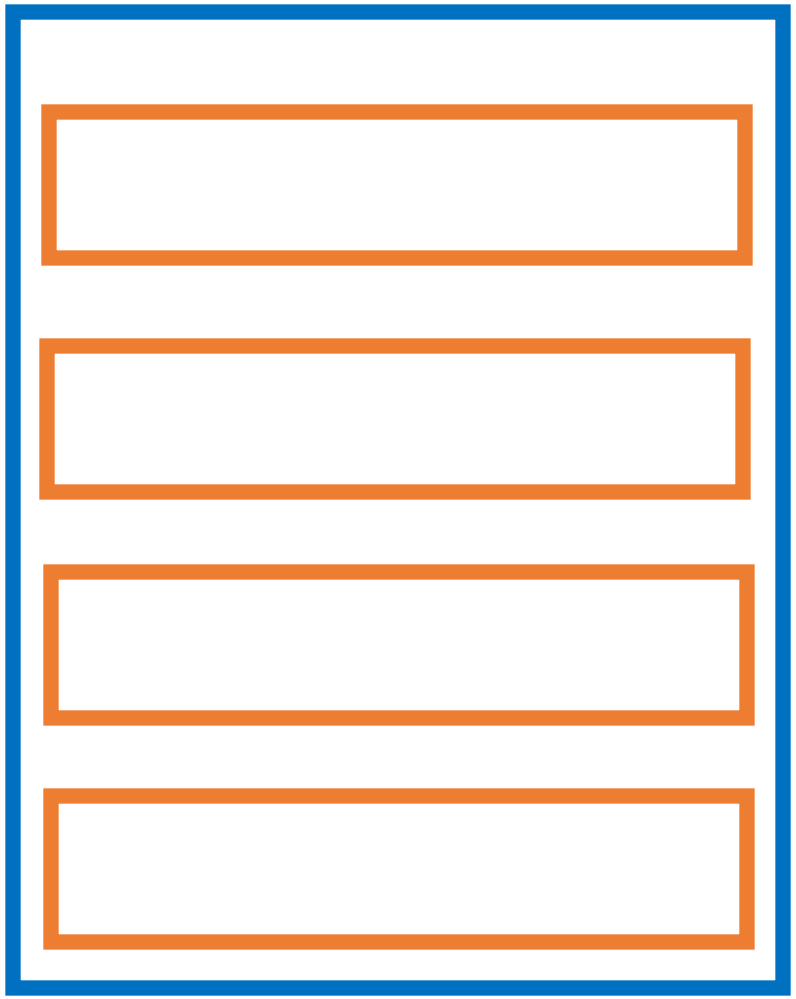
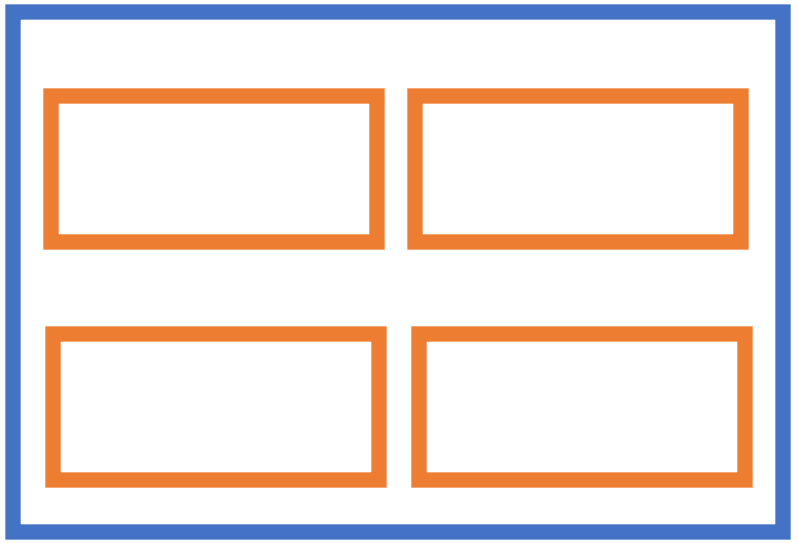
Introduction to CSS Grid

By default, elements with **display:block** will stack on top of each other, like this:



However, sometimes we would prefer elements to be laid out in a grid, like this:

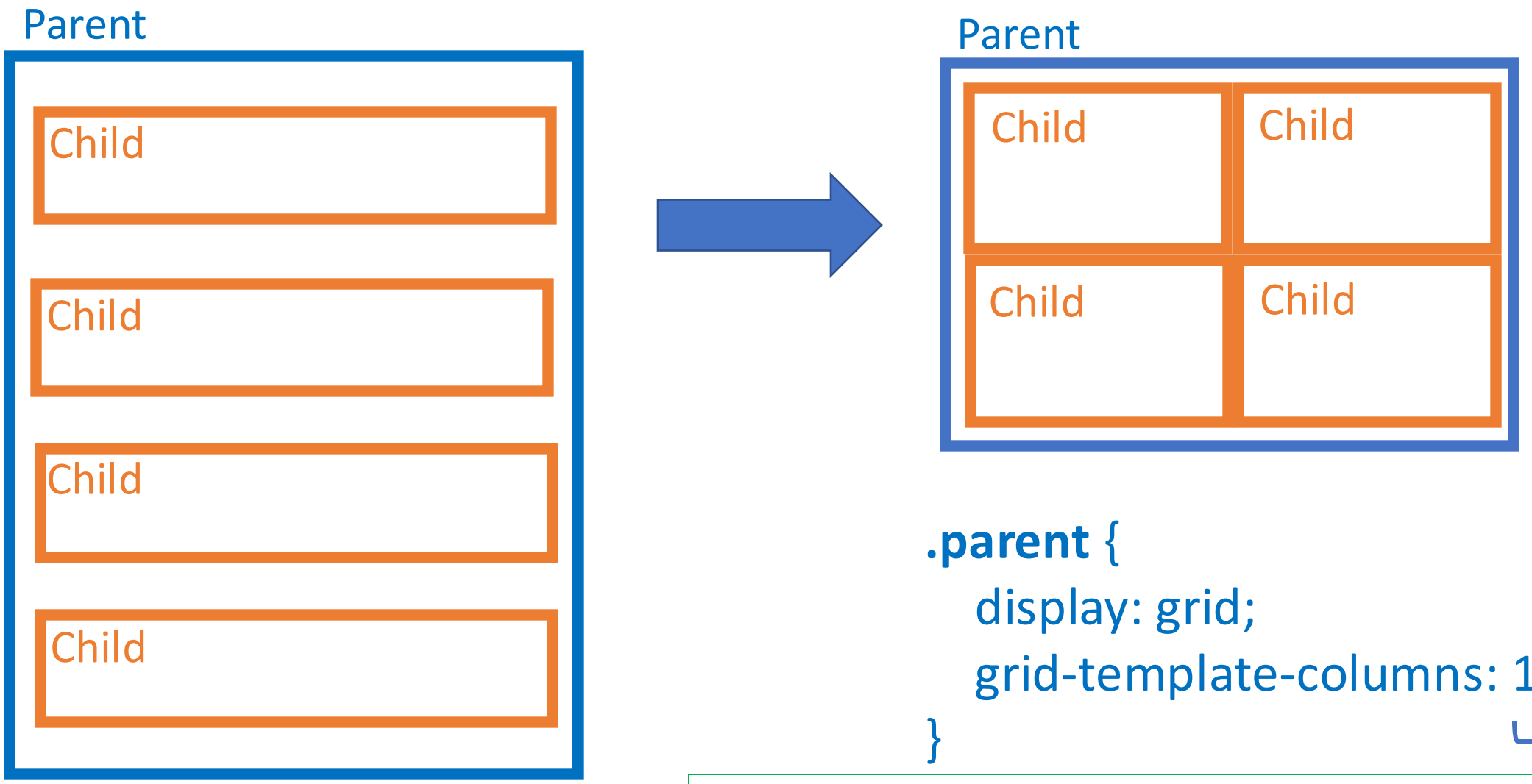


Display:grid

We can create a grid layout with CSS by using **display: grid**

We apply **display:grid** to the **parent** element that contains the elements we would like to become grid cells.

We can then set the widths of the columns in the grid using the **grid-template-columns** property.




```
.parent {
    display: grid;
    grid-template-columns: 1fr 1fr;
}
```

Relative column widths that resize automatically when the parent is resized. Note that we're using the **fr** unit, which stands for **fraction.**

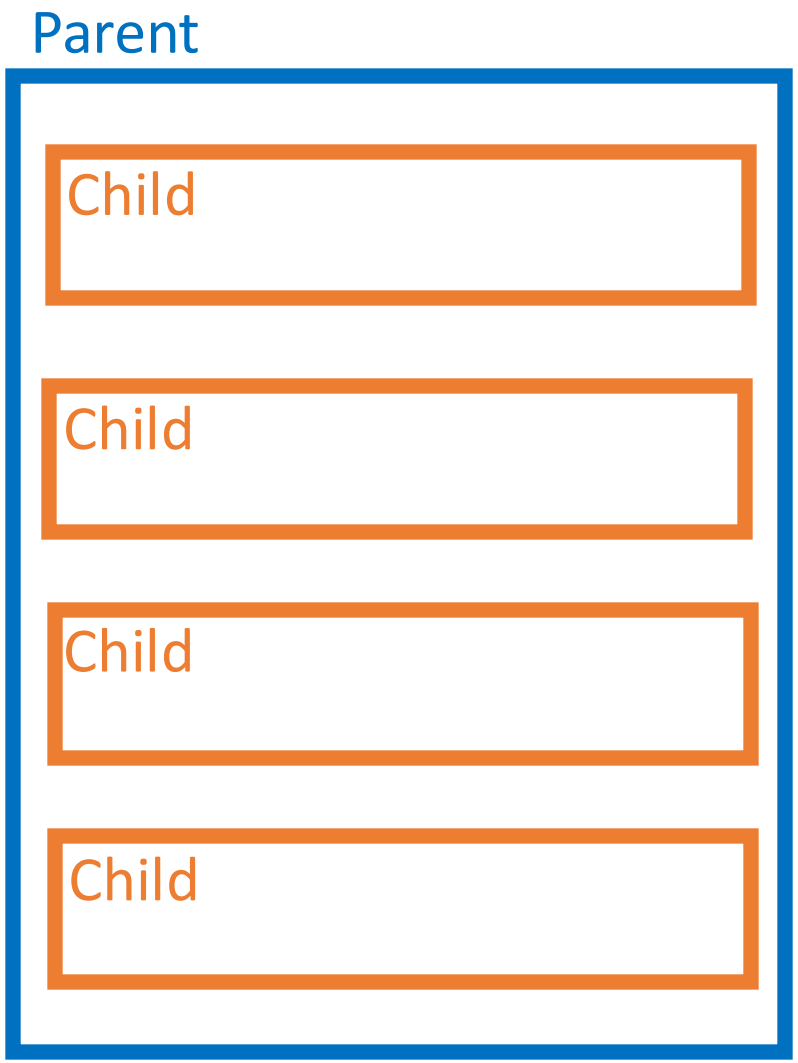
Here's the full code for the example on the previous slide:

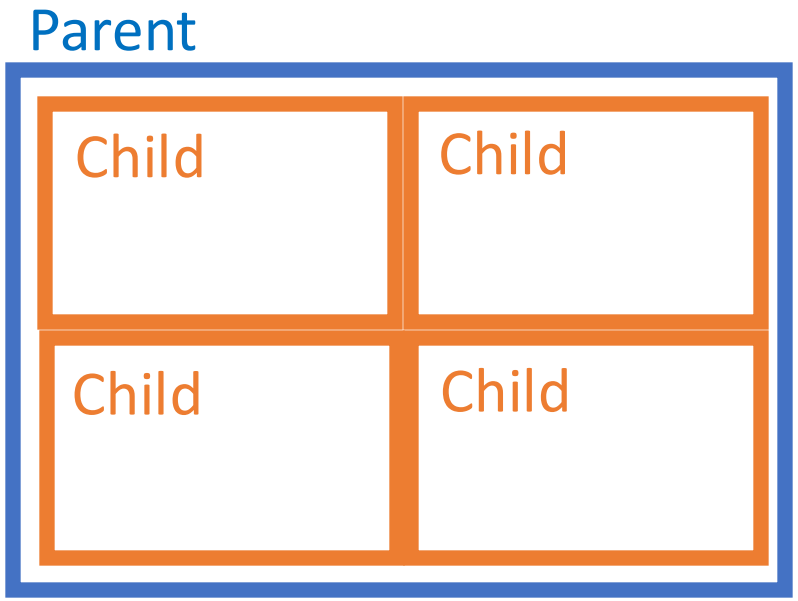
HTML

```
<div class="parent">
  <div class="child"></div>
  <div class="child"></div>
  <div class="child"></div>
  <div class="child"></div>
</div>
```

CSS

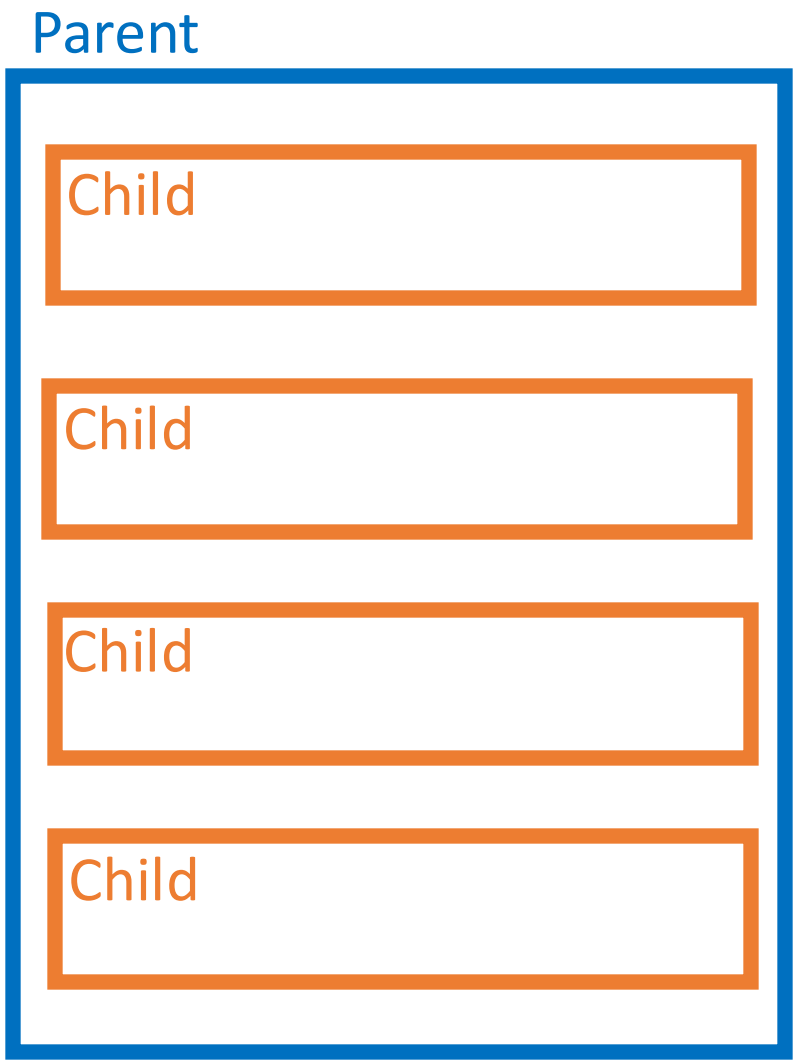
```
.parent {
  border: 3px solid blue;
  display: grid;
  grid-template-columns: 1fr 1fr;
}
.child {
  height: 200px;
  border: 3px solid orange;
}
```

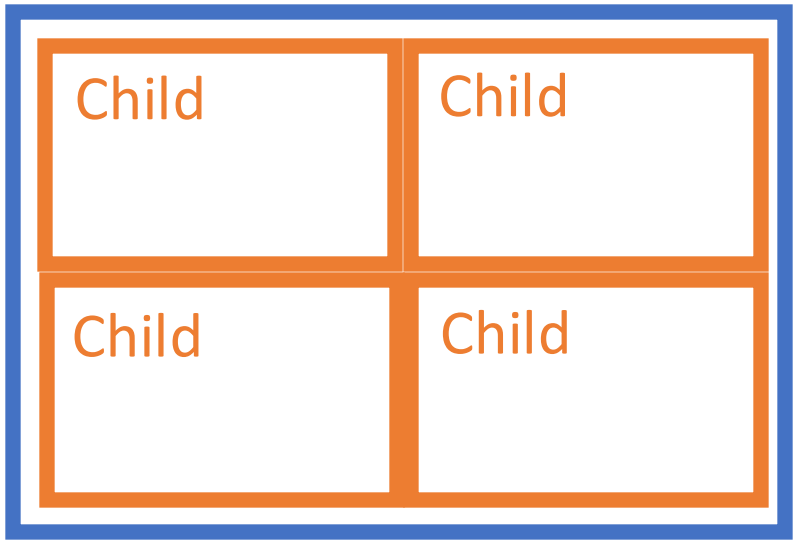





```
.parent {
    display: grid;
    grid-template-columns: 1fr 1fr;
}
```

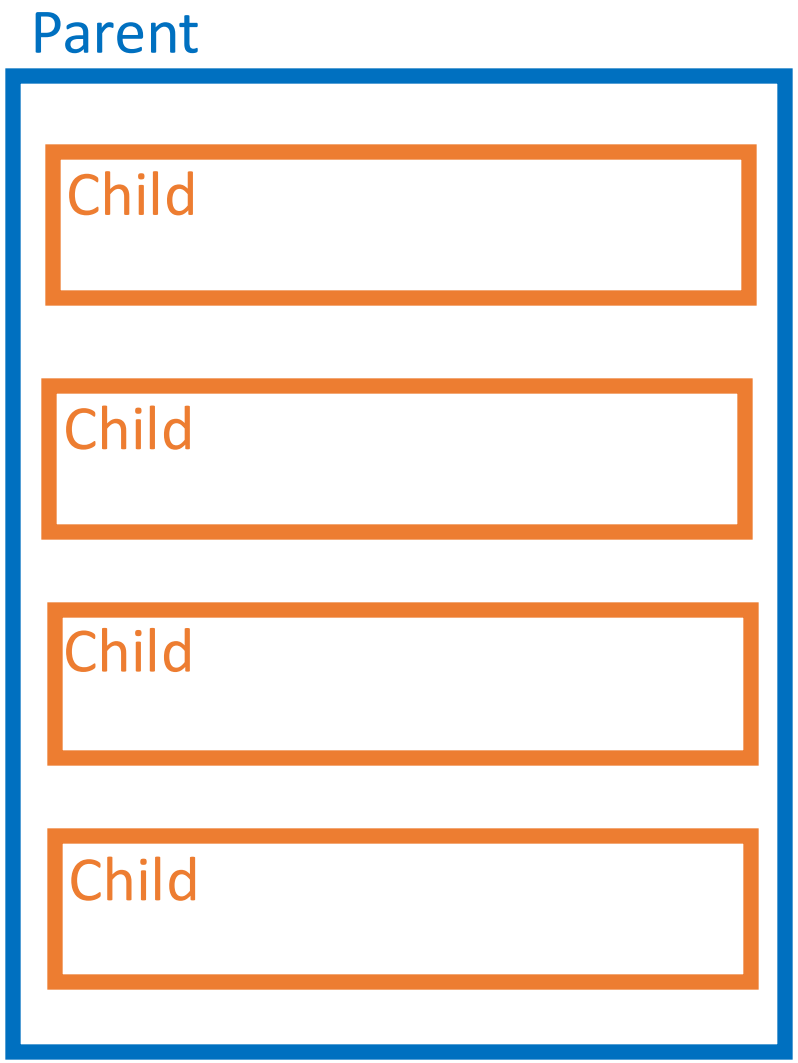
Because we're creating two columns in this example, we're providing two values for the **grid-template-columns** property: one for each column.

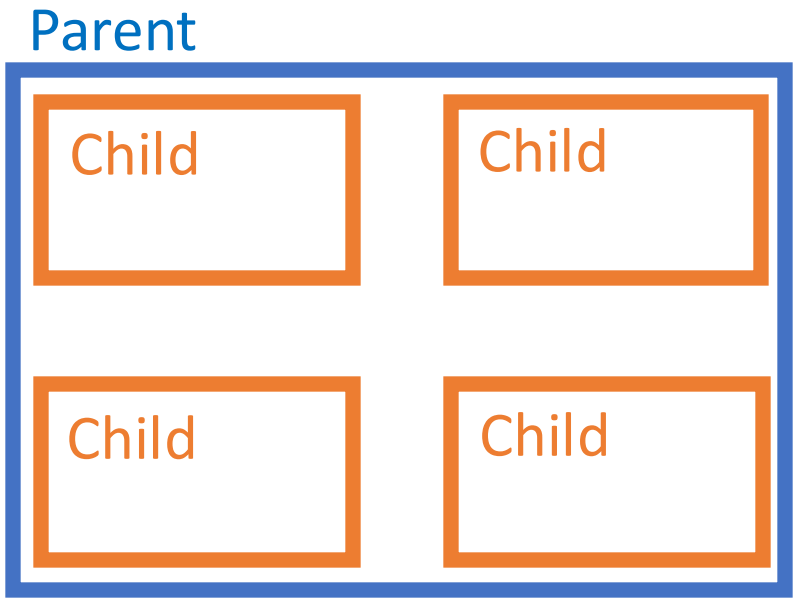





```
.parent {
    display: grid;
    grid-template-columns: 1fr 1fr;
}
```

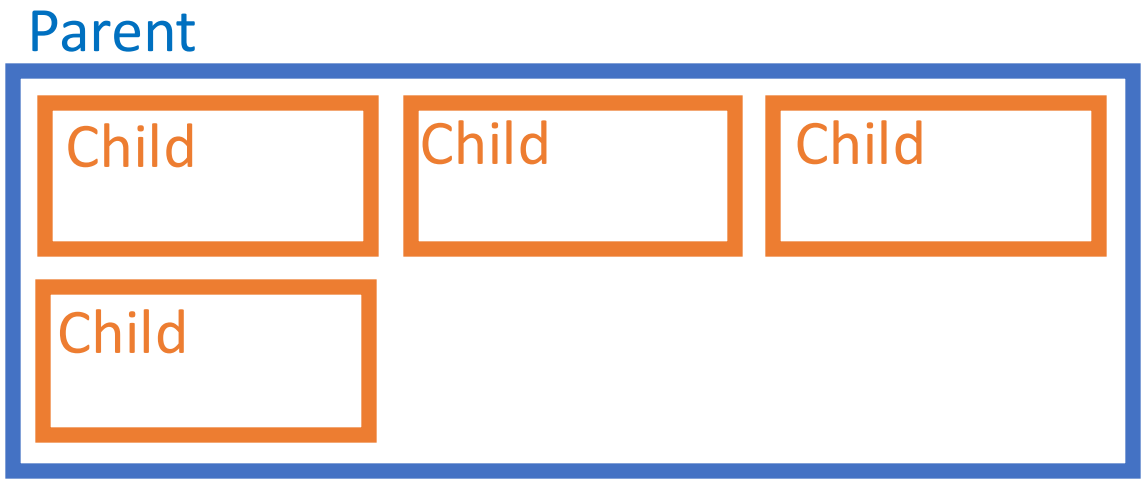
Since both values are the same in this example, that means the columns will have the same width.





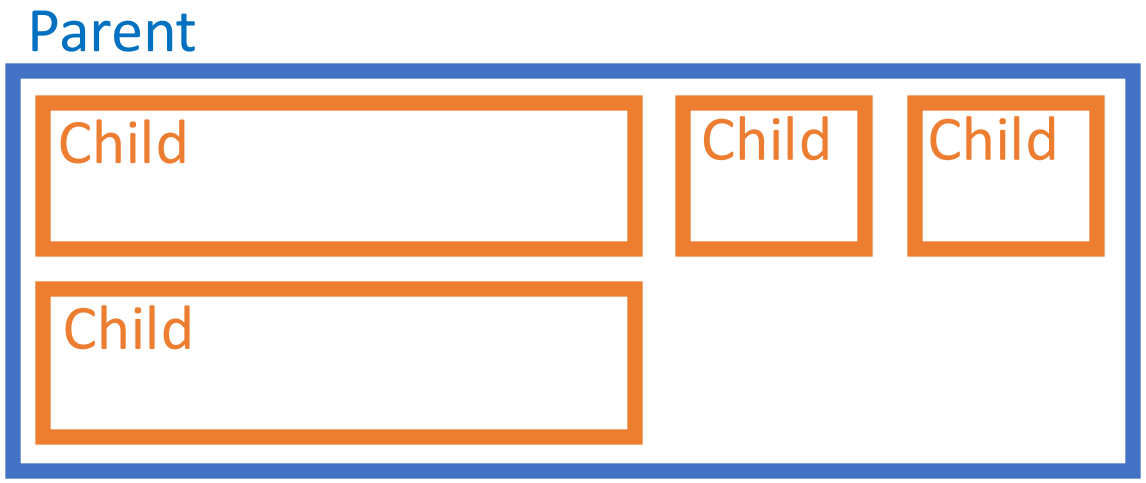

```
.parent {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 1rem;
}
```

The gap property can be used to put space between the grid elements.



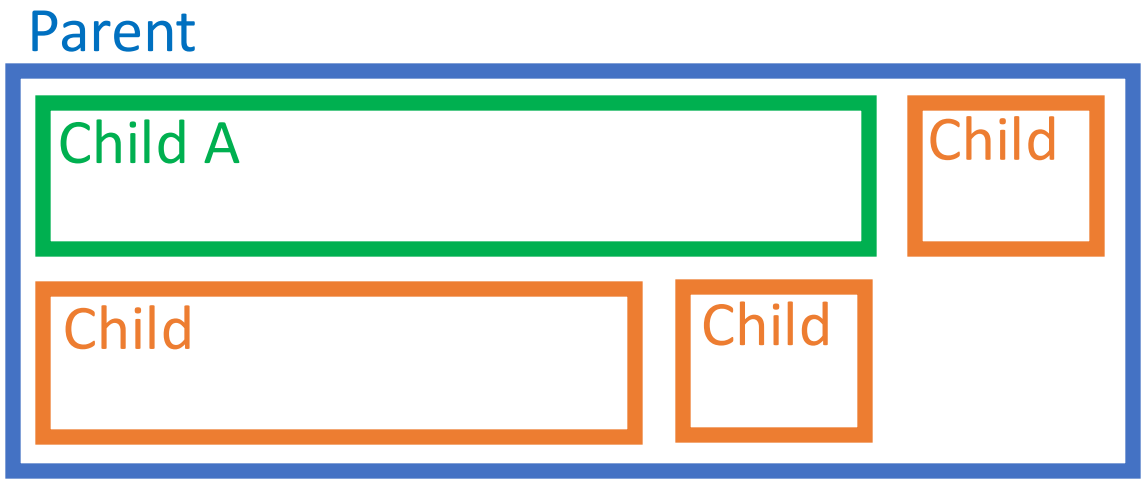

```
.parent {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    gap: 1rem;
}
```

By specifying three values for grid-template-columns, we've set up our grid to have three columns.




```
.parent {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    gap: 1rem;
}
```

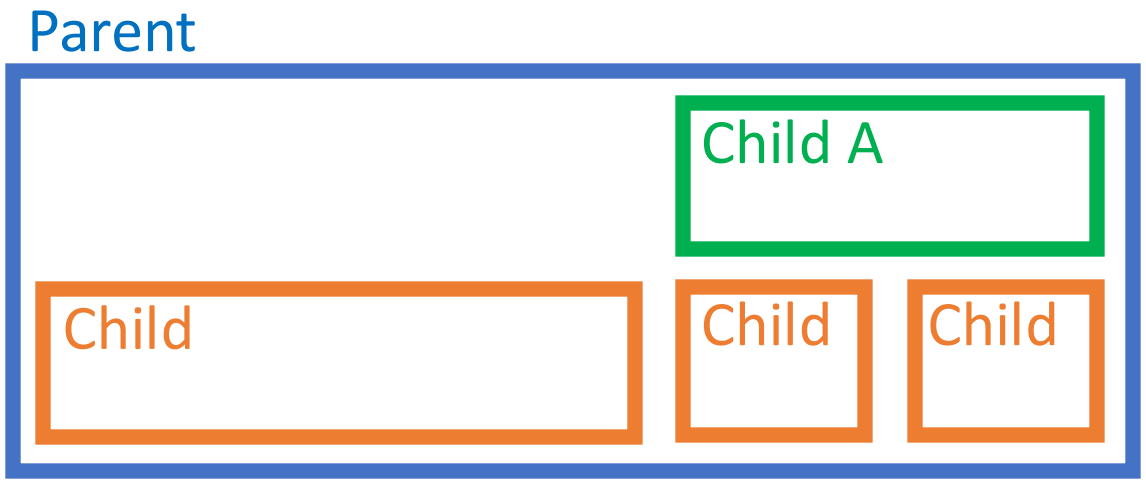
By making the value for the first column 2x the value of the second and third columns, we've made it twice as wide.



Suppose we want **Child A** to span **two** columns instead of **one**. We can use the **grid-column** property in the style for the Child A element.

```
.parent {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    gap: 1rem;
}
```

```
.child-a {
    grid-column: span 2;
}
```



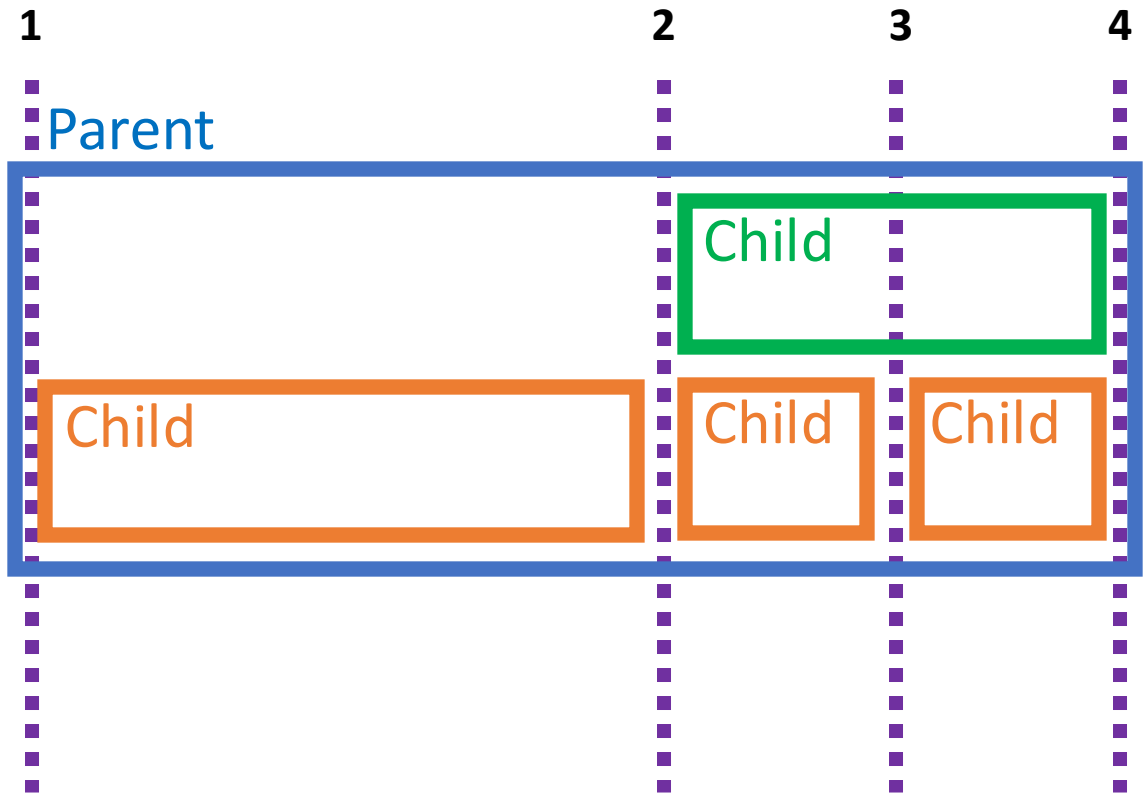
Now suppose we want **Child A** to start in the second column, and span two columns. We can do it like this:

```
.parent {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    gap: 1rem;
}
```

```
.child-a {
    grid-column: 2 / span 2;
}
```

Column where the element starts

Number of columns to span



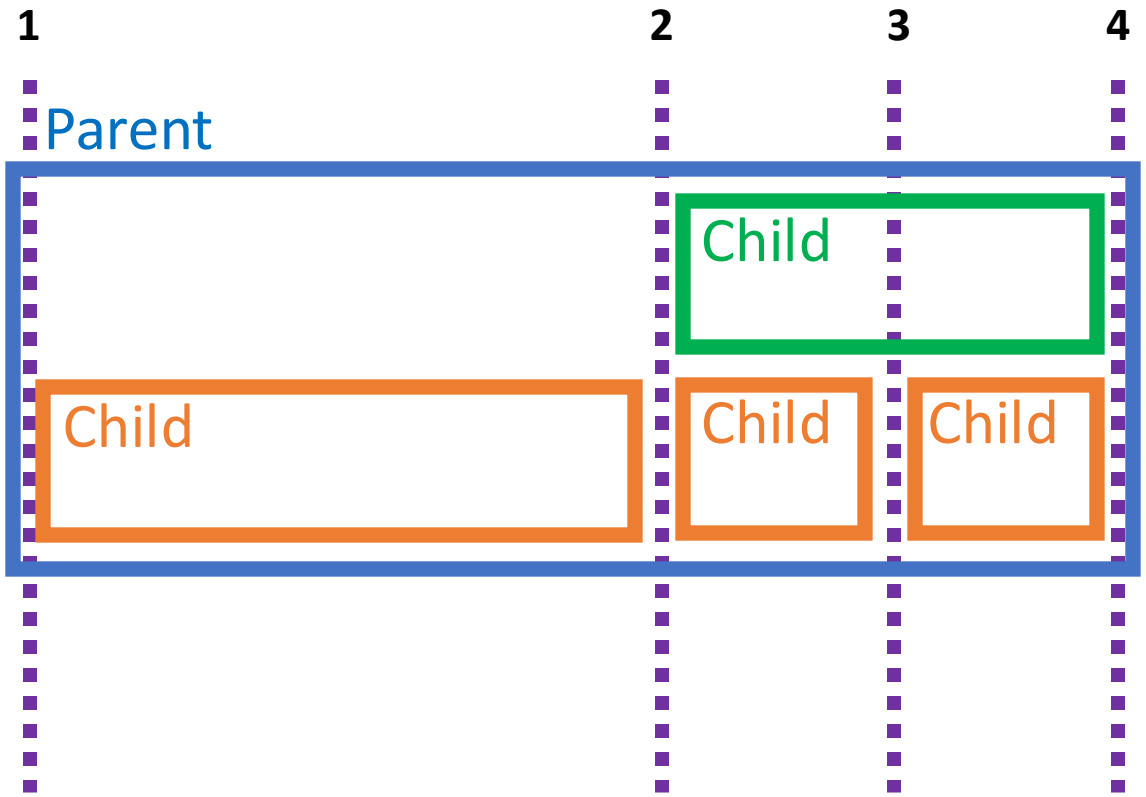
Columns are numbered from left to right:

```
.parent {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    gap: 1rem;
}
```

```
.child-a {
    grid-column: 2 / span 2;
}
```

Column where the element starts

Number of columns to span




```
.parent {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    gap: 1rem;
}
```

We could also do it this way:

```
.child-a {
    grid-column: 2 / 4;
}
```

Column where the element **starts**

Column where the element **ends**

Here's the full code for the example on the previous slide:

HTML

```
<div class="parent">
  <div class="child child-a"></div>
  <div class="child child-b"></div>
  <div class="child child-c"></div>
  <div class="child child-d"></div>
</div>
```

CSS

```
.parent {
  border: 3px solid blue;
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
  gap: 1rem
}
.child {
  height: 200px;
  border: 3px solid orange;
}
.child-a {
  border-color: green;
  grid-column: 2 / 4;
}
```

A few other things about grids:

- Grid column widths can be set using any unit, it doesn't have to be **fr.** For example, one could use **px, rem,** or **%**. If using a fixed-size unit like **px** or **rem**, those columns will NOT resize according to the container:

  grid-template-columns: 200px 1fr 2fr 4rem 15%;

- When using the **grid-column** property to reposition a cell, one can use **negative column numbers** to count from **right to left** instead of **left to right**. For example, for a **5-column** template:

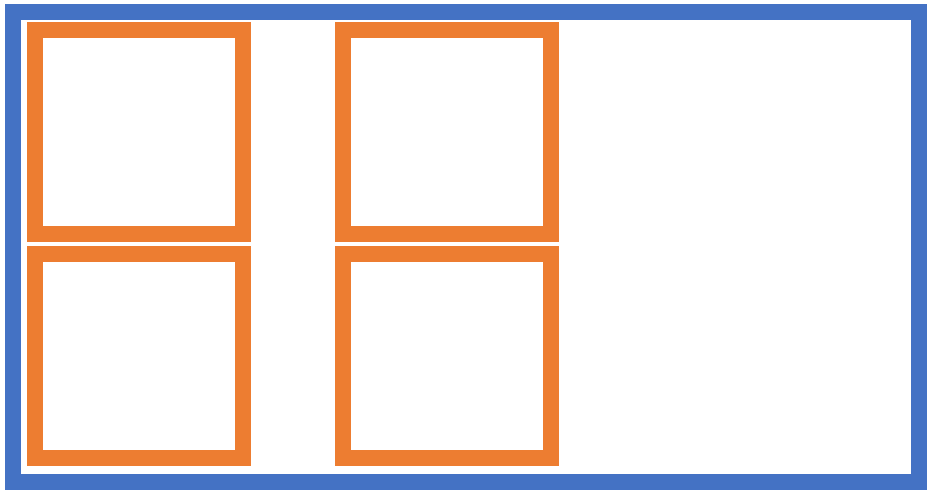
  grid-column: 1 / 6; is the same as **grid-column: -6 / -1;** or
  grid-column: 1 / -1;

Grid Rows

You may have guessed that we can also use the **grid-template-rows** and **grid-row** properties! Try them to see how they work. However, you can often achieve what you want simply by using **grid-template-columns** and **grid-column.** The CSS grid system will automatically create rows in a sensible way that usually suits one's needs.

Grid Justification and Alignment

- The grid system has powerful tools for positioning cells withing grids. Consider the following example:

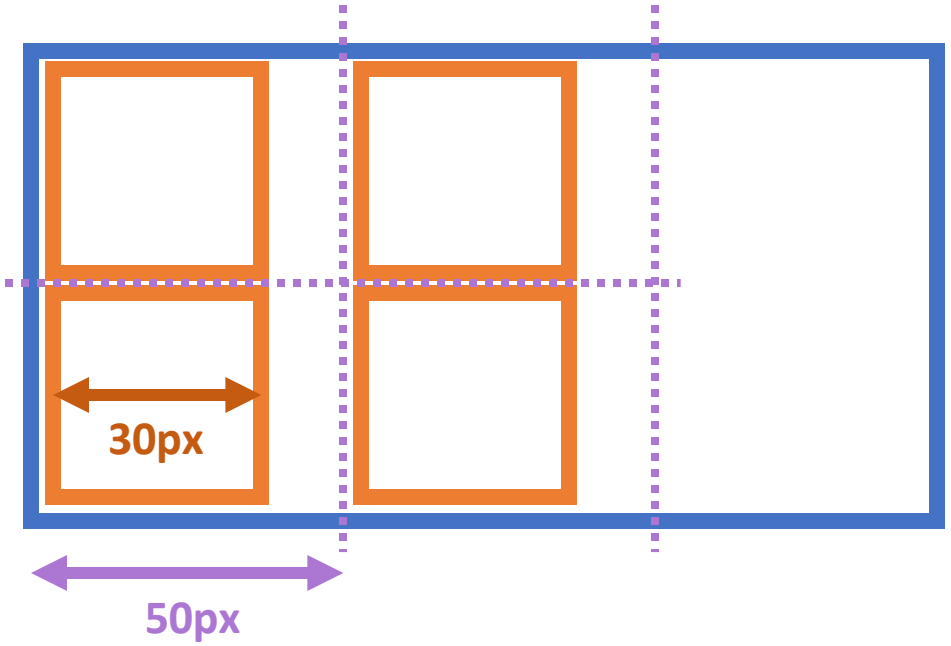


```
.parent {
  border: 3px solid blue;
  display: grid;
  grid-template-columns: 50px 50px;
  width:150px;
}
```

```
.child {
  border: 3px solid orange;
  width:30px;
  height:30px;
}
```

Grid Justification and Alignment

- The grid system has powerful tools for positioning cells withing grids. Consider the following example:



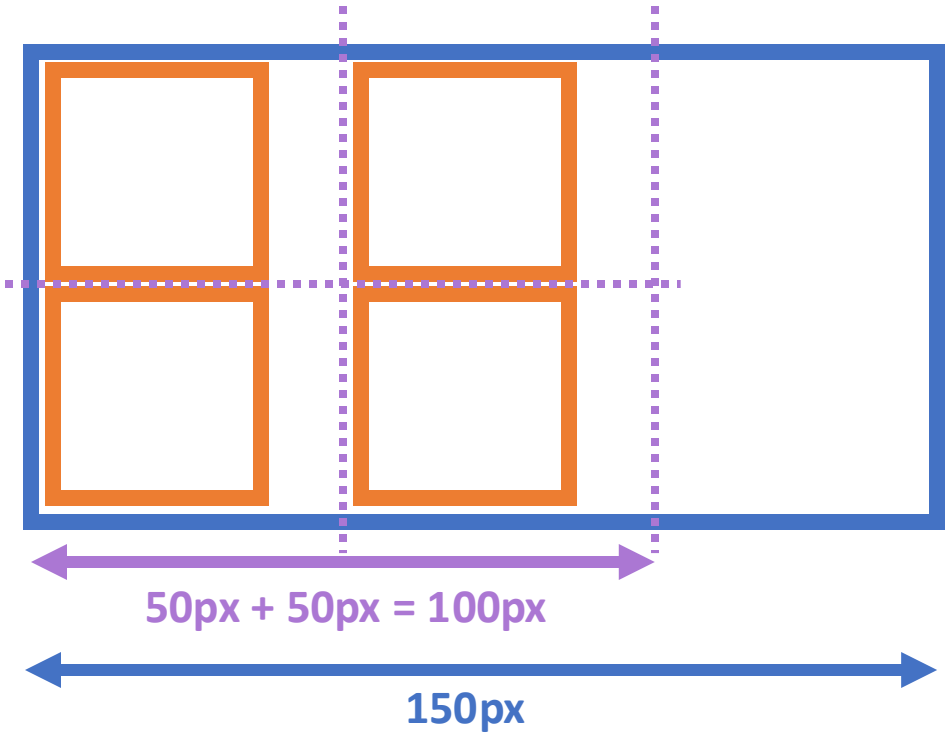

```
.parent {
  border: 3px solid blue;
  display: grid;
  grid-template-columns: 50px 50px;
  width:150px;
}
```

```
.child {
  border: 3px solid orange;
  width:30px;
  height:30px;
}
```

As you can see, the columns are wider than their content, so there's some empty space on the right side of each column.

Grid Justification and Alignment

- The grid system has powerful tools for positioning cells withing grids. Consider the following example:



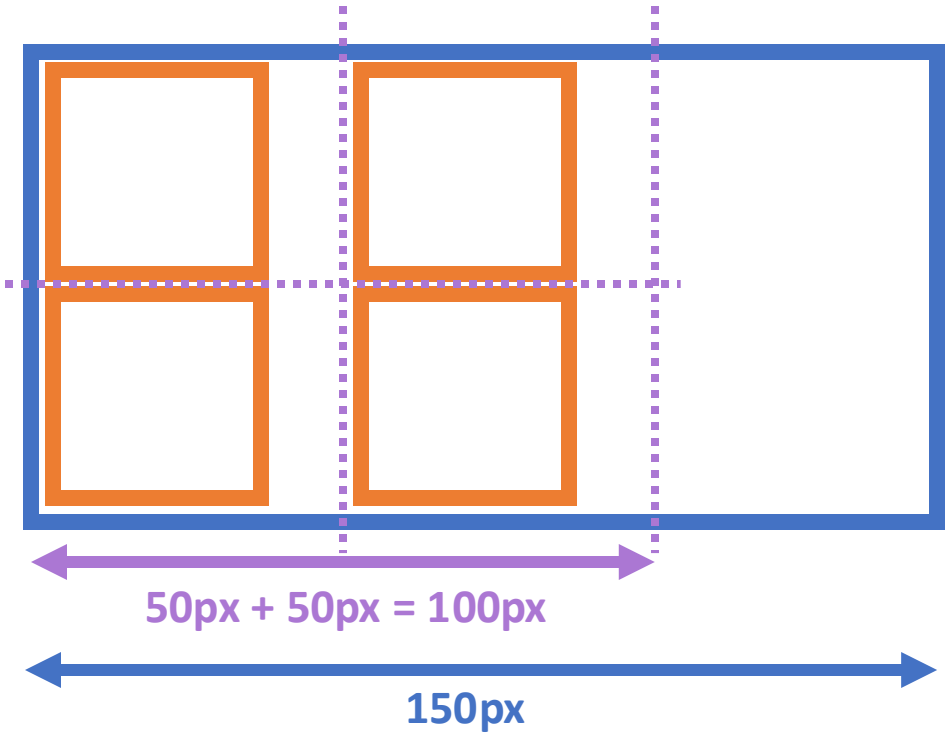

```
.parent {
  border: 3px solid blue;
  display: grid;
  grid-template-columns: 50px 50px;
  width:150px;
}
```

```
.child {
  border: 3px solid orange;
  width:30px;
  height:30px;
}
```

Also, the total width of all columns is less than the width of the container, so there's some empty space on the right side of the grid.

Grid Justification and Alignment

- The grid system has powerful tools for positioning cells withing grids. Consider the following example:



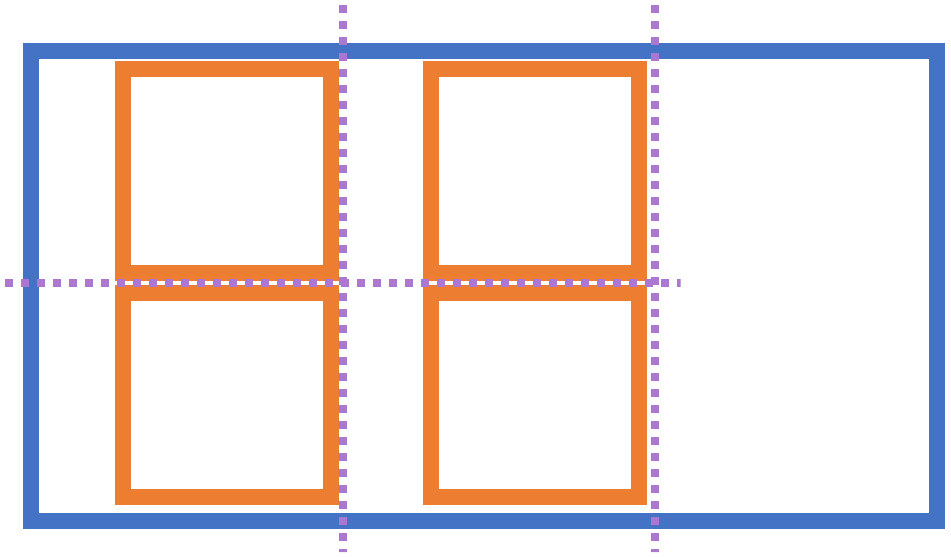

```
.parent {
  border: 3px solid blue;
  display: grid;
  grid-template-columns: 50px 50px;
  width:150px;
}
```

```
.child {
  border: 3px solid orange;
  width:30px;
  height:30px;
}
```

By default, the grid items will sit on the left side of each column, and the whole grid will sit on the left side of the container. But we can change this with **justify-items** and **justify-content**, respectively.

Grid Justification and Alignment

- The grid system has powerful tools for positioning cells withing grids. Consider the following example:



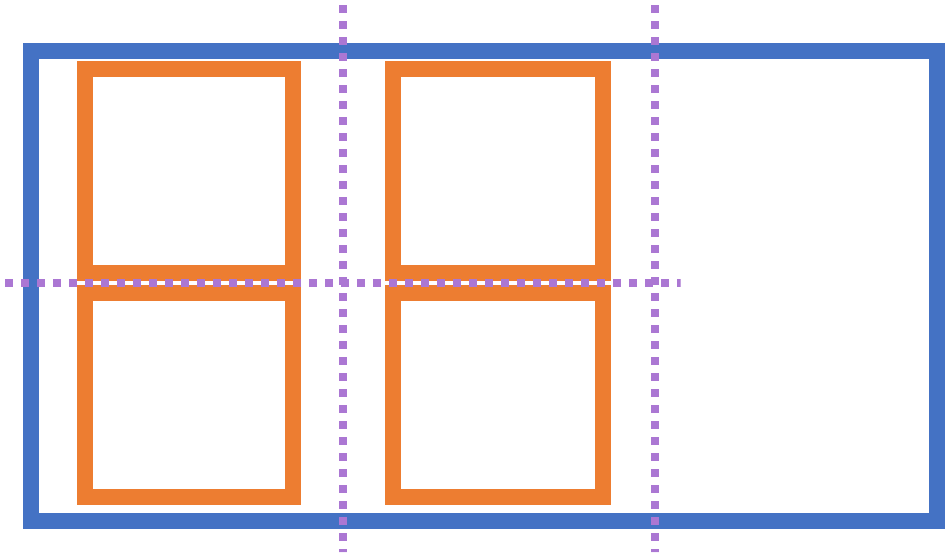
```
.parent {
  border: 3px solid blue;
  display: grid;
  grid-template-columns: 50px 50px;
  width:150px;

  justify-items: end;
}
```

```
.child {
  border: 3px solid orange;
  width:30px;
  height:30px;
}
```

The property **justify-items** positions items **horizontally** within their grid columns. Some values are **end** (right side), **start** (left side, default) and **center.**

Grid Justification and Alignment

- The grid system has powerful tools for positioning cells withing grids. Consider the following example:



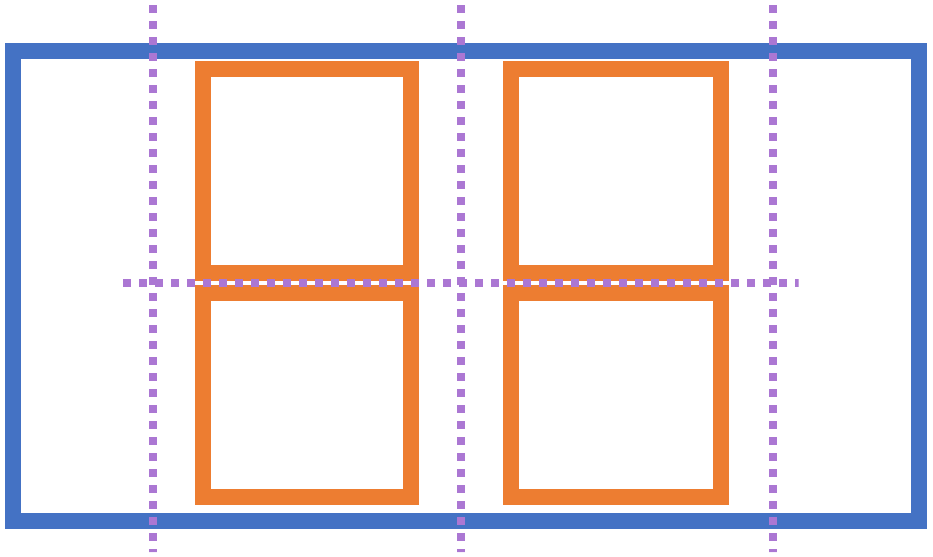
```
.parent {
  border: 3px solid blue;
  display: grid;
  grid-template-columns: 50px 50px;
  width:150px;

  justify-items: center;
}
```

```
.child {
  border: 3px solid orange;
  width:30px;
  height:30px;
}
```

The property **justify-items** positions items **horizontally** within their grid columns. Some values are **end** (right side), **start** (left side, default) and **center.**

Grid Justification and Alignment

- The grid system has powerful tools for positioning cells withing grids. Consider the following example:

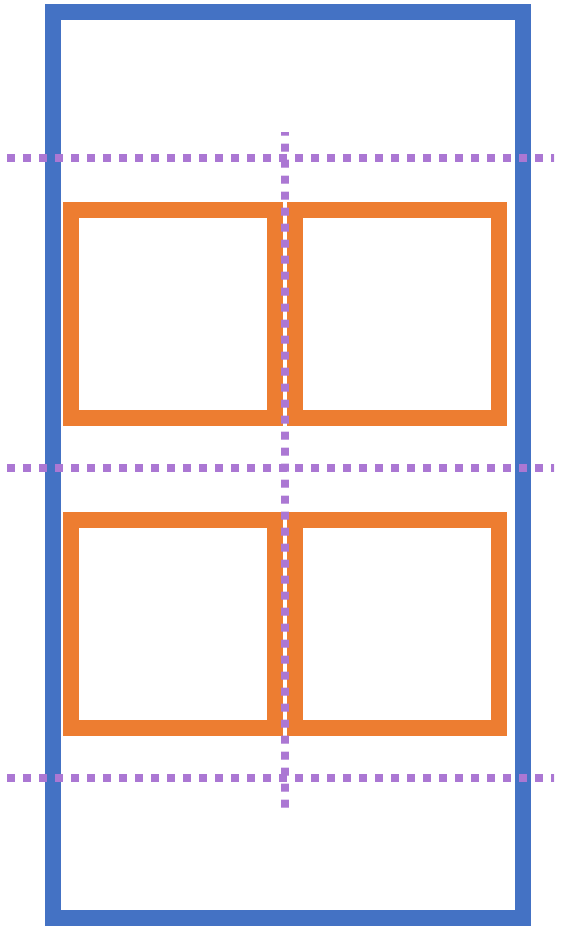


```
.parent {
  border: 3px solid blue;
  display: grid;
  grid-template-columns: 50px 50px;
  width:150px;

  justify-items: center;
  justify-content: center;
}
```

```
.child {
  border: 3px solid orange;
  width:30px;
  height:30px;
}
```

The property **justify-content** can be used to position the whole grid **horizontally** within the container. Some values are **center, end** (right side) and **start** (left side, default).

Grid Justification and Alignment



- The corresponding properties for arranging items and grids **vertically** are **align-items** and **align-content.** On your own, try experimenting with them to see how they work!

Additional Grid Resources

- MDN's grid tutorial: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout
- CSS-Tricks's grid cheat sheet: https://css-tricks.com/snippets/css/complete-guide-grid/
- The Grid Garden browser game (advanced): https://cssgridgarden.com/