

LESSON 4

LAYOUTS

HOMework REVIEW

Image Alt Text

```

```

- Alt text lets the browser fall back to at least a word description of your content if the image doesn't load
- This is increasingly a big deal since bad mobile connections often fail to load imagery
- Screenreaders for visually impaired also rely on this:
<https://www.youtube.com/watch?v=UzffnbBex6c>

DONT DO THIS

Image Sizing

```

```

- Sizing images in HTML worked back in 2002 when everyone was using a desktop tower with IE6
- Today, image height and width is fluid and needs to be controlled in the CSS layer.

CODE INDENTATION

- Close every tag you open
- Use two space tabs for each level of nesting
- Nest content in containers

```
<html>
  <head>
    <title>Something Unique</title>
  </head>
  <body>
    <nav class="container">
      <ul>
        <li>Item</li>
      </ul>
    </nav>
  </body>
</html>
```


DONT DO THIS EITHER

```
<table>
  <tr>
    <td>
      <h1>My page</h1>
    </td>
    <td>
      <p>It's so great now</p>
    </td>
  </tr>
</table>
```

- Tables are for data, not page layouts
- People did this before CSS existed because they had no choice - you don't live in the digital desert like they did

TAKE A LOOK AT ASSIGNMENT 01

CSS POSITION



POSITIONING ELEMENTS

-display controls behavior of the box content sits in:

display: block; - element takes up as much width as possible, following element drops to a new line

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

hi

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

POSITIONING ELEMENTS

`display: inline;` - element takes up only as much width as it needs, `padding` / `margins` only work left + right, not top and bottom. Top and bottom spacing is controlled by `line-height` property because the content is in line.

Pellentesque *inline element* morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

POSITIONING ELEMENTS

`display: inline-block;` - combination of the two concepts. The item(s) will display inline with other items but you can use all margin, padding, height and width properties on an inline-block.

Pellentesque

*inline
block*

*inline
block*

*inline
block*

morbi tristique

senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

POSITIONING ELEMENTS

`display: none;` - nothing shows up at all. It's in the DOM the browser sees but the user won't see it. Seems useless now but just wait till we hit Javascript. You're going to love `display none`.



When building

LAYOUTS

Every Container Should Be A

BLOCK



CSS SIZING

UNITS

-You get five options because the internet gods love you that much:

px (pixels)

% (percentage)

em (literally Ms)

rem (relative Ms)

vh/vw (viewport height / viewport width)

-Concentrate on pixels and percentages to understand box models and layouts at a basic level

PIXELS

-Pixels are very straightforward. Your screen width is measured in pixels so a pixel unit is that part of your screen.

```
/* Moves aside element 15 pixels left */  
aside {  
    margin-left: 15px;  
}
```

-Problem: screens are often different sizes and pixel densities, what would you do here?

PERCENTAGES

-Percentages help fix this. Your screen width is always 100%, so now we don't have to worry about the actual pixel size of the screen.

```
/* Makes container 25% of screen width */  
.homepage-content-container {  
    width: 25%;  
}
```

-NOTE: Percentages are RELATIVE to their containers. If parent HTML element is 50% of page width and child HTML element is set to 50%, the child will be 25% of page width.

USAGE TIPS

- Percentages typically work well for layout, things like margin, padding and width.
- Pixels typically work better for style needs, like font-size on h1, h2, p, etc. em/rem sometimes work even better but don't go there just yet.
- You can use combinations of sizing units within the same selector, but doing the math on that is very difficult.

When building

LAYOUTS

Every Container Sizing Unit Should Be in

PERCENTAGES

**LET'S DO A
CODE ALONG**

A wide-angle photograph of a river flowing through a lush, green landscape. In the foreground, a large, colorful inflatable raft with a white pillow is partially visible, with a person sitting on it. Further down the river, several other people are floating on various inflatables, including a small blue ring and a yellow one with a pink umbrella. The river is bordered by rocky banks and dense green trees. In the background, a small town with colorful buildings is visible on a hillside under a clear blue sky.

FLOATS LAYOUTS

FLOATS

-Floating block level elements is a way to compose layouts where the elements can sit side-by-side



FLOATS

HTML:

```
<section class="container">  
  <div class="element"></div>  
  <div class="element"></div>  
  <div class="element"></div>  
  <div class="element"></div>  
</section>
```

CSS:

```
.element {  
  display: block;  
  float: left;  
  width: 25%;  
}
```


BONUS ROUND



You didn't win \$25000 unfortunately, but
with every float: left; declaration
you get a display: block; for **FREE**

FLOATS

HTML:

```
<section class="container">  
  <div class="element"></div>  
  <div class="element"></div>  
  <div class="element"></div>  
  <div class="element"></div>  
</section>
```

CSS:

```
.element {  
  float: left;  
  width: 25%;  
}
```

NOT QUITE THAT EASY

If you want to start a new row, you need to use the clear property. It should be applied to the floated element's container, like this:

```
HTML: <section class="container clearfix">
      <div class="element"></div>
      <div class="element"></div>
    </section>
```

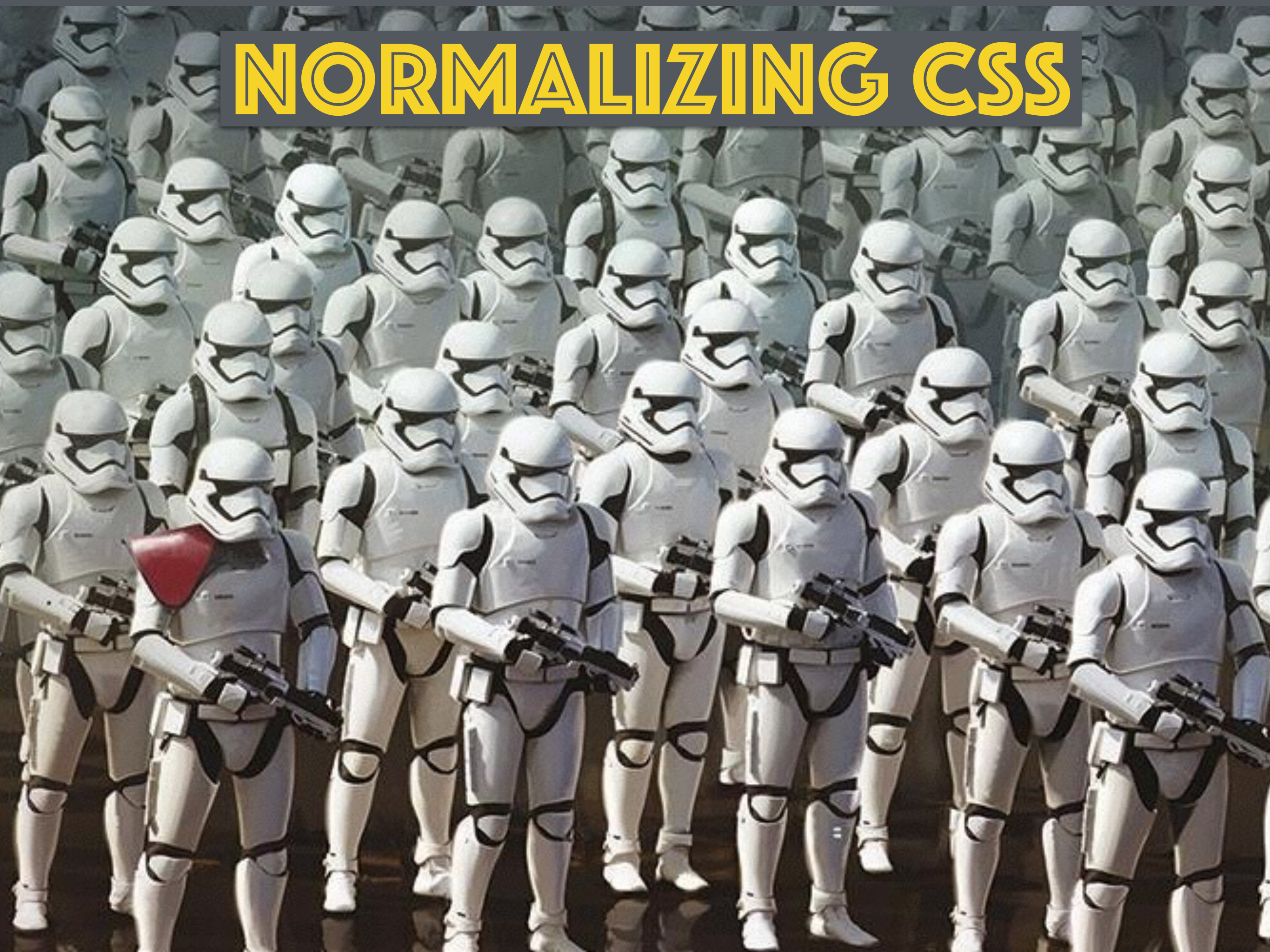
```
CSS: .clearfix:after {
      clear: both;
      content: " ";
      display: block;
    }
```


USAGE TIPS

- When doing multi-column layouts, always float in ONE direction. Since English is a left > right language, I prefer `float: left;` so the containers line up left to right.
- Keep the box model simple when you are starting. Add in padding and border to layouts SLOWLY and test often to understand how it all comes together.
- Clear your floats to start a new row.
- When doing a layout with percentages, don't use `margin left/right` as they will complicate the math.
- Remember, widths are RELATIVE to containers.
50% of 50% is 25% not 50%!!!!

TAKE A LOOK AT ASSIGNMENT 02

NORMALIZING CSS



NORMALIZING OUTPUT

- Browsers are all a little unique in how they render things
- Very smart people have compared and contrasted these very minor differences and fixed them for you - how nice.
- There are many of them but I'm going to make your life simple and point you to the best one:

Normalize.css: <http://necolas.github.io/normalize.css/>

HOW TO USE NORMALIZE

```
<head>  
  <title>Something Unique</title>  
  <link rel="stylesheet" href="css/normalize.css">  
  <link rel="stylesheet" href="css/main.css">  
</head>
```

- 1) Download normalize
- 2) Place normalize CSS before your external CSS
- 3) Code away like normal

Let's check it out in action:

<http://codepen.io/staypuftman/pen/VjPEpJ>

**USE NORMALIZE ON
THIS WEEKS HW**

FOR NEXT TIME

HW Assignment for Week 2 Due Sunday Night
Get ready for Git
Tooling