**1. What is CSS?**

CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation of HTML elements. It allows you to control the layout, colors, fonts, spacing, and positioning of elements in web pages.

**Syntax:**

selector {

property: value;

}

**Example:**

<h1 style="color: blue;">Hello, World!</h1>

**2. CSS Selectors**

Selectors are used to target HTML elements and apply styles.

* **Element Selector**: Targets elements directly (e.g., div, p).
* **Class Selector**: Targets elements with a specific class (e.g., .className). Classes are reusable, meaning you can apply the same class to multiple elements.
* **ID Selector**: Targets elements with a specific ID (e.g., #idName). IDs must be unique within a page, meaning each ID should only be applied to one element.
* **Universal Selector**: Targets all elements (\*).
* **Attribute Selector**: Targets elements based on attributes (e.g., [type="text"]).

**Key Difference between Class and ID:**

* **Class**: A class selector can be applied to multiple elements on a page. It is reusable and is used for styling similar elements that share a common style.
* **ID**: An ID selector is unique and should be used for only one element on a page. It is often used for targeting a specific element, such as in JavaScript for dynamic behavior or when creating unique styles.

**Example:**

h1 {

color: blue;

}

.title {

font-weight: bold;

}

#main {

text-align: center;

}

\* {

margin: 0;

padding: 0;

}

**3. Text Properties**

Text properties are used to manipulate text alignment, decoration, spacing, transformation, and shadows.

* **text-align**: Defines horizontal alignment of text.
* **text-decoration**: Adds decoration (underline, strikethrough).
* **text-transform**: Controls the capitalization of text.
* **letter-spacing**: Sets the spacing between characters.
* **text-shadow**: Adds shadow to text.

**Example:**

h1 {

text-align: center;

text-decoration: underline;

text-transform: uppercase;

letter-spacing: 2px;

text-shadow: 2px 2px 5px gray;

}

**4. Box Model**

The box model is the core concept of CSS layout and refers to the way elements are rendered. It consists of content, padding, border, and margin.

* **Content**: The actual text or elements.
* **Padding**: Space between the content and the border.
* **Border**: Border surrounding the padding and content.
* **Margin**: Space outside the border.

**Example:**

div {

padding: 20px;

border: 5px solid black;

margin: 10px;

}

**5. Positioning**

CSS positioning controls the position of elements on a page.

* **static**: Default position (normal document flow).
* **relative**: Positioned relative to its normal position.
* **absolute**: Positioned relative to its nearest positioned ancestor.
* **fixed**: Positioned relative to the viewport.
* **sticky**: Behaves like relative until a specified scroll point.

**Example:**

div {

position: absolute;

top: 100px;

left: 50px;

}

**6. Display Property**

The display property specifies how an element is displayed on the page.

* **block**: Element occupies the full width.
* **inline**: Element does not start on a new line.
* **inline-block**: Behaves like inline but can accept width/height properties.
* **none**: Hides the element from the page.

**Example:**

div {

display: block;

}

span {

display: inline;

}

**7. Flexbox**

Flexbox is a layout model that allows elements to be arranged in rows or columns.

* **justify-content**: Aligns items on the main axis.
* **align-items**: Aligns items on the cross axis.
* **align-self**: Aligns a single item within its container.
* **flex-direction**: Defines the direction of the flex items.

**Example:**

.container {

display: flex;

justify-content: space-between;

align-items: center;

}

.item {

flex: 1;

}

**8. Grid Layout**

CSS Grid Layout is a two-dimensional system for web design that allows you to create complex layouts.

* **grid-template-columns/rows**: Defines the number of columns/rows in the grid.
* **grid-gap**: Defines the gap between grid items.
* **grid-column**: Specifies where an item starts and ends on the grid.
* **grid-row**: Specifies the row a grid item occupies.

**Example:**

.container {

display: grid;

grid-template-columns: 1fr 2fr 1fr;

grid-gap: 10px;

}

.item {

grid-column: span 2;

}

**9. Background Properties**

Background properties define the background of an element.

* **background-color**: Sets the background color.
* **background-image**: Sets a background image.
* **background-size**: Defines the size of the background image.
* **background-position**: Defines the position of the background image.
* **background-repeat**: Defines whether the background image should repeat.

**Example:**

div {

background-color: lightblue;

background-image: url('background.jpg');

background-size: cover;

}

**10. Border Properties**

CSS border properties define the style, width, and color of the border around elements.

* **border**: Sets all border properties (width, style, color).
* **border-width**: Defines the width of the border.
* **border-style**: Defines the border style (e.g., solid, dotted, dashed).
* **border-color**: Sets the color of the border.

**Example:**

div {

border: 2px solid black;

}

**11. Margin and Padding**

* **margin**: Controls the space outside of an element.
* **padding**: Controls the space inside of an element.

**Example:**

div {

margin: 20px;

padding: 10px;

}

**12. Opacity**

The opacity property controls the transparency of an element.

**Example:**

div {

opacity: 0.5; /\* 50% opacity \*/

}

**13. Z-Index**

The z-index property controls the stacking order of elements.

**Example:**

div {

position: absolute;

z-index: 10; /\* Elements with higher z-index appear above others \*/

}

**14. Transitions**

CSS transitions allow you to change property values smoothly (over a specified duration).

**Syntax:**

transition: <property> <duration> <timing-function> <delay>;

**Example:**

div {

transition: background-color 0.5s ease-in-out;

}

div:hover {

background-color: yellow;

}

**15. Animations**

CSS animations allow you to create keyframe-based animations.

**Syntax:**

@keyframes animationName {

0% { /\* Initial state \*/ }

100% { /\* Final state \*/ }

}

**Example:**

@keyframes move {

0% { left: 0; }

100% { left: 100px; }

}

div {

position: absolute;

animation: move 2s infinite;

}

**16. Media Queries**

Media queries allow you to apply styles based on device characteristics (such as screen size, resolution, etc.).

**Syntax:**

@media (min-width: 600px) {

/\* Styles for devices with width >= 600px \*/

}

**Example:**

@media (max-width: 600px) {

body {

background-color: lightgreen;

}

}

**17. Variables (CSS Custom Properties)**

CSS variables allow you to store values in reusable variables.

**Syntax:**

--variable-name: value;

**Example:**

:root {

--primary-color: blue;

}

div {

background-color: var(--primary-color);

}

**18. Overflow**

The overflow property controls what happens when content overflows its container.

**Possible Values:**

* **visible**: Content is not clipped and may overflow.
* **hidden**: Content is clipped and hidden.
* **scroll**: Scrollbars appear, allowing you to scroll the content.
* **auto**: Scrollbars appear only when needed.

**Example:**

div {

overflow: hidden;

}

**19. List Style Properties**

These properties are used to control the appearance of list items.

* **list-style-type**: Defines the type of list item marker (e.g., disc, circle, square).
* **list-style-position**: Defines whether the list marker is inside or outside the list item.

**Example:**

ul {

list-style-type: circle;

list-style-position: inside;

}