Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

<img> Element

- **Behavior**: By default, it is an inline element.
- **Special Characteristics**: As a replaced element, it can accept width and height properties.
- **Usage**: Does not need display: inline-block to have width and height properties applied.

<button> Element

- **Behavior**: By default, it is an inline element.
- **Special Characteristics**: As a form control, it can accept width and height properties.
- **Usage**: Does not need display: inline-block to have width and height properties applied.

CSS Flexbox

The CSS Flexible Box Layout, commonly known as Flexbox, is a powerful layout model that provides an efficient way to design complex layouts with minimal code. It allows for the distribution of space within a container and aligns items in various ways.

Key Concepts of Flexbox

1. **Flex Container**
    - **Definition:** The parent element that holds flex items.
    - **Properties:**
        - display: flex: Defines a flex container.

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

Flex Items

- **Definition:** The direct children of a flex container.

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

Flex Container Properties

1. **flex-direction**
    - **Purpose:** Defines the direction of the main axis.
    - **Values:**

- row: Default, items are placed horizontally.
- row-reverse: Items are placed horizontally in reverse order.
- column: Items are placed vertically.
- column-reverse: Items are placed vertically in reverse order.

```
.flex-container {
    flex-direction: row;
  }
```

justify-content

- **Purpose:** Aligns flex items along the main axis.
- **Values:**
  - flex-start: Items align to the start of the container.
  - flex-end: Items align to the end of the container.
  - center: Items align to the center of the container.
  - space-between: Items are evenly distributed with the first item at the start and the last item at the end.
  - space-around: Items are evenly distributed with equal space around them.

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

align-items

- **Purpose:** Aligns flex items along the cross axis.
- **Values:**
  - flex-start: Items align to the start of the cross axis.
  - flex-end: Items align to the end of the cross axis.
  - center: Items align to the center of the cross axis.
  - baseline: Items are aligned along their baseline.
  - stretch: Items stretch to fill the container (default).

```
.flex-container {
    align-items: center;
  }
```

flex-wrap

- **Purpose:** Controls whether flex items should wrap onto multiple lines.
- **Values:**
  - nowrap: All flex items will be on one line (default).
  - wrap: Flex items will wrap onto multiple lines.
  - wrap-reverse: Flex items will wrap onto multiple lines in reverse order.

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

align-content

- **Purpose:** Aligns a flex container's lines within the container when there is extra space along the cross axis.
- **Values:**
  - flex-start: Lines are packed to the start of the container.
  - flex-end: Lines are packed to the end of the container.
  - center: Lines are packed to the center of the container.
  - space-between: Lines are evenly distributed.
  - space-around: Lines are evenly distributed with equal space around them.
  - stretch: Lines stretch to fill the container (default).

```
.flex-container {
  align-content: space-between;
}
```

Flex Item Properties

1. **order**
   - **Purpose:** Defines the order of a flex item relative to the other flex items in the same container.
   - **Default Value:** 0

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

flex-grow

- **Purpose:** Defines the ability of a flex item to grow if necessary.
- **Default Value:** 0

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

flex-basis

- **Purpose:** Defines the initial main size of a flex item.
- **Default Value:** auto

```
.flex-item {
  flex-basis: 100px;
}
```

Question: What is the purpose of the display: flex property in CSS?

- **Answer:** The display: flex property defines a flex container, enabling a flexible layout structure where the child elements (flex items) can be aligned and distributed within the container using various flex properties.

Question: How does flex-direction affect the layout of flex items?

- **Answer:** The flex-direction property specifies the direction of the main axis, which determines how flex items are placed in the container. Values include row (default, horizontal), row-reverse, column (vertical), and column-reverse.

Question: What is the difference between justify-content and align-items in a flex container?

- **Answer:** justify-content aligns flex items along the main axis (horizontal by default) and controls the spacing between them. align-items aligns flex items along the cross axis (vertical by default) and controls their positioning within the flex container.

Question: How do you make flex items wrap onto multiple lines?

- **Answer:** You use the flex-wrap property with the value wrap. This allows flex items to wrap onto multiple lines if there is not enough space in a single line.

Question: Explain the order property in Flexbox.

- **Answer:** The order property specifies the order of flex items within a flex container. The default value is 0. Flex items with higher order values will be placed after items with lower values.

Question: How can you control the growth and shrinking of flex items?

- **Answer:** The flex-grow property controls the ability of a flex item to grow relative to the other items, while the flex-shrink property controls the ability of a flex item to shrink relative to the other items. Both properties accept numerical values.

Question: What is flex-basis and how does it differ from width?

- **Answer:** flex-basis specifies the initial main size of a flex item before space distribution based on flex-grow and flex-shrink. It differs from width because it works in conjunction with the flex properties to determine the final size of the item.

Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

CSS Grid:

CSS Grid Layout is a powerful layout system available in CSS that allows developers to create complex, responsive web layouts with ease. It is a two-dimensional system, meaning it can handle both columns and rows, unlike Flexbox, which is primarily a one-dimensional system.

Key Concepts of CSS Grid

1. **Grid Container**
    - **Definition:** The parent element that defines a grid context for its children.
    - **Property:** display: grid or display: inline-grid.

```
.grid-container {
    display: grid;
  }
```

Grid Items

- **Definition:** The direct children of a grid container.

```
<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>
```

Grid Container Properties

1. **grid-template-columns and grid-template-rows**
    - **Purpose:** Defines the number and size of columns and rows in the grid.
    - **Values:** Can be specified in pixels, percentages, fractions (using fr), and other units.

```
.grid-container {
    display: grid;
    grid-template-columns: 100px 200px 1fr;
    grid-template-rows: 100px 200px;
  }
```

grid-column-gap and grid-row-gap

- **Purpose:** Specifies the gap (gutter) size between columns and rows.

```
.grid-container {
    display: grid;
    grid-column-gap: 10px;
    grid-row-gap: 20px;
```

```
    }
```

grid-gap

- **Purpose:** A shorthand for setting both grid-column-gap and grid-row-gap.

```
.grid-container {
      display: grid;
      grid-gap: 10px 20px; /* row-gap column-gap */
    }
```

justify-items and align-items

- **Purpose:** Aligns grid items along the inline (justify) and block (align) axes.
- **Values:** start, end, center, stretch

```
 .grid-container {
      display: grid;
      justify-items: center;
      align-items: stretch;
    }
```

justify-content and align-content

- **Purpose:** Aligns the entire grid within the container along the inline (justify) and block (align) axes.
- **Values:** start, end, center, stretch, space-between, space-around, space-evenly

```
.grid-container {
      display: grid;
      justify-content: space-between;
      align-content: center;
    }
```

Grid Item Properties

1. **grid-column and grid-row**
    - **Purpose:** Specifies the start and end lines for grid items.

```
 .grid-item {
      grid-column: 1 / 3; /* spans from column line 1 to 3 */
      grid-row: 1 / 2; /* spans from row line 1 to 2 */
    }
```

justify-self and align-self

- **Purpose:** Aligns an individual grid item along the inline (justify) and block (align) axes.
- **Values:** start, end, center, stretch

```
.grid-item {
    justify-self: end;
    align-self: center;
  }
```

Interview Questions and Answers

Question: What is the CSS Grid Layout?

- **Answer:** The CSS Grid Layout is a two-dimensional layout system for the web that allows developers to design complex layouts using a grid-based approach. It can handle both columns and rows, unlike Flexbox, which is primarily a one-dimensional layout system.

Question: How do you define a grid container?

- **Answer:** A grid container is defined using the display: grid or display: inline-grid property on a parent element. This establishes a grid formatting context for its children.

Question: What does the grid-template-columns property do?

- **Answer:** The grid-template-columns property specifies the number and size of columns in the grid. It can accept values in pixels, percentages, fractions (using fr), and other units.

Question: How can you create a gap between grid items?

- **Answer:** You can create gaps between grid items using the grid-column-gap, grid-row-gap, or the shorthand grid-gap properties.

Question: Explain the grid-area property.

- **Answer:** The grid-area property assigns a grid item to a named grid area defined by grid-template-areas or specifies the start and end lines for both rows and columns

Question: What is the difference between justify-items and justify-content in Grid Layout?

- **Answer:** justify-items aligns grid items along the inline axis within their grid area, while justify-content aligns the entire grid within the grid container along the inline axis.

Question: How do you make a grid item span multiple rows or columns?

- **Answer:** You can make a grid item span multiple rows or columns using the grid-column and grid-row properties.