INFO2180 - LECTURE 3

CSS CONT'D

THE CASCADE

In CSS, all styles *Cascade* from the top of the stylesheet to the bottom. Therefore, styles can be added or overwritten as the stylesheet progresses.

```css
p {
  background: orange;
  font-size: 24px;
}

p {
  background: green;
}
```

```css
p {
  background: green;
  background: orange;
}
```

There are, however, times where the cascade doesn’t play so nicely. Those times occur when different types of selectors are used and the ***specificity*** of those selectors breaks the cascade.

SPECIFICITY

SPECIFICITY

EVERY SELECTOR IN CSS HAS A SPECIFICITY WEIGHT. A SELECTOR'S SPECIFICITY WEIGHT, ALONG WITH ITS PLACEMENT IN THE CASCADE, IDENTIFIES HOW ITS STYLES WILL BE RENDERED.

http://learn.shayhowe.com/html-css/getting-to-know-css/#specificity

SPECIFICITY WEIGHT

- The type selector has the lowest specificity weight and holds a point value of **`0-0-1`**.
- The class selector has a medium specificity weight and holds a point value of **`0-1-0`**.
- Lastly, the ID selector has a high specificity weight and holds a point value of **`1-0-0`**.

```html
<p id="food"> ... </p>
```

```css
#food {
  background: green;
}
p {
  background: orange;
}
```

`#food` `(1-0-0)` is more specific than `p` `(0-0-1)`.

```html
<div class="hotdog">
  <p> ... </p>
  <p> ... </p>
  <p class="mustard"> ... </p>
</div>
```

```css
.hotdog p {
  background: brown;
}
.hotdog p.mustard {
  background: yellow;
}
```

`.hotdog p.mustard` `(0-2-1)` is more specific than `.hotdog p` `(0-1-1)`.

COLOURS

FOUR (4) PRIMARY WAYS TO REPRESENT COLOURS

- Keywords e.g. `white, red, green, blue`
- Hexadecimal Notation e.g. `#FF6600`
- RGB e.g. `rgb(128, 0, 0) or rgba(128, 0, 0, .5)`
- HSL e.g. `hsl(0, 100%, 25%) or hsla(0, 100%, 25%, .36)`

KEYWORDS

```css
.my-class {
  background: maroon;
}
.some-other-class {
  background: yellow;
}
```

HEXADECIMAL

```
.some-class {
  background: #800000;
}
.another-class {
  background: #fc6;
}
```

`#fc6` is short hand for `#ffcc66`

RED-GREEN-BLUE (RGB)

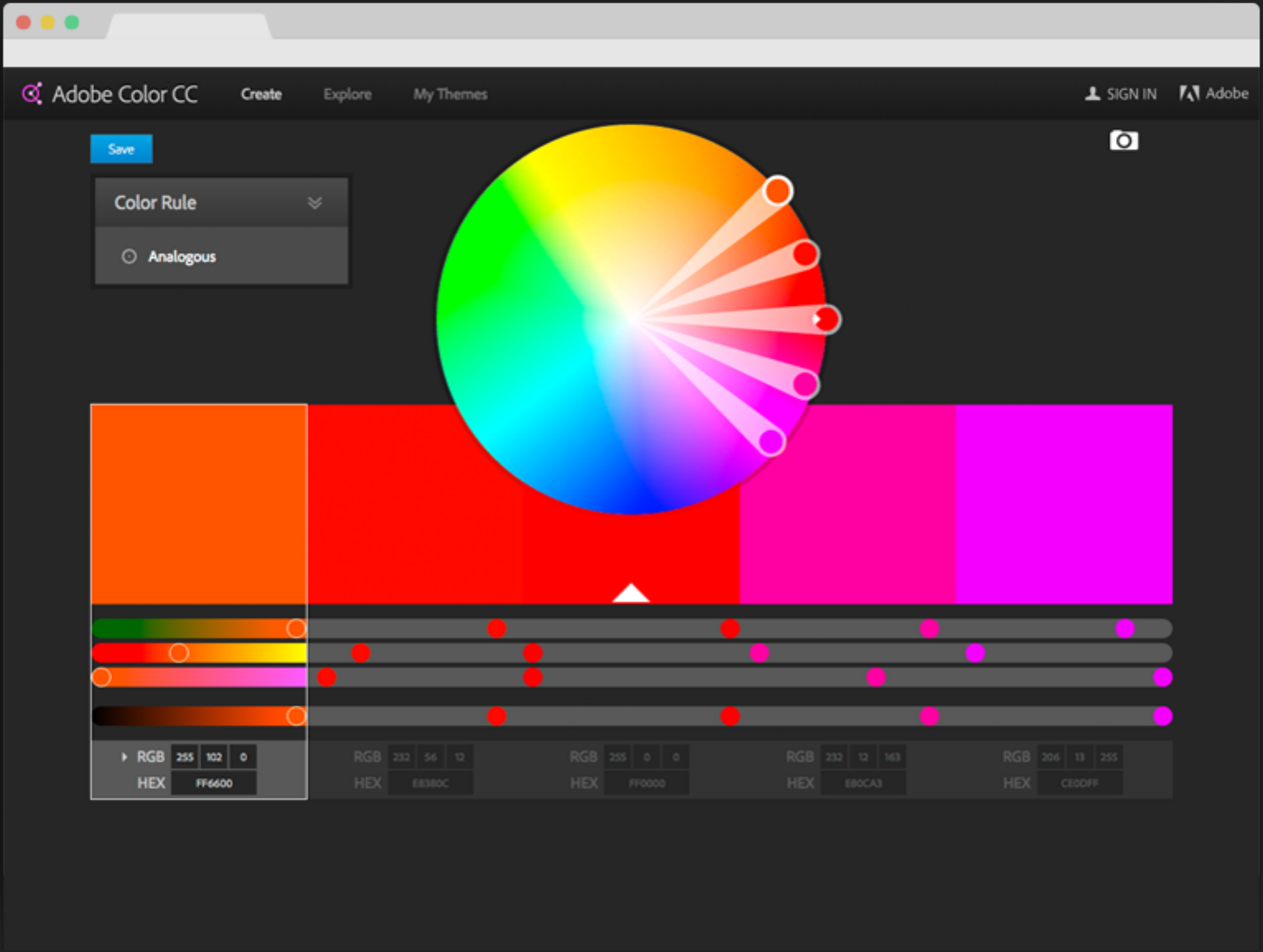
```
.task {
  background: rgb(128, 0, 0);
}
.task {
  background: rgba(128, 0, 0, .25);
}
```

HUE-SATURATION-LIGHTNESS (HSL)

```css
.task {
  background: hsl(0, 100%, 25%);
}
.count {
  background: hsla(60, 100%, 50%, .25);
}
```

Adobe Color CC

https://color.adobe.com/



RGB	255 102 0	232 56 12	255 0 0	232 12 163	206 13 255
HEX	FF6600	E8380C	FF0000	E80CA3	CE0DFF

LENGTHS

LENGTHS

- Pixels
- Percentages
- Em

These are the most popular, but there are others.

EXAMPLE USING PIXELS

```css
p {
   font-size: 14px;
}
```

The pixel is equal to 1/96th of an inch; thus there are 96 pixels in an inch.

EXAMPLE WITH PERCENTAGES

```css
div {
  width: 50%;
}
```

This **`div`** will be 50% of its parent element.

EXAMPLE WITH EM

```
.banner {
  font-size: 14px;
  width: 5em;
}
```

The width will be 5 times its font-size. 5 x 14 = 70px

When a font size is not explicitly stated for an element, the em unit will be relative to the font size of the closest parent element with a stated font size.

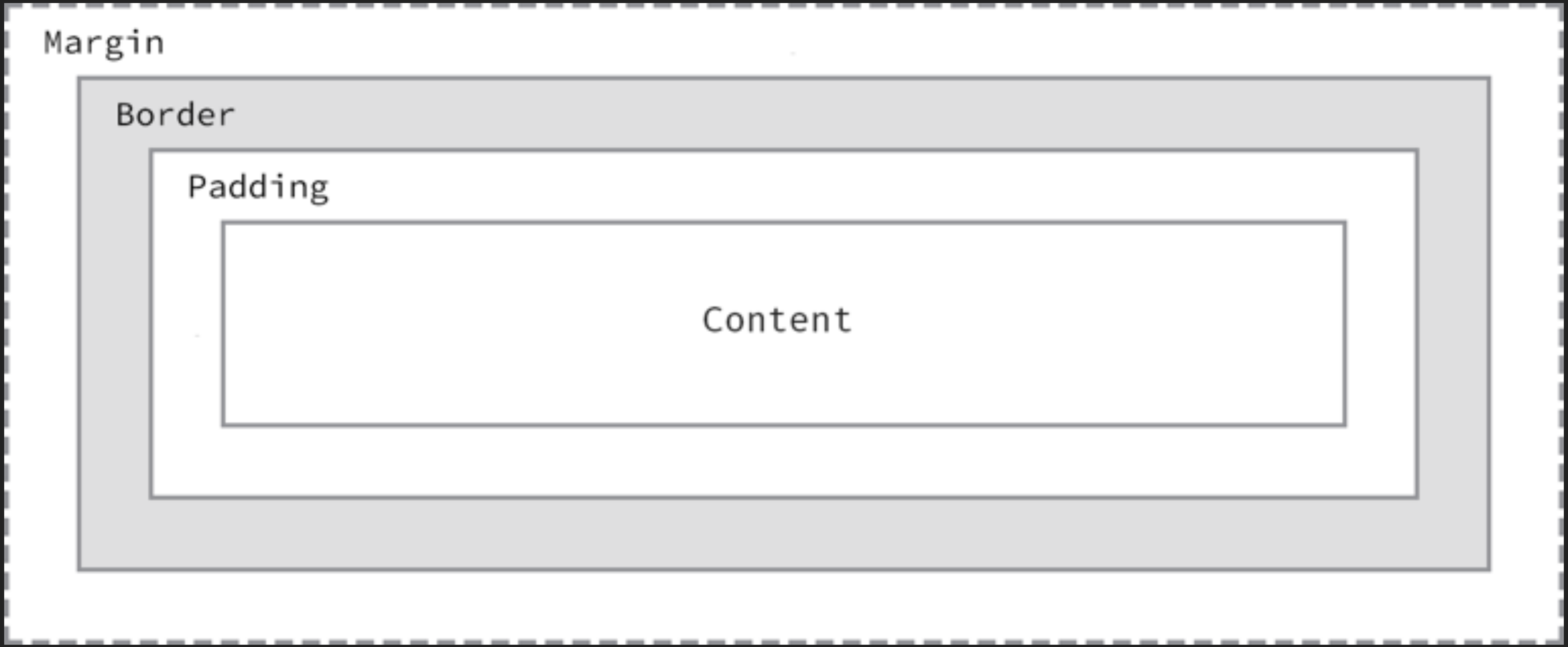
THE BOX MODEL

THE BOX MODEL

EVERY ELEMENT ON A PAGE IS A RECTANGULAR BOX AND MAY HAVE WIDTH, HEIGHT, PADDING, BORDERS, AND MARGINS.

http://learn.shayhowe.com/html-css/opening-the-box-model/

BOX MODEL EXAMPLE



BOX MODEL EXAMPLE

Total width = `margin-right` + `border-right` + `padding-right` + `width` + `padding-left` + `border-left` + `margin-left`

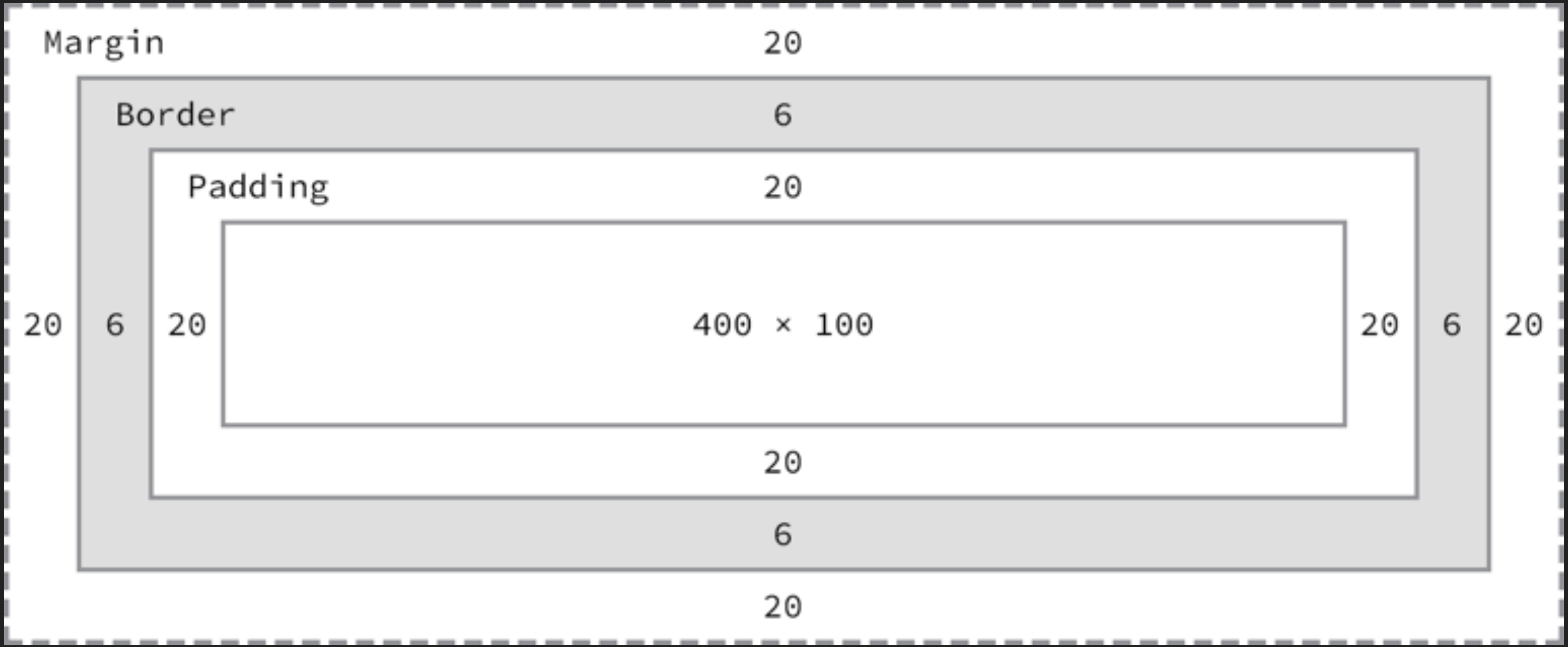
BOX MODEL EXAMPLE

Total height = `margin-top` + `border-top` + `padding-top` + `height` + `padding-bottom` + `border-bottom` + `margin-bottom`

BOX MODEL EXAMPLE

```
div {
  border: 6px solid #949599;
  height: 100px;
  margin: 20px;
  padding: 20px;
  width: 400px;
}
```

BOX MODEL EXAMPLE



So what is the total height and width of this box?

BOX MODEL EXAMPLE

Width: **492px** `= 20px + 6px + 20px + 400px + 20px + 6px + 20px`

Height: **192px** `= 20px + 6px + 20px + 100px + 20px + 6px + 20px`

MARGIN AND PADDING

- Margin - allows us to set the amount of space that surrounds an element. (ie. outside an elements border)
- Padding - allows us to set the amount of space inside an elements border (ie. between the border and the content).
- Some browsers apply default margins and/or padding on elements.

MARGIN AND PADDING DECLARATIONS

```
div {
  margin: 20px;
  padding: 5px;
}
```

All sides share same length

```
div {
  margin: 10px 20px;
  padding: 5px 10px;
}
```

Top/Bottom, Left/Right

```
div {
  margin: 10px 20px 0 15px;
  padding: 5px 10px 0 15px;
}
```

Top, Right, Bottom, Left

BORDERS

- Borders fall between the margin and padding.
- Borders require 3 properties - **`width`**, **`style`** and **`color`**.
- Examples of the most common styles are **`solid`**, **`double`**, **`dashed`**, **`dotted`** and **`none`**.

BORDER DECLARATION

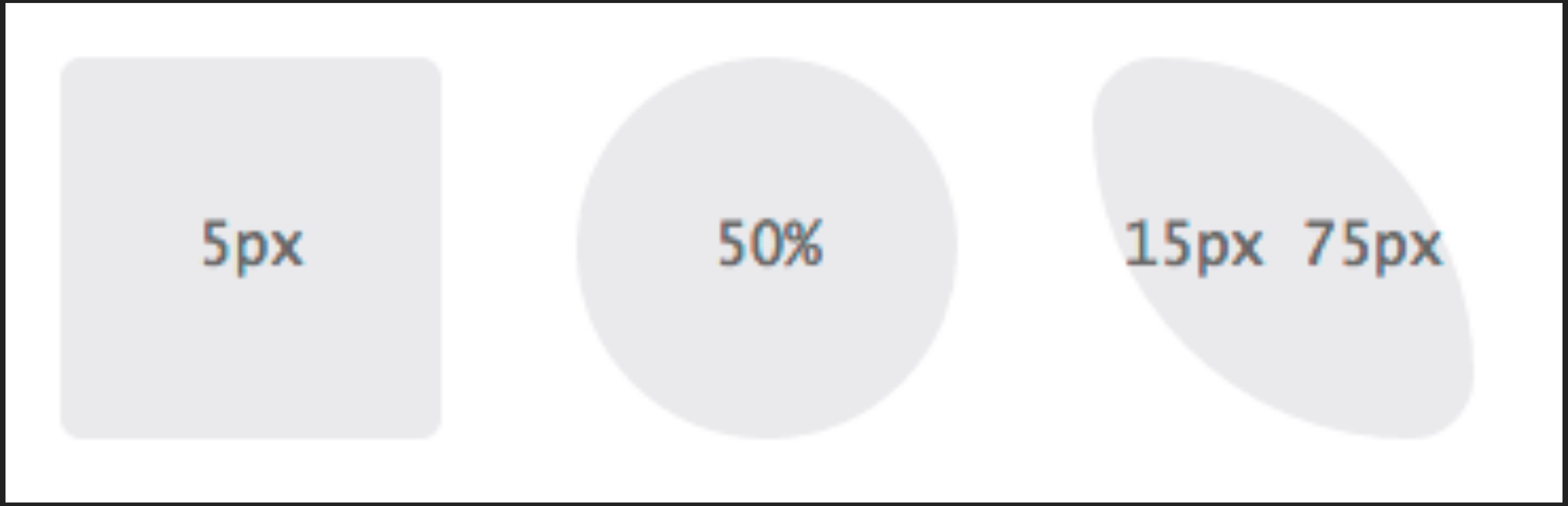
```css
div {
  border: 6px solid #949599;
}
```

You can also set individual borders, e.g. `border-right`, `border-left`, `border-top`, `border-bottom`.

Or properties like `border-top-width`, `border-top-style`, `border-top-color`.

BORDER RADIUS

- This enables rounded corners for an element.



EXAMPLE BORDER RADIUS

```css
div {
  border-radius: 5px;
}
```

A single value will round all four corners of an element equally

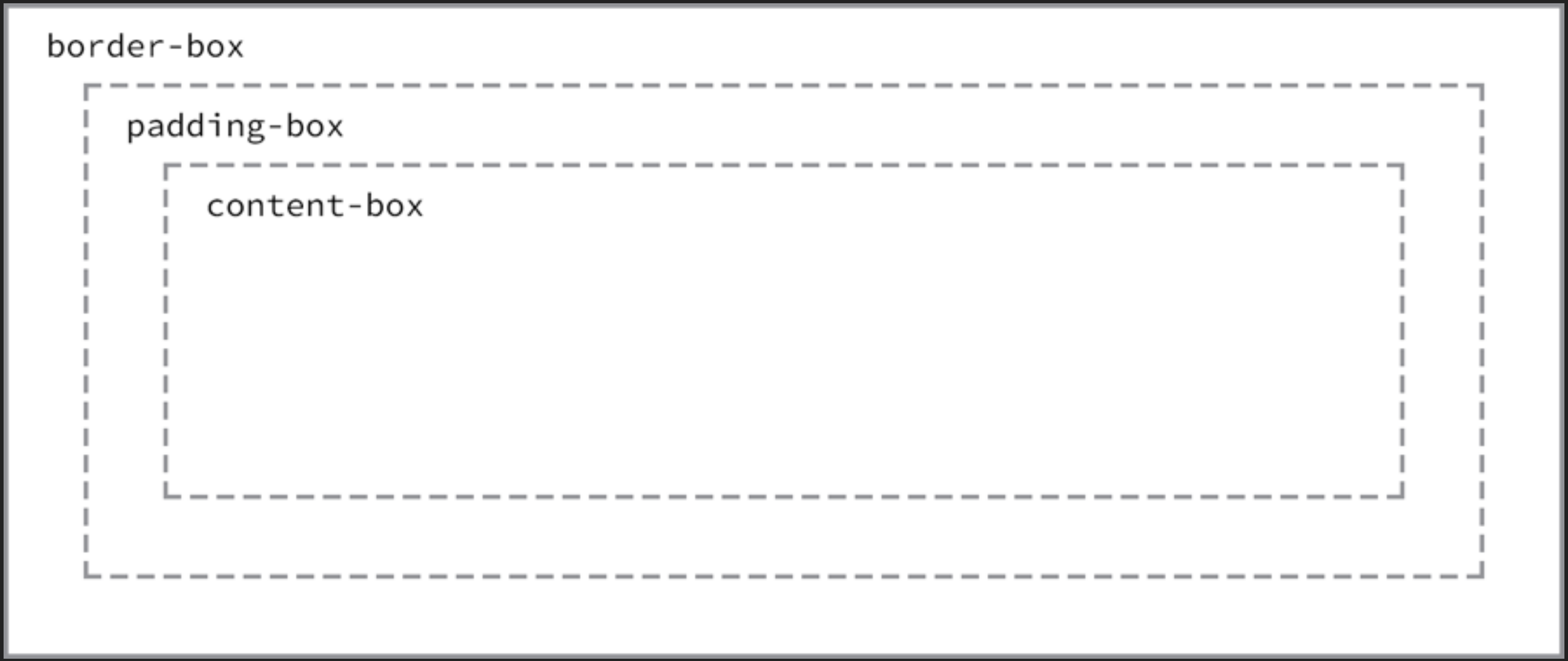
EXAMPLE OF BORDER RADIUS

```
div {
  border-top-right-radius: 5px;
}
```

You can also use `border-top-left-radius`, `border-bottom-right-radius`, `border-bottom-left-radius`

BOX SIZING

- The `box-sizing` CSS property allows us to change the way the box model is calculated.
- Allowed values are `content-box`, `padding-box` and `border-box`.
- `content-box` is the default.


border-box
padding-box
content-box

EXAMPLE OF BOX SIZING

```
div {
  -webkit-box-sizing: content-box;
     -moz-box-sizing: content-box;
          box-sizing: content-box;
}
```

What are those hyphens and letters before the property?

VENDOR PREFIXES

- As CSS3 was being introduced, browsers gradually began to support the new properties and values proposed as part of the specification.
- They were able to make these available to developers before the spec was finalized using vendor prefixes.
- As the CSS3 spec becomes finalized vendor prefixes will become less relevant.

LAYOUTS AND POSITIONING

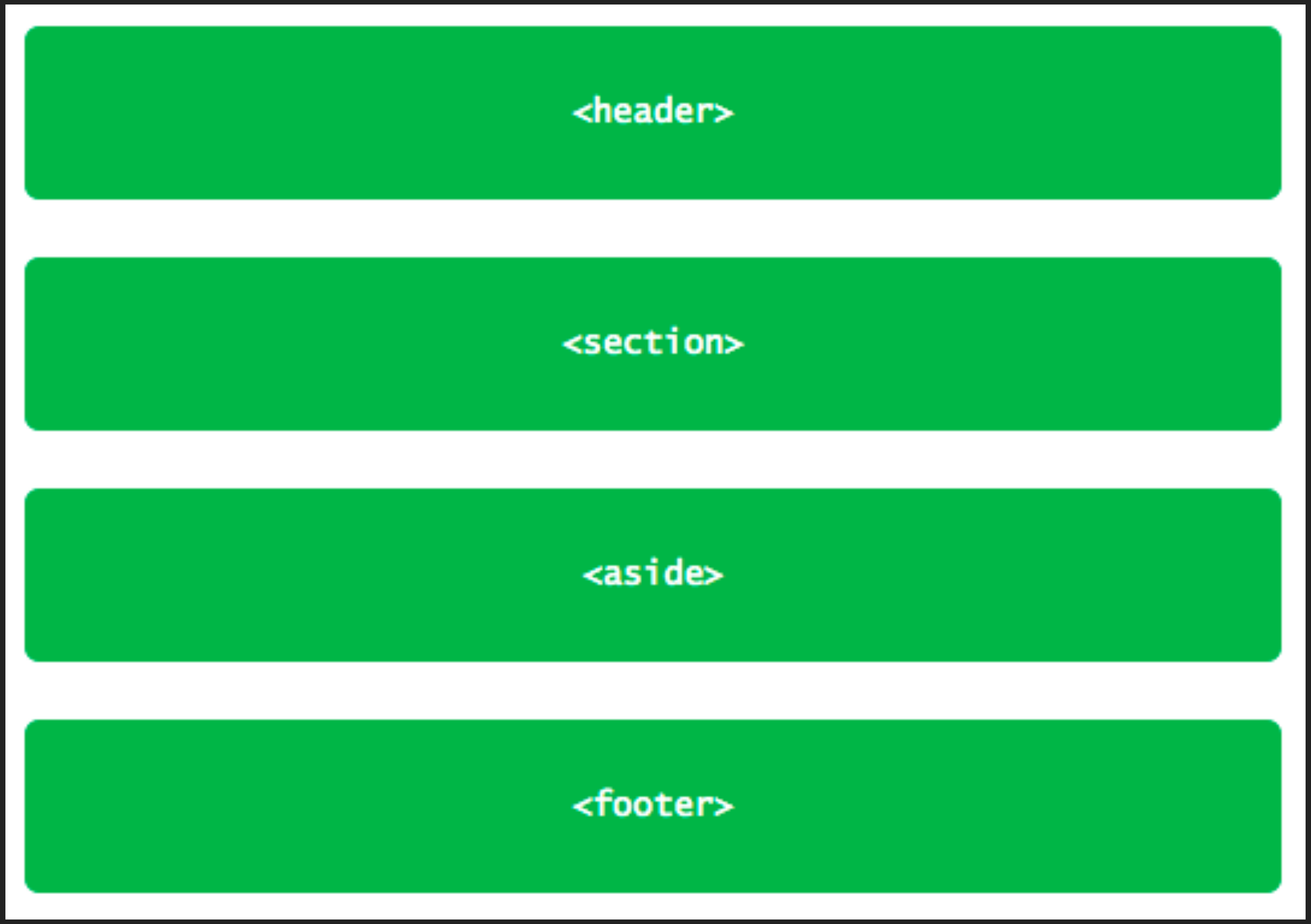
WAYS TO POSITION ELEMENTS

- Floats
- Uniquely Positioning Elements
    - Relative Positioning
    - Absolute Positioning

NORMAL FLOW

```
<header> ... </header>
<section> ... </section>
<aside> ... </aside>
<footer> ... </footer>
```

NORMAL FLOW



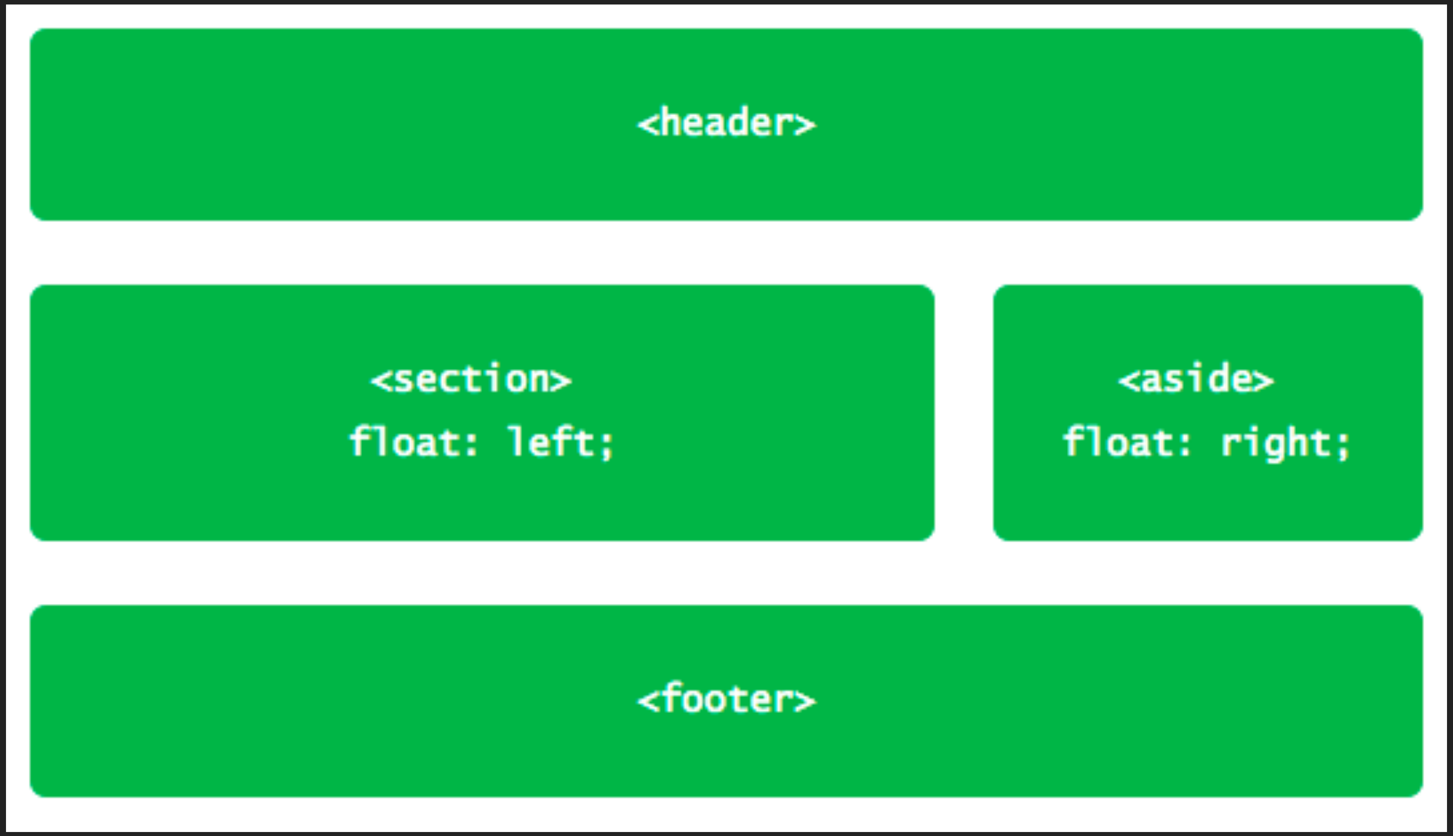
FLOATS

- Allows us to take an element, remove it from the normal flow of a page, and position it to the left or right of its parent element.
- The `float` property accepts a few values, the two most popular ones are `left` and `right`.
- An example could be floating an `<img>` element to the side so that paragraphs of text wrap around it.
- You can also float multiple elements to create a layout.

FLOATS

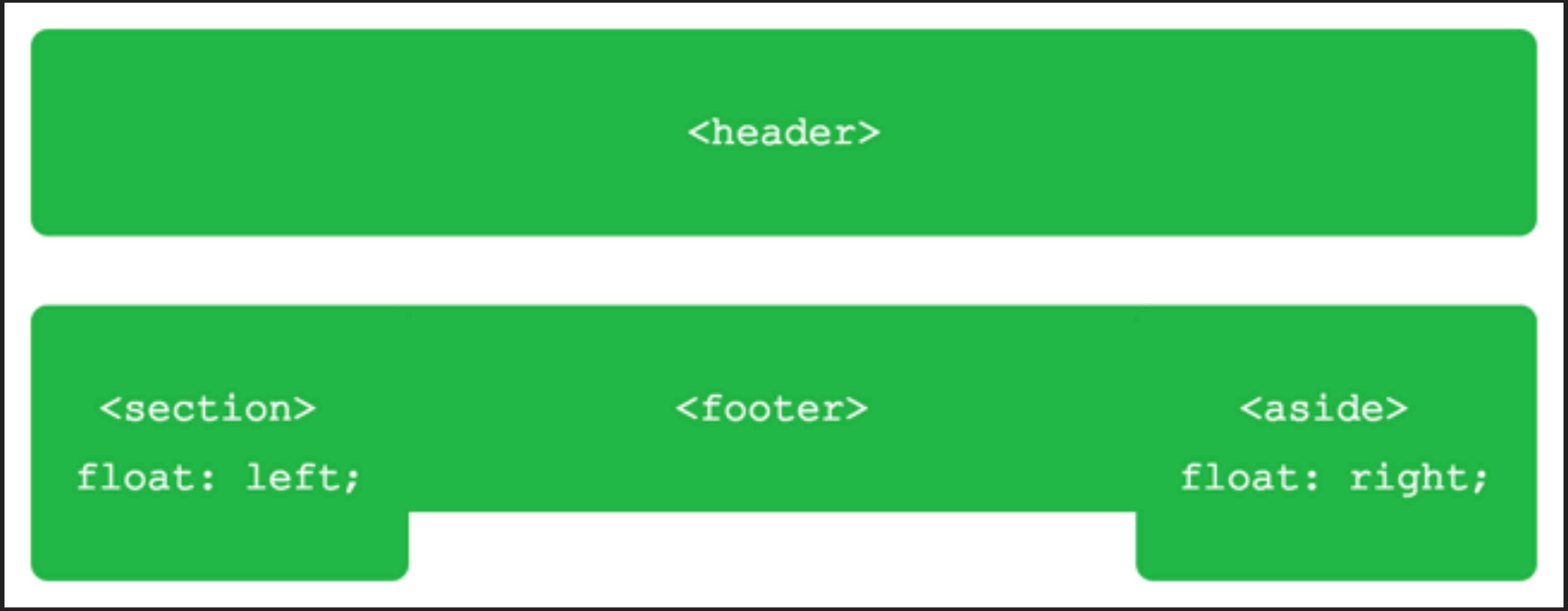
```css
section {
  float: left;
  margin: 0 1.5%;
  width: 63%;
}
aside {
  float: right;
  margin: 0 1.5%;
  width: 30%;
}
```

FLOATS



CLEARING FLOATS

- Sometimes if you are not careful when using floats, you can end up with elements unnecessarily wrapping around a floated element or filling in the available space since it is no longer in the normal flow.



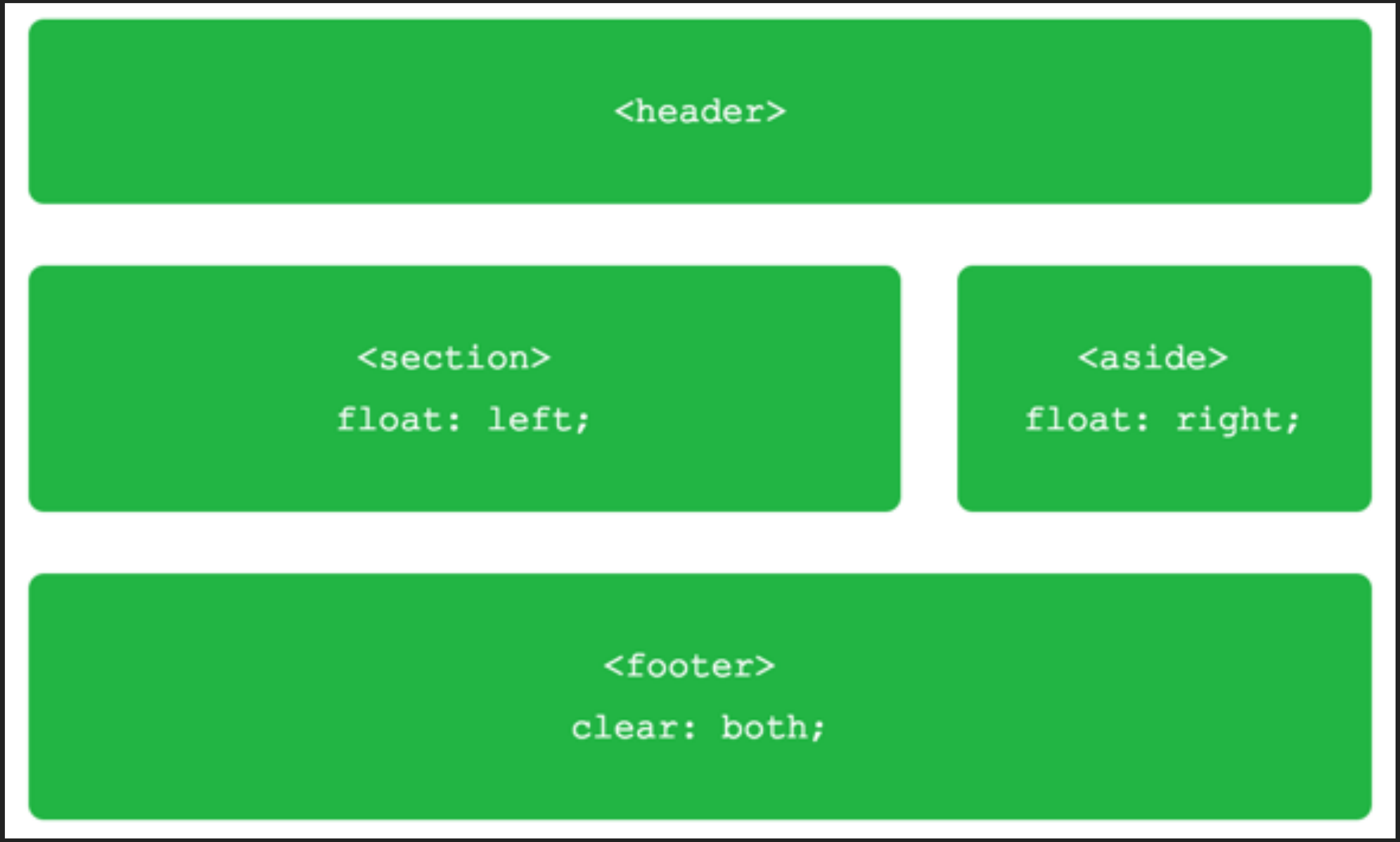
CLEARING FLOATS

- To prevent content from wrapping around floated elements, we need to clear, or contain, those floats and return the page to its normal flow.
- We can do this by using the `clear` property.
- This property accepts a few different values: the most commonly used values being `left`, `right`, and `both`.
- The `left` value will clear left floats, while the `right` value will clear right floats. The `both` value, however, will clear both left and right floats and is often the most ideal value.

CLEARING FLOATS

- So using our previous example. We can apply the following:

```
footer {
  clear: both;
}
```



UNIQUELY POSITIONING ELEMENTS

- There are times we need to precisely position an element. In cases like this we use the `position` property.
- The default position is `static` (normal flow), however, this value can be overwritten with `relative` or `absolute`.
- These work along with the box offset properties `top`, `right`, `bottom` and `left`.

RELATIVE POSITIONING

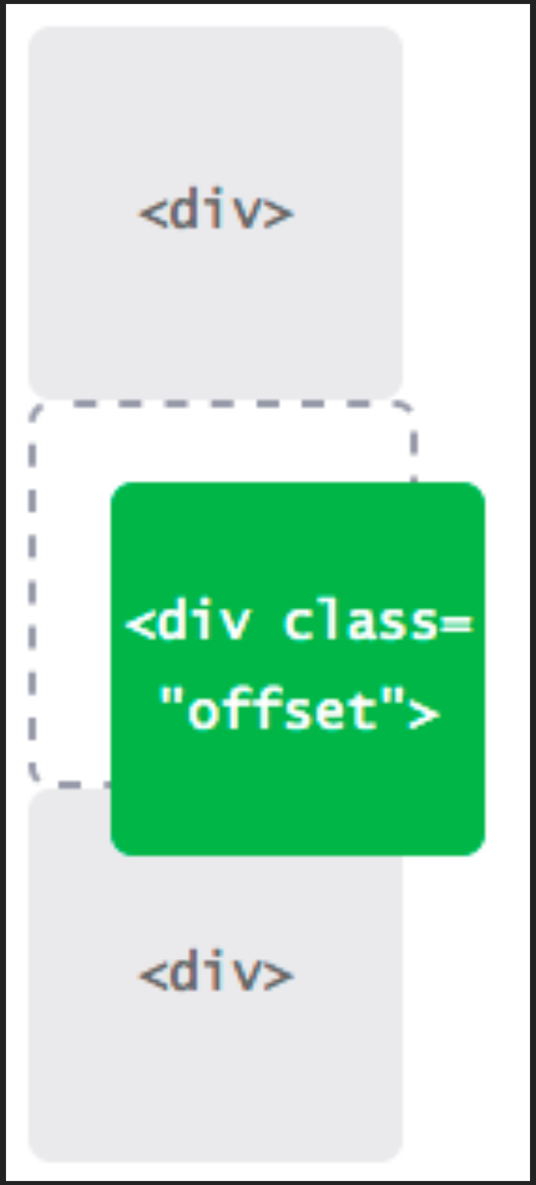
- Allows us to move an element, but keep it in the normal flow of a page, thus preventing other elements from flowing around it.

EXAMPLE OF RELATIVE POSITIONING

```
<div> … </div>
<div class="offset"> … </div>
<div> … </div>
```

```
div {
  height: 100px;
  width: 100px;
}
.offset {
  left: 20px;
  position: relative;
  top: 20px;
}
```

EXAMPLE OF RELATIVE POSITIONING



ABSOLUTE POSITIONING

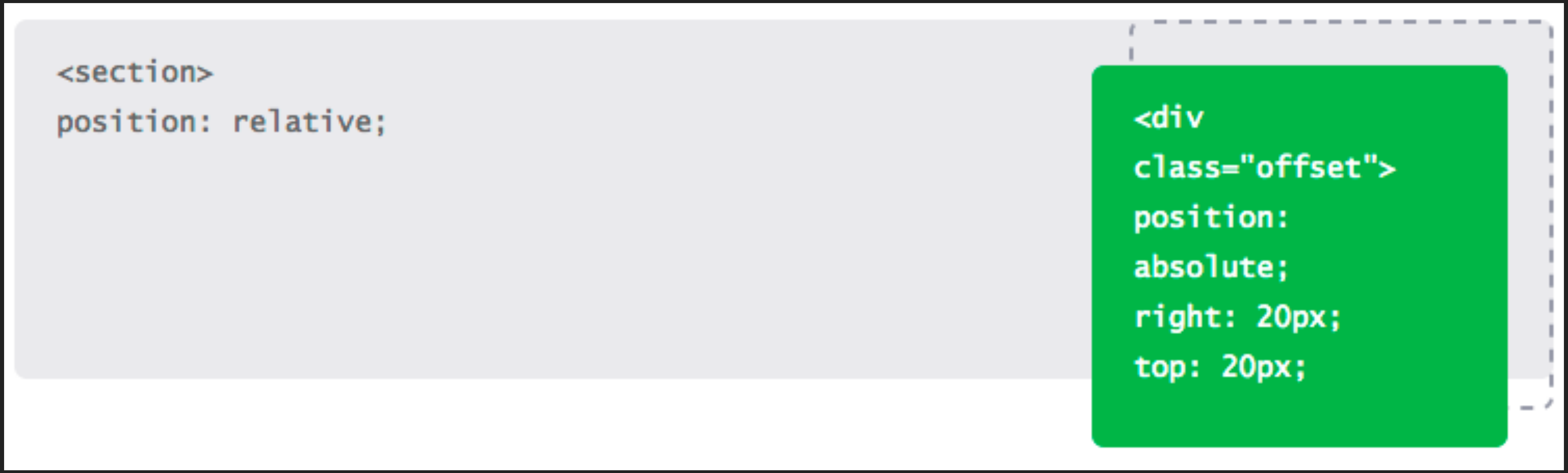
- Similar to the `relative` value for the `position` property, with the exception that the element will not appear in the normal flow of the document and the space it occupied will not be preserved.
- It is also moved in relation to its closest relatively positioned element.

EXAMPLE OF ABSOLUTE POSITIONING

```
<section>
  <div class="offset"> … </div>
</section>
```

```
section {
  position: relative;
}
.offset {
  right: 20px;
  position: absolute;
  top: 20px;
}
```

EXAMPLE OF ABSOLUTE POSITIONING



RESOURCES TO LEARN MORE

- http://learn.shayhowe.com/html-css/
- http://learnlayout.com/
- http://webtypography.net/
- https://developer.mozilla.org/en-US/
- http://cssspecificity.com/

ANY QUESTIONS?