**CSS: LESSON PLAN**

**Lesson 1. What is CSS?**

**Definitions**

CSS stands for Cascading Style Sheets and is a style sheet language used to describe the look and formatting of a document written in HTML or XML. It allows you to control the presentation of web pages, including elements such as layout, colors, fonts, and more.

**Inline, internal, and external CSS**

**Inline CSS:** You can apply styles directly to individual HTML elements using the style attribute. For example, <h1 style="color: blue;">Hello, World!</h1>. Inline CSS has the highest specificity and overrides other styles.

**Internal CSS:** You can include CSS styles within the <style> tags in the <head> section of an HTML document.

**External CSS:** You can create a separate CSS file with a **.css** extension and link it to your HTML document using the **<link>** tag. This allows you to reuse styles across multiple HTML pages. For example,

**CSS Syntax and Selectors**

**Tag Selector** allows you to target all the defined tags. For example, h1 lets you apply css to all <h1> tag.

**ID selectors** target elements based on their unique **id** attribute. To define an ID selector, prepend a hash (**#**) followed by the ID name in your CSS rule. For example, **#header** would target the element with **id="header"**

**Class selectors** allow you to target elements based on the value of their **class** attribute. To define a class selector, prepend a period (**.**) followed by the class name in your CSS rule. For example, **.highlight** would target any element with **class="highlight"**.

**Lesson 2: CSS Box Model**



The CSS box model describes how elements are rendered on a web page. Each element is treated as a rectangular box, consisting of four main components: content, padding, border, and margin.

**Height and Width**: These properties define the dimensions of the content area of an element. You can specify values in **pixels**, or other units.

**max-height**: This property sets the maximum height that an element can have. If the content exceeds this height, it will be truncated or overflowed, depending on other related properties.

**min-height**: This property sets the minimum height that an element should have. If the content is less than this height, the element will expand to accommodate the minimum height.

**max-width**: This property sets the maximum width that an element can have. If the content exceeds this width, it will be truncated or overflowed, depending on other related properties.

**min-width**: This property sets the minimum width that an element should have. If the content is less than this width, the element will expand to accommodate the minimum width.

**Padding:** Padding is the space between the content area and the element's border. It can be set using properties like padding-top, padding-right, padding-bottom, and padding-left.

**Border:** The border surrounds the padding and content areas. You can control its width, style, and color using properties like **border-width**, **border-style**, and **border-color**.

**Examples**

**solid - Defines a solid border**

**dotted - Defines a dotted border**

**dashed - Defines a dashed border**

**double - Defines a double border**

Margin: The margin is the space outside the border, creating the gap between adjacent elements. It can be set using properties like **margin-top**, **margin-right**, **margin-bottom**, and **margin-left**.

**Lesson 3: CSS Typography**

**Font properties: font-family, font-size, font-weight**

The **font-family** property allows you to specify the font family for text elements. You can provide multiple font names separated by commas, and the browser will attempt to use the first available font. For example, **font-family: Arial, sans-serif;**.

The **font-size** property determines the size of the text. You can specify values in pixels, ems, rems or other units. For example, **font-size: 16px;**.

The **font-weight** property controls the thickness or boldness of the text. Common values are **normal** and **bold**, but you can also use numeric values like **400** or **700** for finer control.

**Text properties: text-align, text-decoration**

**The text-align property** determines the alignment of the text within its container. Common values are left, right, center, and justify.

The **text-decoration** property allows you to add visual effects to text, such as underline or line-through. For example, **text-decoration: underline**.

**Lesson 4: CSS Layout**

**Positioning: static, relative, absolute, fixed**

**position: static** is the default positioning value, where elements are rendered in their normal order in the document flow.

**position: relative** allows you to position an element relative to its normal position. Setting the **top**, **right**, **bottom**, and **left** properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

**position: absolute** removes the element from the normal flow and positions it relative to its closest positioned ancestor. It can be useful for creating overlays or positioning elements precisely on the page.

**position: fixed** positions the element relative to the browser window, so it remains fixed even when scrolling. It is commonly used for fixed headers or navigation bars.

**Display property: block, inline, inline-block**

The display property controls how elements are rendered in terms of their box layout:

**display: block** makes the element a block-level element, taking up the full available width and stacking vertically. Block-level elements start on a new line by default.

**display: inline** makes the element an inline-level element, meaning it takes up only as much space as necessary and does not force line breaks. Inline elements can be placed alongside each other.

**display: inline-block** combines characteristics of both block and inline elements. It allows elements to have a width and height and be placed inline.

**Float and clear**

**The float property** allows elements to be positioned to the left or right of their containing element. Floated elements are taken out of the normal flow, and other elements flow around them.

The **clear** property specifies whether an element should be moved below any floated elements. It ensures that no element touches the left or right side of a floated element.

**CSS Grid and Flexbox**

CSS Grid and Flexbox are powerful layout tools that provide more advanced and flexible ways of creating responsive layouts:

CSS Grid allows you to create grid-based layouts with rows and columns. It provides precise control over the placement and sizing of elements within the grid.

Flexbox is designed for simpler one-dimensional layouts, either in a row or a column. It allows you to distribute space among items and control their alignment.

**Lesson 5: CSS Colors and Backgrounds**

**Color values: hex, rgb, named colors**

**CSS offers various ways to define colors:**

**Named colors** provide predefined color names, such as red, blue, or green. These named colors can be convenient to use, but they offer a more limited range of options.

**Hexadecimal (hex)** values are the most common way to specify colors. They consist of a pound sign (#) followed by six hexadecimal digits that represent the red, green, and blue (RGB) values. For example, #FF0000 represents pure red.

**RGB values** allow you to specify colors by their red, green, and blue component values. Each component value ranges from 0 to 255. For example, rgb(255, 0, 0) represents pure red.

**Background properties: background-color, background-image**

**The background-color property** sets the background color of an element. You can use any valid color value to define the background color.

**The background-image** property allows you to set an image as the background of an element. You can specify the image URL, and it will be displayed as the background. You can also control aspects like the background size, repeat behavior, and positioning.

**Background positioning and repeating**

CSS provides several properties to control the positioning and repetition of background images:

**background-size** controls the size of the background image. You can use values like cover to ensure the image covers the entire background area or contain to fit the image within the background area.

**background-repeat** determines whether the background image should repeat horizontally, vertically, or not at all. Common values are repeat, repeat-x, repeat-y, and no-repeat.

**background-position** allows you to specify the starting position of the background image. You can use keywords like top, bottom, left, and right, or you can specify the position in pixels or percentages.

**CSS Transitions**

CSS transitions allow you to create smooth and gradual changes in CSS property values over a specified duration. With transitions, you can animate properties such as **width**, **height**, **color**, **opacity**, and more.

**To define a transition**, you need to specify the CSS property you want to animate, the duration of the transition, and optional timing functions for easing the transition.

.button:hover {

background-color: blue;

}

.button {

background-color: red;

transition-property: background-color;

transition-duration: 0.3s;

}

**Cursor**

The **cursor** [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) property sets the mouse cursor, if any, to show when the mouse pointer is over an element

Help, wait, crosshair, not-allowed, zoom-in, grab, pointer

**Lesson 6: Media Queries**

**/\* PHONE \*/**

**@media (max-width: 767px) {**

**}**

**/\* TABLET \*/**

**@media (min-width: 768px) and (max-width: 991px)**

**{**

**}**

**/\* Laptop \*/**

**@media (min-width: 992px)**

**{**

**}**

**ACTIVITY: MAKE A REGISTRATION FORM**