Module 3 – Frontend – CSS and CSS3

**CSS Selectors & Styling**

**Question 1**: What is a CSS selector? Provide examples of element, class, and ID selectors.

Answer:

* A CSS selector is a pattern used to target specific HTML elements and apply styles to them. It tells the browser which elements to select and style.
* **Element Selector** - An element selector in CSS targets specific HTML elements based on their tag name.
* For example:

h1 {

color: blue;

font-size: 36px;

}

- h1 targets all <h1> elements

- p targets all <p> elements

* **Class Selector** - A class selector in CSS targets elements with a specific class attribute. It's denoted by a dot (.) followed by the class name.
* For Example:

.button {

background-color: #4CAF50;

color: #fff;

padding: 10px 20px;

}

* To use this class selector, you'd add class="button" to the HTML element:
* <button class="button">Click me!</button>
* Class selectors are reusable and can be combined with other classes or selectors. You can also apply multiple classes to an element by separating them with spaces:
* <button class="button primary">Click me!</button>
* **ID Selectors** - ID selectors in CSS target a specific HTML element with a unique ID attribute. They're denoted by a hash (#) followed by the ID name.
* For Example:

#logo {

width: 100px;

height: 50px;

background-color: #333;

}

* To use this ID selector, you'd add id="logo" to the HTML element:
* <div id="logo"></div>
* ID selectors are unique and should only be applied to one element per HTML document.
* ID selectors have higher specificity than class selectors.
* When to use:
* 1. **Unique elements:** Use IDs for elements that appear only once, like a logo or main navigation.
* 2. **Targeting specific elements**: Use IDs when you need to target a specific element with JavaScript or CSS.

**Question 2**: Explain the concept of CSS specificity. How do conflicts between multiple styles get resolved?

Answer:

* CSS Specificity determines which styles are applied when multiple selectors target the same element. It's calculated based on the types of selectors used.
* It is the way which helps the browsers to decide which property value is most relevant for the element.
* The style property has a greater specificity value compare to the selector (except then !important in the stylesheet selector)
* **Specificity Hierarchy:**
* 1. Inline styles (highest): Styles applied directly to an element using the style attribute.
* 2. ID selectors: Selectors using # (e.g., #logo).
* 3. Class selectors, attribute selectors, and pseudo-classes: Selectors using. (e.g., button), or: (e.g., hover).
* 4. Element selectors (lowest): Selectors using element names (e.g., h1, p).
* **Resolving Conflicts** - When multiple styles conflict, the browser applies the styles with the highest specificity. If specificity is equal, the last defined style wins.
* For Example:

/ Element selector (low specificity) /

h1 {

color: blue;

}

/ Class selector (medium specificity) /

.title {

color: red;

}

/ ID selector (high specificity) /

#main-title {

color: green;

}

* <h1 id="main-title" class="title">Hello</h1>
* In this case, the h1 element would be green, because the ID selector #main-title has the highest specificity.

**Question 3**: What is the difference between internal, external, and inline CSS? Discuss the advantages and disadvantages of each approach.

Answer:

* 1. **Internal CSS**: CSS code is written within the HTML document, inside the `<style>` tag, typically in the `<head>` section.
* 2. **External CSS**: CSS code is written in a separate file with a `.css` extension and linked to the HTML document using the `<link>` tag.
* 3. **Inline CSS**: CSS code is written directly within an HTML element using the `style` attribute
* **Internal CSS Advantages:**

- Easy to implement for small projects

- No need to create a separate file

* **Internal CSS Disadvantages:**

- CSS code is mixed with HTML, making it harder to maintain

- Styles are only applied to the current document

* **External CSS Advantages:**

- Separates CSS from HTML, making code organization and maintenance easier

- Styles can be applied to multiple HTML documents

- Easier to manage large projects

* **External CSS Disadvantages:**

- Requires an additional HTTP request to load the CSS file

- Can be overkill for small projects

* **Inline CSS Advantages:**

- Quick and easy to apply styles to a single element

- Useful for dynamic styling using JavaScript

* **Inline CSS Disadvantages:**

- Mixes CSS with HTML, making it harder to maintain

- Styles are only applied to the specific element, not reusable

**CSS Box Model**

**Question 1**: Explain the CSS box model and its components (content, padding, border, margin). How does each affect the size of an element?

Answer:

* The CSS box model is a fundamental concept in web development that describes the structure and layout of HTML elements.
* It consists of four main components:
* 1**. Content Area**: The innermost part of the box, containing the actual content.

- Width and height are defined by the content itself or by setting width and height properties.

* 2. **Padding**: The space between the content area and the border. Padding is used to add space around the content.

- Defined using padding properties (e.g., padding: 10px).

- Adds space between the content area and the border.

* 3. **Border**: The visible outline of the box. Borders can be styled with different widths, styles, and colors.

- Defined using border properties (e.g., border: 1px solid black).

- Can be styled with different widths, styles, and colors.

* 4. **Margin**: The space between the box and other elements. Margins are used to add space between elements.

- Defined using margin properties (e.g., margin: 20px).

- Adds space between the box and other elements.

* **How they affect the size:**

- The width and height properties, when applied, primarily affect the content box.

- Padding and border add to the content size to determine the overall width and height of the rendered element.

- Setting box-sizing: border-box includes padding and border within the specified width and height, making the element size more predictable.

**Question 2**: What is the difference between border-box and content-box box-sizing in CSS? Which is the default?

Answer:

* Content-box is default property of box-sizing
* **Box-sizing** - border-box includes padding and border in the width and height calculation.
* The width and height properties include the content area, padding, and border. This makes it easier to predict the size of an element.
* **Content-box -** box-sizing: content-box is the default value. width and height only apply to the content area.
* Padding and border are added to the width and height, increasing the overall size of the element.
* **Example:**

.element {

width: 100px;

padding: 20px;

border: 5px solid black;

}

* With box-sizing: content-box (default), the total width would be:
* 100px (content) + 20px (padding-left) + 20px (padding-right) + 5px (border-left) + 5px (border-right) = 150px.
* With box-sizing: border-box, the total width would be:
* 100px (includes content, padding, and border).
* Using box-sizing: border-box can simplify your layout calculations and make your CSS more intuitive. Many developers use this value as a global default:

\* {

box-sizing: border-box;

}

**CSS Flexbox**

**Question 1**: What is CSS Flexbox, and how is it useful for layout design? Explain the terms

flex-container and flex-item.

Answer:

* Flexbox is short for the Flexible Box Layout module. Flexbox is a layout method for arranging items in rows or columns.
* Flexbox makes it easier to design a flexible responsive layout structure, without using float or positioning.
* Flexbox simplifies the alignment and distribution of space among item within a container, regardless of their size or order.
* **One-dimensional layouts**: Flexbox is ideal for layouts that require elements to be arranged in a single row or column.
* **Dynamic content**: Flexbox makes it easy to handle dynamic content, such as varying text lengths or image sizes.
* **Responsive design**: Flexbox helps create responsive layouts that adapt to different screen sizes and devices.
* **Key Concepts:**

1**. Flex Container:** The parent element that wraps around the flex items. To create a flex container, you set display: flex or display: inline-flex on the element. The flex container controls the layout of its child elements.

2**. Flex Items:** 2. The child elements of the flex container. These are the elements that are directly nested inside the flex container. Flex items can be any HTML element, such as divs, spans, images, or paragraphs.

* **Example**:

<div class="flex-container">

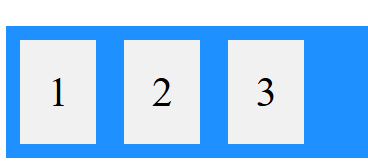
<div class="flex-item">Item 1</div>

<div class="flex-item">Item 2</div>

<div class="flex-item">Item 3</div>

</div>

* **Output**:



**Question 2**: Describe the properties justify-content, align-items, and flex- direction used in Flexbox.

Answer:

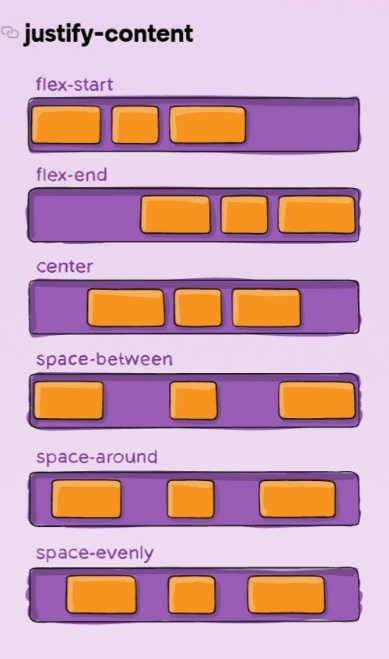
* **justify-content**:

- This defines the alignment along main axis. It helps distribute extra free space leftover when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size.

- It also exerts some control over the alignment of item when they overflow the line.

- The justify-content property accepts five different values:

* flex-start
* flex-end
* center
* space-between
* space-around
* space-evenly



* **align-item:**
* **-** This defines the default behavior for how flex items are laid out along the cross axis on the current line.
* In a flexbox container, the flexbox items are aligned on the cross axis, which is vertical by default.
* The align-item property accepts different values:

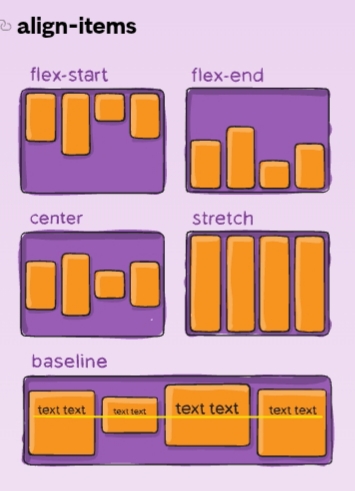
- stretch

- center

- flex-start

- flex-end

- baseline



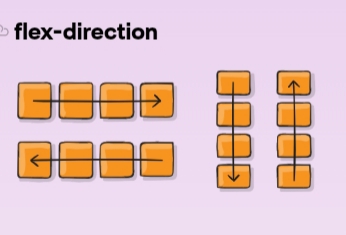
* **flex-direction:**
* This establishes the main-axis, thus defining the direction flex-items are placed in the flex container.
* Flexbox is a single-direction layout concept.
* The align-item property accepts different values:

- row

- column

- row-reverse

- column-reverse



**CSS Grid**

* **Question 1**: Explain CSS Grid and how it differs from Flexbox. When would you use Grid over Flexbox?
* Answer:
* The grid layout module offers a grid-based layout system, with rows and columns. It allows developers to easily create complex web layouts.
* Grids can be used to layout major page areas or small user interface elements. Elements can be placed onto the grid within these column and row lines.
* Grid layout is a two-dimensional grid-based layout system with rows and columns. It makes easier to design web pages without having to use floats and positioning.
* A grid always consists of:
* Grid container – the parent <div> element
* Grid item – the item inside the container <div>
* A grid layout consists of a parent element, with one or more grid items.
* **Display grid property:**

.conatiner{

display: grid;

}

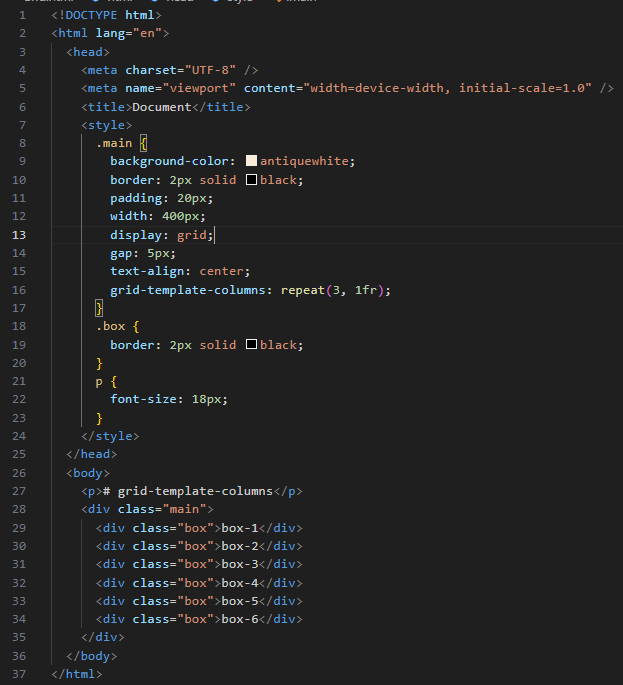


* **Grid Properties:**
* align-content
* align-item
* gap
* grid-template-columns
* grid-template-rows
* grid-template-areas

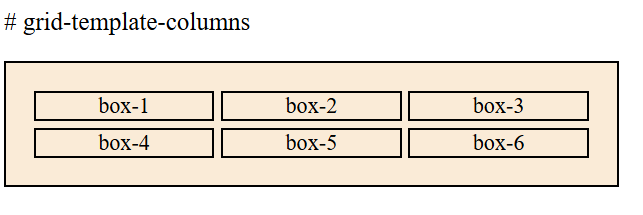
**Question 2**: Describe the grid-template-columns, grid-template-rows, and grid- gap properties. Provide examples of how to use them.

Answer:

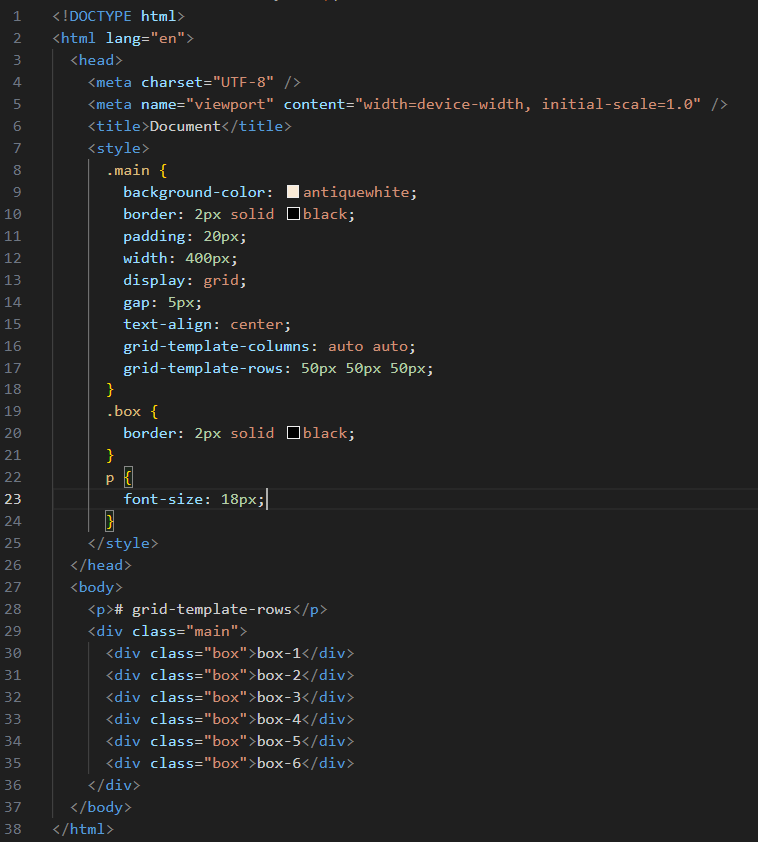
* **grid-template-columns** - This property specifies the number of columns in a grid layout. The values are space separated list, where each values specifies the size of the respective column.



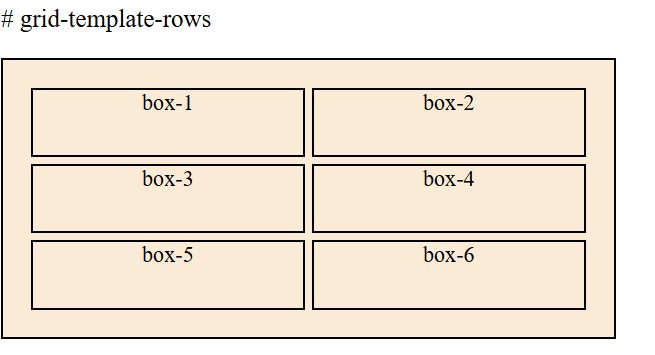
**Output:**



* **grid-template-rows –** This property defines the number and height of rows in a grid layout. It accepts values like none, auto, max-content, min-content, and specific length, allowing for flexible and responsive design.

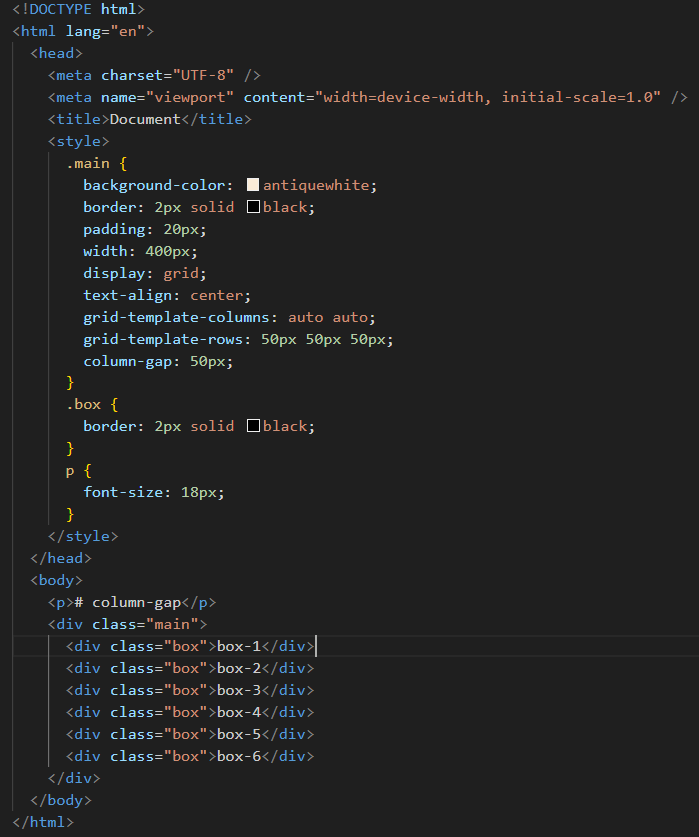


**Output:**

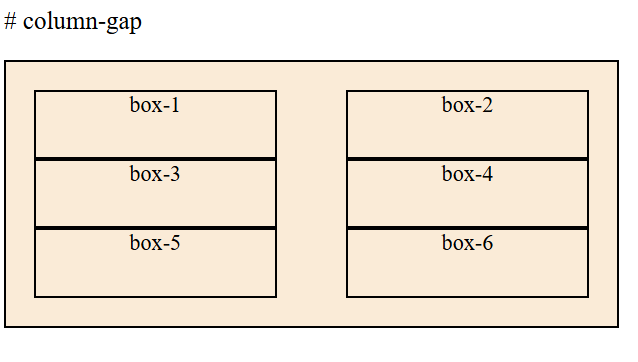
****

* **grid- gap** – It sets the size of the gap between rows and columns in a grid layout, allowing you to easily control the spacing between grid items in both horizontal and vertical directions.

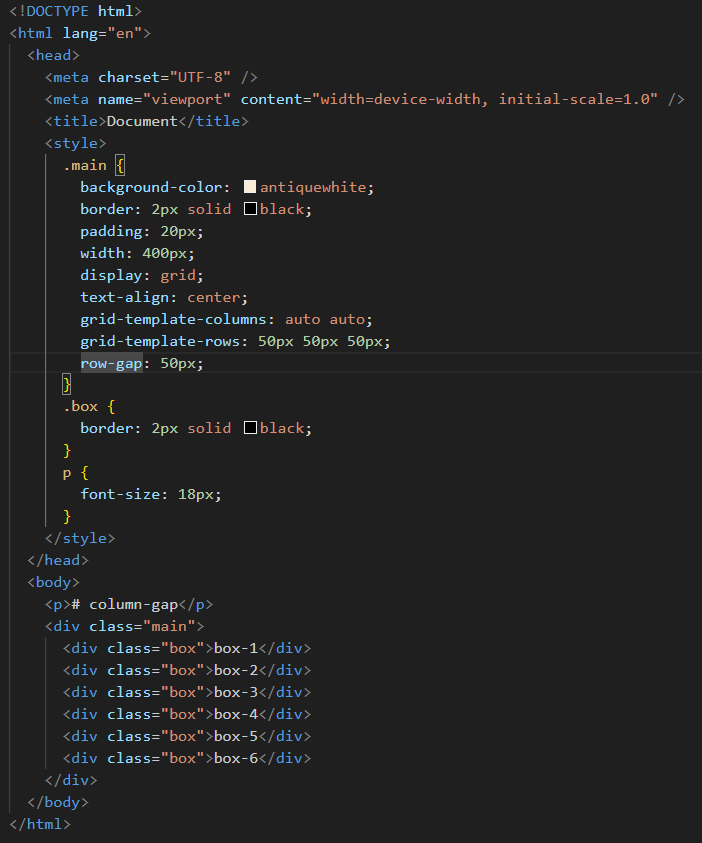
- **Column gap**: It is used to set the size of the gap between



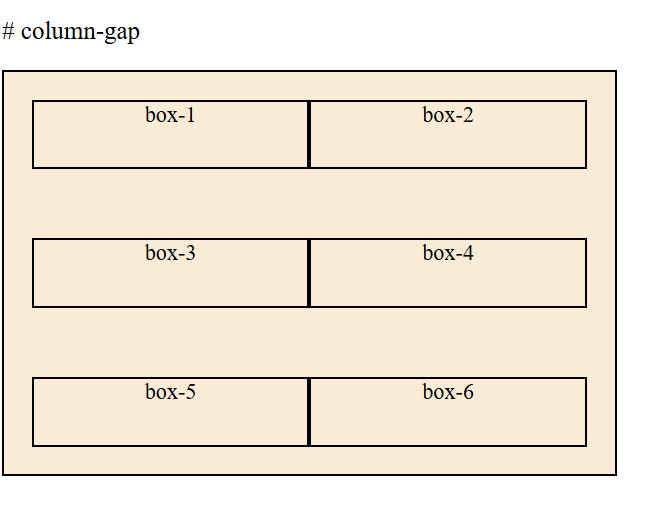
**Output:**

****

* **Row-gap:**

****

**Output:**

****