** Introduction to CSS**

What is CSS?

- CSS stands for Cascading Style Sheets. It is the language used to control the visual presentation of web pages, including layout, colors, fonts, and spacing.
- CSS works alongside HTML. HTML structures the content, while CSS defines how that content should look.

CSS Syntax:

- A CSS rule set consists of a selector and a declaration block:
- **Selector**: Specifies the HTML element you want to style.
- **Declaration Block**: Contains one or more declarations, each consisting of a CSS property and a value.

Types of CSS:

- - **Inline CSS**: Styles are written directly within an HTML element using the `style` attribute.
- ```html
- This is inline CSS
- '''
- - **Internal CSS**: Styles are placed inside the `<style>` tag within the `<head>` section of the HTML file.
- ```html
- <style>
- p { color: green; }
- </style>

- ***
- - **External CSS**: Styles are written in an external `.css` file and linked to the HTML file using the `<link>` tag.
- ```html
- link rel="stylesheet" href="styles.css">
- . "

Slide 3: CSS Selectors

- Selectors are used to "select" HTML elements to apply styles. Different types of selectors target elements in various ways.
- **Basic Selectors:**
- - **Element Selector**: Targets all instances of a specific HTML element.
- ```css
- p { color: blue; } /* Applies to all elements */
- ""
- - **Class Selector**: Targets elements that have a specific class name. Multiple elements can share the same class.
- ```css
- .example { color: red; } /* Applies to all elements with class="example" */

```
- **ID Selector**: Targets a single element with a unique ID. IDs must be unique per page.
    ```css
 #unique-id { color: green; } /* Applies to the element with id="unique-id" */
 Advanced Selectors:
 - **Attribute Selector**: Selects elements based on an attribute or its value.
    ```css
    input[type="text"] { border: 1px solid black; } /* Targets input elements with type="text" */
  - **Pseudo-Classes**: Selects elements based on their state.
- Example: `:hover`, `:focus`, `:nth-child()`
```

```
a:hover { color: orange; } /* Changes color on hover */
...
-**Pseudo-Elements**: Targets specific parts of elements.
- Example: `::before`, `::after`
p::before { content: "Note: "; } /* Inserts content before paragraph */
```

Combinators:

- - **Descendant Selector**: Selects elements that are descendants (children, grandchildren, etc.) of a specified element.
- div p { color: blue; } /* Targets elements inside <div> elements */
-
- - **Child Selector**: Selects direct children of an element.
- div > p { color: red; } /* Targets elements that are direct children of <div> */
- . "

Slide 4: CSS Box Model

- The CSS Box Model is the foundation of layout and spacing in CSS. It describes the space occupied by every element.
- **Components of the Box Model:**
- 1. **Content**: The actual content inside the element, such as text or images.
- 2. **Padding**: The space between the content and the border. It can be controlled on all sides (top, right, bottom, left).
- 3. **Border**: A border around the padding and content. You can control the style, width, and color.
- 4. **Margin**: The space outside the border, separating the element from other elements on the page.

```
**Example:**
```css
div {
width: 200px;
padding: 10px;
border: 1px solid black;
margin: 20px;
}
-**Total width** of the element = `width + padding + border + margin`
- In this case: `200px (width) + 20px (padding) + 2px (border) + 40px (margin)`
```

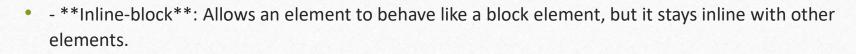
\*\*Box-sizing:\*\*

• - By default, `width` and `height` only include the content. Use `box-sizing: border-box;` to include padding and border in the element's width and height.

.

### ### \*\*Slide 5: CSS Layout Techniques\*\*

- CSS provides different techniques to control the layout of elements on a web page.
- •
- \*\*Display Types:\*\*
- - \*\*Block\*\*: Elements take up the full width of their container (e.g., `<div>`, ``).
- div { display: block; }
- . ...
- - \*\*Inline\*\*: Elements take up only as much width as necessary (e.g., `<span>`, `<a>`).
- span { display: inline; }



- ```css
- img { display: inline-block; }
- \*\*\*

## \*\*Positioning:\*\*

- - \*\*Static\*\*: The default positioning method. Elements flow naturally on the page.
- - \*\*Relative\*\*: Positioned relative to its normal position.
- ```css
- div { position: relative; top: 20px; } /\* Moves the element 20px down \*/
- . ...
- - \*\*Absolute\*\*: Positioned relative to its nearest positioned ancestor.
- ```css
- div { position: absolute; left: 100px; top: 50px; }

- \*\*Fixed\*\*: Positioned relative to the viewport and stays in place when the page is scrolled.
  ```css
 header { position: fixed; top: 0; }
- **Sticky**: Switches between relative and fixed depending on the scroll position.
- ```css

...

nav { position: sticky; top: 0; }

Flexbox Layout:

- **Flexbox** simplifies the creation of flexible, responsive layouts. It aligns items along a single axis (horizontal or vertical).
- ```css
- .container {
- display: flex;
- justify-content: space-between; /* Space items equally along the row */
- align-items: center; /* Vertically align items */
- •

Grid Layout:

```
    - **Grid** divides the webpage into rows and columns, providing more control over layout compared to Flexbox.
    ```css
```

display: grid;

.container {

grid-template-columns: repeat(3, 1fr); /\* 3 equal columns \*/

• }

# \*\*Slide 6: CSS Colors and Backgrounds\*\*

```
Color Values:

- **Hexadecimal Colors**: Used to define colors with a six-character code.

```css

color: #ff5733;

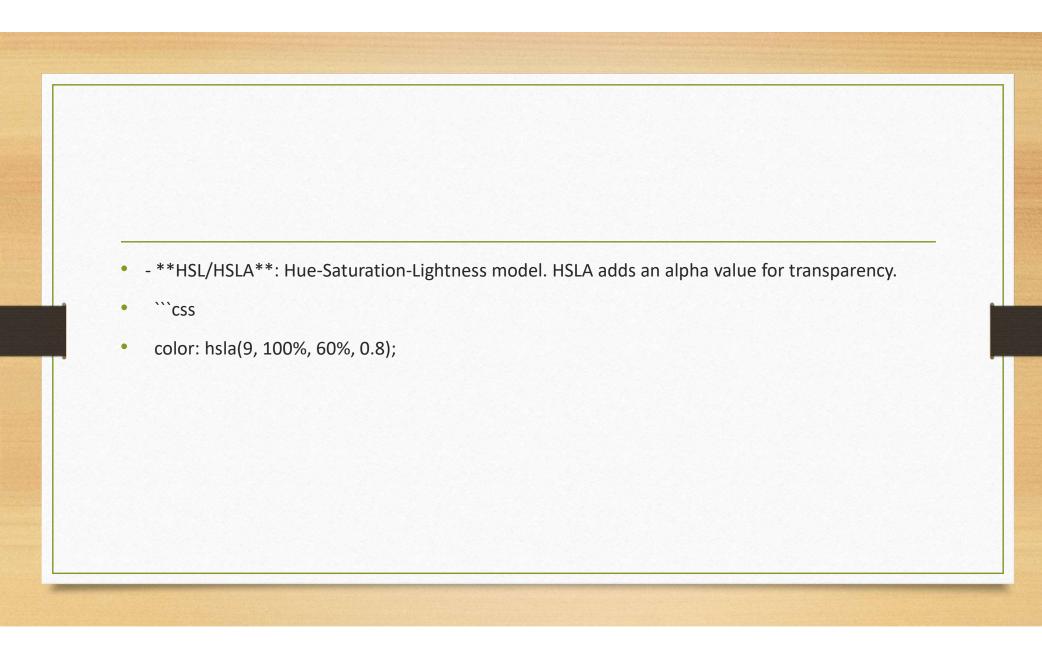
.``
```

- **RGB/RGBA**: Red-Green-Blue color values. RGBA adds an alpha value for transparency.

```css

color: rgba(255, 87, 51, 0.8); /\* 80% opacity \*/

CSS allows you to add colors and backgrounds to elements to improve visual design.



# \*\*Backgrounds:\*\*

```
CSS allows you to set background colors, images, and gradients for elements.
'``css
body {
background-color: lightblue;
background-image: url('image.jpg');
background-size: cover; /* Ensure the image covers the entire background */
```

- You can also apply \*\*CSS Gradients\*\* as backgrounds.```css
- background: linear-gradient(to right, red, yellow);
- \*\*\*
- .

## ### \*\*Slide 7: Typography in CSS\*\*

- Typography is critical in defining how text appears on a webpage.
- \*\*Fonts:\*\*
- - CSS allows you to define font families, sizes, and styles.
- ```css
- h1 {
- font-family: Arial, sans-serif;
- font-size: 32px;
- font-weight: bold;
- font-style: italic;

### \*\*Slide 8: CSS Animations and Transitions\*\*

- CSS provides powerful tools to create visual effects and smooth transitions for user interactions.
- \*\*Transitions:\*\*
- - Transitions allow you to change property values smoothly over a specified duration.
- - \*\*Syntax\*\*: Define the property, duration, timing function, and delay.
- ```css
- button {
- transition: background-color 0.5s ease; /\* Change background color over 0.5 seconds \*/
- •

### \*\*Animations:\*\*

- CSS animations allow for more complex sequences of animations.
- You define keyframes that specify the styles at various points in the animation.
- @keyframes slide {
- from { transform: translateX(0); }
- to { transform: translateX(100px); } /\* Moves element 100px to the right \*/
- •
- .
- .moving-element {
- animation: slide 1s forwards; /\* Plays the animation over 1 second \*/
- •

### \*\*Slide 9: Responsive Design\*\*

```
Responsive design ensures that web pages look good on all devices by adapting layouts to various screen sizes.
Media Queries:

Media queries allow you to apply different styles based on device characteristics (like width, height, resolution).
```css
@media (max-width: 600px) {
body {
background-color: lightgreen; /* Changes background color on small screens */
}
```

Flexible Layouts:

- Use relative units like percentages (`%`), viewport width (`vw`), and viewport height (`vh`) to create fluid layouts.
- ```css
- .container {
- width: 80%; /* Container takes 80% of the viewport width */
- •

Mobile-First Approach:

```
Design for mobile devices first, then enhance for larger screens using media queries.
/* Default styles for mobile */
body {
font-size: 16px;
}
/* Enhancements for larger screens */
@media (min-width: 768px) {
body {
font-size: 18px; /* Increases font size on tablets and larger screens */
}
```