Introduction to CSS
CSS stands for Cascading Style Sheets. It is primarily used to provide styling to web pages, including color, layout, background, font, and border properties. The main objective of CSS is to improve content accessibility, provide enhanced flexibility and control, and specify presentation characteristics.

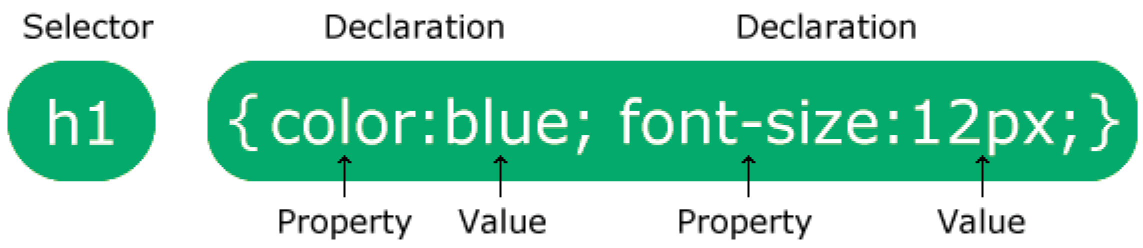
CSS3 stands for Cascading Style Sheets Level 3. It is an advanced version of CSS, used for structuring, styling, and formatting web pages. CSS3 introduces several new features and is supported by all modern web browsers. One of the most significant advancements in CSS3 is the splitting of CSS standards into separate modules, making it simpler to learn and use.

New Features of CSS3

- Combinator: CSS3 introduces a new general sibling combinator, which matches sibling elements using the tilde (~) combinator.
- CSS Selectors: CSS3 selectors are more advanced than the simple selectors offered by CSS, providing a sequence of easy-to-use and simple selectors.
- Pseudo-elements: CSS3 adds many new pseudo-elements for easier and more detailed styling. The new convention of double colons (::) is also introduced.
- Border Style: CSS3 includes new border styling features like border-radius, border-image-slice, border-image-source, and values for "width stretch".
- Background Style Properties: CSS3 introduces new background style properties such as background-clip, background-size, background-style, and background-origin.

CSS Syntax
CSS uses a set of syntax rules to define the styling of web pages. These rules consist of a selector and a declaration block. The selector is used to select the HTML element that you want to style, while the declaration block contains the properties and values that you want to apply to that element.

Here's is the syntax:



For example,

```
h1 {
  font-size: 24px;
```

```
 color: #333;
}
```

Here, h1 is the selector, and the declaration block contains the font-size and color properties with their respective values.

CSS Properties

Some of the most popularly used CSS properties are color, font-family, font-size, background-color, margin, padding, border, and text-align. These properties are fundamental for controlling the visual appearance of web pages, including text styling, element spacing, and border design.
Here's a more detailed look at each:

- color: Sets the text color of an element.
- font-family: Defines the font family to be used for the text.
- font-size: Controls the size of the text.
- background-color: Sets the background color of an element.
- margin: Defines the space between an element and its adjacent elements.
- padding: Defines the space between an element's content and its border.
- border: Creates a border around an element.
- text-align: Controls the horizontal alignment of text within an element.
- display: sets the display type of an element

CSS Selectors

CSS selectors are used to select the HTML elements that you want to style. There are several types of selectors:

- Element selector: selects all elements of a specific type (e.g. h1)
- Class selector: selects elements with a specific class name (e.g. .my-class)
- ID selector: selects elements with a specific ID (e.g. #my-id)
- Attribute selector: selects elements with a specific attribute (e.g. [data-attribute])
- Descendant selector: selects elements that are descendants of another element (e.g. ul li)

Types of CSS

CSS (Cascading Style Sheets) can be implemented in three primary ways: inline, internal, and external. Inline CSS applies styles directly to individual HTML elements using the style attribute. Internal CSS defines styles within the <head> section of an HTML document using a <style> tag. External CSS utilizes separate CSS files that are linked to one or more HTML documents.
Here's a more detailed breakdown:

- **Inline CSS:** This method applies styles directly to an HTML element using the style attribute within the element's tag. For example:

```
<p style="color: blue; font-size: 16px;">This is some text.</p>
```

While convenient for quick changes, inline CSS is generally not recommended for larger projects due to its lack of reusability and potential for clutter.

- **Internal CSS:** This approach involves placing CSS rules within a <style> tag located in the <head> section of an HTML document. This allows you to style the entire document or specific elements within it, offering more organization than inline styles. For example:

```
<!DOCTYPE html>
<html>
<head>
<title>My Page</title>
<style>
  p {
    color: green;
    font-size: 18px;
  }
  h1 {
    color: blue;
  }
</style>
</head>
<body>
  <h1>My Heading</h1>
  <p>This is some text.</p>
</body>
</html>
```

- **External CSS:** This is the preferred method for styling large web pages or projects. It involves creating a separate CSS file (e.g., styles.css) containing all the CSS rules and linking it to the HTML document using the <link> tag in the <head> section. This promotes code organization, reusability, and allows for easy maintenance and updates. For example:

```
<!DOCTYPE html>
<html>
<head>
<title>My Page</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>My Heading</h1>
  <p>This is some text.</p>
```

```
</body>
</html>
```

And the styles.css file would contain the following:

```
p {
  color: green;
  font-size: 18px;
}
h1 {
  color: blue;
}
```

CSS Frameworks

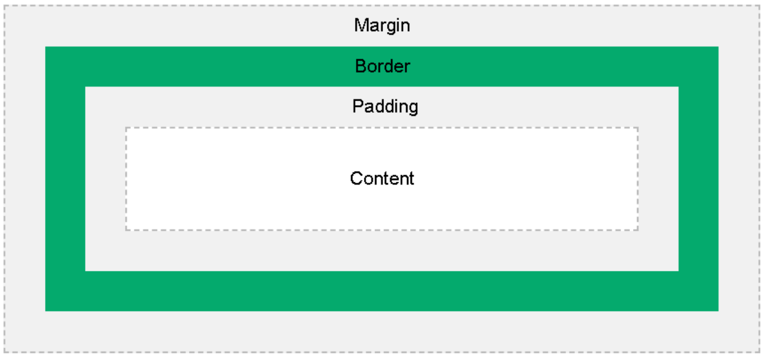
CSS frameworks are pre-written CSS code that developers can use to style their websites. They typically include styles for common elements like buttons, forms, and navigation menus, making it faster and easier to create a website. Some popular CSS frameworks include Bootstrap, Foundation, and Materialize.

Other CSS Frameworks like:

- Bulma
- Tailwind CSS
- UIKit
- Semantic UI
- Tachyons
- Milligram
- Fomantic-UI
- Primer
- Basscss
- Chota
- Open Props
- Vanilla framework
- Ant Design
- Luxa css
- Pico minimal css framework

CSS Box Model

In CSS, the term "box model" is used when talking about design and layout. The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins. The image below illustrates the box model:



Explanation of the different parts:

- Content - The content of the box, where text and images appear
- Padding - Clears an area around the content. The padding is transparent
- Border - A border that goes around the padding and content
- Margin - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

Example

Demonstration of the box model:

```
div {
  width: 300px;
  height: 200px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```

Example

This <div> element will have a total width of 350px and a total height of 80px:

```
div {
  width: 320px;
  height: 50px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;
}
```

Here is the calculation:

  320px (width of content area)

+ 20px (left padding + right padding)

+ 10px (left border + right border)
= 350px (total width)

  50px (height of content area)
+ 20px (top padding + bottom padding)
+ 10px (top border + bottom border)
= 80px (total height)

The total width of an element should be calculated like this:
Total element width = width + left padding + right padding + left border + right border

The total height of an element should be calculated like this:
Total element height = height + top padding + bottom padding + top border + bottom border

Flexbox and Grid Layouts
Flexbox and CSS Grid are two different CSS layout modules used for structuring web pages. Flexbox is designed for one-dimensional layouts (rows or columns), while Grid is for two-dimensional layouts (rows and columns simultaneously).

Flexbox
Flexbox is a one-dimensional layout model primarily used for arranging elements in a single direction, either horizontally (in a row) or vertically (in a column). It's well-suited for simpler layouts, like navigation bars, horizontal lists, or distributing space between items. Flexbox offers features like easy alignment, responsive design capabilities, and simplified code compared to traditional float and positioning methods.

Grid Layouts
CSS Grid is a two-dimensional layout system that allows you to create layouts with rows and columns simultaneously. It's ideal for more complex layouts, responsive designs, and precise placement of elements in a grid-like structure. Grid enables you to create structured layouts, distribute space evenly, and handle whitespace distribution more effectively than Flexbox.

Key Differences

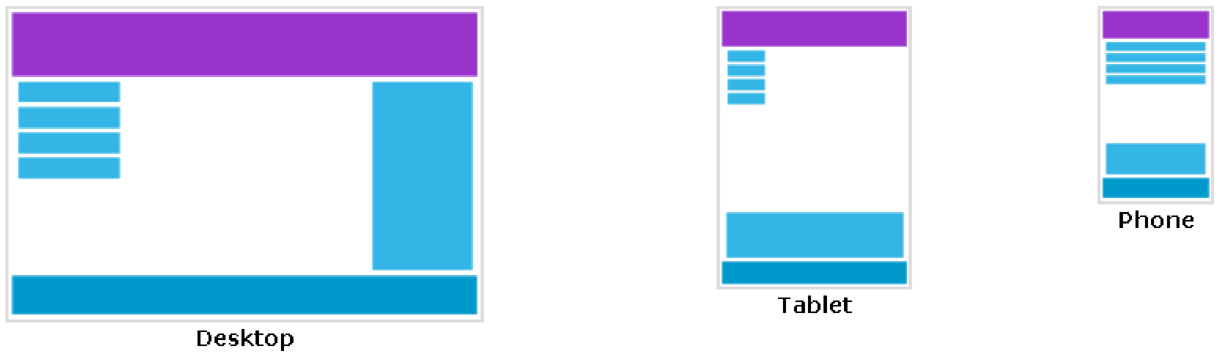
Feature	Flexbox	CSS Grid
Dimensions	One-dimensional (rows or columns)	Two-dimensional (rows and columns)
Use Cases	Simpler layouts, alignment, navigation	Complex layouts, responsive designs

Alignment	Align items within a single axis	Align items within rows and columns
Space Distribution	Distributes space along a single axis	Distributes space in rows and columns
Complexity	Simpler code for basic layouts	More complex code for advanced layouts

Responsive Design Principles

Introduction

- Responsive web design makes your web page look good on all devices.
- Responsive web design uses only HTML and CSS.
- Responsive web design is not a program or a JavaScript.
- Web pages can be viewed using many different devices: desktops, tablets, and phones.
- Your web page should look good, and be easy to use, regardless of the device.
- Web pages should not leave out information to fit smaller devices, but rather adapt its content to fit any device:



It is called responsive web design when you use CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen.

Design Principals

Responsive design ensures websites are accessible and enjoyable across various devices and screen sizes. Key principles include using fluid grids and images, utilizing media queries to adjust layouts based on device characteristics, and focusing on mobile-first design, ensuring a seamless user experience. The Interaction Design Foundation notes that these principles aim to create a consistent and user-friendly experience on all devices.

Here's a more detailed look at the core principles:

1. Fluid Grids:

The Interaction Design Foundation notes that fluid grids use relative units like **percentages** instead of **fixed pixel values** for element **widths**. This allows elements to scale proportionally to the screen size, preventing elements from overflowing or becoming too small on smaller screens.

2. Fluid Images:

Responsive images scale down proportionally to fit their container without distorting the image or losing quality. This ensures images don't break the layout or become unreadable on smaller screens.

3. Media Queries:

Media queries are CSS rules that apply different styles based on device characteristics, such as **screen width**, **orientation**, and **resolution**. Udacity states that they allow you to define specific layouts for different device types, like mobile, tablet, and desktop.

4. Mobile-First Approach:

Designing for smaller screens first ensures that the base layout is optimized for mobile devices and then scales up for larger screens. This approach can lead to a more efficient design process and a more user-friendly experience on all devices.

5. Breakpoints:

Breakpoints are specific screen sizes where the layout and content of the website change. They are used to adapt the design to different screen sizes, ensuring that content remains legible and the layout is functional on all devices.

6. Viewport Configuration:

Setting the viewport configuration tells the browser how to render the page, including scaling and orientation. This is crucial for ensuring that the website looks and behaves correctly on different devices.

7. Testing on Real Devices:

Testing on various real devices and browsers is essential to ensure that the website functions correctly and provides a consistent user experience across all platforms.