# CSS

## 1. What is a CSS Preprocessor? What are Sass, Less, and Stylus? Why do people use them?

CSS preprocessors are scripting language that provide additional syntax and features that are not available in normal CSS for example variables , loops , conditional statements

A CSS Preprocessor is a tool used to extend the basic functionality of default vanilla CSS through its own scripting language. It helps us to use complex logical syntax like – variables, functions, mixins, code nesting, and inheritance to name a few, supercharging your vanilla CSS.

SASS: Sass is the acronym for “Syntactically Awesome Style Sheets”. SASS can be written in two different syntaxes using SASS or SCSS

**SASS vs SCSS**

- SASS is based on indentation and SCSS(Sassy CSS) is not.
- SASS uses .sass extension while SCSS uses .scss extension.
- SASS doesn’t use curly brackets or semicolons. SCSS uses it, just like the CSS.

**SASS Syntax**

```
$font-color: #fff
$bg-color: #00f

#box
    color: $font-color
    background: $bg-color
```

**SCSS Syntax**

```
$font-color: #fff;
$bg-color: #00f;
```

```
#box{
color: $font-color;
background: $bg-color;
}
```

**LESS:** LESS is an acronym for “Leaner Stylesheets”. LESS is easy to add to any javascript projects by using NPM or less.js file. It uses the extension .less.

LESS syntax is the same as the SCSS with some exceptions. LESS uses @ to define the variables.

```
@font-color: #fff;
@bg-color: #00f

#box{
color: @font-color;
background: @bg-color;
}
```

**Stylus:** Stylus offers a great deal of flexibility in writing syntax, supports native CSS as well as allows omission of brackets, colons, and semicolons. It doesn’t use @ or $ for defining variables.

```
/* STYLUS SYNTAX WRITTEN LIKE NATIVE CSS */
font-color= #fff;
bg-color = #00f;

#box {
color: font-color;
background: bg-color;
}

/* OR */

/* STYLUS SYNTAX WITHOUT CURLY BRACES */
```

```
font-color= #fff;
bg-color = #00f;

#box
color: font-color;
background: bg-color;
```

## 2. How do you specify units in the CSS? What are the different ways to do it?

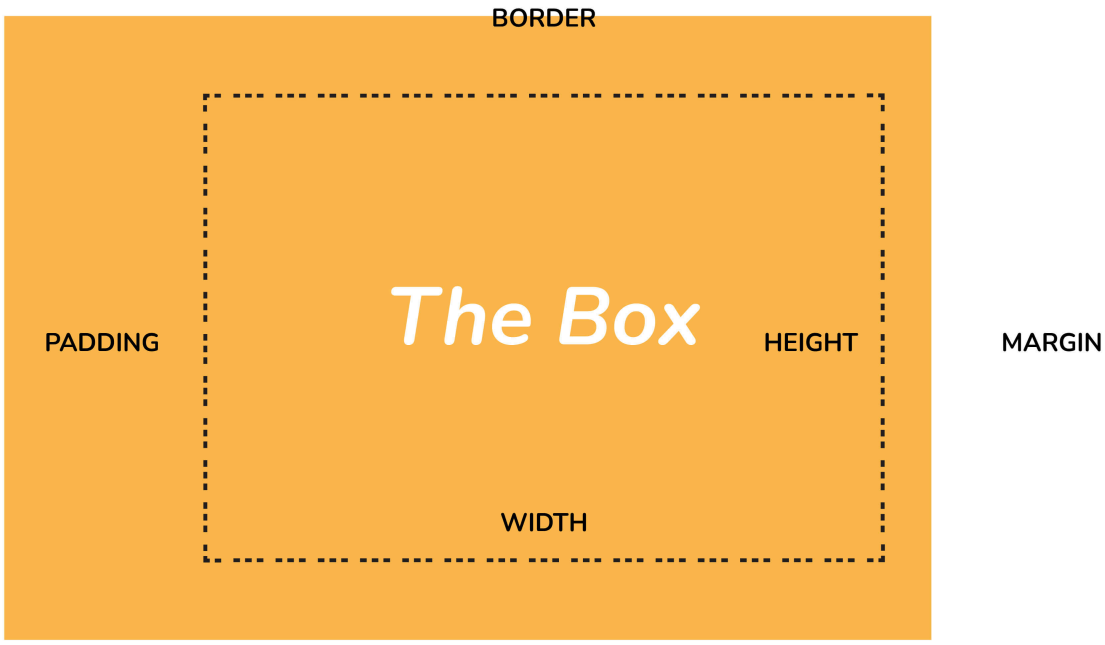
There are different ways to specify units in CSS like px, em, pt, percentage (%). px(Pixel) gives fine-grained control and maintains alignment because 1 px or multiple of 1 px is guaranteed to look sharp. px is not cascade. em maintains relative size. you can have responsive fonts. Em, will cascade 1em is equal to the current font-size of the element or the browser default. If u sent font-size to 16px then 1em = 16px. The common practice is to set default body font-size to 62.5% (equal to 10px).

pt(point) are traditionally used in print. 1pt = 1/72 inch and it is a fixed-size unit.

%(percentage) sets font-size relative to the font size of the body. Hence, you have to set the font-size of the body to a reasonable size.

## 3. What is the Box model in CSS? Which CSS properties are a part of it?

A rectangle box is wrapped around every HTML element. The box model is used to determine the height and width of the rectangular box. The CSS Box consists of Width and height (or in the absence of that, default values and the content inside), padding, borders, margin.




InterviewBit

- **Content:** Actual Content of the box where the text or image is placed.
- **Padding:** Area surrounding the content (Space between the border and content).
- **Border:** Area surrounding the padding.
- **Margin:** Area surrounding the border.

## 4. What is specificity in CSS and how is it calculated?

- Specificity determines which rule takes precedence.
- **Order of specificity**:
  - Inline styles: 1000
  - IDs: 100
  - Classes, attributes, pseudo-classes: 10

    - Elements, pseudo-elements: 1
- More specific rule overrides less specific ones.

## 5.What are the different types of Selectors in CSS?

A CSS selector is the part of a CSS ruleset that actually selects the content you want to style. Different types of selectors are listed below.

**Universal Selector:** The universal selector works like a wildcard character, selecting all elements on a page. In the given example, the provided styles will get applied to all the elements on the page.

```
* {
color: "green";
font-size: 20px;
line-height: 25px;
}
```

**Element Type Selector:** This selector matches one or more HTML elements of the same name. In the given example, the provided styles will get applied to all the ul elements on the page.

```
ul {
  line-style: none;
border: solid 1px #ccc;
}
```

**ID Selector:** This selector matches any HTML element that has an ID attribute with the same value as that of the selector. In the given example, the provided styles will get applied to all the elements having ID as a container on the page.

```
#container {
width: 960px;
margin: 0 auto;
}
```

```
<div id="container"></div>
```

**Class Selector:** The class selector also matches all elements on the page that have their class attribute set to the same value as the class.  In the given example, the provided styles will get applied to all the elements having ID as the box on the page.

```
.box {
padding: 10px;
margin: 10px;
width: 240px;
}

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

**Descendant Combinator:** The descendant selector or, more accurately, the descendant combinator lets you combine two or more selectors so you can be more specific in your selection method.

```
#container .box {
float: left;
padding-bottom: 15px;
}

<div id="container">
   <div class="box"></div>

   <div class="box-2"></div>
</div>

<div class=”box”></div>
```

This declaration block will apply to all elements that have a class of box that is inside an element with an ID of the container. It’s worth noting that

the `.box` element doesn't have to be an immediate child: there could be another element wrapping `.box`, and the styles would still apply.

**Child Combinator:** A selector that uses the child combinator is similar to a selector that uses a descendant combinator, except it only targets immediate child elements.

```
#container> .box {
float: left;
padding-bottom: 15px;
}

<div id="container">
  <div class="box"></div>

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

The selector will match all elements that have a class of `box` and that are immediate children of the `#container` element. That means, unlike the descendant combinator, there can't be another element wrapping `.box` it has to be a direct child element.

**General Sibling Combinator:** A selector that uses a general sibling combinator to match elements based on sibling relationships. The selected elements are beside each other in the HTML.

```
h2 ~p {
margin-bottom: 20px;
}

<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
```

```
<div class="box">
  <p>Paragraph example.</p>
</div>
```

In this example, all paragraph elements (<p>) will be styled with the specified rules, but only if they are siblings of `<h2>` elements. There could be other elements in between the `<h2>` and `<p>`, and the styles would still apply.

**Adjacent Sibling Combinator:** A selector that uses the adjacent sibling combinator uses the plus symbol (+), and is almost the same as the general sibling selector. The difference is that the targeted element must be an immediate sibling, not just a general sibling.

```
p +p {
text-indent: 1.Sem;
margin-bottom: 0;
}

<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>

<div class="box">
  <p>Paragraph example.</p>
  <p>Paragraph example.</p>
</div>
```

The above example will apply the specified styles only to paragraph elements that immediately follow other paragraph elements. This means the first paragraph element on a page would not receive these styles. Also, if another element appeared between two paragraphs, the second paragraph of the two wouldn't have the styles applied.

**Attribute Selector:** The attribute selector targets elements based on the presence and/or value of HTML attributes, and is declared using square brackets.

```
input [type=”text”] {
background-color: #444;
width: 200px;
}

<input type="text">
```

## 6.What are Pseudo elements and Pseudo classes?

**Pseudo-elements** allows us to create items that do not normally exist in the document tree, for example ::after.

- ::before
- ::after
- ::first-letter
- ::first-line
- ::selection

In the below example, the color will appear only on the first line of the paragraph.

```
p: :first-line {
   color: #ffOOOO;
font-variant: small-caps;
}
```

**Pseudo-classes** select regular elements but under certain conditions like when the user is hovering over the link.

- :link
- :visited
- :hover
- :active

- :focus

Example of the pseudo-class, In the below example, the color applies to the anchor tag when it's hovered.

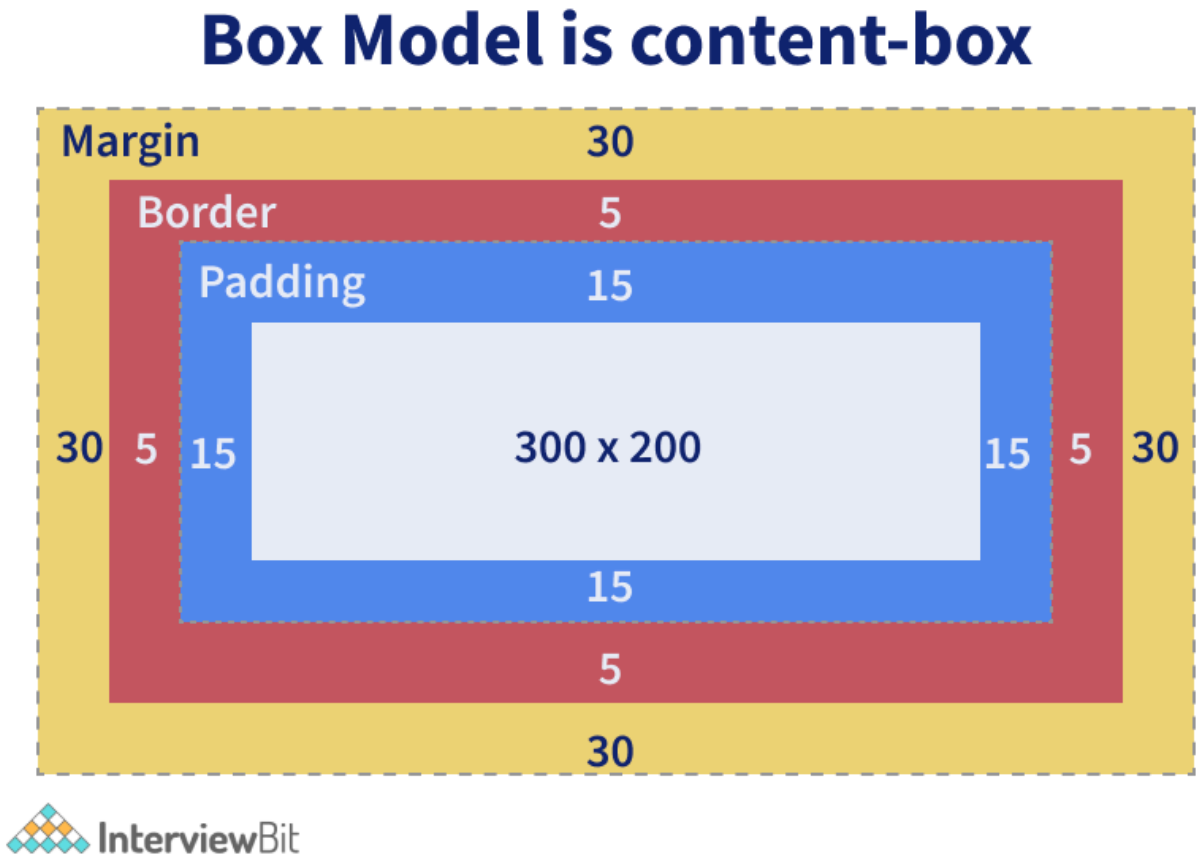
```
/* mouse over link */
a:hover {
color: #FFOOFF;
}
```

## 7. How is border-box different from content-box?

`content-box` is the default value box-sizing property. The height and the width properties consist only of the content by excluding the border and padding. Consider an example as shown:

```
div{
width:300px;
height:200px;
padding:15px;
border: 5px solid grey;
margin:30px;
   -moz-box-sizing:content-box;
   -webkit-box-sizing:content-box;
box-sizing:content-box;
}
```

Here, the box-sizing for the div element is given as content-box. That means, the height and width considered for the div content exclude the padding and border. We will get full height and width parameters specified for the content as shown in the below image.




The `border-box` property includes the content, padding and border in the height and width properties. Consider an example as shown:

```
div{
   width:300px;
   height:200px;
   padding:15px;
   border: 5px solid grey;
   margin:30px;
   -moz-box-sizing:border-box;
   -webkit-box-sizing:border-box;
   box-sizing:border-box;
}
```

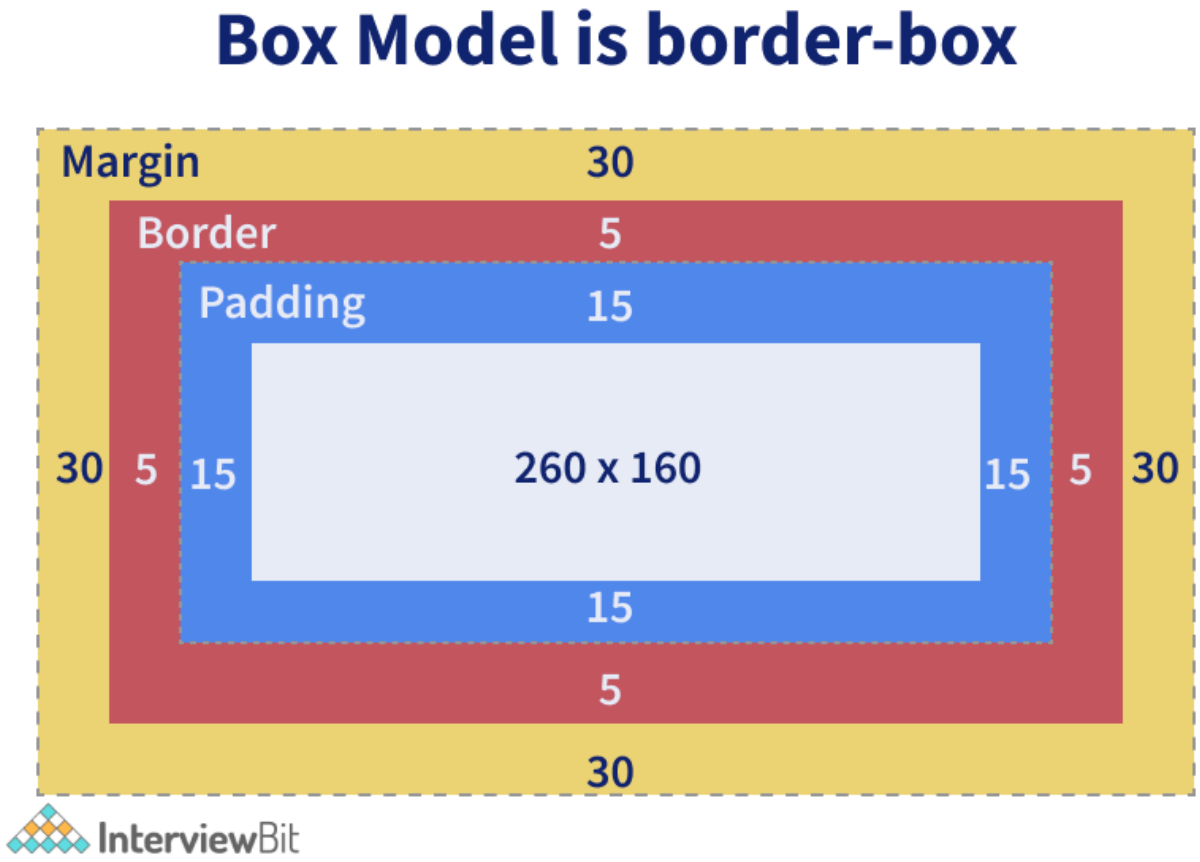
Here, the box-sizing for the div element is given as border-box. That means the height and width considered for the div content will also include the padding and border. This means that the actual height of the div content will be:

```
actual height = height -
                padding on top and bottom -
                border on top and bottom
               = 200 - (15*2) - (5*2)
               = 160 px
```

and the actual width of the div content would be:

```
actual width  = width -
                padding on rightand left -
                border on rightand left
               = 300 - (15*2) - (5*2)
               = 260 px
```

This is represented in the image below:




## 8. What do the following CSS selectors mean?

1. **div, p**
2. **div p**
3. **div ~ p**
4. **div + p**
5. **div > p**

The meaning of the given list of selectors goes as follows:

- `div, p`: This selector implies selecting all div elements and all p elements.

Consider an example below:

```
<h1>Heading 1</h1>
  <div>
     Division 1
     <p> paragraph 1</p> <!-- Will be selected -->
  </div>
  <p> paragraph 2</p>
  <p> paragraph 3</p>
  <div>
     Division 2
  </div>
  <span> Span 1 </span>
```

Here, **all** the div elements and the p elements would be selected by the browser irrespective of their parents or where they are placed. The remaining tags like h1 and span are ignored.

- `div p` : This selector tells to select all p elements that are inside div elements. Consider an example below:

```
  <h1>Heading 1</h1>
<div>
   Division 1
   <p> paragraph 1</p> <!-- Will be selected -->
   <div>
      <p> Inner Div Paragraph </p> <!-- Will be selected -->
   </div>
</div>
<p> paragraph 2</p>
<p> paragraph 3</p>
<div>
   Division 2
</div>
<span> Span 1 </span>
```

Here, `<p> paragraph 1</p>` and `<p> Inner Div Paragraph </p>` would be selected by the browser and the properties are applied. The rest of the paragraph tags are not selected.

- `div ~ p` : This selector tells to select all p elements that have div elements preceeded anywhere. Consider an example,

```
<h1>Heading 1</h1>
<div>
  Division 1
  <p> paragraph 1</p>
</div>
<p> paragraph 2</p> <!-- Will be selected -->
<p> paragraph 3</p> <!-- Will be selected -->
<div>
  Division 2
</div>
<span> Span 1 </span>
```

Here, paragraph 2 and paragraph 3 elements would be selected as marked in the code above.

- `div + p` : This selector says to select all p elements placed immediately after the div element. Consider an example in this case:

```
 <h1>Heading 1</h1>
<div>
   Division 1
   <p> paragraph 1</p>
</div>
<p> paragraph 2</p> <!-- Will be selected -->
<p> paragraph 3</p>
<div>
   Division 2
</div>
<span> Span 1 </span>
```

In this case, we have paragraph 2 element immediately after the div tag. Hence, only that element will be selected.

- `div > p` : This selector says to select all p elements which has div as an immediate parent. In the same example below:

```
 <h1>Heading 1</h1>
<div>
   Division 1
   <p> paragraph 1</p> <!-- Will be selected -->
</div>
<p> paragraph 2</p>
<p> paragraph 3</p>
<div>
   Division 2
</div>
<span> Span 1 </span>
```

Only `<p> paragraph 1</p>` will be selected in this case because it has immediate div as the parent.

## 9. What are the properties of flexbox?

Flexbox stands for flexible box and it was introduced around 2017 in CSS with the purpose of providing an efficient way to handle layouts, align elements within them and distribute spaces amongst the items in dynamic/responsive conditions. It provides an enhanced ability to alter the dimensions of the items and make use of the available space in the container efficiently. In order to achieve this, CSS3 provides some properties.

The properties of flexbox are as follows:

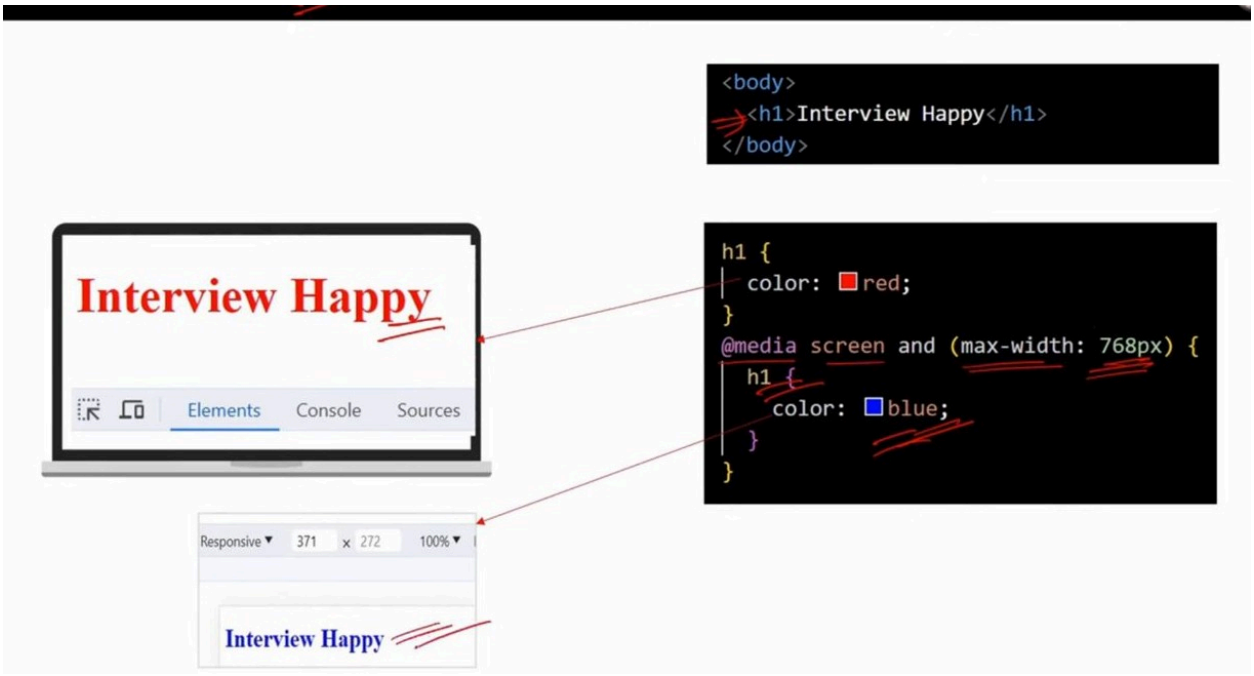
- **flex-direction**: This property helps in defining the direction the container should stack the items targetted for flex. The values of this property can be
  - row: Stacks items horizontally from left to right in the flex container.
  - column: Stacks items vertically from top to bottom in the flex container.

    - row-reverse: Stacks items horizontally from right to left in the flex container.
    - column-reverse: Stacks items vertically from bottom to top in the flex container.
- **flex-wrap**: This property specifies of the flex items should be wrapped or not. Possible values are:
    - wrap: The flex items would be wrapped if needed.
    - nowrap: This is the default value that says the items won’t be wrapped.
    - wrap-reverse: This specifies that the items will be wrapped if needed but in reverse order.
- **flex-flow**: This property is used for setting both flex-direction and flex-wrap properties in one statement.
- **justify-content**: Used for aligning the flex items. Possible values are:
    - center: It means that all the flex items are present at the center of the container.
    - flex-start: This value states that the items are aligned at the start of the container. This is the default value.
    - flex-end: This value ensures the items are aligned at the end of the container.
    - space-around: This value displays the items having space between, before, around the items.
    - space-between: This value displays items with spaces between the lines.
- **align-items**: This is used for aligning flex items.
- **align-content**: This is used for aligning the flex lines.

## 10. What are media queries?

- Allow CSS to adapt based on screen size or device:

```
@media (max-width: 768px) {
  body {
    background-color: lightblue;
  }
}
```




## 11.How to include CSS in the webpage?

There are different ways to include a CSS in a webpage,

1 - External Style Sheet: An external file linked to your HTML document: Using link tag, we can link the style sheet to the HTML page.

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

2 - Embed CSS with a style tag: A set of CSS styles included within your HTML page.

```
<style type="text/css">
```

```
/*Add style rules here*/

</style>
```

Add your CSS rules between the opening and closing style tags and write your CSS exactly the same way as you do in stand-alone stylesheet files.

3 - Add inline styles to HTML elements(CSS rules applied directly within an HTML tag.): Style can be added directly to the HTML element using a style tag.

```
<h2 style="color:red;background:black">Inline Style</h2>
```

4 - Import a stylesheet file (An external file imported into another CSS file): Another way to add CSS is by using the @import rule. This is to add a new CSS file within CSS itself.

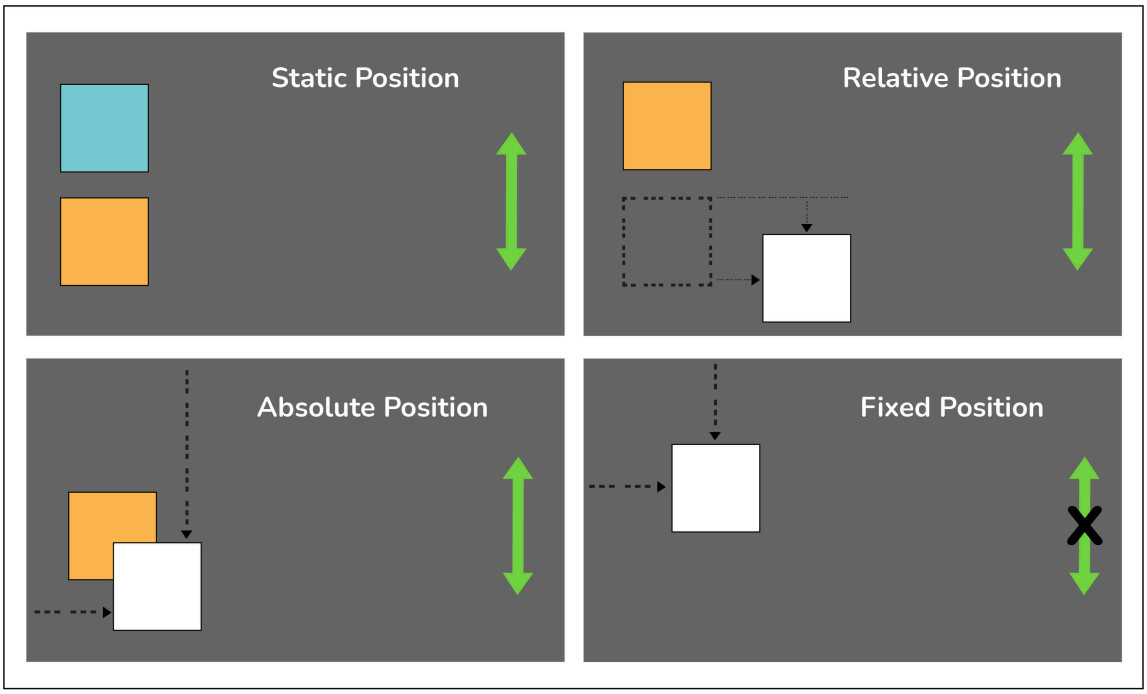
```
@import "path/to/style.css";
```

## 12.Explain CSS position property?

- **Absolute:** To place an element exactly where you want to place it. absolute position is actually set relative to the element's parent. if no parent is available then the relative place to the page itself (it will default all the way back up to the element).
- **Relative:** "Relative to itself". Setting position: relative; on an element and no other positioning attributes, it will no effect on its positioning. It allows the use of z-index on the element and it limits the scope of absolutely positioned child elements. Any child element will be absolutely positioned within that block.
- **Fixed:** The element is positioned relative to the viewport or the browser window itself. viewport doesn't change if you scroll and hence the fixed element will stay right in the same position.
- **Static:** Static default for every single page element. The only reason you would ever set an element to position: static is to forcefully remove some

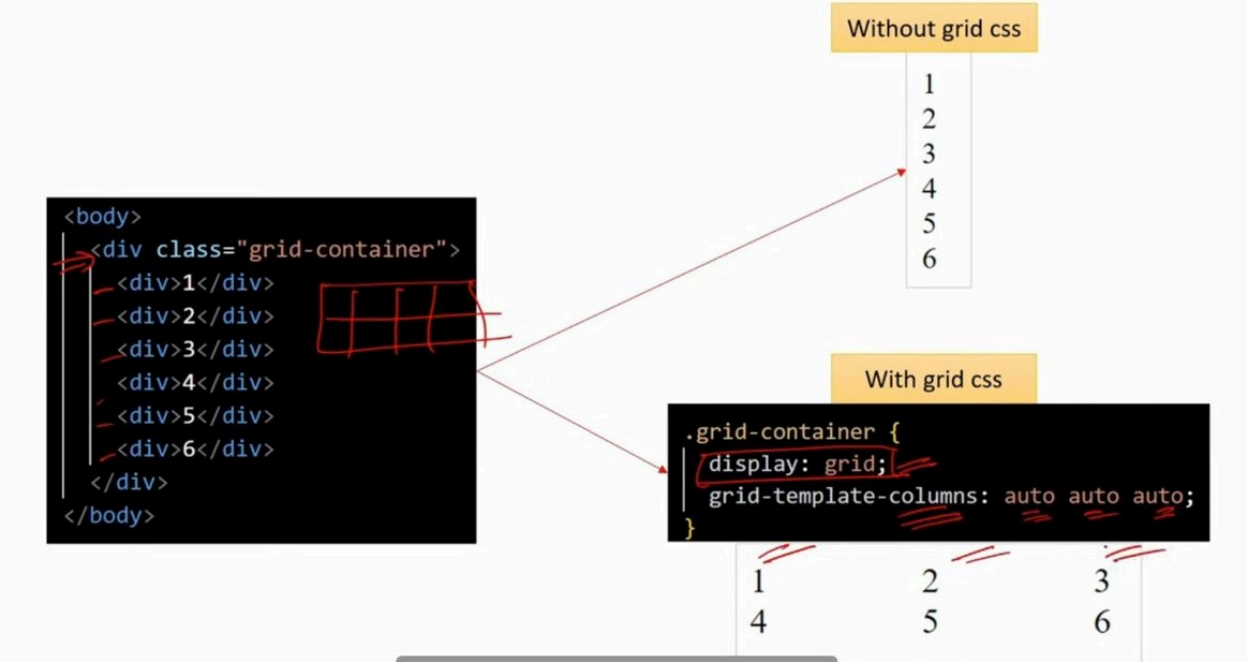
positioning that got applied to an element outside of your control.

- **Sticky:** Sticky positioning is a hybrid of relative and fixed positioning. The element is treated as relative positioned until it crosses a specified threshold, at which point it is treated as fixed positioned.




## 13.What is the grid system?

CSS Grid Layout is the most powerful layout system available in CSS. It is said to be a 2-dimensional system, meaning it can handle both columns and rows, unlike flexbox which is largely a 1-dimensional system.



## 14. Center a Div

> Task: Center a 200×200 div in the middle of the page both vertically and horizontally.

**Concepts Tested**: Flexbox or CSS Grid, `position`, `margin`

- **Centering with Table:**

**HTML:**

```
<div class="cn"><div class="inner">your content</div></div>
```

**CSS:**

```
.cn {
display: table-cell;
width: 500px;
height: 500px;
vertical-align: middle;
text-align: center;
}

.inner {
display: inline-block;
width: 200px;height: 200px;
}
```

- **Centering with Transform**

## Syntax

```
css
CopyEdit
```

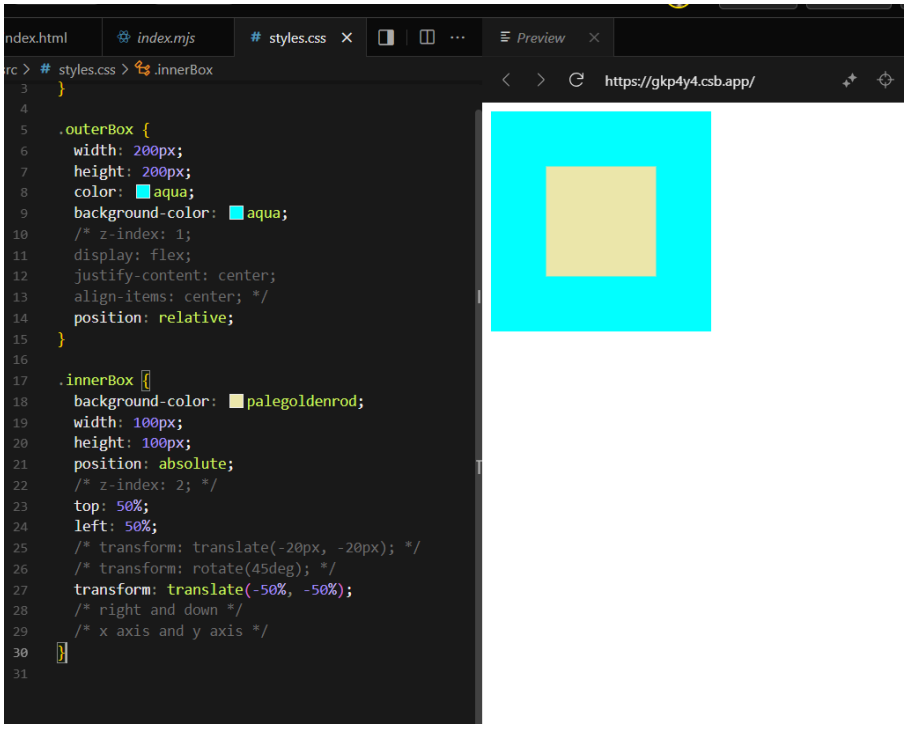
```
transform: translate(x, y);
```

- `x`: how much to move horizontally (right is positive, left is negative).
- `y`: how much to move vertically (down is positive, up is negative).

**HTML:**

```
<div class="cn"><div class="inner">your content</div></div>
```

**CSS:**




```
.cn {
position: relative;
width: 500px;
height: 500px;
}
```

```
.inner {
position: absolute;
top: 50%;left: 50%;
transform: translate(-50%,-50%);
width: 200px;
height: 200px;
}
```

- **Centering with Flexbox**

**HTML:**

```
<div class="cn"><div class="inner">your content</div></div>
```

**CSS:**

```
.cn {
display: flex;
justify-content: center;
align-items: center;
}
```

- **Centering with Grid**

**HTML:**

```
<div class=”wrap_grid”>
   <div id=”container”>vertical aligned text<br />some more text here
   </div>
</div>
```

**CSS:**

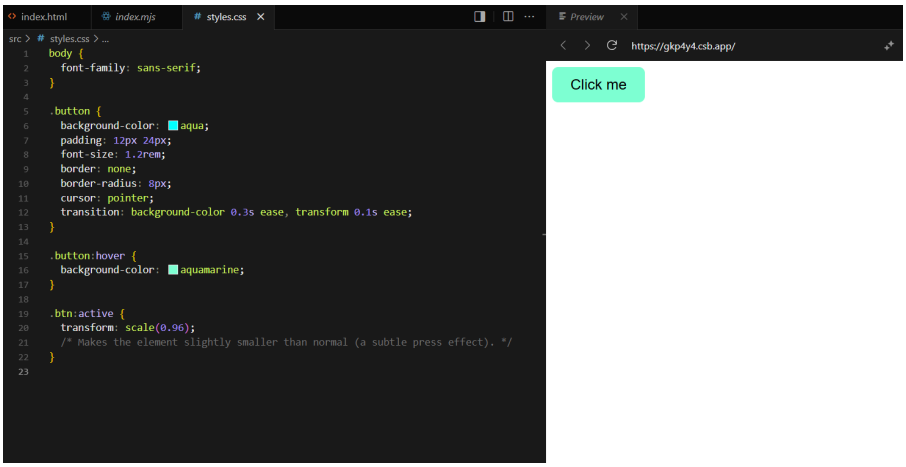
```
.wrap-grid {
display: grid;
```

```
  place-content: center;
}
```

## 15. Responsive Button with Hover & Active States

> Task: Create a button that changes color on hover and scales slightly when clicked.

**Concepts Tested**: Pseudo-classes ( `:hover` , `:active` ), `transform` , `transition` .




```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript Sandbox</title>
    <meta charset="UTF-8" />
  </head>

  <body>
    <div id="app">
      <button class="button">Click me</button>
    </div>
```

```
    <script src="./index.mjs" type="module"></script>
  </body>
</html>
```

```
body {
  font-family: sans-serif;
}

.button {
  background-color: aqua;
  padding: 12px 24px;
  font-size: 1.2rem;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: background-color 0.3s ease, transform 0.1s ease;
}

.button:hover {
  background-color: aquamarine;
}

.btn:active {
  transform: scale(0.96);
  /* Makes the element slightly smaller than normal (a subtle press effect). */
}
```

## Medium Level

## 16. Pure CSS Dropdown Menu

> Task: Create a dropdown menu that appears on hover.

**Concepts Tested**: Nested selectors, `:hover`, `display`, `position`.

```
<!DOCTYPE html>
<html>
 <head>
  <title>JavaScript Sandbox</title>
  <meta charset="UTF-8" />
 </head>

 <body>
  <div id="app">
   <div class="navbar">
    <div class="nav-item">
     <a href="#" class="nav-link">Menu</a>
     <div class="dropdown">
      <a href="#">Option 1</a>
      <a href="#">Option 2</a>
      <a href="#">Option 3</a>
     </div>
    </div>
   </div>
  </div>

  <script src="./index.mjs" type="module"></script>
 </body>
</html>
```

```
body {
 font-family: sans-serif;
}

.navbar {
 background-color: #333;
 padding: 10px;
}
```

```css
.nav-item {
  position: relative;
  display: inline-block;
}

.nav-link {
  color: white;
  padding: 12px 16px;
  text-decoration: none;
  display: inline-block;
}

.nav-link:hover {
  background-color: #444;
}

.dropdown {
  display: none;
  position: absolute;
  background-color: white;
  min-width: 160px;
  box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
  z-index: 1;
}

/* Show dropdown on hover */
.nav-item:hover .dropdown {
  display: block;
}

.dropdown a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
```

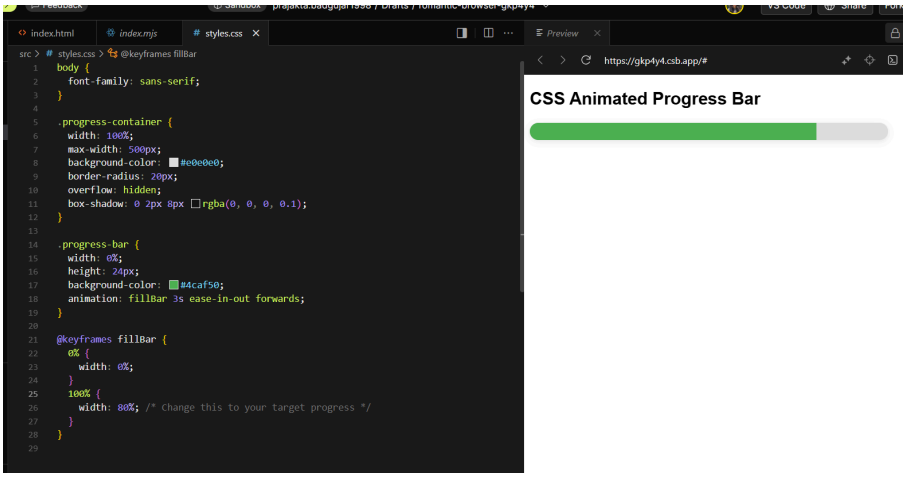
```
}

.dropdown a:hover {
  background-color: #f1f1f1;
}
```

## 17. Progress Bar Animation

> Task: Create a progress bar that fills up using CSS animation.

**Concepts Tested**: `@keyframes`, `width`, `transition`, animation timing.




```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript Sandbox</title>
    <meta charset="UTF-8" />
  </head>

  <body>
    <div id="app">
      <h2>CSS Animated Progress Bar</h2>
```

```
    <div class="progress-container">
      <div class="progress-bar"></div>
    </div>
  </div>

  <script src="./index.mjs" type="module"></script>
 </body>
</html>
```

```
body {
 font-family: sans-serif;
}

.progress-container {
 width: 100%;
 max-width: 500px;
 background-color: #e0e0e0;
 border-radius: 20px;
 overflow: hidden;
 box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

.progress-bar {
 width: 0%;
 height: 24px;
 background-color: #4caf50;
 animation: fillBar 3s ease-in-out forwards;
}

@keyframes fillBar {
 0% {
  width: 0%;
 }
 100% {
  width: 80%; /* Change this to your target progress */
```

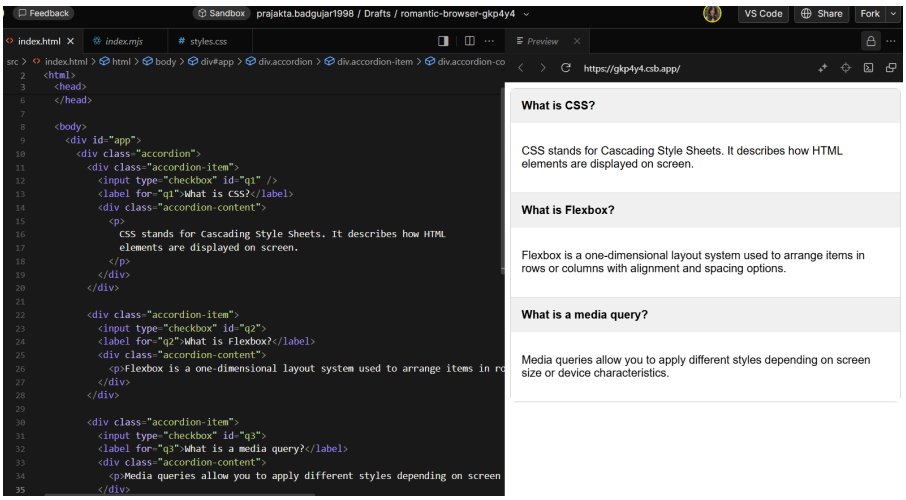
```
  }
}
```

## 18. Accordion Using Pure CSS

> Task: Create a collapsible FAQ section using only CSS.

**Concepts Tested:** `:checked` pseudo-class, hidden inputs, sibling selectors, transitions.

Solution




```
<!DOCTYPE html>
<html>
 <head>
  <title>JavaScript Sandbox</title>
  <meta charset="UTF-8" />
 </head>

 <body>
  <div id="app">
   <div class="accordion">
    <div class="accordion-item">
     <input type="checkbox" id="q1" />
```

```
        <label for="q1">What is CSS?</label>
        <div class="accordion-content">
          <p>
            CSS stands for Cascading Style Sheets. It describes how HTML
            elements are displayed on screen.
          </p>
        </div>
      </div>

      <div class="accordion-item">
        <input type="checkbox" id="q2">
        <label for="q2">What is Flexbox?</label>
        <div class="accordion-content">
          <p>Flexbox is a one-dimensional layout system used to arrange items in r
        </div>
      </div>

      <div class="accordion-item">
        <input type="checkbox" id="q3">
        <label for="q3">What is a media query?</label>
        <div class="accordion-content">
          <p>Media queries allow you to apply different styles depending on scree
        </div>
      </div>
    </div>

    <script src="./index.mjs" type="module"></script>
  </body>
</html>
```

```
body {
  font-family: sans-serif;
}

.accordion {
```

```
  max-width: 600px;
  margin: 0 auto;
  border: 1px solid #ccc;
  border-radius: 8px;
}

.accordion-item {
  border-bottom: 1px solid #ddd;
}

.accordion-item:last-child {
  border-bottom: none;
}

.accordion input {
  display: none;
}

.accordion label {
  display: block;
  padding: 15px;
  cursor: pointer;
  background: #f0f0f0;
  font-weight: bold;
  transition: background 0.3s ease;
}

.accordion label:hover {
  background: #e0e0e0;
}

.accordion-content {
  max-height: 0;
  overflow: hidden;
  background: #fff;
  transition: max-height 0.4s ease, padding 0.3s ease;
```

```
  padding: 0 15px;
}

input:checked ~ .accordion-content {
  max-height: 200px; /* You can adjust based on expected content */
  padding: 15px;
}
```

## 19. Image Gallery with Responsive Columns

> Task: Create a responsive image grid with 3 columns on desktop, 2 on tablet, 1 on mobile.

**Concepts Tested**: Flexbox/Grid, media queries, image scaling.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Responsive Image Gallery</title>
  <style>
    body {
      margin: 0;
      font-family: sans-serif;
      padding: 20px;
      background-color: #f9f9f9;
    }

    .gallery {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      gap: 15px;
    }
```

```
  .gallery img {
    width: 100%;
    height: auto;
    display: block;
    border-radius: 10px;
    object-fit: cover;
    box-shadow: 0 2px 6px rgba(0,0,0,0.1);
  }

  /* Tablet: 2 columns */
  @media (max-width: 900px) {
    .gallery {
      grid-template-columns: repeat(2, 1fr);
    }
  }

  /* Mobile: 1 column */
  @media (max-width: 600px) {
    .gallery {
      grid-template-columns: 1fr;
    }
  }
 </style>
</head>
<body>

 <h2>Responsive Image Gallery</h2>

 <div class="gallery">
   <img src="https://source.unsplash.com/random/400×300?sig=1" alt="Image 1
   <img src="https://source.unsplash.com/random/400×300?sig=2" alt="Image
   <img src="https://source.unsplash.com/random/400×300?sig=3" alt="Image
   <img src="https://source.unsplash.com/random/400×300?sig=4" alt="Image
   <img src="https://source.unsplash.com/random/400×300?sig=5" alt="Image
   <img src="https://source.unsplash.com/random/400×300?sig=6" alt="Image
```

```
  </div>

</body>
</html>
```