

# HTML



# CSS



## HTML & CSS: LEVEL 1

Instructor: Beck Johnson

Week 4



# SESSION OVERVIEW

- Review box model of CSS
- Classes and ids
- Floats!
- Using web fonts



**REVIEW!**

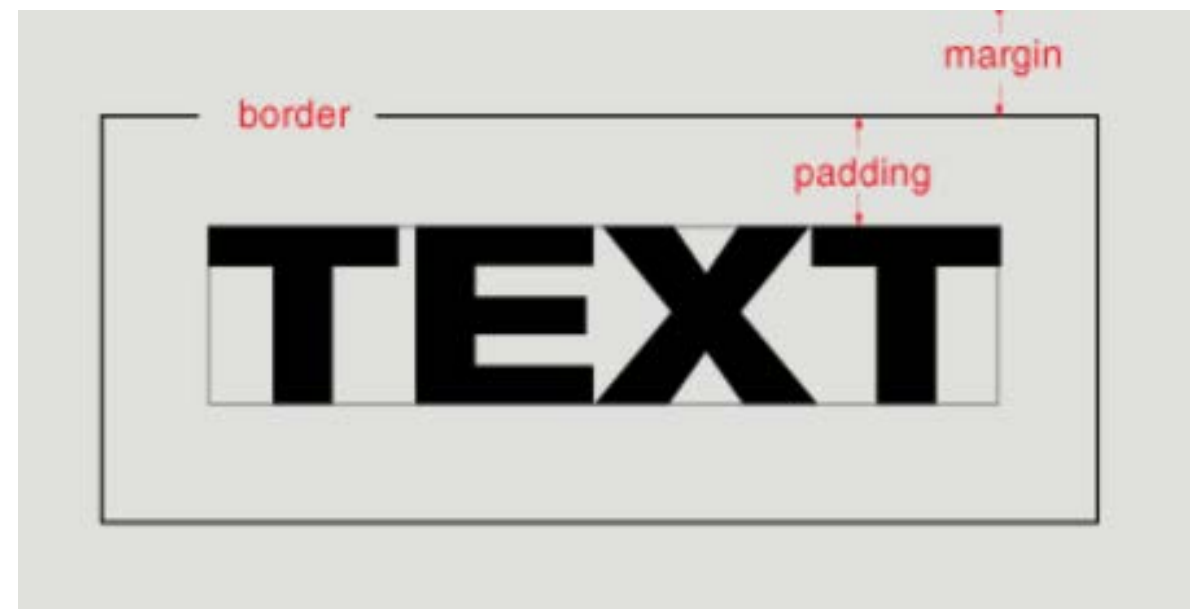
# { CSS BOX MODEL

**CONTENT:** stuff in the box

**PADDING:** bubble wrap and packing peanuts

**BORDER:** sides of the box

**MARGIN:** space between multiple boxes



# { PADDING

- Padding creates space **inside** an element
- Padding affects how far content is from the border

*Padding* is the space inside the border, between the border and the actual image or text.

# { MARGIN

- Margin creates space **outside** an element.
- Margin affects how far elements are from each other

*Margin* is the space between one object and its surrounding elements.

# { BORDER

Between margin and padding, you can set a **border**

- Width (usually in pixels)
- Border style (solid, dotted, dashed, etc)
- Color

p

**border:** 2px dotted #ff0000;

}

# { BORDER-RADIUS

To make an element appear curved, use the property **border-radius**

- The value is a number (in px or em) or percentage
- You can use **border-radius** even if you don't explicitly set a **border**

```
li {  
    border-radius: 50%;  
    height: 3em;  
    width: 3em;  
}
```





# { BLOCK ELEMENTS

## BLOCK ELEMENTS

- Expand naturally to fill their parent container
- Can have margin and/or padding

**BLOCK ELEMENTS EXPAND NATURALLY**



**AND NATURALLY DROP BELOW OTHER ELEMENTS**



# { INLINE ELEMENTS

## INLINE ELEMENTS

- Flow along with text content
- Ignores height, width, top margin, and bottom margin
- Honors left and right margins (and any padding)

### INLINE ELEMENTS FLOW WITH TEXT

PELLENTE SQUE HABITANT MORBI TRISTIQUE SENECTUS  
ET NETUS ET MALESUADA FAMES AC TURPIS EGESTAS.  
VESTIBULUM **INLINE ELEMENT** VITAE, ULTRICIES  
EGET, TEMPOR SIT AMET, ANTE. DONEC EU LIBERO SIT  
AMET QUAM EGESTAS SEMPER. AENEAN ULTRICIES MI  
VITAE EST. MAURIS PLACERAT ELEIFEND LEO.

# { INLINE-BLOCK ELEMENTS

## INLINE-BLOCK ELEMENTS

- A hybrid of block and inline
- Flow along with text content
- Has height, width, margin, and padding



# <SPAN> ELEMENTS

<span></span>

A <span> is a **generic inline element**

- No default style
- Used to style inline content

# <DIV> ELEMENTS

<div></div>

A <div> is a **generic block element**

- No default style
- Heavily used as a wrapper for other elements, to create complex layouts

**QUESTIONS?**



# ID & CLASS SELECTORS

# CLASSES AND IDS

CSS lets us target **all** paragraphs like this:

```
p {  
    font-size: 20px;  
}
```

But what if we want to style only **some** paragraphs?



# CLASSES AND IDS

You can add **class** and **id** to any HTML element to identify it for styling.

- You decide the **class** and **id** values – be descriptive!

```
<p class="important">Big text</p>
```

```
<p class="anyLettersOrNumbersOr_Or-">Still  
totally valid</p>
```

# CLASSES AND IDS

Adding a **class** or **id** does nothing to an element by default.

- Classes and ids don't have any styling information by themselves
- They require you to add CSS selectors if you want styling to be applied

# CLASSES AND IDS

**Multiple** elements can have the same **class**

- A class is like a barcode – all of the same products have the same barcode



Only **one** element per page can use the same **id**

- An id is like a serial number – it uniquely identifies one specific instance of a product



# CLASS SELECTORS IN CSS

- In CSS, target a **class** with a **period**
- Will style **all** types of elements that have that **class**:

```
.kittens { color: gray; }
```

```
<p class="kittens">This will be gray.</p>
```

```
<div class="kittens">This will be gray too.</div>
```

# CLASS ATTRIBUTES

Elements can have **multiple** classes, separated by a space:

```
<p class="important margin-sm">Big text with a  
margin</p>
```

```
.important { font-size: 20px; }
```

```
.margin-sm { margin: 5px; }
```

# CLASS SELECTORS IN CSS

Child selectors work with classes:

```
.card p { padding: 16px; }
```

“Any paragraph that is **inside** an element with a **class** of **card** gets 16px of padding.”

```
<div class="card">  
  <h2>This will not get padding</h2>  
  <p>But this will</p>  
</div>
```

# ID ATTRIBUTES

- An **id** can only be used **once** per page
- Elements **cannot** have multiple **id** attributes

```
<div id="mainContent">  
    <!-- This better be the only main -->  
</div>
```

# ID SELECTORS IN CSS

In CSS, target an id with a **hash**:

```
#kittenContainer {  
    color: gray;  
}
```

```
<div id="kittenContainer"></div>
```



# IDS FOR ANCHORING

If you put a hash followed by the element's **id** in the URL, the browser will **jump** to that location on the same page:

```
<a href="#kittenContainer">Proceed  
directly to kittens</a>
```

# ID ATTRIBUTES

**Q:** What horrible thing will happen if you use an **id** twice on the same page?

**A:** Well...actually nothing.

- But your page won't validate
- Jump links will go to whatever **id** appears first
- And any Javascript that needs to locate that specific element will fail

# MIXING CLASS AND ID ATTRIBUTES

An element can have both **id** and **class** attributes.

```
<div id="puppyContainer" class="small fluffy"></div>
```

```
<div id="birdContainer" class="small feathery"></div>
```

# HOW TO CHOOSE - CLASS OR ID?

If you think it's likely or possible that you'll want to apply the same style to multiple things, definitely use **class**

If your element is guaranteed to be the only one on the page, you can use **id** – or you can still use **class**

If your element needs to be linked to directly, use **id**

# CLASS COMPONENTS

The most common use of classes is to define reusable components.



Located two hours south of Sydney in the Southern Highland of New South Wales...

[SHARE](#)

[LEARN MORE](#)



Largest monolith (geological feature consisting of a single massive stone or rock) in the world.

[SHARE](#)

[LEARN MORE](#)

# CLASS COMPONENTS

A component is an outline defined in HTML that will have the same markup every time it's used, but with different content inside it.

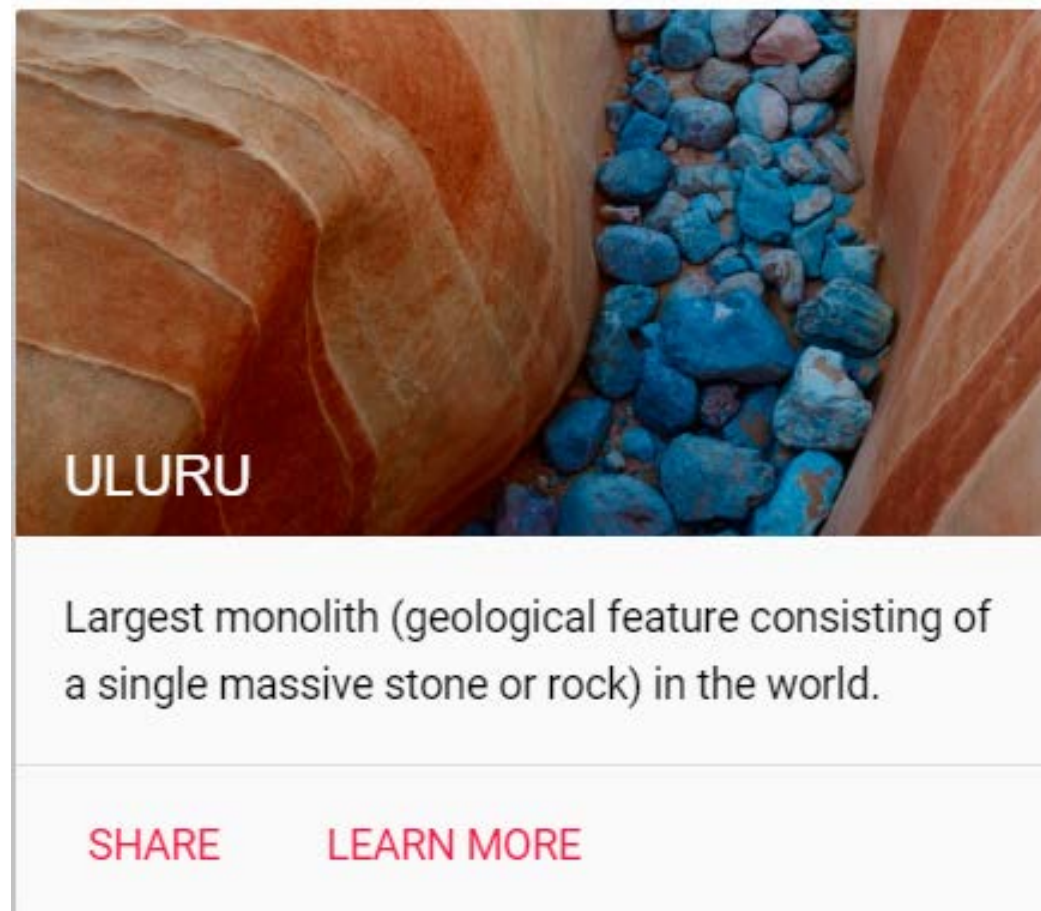
```
<div class="card">

  <div class="image">
    <h2></h2>
  </div>

  <p></p>

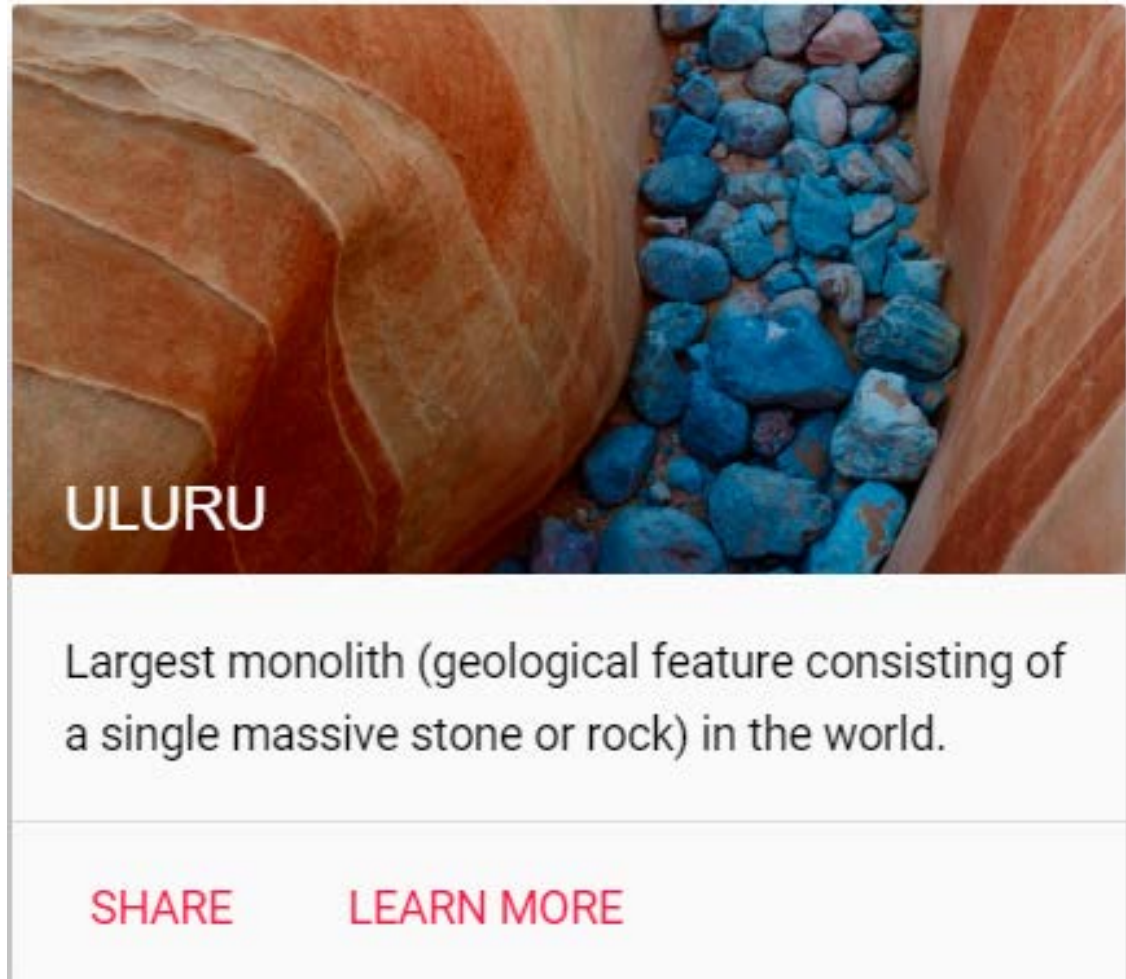
  <div class="action-bar">
    <a></a>
    <a></a>
  </div>

</div>
```



# CHILDREN IN CLASS

```
.card {  
  display: inline-block;  
  width: 344px;  
  height: 304px;  
}  
  
.card p {  
  padding: 16px;  
  margin: 0;  
  line-height: 1.6;  
}  
  
.card .action-bar {  
  padding: 0 8px;  
  border-top: 1px solid #E0E0E0;  
  height: 52px;  
}
```



See a [live demo](#)



**PRACTICE TIME!**



# ASSIGNMENT

Give an element on your page a descriptive **class**

- Apply a special style using a CSS **class** selector
- Style a child element of this element

Create another class and apply it to **two different** types of elements

- Bonus points: apply to an element that already has a class. What happens if the styles conflict? How would you make sure the result is what you want?

Assign an **id** to an element on your page

- Apply a unique style using an **id** selector
- Create a link in your nav that jumps to that element



# WEB LAYOUTS

# WEB LAYOUTS

With CSS, we can use a variety of properties to arrange elements on the screen by adjusting the flow of the page.

Basically, you can put elements anywhere...which can be both a good and a bad thing!

# 3 WEB LAYOUT PROPERTIES

- **display:** dictates how elements behave within the box model
- **float:** moves elements around within the page flow
- **position:** takes elements entirely out of the page flow

# DISPLAY PROPERTY

The `display` property tells the browser what type of box model to use:

- `inline`
- `inline-block`
- `block`

This changes how padding, margin, height and width affect an element.

You also can set `display: none` to hide an element entirely.

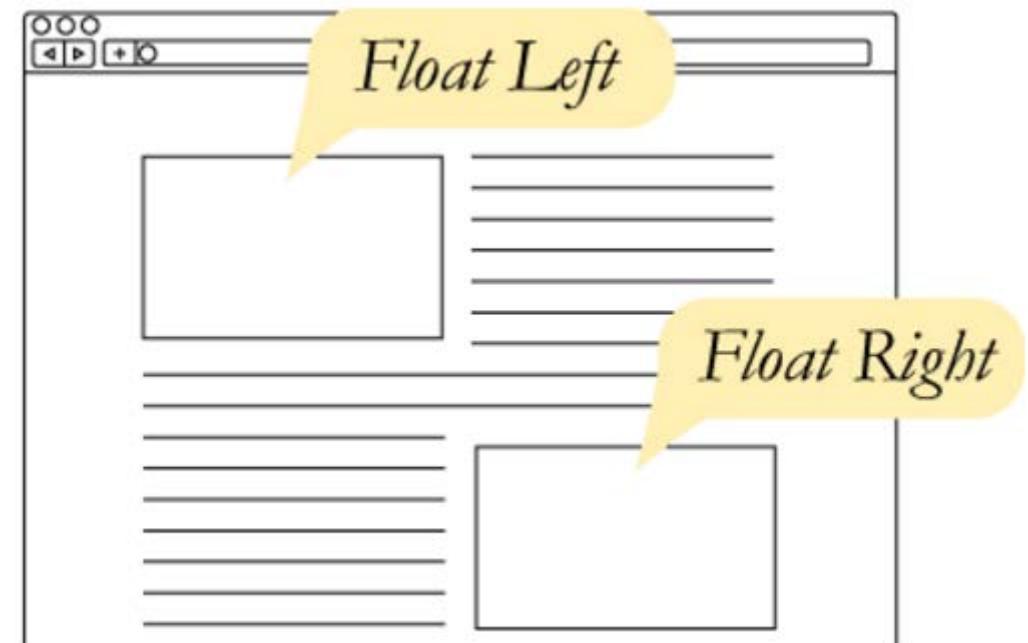
# CSS FLOATS

Up to now, elements have displayed sequentially, in the order that you placed them in your HTML.

The **float** property takes an element out of the normal flow and “floats” it to the left or right side of its container.

- This allows other content to flow around it

```
img { float: left; }
```



# CSS FLOATS

The three values for `float` are:

- `left`
- `right`
- `none`

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo sequi veniam ea enim nesciunt doloremque delectus sint consectetur qui magnam. Recusandae, hic quidem officia, asperiores sit libero sapiente totam eum.



By default, elements are  
`float: none`

Doloribus nisi ratione necessitatibus unde veritatis commodi veniam quas eaque fugiat nihil esse, id? Tempora quis quod impedit quia, facere incidunt, voluptatum dicta in dolores suscipit temporibus quam eos odit?

Doloribus nisi ratione necessitatibus unde veritatis commodi veniam quas eaque fugiat nihil esse, id? Tempora quis quod impedit quia, facere incidunt, voluptatum dicta in dolores suscipit temporibus quam eos odit?

# CSS FLOATS

```
img { float: right; }
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo sequi veniam ea enim nesciunt doloremque delectus sint consectetur qui magnam. Recusandae, hic quidem officia, asperiores sit libero sapiente totam eum.

Doloribus nisi ratione necessitatibus unde veritatis commodi veniam quas eaque fugiat nihil esse, id? Tempora quis quod impedit quia, facere incidunt, voluptatum dicta in dolores suscipit temporibus quam eos odit?

Doloribus nisi ratione necessitatibus unde veritatis commodi veniam quas eaque fugiat nihil esse, id? Tempora quis quod impedit quia, facere incidunt, voluptatum dicta in dolores suscipit temporibus quam eos odit?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo sequi veniam ea enim nesciunt doloremque delectus sint consectetur qui magnam. Recusandae, hic quidem officia, asperiores sit libero sapiente totam eum.





# CSS FLOATS

```
img { float: left; }
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo sequi veniam ea enim nesciunt doloremque delectus sint consectetur qui magnam. Recusandae, hic quidem officia, asperiores sit libero sapiente totam eum.



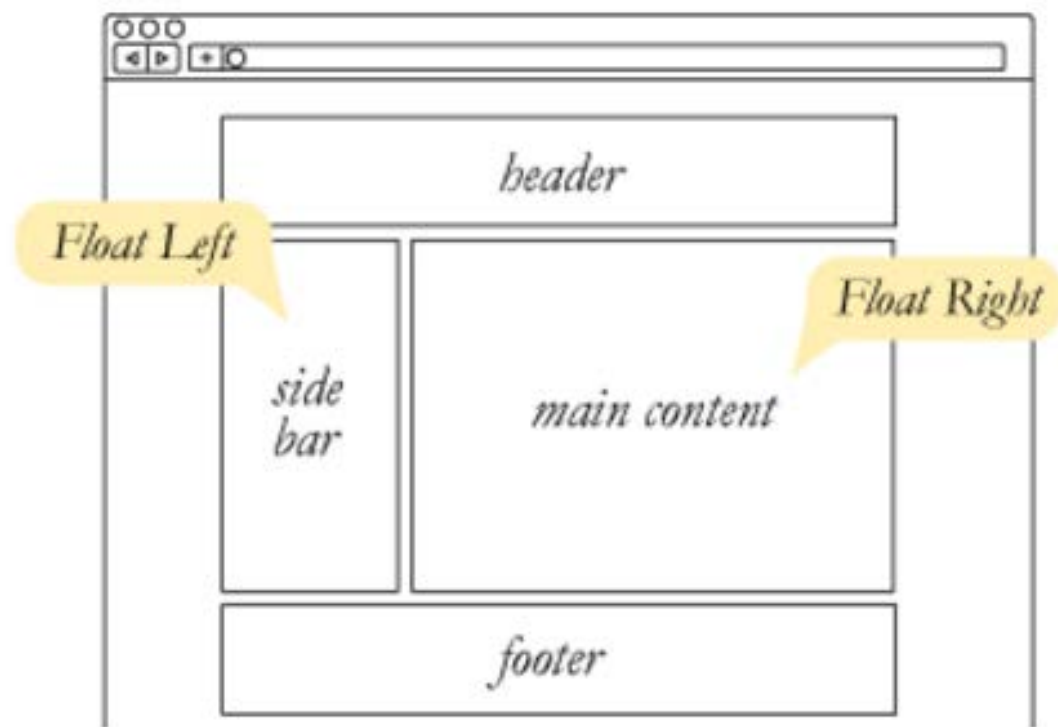
Doloribus nisi ratione necessitatibus unde veritatis commodi veniam quas eaque fugiat nihil esse, id? Tempora quis quod impedit quia, facere incidunt, voluptatum dicta in dolores suscipit temporibus quam eos odit?

Doloribus nisi ratione necessitatibus unde veritatis commodi veniam quas eaque fugiat nihil esse, id? Tempora quis quod impedit quia, facere incidunt, voluptatum dicta in dolores suscipit temporibus quam eos odit?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo sequi veniam ea enim nesciunt doloremque delectus sint consectetur qui magnam. Recusandae, hic quidem officia, asperiores sit libero sapiente totam eum.

# CSS FLOATS

**float** can be used to float text around images, but it also can be used to create entire page layouts.



# CSS FLOATS

For example, this layout was built using **float**.  
How do you think it was done?



## BY BECK JOHNSON

Beck Johnson is a developer based in Seattle, WA. She plays board games in her spare time. Say hi at [beckjohnson@gmail.com](mailto:beckjohnson@gmail.com)

---

# CSS FLOATS



```
.avatar { float: left; }      .bio { float: right; }
```

Let's try that...

# CSS FLOATS



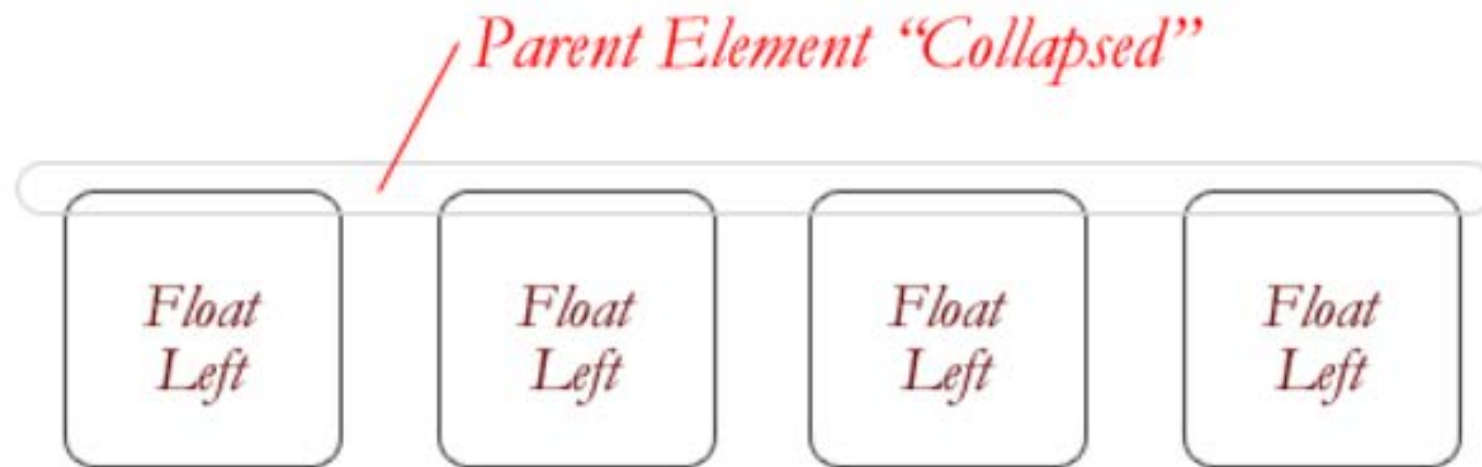
## BY BECK JOHNSON

Beck Johnson is a developer based in Seattle, WA. She plays board games in her spare time. Say hi at [beckjohnson@gmail.com](mailto:beckjohnson@gmail.com)

The container thinks it has no content!

- It collapsed to the size of its padding (you can see the top and bottom **border**)
- The floated content is spilling out

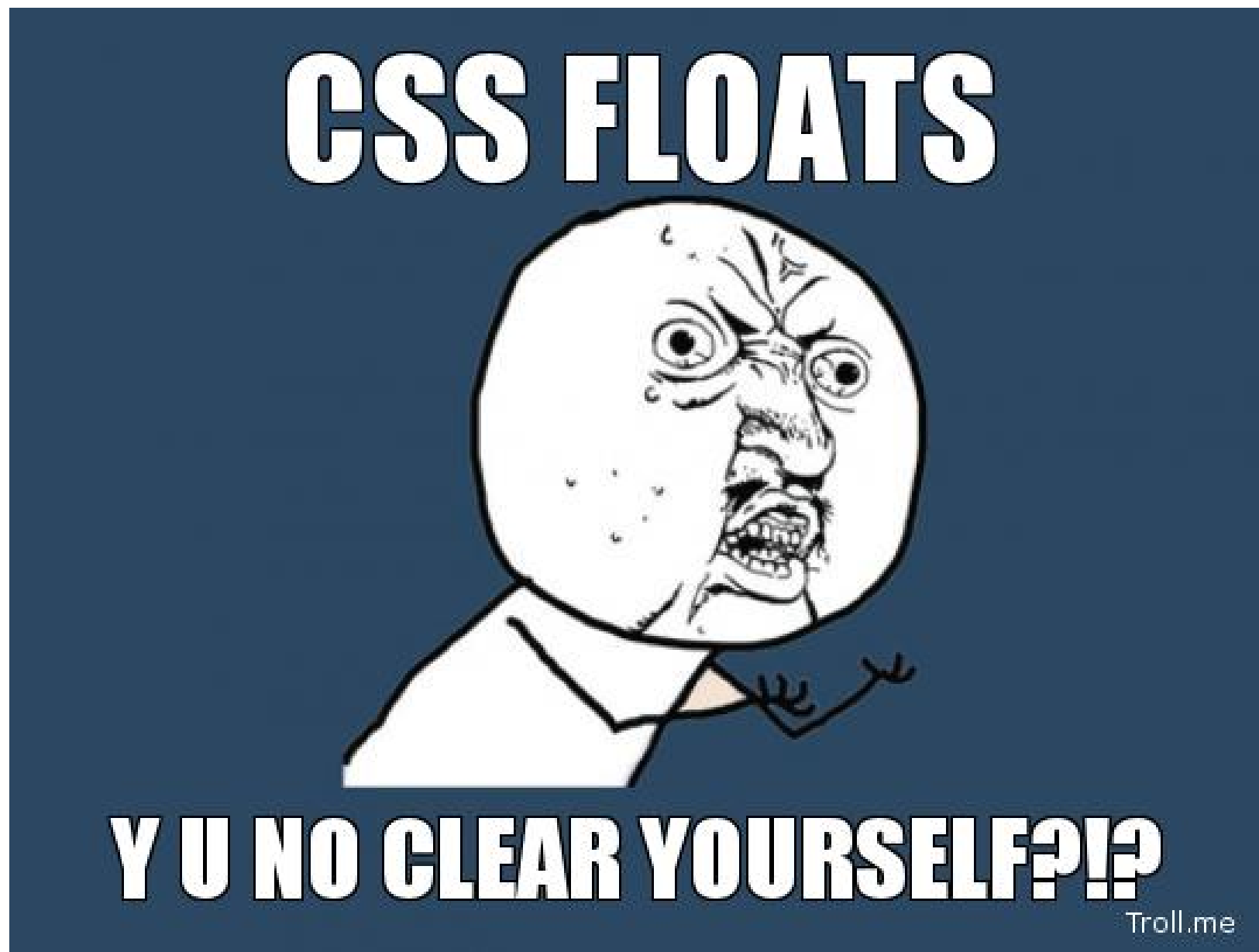
# CSS FLOATS



If you float an element, it is no longer in the normal document flow.

If all elements in a container are floated, that means that the container is effectively “empty.”

# HOW TO FIX FLOATS?



# HOW TO FIX FLOATS?

There are 2 ways to fix this:

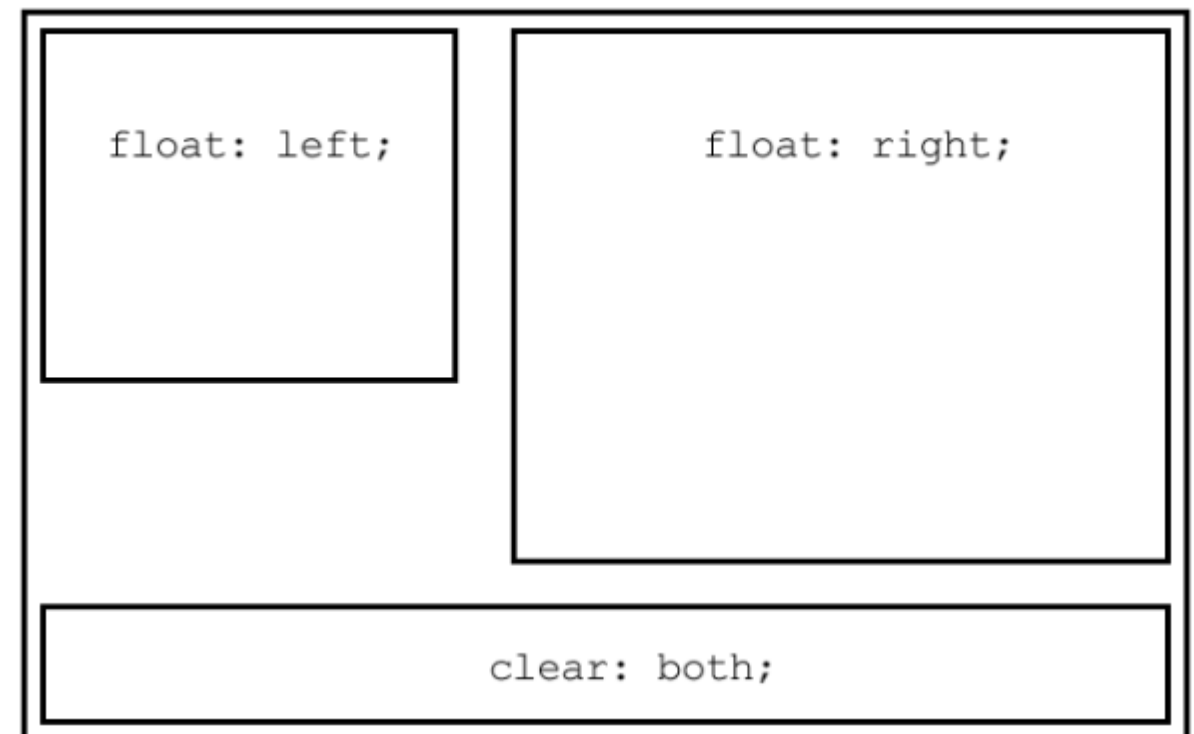
1. Apply the CSS rule `clear: both` to an element **after** the floated content
2. Apply a CSS rule using the property `overflow` to the **container**



# THE CLEAR PROPERTY

The **clear** property is the sister property to **float**

- It doesn't do much until there are floated elements on the page
- An element with **clear** applied to it will force itself **below** the floated element
- Everything after that will be back in the normal flow
- This “stretches” out the container and keeps it from collapsing



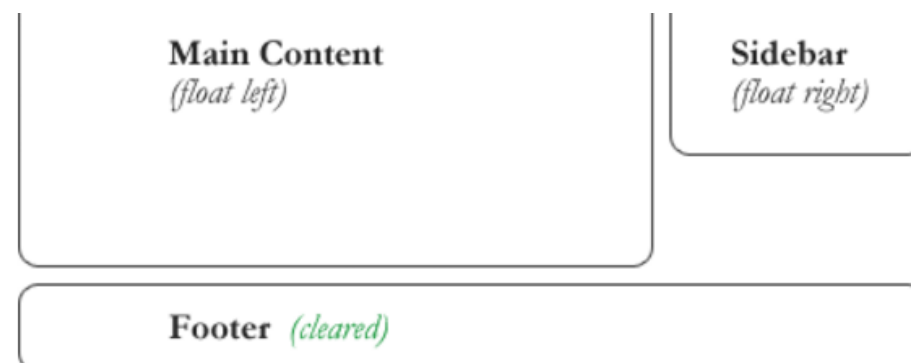
# THE CLEAR PROPERTY

**clear** has similar values to **float**:

- **clear: none** – the element does **not** move down to clear past floating elements (this is the default value)



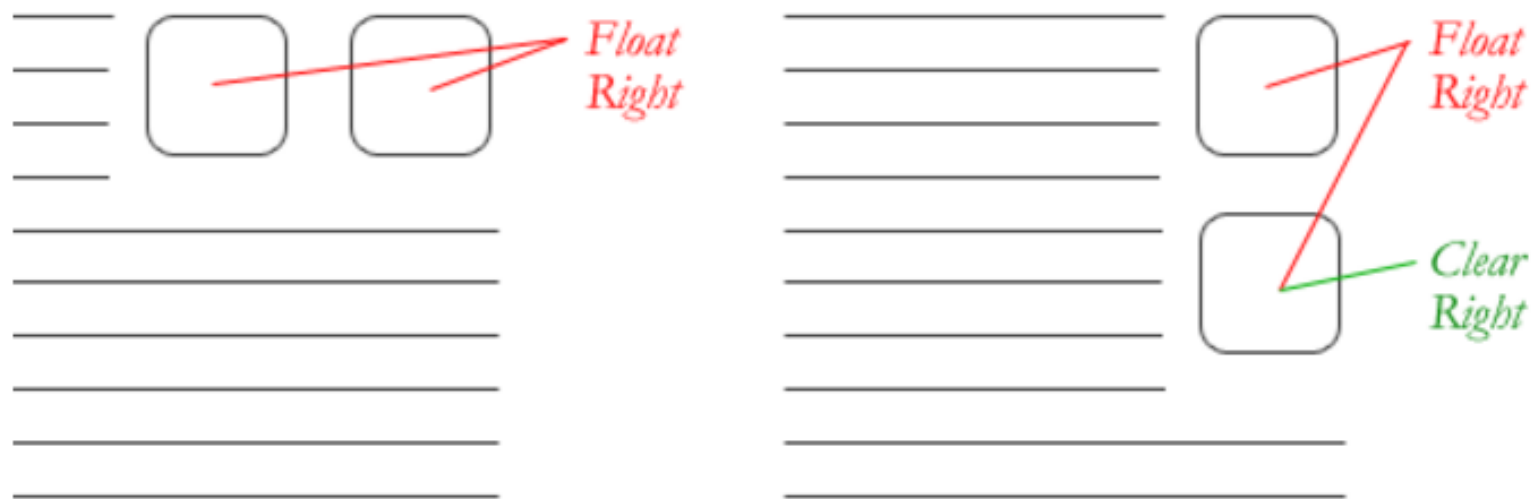
- **clear: both** – the element is moved down to clear **both** right- and left-floated elements



# THE CLEAR PROPERTY

Sometimes, you want to let some content after a **clear** continue floating, but not other content:

- **clear: left** – only clear **left-floated** elements
- **clear: right** – only clear **right-floated** elements



# THE CLEAR PROPERTY

So to solve our problem, you could add this empty `div` after the bio container:



**BY BECK JOHNSON**

Beck Johnson is a developer based in Seattle, WA. She plays board games in her spare time. Say hi at [beckjohnson@gmail.com](mailto:beckjohnson@gmail.com)

```
<div style="clear: both"></div>
```

(We could apply the class to any type of element, but the benefit of using a `div` is that it has no style of its own.)

# THE MAGIC FLOAT FIX

The preferred solution is to use a class that automatically clears itself.

- Apply a class to the parent container called `clearfix`
- Create a CSS rule for `clearfix` class using the **pseudo-element** `:after`

```
.clearfix:after {  
    content: "";  
    display: block;  
    clear: both;  
}
```

# THE MAGIC FLOAT FIX

The **pseudo-element** `:after` inserts a tiny bit of content (specified by the `content` property) after the parent element

- In this case, the content is empty (`""`), but it's sufficient to trigger the `clear: both` rule

(There is also a **pseudo-element** `:before` that inserts `content` before the parent element – you'll see this used when we look at icon fonts!)

# THE OVERFLOW PROPERTY

The other way to force a container to expand around floated content is to apply a CSS rule with **overflow** to the container that the floated content is inside.

**Any** valid value for **overflow** will cause floated content to stretch out the container

- Too complicated to explain, but it basically forces the container to re-assess the content inside it

# THE OVERFLOW PROPERTY

**overflow** is a CSS property that governs how content looks when it breaks out of its container.

By default, elements have **overflow: visible**, which means all content is fully visible

- Even if that means overflowing its container!





# THE OVERFLOW PROPERTY

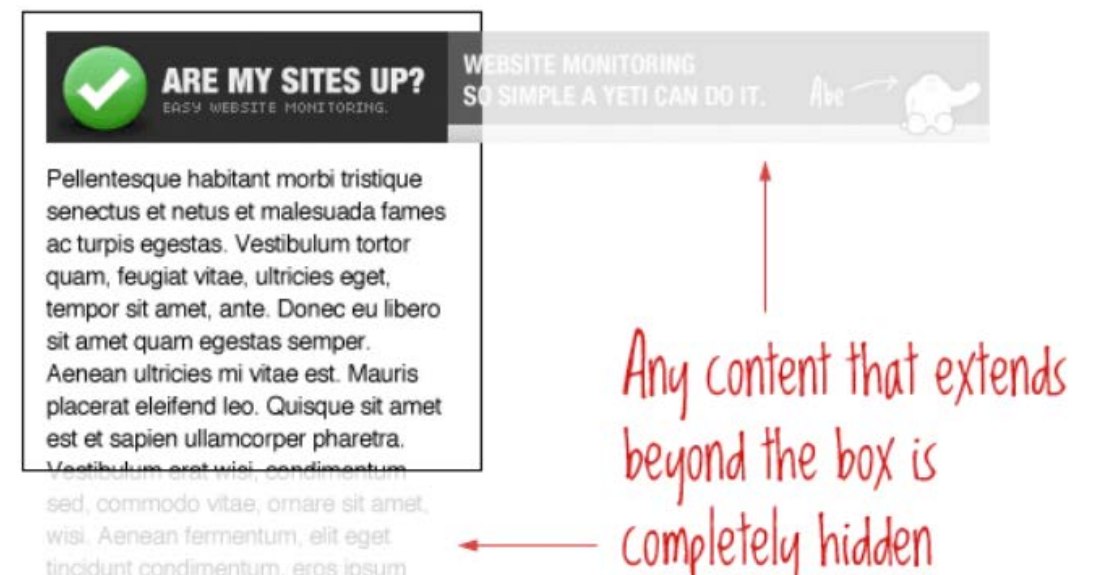
**overflow: scroll** makes scrollbars appear both horizontally and vertically...even if they don't need to be there.

- None of the content that would overflow appears outside the box



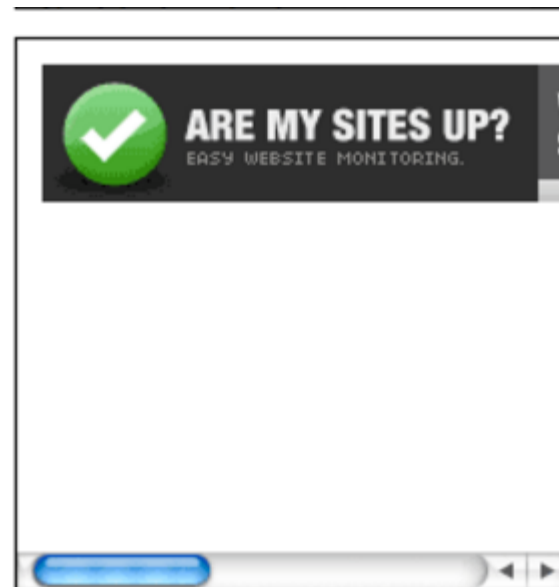
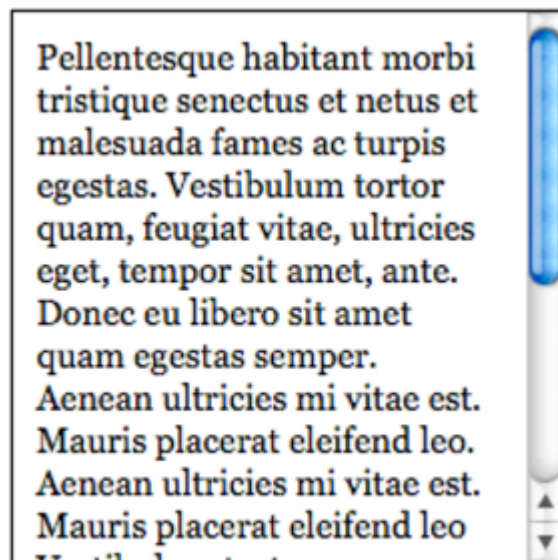
**overflow: hidden** cuts off any content that “sticks out” of its box

- No way to scroll, so content is no longer accessible



# THE OVERFLOW PROPERTY

**overflow:** **auto** only adds scrollbars when the content requires it (which may mean no scrollbars are added at all)



# THE CLEAR PROPERTY

So to solve our problem, you could add this CSS rule to the floated `div`:

```
.bio { overflow: visible; }
```



## BY BECK JOHNSON

Beck Johnson is a developer based in Seattle, WA. She plays board games in her spare time. Say hi at [beckjohnson@gmail.com](mailto:beckjohnson@gmail.com)

---



**PRACTICE TIME!**

# ASSIGNMENT

Create a container that has an image floated to the side of some text.

- Give the container a background color, gradient, or borders (to make sure floated content is really clearing properly)
- Try both techniques to force the container to expand around floated content
  - What happens when you use different **overflow** options?
- Apply box model properties like padding and margin so that your content looks nice

# WEB FONTS

# WEB FONTS

Remember that `font-family` looks for a font installed on the user's local machine.

```
body { font-family: Tahoma, sans-serif; }
```

If the Tahoma font isn't found, the browser will default to a generic sans-serif font instead.

What if you want to use an interesting font that most people aren't likely to have installed?

# WEB FONTS

The absolutely easiest way to get custom fonts is to link to a Google font stylesheet in the head of your page:

```
<link href="https://fonts.googleapis.com/css?family=Roboto"
rel="stylesheet">
```

Then just use the font by name in **font-family**, just like you would a web-safe font:

```
p { font-family: Roboto; }
```



# WEB FONTS

## Downsides:

































- Relies on a 3<sup>rd</sup> party to provide assets
- So if the request times out, the font won't download (so always provide fallbacks!)

## Upsides:

- Extremely easy
- Possibility that user already has the font in their web cache due to visiting other sites that use the same font

# ICON FONTS

Font Awesome is a free icon font that is used in many real-world projects.

 bullhorn	 bullseye	 bus	 cab (alias)
 calculator	 calendar	 calendar-check-o	 calendar-minus-o
 calendar-o	 calendar-plus-o	 calendar-times-o	 camera
 camera-retro	 car	 caret-square-o-down	 caret-square-o-left
 caret-square-o-right	 caret-square-o-up	 cart-arrow-down	 cart-plus
 cc	 certificate	 check	 check-circle
 check-circle-o	 check-square	 check-square-o	 child
 circle	 circle-o	 circle-o-notch	 circle-thin

Characters are replaced with vector images

- So to color or re-size icons on your site, just use the CSS **font** properties we already learned.

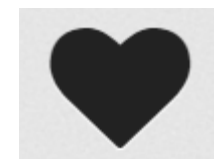
# ICON FONTS

To quickly start using Font Awesome, add this stylesheet to the head of your page:

```
<link href="http://maxcdn.bootstrapcdn.com/font-awesome/latest/css/font-awesome.min.css" rel="stylesheet">
```

Click an icon from the Font Awesome list, and copy the markup they provide, like:

```
<span class="fa fa-heart"></span>
```



# FONT AWESOME

You can put a font awesome class on any element:

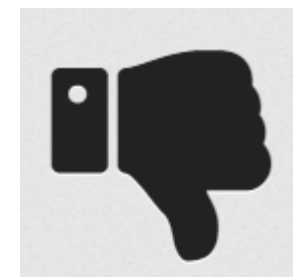
```
<span class="fa fa-paw"></span>
```



```
<i class="fa fa-cog"></i>
```



```
<h1 class="fa fa-thumbs-down"></h1>
```



# WEB FONTS

The other way to ensure people see the correct font is to download the font onto their computer when they load your page.

- Have to consider licensing fees – most fonts cost money
- There are free fonts available for download at websites like Font Squirrel or Font Spring
- Some fonts prohibit commercial use, or limit the number of pageviews

# @FONT-FACE

```
@font-face {  
  font-family: MyWebFont;  
  src: url(webfont.eot);  
       url(webfont.eot?#iefix) format('embedded-opentype'),  
       url(webfont.woff) format('woff'),  
       url(webfont.ttf) format('truetype'),  
       url('webfont.svg#svgFontName') format('svg');  
}
```

The `@font-face` declaration should appear before any other styles.

Different browsers support different font filetypes – modern browsers use woff, IE needs eot, and mobile devices need ttf or svg.



**PRACTICE TIME!**

# ASSIGNMENT

Find a free font from Google fonts and use it on your site by including the font stylesheet.

- Apply the font to some elements on the page

Include the Font Awesome stylesheet.

- Display at least two different icons
- Make them different sizes and/or colors
- Bonus points: what other CSS can you apply to the icons?



# “HOMEWORK”

- Practice!
- Optional: read chapters 15 and 17 of *HTML and CSS: Design and Build Websites*

