CSS Notes

Table of Contents

Introduction to CSS

- **CSS** (Cascading Style Sheets) is a language used to describe the style of an HTML document.
- CSS controls the layout of multiple web pages all at once.

CSS Syntax

- CSS is made up of rules, and each rule has a **selector** and **declaration block**.
- **Selector**: Specifies the HTML element to be styled.
- **Declaration Block**: Contains one or more declarations separated by semicolons.

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

Example:

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

Selectors

Basic Selectors

- **Universal Selector (`*`)**: Selects all elements.
- **Type Selector (`element`)**: Selects all elements of a given type.
- **Class Selector (`.classname`)**: Selects all elements with a specific class.
- **ID Selector (`#id`)**: Selects a single element with a specific ID.

Example:

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

h1 {
    color: blue;
}

.intro {
    font-style: italic;
}

#header {
    background-color: lightgrey;
}
```

Combinators

- **Descendant Selector (`ancestor descendant`)**: Selects all elements that are descendants of a specified element.
- **Child Selector (`parent > child`)**: Selects all elements that are direct children of a specified element.
- **Adjacent Sibling Selector (`element + element`)**: Selects an element that is directly after another element.
- **General Sibling Selector (`element ~ element`)**: Selects all elements that are siblings of a specified element.

Example:

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

ul > li {
    list-style-type: none;
}

h1 + p {
    margin-top: 0;
}
```

Pseudo-classes and Pseudo-elements

- **Pseudo-class**: Used to define a special state of an element.
  - Example: `:hover`, `:focus`, `:nth-child(n)`
- **Pseudo-element**: Used to style specific parts of an element.
  - Example: `::before`, `::after`, `::first-letter`, `::first-line`

Example:

```css
a:hover {
    color: orange;
}

p::first-line {
    font-weight: bold;
}
```

Box Model

The CSS **box model** describes the rectangular boxes generated for elements in the document tree.

Margin

- **Margin**: The space outside the border of an element.
- Can be set using: `margin-top`, `margin-right`, `margin-bottom`, `margin-left`, or `margin`.

Example:

```css
div {
    margin: 20px;
}
```

Border

- **Border**: The area between the element's padding and margin.
- Can be customized with: `border-width`, `border-style`, `border-color`.

Example:

```
p {
    border: 1px solid black;
}
```

Padding

- **Padding**: The space between the content and the border of an element.
- Can be set using: `padding-top`, `padding-right`, `padding-bottom`, `padding-left`, or `padding`.

Example:

```
div {
    padding: 10px;
}
```

Content

- **Content**: The actual content inside the box, such as text or images.

Positioning

Static Positioning

- Default positioning for elements. Elements are positioned according to the normal document flow.

Relative Positioning

- Position relative to its normal position. Other elements are not affected.

Example:

```
.relative {
    position: relative;
    top: 10px;
    left: 20px;
}
```

Absolute Positioning

- Positioned relative to the nearest positioned ancestor.

Example:

```
.absolute {
    position: absolute;
    top: 50px;
    right: 0;
}
```

Fixed Positioning

- Positioned relative to the browser window. It stays in the same place even when the page is scrolled.

Example:

```
.fixed {
    position: fixed;
    bottom: 0;
    width: 100%;
}
```

Sticky Positioning

- A hybrid between relative and fixed positioning. The element is positioned based on the user's scroll position.

Example:

```
.sticky {
    position: -webkit-sticky;
    position: sticky;
    top: 0;
}
```

Flexbox

- **Flexbox** is a one-dimensional layout method for laying out items in rows or columns.
- Parent container becomes a flex container by setting `display: flex`.

Example:

```
.container {
    display: flex;
```

```
    justify-content: space-between;
}
```

Grid

- **Grid** is a two-dimensional layout system for laying out items in rows and columns.
- A grid container is defined with `display: grid`.

Example:

```
.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 10px;
}
```

Responsive Design

- Making web pages look good on all devices (desktops, tablets, and phones).
- **Viewport** meta tag is crucial.

Example:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- Use percentages, ems, and rems for flexible layouts.

Animations

- CSS animations allow elements to transition between styles.
- Use `@keyframes` to define the animation.

Example:

```
@keyframes example {
    from {background-color: red;}
    to {background-color: yellow;}
}

.animated {
    animation-name: example;
    animation-duration: 4s;
}
```

Media Queries

- Media queries are used to apply different styles for different devices or screen sizes.

Example:

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

These notes cover essential CSS concepts. You can expand on each topic by adding more examples or explanations as you continue to learn!