CSS – Flexbox

Agenda

In this session, we will discuss:

- Flex Grow
- Flex Shrink
- Flex Basis
- Flex
- Media Queries
- Creating a Responsive Web Page using Flexbox

Flex Grow

- The flex-grow property specifies the ability of a flex item to grow within the available space in a flex container.
- It accepts a non-negative unitless value, indicating the factor by which the item should grow relative to other flex items.
- A value of 0 means the item will not grow and it will maintain its initial size.
- A value greater than 0 means the item will grow proportionally. For example, if one item has a flex-grow of 2 and another has a flex-grow of 1, the first item will grow twice as much as the second item.
- Example:

```
    .flex-item
        { flex-grow: 1;}
```

- Here, each item has an equal chance to grow within the available space.

Flex Shrink

- The flex-shrink property specifies the ability of a flex item to shrink when there's insufficient space in the flex container.
- It accepts a non-negative unitless value, indicating the factor by which the item should shrink relative to other flex items.
- A value of 0 means the item will not shrink and it will maintain its initial size.
- A value greater than 0 means the item will shrink proportionally. For example, if one item has a flex-shrink of 2 and another has a flex-shrink of 1, the first item will shrink twice as much as the second item.
- Example:

```
.flex-item {
        flex-shrink: 1;}
```

- Here, each item has an equal chance to shrink if needed.

Flex Basis

- The flex-basis property specifies the initial size of a flex item before any remaining space is distributed.
- It can take values like auto, a fixed size (e.g., 200px), or a percentage (e.g., 30%).
- When using auto, the initial size is determined by the item's content or its specified width, if any.
- When specified as a fixed size or percentage, the item will have a predetermined initial size.
- Example:

```
.flex-item {
        flex-basis: 200px; /* Initial size of the item is 200 pixels. */
    }
```

Flex

- **Flex** property is a shorthand for **flex-grow**, **flex-shrink,** and **flex-basis**.

```
.flex-item {
     flex: 1 1 auto; /* Equivalent to flex-grow: 1; flex-shrink: 1; flex-basis: auto; */
 }
```

- These properties allow you to create flexible and responsive layouts within a flex container by controlling how items grow, shrink, and initially size themselves in relation to one another.

Media Queries

- Media queries in CSS allow you to apply different styles to your web content based on the characteristics of the user's device or screen.
- This enables you to create responsive designs that adapt to various screen sizes, orientations, and other conditions.
- Media queries are typically used within your CSS stylesheets and are written using the @media rule.
- Basic Syntax:

```
    @media media_type and (media_feature) {
  /* CSS rules to apply when the media query matches */
}
```

Media Queries

- media_type: Specifies the type of media you are targeting, such as screen, print, speech, etc.
- media_feature: Represents the condition or feature you want to test, like screen width, height, orientation, resolution, etc.
- Inside the curly braces, you define the CSS rules that should be applied if the specified condition is met.
- For example:

```
   /* Apply styles when the screen height is 600 pixels or less */
@media screen and (max-height: 600px) {
  /* CSS rules for screens shorter than or equal to 600px in height */
}
```

Create a Responsive Web Page using Flexbox

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive Flexbox Layout</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <h1>Responsive Flexbox Layout</h1>
  </header>
```

Create a Responsive Web Page using Flexbox

```
<div class="container">
    <div class="content-section">
        <div class="card">
            <img src="https://picsum.photos/200" alt="">
            <h2>Card One</h2>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Necessitatibus, 
ea!</p>
        </div>
        <div class="card">
            <img src="https://picsum.photos/200" alt="">
            <h2>Card Two</h2>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Necessitatibus, 
ea!</p>
        </div>
```

Create a Responsive Web Page using Flexbox

```
<div class="card">
            <img src="https://picsum.photos/200" alt="">
            <h2>Card Three</h2>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Necessitatibus,
ea!</p>
        </div>

    </div>
  </div>
</body>
</html>
```

Create a Responsive Web Page using Flexbox

styles.css

```
*{
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
  font-family: sans-serif;
}
header{
  text-align: center;
  margin-top: 3rem;
  margin-bottom: 5rem;
}
```

```
.content-section{
  display: flex;
  justify-content: center;
  align-items: center;
  text-align: center;

}
```

Create a Responsive Web Page using Flexbox

```
.content-section .card{
  flex: 1;
  padding: 10px;
  box-shadow: 0px 2px 8px 0px
rgba(0,0,0,0.2);
  margin: 2rem;
}

.content-section .card img{
  width: 100%;
  height: auto;
}
```

```
.content-section .card p{
  font-size: 16px;
}

@media screen and (max-width:
768px) {
  .content-section{
    flex-direction: column;
  }
}
```

Create a Responsive Web Page using Flexbox

- In the index.html, three cards are created using the **card** class inside the class **content-section**.
- Each **card** contains an image, a heading2, and a paragraph.
- The **content-section** class is displayed as flex.
- By default, the **flex-direction** is row.
- In the media queries, we have set the **flex-direction** property to column when the max-width of the screen is 768px.
- This implies that when the size of the screen becomes 768px or smaller, the cards will get stacked under each other.
- This makes the page mobile-responsive.

Summary

Here’s a brief recap:

- The flex-grow and flex-shrink property specifies the ability of a flex item to grow or shrink within the available space in a flex container.
- The flex-basis property specifies the initial size of a flex item before any remaining space is distributed.
- Media queries in CSS allow you to apply different styles to your web content based on the characteristics of the user's device or screen.
- We illustrated how to create a responsive web page using flexbox.