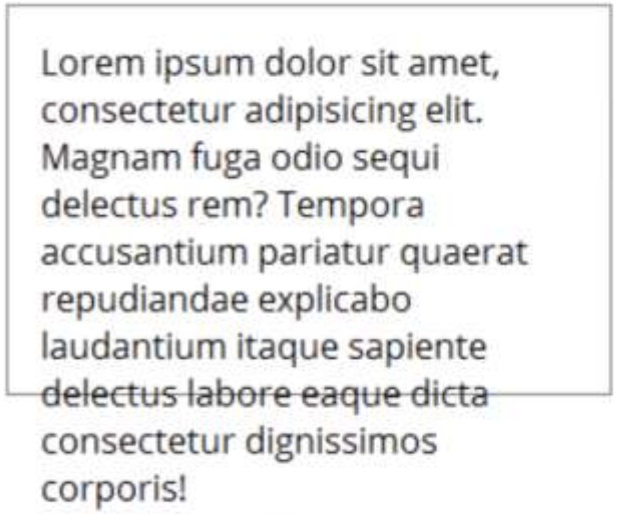
sourcemind

GRID
FLEX

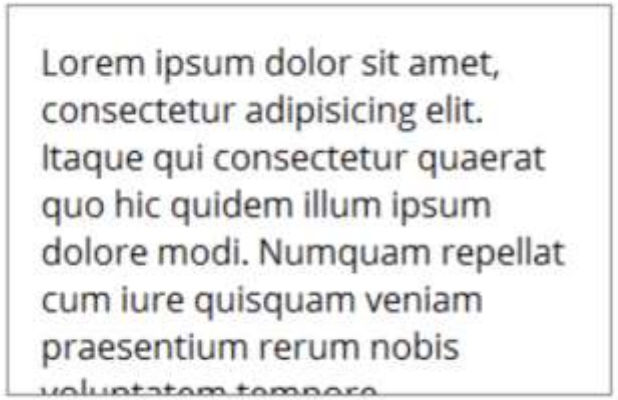
CSS Overflow

The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.



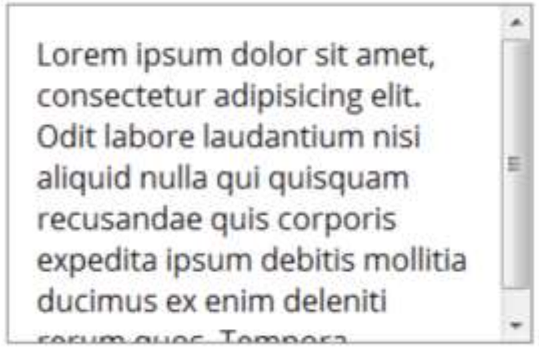
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Magnam fuga odio sequi delectus rem? Tempora accusantium pariatur quaerat repudiandae explicabo laudantium itaque sapiente delectus labore eaque dicta consectetur dignissimos corporis!

overflow-y: visible;



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Itaque qui consectetur quaerat quo hic quidem illum ipsum dolore modi. Numquam repellat cum iure quisquam veniam praesentium rerum nobis

overflow-y: hidden;



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Odit labore laudantium nisi aliquid nulla qui quisquam recusandae quis corporis expedita ipsum debitis mollitia ducimus ex enim deleniti

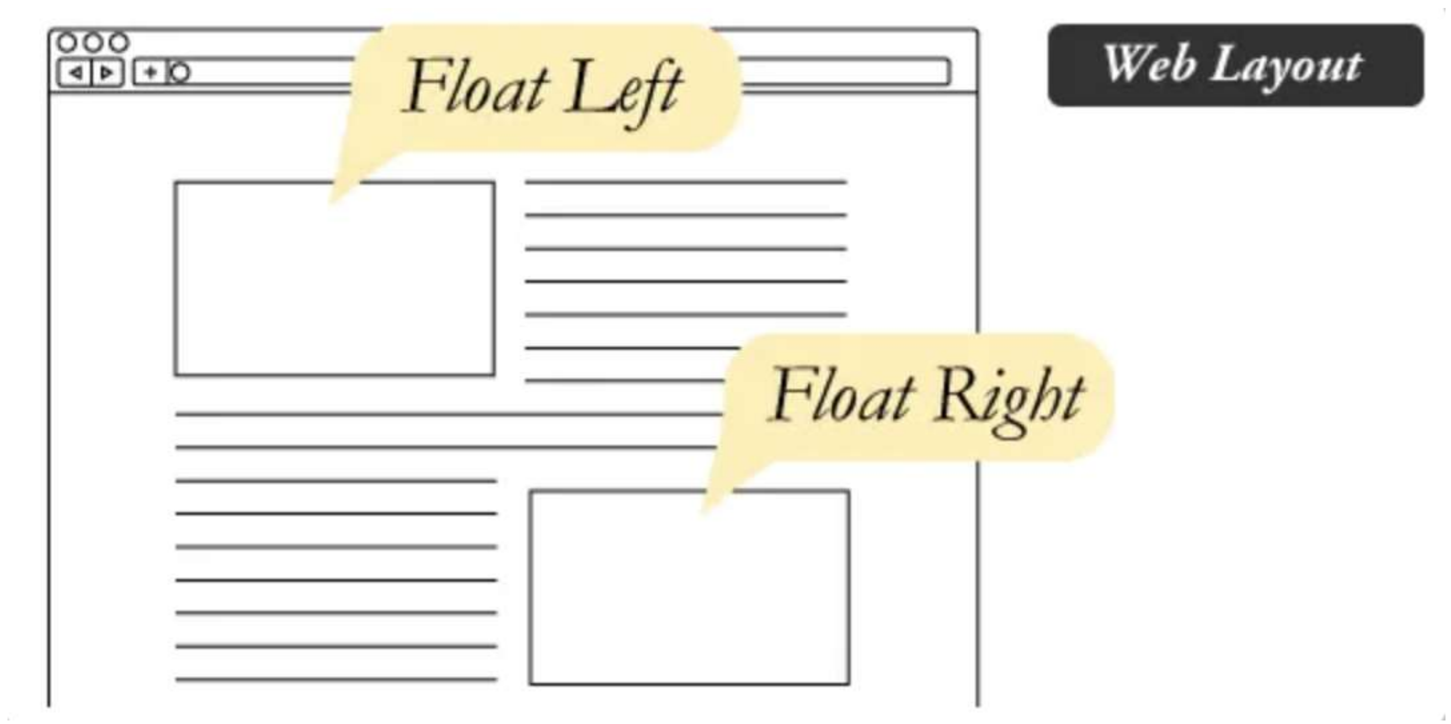
overflow-y: scroll;

- `visible` - Default. The overflow is not clipped. The content renders outside the element's box
- `hidden` - The overflow is clipped, and the rest of the content will be invisible
- `scroll` - The overflow is clipped, and a scrollbar is added to see the rest of the content
- `auto` - Similar to `scroll`, but it adds scrollbars only when necessary

Note: The `overflow` property only works for block elements with a specified height.

The picture can't be displayed.

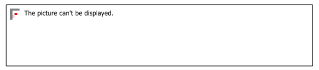
CSS Float



- `left` - The element floats to the left of its container
- `right` - The element floats to the right of its container
- `none` - The element does not float (will be displayed just where it occurs in the text). This is default
- `inherit` - The element inherits the float value of its parent

You can set the margin property to `auto` to horizontally center the element within its container.

```
margin: auto;
```

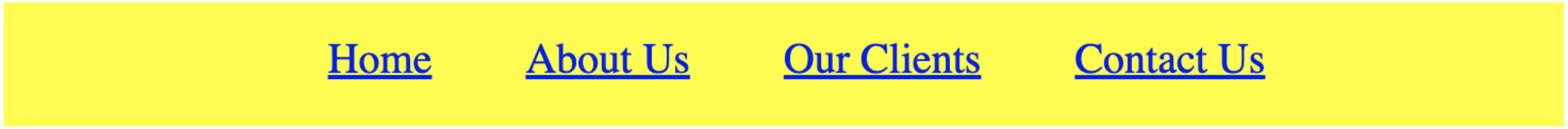


CSS Inline Block

Compared to `display: inline`, the major difference is that `display: inline-block` allows to set a width and height on the element.

Also, with `display: inline-block`, the top and bottom margins/paddings are respected, but with `display: inline` they are not.

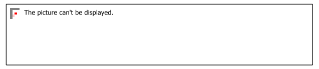
Navigation Bar



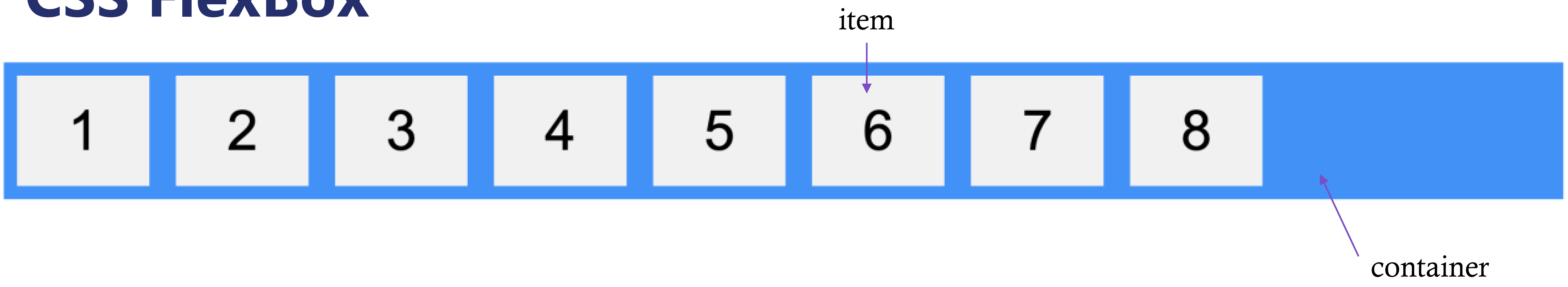
```html
<ul class="nav">
  <li><a href="#home">Home</a></li>
  <li><a href="#about">About Us</a></li>
  <li><a href="#clients">Our Clients</a></li>
  <li><a href="#contact">Contact Us</a></li>
</ul>
```

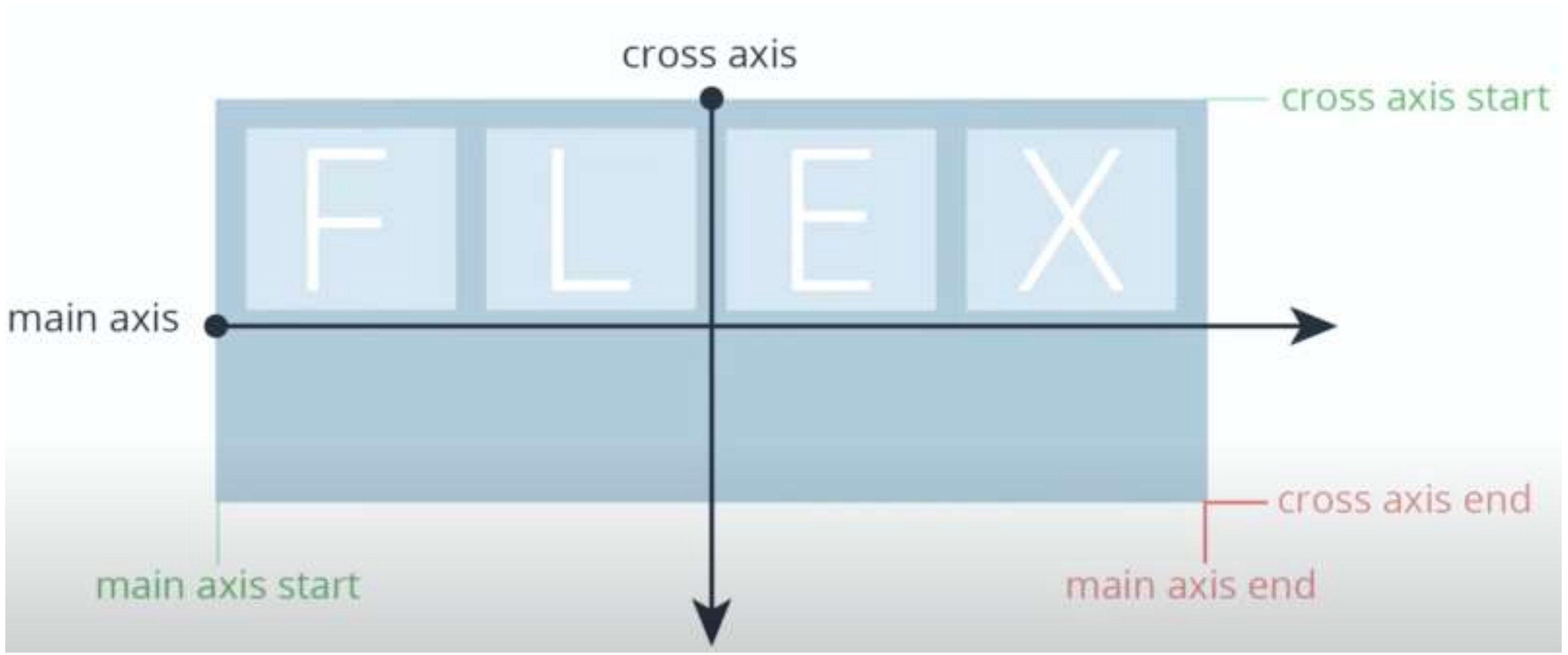
```css
.nav {
  background-color: yellow;
  list-style-type: none;
  text-align: center;
  margin: 0;
  padding: 0;
}

.nav li {
  display: inline-block;
  font-size: 20px;
  padding: 20px;
}
```

```css
*
  p {
    background-color: red !important;
  }
```



CSS FlexBox





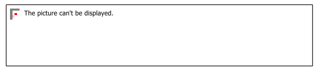
```
.flex-container {
  display: flex;
}
```

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

CSS FlexBox (Container)

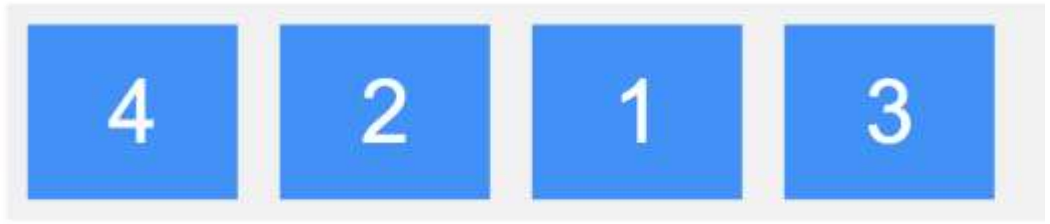
- `flex-direction` which direction the container wants to stack the flex items.
- `flex-wrap` specifies whether the flex items should wrap or not.
- `flex-flow` is a shorthand property for setting both the `flex-direction` and `flex-wrap` properties.
- `justify-content` property is used to align the flex items:
- `align-items` property is used to align the flex items.
- `align-content` property is used to align the flex lines.

```css
.flex-container {
  display: flex;
  flex-flow: row wrap;
}
```

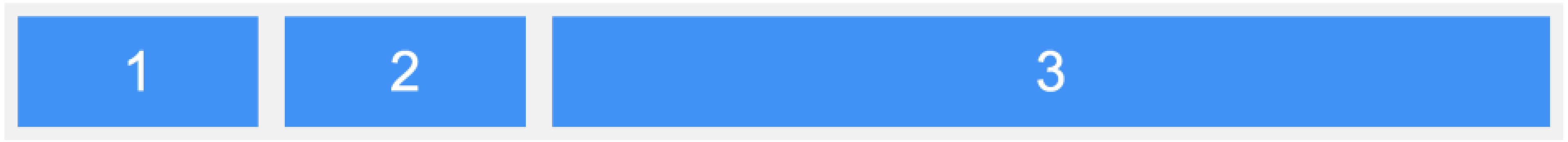

The picture can't be displayed.

CSS FlexBox (Item)

The `order` property specifies the order of the flex items.

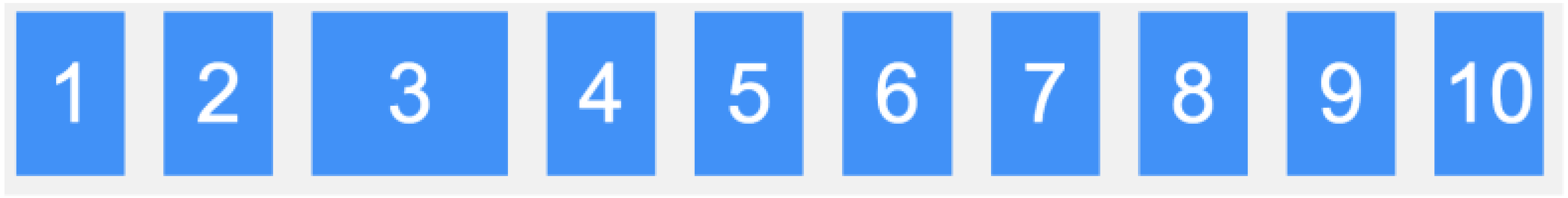


The `flex-grow` property specifies how much a flex item will grow relative to the rest of the flex items.



```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```

The `flex-shrink` property specifies how much a flex item will shrink relative to the rest of the flex items.



```
<div style="flex-shrink: 0">3</div>
```

The `flex-basis` property specifies the initial length of a flex item.

The `flex` property is a shorthand property for the `flex-grow`, `flex-shrink`, and `flex-basis` properties.

```
<div style="flex: 0 0 200px">3</div>
```

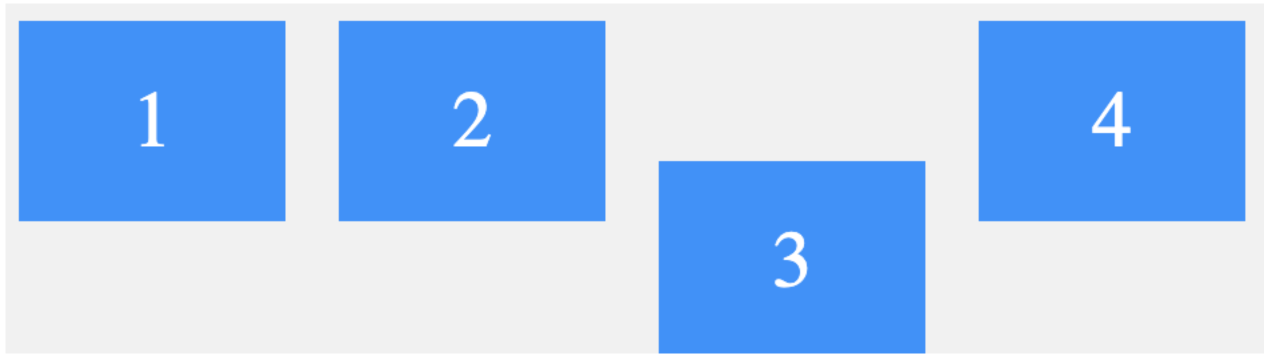
The picture can't be displayed.

CSS FlexBox (Item)

The align-self Property

The `align-self` property specifies the alignment for the selected item inside the flexible container.

The `align-self` property overrides the default alignment set by the container's `align-items` property.

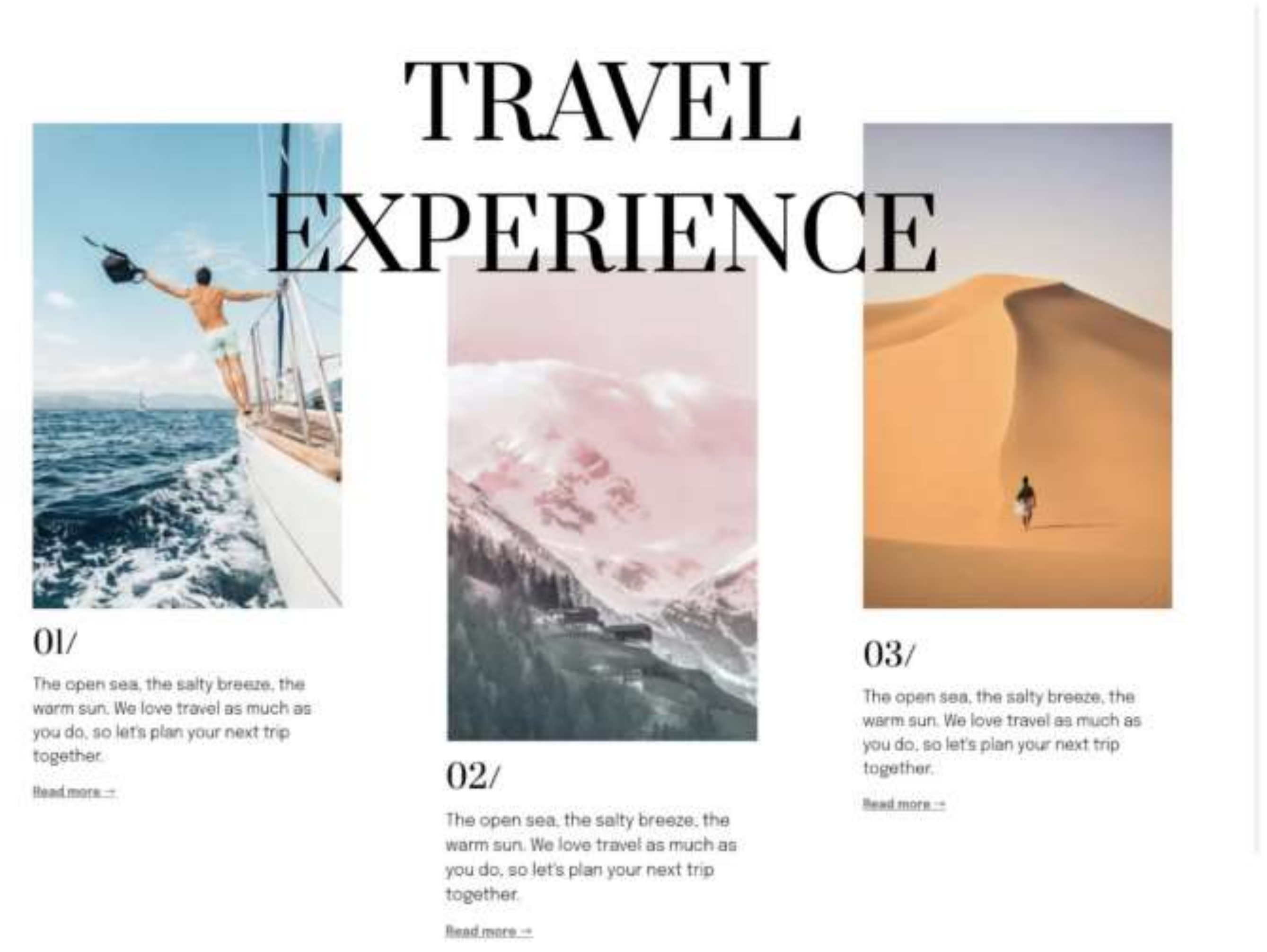


```
.flex-container {
  display: flex;
  height: 200px;
  background-color: #f1f1f1;
  align-items: start
}
```

```
<div style="align-self: center">3</div>
```

The picture can't be displayed.

CSS FlexBox (Example)



TRAVEL EXPERIENCE

01/

The open sea, the salty breeze, the warm sun. We love travel as much as you do, so let's plan your next trip together.

Read more →

02/

The open sea, the salty breeze, the warm sun. We love travel as much as you do, so let's plan your next trip together.

Read more →

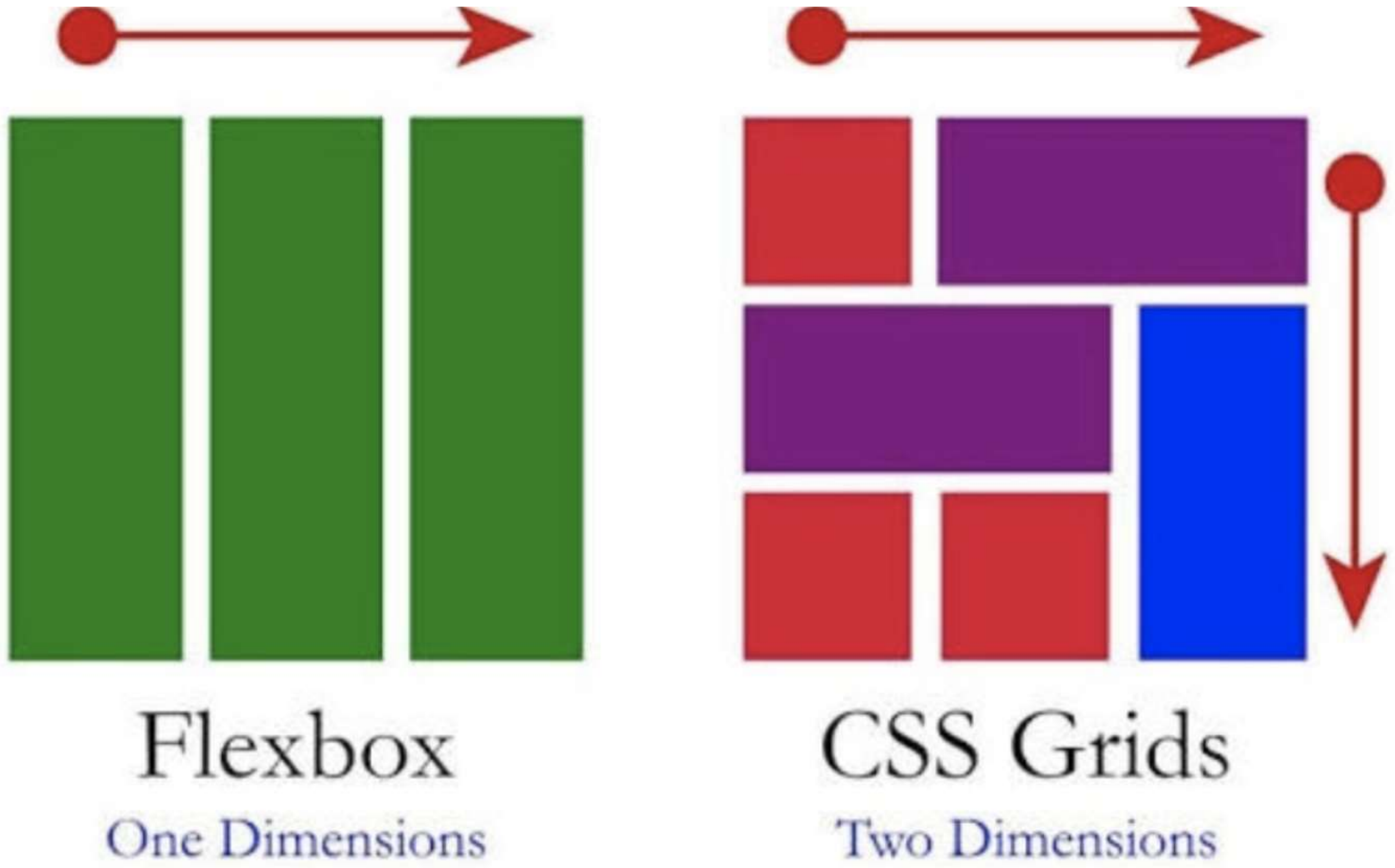
03/

The open sea, the salty breeze, the warm sun. We love travel as much as you do, so let's plan your next trip together.

Read more →

The picture can't be displayed.

CSS GRID VS CSS FLEX

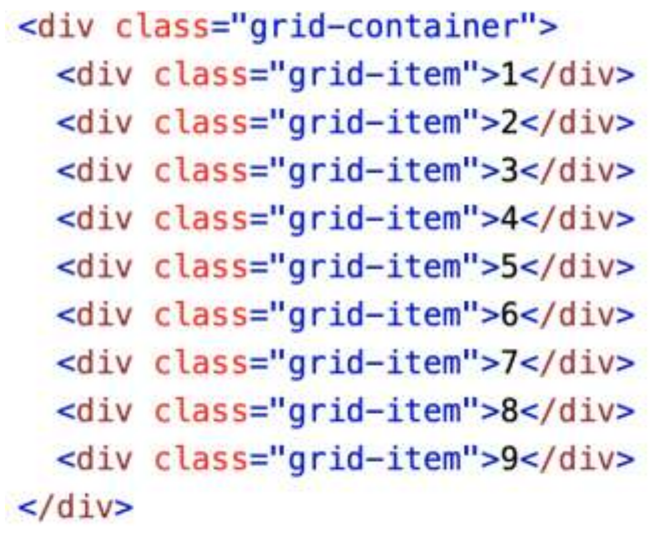


re can't be displayed.

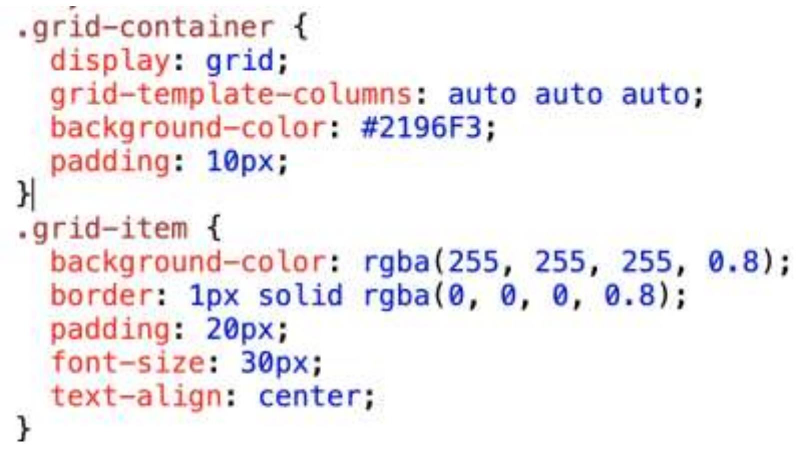
CSS GRID

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

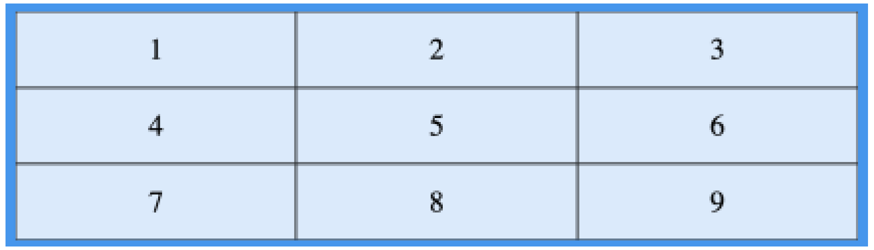
The picture can't be displayed.



```html
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```



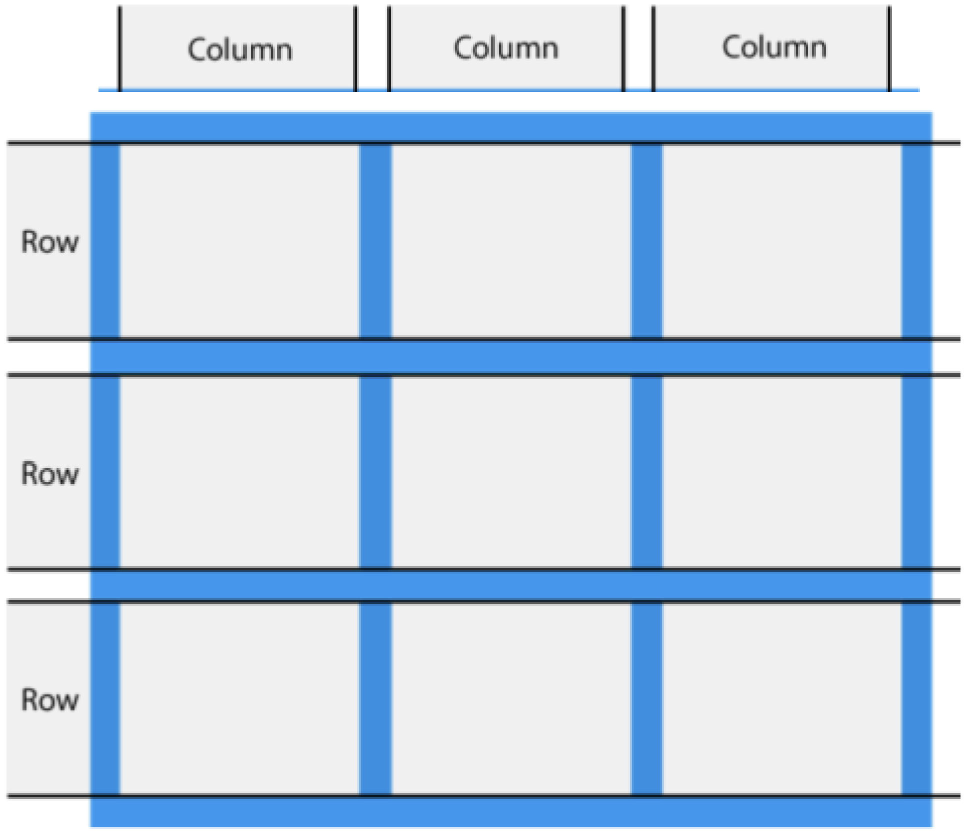
```css
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}
.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
```



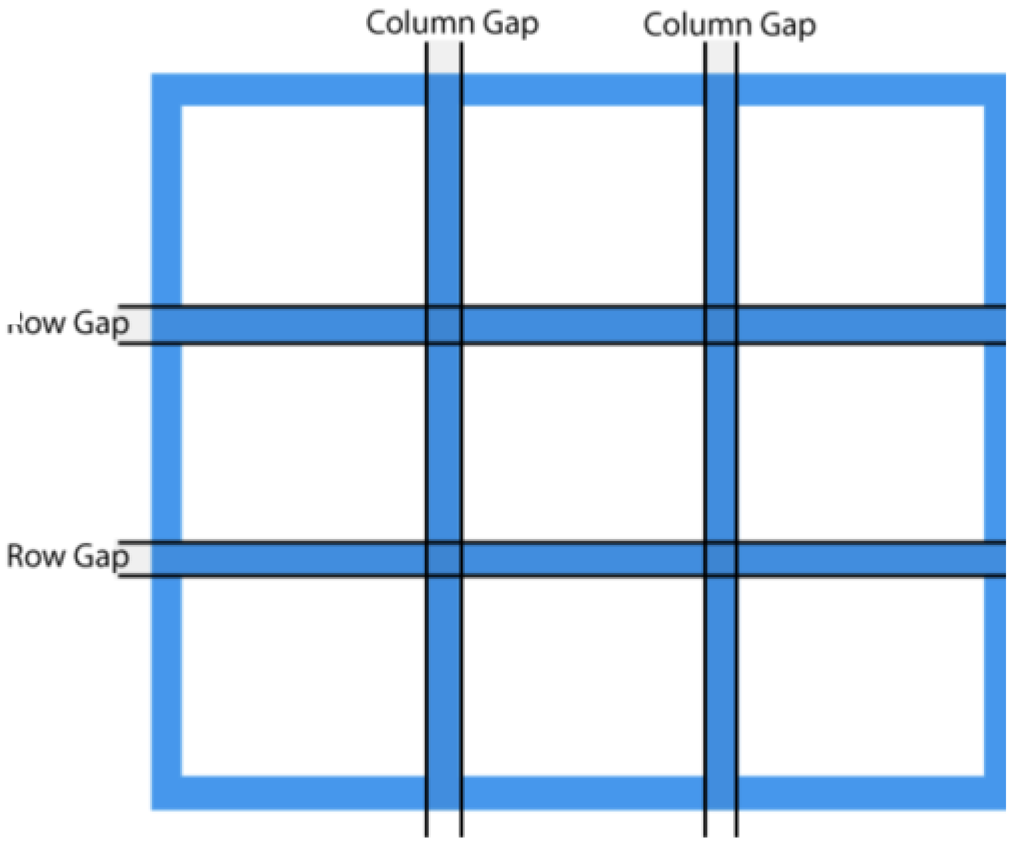
1	2	3
4	5	6
7	8	9

The picture can't be displayed.

GRID Columns & Rows



- `column-gap`
- `row-gap`
- `gap`



The picture can't be displayed.

CSS Grid (Container)

The `grid-template-columns` property defines the number of columns in your grid layout, and it can define the width of each column.

The `grid-template-rows` property defines the height of each row.

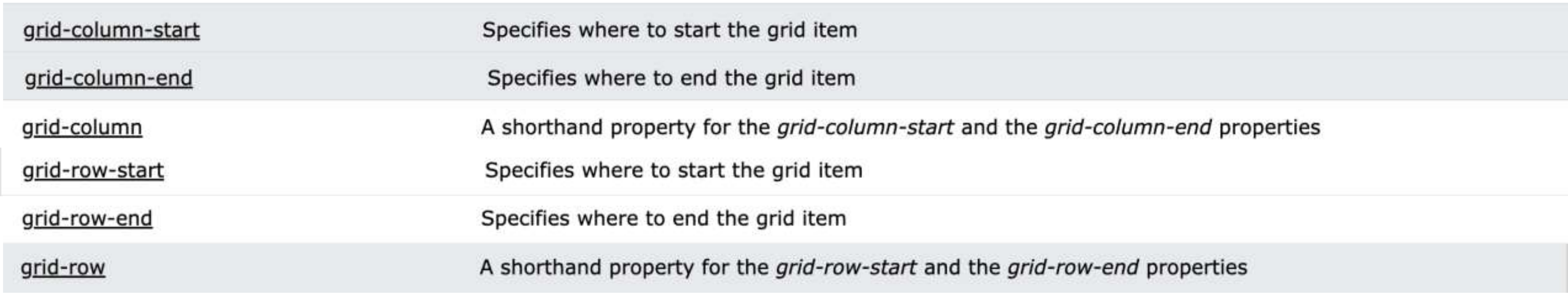
** You can use `fr(part)` for every column*

The `justify-content` property is used to align the whole grid inside the container.

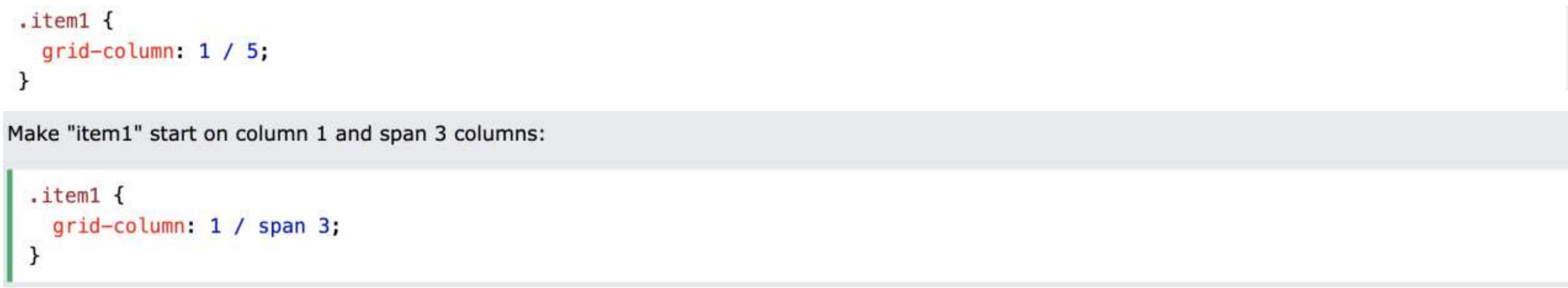
The `align-content` property is used to *vertically* align the whole grid inside the container.

The picture can't be displayed.

CSS Grid (Items)



grid-column-start	Specifies where to start the grid item
grid-column-end	Specifies where to end the grid item
grid-column	A shorthand property for the *grid-column-start* and the *grid-column-end* properties
grid-row-start	Specifies where to start the grid item
grid-row-end	Specifies where to end the grid item
grid-row	A shorthand property for the *grid-row-start* and the *grid-row-end* properties



```
.item1 {
  grid-column: 1 / 5;
}
```

Make "item1" start on column 1 and span 3 columns:

```
.item1 {
  grid-column: 1 / span 3;
}
```

The picture can't be displayed.

CSS Grid (Items)

The `grid-area` property can be used as a shorthand property for the `grid-row-start`, `grid-column-start`, `grid-row-end` and the `grid-column-end` properties.

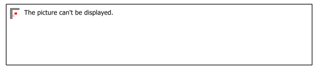
```
.item8 {
  grid-area: 1 / 2 / 5 / 6;
}
```

```
.item8 {
  grid-area: 2 / 1 / span 2 / span 3;
}
```

align-self, justify-self

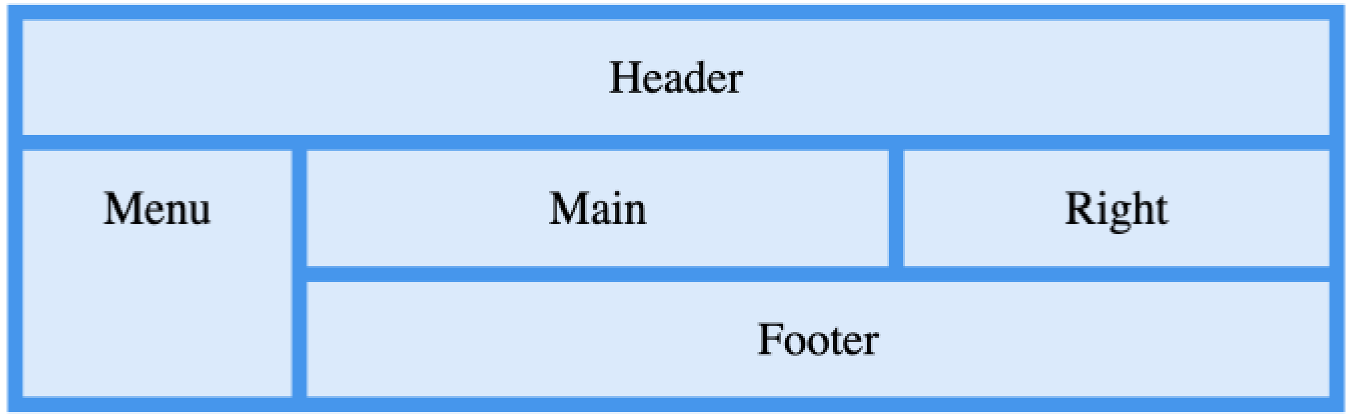
Naming Grid Items

The `grid-area` property can also be used to assign names to grid items.



The picture can't be displayed.

```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'menu footer footer footer footer footer';
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
</head>
<body>

<h1>Grid Layout</h1>

<p>This grid layout contains six columns and three rows:</p>

<div class="grid-container">
  <div class="item1">Header</div>
  <div class="item2">Menu</div>
  <div class="item3">Main</div>
  <div class="item4">Right</div>
  <div class="item5">Footer</div>
</div>
```



The picture can't be displayed.