item">5</div>
        <div class="item">6</div>
</div>
```

CSS:-

```
.grid-container {
    display: grid;
    grid-template-columns: auto auto auto;
    grid-gap: 15px;
    background-color: #2196F3;
    padding: 20px;
    text-align: center;
  }

.item{
    background-color: aliceblue;
    border: #2196F3;
    border-radius: 20px;
}
```

```
.item1{
    grid-column: 1/3;
}
```

Output:-



****

11. Which property will you use to merge cells horizontally in a grid?

grid-row-start and grid-row-end property can be used to merge cells horizontally. It also has a shorthand i.e., grid-row: grid-row-start/grid-row-end;

HTML:-

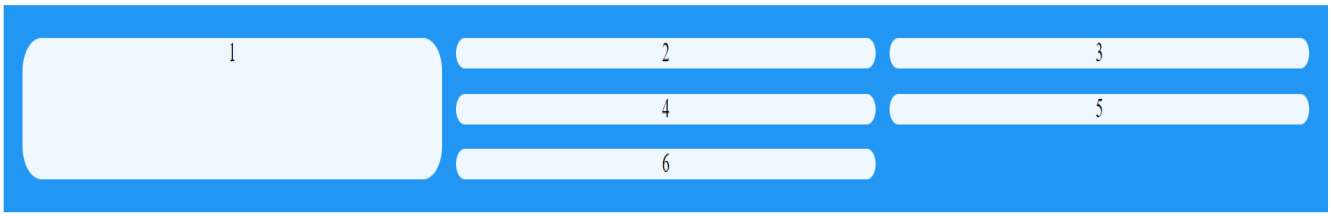
```
<div class="grid-container">
        <div class="item item1">1</div>
        <div class="item">2</div>
        <div class="item">3</div>
        <div class="item">4</div>
        <div class="item">5</div>
        <div class="item">6</div>
</div>
```

CSS:-

```
.grid-container {
    display: grid;
    grid-template-columns: auto auto auto;
```

```
    grid-gap: 15px;
    background-color: #2196F3;
    padding: 20px;
    text-align: center;
  }

.item{
    background-color: aliceblue;
    border: #2196F3;
    border-radius: 20px;
}

.item1{
    grid-row: 1/4;
}
```

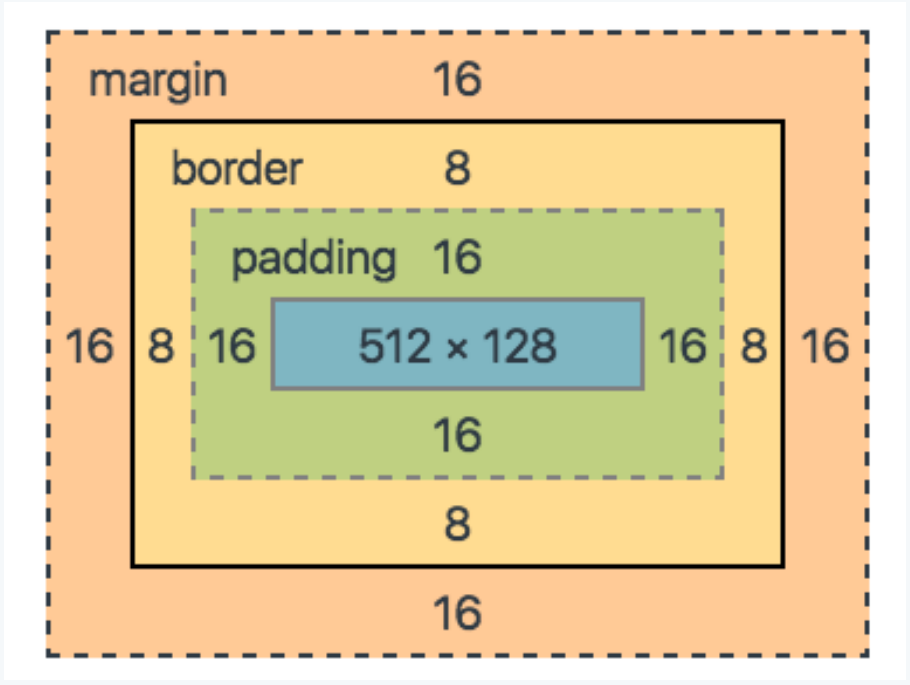
Output:-



****

12. What is a Box-model?

The box model in CSS is a set of rules that determine how your web page is rendered on the internet. In this model, a rectangular box is generated for HTML elements. This box consists of first content, then padding, then border, and finally margin.



****

13. What is the difference between display none and visibility hidden?

display: none;

- The display: none property is used to hide elements without deleting them. It does not take up any space.
- In this case, the tag is removed from the page, which allows other elements to fill in.

visibility: hidden;

- The visibility: hidden property also hides an element, but affects the layout i.e., it takes up the space.
- The element is hidden from view but not the page flow.

****

14. What are CSS Combinators?

Combinators combine the selectors to provide them a useful relationship and the position of content in the document. This comes in handy when you want to style certain elements all at once rather than styling elements individually. CSS combinator selectors are as follows.

1. Descendant Selector (space)
2. Child Selector (>)
3. Adjacent Sibling Selector (+)
4. General Sibling Selector (~)

Descendant Selector : The descendant selector matches all elements that are descendants of a specified element.

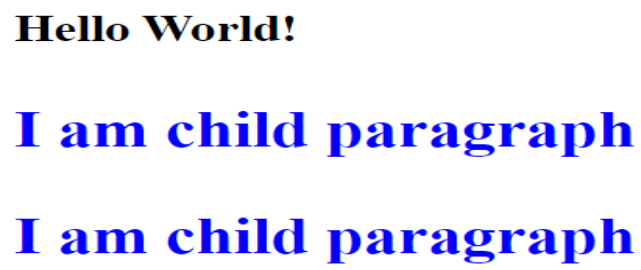
HTML :

```
<div>
  <h2>Hello World!</h2>
  <p>I am child paragraph</p>
  <p>I am child paragraph</p>
 </div>
```

CSS :

```
 div p {  /* --> (div is the parent and the p is the child) */
            color: blue;
            font-family: 'Times New Roman';
            font-weight: bolder;
            font-size: xx-large;
        }
```

Output:



Child Selector : The child selector will only affect elements under a direct parent. A good way to remember this is that a child will only listen to its direct parents.

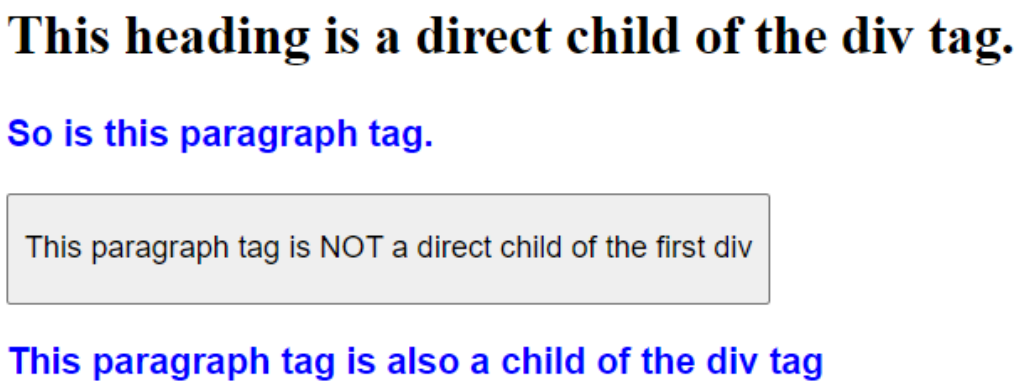
HTML :

```
<div>
    <h2>This heading is a direct child of the div tag.</h2>
    <p>So is this paragraph tag.</p>
    <button>
        <p>This paragraph tag is NOT a direct child of the first div</p>
    </button>
    <p>This paragraph tag is also a child of the div tag</p>
</div>
```

CSS :

```
div > p   {
           color: blue;
           font-family: sans-serif;
           font-weight: bold;
          }
```

Output :



Adjacent Sibling Selector : To use the adjacent sibling selector in your CSS stylesheet, you simply need the plus (+) sign. The adjacent sibling selector is used to select an element that is directly after another specific element.

HTML :

```
<div>
    <h2>This is a H2 tag.</h2>
    <p>This p tag immediately follows the heading.</p>
    <button>
        <p>This p tag is NOT immediately following a H2 tag.</p>
    </button>
    <h2>Howdy! This is another H2 tag.</h2>
    <p>This p tag is after another H2 tag and gets formatted.</p>
</div>
```

CSS :

```
h2+p {
            color: blue;
            font-family: sans-serif;
            font-weight: bold;
        }
```

Output :

General Sibling Selectors :The general sibling selector follows through on all elements under the targeted HTML element selected. This means that if another element tag interrupts the direct adjacent flow, the style rules can still follow through.

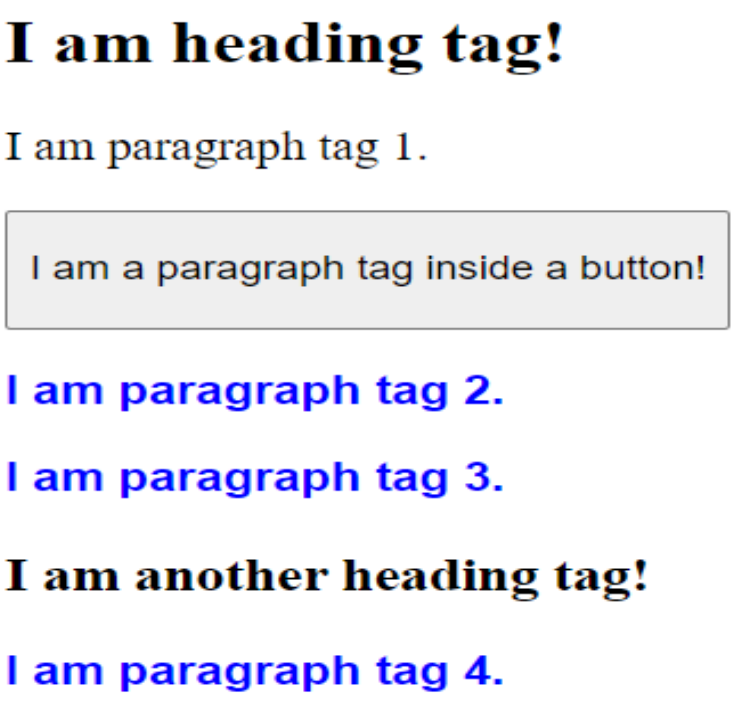
HTML :

```
<div>
        <h2>I am heading tag!</h2>
        <p>I am paragraph tag 1.</p>
        <button>
            <p>I am a paragraph tag inside a button!</p>
        </button>
        <p>I am paragraph tag 2.</p>
        <p>I am paragraph tag 3.</p>
        <h3>I am another heading tag!</h3>
        <p>I am paragraph tag 4.</p>
    </div>
```

CSS :

```
button~p {
            color: blue;
            font-family: sans-serif;
            font-weight: bold;
        }
```

Output :



****

14. What are CSS Pseudo Selectors?

CSS pseudo-classes are used to add styles to selectors, but only when those selectors meet certain conditions. A pseudo class is expressed by adding a colon (:) after a selector in CSS, followed by a pseudo-class such as "hover", "focus", or "active", like this:

```
a:hover {
           /*add your style here */
        }
```

Syntax:

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

Some Examples of Pseudo-Class:

:active	It is used to add style to an active element.
:hover	It adds special effects to an element when the user moves the mouse pointer over the element.
:link	It adds style to the unvisited link.
:visited	It adds style to a visited link.
:lang	It is used to define a language to use in a specified element.
:focus	It selects the element which is focused by the user currently.

:first-child	It adds special effects to an element, which is the first child of another element.

For more Pseudo Class refer this :
https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Pseudo-classes_and_pseudo-elements

****

Easy Implementation

1. Replicate the button of PrepBytes website.

Code:-

```
button {
    width: 200px;
    height:30px;
    background-color: rgb(218, 93, 93);
    border: 2px solid black;
    border-radius: 5px;
      }

   button:hover{
    background-color: lightblue;
}

<button>Enroll Now</button>
```

Output:-

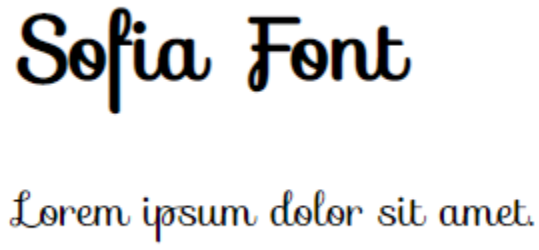
****

2. Write a text and add any google font.

Code:-

```
<link rel="stylesheet"href="https://fonts.googleapis.com/css?family=Sofia">

<style>
body {
  font-family: "Sofia", sans-serif;
}
</style>
</head>
<body>

<h1>Sofia Font</h1>
<p>Lorem ipsum dolor sit amet.</p>

</body>
```

Output:-



****

3. Create a box, increase its width on hover, add transition on hover.

Code:-

```
<div class="box"></div>

.box{
            background-color: #7ce02f;
            width: 80px;
            margin: 20px auto;
            height: 80px;
            transition : width 2s;
        }


  .box:hover {

            width: 200px;
           }
```

Output:-

****

4. Create a circle using border-radius.

If we give equal height and width to a box and give 50% to its border-radius. Then we can achieve a circle.

Code:-

```
<div class="circle"></div>
```

```
   .circle{
            width: 100px;
            height: 100px;
            border-radius: 50%;
            background-color: blue;
        }
```

Output:-

****

5. Write code to show implementation of media-queries.

Media queries is CSS property to add responsiveness to the webpage so that it can be displayed on the screen smoothly.

```
/* When the browser is at least 600px and above */
   @media screen and (min-width: 600px) {
```

```
            .element {
                /* Apply some styles */
            }
        }
```

****

6. Write code to show any animation example.

By using animation property we can add some cool transitions and animations to the element. To use animation, you must define keyframes for the animation. Below example will change the box color in every 0.25seconds starting from red till green.

```
/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-iteration-count: infinite
}

  @keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}

}
```

****

Difficult Implementation

_1. Create header -- left side will have a logo and right side will have 4 buttons.

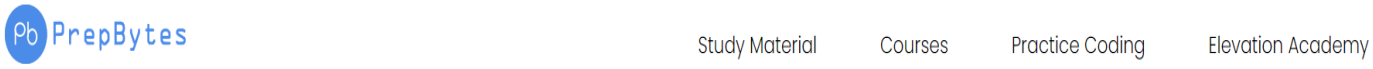
HTML:-

```
<div>
        <div class="navbar">
            <div class="nav-button">Study Material</div>
            <div class="nav-button">Courses</div>
            <div class="nav-button">Practice Coding</div>
            <div class="nav-button">Elevation Academy</div>
        </div>
        <div class="logo"><img
src="https://s3.ap-south-1.amazonaws.com/www.prepbytes.com/navbar/logoPrepBytes.s
vg" height="40px" alt=""></div>
</div>
```

CSS:-

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300&family=Roboto+Seri
f:wght@100&display=swap');

        img {
            margin-top: 15px;
        }

        .navbar {
            float: right;
            margin-right: 70px;
            display: flex;
        }
        .nav-button {
            padding: 30px;
            font-family: 'Poppins', sans-serif;


        }
```

Output:-



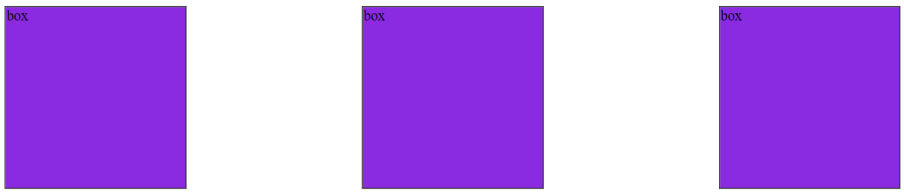
****

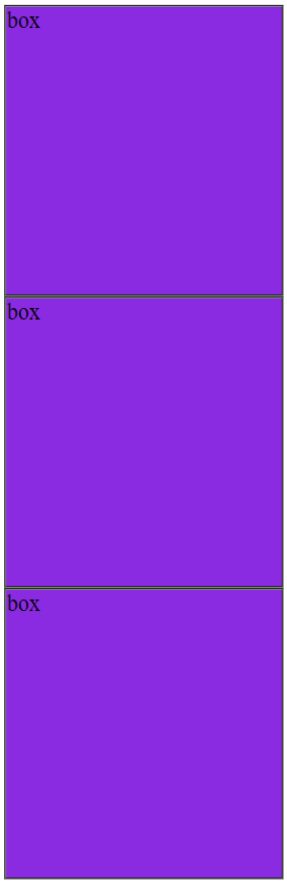
2. Create 3 cards of height & width 200 px with texts in it, the card should be horizontally placed in web view and vertically in mobile view.

Code:-

```
<div class="container">
    <div class="card">box</div>
    <div class="card">box</div>
    <div class="card">box</div>
</div>

.container
{
    display: flex;
    justify-content: space-evenly;
}
.card
{
    border: 2px groove gray;
    width: 200px;
    height: 200px;
}
@media screen and (max-width:720px) {
    .container{
        flex-direction: column
    }
   }
```

Output:-





****

3. Create a button like Login and SignUp button of Prepbytes website.

If mobile : buttons color should be green and they should be below each other

If iPad : buttons color should be red and they should be below each other

If desktop : buttons color should be blue and they should be side-by-side each other

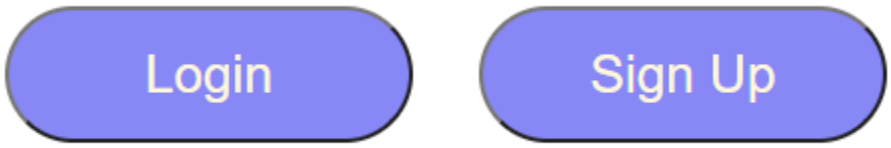
Code:-

```
.button {
            background-color: rgb(135, 135, 245);
            height: 50px;
            width: 150px;
            border-radius: 35px;
            margin:12px;
            font-size: larger;
            color: cornsilk;

        }

        div {
            display: flex;
            justify-content: center;
        }

        @media only screen and (max-width: 800px) and (min-width: 480px) {
            div{
                display: flex;
                flex-direction: column;
                justify-content: center;
                align-items: center;
            }
            .button {
            background-color: green;
            height: 50px;
            width: 150px;
```

```
        border-radius: 35px;
        margin:12px;
        font-size: larger;
        color: cornsilk;

    }




    }
    @media only screen and (max-width: 480px) and (min-width: 320px) {
        div{
            display: flex;
            flex-direction: column;
            justify-self: center;
            align-items: center;

        }
        .button {
        background-color: red;
        height: 50px;
        width: 150px;
        border-radius: 35px;
        margin:12px;
        font-size: larger;
        color: cornsilk;

    }

    }


<div>
    <button class="login button">Login</button>
    <button class="signUp button">Sign Up</button>
</div>
```

Output:-

Desktop Screen :-



iPad Screen :-

Mobile Screen :-

****

4. Write code to show implementation of flex.

Here a navbar with 4 buttons has been created, using flex we can add the flex property to them and adjust and align the items inside them.

Code:-

```
<h1>This is Flexbox Tutorial</h1>
    <div class="container">
        <div class="item" id="item-1">First Box</div>
        <div class="item" id="item-2">Second Box</div>
        <div class="item" id="item-3">Third Box</div>
    </div>
```

```
.container {
            height: 544px;
            width: 100%;
            border: 2px solid black;
            display: flex;
            /* Initialize the container as a flex box */

            /* Flex properties for a flex container */

            /* flex-direction: row; (Default value of flex-direction is row) */
            /* flex-direction: column;
            flex-direction: row-reverse;
            flex-direction: column-reverse; */


            /* flex-wrap: wrap; (Default value of flex-direction is no-wrap) */
            /* flex-wrap: wrap-reverse; */

            /* This is a shorthand for flex-direction: and flex-wrap: ;; */
            /* flex-flow: row-reverse wrap; */

            /* justify-content will justify the content in horizontal direction
*/
            /* justify-content: center; */
```

```
            /* justify-content: space-between; */
            /* justify-content: space-evenly; */
            /* justify-content: space-around; */

            /* justify-content will justify the content in vertical direction */
            /* align-items: center; */
            /* align-items: flex-end; */
            /* align-items: flex-start; */
            /* align-items: stretch; */
        }

        .item {
            width: 200px;
            height: 200px;
            background-color: tomato;
            border: 2px solid green;
            margin: 10px;
            padding: 3px;
        }

        #item-1 {
            /* Flex properties for a flex item */
            /* Higher the order, later it shows up in the container */
            /* order: 2; */

            /* flex-grow: 2;
            flex-shrink: 2; */
        }

        #item-2 {
            /* flex-grow: 3;
            flex-shrink: 3 ; */
            flex-basis: 160px;
            /* when flex-direction is set to row flex-basis: will control width
*/
            /* when flex-direction is set to column flex-basis: will control
height */
        }
```

```
#item-3 {
    /* flex: 2 2 230px; */
    align-self: flex-start;
    align-self: flex-end;
    align-self: center;
}
```

Output:-

****

6. Show implementation of CSS transitions.

The transition-property specifies the CSS properties to which you want the transition effect. Only these CSS properties are animated.

Code:-

```
div {
     width: 100px;
     height: 100px;
     background: lightblue;
     transition-property: width;
```

```
    transition-duration: 2s;
    transition-timing-function: linear;
    transition-delay: 1s;
    }

div: hover{
            width: 300px;
       }
```

****

7. Show implementation of CSS transform.

CSS **transforms** are a collection of *functions* that allow to **shape elements** in particular ways:

- **translate**: moves the element along up to 3 axis (x, y and z)
- **rotate**: moves the element around a central point
- **scale**: resizes the element
- **skew**: distorts the element

Code:-

```
@keyframes translating {
            0% {
                transform: translate (0, 0);
            }

            25% {
                transform: translate (240px, 0);
            }

            50% {
                transform: translate (240px, 140px);
            }

            75% {
                transform: translate (0, 140px);
```

```
        }

        100% {
            transform: translate (0, 0);
        }
    }

    p {
        animation: translating 4s linear infinite;
    }
```

****

8. Show implementation of Grid.

CSS:-

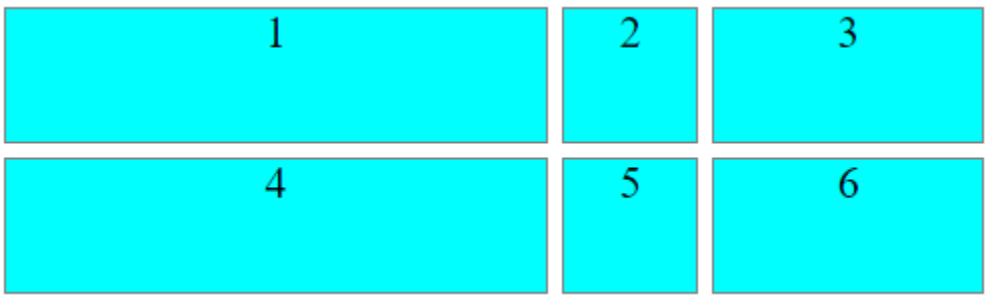
```
.container {
        display: grid;
        grid-template-columns: 200px 50px 100px;
        grid-template-rows: 50px 50px;
        grid-gap: 5px;
        text-align: center;
        justify-content: center;
    }

    .item{
        background-color: aqua;
        border: 1px solid gray;

    }
    .item1 {
        grid-column-start: 1;
        grid-column-end: 3;
    }
```

```
        .item3 {
            grid-row-start: 2;
            grid-row-end: 4;
        }

        .item4 {
            grid-column-start: 2;
            grid-column-end: 4;
        }
```

HTML:-

```
<div class="container">
        <div class="item item-1">1</div>
        <div class="item item-2">2</div>
        <div class="item item-3">3</div>
        <div class="item item-4">4</div>
        <div class="item item-5">5</div>
        <div class="item item-6">6</div>
    </div>
```

Output:-



****

9. Create a masterhead - add a background image and put a text at the center, with a button below it.

HTML:-

```
<div class="card">
        <h1>Hi! I am John</h1>
        <input type="button" value="Submit" class="btn" />
</div>
```

CSS:-

```
.card {
        text-align: center;
        height: 300px;
        background-image:
url("https://s3.ap-south-1.amazonaws.com/www.prepbytes.com/images/homepage/masteh
ead(web).svg");
        background-repeat: no-repeat;
        background-size: cover;
        }


.btn{
        margin: 20px auto;
        width: 200px;
        background-color: bisque;
        height: 25px;
        border-radius: 13px;
        }
```

Output:-


Hi! I am John
Submit
Master Problem Solving
Content Personalisation

****