Treasa Geethu P J
https://github.com/geethugithub7

HERE'S WHAT WE'LL COVER:

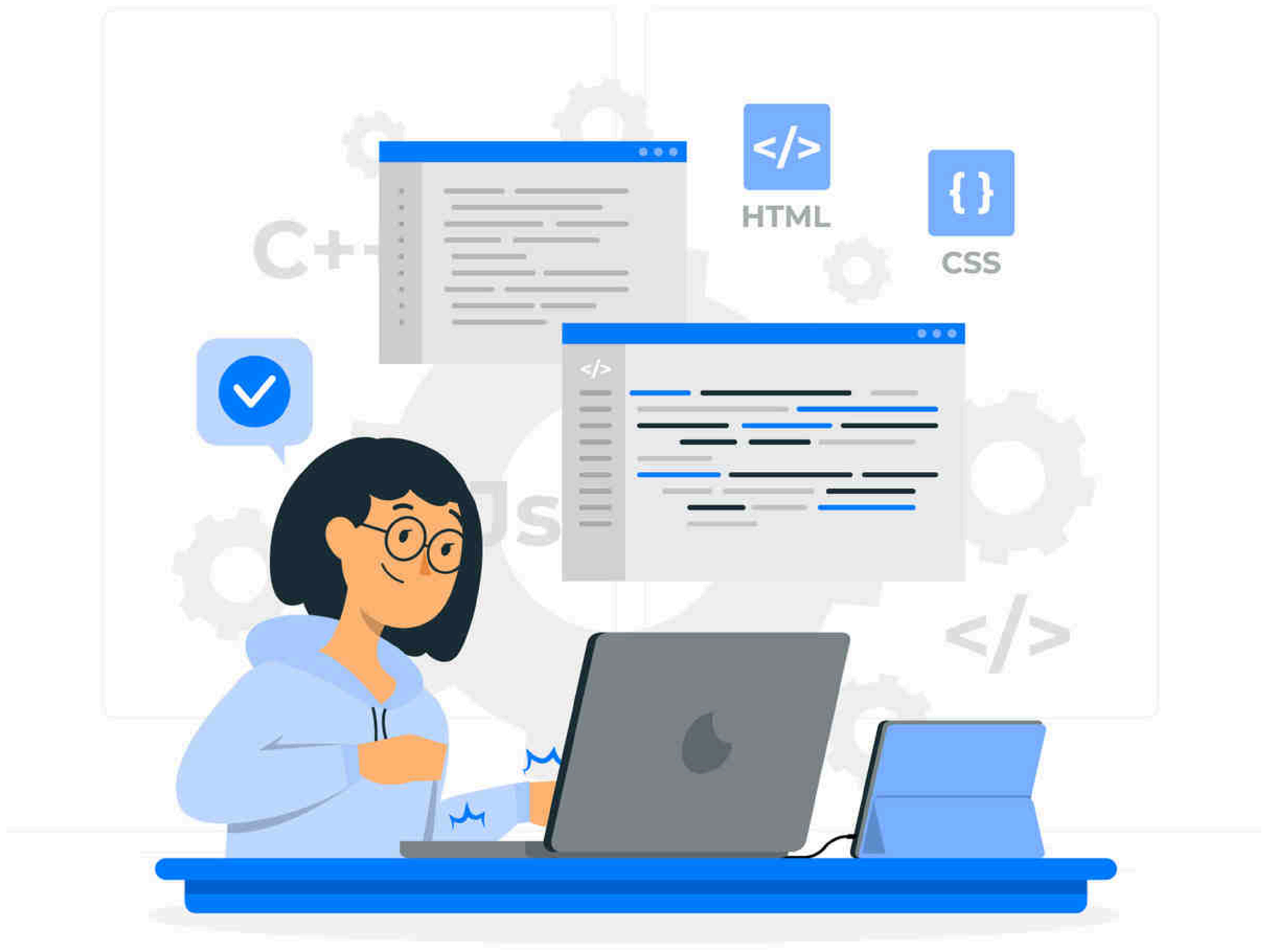
- CSS Flexbox
- Understanding flexbox layout
- Creating flexible layouts

Week 2: CSS Flexbox

Day 6

Treasa Geethu P J
https://github.com/geethugithub7

Flexbox is a powerful layout model in CSS that enables you to arrange elements horizontally or vertically in a flexible and responsive way.



Week 2: CSS Flexbox

Day 6

Here's a breakdown of key concepts in CSS Flexbox:

1. The Flex Container:

- Imagine a container element that holds all the elements you want to arrange using Flexbox. This container element needs the **'display: flex'** property applied to activate flexbox mode.

2. Flex Items:

- These are the individual elements (like divs, paragraphs, images) that you want to position within the flex container.

3. Flexbox Properties:

- Flex Direction (flex-direction): This property controls the main axis along which the flex items are laid out.
    - **row (default)** - Arranges items horizontally from left to right.
    - **column** - Arranges items vertically from top to bottom.
    - **row-reverse** - Arranges items horizontally from right to left.
    - **column-reverse** - Arranges items vertically from bottom to top.

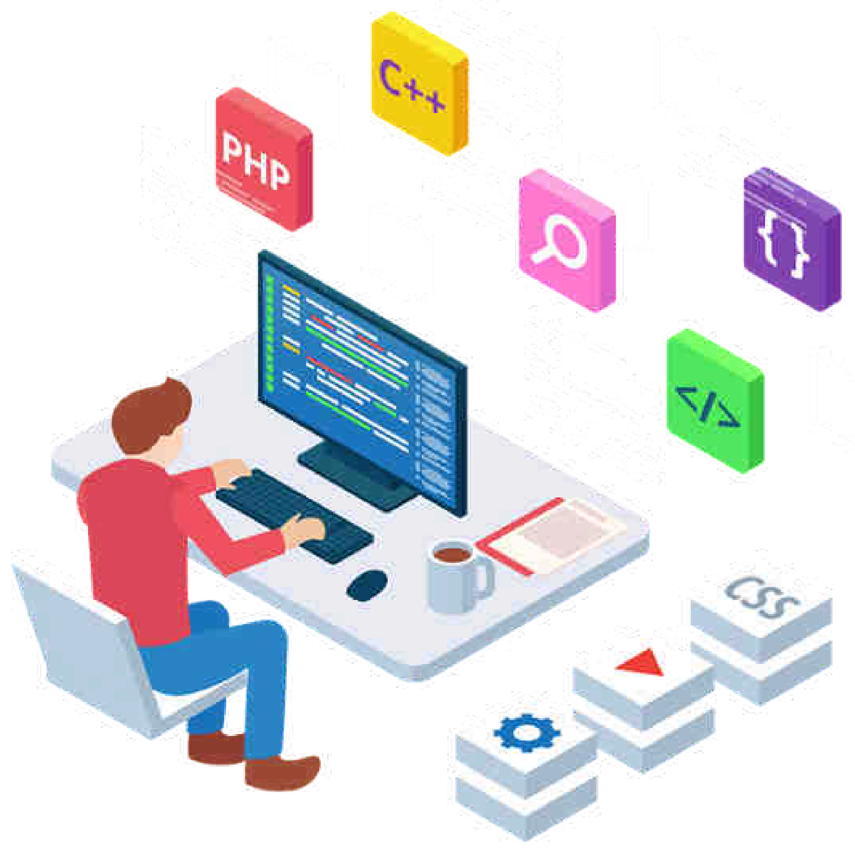
Treasa Geethu P J
https://github.com/geethugithub7

- **Justify Content (justify-content):** This property controls how flex items are distributed along the main axis of the container.
  - **flex-start (default)** - Aligns items to the beginning of the container.
  - **flex-end** - Aligns items to the end of the container.
  - **center** - Centers items within the container.
  - **space-between** - Distributes items evenly with space in between.
  - **space-around** - Distributes items evenly with space around them (including the beginning and end of the container).

- **Align Items (align-items):** This property controls how flex items are aligned along the cross axis (perpendicular to the main axis).
  - **flex-start (default)** - Aligns items to the top for row or left for column.
  - **flex-end** - Aligns items to the bottom for row or right for column.
  - **center** - Centers items along the cross axis.
  - **baseline** - Aligns items to the baseline of their text content.
  - **stretch (default for flex-direction**: column) - Stretches items to fill the entire height of the container.

Treasa Geethu P J
https://github.com/geethugithub7

- **Additional Flexbox Properties:**
  - **flex-grow:** Sets the flex grow factor, allowing items to grow to fill available space proportionally.
  - **flex-shrink:** Sets the flex shrink factor, allowing items to shrink if there's not enough space.
  - **flex-basis:** Sets the default size of flex items before any grow or shrink is applied.
- **Benefits of Flexbox:**
  - **Flexibility and Responsiveness:** Easily create layouts that adapt to different screen sizes and devices.
  - **Simplified Code:** Less complex code compared to traditional floats for layout.
  - **Alignment Power:** Precise control over how elements are aligned within the container.



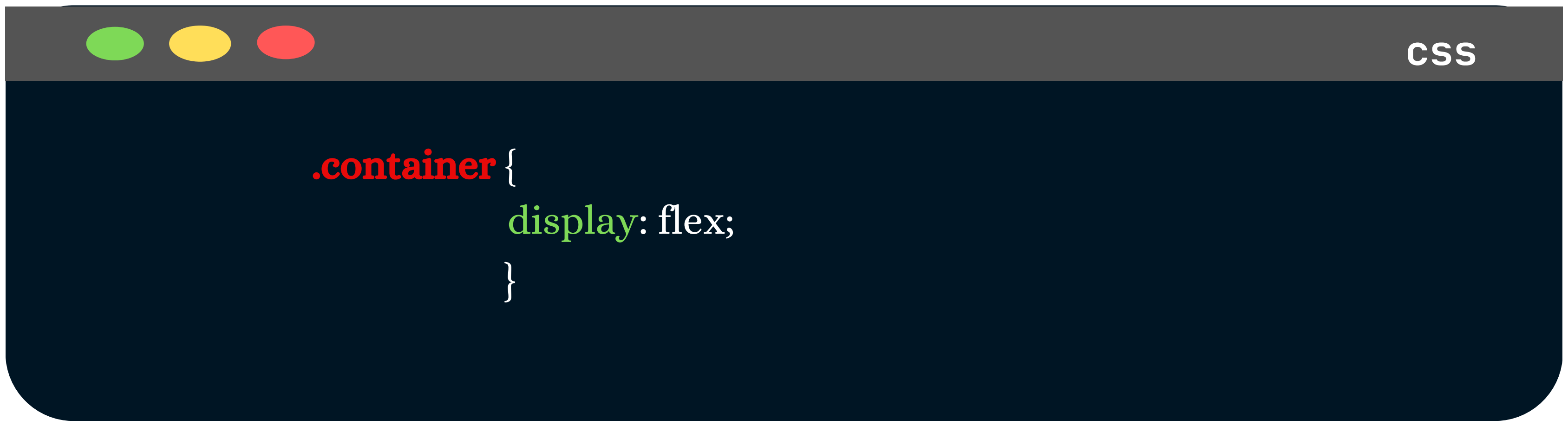
Treasa Geethu P J
https://github.com/geethugithub7

Creating flexible layouts with CSS Flexbox involves utilizing the various properties and concepts of Flexbox to design dynamic and responsive web layouts.

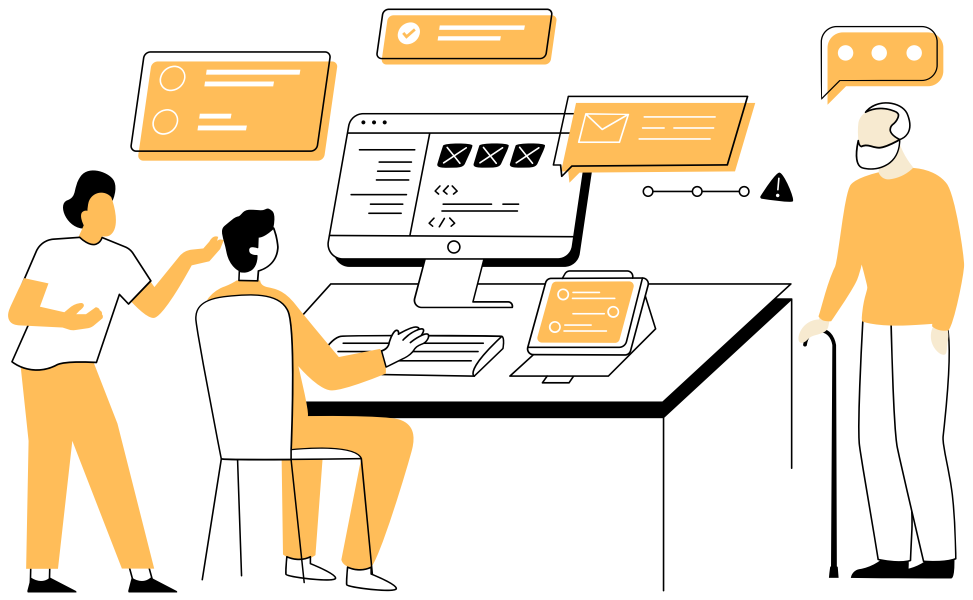
Here's a step-by-step guide to creating flexible layouts using Flexbox

1. Set up the Flex Container:

Start by creating a container element and applying **‘display’: ‘flex’;** or **‘display’: ‘inline-flex’**; to it. This establishes it as a flex container.



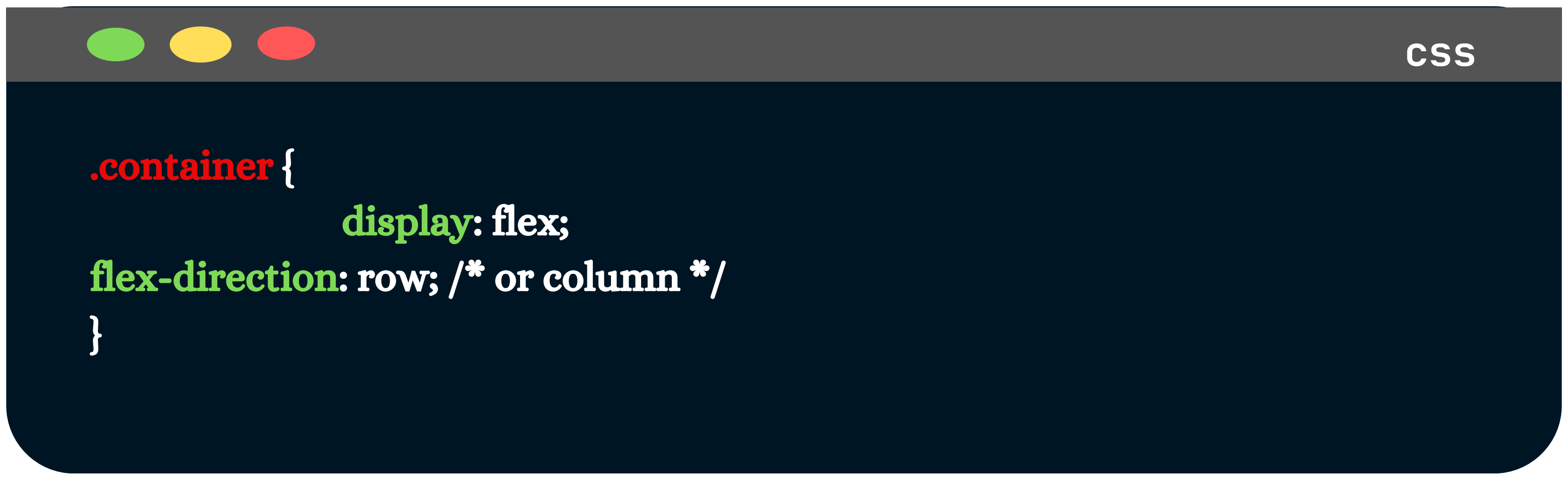
```css
.container {
        display: flex;
        }
```



Treasa Geethu P J
https://github.com/geethugithub7

2. Choose the Direction:

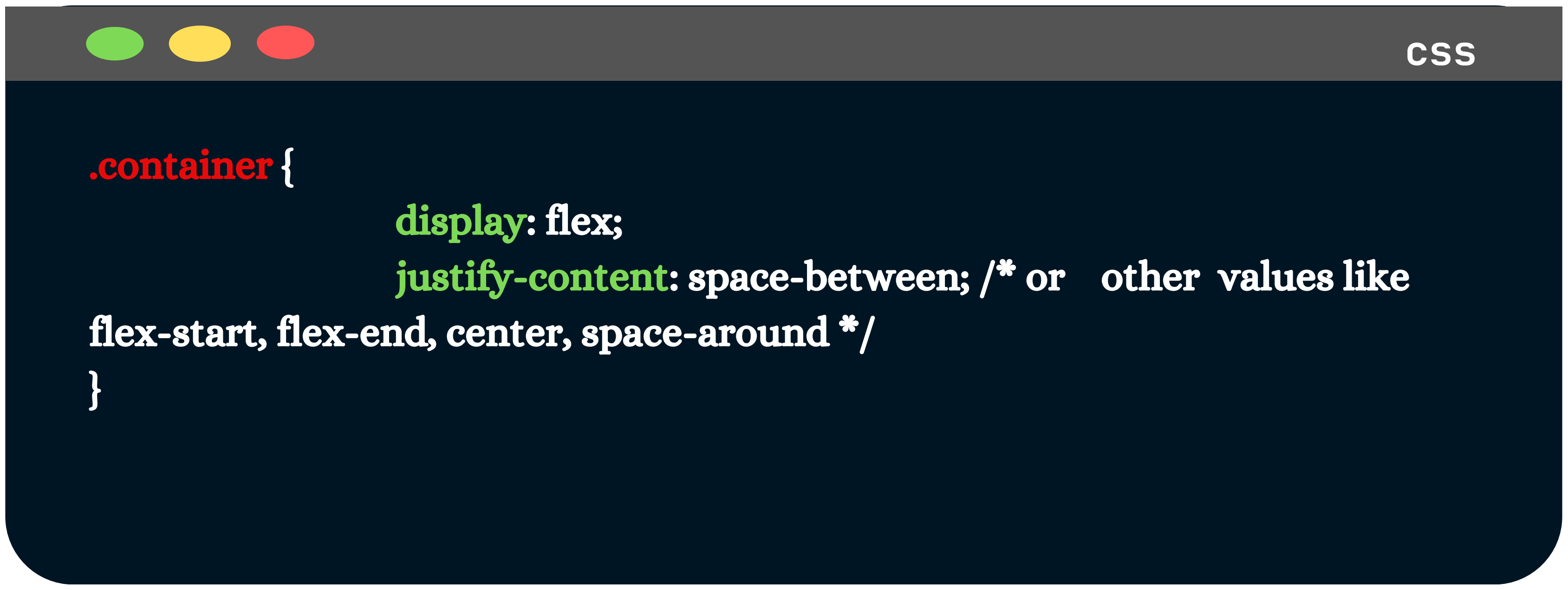
Decide whether you want your layout to be in rows or columns by setting the ‘**flex-direction**’ property.



```css
.container {
                    display: flex;
flex-direction: row; /* or column */
}
```

3. Distribute Space:

Use ‘**justify-content**’ to control how flex items are aligned along the main axis.



```css
.container {
                        display: flex;
                        justify-content: space-between; /* or   other  values like
flex-start, flex-end, center, space-around */
}
```

4. Align Items:

Use **‘align-items’** to control how flex items are aligned along the cross axis.

```css
.container {
                display: flex;
                align-items: center; /* or other values like flex-start, flex-end,
center, baseline, stretch */
}
```

5. Make Items Flexible:

Adjust the flex properties of individual flex items to control their flexibility.

```css
.item {
  flex : 1; /* This will make all items grow and shrink equally
to fill the available space */
}
```

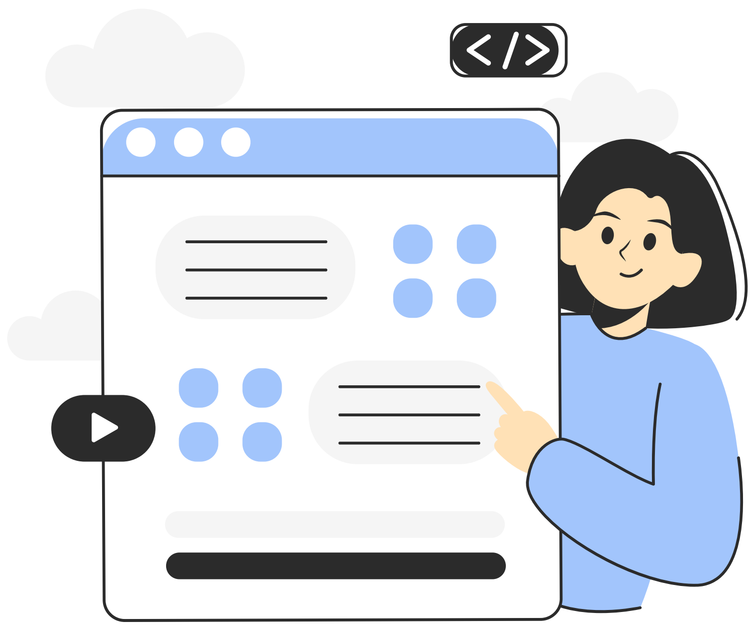
Treasa Geethu P J
https://github.com/geethugithub7

6. Wrap Items (Optional):

If you want your flex items to wrap onto multiple lines, use **‘flex-wrap’**.

css

```
.container {
                display: flex;
                flex-wrap: wrap; /* or other values like nowrap, wrap-reverse
*/
}
```



Treasa Geethu P J
https://github.com/geethugithub7

Example: Creating a Flexible Card Layout:

Let's imagine you want to create a layout with multiple product cards displayed side-by-side. Here's how Flexbox can help:

HTML

```html
<div class="product-card-container">  <div class="product-card">...
</div>
  <div class="product-card">...</div>
  <div class="product-card">...</div>
</div>
```

CSS

```css
.product-card-container {
  display: flex;  /* Activate Flexbox */
  flex-direction: row;  /* Arrange cards horizontally */
  justify-content: space-between; /* Distribute cards evenly */
  /* Add styling for margins/padding as needed */
}

.product-card {
  /* Individual product card styles */
}
```