ARCHITECTING CSS WITH **BOX MODEL**

LAYOUTS

Individual **elements** form a **layout** when they are put together on a **page**.

Using **CSS** we rely on the **box model** to control the **width** and **behavior** of each **element** without the **layout**.

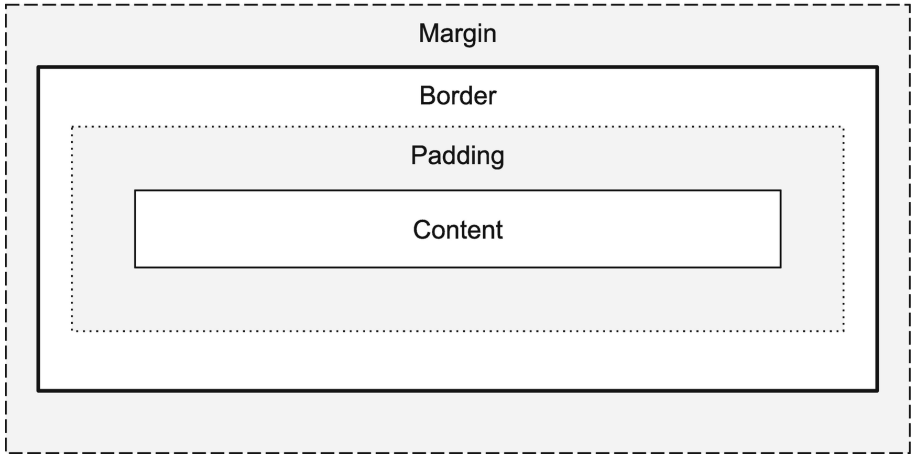
To control how **elements** place themselves in relationship to each other, we can use **properties** such as **display** and **float**.

BOX MODEL

The base for laying out **content** is rooted in the **box model** which describes the **rectangular boxes** that are generated for **elements** in the **document tree**.

BOX MODEL

The **content** is enveloped by **padding**, **border**, and **margin** boxes.



BOX SIZING

The **box-sizing property** that defines the **height** and **width** of an **element** by default has a value of **content-box**.

It means that when a **width** and **height** is defined for an **element**, it is only applied to the **content**.

Adding **padding** or **margin** to the **element** therefore increases the percentage **width** of the total available **viewport** that the **element** utilizes.

CONTENT-BOX

Consider a **two-column layout**, with each **div** equaling **50%** of the **width** of the **viewport**.

The amount of **padding** applied to each **column** needs to be subtracted from the **width** given to the **element** or the **total width** of both **elements** will exceed **100%**.

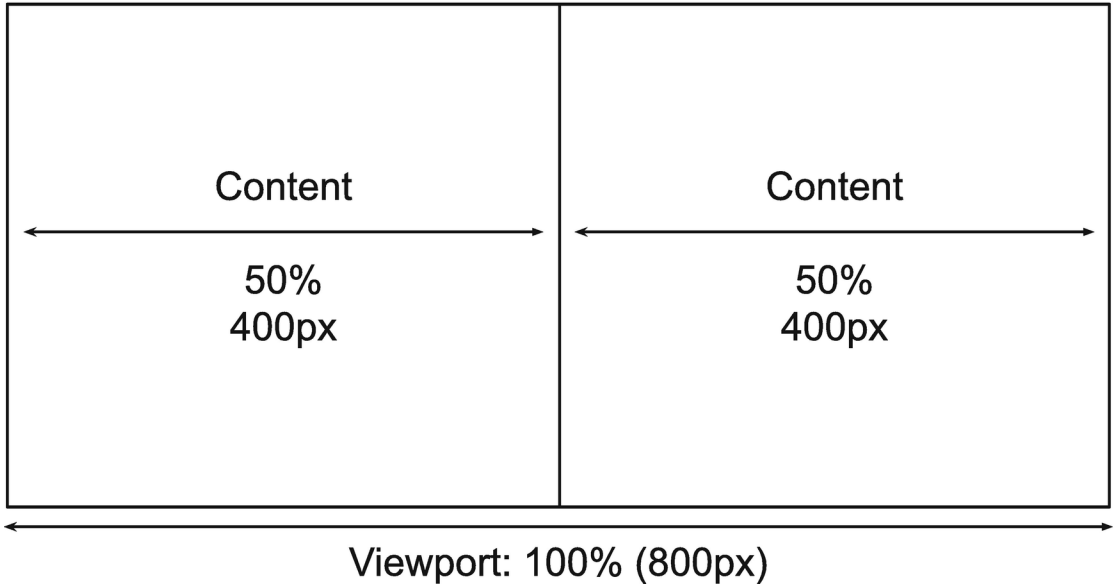
CONTENT-BOX

Consider a **viewport** of **800px wide** containing a **layout** with two **divs**.

If no **padding**, **margin**, or **border** is added to the **divs**, and they are each given a **width** of **50%**, their combined **width** will equal **100%** of the **viewport**.

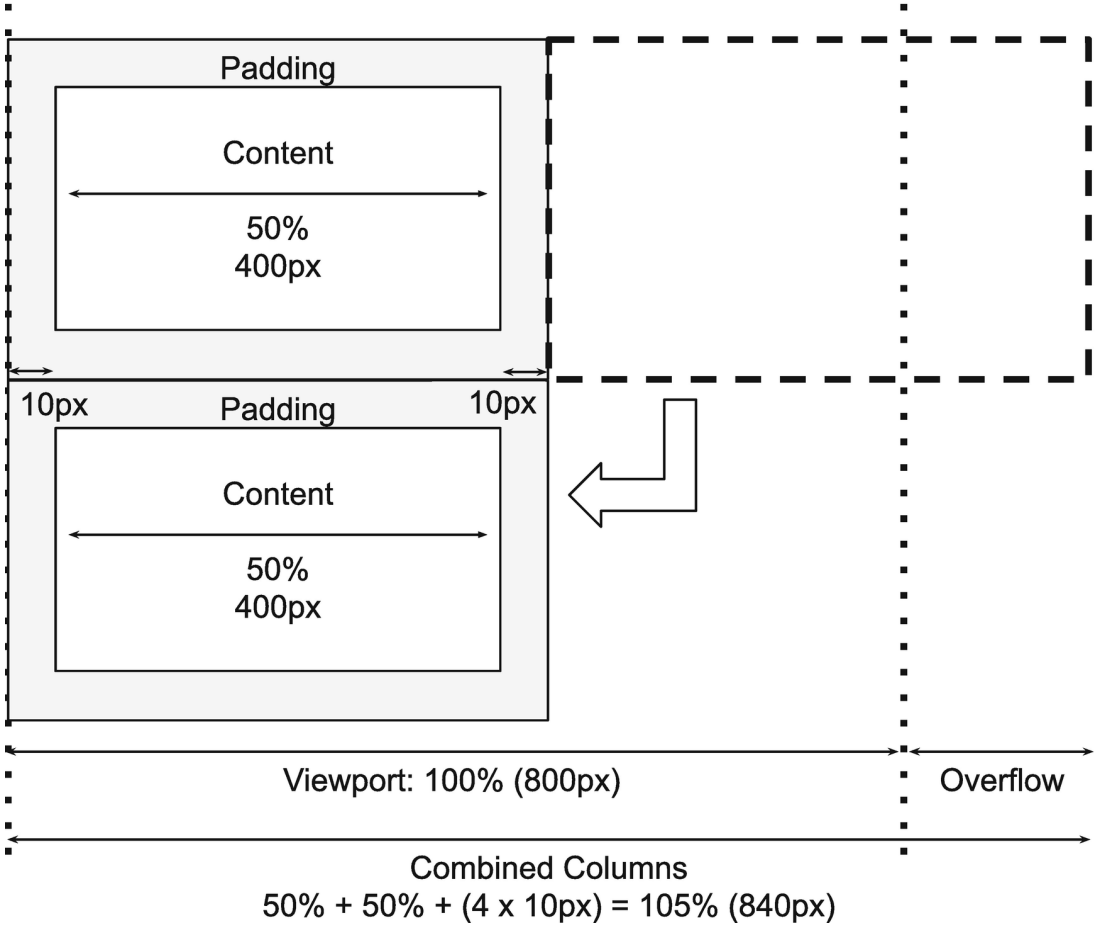
If they are **floated**, they will sit perfectly **side by side** and take up **100%** of the **screen**.

CONTENT-BOX



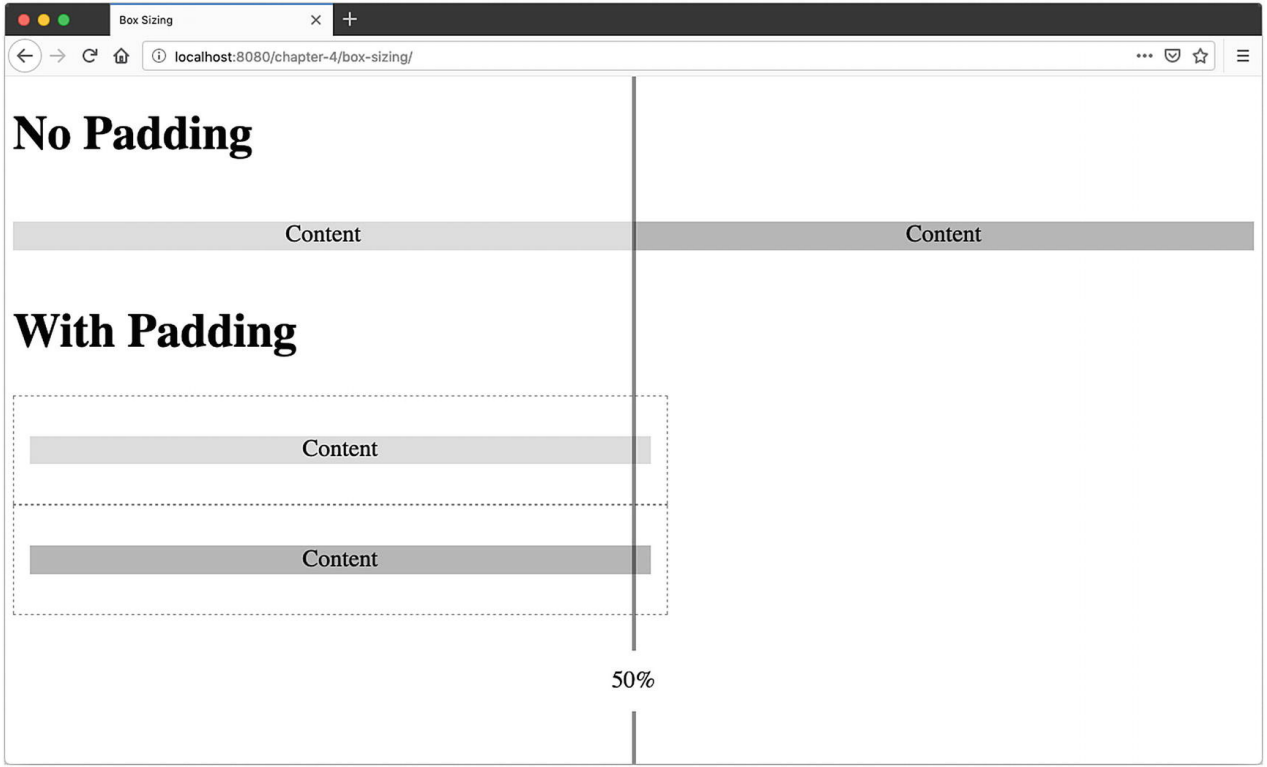
CONTENT-BOX

If **padding** is added to the **columns**, the **width** of the **columns** will increase by the **padding** amount, causing them to exceed the **width** of the **viewport**.


Padding
Content
50%
400px
10px
Padding
10px
Content
50%
400px
Viewport: 100% (800px)
Overflow
Combined Columns
50% + 50% + (4 x 10px) = 105% (840px)

CONTENT-BOX

If the **divs** are **floated**, the second **div** would therefore be **pushed below** the first as their combined **width** is now greater than **100%** of the **container**.


Box Sizing
localhost:8080/chapter-4/box-sizing/
No Padding
Content
Content
With Padding
Content
Content
50%

CONTENT-BOX

Border will behave the same way as **padding**.

Any **border** width applied will need to be included in the sum of **content** and **padding** to calculate the full **width** or **height** of the **elements** included in the **layout**.

MARGIN COLLAPSE

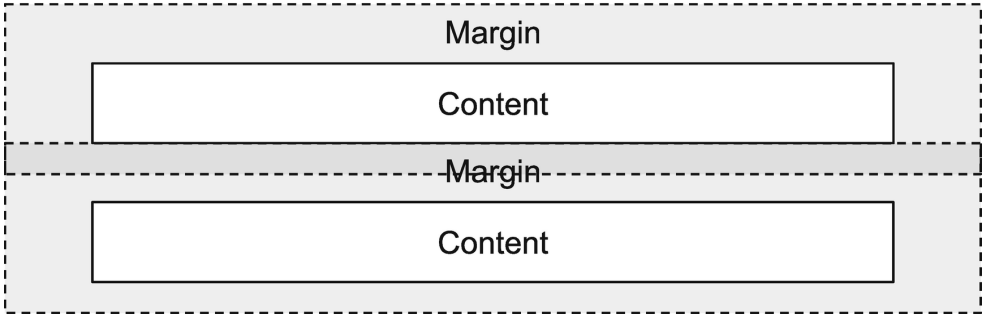
When **sibling elements** both have **padding**, **padding** from both is applied, and the **space** between the two **elements** is the sum of both sets of **padding**.

Margins behave a little differently than **padding**.

MARGIN **COLLAPSE**

Margin, depending on their **context**, can **collapse**.

Margin collapsing is when **top** and **bottom margins** are combined (or **collapsed**) into a **single margin** equal to the **largest** of the **margins** applied.



MARGIN COLLAPSE

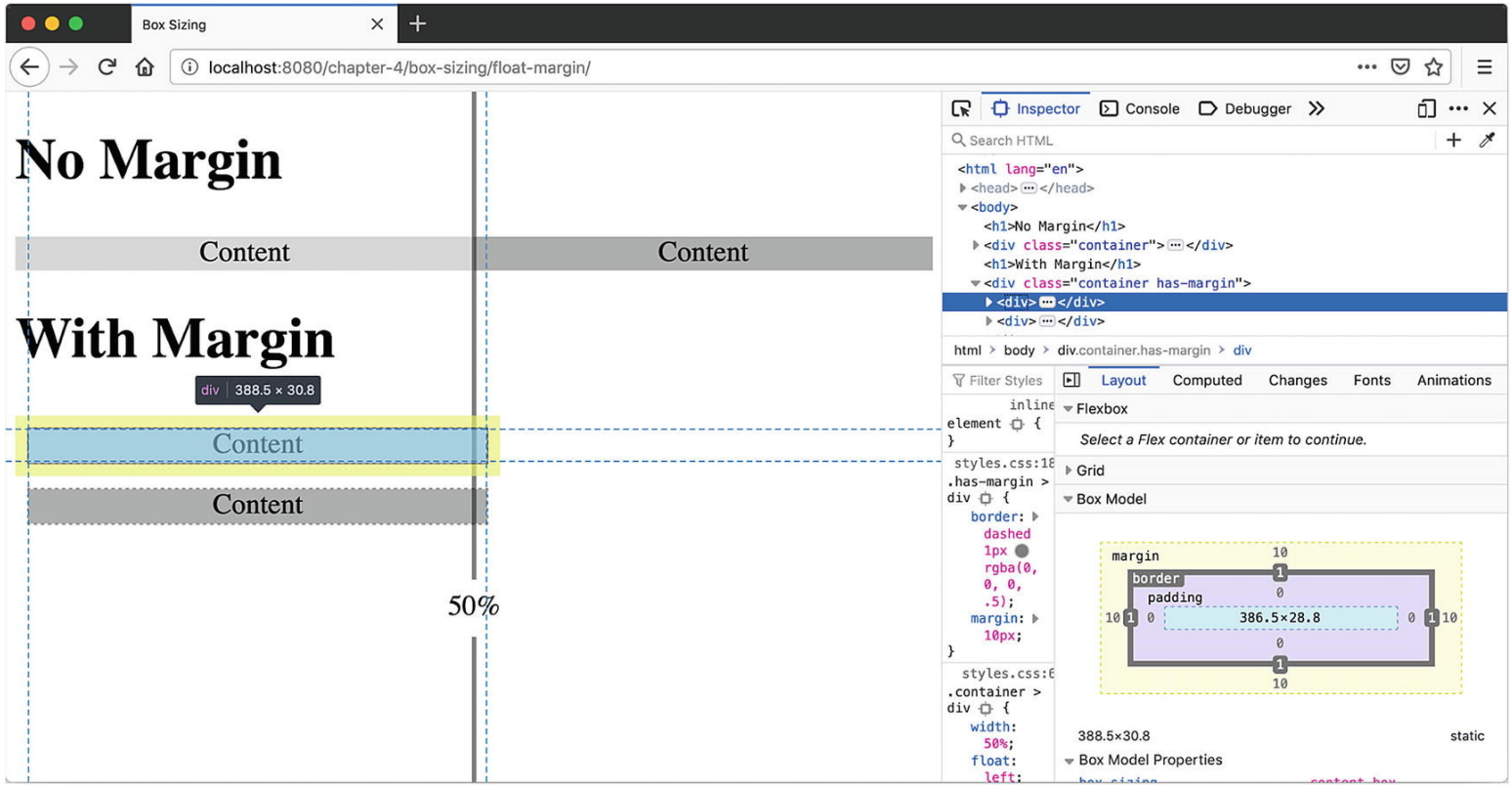
If all **margins** are **negative**, then this **margin** is the **size** of the **most negative margin**.

Left and **right margins** do not collapse.

MARGIN COLLAPSE

If we take the earlier example, where the **columns** have been **floated** and replace the **padding** for **margin**, then the **margin** does not **collapse**.

The **columns** will still **stack** as their **combined total width** is greater than **100%** because **(50 % + 2 x 10px) x 2 = 105%,** but because the **divs** are **floated**, then the **margins** do not **collapse**.


Box Sizing
localhost:8080/chapter-4/box-sizing/float-margin/
No Margin
Content
Content
With Margin
div | 388.5 × 30.8
Content
Content
50%
Inspector
Console
Debugger
Search HTML
<html lang="en">
<head> </head>
<body>
<h1>No Margin</h1>

<h1>With Margin</h1>

html > body > div.container.has-margin > div
Filter Styles
Layout
Computed
Changes
Fonts
Animations
element { }
.has-margin > div {
border: dashed 1px rgba(0, 0, 0, .5);
margin: 10px;
}
.container > div {
width: 50%;
float: left;
Flexbox
Select a Flex container or item to continue.
Grid
Box Model
margin
border
padding
386.5×28.8
388.5×30.8
static
Box Model Properties

SUMMARY

The benefit of keeping the **box-sizing** value as **content-box** is that when a **width** or **height** value is assigned to the **content**, it will not be subject to **side effects** from **padding** added.

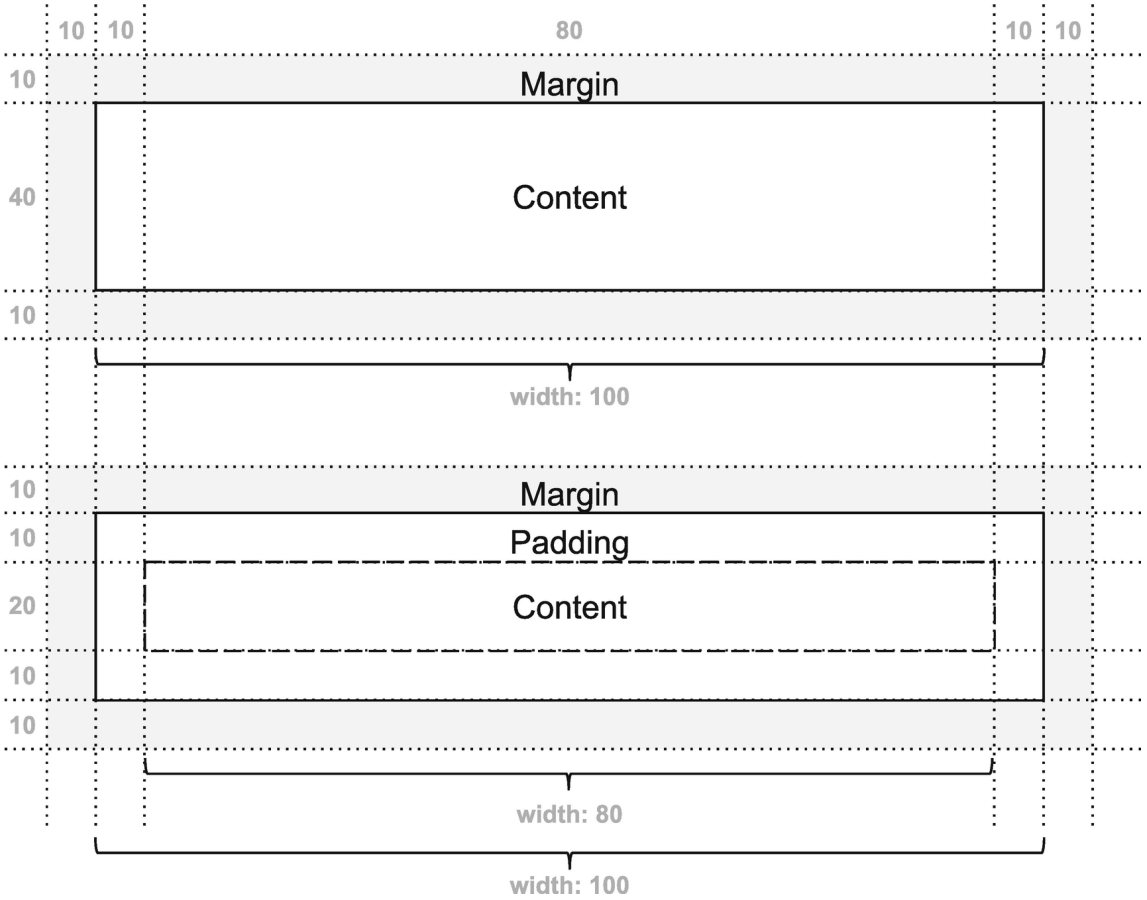
The **content** will be exactly the **height** or **width** it was assigned by us.

BORDER-BOX

Assigning **box-sizing** to **border-box** changes how an **element's width** and **height** is calculated.

Instead of encompassing just the **content**, it takes in the **content**, **padding**, and **border**.

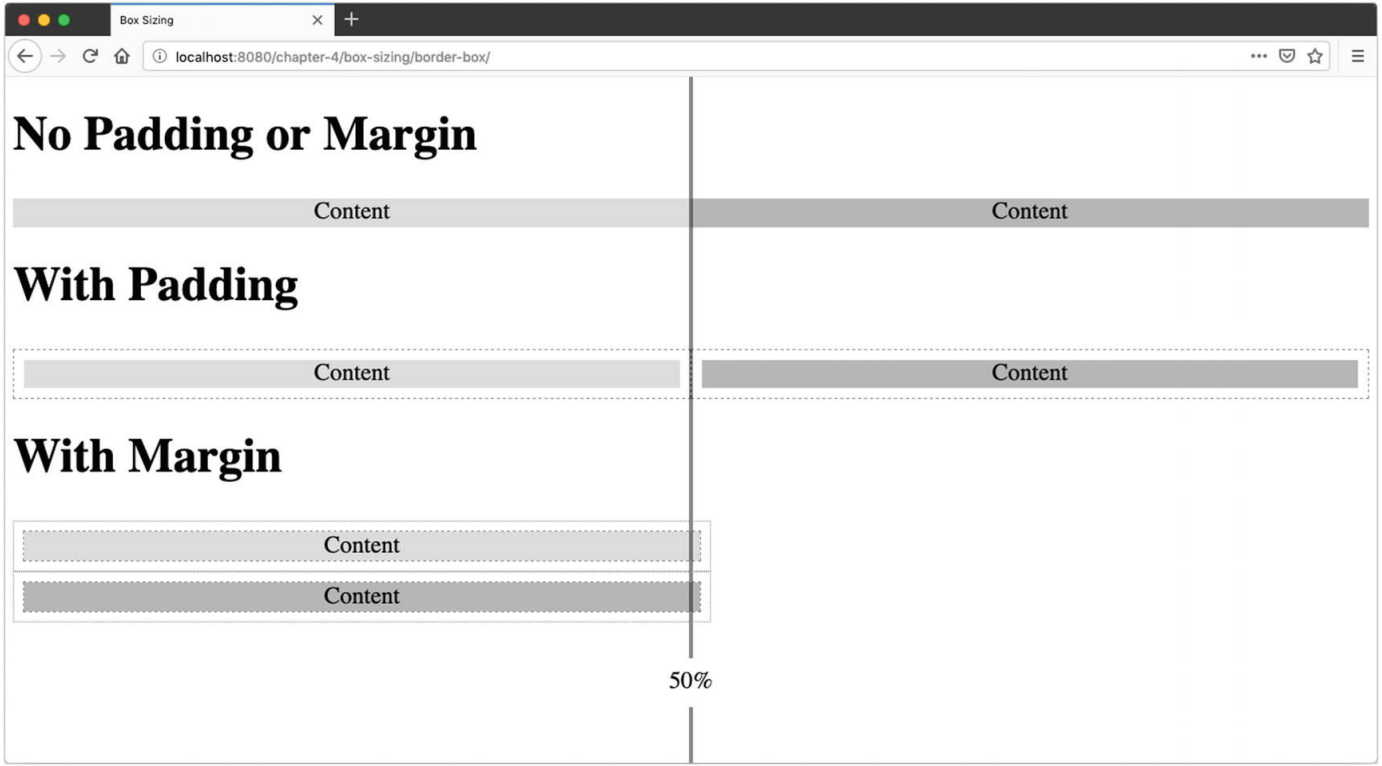
When **padding** or **border** is added, the **width** and **height** of the **content** itself is therefore decreased.


10
10
80
10
10
10
Margin
40
Content
10
width: 100
10
Margin
10
Padding
20
Content
10
10
width: 80
width: 100

BORDER-BOX

If we take the example of the **two floated columns**, we will see that the two **columns** retain a **width** of **50%**.

The **margin** still behaves the same way as with **content-box**.


Box Sizing
localhost:8080/chapter-4/box-sizing/border-box/
No Padding or Margin
Content
Content
With Padding
Content
Content
With Margin
Content
Content
50%

BORDER-BOX

Box-sizing is not **inherited** so it will need to be applied to all **elements** for which it needs to be changed.

DISPLAY

Margin and **padding** allow for manipulating the **display** of the **element**.

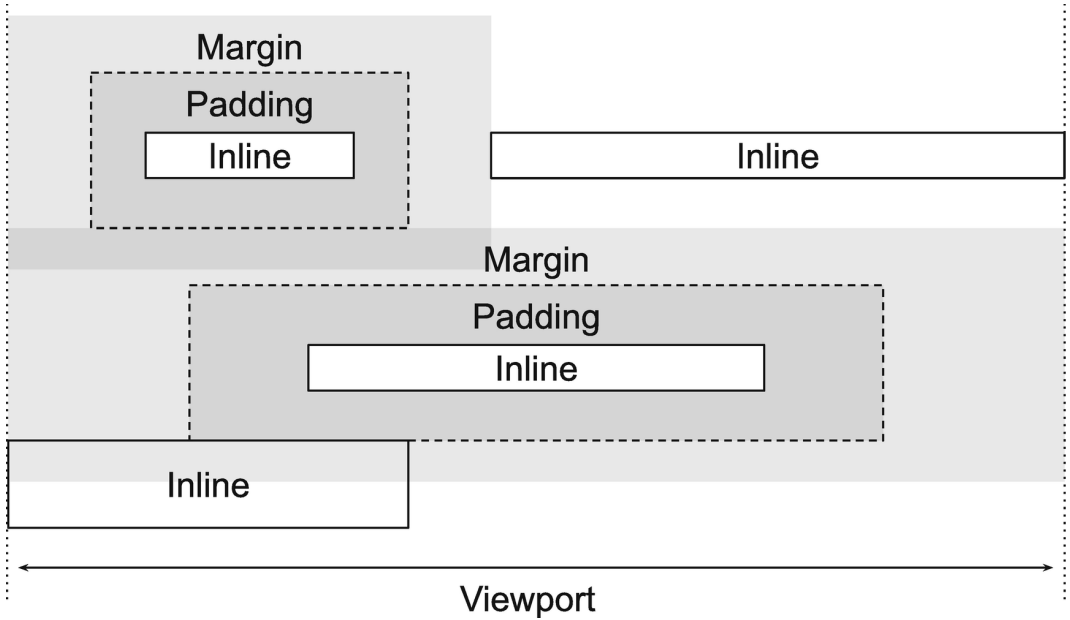
The **display property** manipulates how **elements** are displayed **in relationship to one another** by specifying the **type of rendering** the **box** will use for the **element**.

INLINE

Considered **flow content**, **inline elements** are placed **inline** with the **text** when in a **flow layout**.

When the **content** is displayed **inline**, by default **elements** go from **left to right** and set themselves **side to side**, width permitting.

INLINE


Margin
Padding
Inline
Inline
Margin
Padding
Inline
Inline
Viewport

INLINE

By default, **elements**, regardless of **padding**, and **margin**, will align themselves to the **text baseline**.

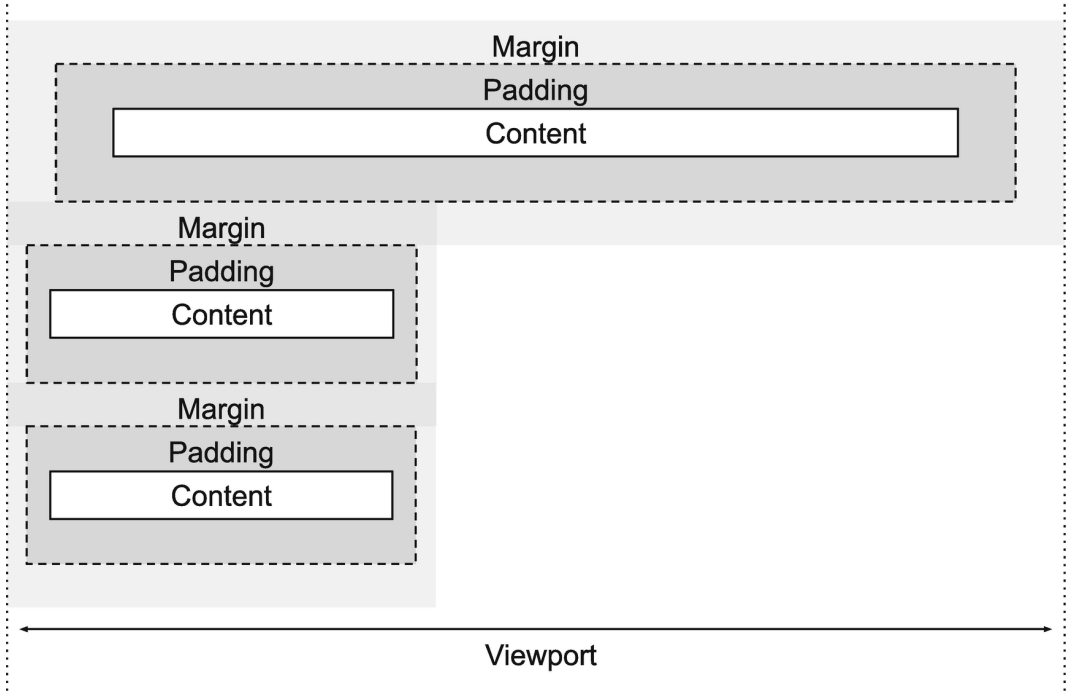
If the **width** does not permit, the **content** will **wrap below**.

BLOCK ELEMENTS

Considered **flow content**, **block elements stack atop one another** unless they are affected by another **property** such as **float**.

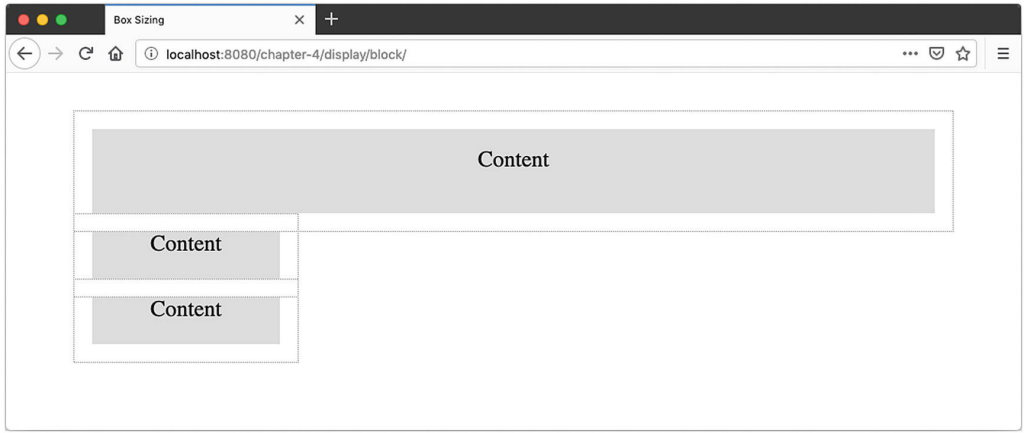
By default, **block-level elements** will take the **full width** of the **viewport**.

BLOCK ELEMENTS



BLOCK ELEMENTS

If a **width** is applied, even if there is still enough **room inline** of the **element**, the **block element** will still place itself **below the previous**.



BLOCK ELEMENTS

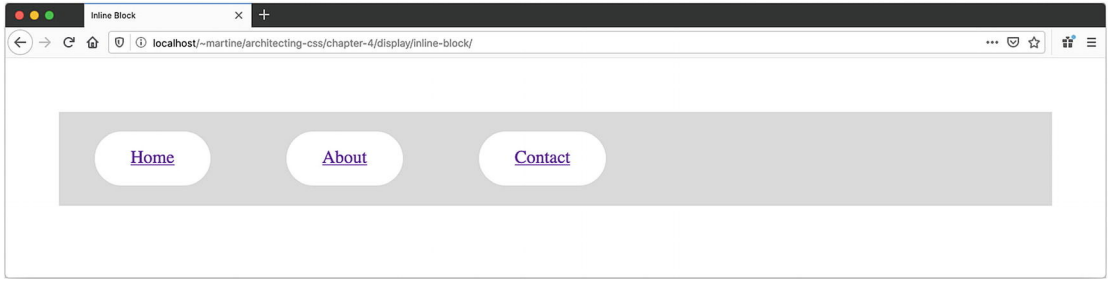
If an **inline element** is placed after a **block element**, the **inline element** will still be placed after the **block element**.

INLINE-BLOCK

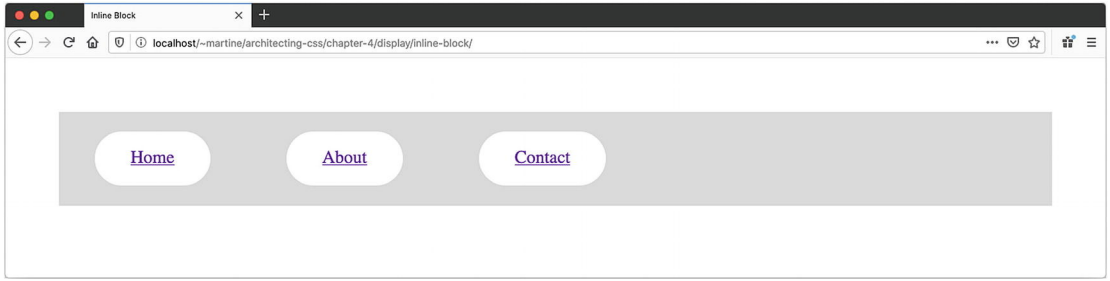
Inline-block utilizes concepts from both **block** and **inline**.

It will behave like a **block element** but **flow** with the **surrounding content** as if it were **inline**.

A common use case for **inline-block** is **horizontal navigation element**.




```
<body>
 <nav>
  <ul>
   <li><a href="">Home</a></li>
   <li><a href="">About</a></li>
   <li><a href="">Contact</a></li>
  </ul>
 </nav>
</body>
```




```
html, body {
 font-size: 24px;
 padding: 36px;
 margin: 0;
}
ul {
 margin: 0;
 padding-left: 0;
 background: lightgray;
}
…
```

```
…
li {
 list-style-type: none;
 display: inline-block;
 margin: 2rem;
}
a {
 padding: 1rem 2rem;
 background: white;
 border-radius: 2rem;
}
```