HTML/CSS (Advanced)

Introduction to Internet and Web

부산대학교 정보·의생명 공학대학
정보컴퓨터공학부

Table of Contents

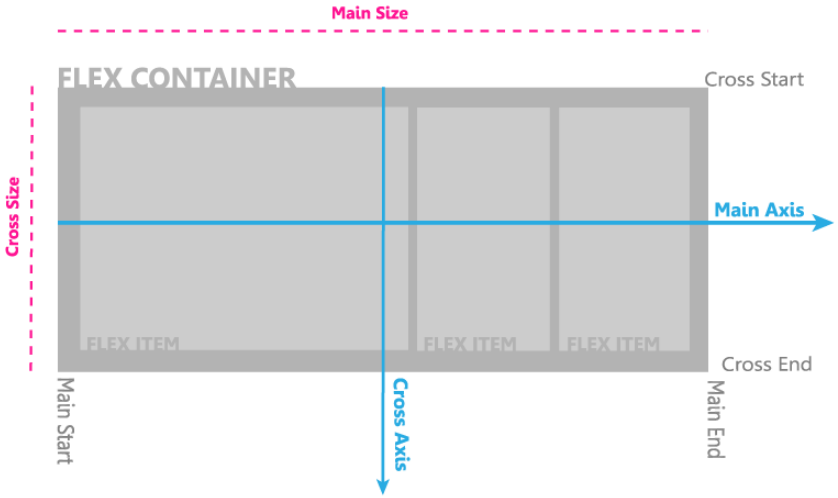
FLEXBOX

Flexbox

❖ A one-dimensional layout model.

- main-axis is defined by flex-direction (row/row-reverse/column/column-reverse)
- cross axis runs perpendicular to the main axis.
  - if your flex-direction (main axis) is set to row or row-reverse the cross axis runs down the columns

❖ A method that could offer space distribution between items in an interface and powerful alignment capabilities.



https://tympanus.net/codrops/css_reference/flexbox/

FLOATS

(MAGAZINE-STYLE LAYOUTS)

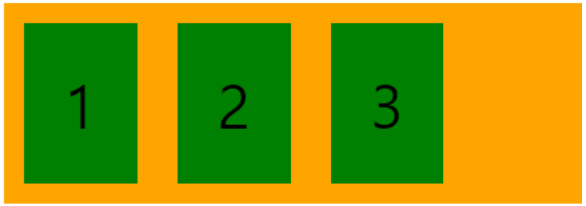
FLEXBOX

(OVERALL PAGE STRUCTURE)

https://www.kindpng.com/imgv/hxJiioi_css-floats-for-text-wrapping-around-a-box/

Flexbox

- To start using the Flexbox model, you need to first define a flex container.
- As soon as we do this the direct children of that container become flex items.
- Default values
  - Items display in a row.
  - The items start from the start edge of the main axis.
  - The items do not stretch on the main dimension, but can shrink.
  - The items will stretch to fill the size of the cross axis.
  - The flex-basis property is set to auto.
  - The flex-wrap property is set to nowrap.

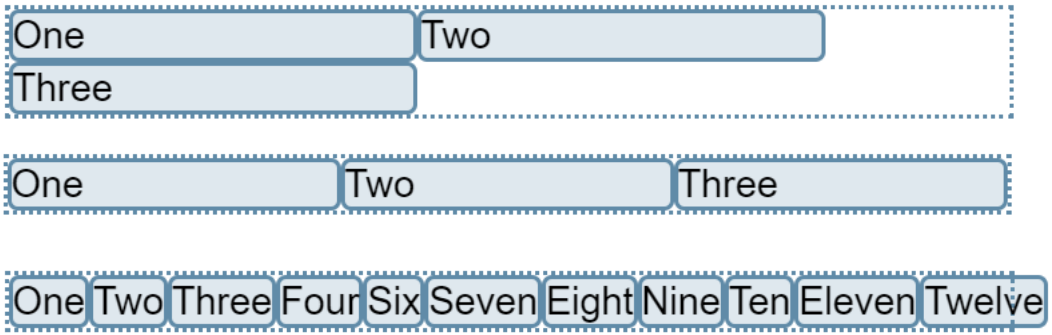



```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  background-color: orange;
}
.flex-container > div {
  background-color: green;
  margin: 10px; padding: 20px;
  font-size: 30px;
}
</style>
</head>
<body>
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
</body>
</html>
```

Flex Container

- **The flex-wrap property specifies whether the flex items should wrap or not.**
  - wrap / nowrap (initial value)
- **While flexbox is a one dimensional model, it is possible to cause our flex items to wrap onto multiple lines.**
  - wrap: if items be too large to all display in one line, they will wrap onto another line.
  - nowrap: they will instead shrink to fit the container. Using nowrap would cause an overflow if the items were not able to shrink, or could not shrink small enough to fit.
- **example**

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



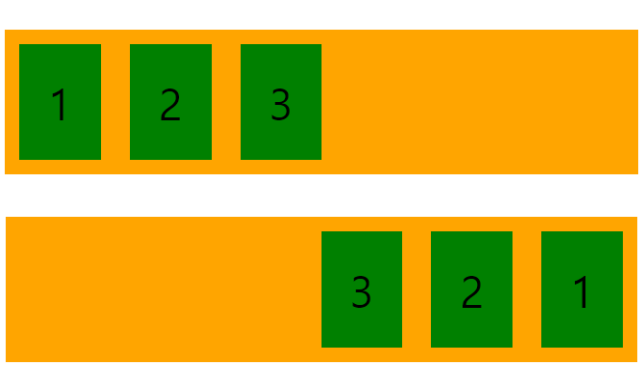
Flex Container

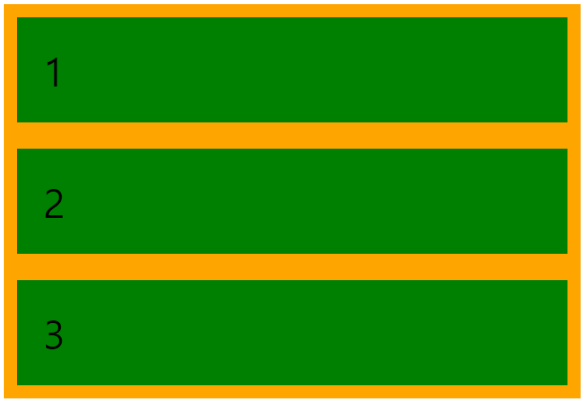
❖ The flex-direction property defines in which direction the container wants to stack the flex items.

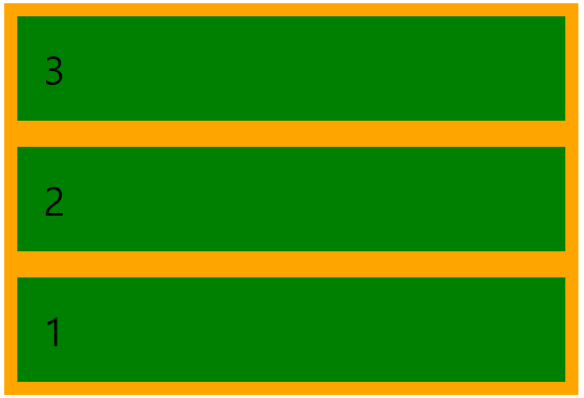
```
.flex-container {
  display: flex;
  flex-direction: row;
  background-color: orange;
}
```

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

- The first value specified is flex-direction and the second value is flex-wrap.
- e.g., flex-flow: row wrap;



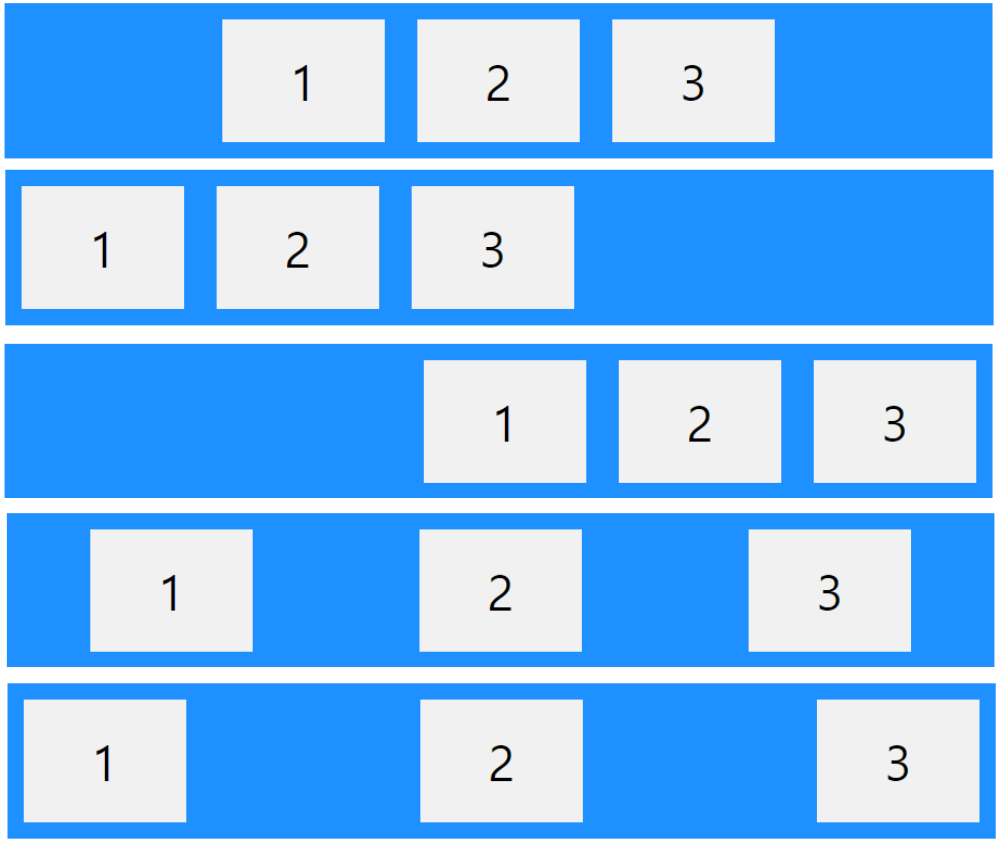




Flex Container

❖ **The justify-content property is used to align the flex items**

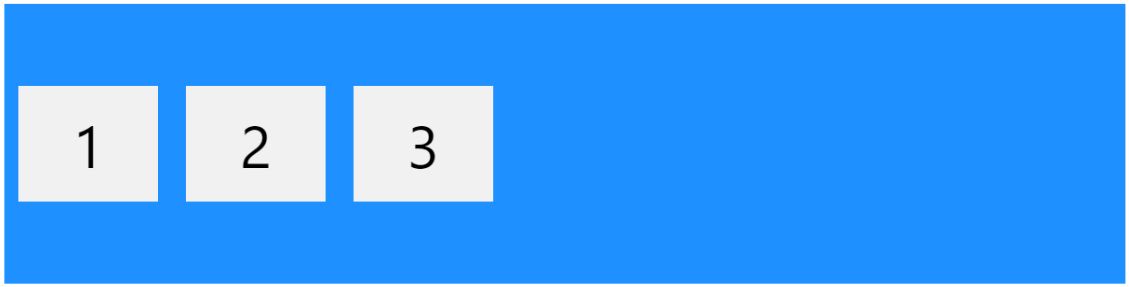
- center, flex-start(default), flex-end, space-around, space-between



Flex Container

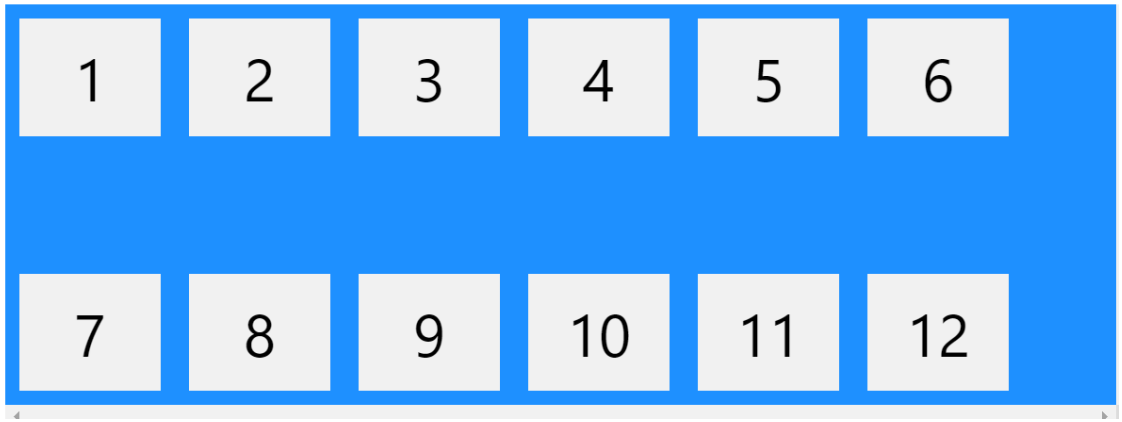
❖ The align-items property is used to align the flex items.

- center, flex-start, flex-end, stretch(default), baseline



❖ The align-content property is used to align the flex lines.

- space-between, space-around, stretch (default)



Flex Items

❖ Available space

- flex-basis
- flex-grow
- flex-shrink

❖ The flex-basis is what defines the size of that item in terms of the space it leaves as available space.

- initial value: auto
  - the browser looks to see if the items have a size
  - If the items don't have a size then the content's size is used as the flex-basis

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

Flex Items

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

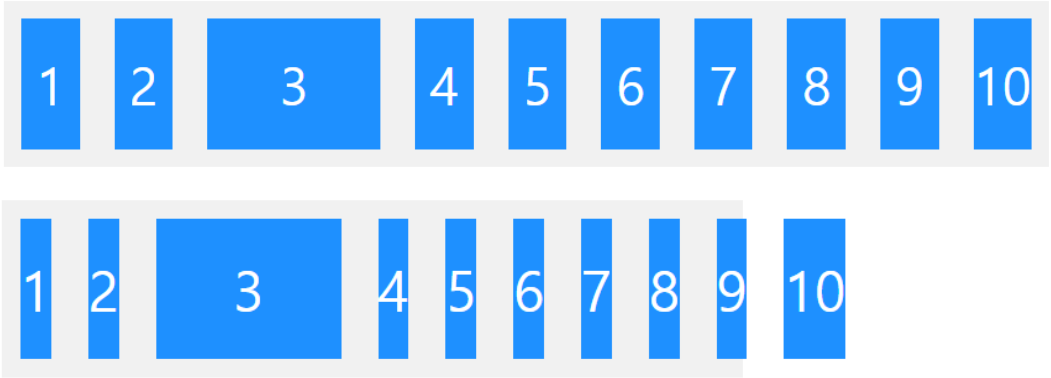
- initial value : 0

```
<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.

- initial value: 1

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-shrink: 0">3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</div>
```



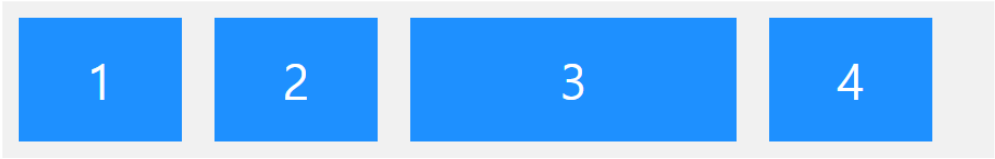
Flex Items

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

- flex: initial = flex 0 1 auto
  - Items will not grow larger than their flex-basis size
  - Items can shrink if they need to rather than overflowing
  - Items will either use any size set on the item in the main dimension, or they will get their size from the content size
- flex: auto = flex: 1 1 auto
- flex: none = flex: 0 0 auto
- flex: N = flex: N 1 0

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex: 0 0 200px">3</div>
  <div>4</div>
</div>
```

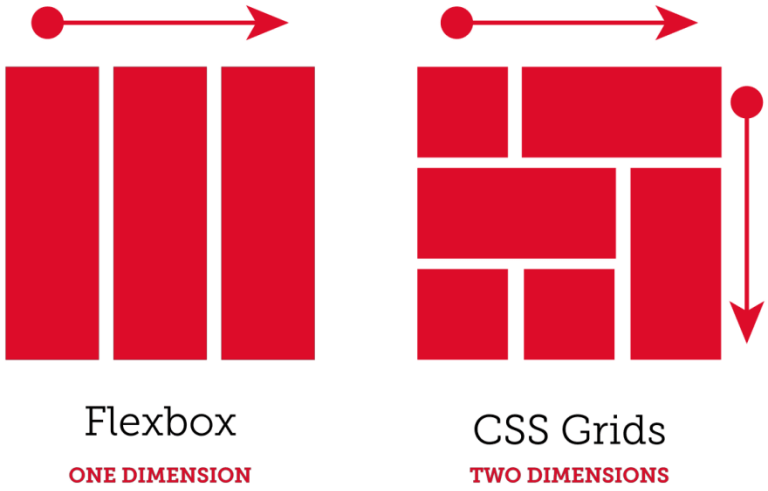
Make the third flex item not growable (0), not shrinkable (0), and with an initial length of 200 pixels:



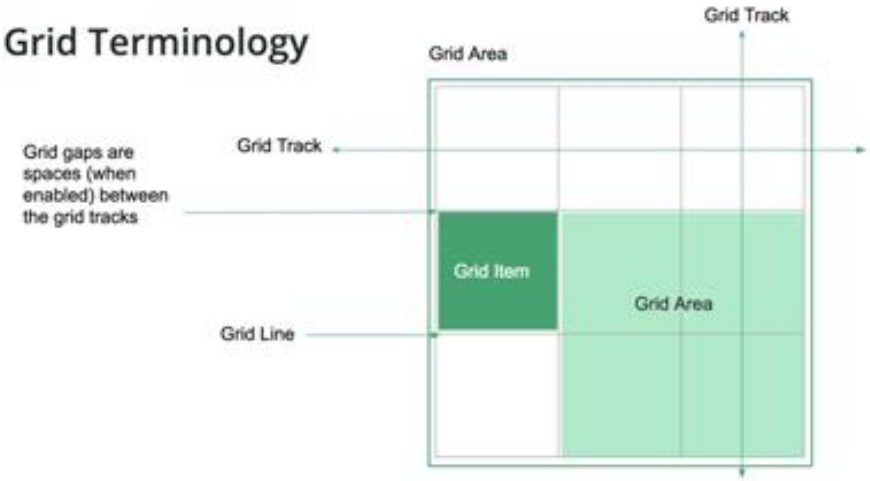
GRID LAYOUT

Grid Layout

- ❖ A two-dimensional grid system
- ❖ Terminology
  - Grid lines
  - Grid cell (Grid Item)
  - Grid Area
  - Grid Track: Grid Row, Grid Column
  - Grid Gutter (Grid Gap)



https://velog.io/@denmark-choco/CssFlex-Grid



Grid Container

❖ **Grid containers consist of grid items, placed inside columns and rows.**

1	2	3
4	5	6
7	8	9

❖ **grid-template-columns Property**

- defines the number of columns in your grid layout, and it can define the width of each column.
- The new fr unit represents a fraction of the available space in the grid container.
- e.g., grid-template-columns: 1fr 1fr 2fr;

1	2	3
4	5	6
7	8	9

```
<!DOCTYPE html>
<html>
<head>
<style>
.grid-container {
  display: grid; padding: 10px;
  grid-template-columns: auto
auto auto;
  background-color: #2196F3;
}
.grid-item {  ... }
</style>
</head>
<body>
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
    ...
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
</body>
</html>
```

Grid Container

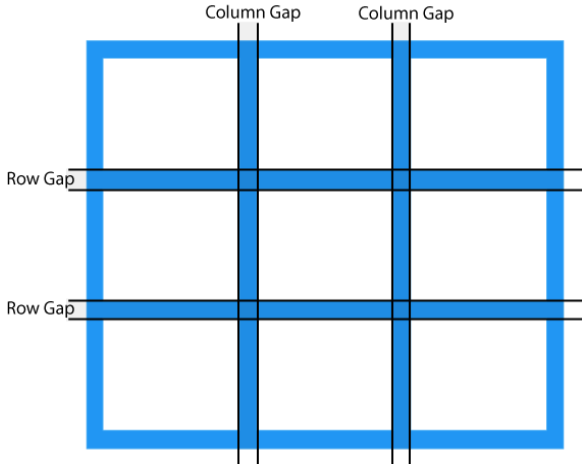
❖ The grid-template-rows property defines the height of each row.

```
.grid-container {
  display: grid;
  grid-template-rows: 80px 200px;
}
```

1	2	3	4
5	6	7	8

❖ The gap property is a shorthand property for the row-gap and the column-gap properties:

```
.grid-container {
  display: grid;
  gap: 50px 100px;
}
```



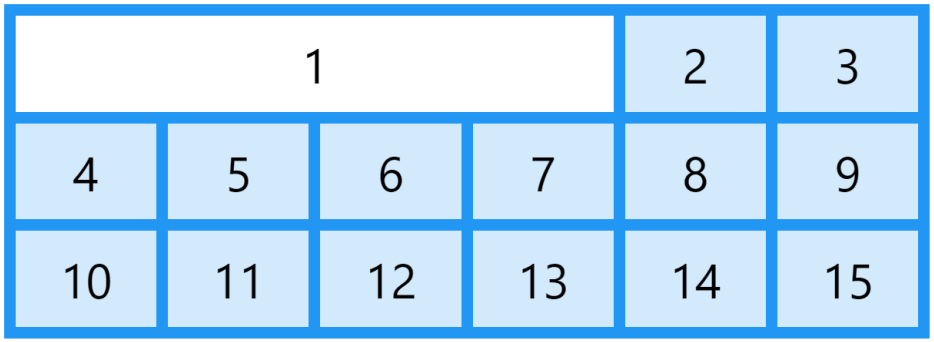
Grid Item

❖ **The grid-column property defines on which column(s) to place an item.**

- A shorthand property for the grid-column-start and the grid-column-end properties.
- To place an item, you can refer to line numbers, or use the keyword "span" to define how many columns the item will span.

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

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



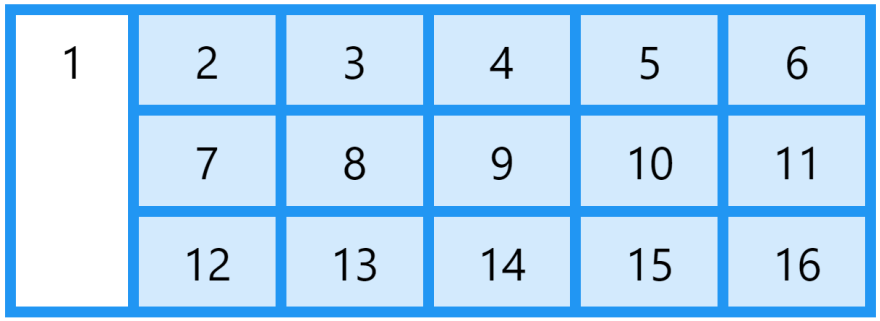

```
.grid-container {
    display: grid;
    grid-template-columns: auto
auto auto auto auto auto;
    gap: 10px;
    background-color: #2196F3;
    padding: 10px;
  }
...

<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
    ...
```

Grid Item

❖ The grid-row property defines on which row to place an item.

- The grid-row property is a shorthand property for the grid-row-start and the grid-row-end properties.
- To place an item, you can refer to line numbers, or use the keyword "span" to define how many rows the item will span.

```
.item1 {
  grid-row: 1 / span 3;
}
.item1 {
  grid-row: 1 / 4;
}
```

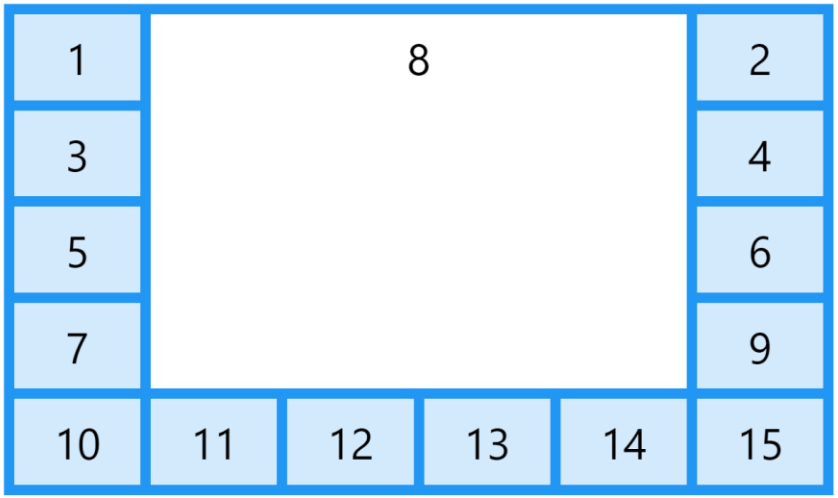



```
.grid-container {
  display: grid;
  grid-template-columns: auto
auto auto auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
...
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
    ...
```

Grid Item

❖ 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;
}
```

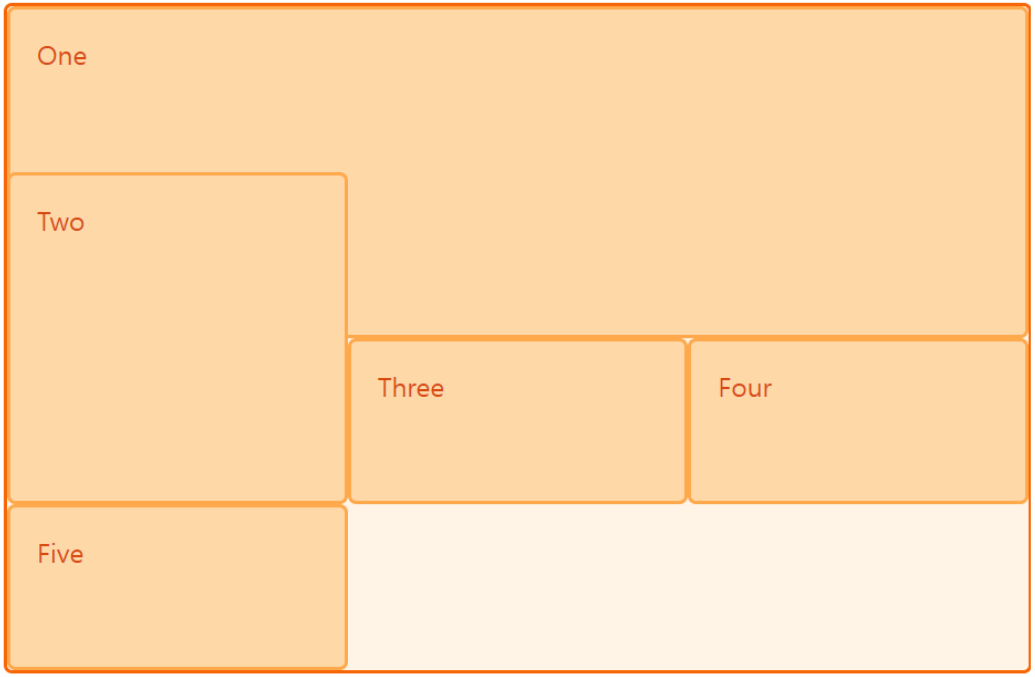


Grid Item

❖ Layering Items with z-index

```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-auto-rows: 100px;
}

.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}

.box2 {
  grid-column-start: 1;
  grid-row-start: 2;
  grid-row-end: 4;
}
```

```
<div class="wrapper">
  <div class="box box1">One</div>
  <div class="box box2">Two</div>
  <div class="box box3">Three</div>
  <div class="box box4">Four</div>
  <div class="box box5">Five</div>
</div>
```

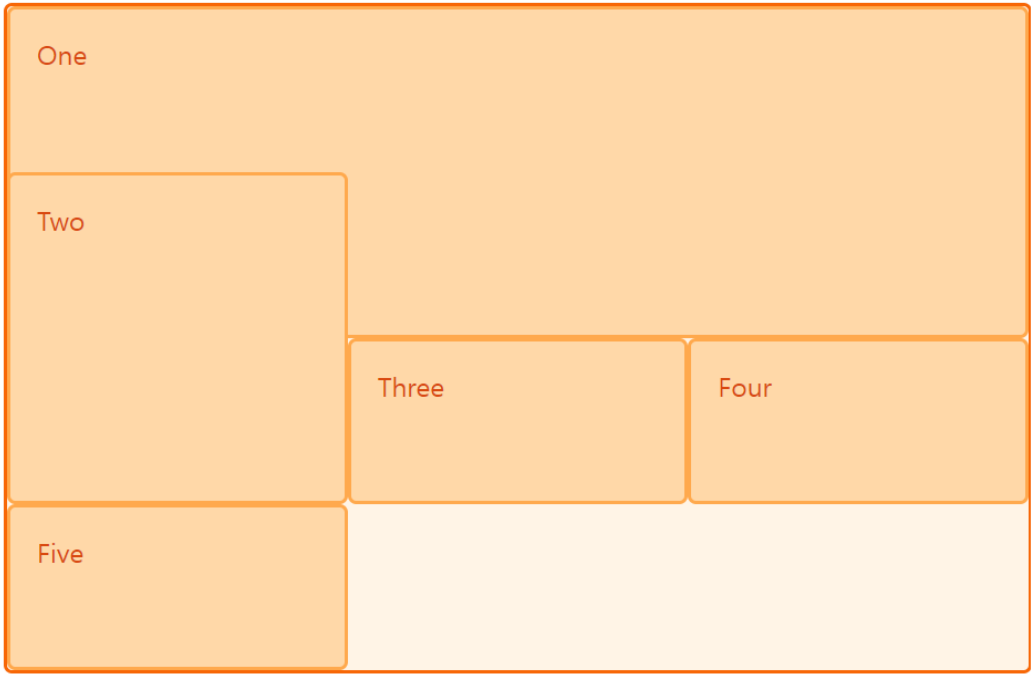


Grid Item

❖ Layering Items with z-index

```
.wrapper {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-auto-rows: 100px;
  }

  .box1 {
    grid-column-start: 1;
    grid-column-end: 4;
    grid-row-start: 1;
    grid-row-end: 3;
    z-index: 2;
  }

  .box2 {
    grid-column-start: 1;
    grid-row-start: 2;
    grid-row-end: 4;
    z-index: 1;
  }
```

```
<div class="wrapper">
  <div class="box box1">One</div>
  <div class="box box2">Two</div>
  <div class="box box3">Three</div>
  <div class="box box4">Four</div>
  <div class="box box5">Five</div>
</div>
```

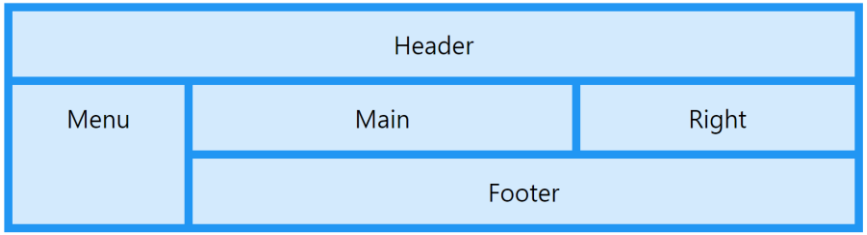


Grid Item

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

- Named grid items can be referred to by the grid-template-areas property of the grid container.
  - Each row is defined by apostrophes (' ')
  - The columns in each row is defined inside the apostrophes, separated by a space.

```
<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>
```

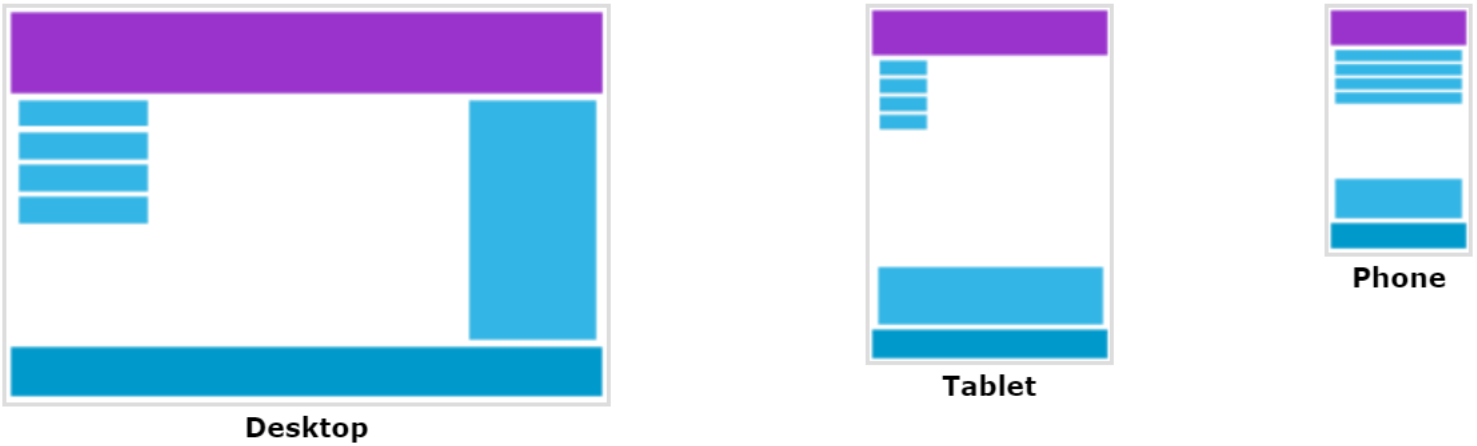



```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.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;
}
...
```

RESPONSIVE WEB DESIGN

Responsive Web Design

- ❖ **Responsive web design makes your web page look good on all devices.**
- ❖ **responsive web design isn't a separate technology**
  - it is a term used to describe an approach to web design or a set of best practices, used to create a layout that can respond to the device being used to view the content



Viewport

- The viewport is the user's visible area of a web page.
- The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.
- This gives the browser instructions on how to control the page's dimensions and scaling

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

  - The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
  - The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

Without the viewport meta tag

With the viewport meta tag

Media Queries

- Media query is a CSS technique introduced in CSS3.
- It uses the @media rule to include a block of CSS properties only if a certain condition is true.

```
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

Media Queries

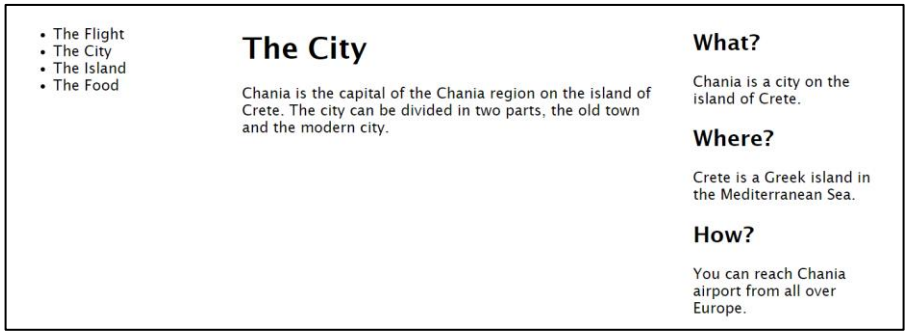
```
/* For desktop: */
  [class*="col-"] {
    float: left;
    padding: 15px;
  }
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

@media only screen and (max-width: 768px) {
  /* For mobile phones: */
  [class*="col-"] {
    width: 100%;
  }
}
```

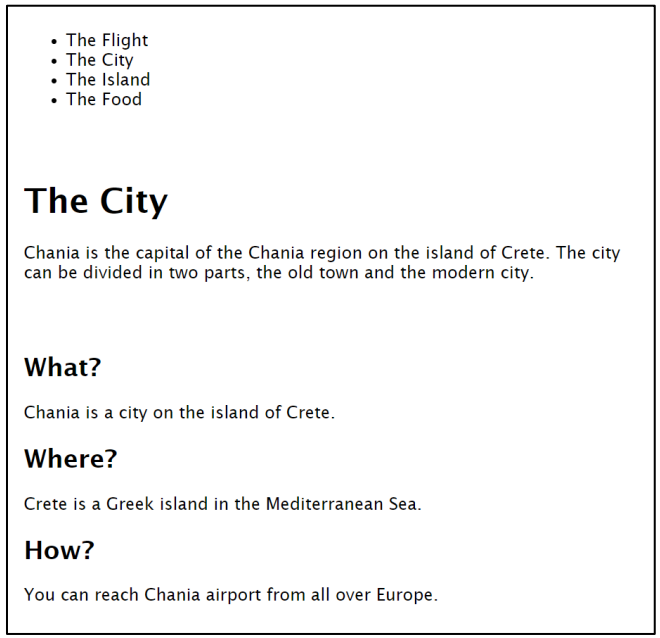
```
  <div class="row">
  <div class="col-3">
    <ul>
    <li>The Flight</li>
    <li>The City</li>
    <li>The Island</li>
    <li>The Food</li>
    </ul>
  </div>

  <div class="col-6">
    <h1>The City</h1>
    <p>Chania is the capital of the Chania region on
the island of Crete. The city can be divided in two
parts, the old town and the modern city.</p>
  </div>

  <div class="col-3">
      <h2>What?</h2>
      <p>Chania is a city on the island of Crete.</p>
      <h2>Where?</h2>
      <p>Crete is a Greek island in the Mediterranean
Sea.</p>
      <h2>How?</h2>
      <p>You can reach Chania airport from all over
Europe.</p>
  </div>
</div>
```

https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_breakpoints

Media Queries

```
/* For desktop: */
  [class*="col-"] {
    float: left;
    padding: 15px;
  }
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

@media only screen and (max-width: 768px) {
  /* For mobile phones: */
  [class*="col-"] {
    width: 100%;
  }
}
```





https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_breakpoints

Media Queries

- There are tons of screens and devices with different heights and widths, so it is hard to create an exact breakpoint for each device.
- To keep things simple you could target five groups:

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px
and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px
and up) */
@media only screen and (min-width: 1200px) {...}
```

Images and videos

❖ If the width property is set to a percentage and the height property is set to "auto", the image will be responsive and scale up and down:

```
img {
  width: 100%;
  height: auto;
}
```

```
video {
  width: 100%;
  height: auto;
}
```

❖ If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size

```
img {
    max-width: 100%;
    height: auto;
  }
...
<img src="img_chania.jpg" width="460" height="345">
```

Resize the browser window to see how the image will scale when the width is less than 460px.

요 약

- FLEX Box
- Grid Layout
- Responsible Web Design

- [참조]
  - https://www.w3schools.com/css/css3_mediaqueries_ex.asp
  - https://www.w3schools.com/css/css_rwd_frameworks.asp