Lesson:

Introduction to Grid

PW SKILLS

Topics Covered

- Introduction to Grid
- Simple example of Grid
- Advantages of Grid

Introduction to Grid

Grid is a 2-dimensional (rows and columns) layout system that allows developers to construct complex 2D layouts easily.

Syntax

Grid is a 2-dimensional (rows and columns) layout system that allows developers to construct complex 2D layouts easily.

```
Unset
.element {
  display: grid;
}
```

Here, **display: grid** enables grid layout. But it does not change anything to UI until we use other grid properties to element.

Structure of Grid

Like every other layout, the grid also follows some structure, let's talk about it here,

Consider the below code, it defines a div, **.container** and it has four child elements.

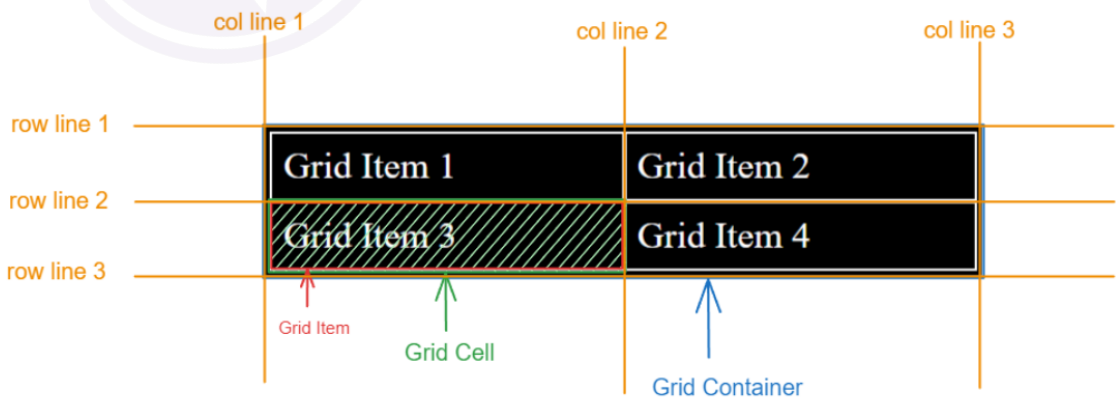
```
Unset
<div class="container">
      <div>Grid Item 1</div>
      <div>Grid Item 2</div>
      <div>Grid Item 3</div>
      <div>Grid Item 4</div>
</div>
```

Apply some CSS, to show the structure of grid clearly,

```
Unset
.container {
    display: grid;
    grid-template-columns: auto auto;
    border: 2px solid black;
}

.container > div {
    padding: 5px;
    background-color: black;
    color: white;
    border: 1px solid white;
}
```

Browser Output



- **Grid Container:** The outermost element, where display: grid is applied, as shown in figure.
- **Grid Item:** Every direct child of a grid container is called a grid item, as shown in the figure.
- **Grid Cell:** This is the smallest unit of a grid container, as shown in the figure.
- **Col lines:** There are grid lines, representing starting and ending positions of a column.
- **Row lines:** There are grid lines, representing starting and ending positions of a row.

Note: A Grid item can span over one or many grid cells, we will talk about it in later sections.

Example of a Grid
Let's take a simple example, to understand how grid layout works.

The below code will display the page layout as shown in **Figure 1**, and we want to arrange different sections(header, footer, nav, main) in such a way that it looks like **Figure 2.**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Grid Layout</title>
    <style>
      body {
        min-height: 100vh;
        font-size: 50px;
      }

      header {
        background-color: royalblue;
        color: white;
        padding: 5px;
      }

      nav {
        background-color: darkorange;
        color: white;
        padding: 5px;
      }

      main {
        background-color: burlywood;
        color: black;
        padding: 5px;
      }

      footer {
        background-color: tomato;
        color: white;
        padding: 5px;
      }
    </style>
  </head>
  <body>
    <header>Header</header>
    <nav>Nav</nav>
    <main>Main</main>
    <footer>@footer</footer>
  </body>
</html>
```

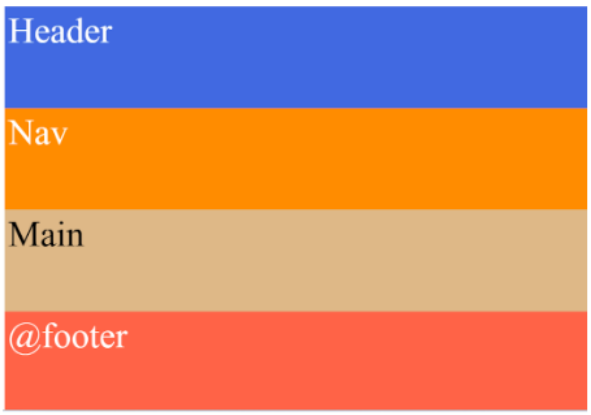


Figure 1: Original

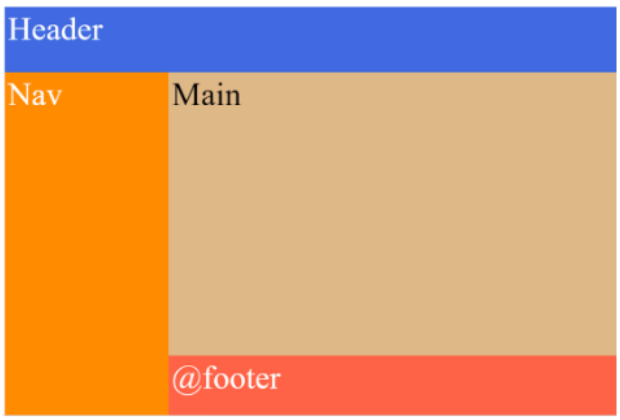


Figure 2: Expected Output

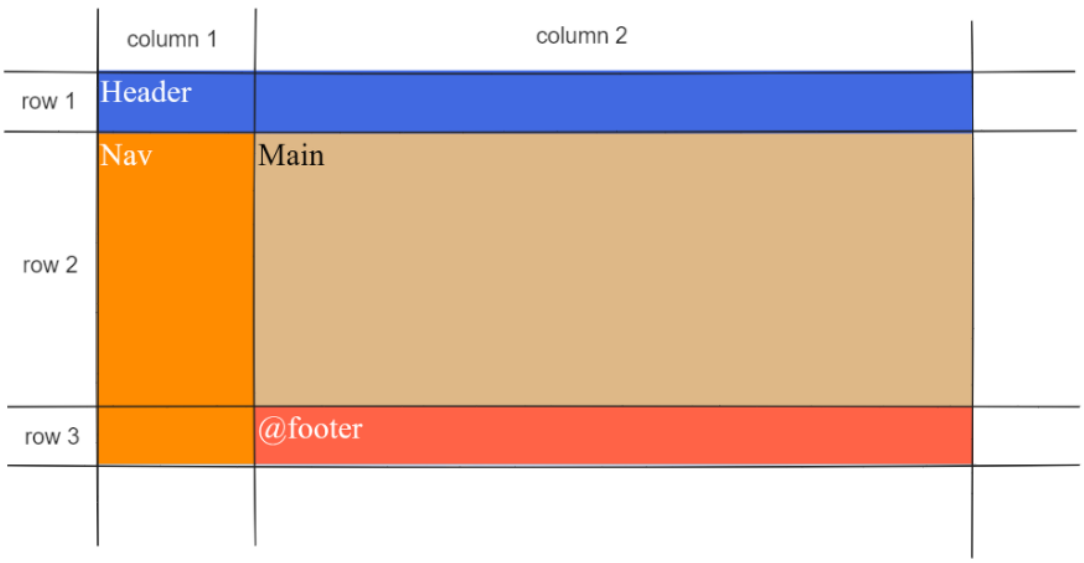
Before moving to modify code, let's understand a few terms,

1. **Grid container** - It is like a parent element, whose children we want to arrange. In our example, the body is our grid container.
2. **Grid items** - It is a child of a grid container. In our example, the header, footer, nav, and main are grid items.

Let's modify CSS, and add **display: grid** to grid-container **(body)**, it will allow us to use all grid properties.

```
Unset
body {
   display: grid;
}
```

As per the expected output, we can see there are three rows, and two columns required,



let's define them using grid container properties,

```
Unset
body {
   display: grid;
   grid-template-columns: 5em 1fr;
   grid-template-rows: 2em 1fr 2em;
}
```

In the above code,

- **grid-template-columns:** 5em 1fr creates 2 rows. **row1** with width of **5em**, and **row2** will have remaining available space (1fr).
- **grid-template-rows:** 2em 1fr 2em creates 3 columns. **column1** with height of **2em**, **column2** with available space, and **column3** with **2em.**

Note: The unit fr.
In CSS (Cascading Style Sheets), **fr** is a unit of measurement used in grid and flexbox layouts to define the size of a grid item or flex item. The **fr** stands for **fraction** and it represents a fraction of the available space within a container.

For example, if you have a grid container with three columns, and you set the column widths to "1fr 2fr 1fr," the available horizontal space will be divided into four equal parts, and the first and third columns will each take up one part, while the second column will take up two parts. This allows you to create responsive and flexible layouts where the column sizes adjust automatically based on the available space.

Continue Example: Here is how it looks now,

Still, it does not look like the expected output. The **header** section spans over two columns and the **nav** section spans over two rows.

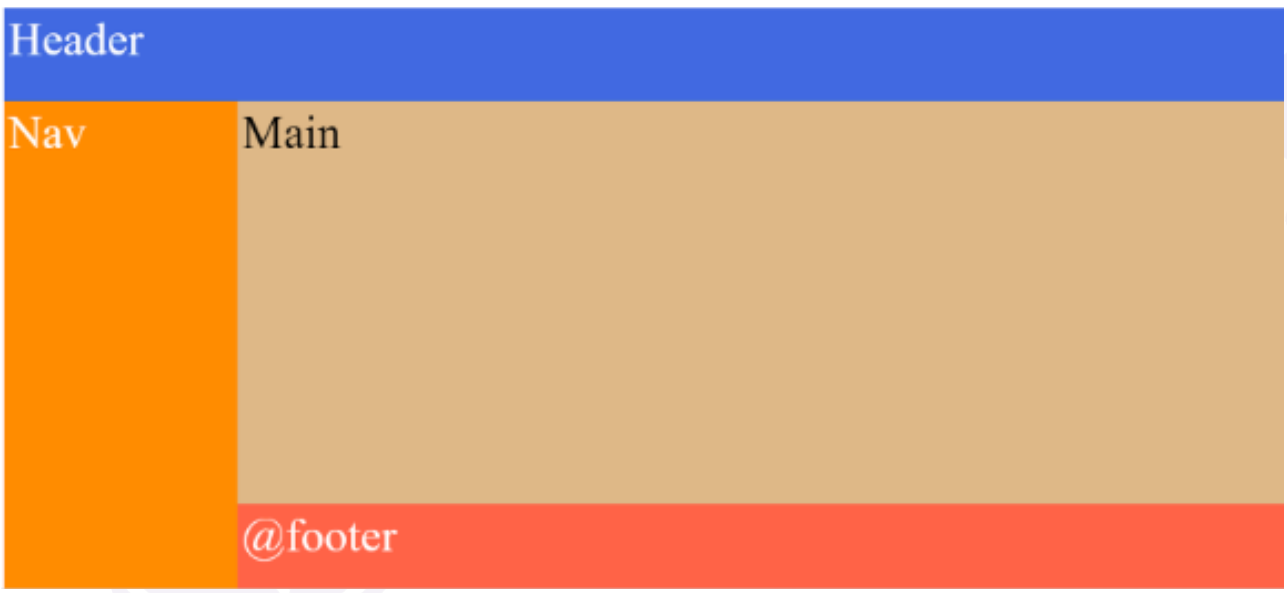
Still, it does not look like the expected output. The **header** section spans over two columns and the **nav** section spans over two rows.

```
Unset
header {
    grid-column: span 2;
}

nav{
   grid-row: span 2;
}
```

grid-column property is used to span a cell over two or more columns. Similarly, the **grid-row** property is used to span a cell over two or more rows. These are grid item properties.

Now it looks exactly the same as the expected output.



Advantages of Grid

Flexibility: CSS Grid allows for flexible and responsive layouts that can adapt to different screen sizes and device orientations.

Simplicity: Grid has a steeper learning curve than some other layout methods. It is easy to create complex layouts with just a few lines of code, **reducing the need for nested HTML elements** and CSS classes.

Precise positioning: CSS Grid offers precise control over the positioning of individual elements within the grid, allowing for pixel-perfect layouts.