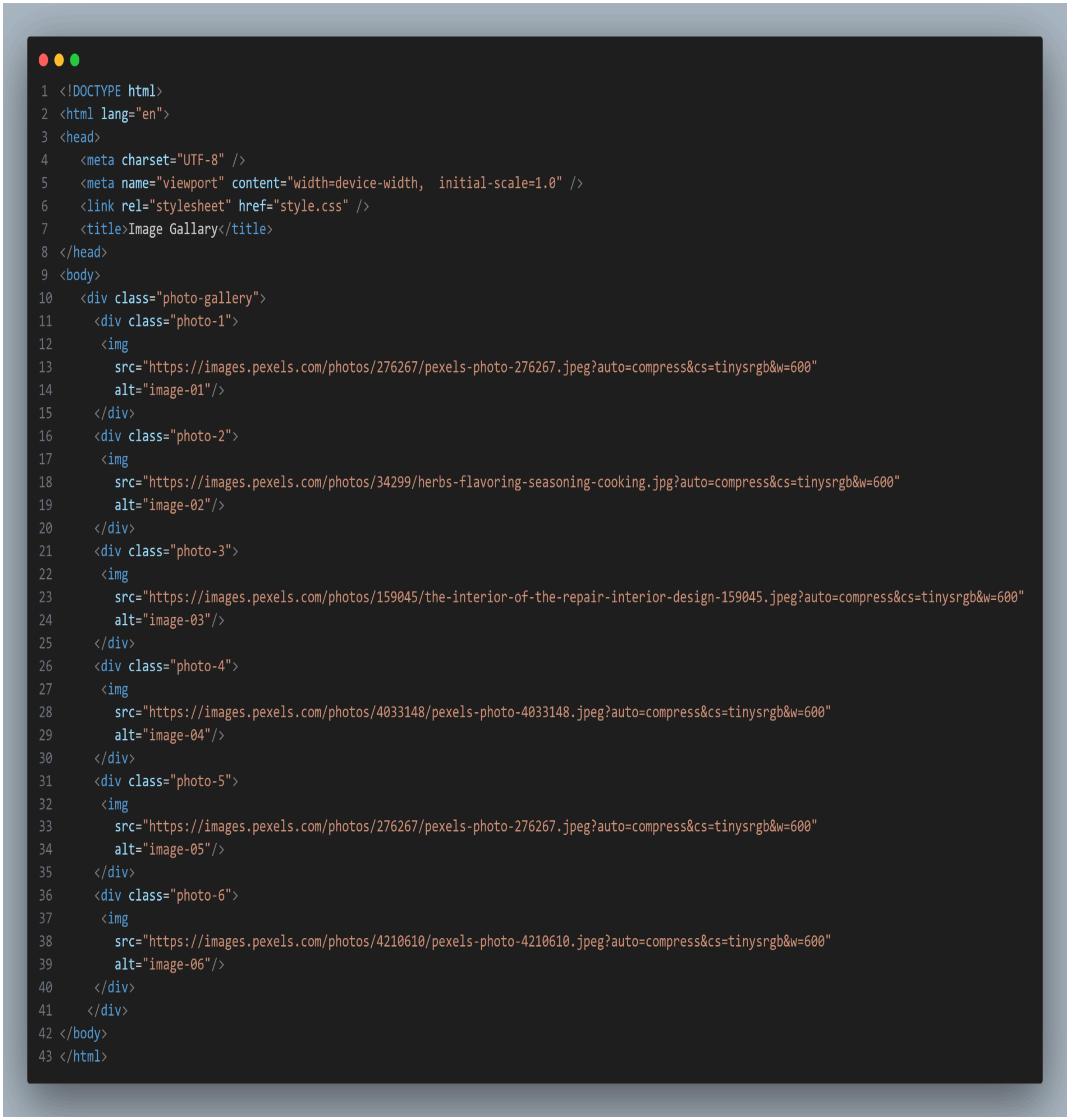
Grid in CSS Assignment

Assignment Question:

1. Create an image gallery using a CSS grid.

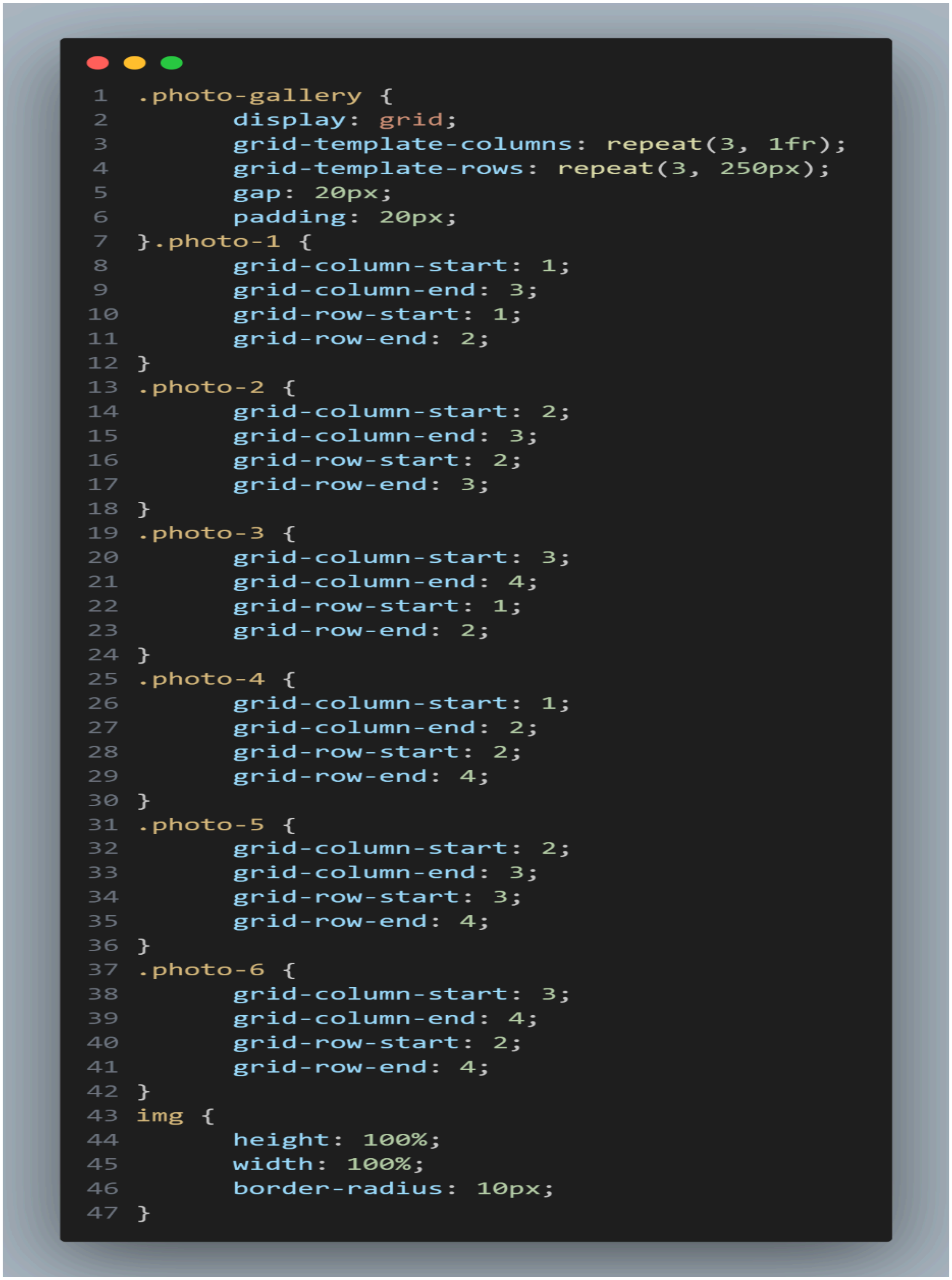
Answer:

Index.html



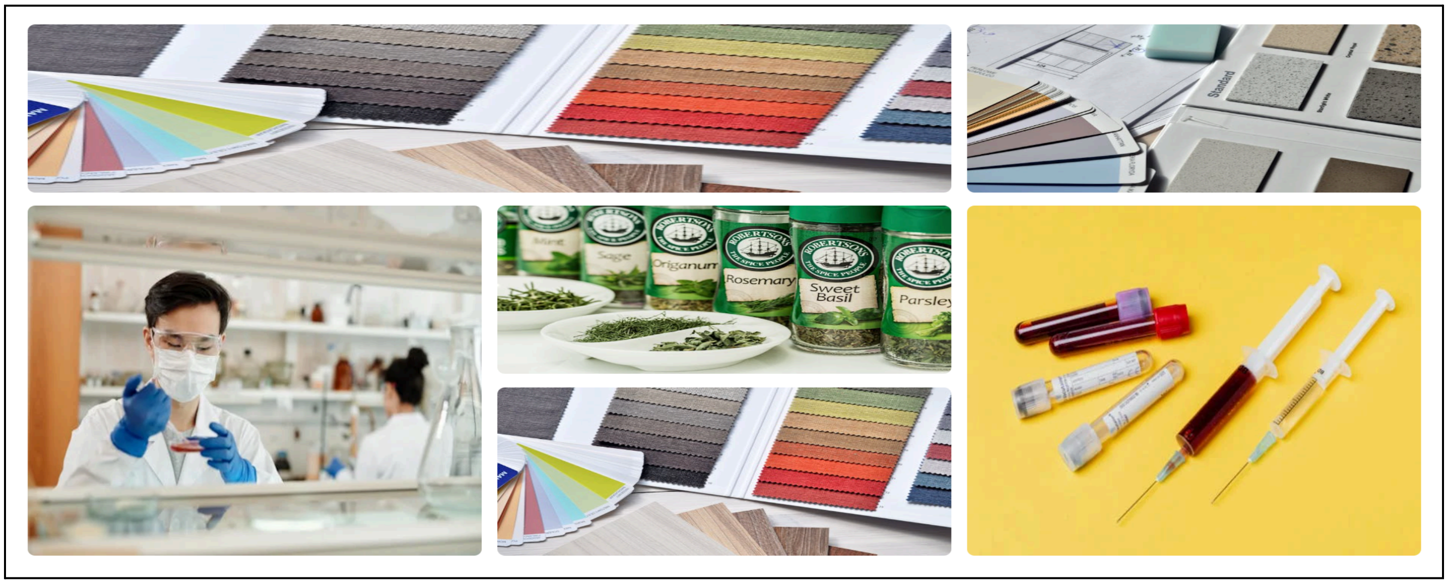
```html
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8" />
   <meta name="viewport" content="width=device-width,  initial-scale=1.0" />
   <link rel="stylesheet" href="style.css" />
   <title>Image Gallary</title>
</head>
<body>
   <div class="photo-gallery">
     <div class="photo-1">
      <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/276267/pexels-photo-276267.jpeg?auto=compress&cs=tinysrgb&w=600"
        alt="image-05"/>
     </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-06"/>
     </div>
    </div>
</body>
</html>
```

Style.css



```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;
}
```

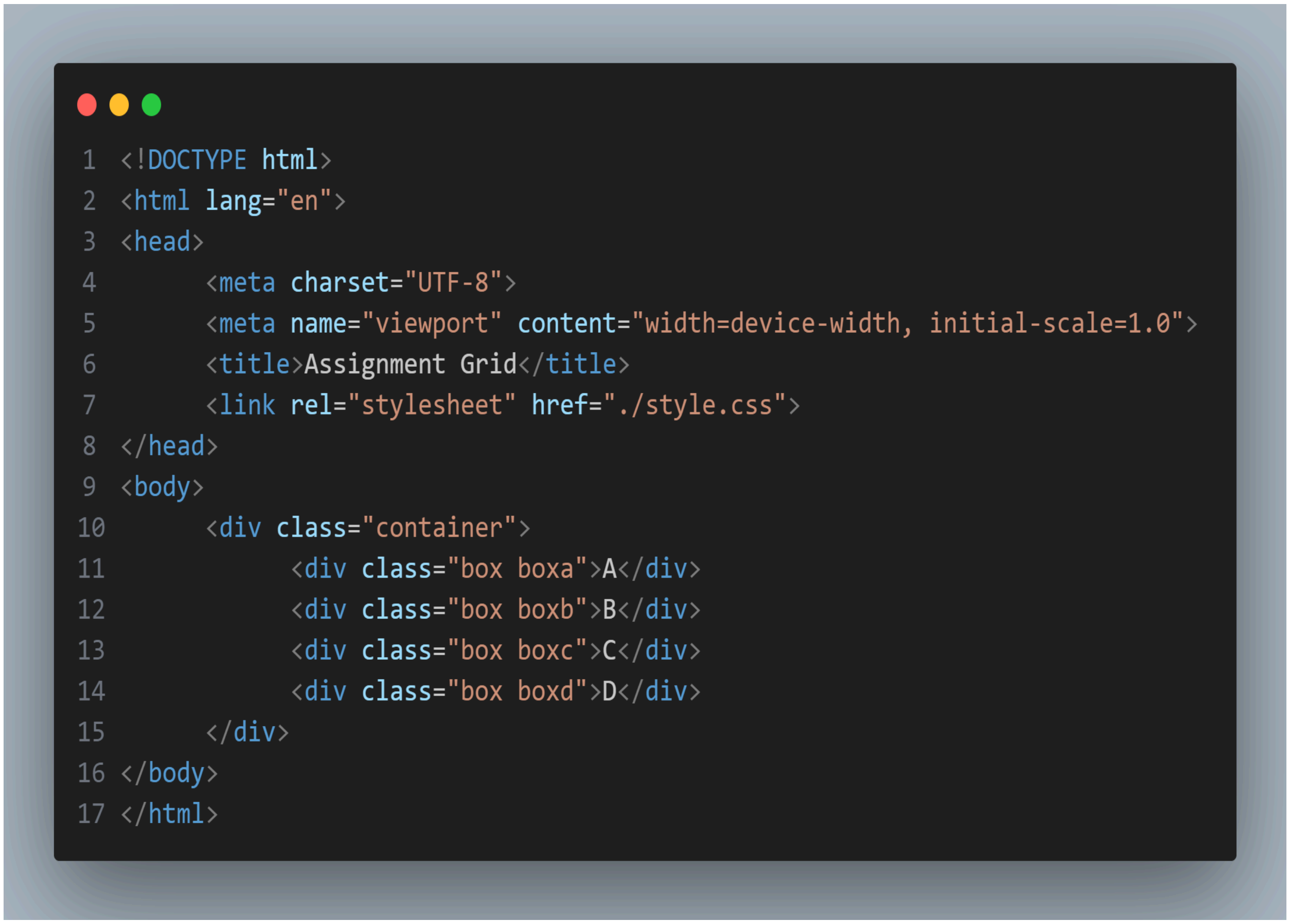
Browser Output:



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

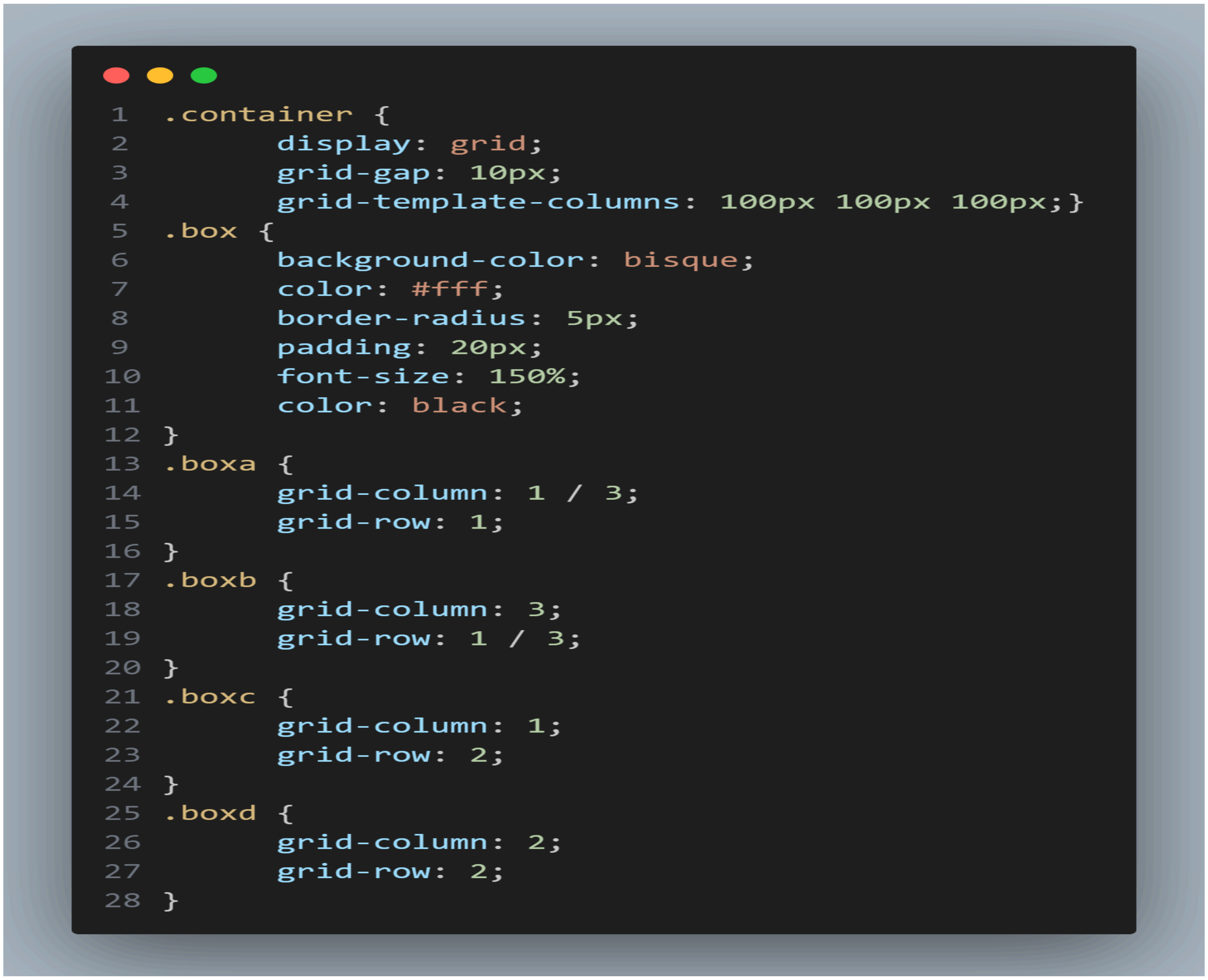
Answer:

Index.html



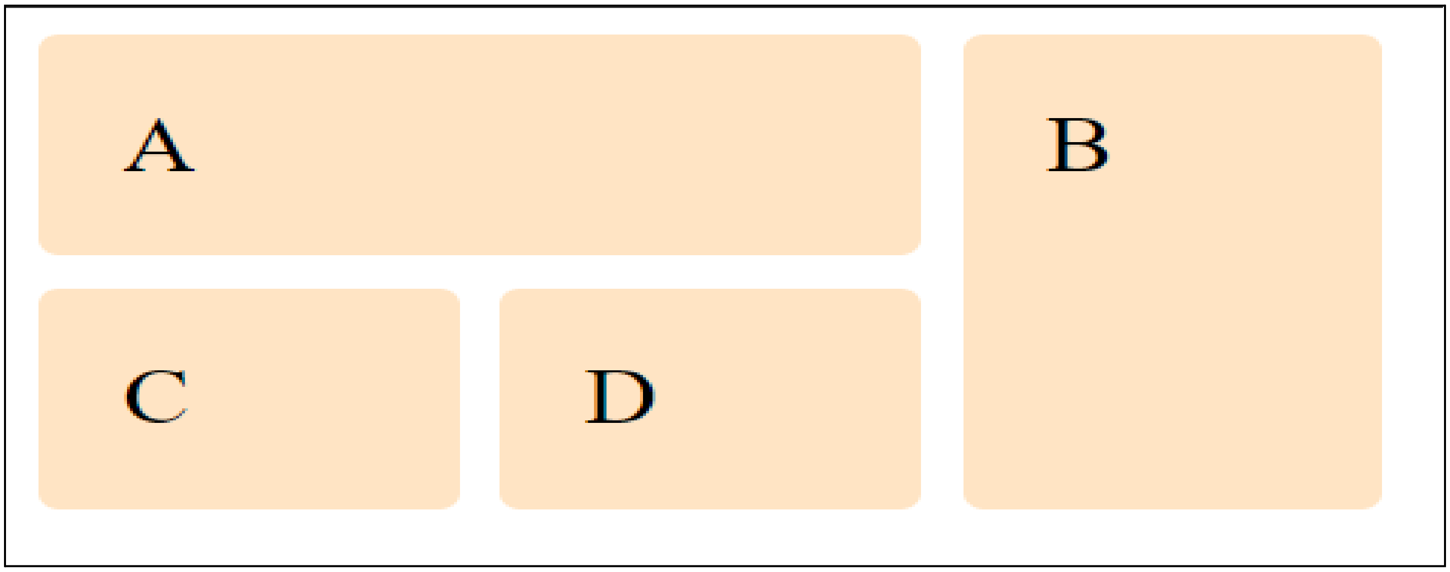
```html
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Assignment Grid</title>
      <link rel="stylesheet" href="./style.css">
</head>
<body>
      <div class="container">
            <div class="box boxa">A</div>
            <div class="box boxb">B</div>
            <div class="box boxc">C</div>
            <div class="box boxd">D</div>
      </div>
</body>
</html>
```

Style.css



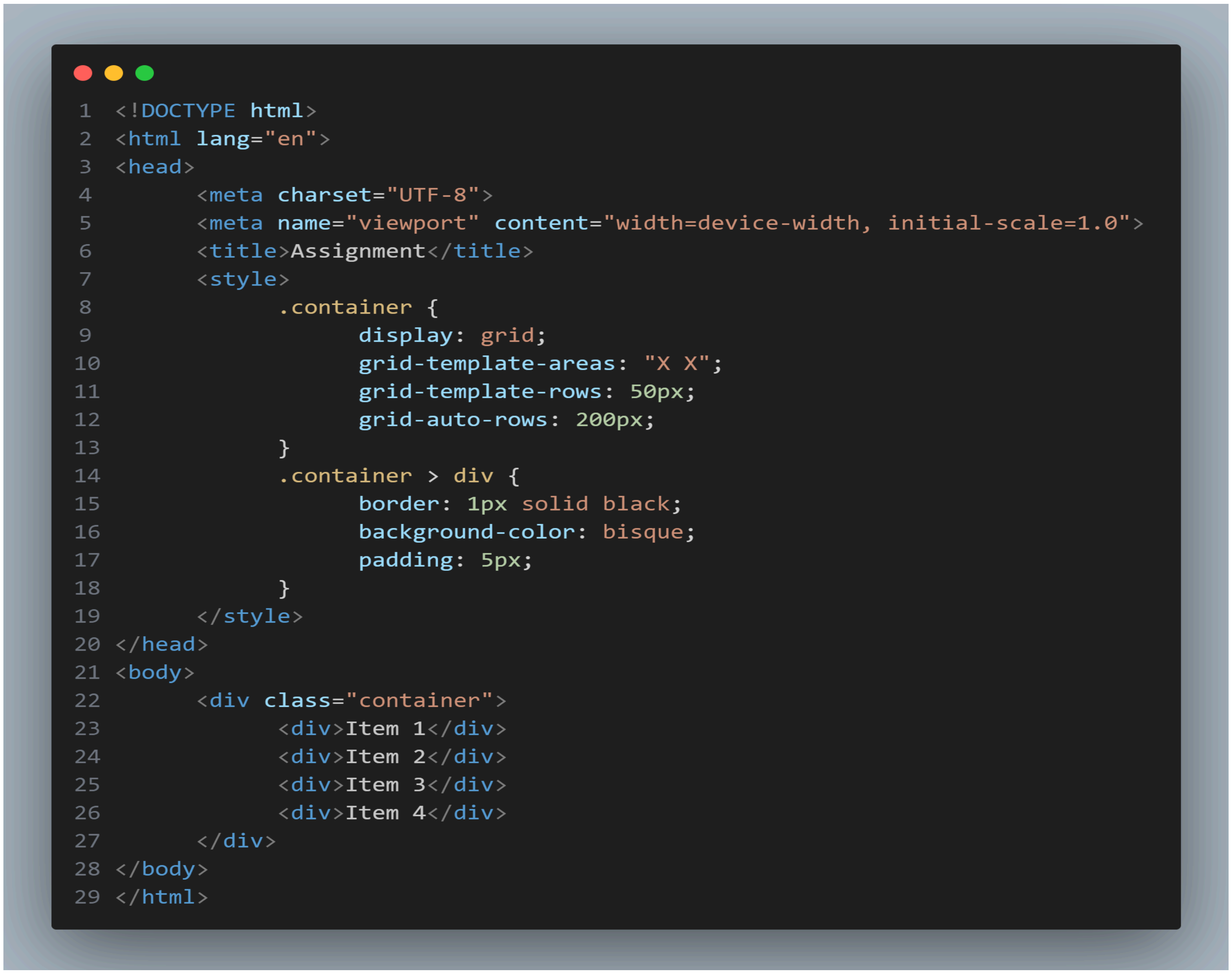
```css
.container {
        display: grid;
        grid-gap: 10px;
        grid-template-columns: 100px 100px 100px;}
.box {
        background-color: bisque;
        color: #fff;
        border-radius: 5px;
        padding: 20px;
        font-size: 150%;
        color: black;
}
.boxa {
        grid-column: 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;
}
```

Browser Output:



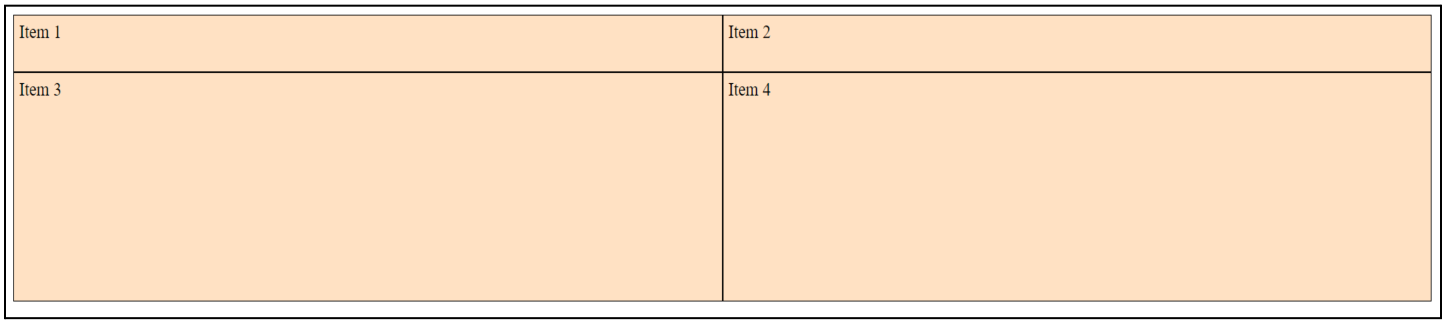
3. Explain the use of grid-auto-row and grid-auto-column using code examples.

Answer: 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,



```html
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Assignment</title>
      <style>
            .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;
            }
      </style>
</head>
<body>
      <div class="container">
            <div>Item 1</div>
            <div>Item 2</div>
            <div>Item 3</div>
            <div>Item 4</div>
      </div>
</body>
</html>
```

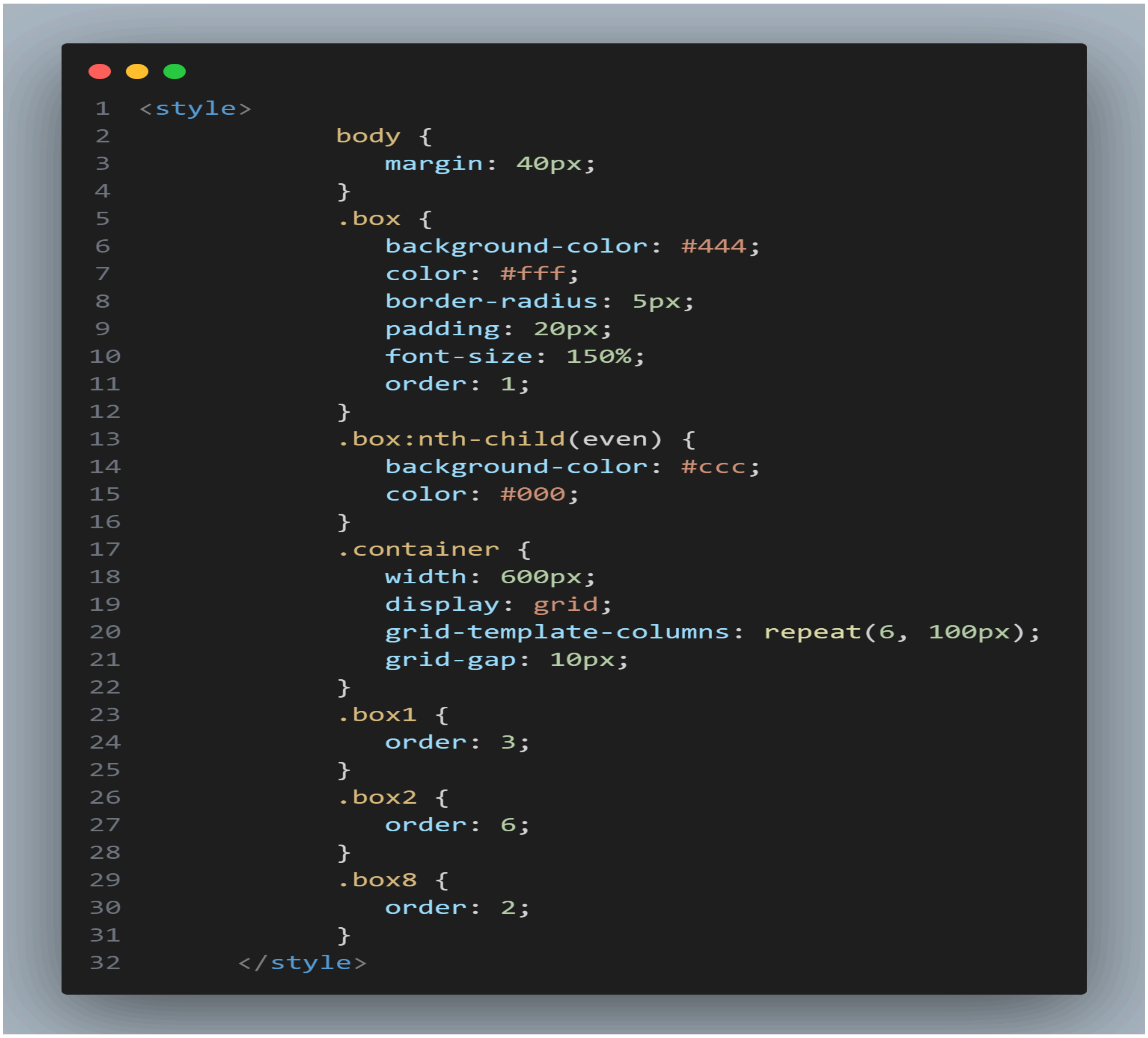
Browser Output:



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.

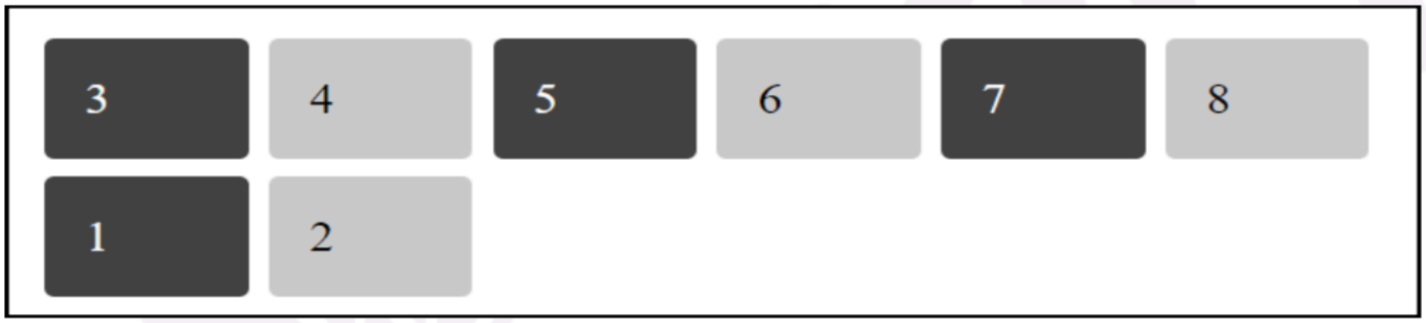
4. Write CSS to show numbers as shown in the figure, without altering the html file.

Answer:



```
<style>
            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;
            }
      </style>
```

Browser Output:



```
3    4    5    6    7    8
1    2
```

5. Explain the difference between justify-items and justify-self using code examples.

Answer: Certainly! The `justify-items` and `justify-self` properties in CSS are used to align items in a grid or flex container along the inline (row) axis. Here's a breakdown of their differences along with examples to illustrate how each property works.

`justify-items`

The `justify-items` property aligns items along the row axis within their containing block for all items in a grid container. It sets the default alignment for all the grid items inside the container.

Syntax`css :
justify-items: start | end | center | stretch;
- `start`: Items are aligned to the start of the container.
- `end`: Items are aligned to the end of the container.
- `center`: Items are centered within the container.
- `stretch`: Items are stretched to fill the container (default).

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>justify-items Example</title>
   <style>
      .grid-container {
         display: grid;
         grid-template-columns: repeat(3, 1fr);
         justify-items: center;
         gap: 10px;
      }
      .grid-item {
         background-color: lightblue;
         padding: 20px;
      }
   </style>
</head>
<body>
   <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>
</body>
</html>
```

In this example, all grid items are centered along the row axis within their grid cells.

`justify-self`

The `justify-self` property aligns a single item within its containing block along the row axis. It can override the `justify-items` property for individual grid items.

Syntax css
justify-self: start | end | center | stretch;
- `start`: Item is aligned to the start of its container.
- `end`: Item is aligned to the end of its container.
- `center`: Item is centered within its container.
- `stretch`: Item is stretched to fill its container (default).

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>justify-self Example</title>
    <style>
        .grid-container {
            display: grid;
            grid-template-columns: repeat(3, 1fr);
            justify-items: center; /* Default alignment for all items */
            gap: 10px;
        }
        .grid-item {
            background-color: lightblue;
            padding: 20px;
        }
        .grid-item:nth-child(2) {
            justify-self: end; /* Specific alignment for the second item */
        }
    </style>
</head>
<body>
    <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>
</body>
</html>
```
In this example, while all grid items are centered along the row axis by default, the second item is aligned to the end of its grid cell due to the `justify-self` property.

Summary
`justify-items`: Sets the alignment for all items within the container.
`justify-self`: Sets the alignment for an individual item, overriding `justify-items` for that item.

These properties are particularly useful for fine-tuning the layout of grid items in CSS Grid Layout.