**CSS Course Outline**

**Module 1: Introduction to CSS**

1. **What is CSS?**
2. **Types of CSS** 
   * **Inline CSS**
   * **Internal CSS**
   * **External CSS**
3. **Basic CSS Syntax & Structure**
4. **Applying CSS to HTML** 
   * **Example Code Snippets**

**Module 2: CSS Selectors & Specificity**

1. **Basic Selectors** 
   * **Element Selector**
   * **Class Selector**
   * **ID Selector**

**Sure! Below is a detailed content outline based on the topics you provided, along with relevant examples to demonstrate each concept.**

**Module 2: Advanced CSS Selectors**

**1. Attribute Selectors**

**Attribute selectors target HTML elements based on their attributes and values. These can be used to select elements with a specific attribute or value.**

**Example:**

**/\* Select all anchor tags with a title attribute \*/**

**a[title] {**

**color: blue;**

**}**

**/\* Select input elements with type="text" \*/**

**input[type="text"] {**

**border: 2px solid black;**

**}**

**/\* Select all paragraphs with a class that contains the word 'intro' \*/**

**p[class\*="intro"] {**

**font-size: 16px;**

**}**

**2. Grouping Selectors**

**Grouping selectors allow you to apply the same styles to multiple selectors, separated by commas.**

**Example:**

**/\* Apply the same styles to h1, h2, and h3 elements \*/**

**h1, h2, h3 {**

**font-family: Arial, sans-serif;**

**color: darkblue;**

**}**

**Module 3: Pseudo-classes & Pseudo-elements**

**1. Pseudo-classes**

**Pseudo-classes are used to define special states of an element. They are usually used to style elements based on user interaction or their position in the document.**

**Example:**

**/\* Style an anchor tag when the user hovers over it \*/**

**a:hover {**

**color: red;**

**}**

**/\* Style the first child of any unordered list \*/**

**ul li:first-child {**

**font-weight: bold;**

**}**

**/\* Style a visited link \*/**

**a:visited {**

**color: purple;**

**}**

**2. Pseudo-elements**

**Pseudo-elements allow you to style parts of an element, like the first letter or line of text.**

**Example:**

**/\* Add content before each paragraph \*/**

**p::before {**

**content: "Note: ";**

**font-weight: bold;**

**}**

**/\* Style the first letter of every paragraph \*/**

**p::first-letter {**

**font-size: 2em;**

**color: orange;**

**}**

**Module 3: CSS Specificity & Inheritance**

**1. CSS Specificity**

**Specificity determines which CSS rule is applied by the browser when there are conflicting rules. It is calculated based on the number of IDs, classes, and elements.**

**Example:**

**/\* Less specific \*/**

**p {**

**color: blue;**

**}**

**/\* More specific \*/**

**div p {**

**color: green;**

**}**

**/\* Even more specific \*/**

**#main div p {**

**color: red;**

**}**

**2. CSS Inheritance**

**Some properties, like font-size or color, are inherited from parent elements to child elements. You can override inherited styles.**

**Example:**

**/\* This color will be inherited by child elements \*/**

**body {**

**color: #333;**

**}**

**/\* The child div element will inherit the color from the body \*/**

**div {**

**/\* Inherited: color will be #333 \*/**

**}**

**Module 4: Box Model & Layout**

**1. Understanding the Box Model**

**The box model consists of four parts: content, padding, border, and margin.**

**Example:**

**/\* Box model with content, padding, border, and margin \*/**

**div {**

**width: 300px;**

**height: 200px;**

**padding: 20px;**

**border: 2px solid black;**

**margin: 10px;**

**}**

**2. Box Model Properties**

* **Margin: The outermost space around the element.**
* **Padding: The space between the element’s content and its border.**
* **Border: The line surrounding the element's padding.**
* **Content: The actual content of the element.**

**Example:**

**div {**

**margin: 10px;**

**padding: 20px;**

**border: 2px solid black;**

**width: 200px; /\* Content box \*/**

**}**

**3. CSS Display Property**

**The display property defines how elements are displayed.**

**Example:**

**/\* Block-level element \*/**

**div {**

**display: block;**

**}**

**/\* Inline element \*/**

**span {**

**display: inline;**

**}**

**/\* Inline-block element \*/**

**button {**

**display: inline-block;**

**}**

**4. CSS Positioning**

**The positioning of elements can be controlled using properties like position, top, left, right, and bottom.**

**Example:**

**/\* Static positioning (default) \*/**

**div {**

**position: static;**

**}**

**/\* Relative positioning (offsets from the normal position) \*/**

**div {**

**position: relative;**

**top: 20px;**

**}**

**/\* Absolute positioning (positioned relative to the nearest positioned ancestor) \*/**

**div {**

**position: absolute;**

**top: 50px;**

**left: 30px;**

**}**

**/\* Fixed positioning (fixed to the viewport) \*/**

**div {**

**position: fixed;**

**bottom: 0;**

**}**

**/\* Sticky positioning (behaves like relative until scrolled) \*/**

**div {**

**position: sticky;**

**top: 0;**

**}**

**5. Z-index & Overflow**

**z-index controls the stacking order of elements. overflow defines what happens when content overflows an element’s box.**

**Example:**

**/\* Elements with higher z-index appear on top \*/**

**div {**

**position: absolute;**

**z-index: 10;**

**}**

**div.overlay {**

**position: absolute;**

**z-index: 20; /\* Stacks on top of the previous div \*/**

**}**

**/\* Overflow property \*/**

**.container {**

**width: 300px;**

**height: 200px;**

**overflow: auto; /\* Adds a scrollbar if content overflows \*/**

**}**

**Module 5: CSS Flexbox**

**1. Introduction to Flexbox**

**Flexbox is a layout module designed for distributing space along a row or column, and aligning content within a container.**

**2. Flex Container Properties**

* **display: flex: Defines a flex container.**
* **flex-direction: Defines the direction of the flex items.**
* **justify-content: Aligns items horizontally.**
* **align-items: Aligns items vertically.**

**Example:**

**/\* Flex container \*/**

**.container {**

**display: flex;**

**flex-direction: row; /\* Items in a row \*/**

**justify-content: space-between;**

**align-items: center;**

**}**

**/\* Flex items \*/**

**.item {**

**flex-grow: 1; /\* Item will grow to fill available space \*/**

**}**

**3. Flex Items Properties**

* **flex-grow: Defines how much a flex item will grow.**
* **flex-shrink: Defines how much a flex item will shrink.**
* **flex-basis: Defines the initial size of a flex item.**

**Example:**

**.item {**

**flex-grow: 2; /\* This item grows twice as much as others \*/**

**flex-basis: 100px; /\* Starts with 100px width \*/**

**}**

**Module 6: CSS Grid**

**1. Introduction to CSS Grid**

**CSS Grid is a two-dimensional layout system for web pages, allowing items to be arranged into rows and columns.**

**2. Grid Container Properties**

* **grid-template-columns: Defines the number and size of columns.**
* **grid-template-rows: Defines the number and size of rows.**
* **gap: Defines the space between grid items.**

**Example:**

**/\* Grid container \*/**

**.container {**

**display: grid;**

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

**gap: 10px;**

**}**

**/\* Grid items \*/**

**.item {**

**grid-column: span 2; /\* Item spans across 2 columns \*/**

**}**

**Module 7: Responsive Web Design (RWD)**

**1. What is Responsive Web Design?**

**Responsive Web Design (RWD) ensures that a website looks good on all devices (desktops, tablets, and mobile).**

**2. Introduction to Media Queries**

**Media queries are used to apply different styles for different screen sizes.**

**Example:**

**/\* Style for screens larger than 600px \*/**

**@media screen and (min-width: 600px) {**

**body {**

**font-size: 18px;**

**}**

**}**

**/\* Style for screens smaller than 600px \*/**

**@media screen and (max-width: 599px) {**

**body {**

**font-size: 14px;**

**}**

**}**

**3. CSS Units & Relative Sizing**

**CSS offers several units for setting sizes, including em, rem, %, vw, vh.**

**Example:**

**/\* Relative sizing \*/**

**body {**

**font-size: 1rem; /\* Relative to the root element \*/**

**}**

**/\* Viewport-based sizing \*/**

**.container {**

**width: 50vw; /\* 50% of the viewport width \*/**

**height: 50vh; /\* 50% of the viewport height \*/**

**}**

**4. Mobile-First Design Approach**

**Design mobile-friendly layouts first, then progressively enhance them for larger screens.**

**5. CSS Frameworks (Bootstrap, Tailwind CSS)**

**These frameworks provide pre-built classes for building responsive and modern websites.**

**Module 8: CSS Preprocessors (Optional)**

**1. Introduction to SASS & LESS**

**SASS and LESS are CSS preprocessors that offer features like variables, nesting, and mixins.**

**2. Using Variables & Nesting**

**Variables store reusable values. Nesting allows styles to be organized in a hierarchical manner.**

**Example (SASS):**

**$primary-color: blue;**

**nav {**

**background-color: $primary-color;**

**ul {**

**list-style: none;**

**}**

**}**