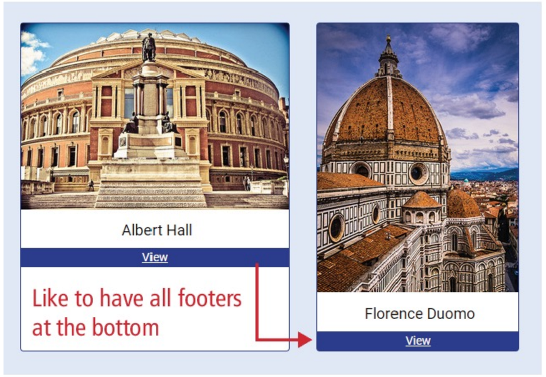
Fundamentals of Web Development

Third Edition by Randy Connolly and Ricardo Hoar

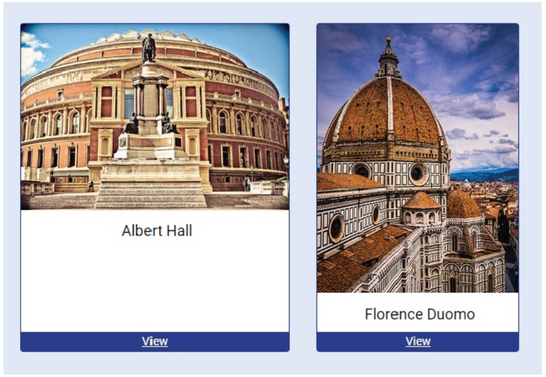
Chapter 7

CSS 2: Layout

Flexbox Cards



```
<div class="card">
   <div class="content">
      <img src="w.jpg" />
      <h3>Albert Hall</h3>
   </div>
   <footer>
      <a href="#">View</a>
   </footer>
</div>
```

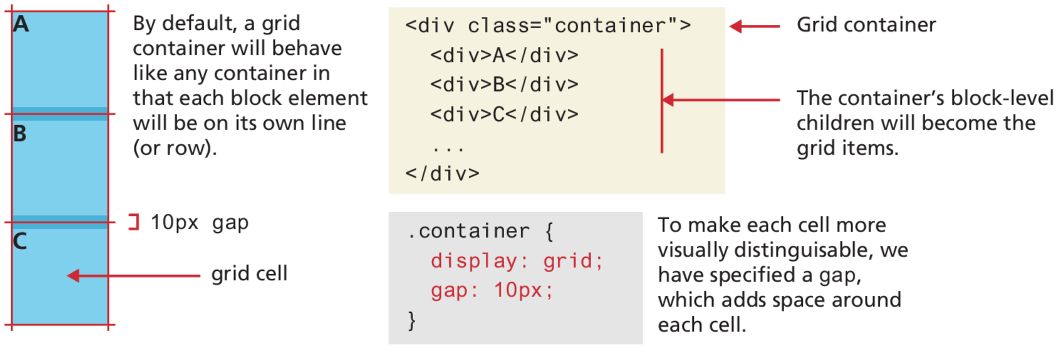


```
.card {
  ...
  display: flex;
  flex-direction: column;
}
.card .content {
   flex: 1 1 auto;
}
```

Grid Layout

Grid layout is adjustable, powerful, and, compared to floats, positioning, and even flexbox, is relatively easy to learn and use!

- Each blocklevel child in a parent container whose **display** property is set to **grid** will be automatically placed into a grid cell



By default, a grid container will behave like any container in that each block element will be on its own line (or row).

```
<div class="container">
  <div>A</div>
  <div>B</div>
  <div>C</div>
  ...
</div>
```

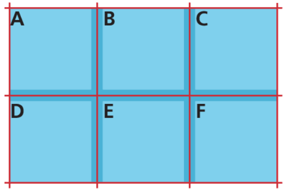
The container's block-level children will become the grid items.

```
.container {
  display: grid;
  gap: 10px;
}
```

To make each cell more visually distinguisable, we have specified a gap, which adds space around each cell.

Specifying Grid Structure

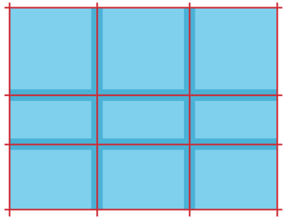
grid-template-columns is used for adding columns by specifying each column's width using the **fr** unit.



```
.container {
  display: grid;
  gap: 10px;
  grid-template-columns: 1fr 1fr 1fr;
}
```

This makes each column equal to an equal fraction of the available space.

You can specify the number of columns per row/line via the `grid-template-column` property.



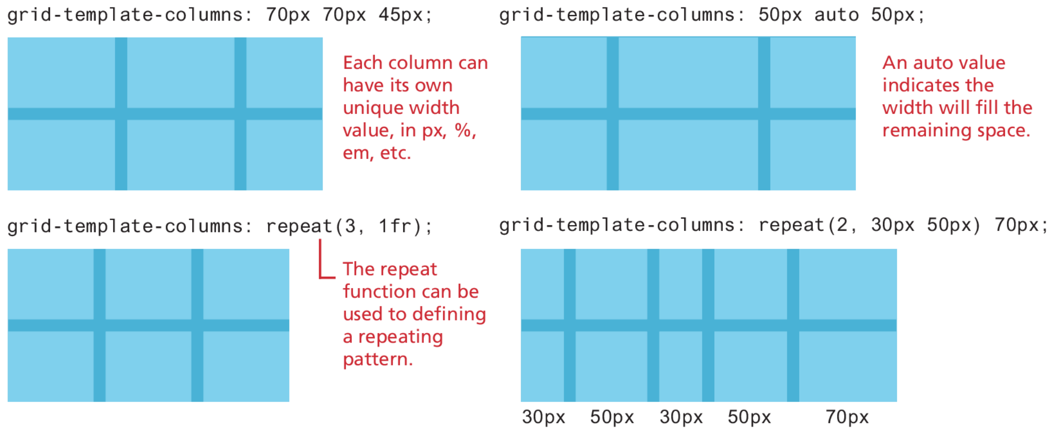
```
.container {
  ...
  height: 300px;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 80px 30px 50px;
}
```

While you often do not need to set a row height, you can make rows taller than their actual content.

Specifying column widths

Column widths can be specified

The CSS **repeat()** function provides a way to specify repeating patterns of columns.



```
grid-template-columns: 70px 70px 45px;
```

Each column can have its own unique width value, in px, %, em, etc.

```
grid-template-columns: 50px auto 50px;
```

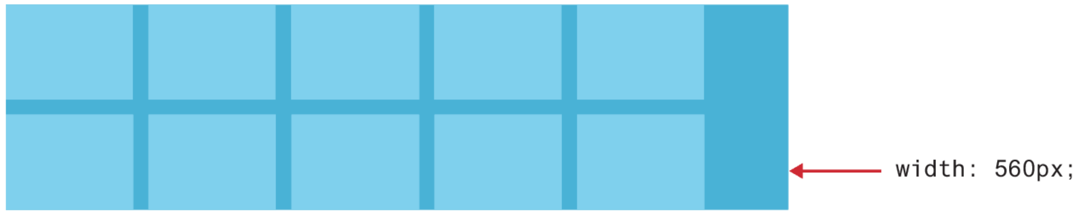
An auto value indicates the width will fill the remaining space.

```
grid-template-columns: repeat(3, 1fr);
```

The repeat function can be used to defining a repeating pattern.

```
grid-template-columns: repeat(2, 30px 50px) 70px;
```

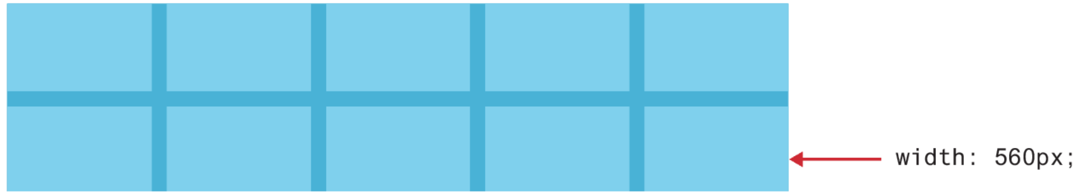
Specifying column widths (ii)



```
grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
```

Fills the row with as many columns as can fit into the container space.

The min column size is 100px and the max is whatever size is necessary to allow each column to be equal sized.



```
grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
```

Fits the columns into space by expanding the column width into the available container space.

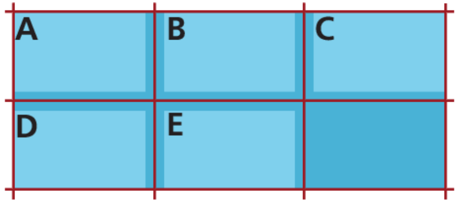
Grid

```
<!-- CSS Grid Approach -->
<div class="container">
  <img src=1.gif />
  <img src=2.gif />
  <img src=3.gif />
  <img src=4.gif />
  <img src=5.gif />
  <img src=6.gif />
</div>

<!-- CSS for grid approach -->
.container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(100px, 1fr);
}
.container img { display: block; }
```

Explicit Grid Placement Example 1

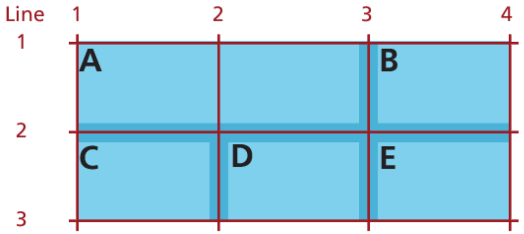
```
<div class="container">
  <div class="a">A</div>
  <div class="b">B</div>
  <div class="c">C</div>
  <div class="d">D</div>
  <div class="e">E</div>
</div>
```

```
.container {
  display: grid;
  gap: 10px;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(2,200px);
}
```



With implicit layout, grid items are placed automatically.

Example **1**



```
.a {
  grid-column-start: 1;
  grid-column-end: 3;
}
```

The start and end numbers refer to the line number not the column number.

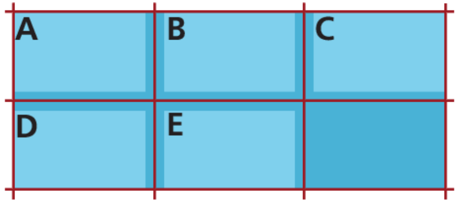
The same effect also possible using either of the following:

```
grid-column: 1 / 3;
grid-column: 1 / span 2;
```

Explicit Grid Placement Example 2

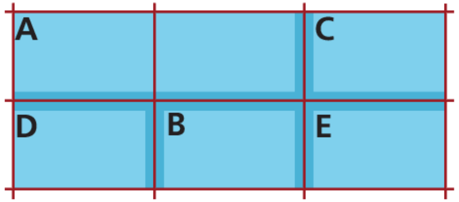
```
<div class="container">
  <div class="a">A</div>
  <div class="b">B</div>
  <div class="c">C</div>
  <div class="d">D</div>
  <div class="e">E</div>
</div>
```

```
.container {
  display: grid;
  gap: 10px;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(2,200px);
}
```



With implicit layout, grid items are placed automatically.

2

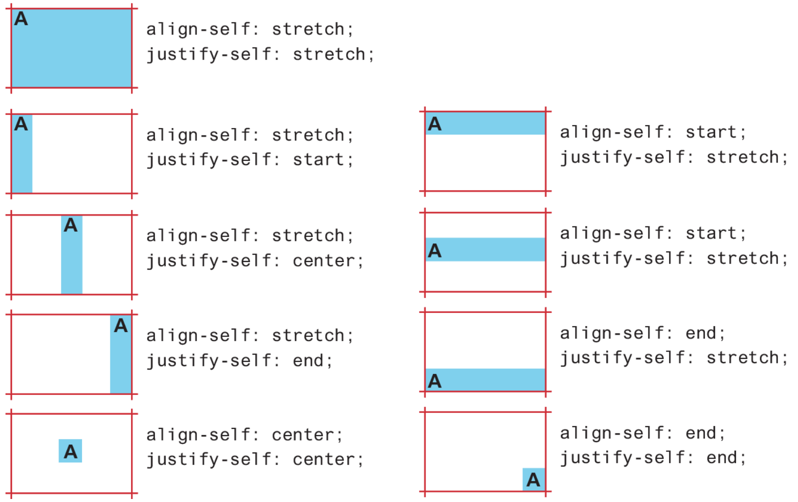


```
.b {
  grid-row: 2;
  grid-column: 2;
}
```

Grid cells can be placed into any row and column.

Cell properties

- **align-self and justify-self** control the cell content’s horizontal and vertical alignment within its grid container.

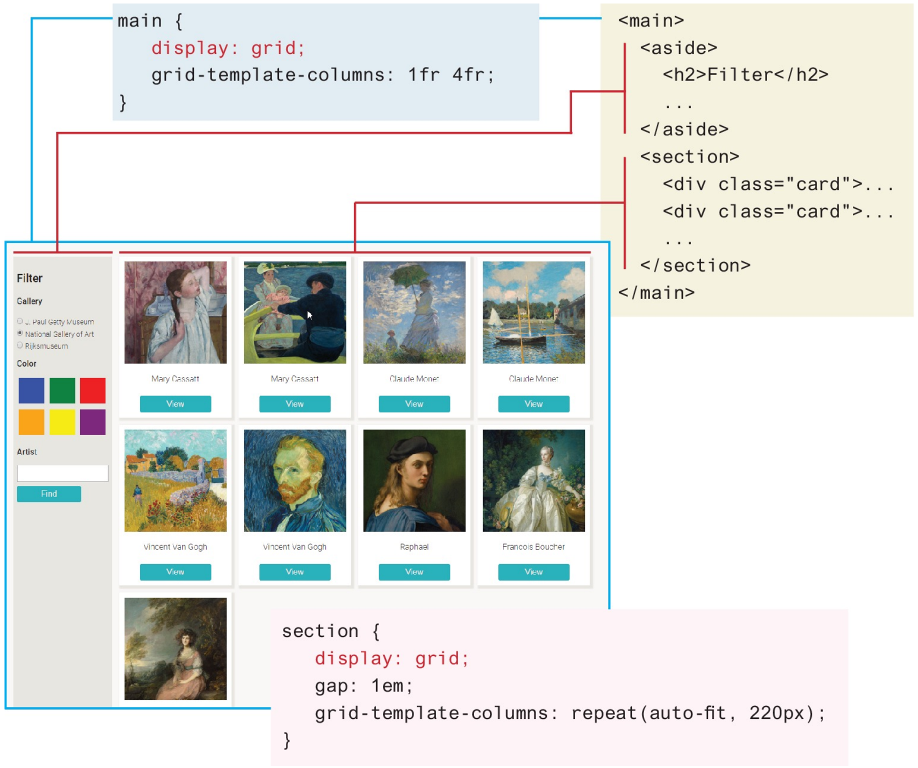


```
align-self: stretch;
justify-self: stretch;

align-self: stretch;
justify-self: start;

align-self: stretch;
justify-self: center;

align-self: stretch;
justify-self: end;

align-self: center;
justify-self: center;

align-self: start;
justify-self: stretch;

align-self: start;
justify-self: stretch;

align-self: end;
justify-self: stretch;

align-self: end;
justify-self: end;
```

- You can similarily control cell alignment within a grid container using **align-items** and **justify-items**

Nested Grid

- **align-self and justify-self** control the cell content’s horizontal and vertical alignment within its grid container.



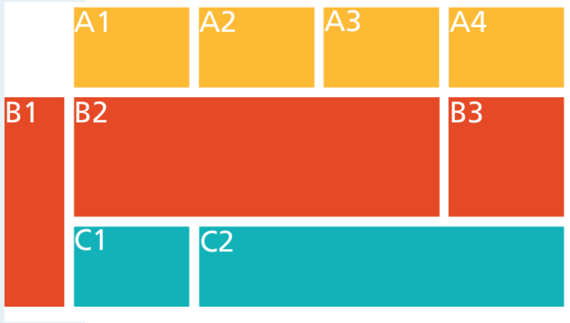
```
main {
   display: grid;
   grid-template-columns: 1fr 4fr;
}
```

```
<main>
  <aside>
    <h2>Filter</h2>
    ...
  </aside>
  <section>
    <div class="card">...
    <div class="card">...
    ...
  </section>
</main>
```

```
section {
   display: grid;
   gap: 1em;
   grid-template-columns: repeat(auto-fit, 220px);
}
```

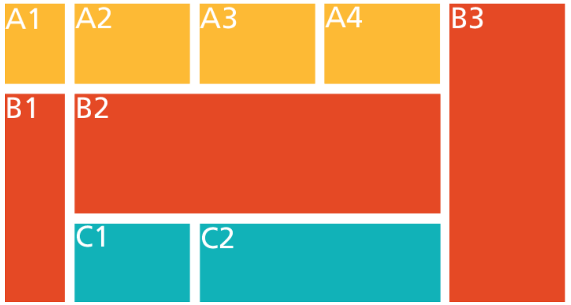
Grid Areas

```
<style>
.container {
  grid-gap: 10px;
  display: grid;
  grid-template-rows: 100px 150px 100px;
  grid-template-columns: 75px 1fr 1fr 1fr 1fr;
  grid-template-areas: ". a1 a2 a3 a4"
                       "b1 b2 b2 b2 b3"
                       "b1 c1 c2 c2 c2";
}
.a1 { grid-area: a1; }
.a2 { grid-area: a2; }
.a3 { grid-area: a3; }
.a4 { grid-area: a4; }
.b1 { grid-area: b1; }
.b2 { grid-area: b2; }
.b3 { grid-area: b3; }
.c1 { grid-area: c1; }
.c2 { grid-area: c2; }
</style>

<section class="container">
  <div class="yellow a1">A1</div>
  <div class="yellow a2">A2</div>
  <div class="yellow a3">A3</div>
  <div class="yellow a4">A4</div>
  <div class="orange b1">B1</div>
  <div class="orange b2">B2</div>
  <div class="orange b3">B3</div>
  <div class="cyan c1">C1</div>
  <div class="cyan c2">C2</div>
</section>
```

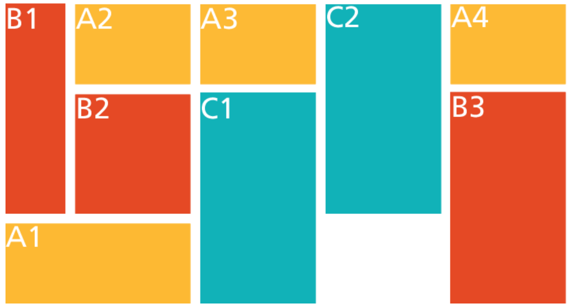


LISTING 7.2 Using grid areas

Grid Areas (ii)



```
grid-template-areas: "a1 a2 a3 a4 b3"
                     "b1 b2 b2 b2 b3"
                     "b1 c1 c2 c2 b3";
```

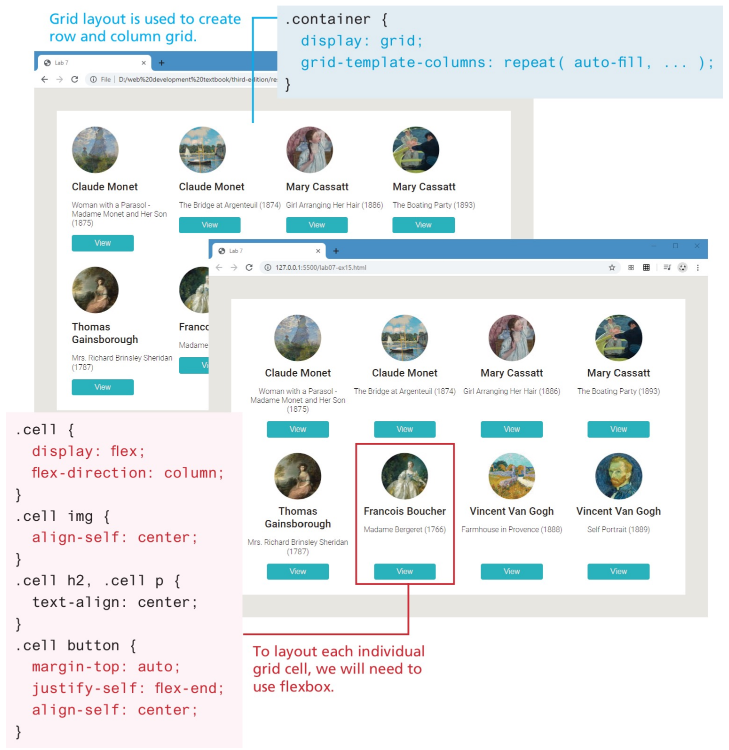


```
grid-template-areas: "b1 a2 a3 c2 a4"
                     "b1 b2 c1 c2 b3"
                     "a1 a1 c1 .. b3";
```

LISTING 7.2 Using grid areas

Grid and Flexbox Together

- **grid** and **flexbox** each have their strengths and these strengths can be combined
- **grid** layout is ideal for constructing the layout structure of your page
- **flexbox** is ideal for laying out the contents of a grid cell.


Grid layout is used to create row and column grid.

```
.container {
  display: grid;
  grid-template-columns: repeat( auto-fill, ... );
}
```

```
.cell {
  display: flex;
  flex-direction: column;
}
.cell img {
  align-self: center;
}
.cell h2, .cell p {
  text-align: center;
}
.cell button {
  margin-top: auto;
  justify-self: flex-end;
  align-self: center;
}
```

To layout each individual grid cell, we will need to use flexbox.