Box Model, Flex Box & Grid Layout

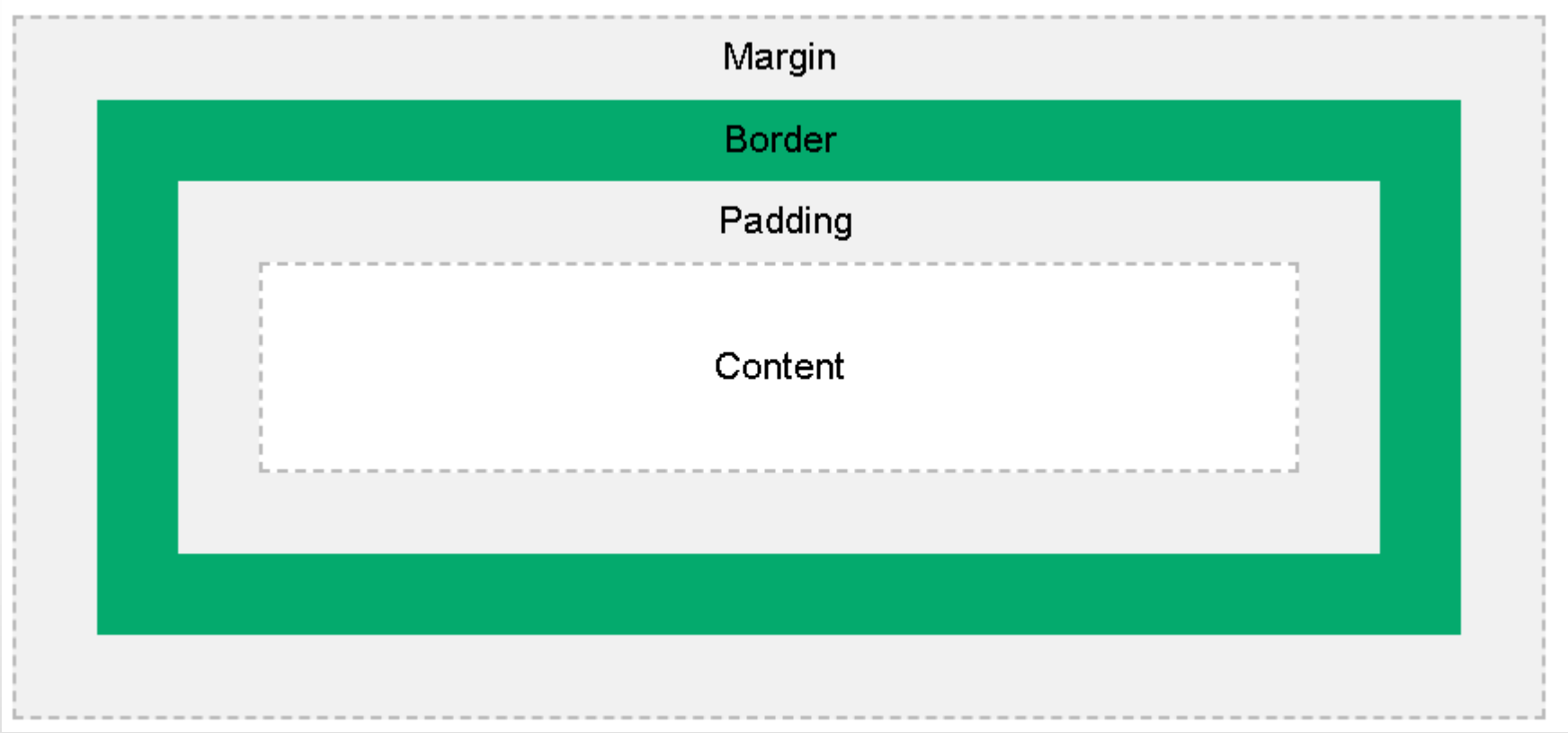
Compiled By
Sunil Bahadur Bist
Lecturer, NAST Dhangadhi

Box Model

- All HTML elements can be considered as boxes.
- In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element.
- It consists of:
  - content,
  - padding,
  - borders and
  - margins

```
div {
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```

Box Model



Box Model

- Explanation of the different parts:
  - Content - The content of the box, where text and images appear
  - Padding - Clears an area around the content. The padding is transparent
  - Border - A border that goes around the padding and content
  - Margin - Clears an area outside the border. The margin is transparent
- The box model allows us to add a border around elements, and to define space between elements.

Box Model

- This element will have a total width of 350px and a total height of 80px:

```
div {
  width: 320px;
  height: 50px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;
}
```

Box Model

- Here is the calculation:

  320px (width of content area)
  + 20px (left padding + right padding)
  + 10px (left border + right border)
  = 350px (total width)

  50px (height of content area)
  + 20px (top padding + bottom padding)
  + 10px (top border + bottom border)
  = 80px (total height)

Box Model

- **The total width of an element should be calculated like this:**

  Total element width = width + left padding + right padding
  + left border + right border

- **The total height of an element should be calculated like this:**

  Total element height = height + top padding + bottom padding
  + top border + bottom border

Flexbox

- Flexbox is short for the Flexible Box Layout module.
- Flexbox is a layout method for arranging items in rows or columns.
- Flexbox makes it easier to design a flexible responsive layout structure, without using float or positioning.

Flex Box Layout Module

- Before the Flexible Box Layout module, there were four layout modes:
  - Block, for sections in a webpage
  - Inline, for text
  - Table, for two-dimensional table data
  - Positioned, for explicit position of an element
- CSS flexbox is supported in all modern browsers.

Flexbox Components

- A flexbox always consists of:
  - a **Flex Container** - the parent (container) element
  - **Flex Items** - the items inside the container
- To start using CSS Flexbox, you need to first define a flex container.
- The flex container becomes flexible by setting the display property to flex.

CSS Flexbox Example

```
<style>
.flex-container {
  display: flex;
  background-color: #ccc;
}

.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
   color:red;
}
</style>
```

CSS Flexbox Example

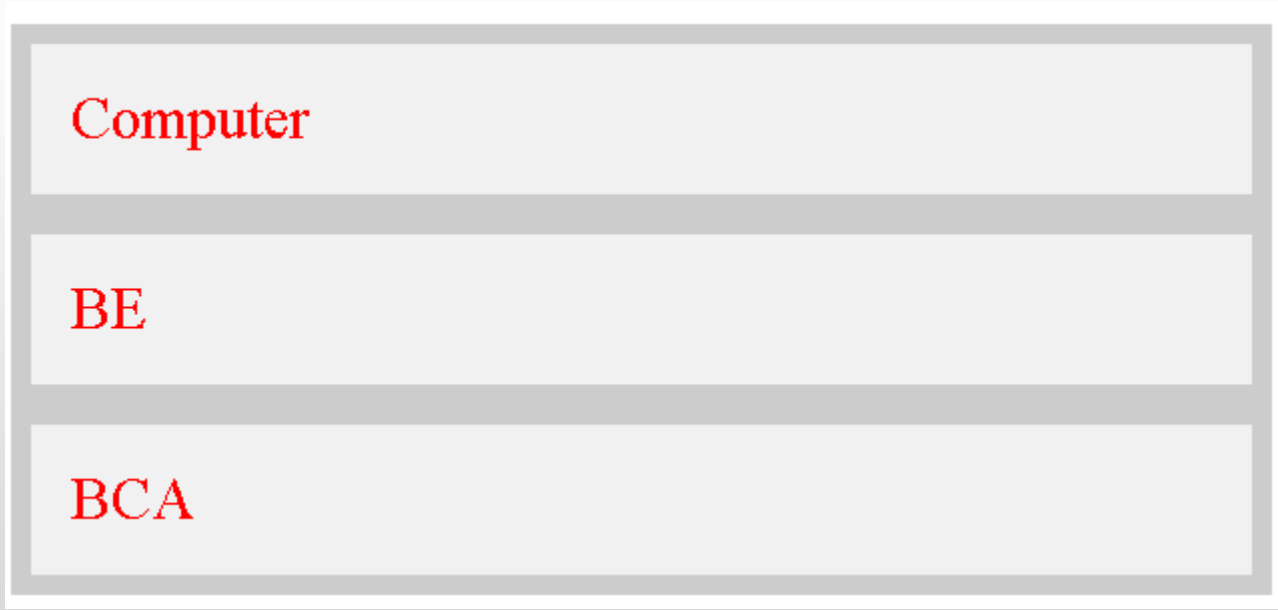
```
<body>
<div class="flex-container">
  <div>BCA</div>
  <div>BE</div>
  <div>Computer</div>
</div>
</body>
```

Flex container properties

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

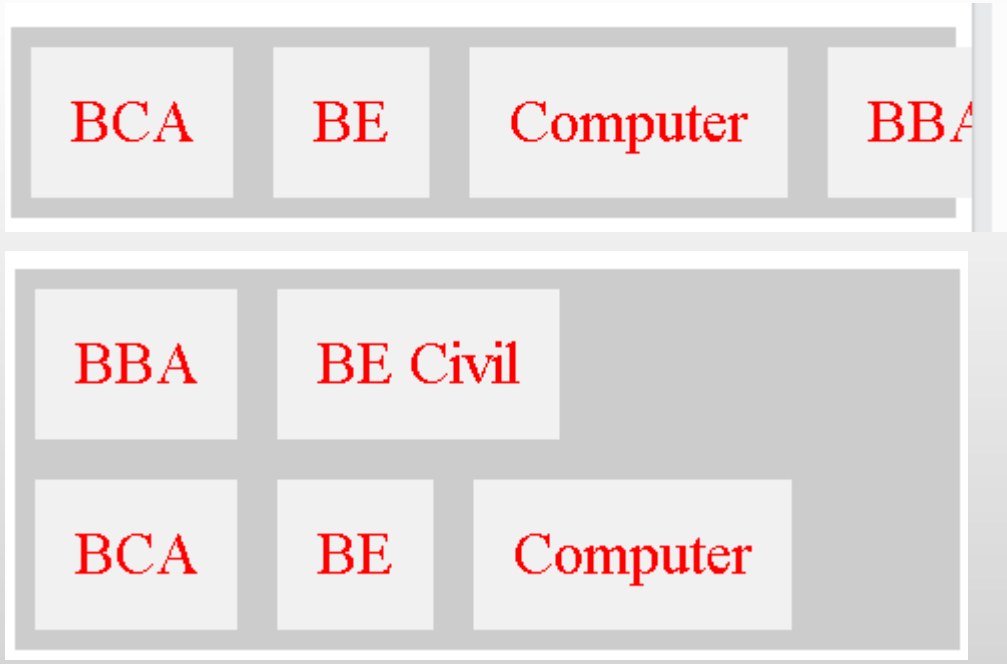
Flex container properties

- The flex-direction property specifies the display-direction of flex items in the flex container.
- The flex-direction property can have one of the following values:
- flex-direction
  - row
  - column
  - row-reverse
  - column-reverse



Flex container properties

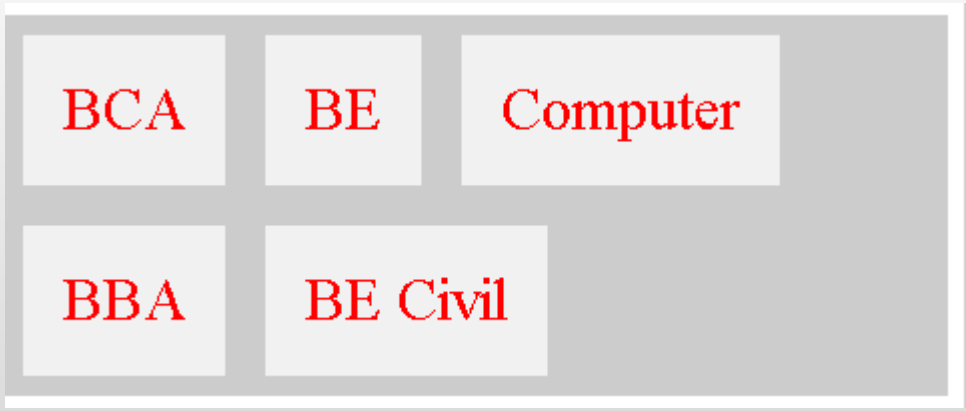
- The flex-wrap property specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line.
- The flex-wrap property can have one of the following values:
- flex-wrap
  - nowrap
  - wrap
  - wrap-reverse



Flex container properties

- The flex-flow property is a shorthand property for setting both the flex-direction and flex-wrap properties.
- flex-flow

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



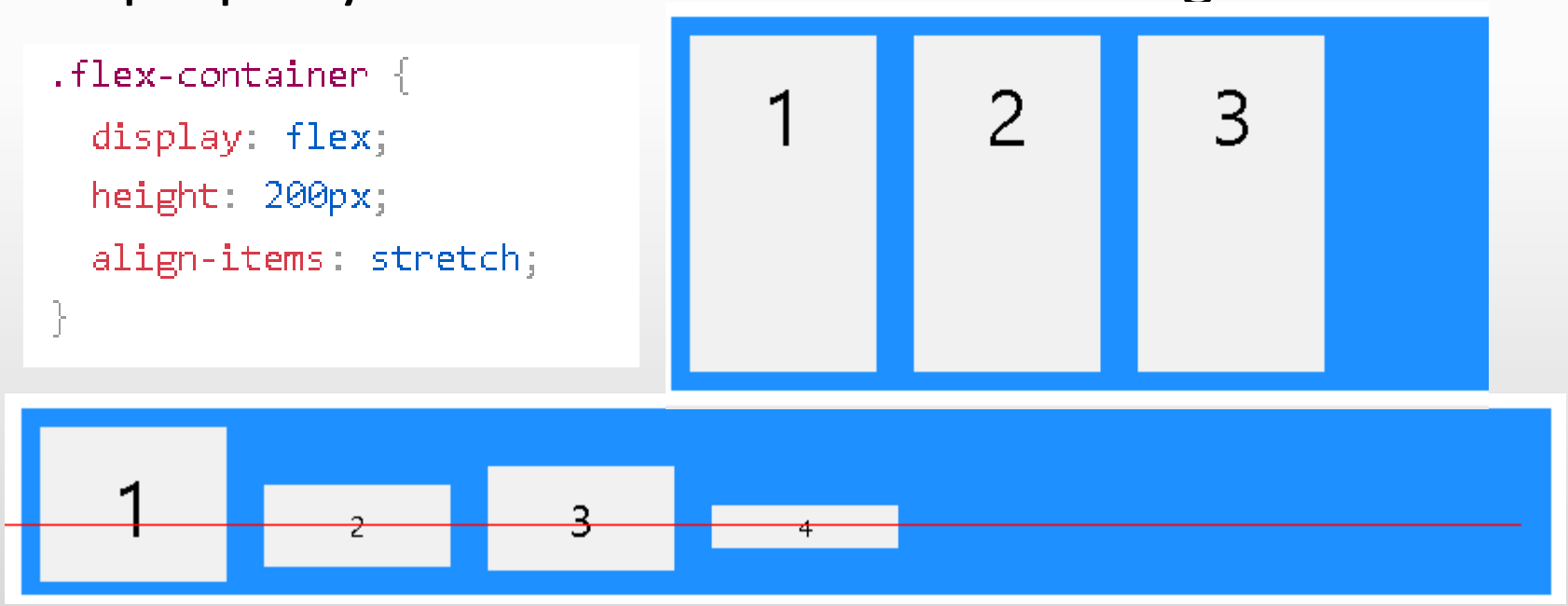
Flex container properties

- The justify-content property is used to align the flex items when they do not use all available space on the main-axis (horizontally).
- The justify-content property can have one of the following values:
- justify-content
  - center
  - flex-start
  - flex-end
  - space-around
  - space-between
  - space-evenly

Flex container properties

- The align-items property is used to align the flex items when they do not use all available space on the cross-axis (vertically).
- The align-items property can have one of the following values:
- align-items
  - center
  - flex-start
  - flex-end
  - stretch
  - baseline
  - normal

```
.flex-container {
  display: flex;
  height: 200px;
  align-items: stretch;
}
```

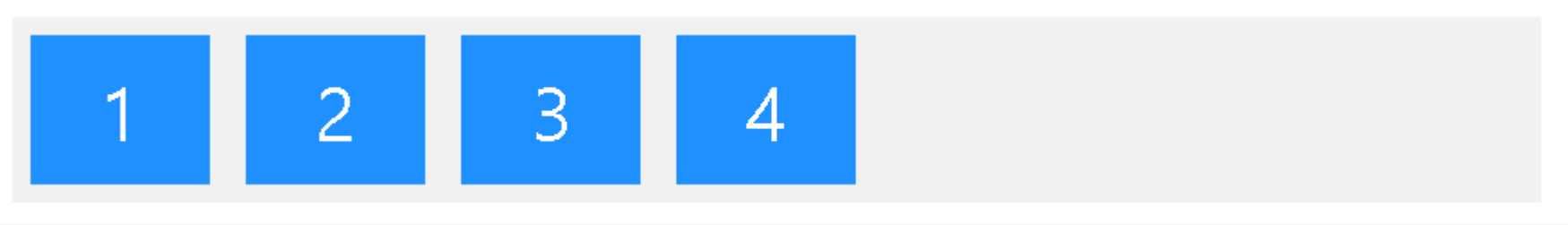


Flex container properties

- The align-content property is used to align the flex lines.
- The align-content property is similar to align-items, but instead of aligning flex items, it aligns the flex lines.
- The align-content property can have one of the following values:
- justify-content
  - center
  - flex-start
  - flex-end
  - space-around
  - space-between
  - space-evenly

Flex Items

- The direct child elements of a flex container automatically becomes flex items.



- The element above represents four blue flex items inside a grey flex container.

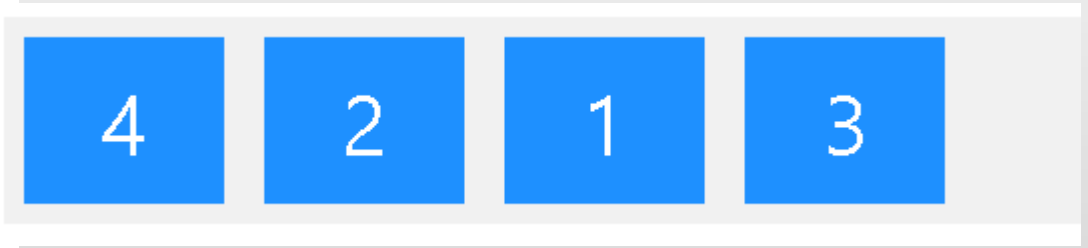
Flex Item properties

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

Flex Item properties

- The **order** property specifies the order of the flex items inside the flex container.
- The first flex item in the code does not have to appear as the first item in the layout.
- The order value must be a number, default value is 0.

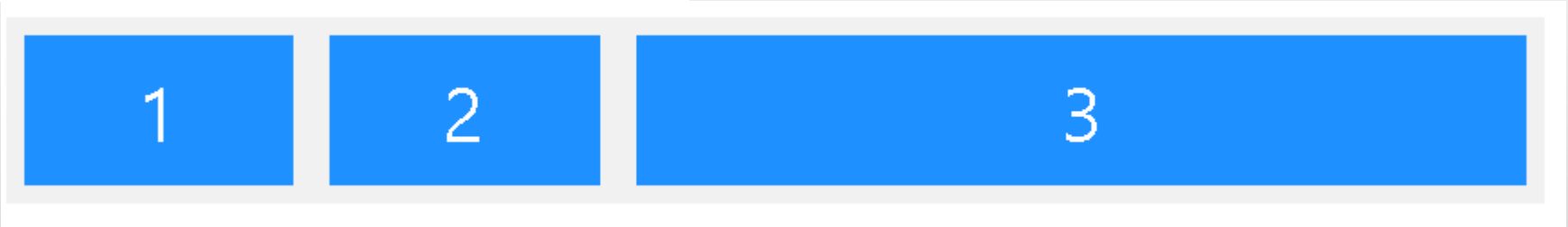
```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```



Flex Item properties

- 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>
```



Flex Item properties

- The flex-shrink property specifies how much a flex item will shrink relative to the rest of the flex items.
- The value must be a number, default value is 1.
- The flex-basis property specifies the initial length of a flex item.

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-basis: 200px">3</div>
  <div>4</div>
</div>
```

Flex Item properties

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

Grid Layout

- The Grid Layout Module offers a grid-based layout system, with rows and columns.
- The Grid Layout Module allows developers to easily create complex web layouts.
- The Grid Layout Module makes it easier to design a responsive layout structure, without using float or positioning.
- The CSS grid properties are supported in all modern browsers.

Grid Components

- A grid always consists of:
  - Grid Container - the parent (container) element
  - Grid Items - the items inside the container
- A grid layout consists of a parent element (the grid container), with one or more grid items.
- All direct children of the grid container automatically become grid items.

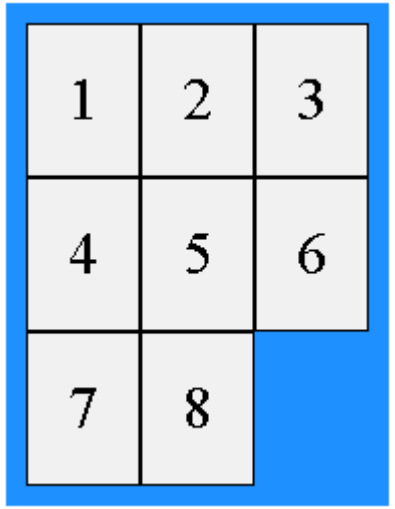
Grid Example

- The element becomes a grid container when its display property is set to grid or inline-grid.

```
<style>
.container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: dodgerblue;
  padding: 10px;
}
.container > div {
  background-color: #f1f1f1;
  border: 1px solid black;
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
```

Grid Example

1	2	3
4	5	6
7	8	



Grid Properties

- The vertical lines of grid items are called *columns*.
- The horizontal lines of grid items are called *rows*.
- The spaces between each column/row are called *gaps*.
- You can adjust the gap size by using one of the following properties:
  - column-gap
  - row-gap
  - gap
- The column-gap property specifies the gap between the columns in a grid.
- The row-gap property specifies the gap between the rows in a grid.

Grid Properties

- The gap property is a shorthand property for row-gap and column-gap
- The lines between columns are called column lines.
- The lines between rows are called row lines.
- We can specify where to start and end a grid item by using the following properties:
  - grid-column-start
  - grid-column-end
  - grid-row-start
  - grid-row-end
  - grid-column
  - grid-row
- You can refer to line numbers when placing a grid item in a grid container.

Grid Example

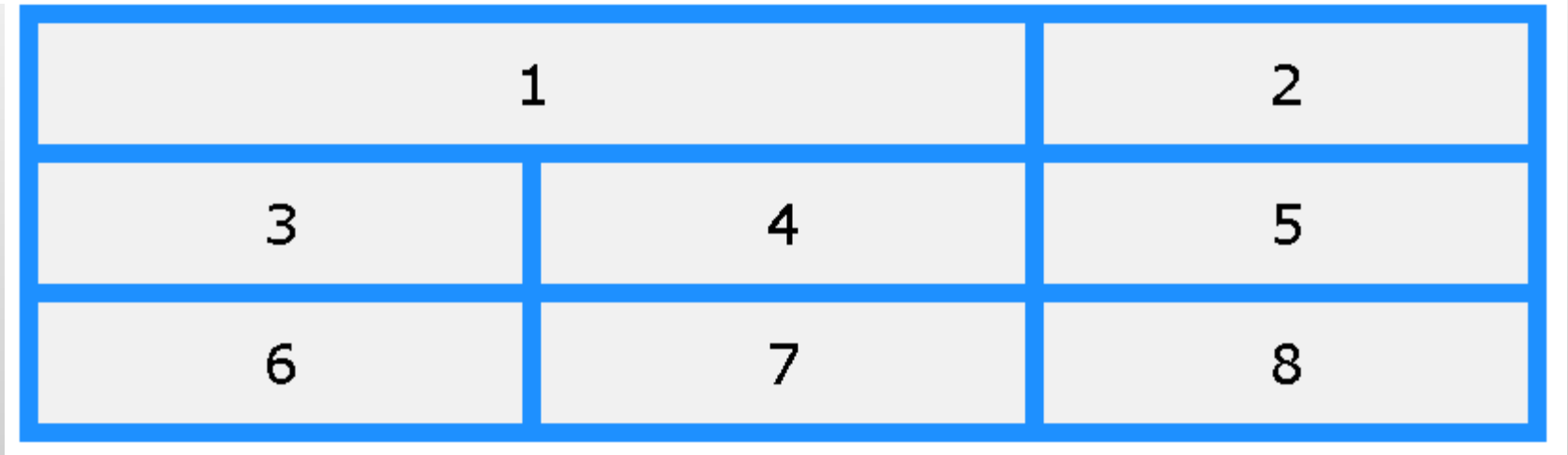
```
.grid-container {
  display: grid;
  grid-template-columns: 80px 200px auto 40px;
}
```

1	2	3	4
5	6	7	8

Grid Items

- A grid container contains one or more grid items.
- By default, a container has one grid item for each column, in each row, but you can style the grid items so that they will span multiple columns and/or rows.

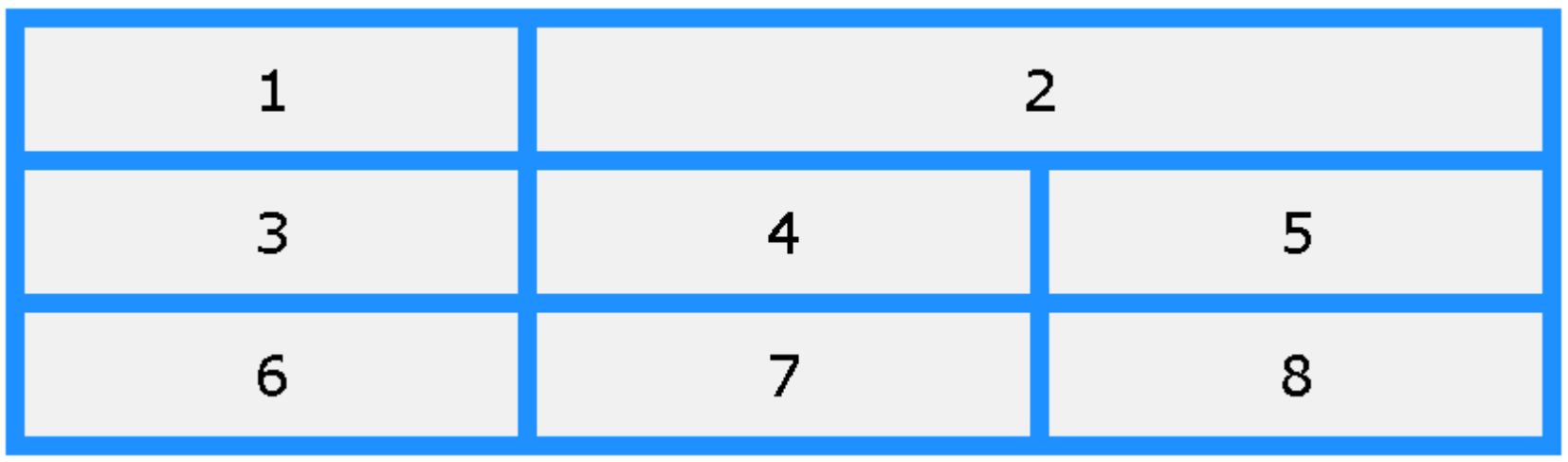
```
.item1 {
  grid-column: 1 / span 2;
}
```



Grid Items

- Make "item2" start on column 2 and span 2 columns

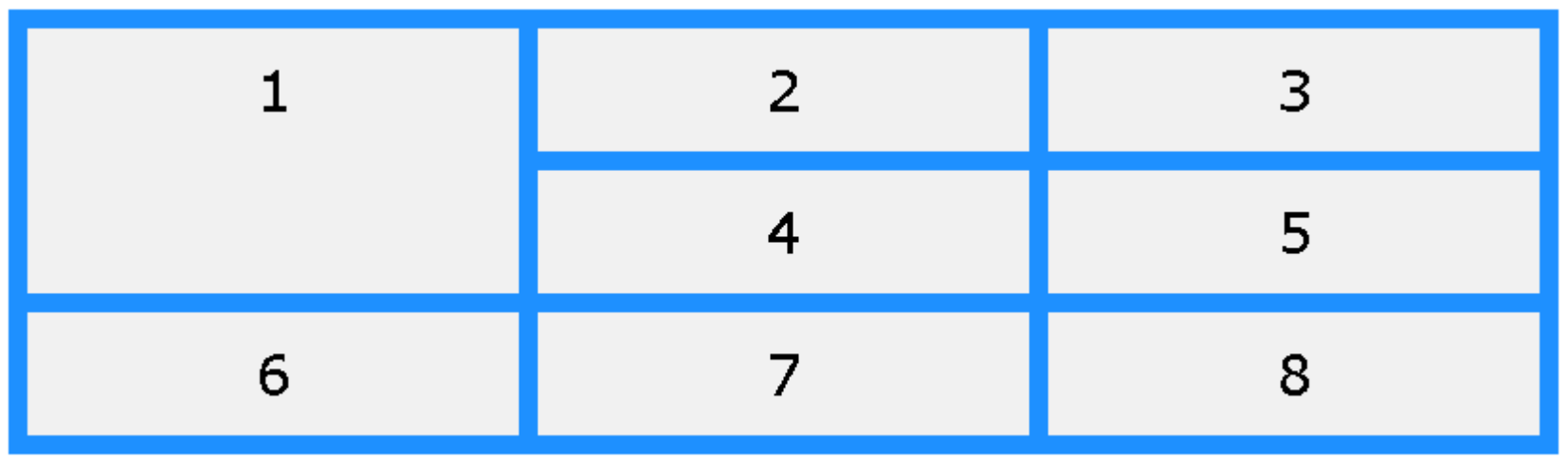
```
.item2 {
  grid-column: 2 / span 2;
}
```



Grid Items

- Place the first grid item at row line 1, and let it end on row line 3

```
.item1 {
  grid-row-start: 1;
  grid-row-end: 3;
}
```

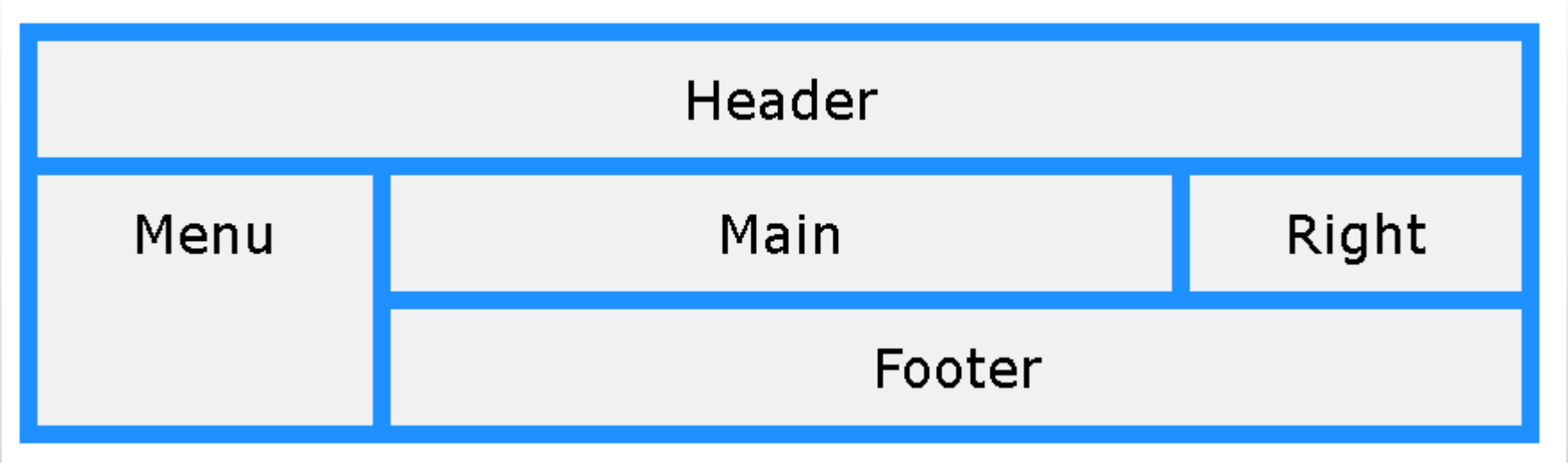


Example

```
<style>
.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu main main main main right'
    'menu footer footer footer footer footer';
  gap: 10px;
  background-color: dodgerblue;
  padding: 10px;
}
.grid-container > div {
 background-color: #f1f1f1;
color:#000;
  padding: 10px;
  font-size: 30px;
  text-align: center;
}
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }
</style>
```

Example

```
<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>
```

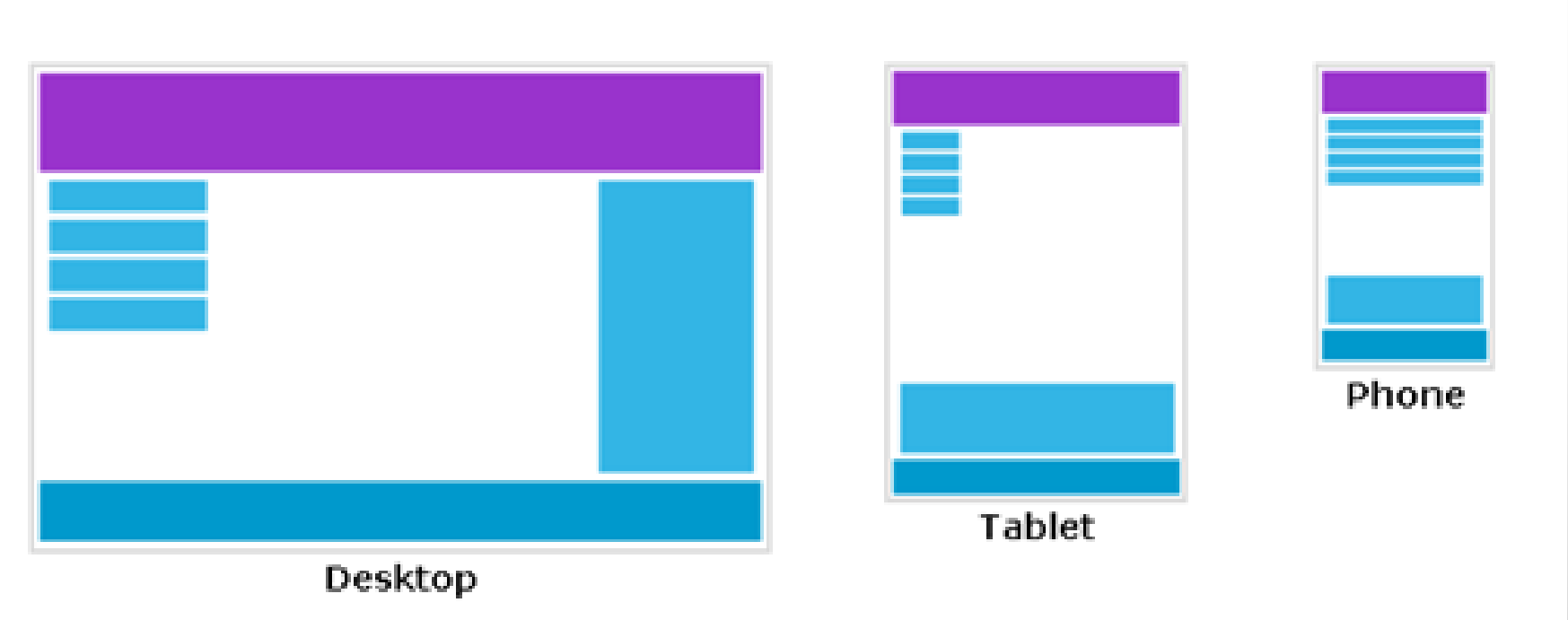


Responsive Design

- Responsive web design makes your web page look good on all devices.
- Responsive web design uses only HTML and CSS.
- Responsive web design is not a program or a JavaScript.
- Web pages can be viewed using many different devices: desktops, tablets, and phones.
- Your web page should look good, and be easy to use, regardless of the device.

Responsive Design

- Web pages should not leave out information to fit smaller devices, but rather adapt its content to fit any device:



Responsive Design Principals

- Responsive design ensures websites are accessible and enjoyable across various devices and screen sizes.
- Key principles include using fluid grids and images, utilizing media queries to adjust layouts based on device characteristics, and focusing on mobile-first design, ensuring a seamless user experience.
- The Interaction Design Foundation notes that these principles aim to create a consistent and user-friendly experience on all devices.

Responsive Design Principals

- Here's a more detailed look at the core principles:

1. Fluid Grids
2. Fluid Images
3. Media Queries
4. Mobile-First Approach
5. Breakpoints
6. Viewport Configuration
7. Testing on Real Devices