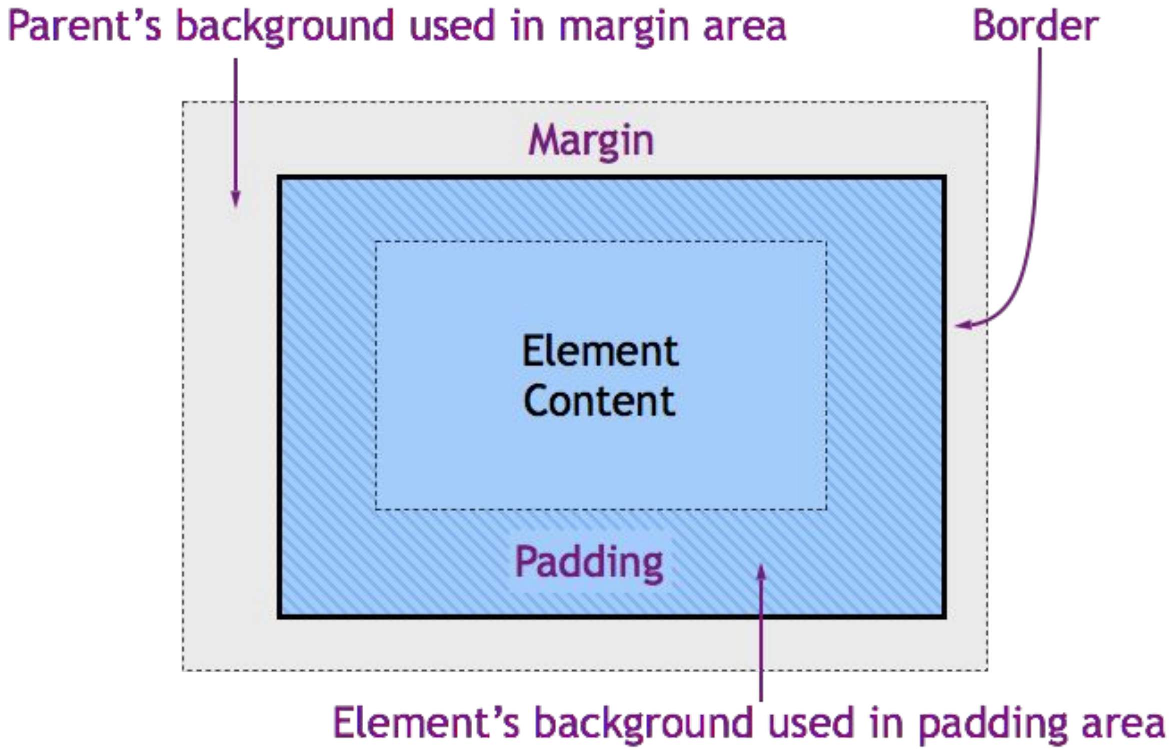
More CSS

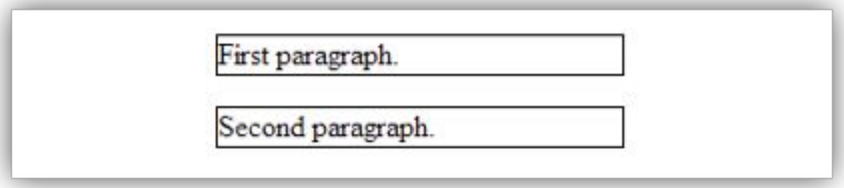
COEN 161

CSS Box Model


Parent's background used in margin area
Border
Margin
Element Content
Padding
Element's background used in padding area

How to: Centering

```
p {
    margin-left: auto;
    margin-right: auto;
    width: 750px;
}
```


First paragraph.

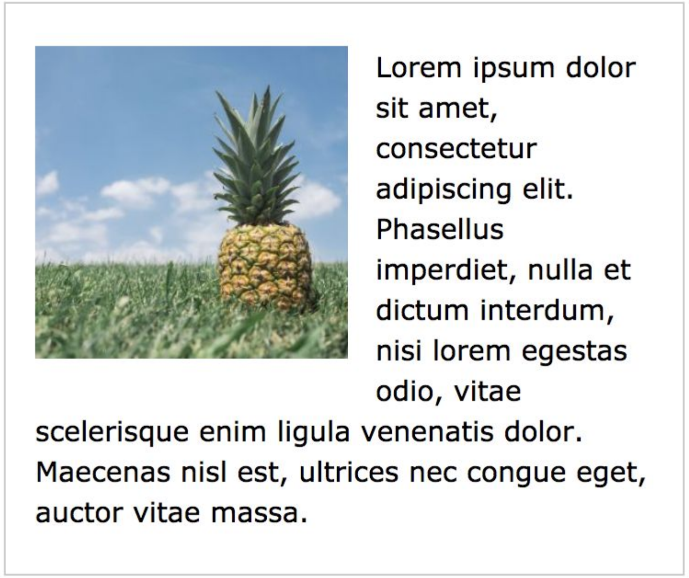
Second paragraph.

- The browser calculates the margin by dividing remaining width evenly for each left/right margin.
- Note: This only works *if* width is less than the width of its container

Float

- Removes a document from the normal document flow and “floats” *left* or *right*
- Default value is *none,* element stays where it normally displays
- Text will wrap around the floated element

```
img {
    float: left;
}

<p><img src="pineapple.jpg"
alt="Pineapple">Lorem ipsum
dolor...diam velit.</p>
```

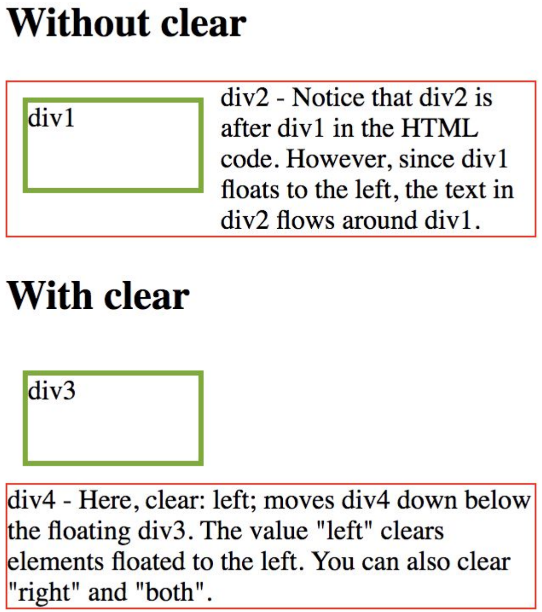


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa.

Clear

- Specifies what side of an element floating elements are **not** allowed
- Like float, you can specify *left*, *right*, or *none* but you can also clear *both*

```
.div1 { float: left; … }   <h2>Without clear</h2>
                           <div class="div1">div1</div>
.div2 {...}                <div class="div2">div2</div>

.div3 { float: left; … }   <h2>With clear</h2>
                           <div class="div3">div3</div>
.div4 { clear: left; … }   <div class="div4">div4</div>
```



Without clear

div1

div2 - Notice that div2 is after div1 in the HTML code. However, since div1 floats to the left, the text in div2 flows around div1.

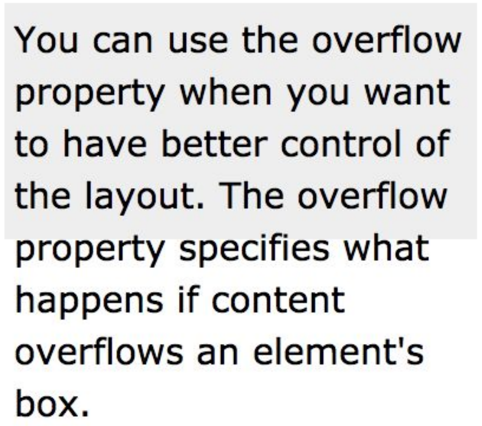
With clear

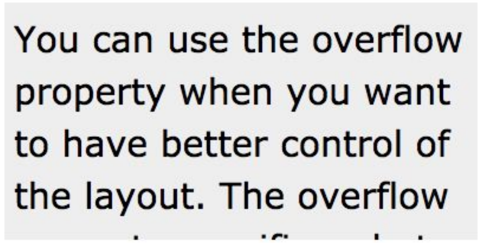
div3

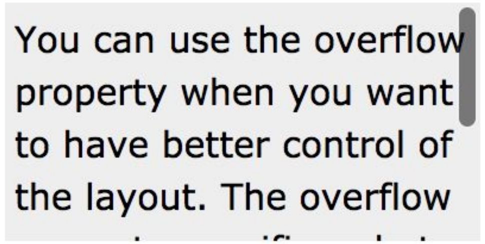
div4 - Here, clear: left; moves div4 down below the floating div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".

Overflow

- Specifies what to do if an element’s content is larger than the specified space
  - visible - Default. The overflow is not clipped. It 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, but a scrollbar is added to see the rest of the content
  - auto - If overflow is clipped, a scrollbar should be added to see the rest of the content


You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

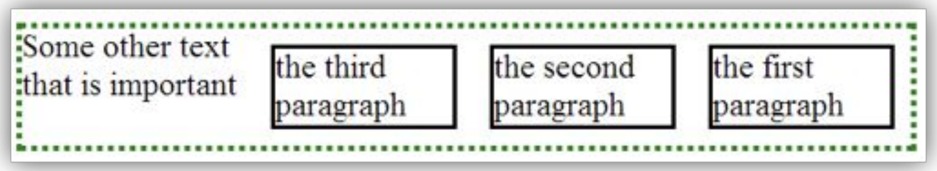

You can use the overflow property when you want to have better control of the layout. The overflow


You can use the overflow property when you want to have better control of the layout. The overflow

How to: Multi-column layouts

```
p {
    float: right;
    width: 20%;
    margin: 0.5em;
    border: 2px solid black;
}

div {
    border: 3px dotted green;
    overflow: hidden;
}
```

```
<div>
    <p>the first paragraph</p>
    <p>the second paragraph</p>
    <p>the third paragraph</p>
    Some other text that is important
</div>
```


Some other text that is important

the third paragraph

the second paragraph

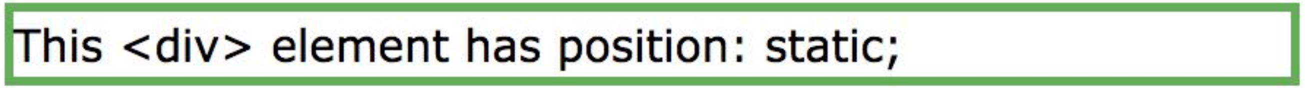
the first paragraph

Positioning

- Property that lets you define how to place an element
- There are 5 position values
  - static
  - relative
  - fixed
  - absolute
  - sticky
- Elements can be positioned using the *top, right, bottom,* and *left* properties
- These properties can’t be set until *position* is set

Positioning: Static

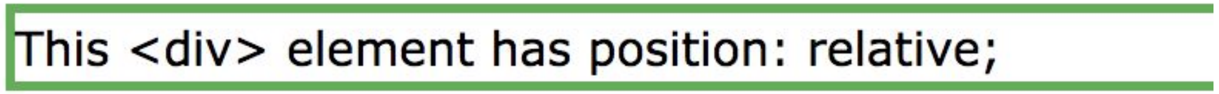
- Elements are positioned *static* by default
- They are not affected by *top, right, bottom,* and *left*


This <div> element has position: static;

```
div.static {
    position: static;
    border: 3px solid #73AD21;
}
```

Positioning: Relative

- Elements are positioned *relative* to their original position
- Setting *top, right, bottom,* and *left* moves the element from its original position

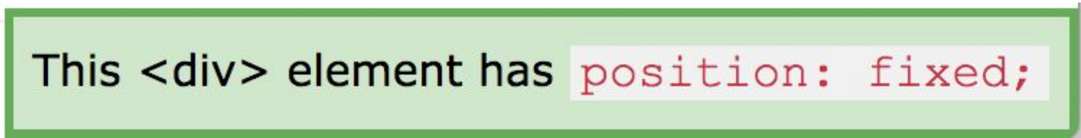

This <div> element has position: relative;

```
div.static {
    position: relative;
    left: 30px;
    border: 3px solid #73AD21;
}
```

Positioning: Fixed

- Elements are positioned relative to the browser window
- Setting *top, right, bottom,* and *left* position the element in the window

```
div.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 3px solid #73AD21;
}
```



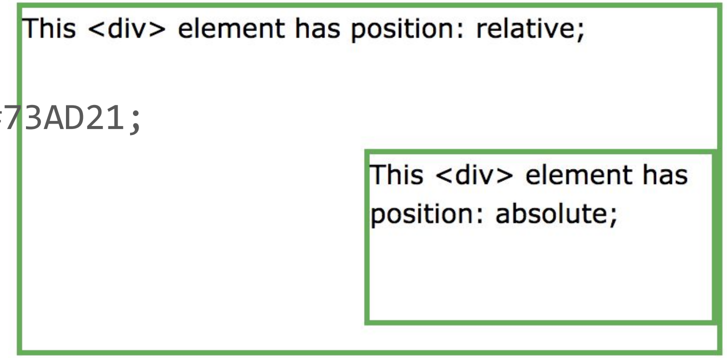
This <div> element has `position: fixed;`

Positioning: Absolute

- Elements are positioned relative to the nearest *positioned ancestor* (instead of the browser window)
- If an element has no positioned ancestor, it is relative to the document body

```
div.relative {
    position: relative;
    width: 400px;
    height: 200px;
    border: 3px solid #73AD21;
}
```

```
div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #73AD21;
}
```

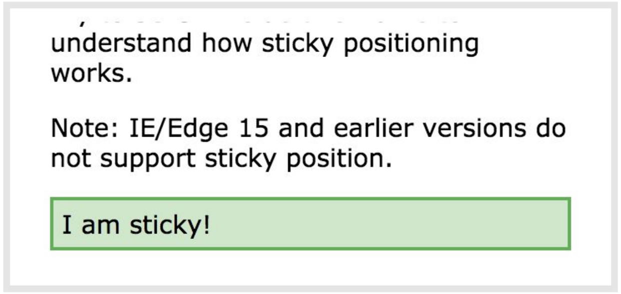

This <div> element has position: relative;

This <div> element has position: absolute;

Positioning: Sticky

- Elements are positioned based on the user's scroll position
- The element is *relative* until the user scrolls past the elements original position and it becomes *fixed* and vice versa

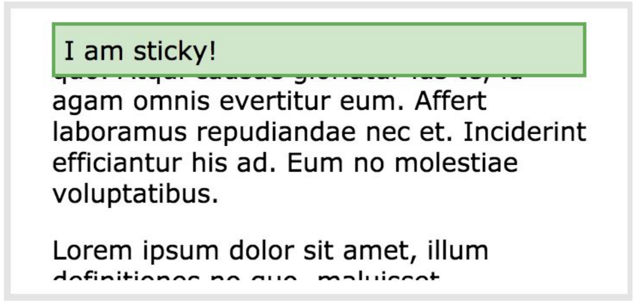
```
div.sticky {
    position: -webkit-sticky; /* Safari */
    position: sticky;
    top: 0;
    background-color: green;
    border: 2px solid #4CAF50;
}
```



understand how sticky positioning works.

Note: IE/Edge 15 and earlier versions do not support sticky position.

I am sticky!



I am sticky!

agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

Lorem ipsum dolor sit amet, illum

z-index

- z-index controls the *stack order* of an element
- The elements with a more *positive* z-index come in front of elements with a lesser z-index

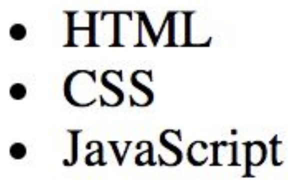
```
z-index: -1;
```

- This only works on *positioned elements*

Display

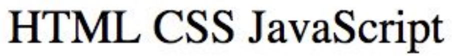
- Very important in controlling CSS layout
- Controls if/how an element is displayed
- Possible values: `inline`, `block`, **`none`**

```
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>
```



- HTML
- CSS
- JavaScript

```
li {
    display: inline;
}
```

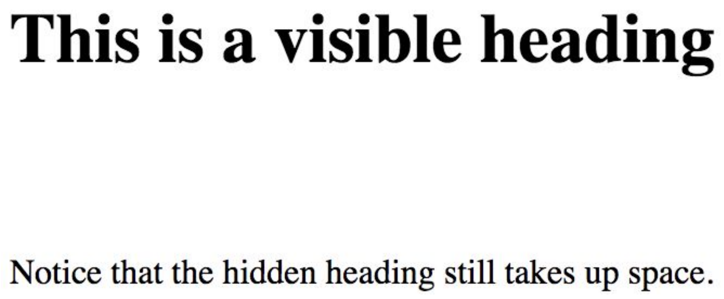


HTML CSS JavaScript

Visibility

- Another property used to control if an element is shown
- Possible values: `hidden`, `visible`

```
h1.hidden {
    visibility: hidden;
}

<h1>This is a visible heading</h1>
<h1 class="hidden">This is a hidden
heading</h1>
<p>Notice that the hidden heading still
takes up space.</p>
```



This is a visible heading

Notice that the hidden heading still takes up space.

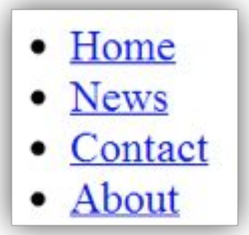
display:none vs visibility

- With display:none the element is taken out of the document flow, as if it wasn’t there to begin with
- With visibility:hidden the element is still in the document and takes up the original space it would take
- Example

How to: Navigation Bar

- Nav bars can be thought of like a list of options

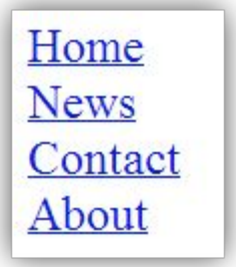
```
<ul>
    <li><a href="home.html">Home</a></li>
    <li><a href="news.html">News</a></li>
    <li><a href="contact.html">Contact</a></li>
    <li><a href="about.html">About</a></li>
</ul>
```


- Home
- News
- Contact
- About

How to: Navigation Bar

- Let’s remove the bullets and any margin or padding

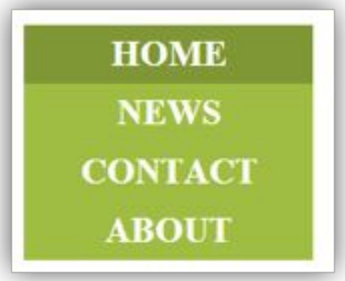
```
<ul>
    <li><a href="home.html">Home</a></li>
    <li><a href="news.html">News</a></li>
    <li><a href="contact.html">Contact</a></li>
    <li><a href="about.html">About</a></li>
</ul>

ul {
    list-style-type:none;
    margin:0;
    padding:0;
}
```


Home
News
Contact
About

How to: Navigation Bar

- Now let’s add some additional styling to make it look like buttons

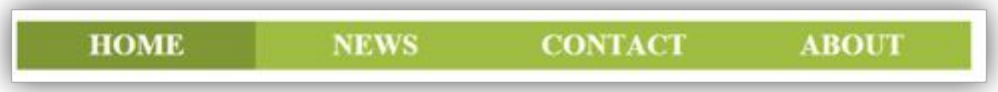
```
a:link,a:visited{
     display:block;
     width:120px;
     font-weight:bold;
     color:#FFFFFF;
     background-color:#98bf21;
     text-align:center;
     padding:4px;
     text-decoration:none;
     text-transform:uppercase;
}
a:hover,a:active{background-color:#7A991A;}
```



How to: Navigation Bar

- Let’s make it look more like a bar than a list

```
a:link,a:visited{
     display:block;
     width:120px;
     font-weight:bold;
     color:#FFFFFF;
     background-color:#98bf21;
     text-align:center;
     padding:4px;
     text-decoration:none;
     text-transform:uppercase;
}

a:hover,a:active{background-color:#7A991A;}
```



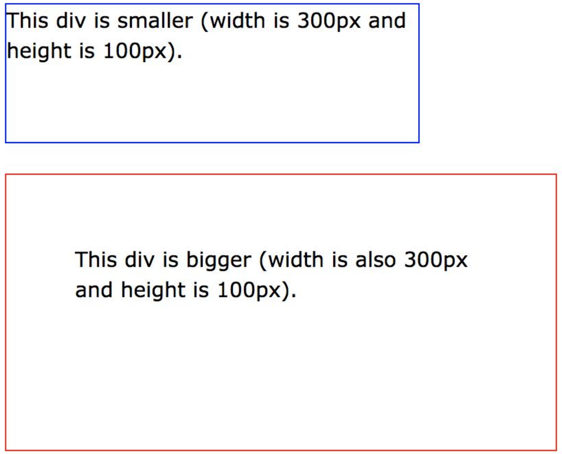
```
ul{
     list-style-type:none;
     margin:0;
     padding:0;
     overflow:hidden;
}

li{float:left;}
```

box-sizing

- By default, the width and height of an element is calculated like this:

  width + padding + border = actual width of an element

  height + padding + border = actual height of an element


This div is smaller (width is 300px and height is 100px).

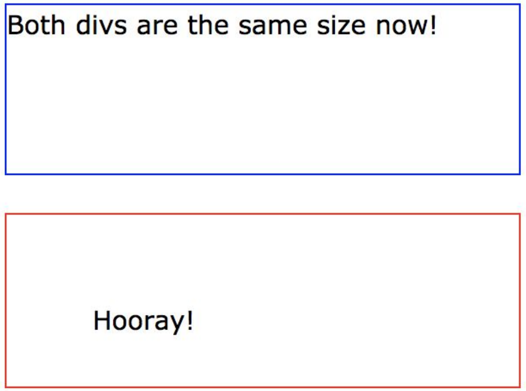
This div is bigger (width is also 300px and height is 100px).

```
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue;
}
```

```
.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid red;
}
```

box-sizing

- The *box-sizing* property allows us to include the padding and border in an element's total width and height.
- The *border-box* value makes sure that margin and padding are included



Both divs are the same size now!

Hooray!

```
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue;
  box-sizing: border-box;
}
```

```
.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid red;
  box-sizing: border-box;
}
```

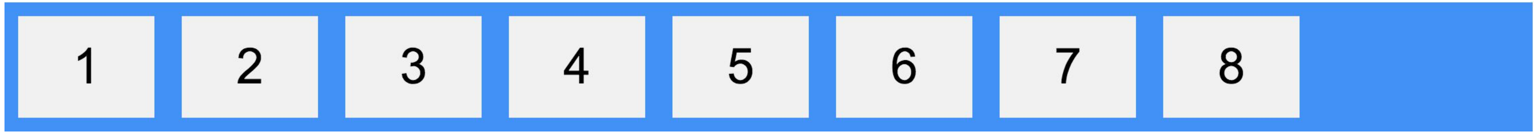
box-sizing

- Since the result of using the `box-sizing: border-box;` is so much better, many developers want all elements on their pages to work this way.
- The code below ensures that all elements are sized in this more intuitive way

```
* {
    box-sizing: border-box;
}
```

Flexbox

- Before the Flexbox 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
- The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

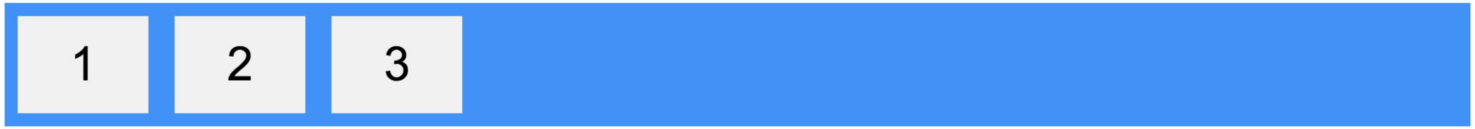


Flexbox

- To start using the Flexbox model, you need to first define a flex container

```html
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

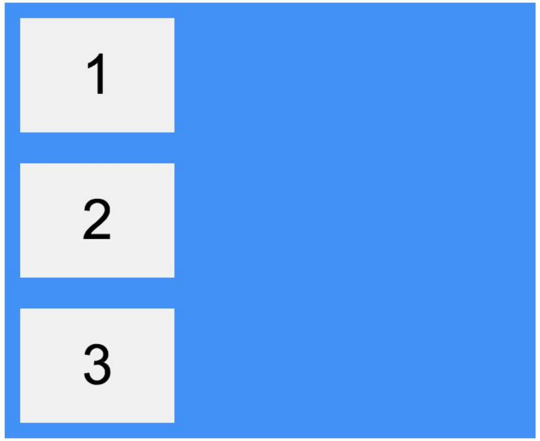
```css
.flex-container {
  display: flex;
}
```



Flexbox

- The `flex-direction` property defines in which direction the container wants to stack the flex items.
- It can have the values `column`, `row`, `column-reverse`, or `row-reverse`

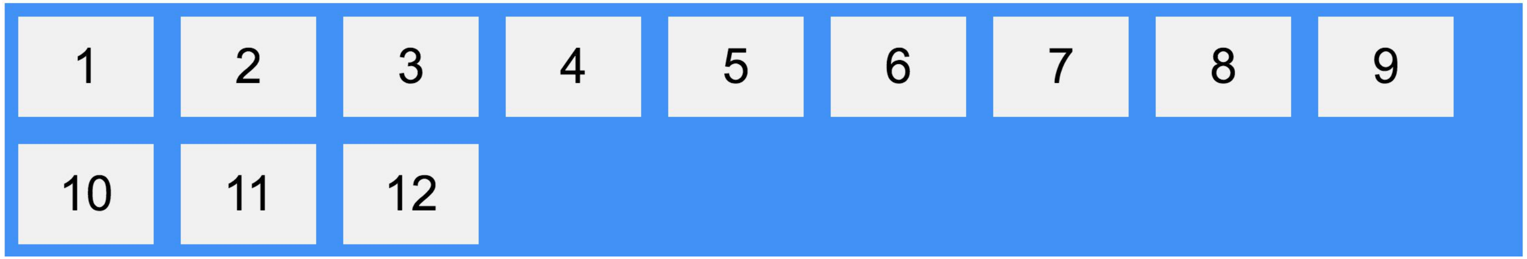
```
.flex-container {
  display: flex;
  flex-direction: column;
}
```



Flexbox

- The `flex-wrap` property specifies whether the flex items should wrap or not.
- It can have the values: `wrap, nowrap, or wrap-reverse`

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



Flexbox

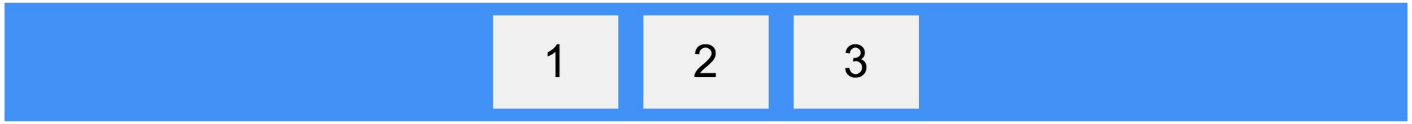
- The `flex-flow` property is a shorthand property for setting both the `flex-direction` and `flex-wrap` properties

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

Flexbox

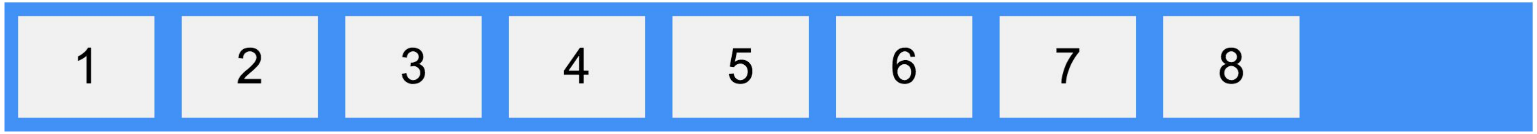
- The `justify-content` property is used to align the flex items
- The possible values are: `center`, `flex-start`, `flex-end`, `space-around`, `space-between`

```
.flex-container {
  display: flex;
  justify-content: center;
}
```



Flexbox

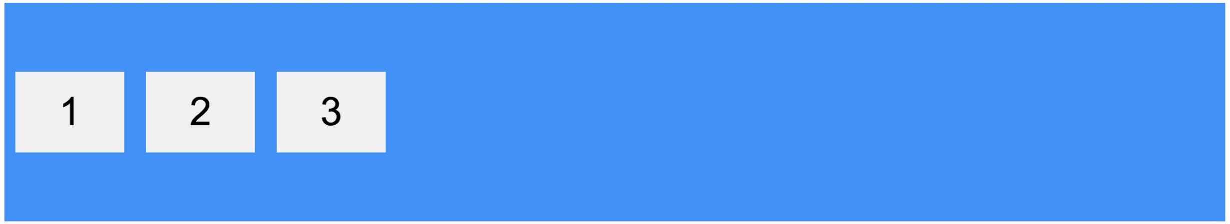
- Before the Flexbox Layout module, there were four layout modes:
  - Block, for sections in a webpage
  - Inline, for text
  - Table, for two-dimensional table data
  - Positioned, for explicit position of an element
- The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.



Flexbox

- The `align-items` property is used to align the flex items vertically
- Possible values are: `center`, `flext-start`, `flex-end`, `stretch`, `baseline`

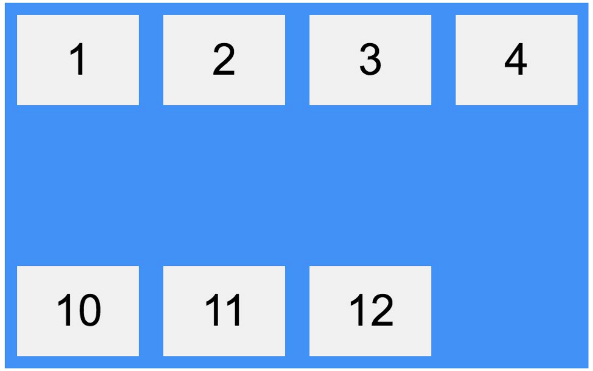
```
.flex-container {
  display: flex;
  height: 200px;
  align-items: center;
}
```



Flexbox

- The `align-content` property is used to align the flex lines
- Possible values are: `center, flext-start, flex-end, stretch, space-around, space-between`

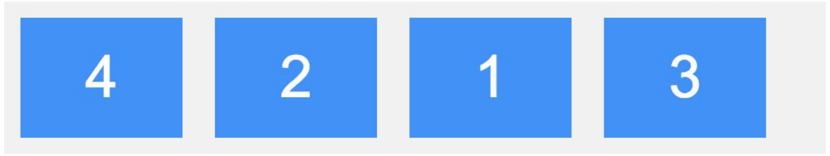
```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: space-between;
}
```



Flexbox Items

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

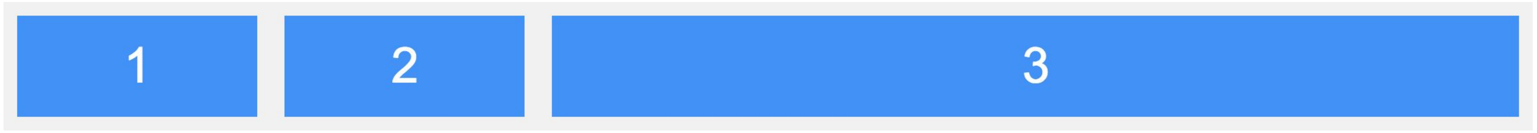
```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```



Flexbox Items

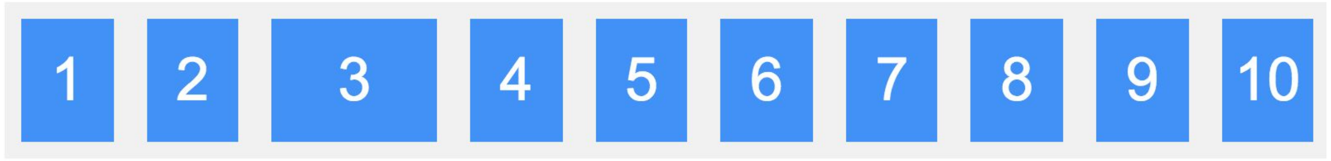
- The `flex-grow` property specifies how much a flex item will grow relative to the rest of the flex items
- The value must be a number, the default value is 0

```
<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>
```



Flexbox Items

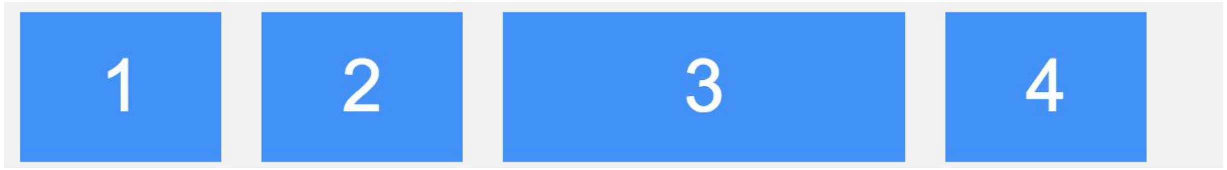
- The `flex-shrink` property specifies how much a flex item will shrink relative to the rest of the flex items
- The value must be a number, default value is 1



Flexbox Items

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

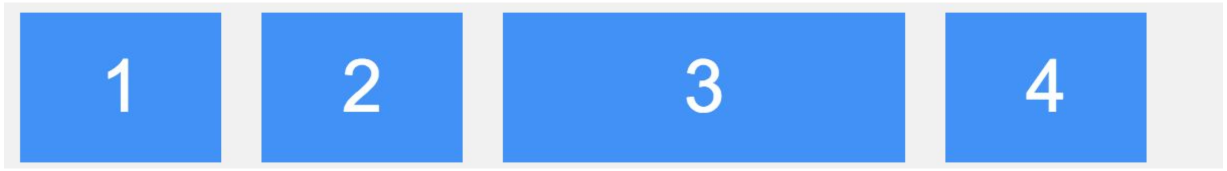
```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-basis: 200px">3</div>
  <div>4</div>
</div>
```



Flexbox Items

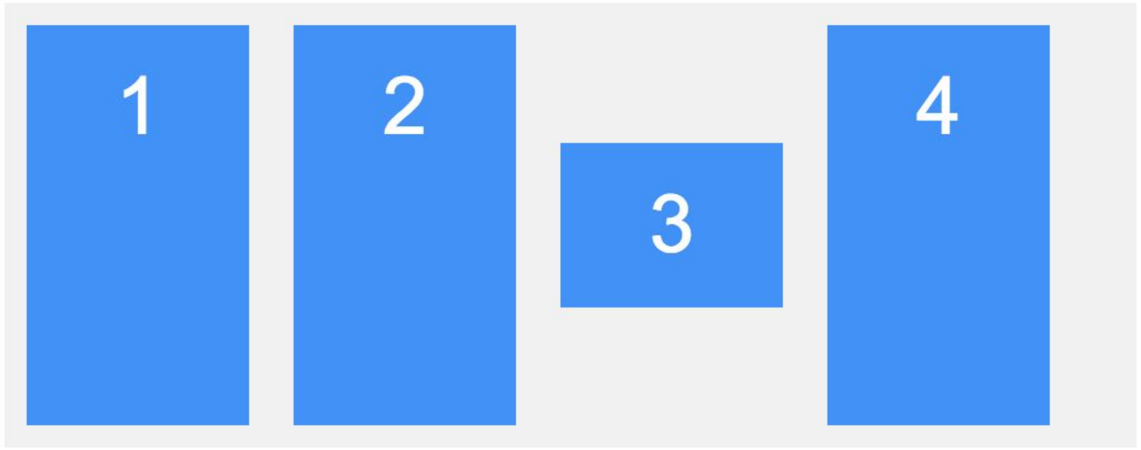
- The `flex` property is a shorthand property for the `flex-grow`, `flex-shrink`, and `flex-basis` properties

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



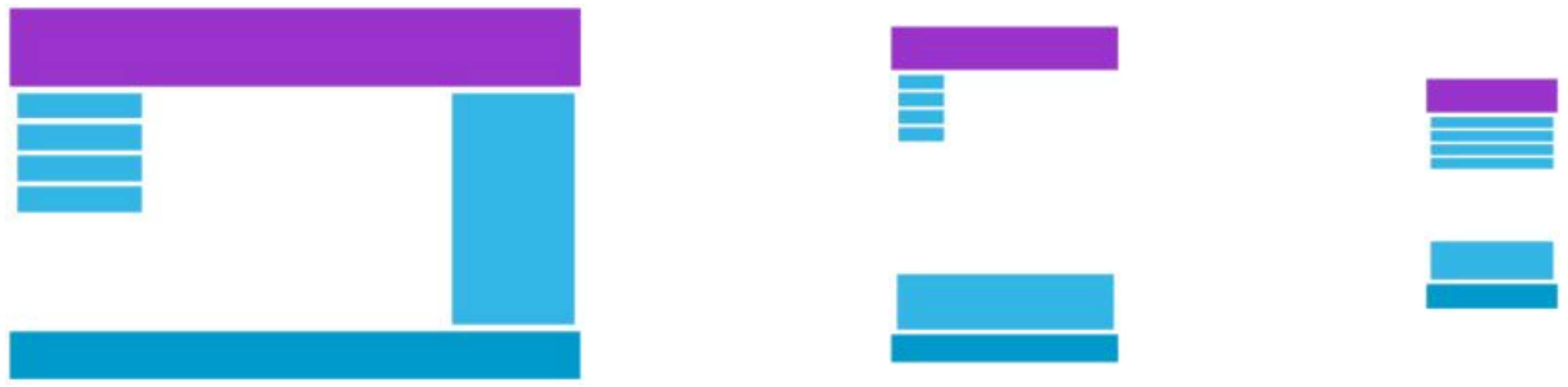
Flexbox Items

- 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



Responsive Design

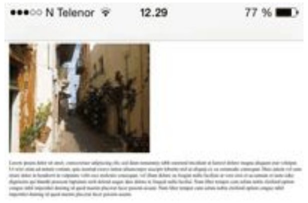
- A way to layout content appropriately depending on the type of device

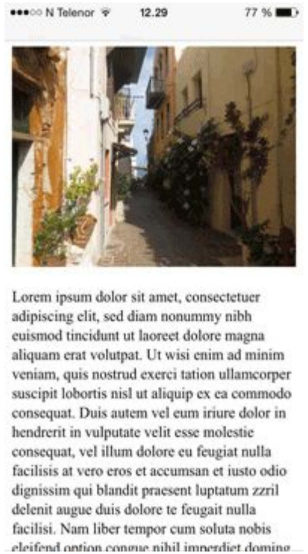


Viewport

- HTML5 introduced the viewport meta tag to allow for responsive design

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





Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming

Media queries

- Media queries were introduced in CSS3
- The `@media` rule is used to include a block of CSS only if a condition is true
- For responsive design that condition is based on screen width

```
@media only screen and (max-width: 500px) {
    body {
        background-color: lightblue;
    }
}
```

Media queries

- The rule of thumb is to always design mobile first, adding breakpoints for bigger screens

```css
/* For desktop: */
.navbar {
  display: flex;
  background-color: #333;
}

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

```css
/* For mobile */
@media screen and (max-width: 700px) {
  .row, .navbar {
    flex-direction: column;
  }
}
```

Media queries

- You can even add a breakpoint for tablets

```
/* For mobile phones: */
.row, .navbar {
  flex-direction: column;
}


@media only screen and (min-width: 600px) {
      /* For tablets: */
      .row {
        flex-direction: column;
      }



}
```

```
@media only screen and (min-width: 768px) {
      /* For desktop: */
      .navbar {
        display: flex;
        background-color: #333;
      }

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

CSS3

- CSS3 is the latest standard for CSS. Some of the most important CSS3 modules are:
  - Selectors - *starts-with*, *ends-with,* and *contains* attribute selector
  - Backgrounds and Borders - new colors, gradients, border-radius
  - Text Effects - *text-shadow, text-overflow, word-wrap*
  - 2D/3D Transformations
  - Animations
  - Multiple Column Layout
  - User Interface - *box-sizing*

Resources

https://www.w3schools.com/css/css_align.asp

https://www.w3schools.com/css/css_navbar.asp

https://www.w3schools.com/css/css3_flexbox.asp

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

https://www.w3schools.com/css/css_rwd_viewport.asp

https://www.w3schools.com/css/css_rwd_mediaqueries.asp

http://www.css3.info/preview/