Advanced CSS Properties

Estimated time: **30** minutes

Introduction

Cascading Style Sheets (CSS) is a language used to describe the presentation of a document written in HTML. CSS enhances the look and feel of web pages by allowing developers to apply styles to elements. This reading aims to explain various CSS properties and Flexbox properties, including their usage, examples and values, to help you build visually appealing web pages.

Objective

By the end of this reading you will:

- Gain a good understanding of various CSS properties.
- Learn about Flexbox properties.
- Understand how these properties affect HTML elements.
- Learn to use these properties effectively in web designs.

CSS Properties

1. text-decoration

Controls the decoration applied to text, such as underlines, lines above or through text.
This property can enhance the visual emphasis of text or indicate a change in text state (like a link being visited).

- **Example:** `text-decoration: underline;`
  - This applies an underline to text.
- **Values:**
  - `none`: No decoration.
  - `underline`: Underlines the text.
  - `overline`: Adds a line above the text.
  - `line-through`: Strikes through the text.
  - **Default value:** `none`

2. font-weight

Specifies the thickness or boldness of the font characters.
This property is used to make text stand out by adjusting its weight, which can enhance readability or create emphasis.

- **Example:** `font-weight: bold;`
  - This makes the font bold.
- **Values:**
  - `normal`: Normal weight.
  - `bold`: Bold weight.
  - `bolder`: Bolder than the parent element.
  - `lighter`: Lighter than the parent element.
  - `100-900`: Numeric values (100 is the thinnest, 900 is the thickest).
  - **Default value:** `normal`

3. margin

Sets the space around an element, outside its border. Margins create space between elements, helping to separate and organize content visually.

- **Example:** `margin: 20px;`
  - This adds a 20px margin on all sides of an element.
- **Values:**
  - `auto`: Automatically calculates the margin.
  - `length`: Specifies a fixed margin (e.g., 20px).
  - `percentage`: Specifies a margin relative to the containing element's width.
  - **Default value:** `0`

4. padding

Sets the space inside an element, between its content and its border.
Padding increases the area within the element, creating spacing between the content and the edges of the element.

- **Example:** `padding: 10px;`
  - This adds 10px padding inside an element.
- **Values:**
  - `length`: Specifies a fixed padding (e.g., 10px).
  - `percentage`: Specifies padding relative to the containing element's width.
  - **Default value:** `0`

5. float

Specifies whether an element should float to the left or right of its container.
Floating elements are taken out of the normal document flow, allowing text and inline elements to wrap around them.

- **Example:** `float: left;`
  - This makes an element float to the left.
- **Values:**
  - `none`: The element does not float.
  - `left`: The element floats to the left.
  - `right`: The element floats to the right.
  - **Default value:** `none`

6. border-radius

Defines the radius of the corners of an element's border box, creating rounded corners.
This property is used to soften the appearance of boxes, buttons, and other UI elements.

- **Example:** `border-radius: 10px;`
  - This rounds the corners of an element with a 10px radius.
- **Values:**
  - `length`: Specifies a fixed radius (e.g., 10px).
  - `percentage`: Specifies a radius relative to the element's dimensions.
  - **Default value:** `0`

7. width and height

Sets the width and height of an element.
These properties determine the size of the element, affecting how much space it occupies within its container.

- **Example:** `width: 200px; height: 100px;`
  - This sets the width to 200px and height to 100px.
- **Values:**
  - `auto`: The browser calculates the dimension.
  - `length`: Specifies a fixed dimension (e.g., 200px).
  - `percentage`: Specifies a dimension relative to the containing element's dimensions.
  - **Default value:** `auto`

8. position

Specifies the type of positioning method used for an element.
Positioning determines how an element is placed within its containing block and how it interacts with other elements.

- **Example:** `position: relative;`
  - This sets the element's position relative to its normal position.
- **Values:**
  - `static`: The default positioning.
  - `relative`: Positioned relative to its normal position.
  - `absolute`: Positioned relative to the nearest positioned ancestor.
  - `fixed`: Positioned relative to the browser window.
  - `sticky`: Switches between relative and fixed based on the scroll position.
  - **Default value:** `static`

Flexbox Properties

What is Flexbox?

Flexbox (Flexible Box Layout) is a CSS layout module designed to align and distribute space among items in a container, even when their size is unknown or dynamic. It allows you to create complex layouts with less code and provides greater control over alignment, direction, and order of items within a container.

1. display: flex

Defines an element as a flex container, enabling flexbox layout for its children.
This property makes it possible to use flexbox properties on the child elements to control their alignment, direction, and distribution.

- **Example:** `display: flex;`
  - This turns the container into a flex container.
- **Values:**
  - `flex`: Turns the element into a block-level flex container.
  - `inline-flex`: Turns the element into an inline-level flex container.
  - **Default value:** `block` (for the display property in general)

2. justify-content

Aligns flex items along the main axis, which is horizontal by default.
This property helps control the distribution of space between and around flex items within a container.

- **Example:** `justify-content: center;`
  - This centers items along the main axis.
- **Values:**
  - `flex-start`: Aligns items at the start of the main axis.

- `flex-end`: Aligns items at the end of the main axis.
- `center`: Centers items along the main axis.
- `space-between`: Distributes items evenly with the first item at the start and the last item at the end.
- `space-around`: Distributes items evenly with equal space around them.
- `space-evenly`: Distributes items evenly with equal space between them.
- **Default value:** `flex-start`

3. flex-direction

Defines the direction in which flex items are placed within the flex container.
This property determines whether the items are displayed in rows or columns, and can also reverse the order.

- **Example:** `flex-direction: column;`
  - This arranges items in a column.
- **Values:**
  - `row`: Default direction, items are placed in a row.
  - `row-reverse`: Items are placed in a row in reverse order.
  - `column`: Items are placed in a column.
  - `column-reverse`: Items are placed in a column in reverse order.
  - **Default value:** `row`

4. flex-wrap

Specifies whether flex items should wrap onto multiple lines when they overflow the container.
This property helps manage the layout when there are too many items to fit in one line.

- **Example:** `flex-wrap: wrap;`
  - This allows items to wrap onto multiple lines if necessary.
- **Values:**
  - `nowrap`: Items do not wrap.
  - `wrap`: Items wrap onto multiple lines.
  - `wrap-reverse`: Items wrap onto multiple lines in reverse order.
  - **Default value:** `nowrap`

5. align-items

Aligns flex items along the cross-axis, which is vertical by default.
This property is used to control the alignment of items within the flex container's cross-direction.

- **Example:** `align-items: stretch;`
  - This stretches items to fill the container's cross-axis.
- **Values:**
  - `flex-start`: Aligns items at the start of the cross-axis.
  - `flex-end`: Aligns items at the end of the cross-axis.
  - `center`: Centers items along the cross-axis.
  - `baseline`: Aligns items' baselines.
  - `stretch`: Stretches items to fill the container.
  - **Default value:** `stretch`

6. align-self

Aligns a single flex item along the cross-axis within its flex container.
This property allows individual items to override the alignment set by `align-items`.

- **Example:** `align-self: center;`
  - This centers the item within its flex container along the cross-axis.
- **Values:**
  - `auto`: Inherits the alignment from the parent container.
  - `flex-start`: Aligns the item at the start of the cross-axis.
  - `flex-end`: Aligns the item at the end of the cross-axis.
  - `center`: Centers the item along the cross-axis.
  - `baseline`: Aligns the item's baseline with the parent’s baseline.
  - `stretch`: Stretches the item to fill the container.
  - **Default value:** `auto`

7. flex-flow

Shorthand property for `flex-direction` and `flex-wrap`.
This property combines both the direction and wrapping behavior into a single declaration.

- **Example:** `flex-flow: row wrap;`
  - This arranges items in a row and allows them to wrap onto multiple lines.
- **Values:**
  - Combines values for `flex-direction` and `flex-wrap`.
  - **Default value:** `row nowrap`

8. justify-self

Aligns an individual flex item along the main axis.
This property affects the alignment of a single item relative to its container's main axis.

- **Example:** `justify-self: center;`
  - This centers a single item within its flex container along the main axis.
- **Values:**
  - `auto`: Inherits the alignment from the parent container.
  - `flex-start`: Aligns the item at the start of the main axis.
  - `flex-end`: Aligns the item at the end of the main axis.
  - `center`: Centers the item along the main axis.
  - `baseline`: Aligns the item's baseline with the parent's baseline.
  - `stretch`: Stretches the item to fill the container.
  - **Default value:** `auto`

Summary

CSS properties control the presentation of HTML elements on a web page. Understanding and using these properties correctly allows you to create visually appealing and well-structured web pages. Flexbox provides a powerful layout system to arrange and align items within a container efficiently, making it easier to design responsive and flexible layouts.

Author

Rajashree Patil