


CSS
Cascading Style Sheets

Class # 4

CSS Properties(Flexbox)

Flexbox is a CSS layout module that provides a flexible way to arrange and align elements within a container. It allows you to create responsive and dynamic layouts with ease.

Usage:

- **Flex Container:**
  - To use Flexbox, you need to define a flex container by setting its display property to flex or inline-flex.
  - Example: display: flex;
- **Flex Items:**
  - Elements inside a flex container are called flex items.
  - By default, flex items will arrange themselves in a single row (if the flex direction is set to row) or a single column (if the flex direction is set to column).Example: flex-direction: row;

CSS Properties(Flexbox)

- **Usage:**

- **Main Axis and Cross Axis:**
  - Flexbox operates along two axes: the main axis and the cross axis.
  - The main axis is determined by the flex direction (row or column).
  - The cross axis is perpendicular to the main axis.
  - The flex items can be aligned and distributed along both axes using various properties.
- **Justify Content:**
  - The justify-content property defines how flex items are positioned and aligned along the main axis.
  - It controls the distribution of space between and around the flex items.
  - Example: justify-content: center;
- **Align Items:**
  - The align-items property defines how flex items are positioned and aligned along the cross axis.
  - It controls the alignment of flex items when they do not fill the entire cross axis.
  - Example: align-items: center;

CSS Properties(Flexbox)

- **Usage:**

- **Flex Items Ordering:**
  - The order property allows you to specify the order in which flex items are displayed.
  - By default, flex items have an order of 0, and higher values will make them appear later in the order.
  - Example: order: 1;
- **Flex Item Sizing:**
  - The flex property allows you to specify the flexibility of flex items.
  - It consists of three values: flex-grow, flex-shrink, and flex-basis.
  - Example: flex: 1 0 auto;
- **Flex Wrap:**
  - By default, flex items will try to fit within a single line. If there is not enough space, they may shrink or overflow.
  - The flex-wrap property allows you to control whether flex items should wrap to a new line when they exceed the container's width.
  - Example: flex-wrap: wrap;

Example

Here's an example of how you can use Flexbox to create a simple navigation menu:

HTML:

```
<nav class="menu">
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Services</a>
  <a href="#">Contact</a>
</nav>
```

Example

CSS:

```
.menu {
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #f2f2f2;
  padding: 10px;
}

.menu a {
  margin: 0 10px;
  text-decoration: none;
  color: #333;
  font-weight: bold;
}
```

Example

In this example, the .menu class represents the flex container, and the <a> elements inside it are the flex items. We set the display property of the container to flex to activate Flexbox.

By using justify-content: center;, the flex items are horizontally centered within the container. And with align-items: center;, they are vertically aligned in the center.

The .menu a selector styles the individual links within the navigation. In this case, we set some margin, remove the default text decoration, set the text color to dark gray, and make the text bold.

By using Flexbox, the navigation menu is horizontally centered and the links are evenly spaced out, regardless of the screen size.

CSS Grid Layout

The CSS Grid Layout is a powerful two-dimensional layout system that allows you to create complex grid-based layouts in CSS. It provides a flexible way to arrange and align content within a container.

HTML:

```
<div class="grid-container">
  <div class="grid-item">Item 1</div>
  <div class="grid-item">Item 2</div>
  <div class="grid-item">Item 3</div>
  <div class="grid-item">Item 4</div>
</div>
```

CSS Grid Layout

```
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr; /* Two equal-width columns */
  grid-gap: 10px; /* Gap between grid items */
}

.grid-item {
  background-color: #f2f2f2;
  padding: 10px;
}
```

CSS Grid Layout

In this example, we create a grid container using the .grid-container class. By setting display: grid;, we activate the CSS Grid Layout on the container.

Using the grid-template-columns property, we define the number and size of the columns in the grid. In this case, we create two equal-width columns using the value 1fr (fractional unit). You can adjust the number and width of columns as per your requirement.

The grid-gap property specifies the gap or spacing between the grid items.

The .grid-item class represents the individual grid items within the grid container. We set a background color and padding to style the grid items.

By using the CSS Grid Layout, you can create various complex grid-based layouts with ease, allowing you to precisely control the placement and alignment of elements on the web page.