CSS Theory Note

CSS (Cascading Style Sheets) is a style sheet language used to define the look and layout of web pages. It controls how HTML elements are displayed on the screen, on paper, or in other media.

---

1. Basic Concepts of CSS

- Selector: Targets the HTML element(s) to style.
- Property: Defines the aspect of the element to style (e.g., color, font-size).
- Value: Specifies the value of the property.

Example:

```css
p {
  color: blue;
  font-size: 16px;
}
```

In the example above:

- `p` is the selector.
- `color` and `font-size` are properties.
- `blue` and `16px` are values.

---

2. CSS Syntax

A CSS rule consists of a selector and a declaration block**. The declaration block contains one or more declarations separated by semicolons.

```css
selector {
  property: value;
}
```

3. Types of CSS

Inline CSS: Defined directly within the HTML element using the `style` attribute.

```html
<h1 style="color: red;">This is a red heading</h1>
```

- Internal CSS: Placed within a `<style>` tag inside the `<head>` section of an HTML document.

```html
<style>
  h1 {
```

```
    color: red;
  }
</style>
```

- External CSS: Linked to an external stylesheet file using the `<link>` element.

```html
<link rel="stylesheet" href="styles.css">
```

4. CSS Selectors

CSS selectors define which HTML elements to apply styles to.

Common Selectors:

- Element Selector: Selects elements by their tag name.

```
p {
  color: green;
}
```

- Class Selector: Selects elements with a specific class name (uses a dot `.`).

```
.example {
  background-color: yellow;
```

```
}
```

- **ID Selector**: Selects an element with a specific ID (uses a hash `#`).

```
#header {
  text-align: center;
}
```

-Universal Selector: Selects all elements (`*`).

```
* {
  margin: 0;
  padding: 0;
}
```

Attribute Selector: Selects elements based on an attribute or attribute value.

```
a[target="_blank"] {
  color: blue;
}
```

5. CSS Box Model

The CSS Box Model is a fundamental concept that defines how elements are displayed on the page. Every element is considered a box, which consists of:

1. Content: The actual content (e.g., text, image).

2. Padding: Space between the content and the border.

3. Border: Surrounds the padding (optional).

4. Margin: Space outside the border, separating the element from others.

Box Model Properties:

- `width` and `height`: Define the content's dimensions.

- `padding`: Space between the content and the border.

- `border`: Thickness and style of the border.

- `margin`: Space outside the border.

Example:

```css
div {
  width: 200px;
  padding: 10px;
  border: 2px solid black;
  margin: 20px;
}
```

6. CSS Positioning

CSS positioning allows you to control the layout of elements relative to other elements or the viewport.

Types of Positioning:

1. Static (default): Elements are positioned according to the normal flow of the document.

2. Relative: Positioned relative to its normal position.

3. Absolute: Positioned relative to the nearest positioned ancestor (ignores normal document flow).

4. Fixed: Positioned relative to the browser window (remains fixed during scrolling).

5. Sticky: Switches between relative and fixed, based on scroll position.

Example:

```
div {
  position: absolute;
  top: 50px;
  left: 100px;
}
```

7. CSS Display and Visibility

- display: Defines how an element is displayed in the document.

- `block`: Element occupies the full width and starts on a new line.

- `inline`: Element takes only as much width as necessary and doesn't start on a new line.

- `inline-block`: Acts like an inline element but can have a width and height.

- `none`: Hides the element and removes it from the layout.

-visibility: Controls whether an element is visible or hidden.

- visible: Element is visible (default).

- `hidden`: Element is invisible, but space is still occupied.

---

8. CSS Flexbox

Flexbox is a layout model designed to align and distribute space among items within a container. It's especially useful for creating responsive designs.

Important Flexbox Properties:

- `display: flex`: Defines a flex container.

- `flex-direction`: Defines the direction of the flex items (`row`, `column`, etc.).

- `justify-content`: Aligns items along the main axis (`flex-start`, `flex-end`, `center`, `space-between`, `space-around`).

- `align-items`: Aligns items along the cross-axis (`flex-start`, `flex-end`, `center`, `baseline`, `stretch`).

- `flex-wrap`: Specifies whether items should wrap or stay on one line.

Example:

```css
.container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}
```

9. CSS Grid

CSS Grid is a two-dimensional layout system that allows for the creation of complex, responsive layouts.

Important Grid Properties:

- `display: grid`: Defines a grid container.

- `grid-template-columns` / `grid-template-rows`: Defines the number and size of columns/rows.

- `grid-gap`: Defines spacing between rows and columns.

- `grid-area`: Assigns elements to specific areas in the grid.

Example:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-gap: 10px;
}
```

10. CSS Colors

- Named Colors: Predefined color names like `red`, `blue`, `green`, etc.
- Hexadecimal: Uses hex codes (e.g., `#FF5733`).
- RGB: Uses the RGB model (e.g., `rgb(255, 87, 51)`).
- RGBA: Like RGB, but with an alpha value for transparency (e.g., `rgba(255, 87, 51, 0.5)`).
- HSL: Uses hue, saturation, and lightness (e.g., `hsl(9, 100%, 60%)`).

11. Responsive Design with CSS

Responsive design ensures that web pages look good on all devices by adapting to different screen sizes.

Techniques:

-Media Queries: Apply styles based on the device's screen size.

```css
@media (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

- **Fluid Layouts**: Use percentage-based widths and flexible units like `em`, `rem`, and `vh/vw` instead of fixed `px` units.

- **Responsive Images**: Use the `max-width` property to ensure images scale appropriately.

```css
img {
  max-width: 100%;
  height: auto;
}
```

12. CSS Animations

CSS can be used to create animations that change the style of elements over time.

Key Properties:

- `@keyframes`: Defines the animation.

- `animation-name`: Specifies the keyframe animation to use.

- `animation-duration`: Sets how long the animation lasts.

- `animation-iteration-count`: Sets how many times the animation will run.

Example:

```css
@keyframes slideIn {
  from {
    transform: translateX(-100%);
  }
  to {
    transform: translateX(0);
  }
}

div {
  animation-name: slideIn;
  animation-duration: 2s;
}
```

13. CSS Transitions

CSS transitions allow for smooth changes between different states of an element (e.g., hover effects).

Example:

```
button {
  background-color: blue;
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: green;
}
```

14. Z-Index and Stacking Context

- Z-index: Controls the stack order of elements. Higher values are displayed on top of lower values.

```
 div {
   position: absolute;
   z-index: 10;
 }
```

Stacking Context: Refers to how elements are stacked along the z-axis in relation to each other. The `z-index` property only applies to positioned elements (`relative`, `absolute`, `fixed`).

15. Pseudo-classes and Pseudo-elements**

Pseudo-classes:

Used to define a special state of an element.

- `:hover`: Applies when the user hovers over an element.