Introduction to Flexbox

Web Development Essentials - Session 7

Session Overview

- **Learning Goals for Today**:
  - Understand the Flexbox layout model and how it simplifies page layout
  - Learn how to align and distribute elements within a Flexbox container using properties like `justify-content` and `align-items`

What is Flexbox?

- **Definition**: Flexbox, or the Flexible Box Layout, is a CSS layout model designed to arrange elements in one-dimensional space—either horizontally or vertically.
- **Purpose**: It makes it easier to design flexible, responsive layouts that adapt to different screen sizes without using floats or positioning.
- **Key Concepts**:
    - **Flex Container**: The parent element where Flexbox is applied.
    - **Flex Items**: The direct children of the flex container that are laid out according to the Flexbox rules.

Flexbox Layout Model Overview

Key Features of Flexbox:

- One-dimensional layout: Flexbox is perfect for layouts in a single direction (row or column).
- Dynamic sizing: Flex items can grow or shrink based on available space.
- Aligning and distributing space: Flexbox offers powerful alignment tools.

Basic Flexbox Structure:

- **Flex Container**: The parent element with `display: flex;`.
- **Flex Items**: The child elements inside the container.

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

Flexbox Axis

Main Axis vs Cross Axis:

- **Main Axis**: Defined by the `flex-direction` property. Determines how flex items are placed within the flex container (row or column).
- **Cross Axis**: Perpendicular to the main axis.
- Example:
  - `flex-direction: row;` → Main axis is horizontal.
  - `flex-direction: column;` → Main axis is vertical.

Flexbox Example Layout

```css
.container {
  display: flex;
  flex-direction: row;
}
.item {
  flex: 1;
  padding: 10px;
  background-color: lightblue;
}
```

In this example, the flex container arranges items in a row, and each item stretches to take up equal space.

Aligning Items Using Flexbox

Aligning Flex Items:

- Flexbox offers several properties to align and distribute space among flex items:
  - **justify-content**: Aligns items along the main axis (horizontal for `row`, vertical for `column`).
  - **align-items**: Aligns items along the cross axis (perpendicular to the main axis).

justify-content Property

Purpose: Controls how space is distributed along the main axis.

Common Values:

1. `flex-start`: Align items to the start of the main axis.
2. `flex-end`: Align items to the end of the main axis.
3. `center`: Center items along the main axis.
4. `space-between`: Distribute items with equal space between them.
5. `space-around`: Distribute items with equal space around them.

```
.container {
  display: flex;
  justify-content: center;
}
```

Combining justify-content and align-items

Aligning Both Axes:

- You can use both properties together to control alignment on the main axis (`justify-content`) and cross axis (`align-items`).

```
.container {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
}
```

flex-direction Property

Purpose: Defines the direction in which flex items are laid out.

Common Values:

1. `row`: Flex items are placed in a horizontal row.
2. `row-reverse`: Flex items are placed in reverse order in a horizontal row.
3. `column`: Flex items are placed in a vertical column.
4. `column-reverse`: Flex items are placed in reverse order in a vertical column.

```
.container {
  display: flex;
  flex-direction: column;
}
```

flex-wrap Property

Purpose: Controls whether flex items wrap onto multiple lines.

Common Values:

1. `nowrap`: Flex items are displayed on a single line (default).
2. `wrap`: Flex items wrap onto multiple lines if they don't fit in a single line.
3. `wrap-reverse`: Flex items wrap in reverse order.

```
.container {
  display: flex;
  flex-wrap: wrap;
}
```

Flexbox in Action: Example

```css
.container {
  display: flex;
  justify-content: space-around;
  align-items: center;
  flex-wrap: wrap;
}
.item {
  background-color: lightcoral;
  padding: 20px;
  margin: 10px;
  flex-basis: 200px;
}
```

Result: Flex items are evenly spaced, centered along the cross axis, and wrap onto multiple lines if necessary.

Hands-On Activity

Goal: Use Flexbox to create a responsive layout.

- Align items using `justify-content` and `align-items`.
- Apply Flexbox to create a row or column of items that adjust with the screen size.

Instructions:

- Create a flex container with 4-5 items.
- Experiment with different values for `justify-content`, `align-items`, and `flex-direction`.

Summary

What We Learned Today:

- What Flexbox is and how it helps create flexible, responsive layouts.
- How to use key Flexbox properties like `justify-content` and `align-items`.
- How to arrange items along the main and cross axes with Flexbox.

Questions?

Q&A Session

- Any questions before we wrap up?

Thank You & See You in the Next Class!