Grid in CSS - Answers

1. Create an image gallery using a CSS grid.

   HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Grid</title>
    <link href="grid.css" rel="stylesheet">
</head>
<body>
    <div class="grid-container">
        <div class="grid-item item-1">
        <img src="image2.jpg" alt="img2">
        </div>
        <div class="grid-item item-2">
        <img src="image3.jpg" alt="img3">
        </div>
        <div class="grid-item item-3">
        <img src="image4.jpg" alt="img4">
        </div>
        <div class="grid-item item-4">
        <img src="image5.jpg" alt="img5">
        </div>
        <div class="grid-item item-5">
        <img src="image6.jpg" alt="img6">
        </div>
        <div class="grid-item item-6">
        <img src="image7.jpg" alt="img7">
        </div>
    </div>
</body>
</html>
```

   CSS:

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

```
    grid-template-columns:repeat(3, 1fr);
    grid-auto-rows: minmax(180px, auto);
    gap:10px;
    max-width:900px;
    width:100%;
    padding:10px;
    box-sizing:border-box;
    background-color:#ffff1f;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
.grid-item img{
    width: 100%;
    height: 100%;
    object-fit:cover;
    display:block;
    border-radius: 5px;
}
.item-1{
    grid-column:1/3;
}
.item-4{
    grid-row:2/4;
}
.item-5{
    grid-row:3/4;
}
.item-6{
    grid-row:span 2;
}
```

OUTPUT: Preview

---

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

   HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Grid</title>
    <link rel="stylesheet" href="design.css">
</head>
<body>
    <div class="container">
        <div class="element item1">
            A
        </div>
        <div class="element item2">
            B
        </div>
        <div class="element item3">
            C
        </div>
        <div class="element item4">
            D
        </div>
    </div>
</body>
</html>
```

CSS:

```
.container{
    display: grid;
    border:2px solid black;
    padding:10px;
    max-width:60%;
    width:200px;
    grid-template-columns: repeat(3, 1fr);
    grid-template-rows: repeat(2, 1fr);
    gap:10px;
}
.element{
    background-color: aqua;
    padding: 10px;
    border-radius: 10px;
    display:block;
    object-fit: cover;
}
.item1{
```

```
    grid-column:1/3;
    grid-row: 1/2;
}
.item2{
    grid-row:1/4;
}
```

OUTPUT: Preview

---

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

grid-auto-rows:

This property specifies the size of any automatically created row tracks.

SYNTAX:

Grid-auto-rows: <track-size> | <track-size> ….

<track-size> can be any valid track size value, such as:

- px (pixels)
- % (percentage)
- fr (fractional unit)
- auto (based on content size)
- min-content
- max-content
- minmax(min, max)

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>grid-auto-rows</title>
    <style>
        .container {
            display: grid;
            grid-template-columns: repeat(3, 1fr);
```

```
            grid-template-rows: 100px 100px;
            grid-auto-rows: 50px;
            border: 2px solid #333;
            width: 80%;
            margin: 20px auto;
        }

        .item {
            background-color: lightblue;
            border: 1px solid steelblue;
            display: flex;
            justify-content: center;
            align-items: center;
            font-size: 1.2em;
        }

        .item-4 {
            grid-row: 4;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="item item-1">Item 1 (100px)</div>
        <div class="item item-2">Item 2 (100px)</div>
        <div class="item item-3">Item 3 (100px)</div>
        <div class="item item-4">Item 4 (50px - Implicit)</div>
        <div class="item item-5">Item 5 (100px)</div>
        <div class="item item-6">Item 6 (100px)</div>
    </div>
</body>
</html>
```

grid-auto-columns:

This property specifies the size of any automatically created column track.

SYNTAX:

grid-auto-columns: <track-size> | <track-size> ....

<track-size> values are the same as for grid-auto-rows.

CODE:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="variable" content="width=device-width,
initial-scale=1.0">
        <title>Grid-auto-column</title>
        <style>
            .container{
                display:grid;
                grid-template-columns:100px 100px;
                grid-template-rows:repeat(2, 100px);
                grid-auto-columns: 70px;
                border: 2px solid black;
                width:70%;
                margin:20px auto;
                grid-auto-flow: row;
            }
            .item{
                background-color: lightcoral;
                border:1px solid firebrick;
                display:flex;
                justify-content:center;
                align-content:center;
                font-size:1.2em;
            }
            .item3{
                grid-column:4;
                grid-row:1;
            }
        </style>
    </head>
    <div class="container">
        <div class="item item1">Item 1 (100px)</div>
        <div class="item item2">Item 2 (100px)</div>
        <div class="item item3">Item 3 (70px - Implicit)</div>
        <div class="item item4">Item 4 (100px)</div>
        <div class="item item5">Item 5 (100px)</div>
    </div>
</html>
```

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

    <div class=”box box1”> 1
    <div class=”box box2”> 2
    <div class=”box box3”> 3
    <div class=”box box4”> 4
    <div class=”box box5”> 5
    <div class=”box box6”> 6
    <div class=”box box7”> 7
    <div class=”box box8”> 8

CODE:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Box</title>
        <style>
            .container{
                display:grid;
                grid-template-columns:repeat(6,1fr);
                grid-auto-rows: 40px 40px;
                width:80%;
                border:2px solid black;
                padding:10px;
                margin:20px;
                gap:10px;
            }
            .box{
                display: flex;
                justify-content: center;
                align-content: center;
                background-color: navajowhite;
                border:1px solid red;
```

```
            }
            .box1{
                grid-row: 2;
            }
            .box2{
                grid-row: 2;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="box box1"> 1 </div>
            <div class="box box2"> 2 </div>
            <div class="box box3"> 3 </div>
            <div class="box box4"> 4 </div>
            <div class="box box5"> 5 </div>
            <div class="box box6"> 6 </div>
            <div class="box box7"> 7 </div>
            <div class="box box8"> 8 </div>
        </div>
    </body>
</html>
```

OUTPUT: Preview

---

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

   JUSTIFY-ITEMS:

   Justify-items is applied to the grid container. It dictates how items are justified horizontally within their respective grid-cells.

   SYNTAX:

   .grid-container {
   justify-items: start | end | center | stretch;
   }

   CODE:

```
<!DOCTYPE html>
<html lang="en">
```

```
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Justify-items</title>
        <style>
            .container{
                display: grid;
                grid-template-columns: repeat(3,150px);
                grid-template-rows: 100px;
                border:2px solid #4CAF50;
                padding:50px;
                gap:10px;
                justify-items: center;
                background-color: antiquewhite;
            }
            .item{
                background-color: crimson;
                border: 1px solid black;
                padding:10px;
                font-size: 1.2em;
                text-align: center;
                width:80px;
                color:aliceblue;
            }
            .container-start{
                justify-items: start;
            }
            .container-end{
                justify-items: end;
            }
            .container-center{
                justify-items:center;
            }
            .container-stretch{
                justify-items:stretch;
            }
        </style>
    </head>
    <body>
        <h2>Justify-items: Start</h2>
        <div class="container container-start">
            <div class="item">Item1</div>
```

```
            <div class="item">Item2</div>
            <div class="item">Item3</div>
        </div>
        <h2>Justify-items: End</h2>
        <div class="container container-end">
            <div class="item">Item1</div>
            <div class="item">Item2</div>
            <div class="item">Item3</div>
        </div>
        <h2>Justify-items: Center</h2>
        <div class="container container-center">
            <div class="item">Item1</div>
            <div class="item">Item2</div>
            <div class="item">Item3</div>
        </div>
        <h2>Justify-items: Stretch</h2>
        <div class="container container-stretch">
            <div class="item">Item1</div>
            <div class="item">Item2</div>
            <div class="item">Item3</div>
        </div>
    </body>
</html>
```

JUSTIFY-SELF:

justify-self is applied to an individual grid-item. It overrides any justify-items setting for that particular item.

SYNTAX:

.grid-item{

Justify-self: start | end | center | stretch;

}

CODE:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Justify-Self</title>
        <style>
```

```
            .container{
                display:grid;
                border:2px solid #FFC107;
                grid-template-columns: repeat(3, 150px);
                grid-template-rows: 100px;
                padding:10px;
                justify-self: start;
                gap:10px;
            }
            .item{
                background-color: #fb0101;
                border:2px solid blue;
                text-align: center;
                font-size: 1.2em;
                color: white;
                width: 80px;
                padding: 10px;
            }
            .item1{
                justify-self: start;/*Default value*/
            }
            .item2{
                justify-self: center;
            }
            .item3{
                justify-self: end;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="item item1">Self: Start (default)</div>
            <div class="item item2">Self: Center</div>
            <div class="item item3">Self: End</div>
        </div>
    </body>
</html>
```

ANALOGY:

Imagine a classroom:

- Justify-item is like the teacher telling everyone to sit at the front of their desk (start), middle of their desk (center), or back of their desk (end).
- Justify-self is a student deciding, "Nah, I'm going to sit in the *middle* of my desk," even if the teacher said to sit at the front. This student's individual decision overrides the general instruction for just themselves.
