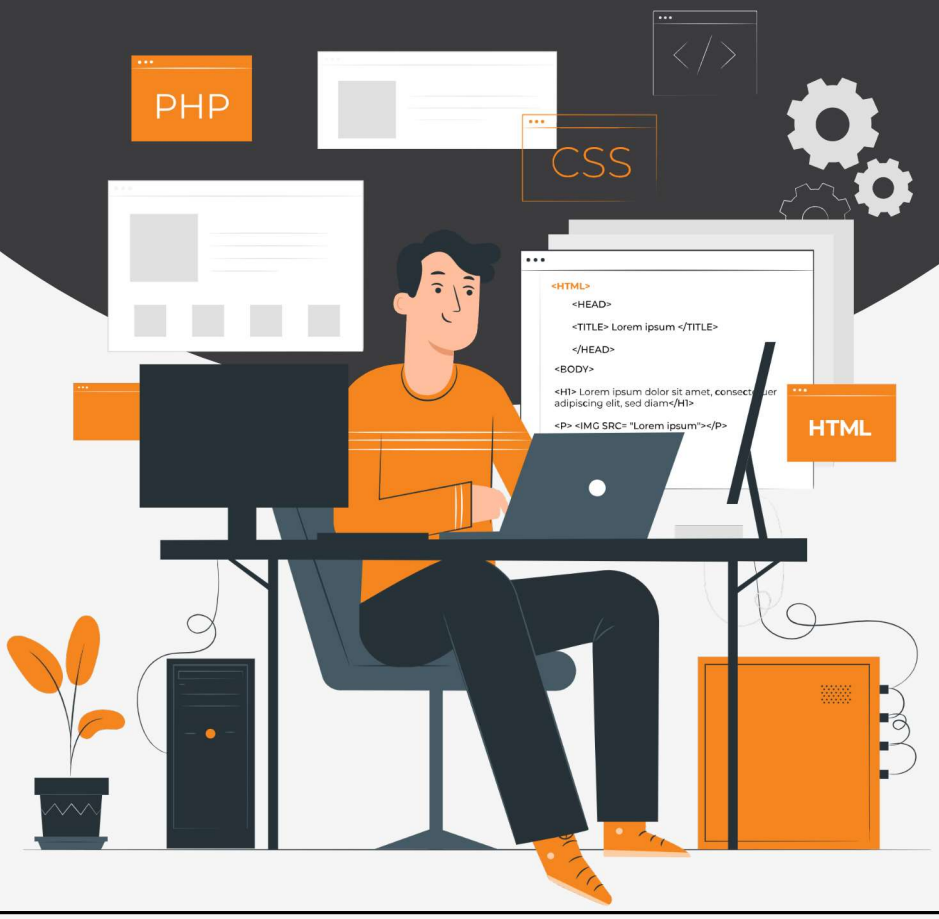
# Grid in CSS

# Assignment Solution




PW SKILLS

# TASK 1

**Problem Statement**

Create an image gallery using a CSS grid.

**Expected Behaviour**


**Solution**

**index.html**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="index.css" />
    <title>Image Gallary</title>
  </head>
  <body>
    <div class="photo-gallery">
      <div class="photo-1">
        <img
src="https://images.pexels.com/photos/276267/pexels-photo-276267.jpeg?
auto=compress&cs=tinysrgb&w=600"
          alt="image-01"
        />
      </div>
      <div class="photo-2">
        <img
          src="https://images.pexels.com/photos/34299/herbs-flavoring-seasoning-cooking.jpg?
auto=compress&cs=tinysrgb&w=600"
          alt="image-02"
        />
```

```
      </div>
      <div class="photo-3">
        <img
          src="https://images.pexels.com/photos/159045/the-interior-of-the-repair-interior-
design-159045.jpeg?auto=compress&cs=tinysrgb&w=600"
          alt="image-03"
        />
      </div>
      <div class="photo-4">
        <img
          src="https://images.pexels.com/photos/4033148/pexels-photo-4033148.jpeg?
auto=compress&cs=tinysrgb&w=600"
          alt="image-04"
        />
      </div>
      <div class="photo-5">
        <img
          src="https://images.pexels.com/photos/6074935/pexels-photo-6074935.jpeg?
auto=compress&cs=tinysrgb&w=600"
          alt="img-5"
        />
      </div>
      <div class="photo-6">
        <img
          src="https://images.pexels.com/photos/4210610/pexels-photo-4210610.jpeg?
auto=compress&cs=tinysrgb&w=600"
          alt="image-6"
        />
      </div>
    </div>
  </body>
</html>
```

**index.css**

```
    .photo-gallery {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      grid-template-rows: repeat(3, 250px);
      gap: 20px;
      padding: 20px;
    }

    .photo-1 {
      grid-column-start: 1;
      grid-column-end: 3;
      grid-row-start: 1;
      grid-row-end: 2;
    }
```

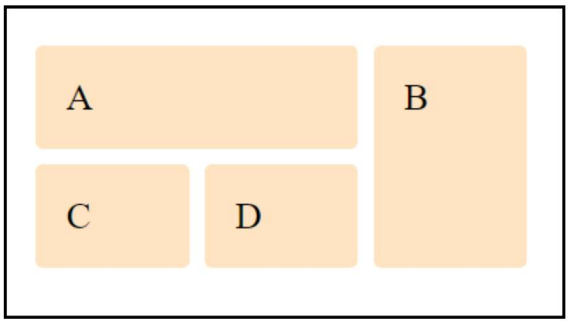
```
.photo-2 {
  grid-column-start: 2;
  grid-column-end: 3;
  grid-row-start: 2;
  grid-row-end: 3;
}

.photo-3 {
  grid-column-start: 3;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 2;
}

.photo-4 {
  grid-column-start: 1;
  grid-column-end: 2;
  grid-row-start: 2;
  grid-row-end: 4;
}

.photo-5 {
  grid-column-start: 2;
  grid-column-end: 3;
  grid-row-start: 3;
  grid-row-end: 4;
}

.photo-6 {
  grid-column-start: 3;
  grid-column-end: 4;
  grid-row-start: 2;
  grid-row-end: 4;
}

img {
  height: 100%;
  width: 100%;
  border-radius: 10px;
}
```

# TASK 2

**Problem Statement**

Write code to arrange containers with texts A, B, C, and D as shown in the below image.

**Expected Output**




**Solution**

**index.html**

```
<div class="container">
    <div class="box boxa">A</div>
    <div class="box boxb">B</div>
    <div class="box boxc">C</div>
    <div class="box boxd">D</div>
</div>
```

**index.css**

```
    .container {
      display: grid;
      grid-gap: 10px;
      grid-template-columns: 100px 100px 100px;
    }

    .box {
      background-color: bisque;
      color: #fff;
      border-radius: 5px;
      padding: 20px;
      font-size: 150%;
      color: black;
    }

    .boxa {
      grid-column: 1 / 3;
      grid-row: 1;
    }

    .boxb {
      grid-column: 3;
```

```
    grid-row: 1 / 3;
  }

  .boxc {
    grid-column: 1;
    grid-row: 2;
  }

  .boxd {
    grid-column: 2;
    grid-row: 2;
  }
```

## TASK 3

**Problem Statement**
Explain the use of grid-auto-row and grid-auto-column using code examples.

**Solution**

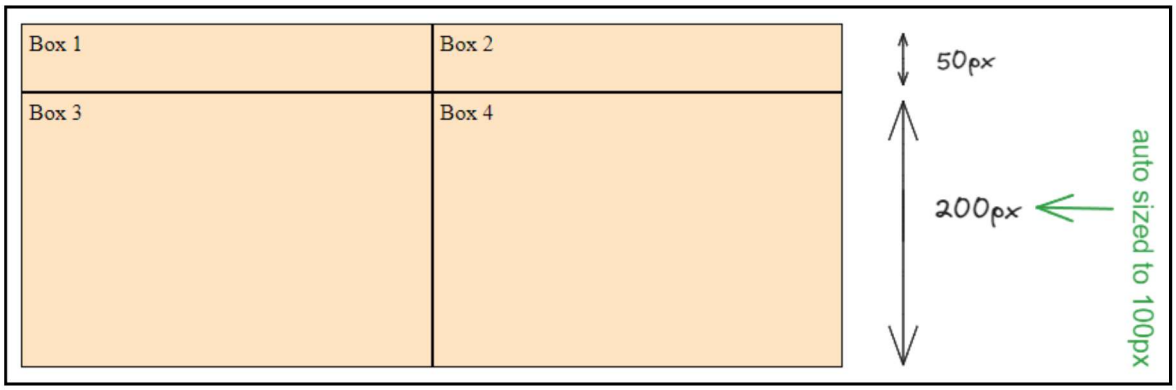
grid-auto-rows and grid-auto-column properties specify the height and width of rows that are automatically created when there is no explicit row definition and column definition respectively.

Here is an example,
**index.html**

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

**index.css**

```
.container {
    display: grid;
    grid-template-areas: "X X";
    grid-template-rows: 50px;
    grid-auto-rows: 200px;
  }

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



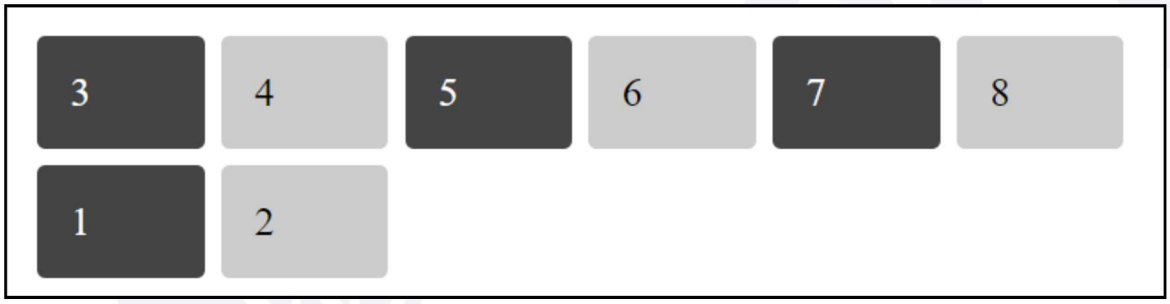

This example creates a grid container with three columns defined by grid-template-areas. The first row is displayed with a height of 50px because we have mentioned grid-template-rows for the first row as 50px. And the remaining rows will be displayed with a height of 100px, because of the grid-auto-rows:200px property. That's why the second row is displayed with a height of 200px.

## TASK 4

**Problem Statement**

Write CSS to show numbers as shown in the figure, without altering the html file.

**Expected Output**




**Solution**

**index.css**

```
body {
    margin: 40px;
  }

  .box {
    background-color: #444;
    color: #fff;
    border-radius: 5px;
    padding: 20px;
    font-size: 150%;
    order: 1;
  }

  .box:nth-child(even) {
    background-color: #ccc;
```

```
    color: #000;
}

.container {
  width: 600px;
  display: grid;
  grid-template-columns: repeat(6, 100px);
  grid-gap: 10px;
}

.box1 {
  order: 3;
}

.box2 {
  order: 6;
}

.box8 {
  order: 2;
}
```

## TASK 5

**Problem Statement**
Explain the difference between justify-items and justify-self using code examples.

**Solution**
The main difference between them is that justify-items apply to grid containers and justify-self applies to grid-items.

Let us understand them by taking an example, below code, for example having 6 grid items arranged in a 3x2 grid.

**index.html**

```
<div class="container">
      <div class="box1">Box 1</div>
      <div class="box2">Box 2</div>
      <div class="box3">Box 3</div>
      <div class="box4">Box 4</div>
      <div class="box5">Box 5</div>
      <div class="box6">Box 6</div>
</div>
```

**index.css**

```
.container {
      display: grid;
      grid-template-columns: auto auto auto;
```

```
      border: 1px solid black;
}

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

**Output**

| | | |
|---|---|---|
| Box 1 | Box 2 | Box 3 |
| Box 4 | Box 5 | Box 6 |

Lets apply justify-items: center on grid container,

**index.css**

```
.container {
  ...
  justify-items: center;
}
```

**Output**

| | | |
|---|---|---|
| Box 1 | Box 2 | Box 3 |
| Box 4 | Box 5 | Box 6 |

You see above grid items arranged center along the inline axis, within their respective grid cells.

This property allows us to set the default way of justifying each item in their respective grid cells.

justify-self and align-self properties control the alignment of any specific grid item within its grid cell along the horizontal and vertical axes, respectively. The values can be a start, end, center, or stretch.