
RESPONSIVE
DESIGN

BACKGROUND

-The iPhone in 2007 changed how people used the internet. It took until ~2009 to become popular.

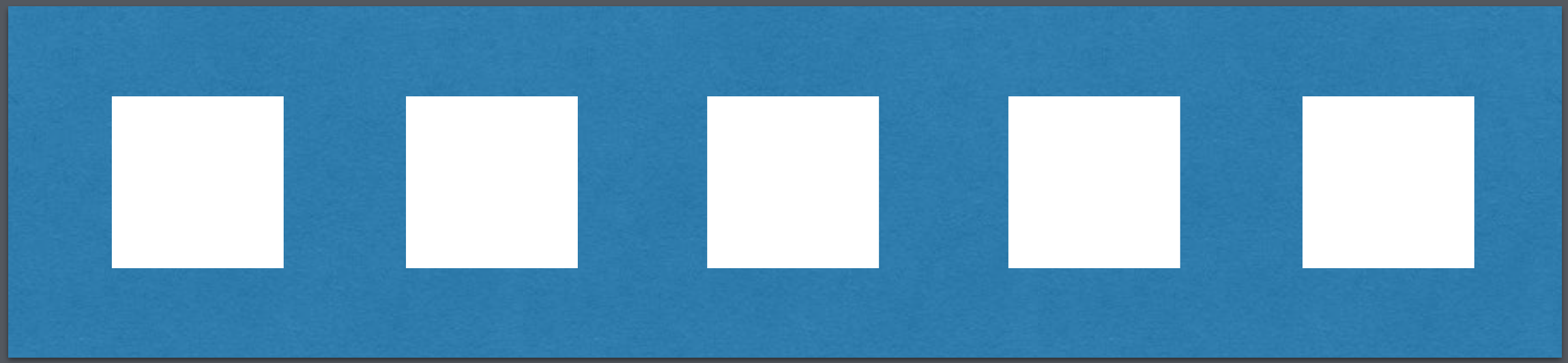
-This was first addressed by creating mobile-specific sites (started with m.site.com) but now you had to make multiple sites! Yuck.

BACKGROUND

-As browsers evolved into 2010, it became possible to build one site where you could adjust styles based on screen widths, creating one site for all devices.

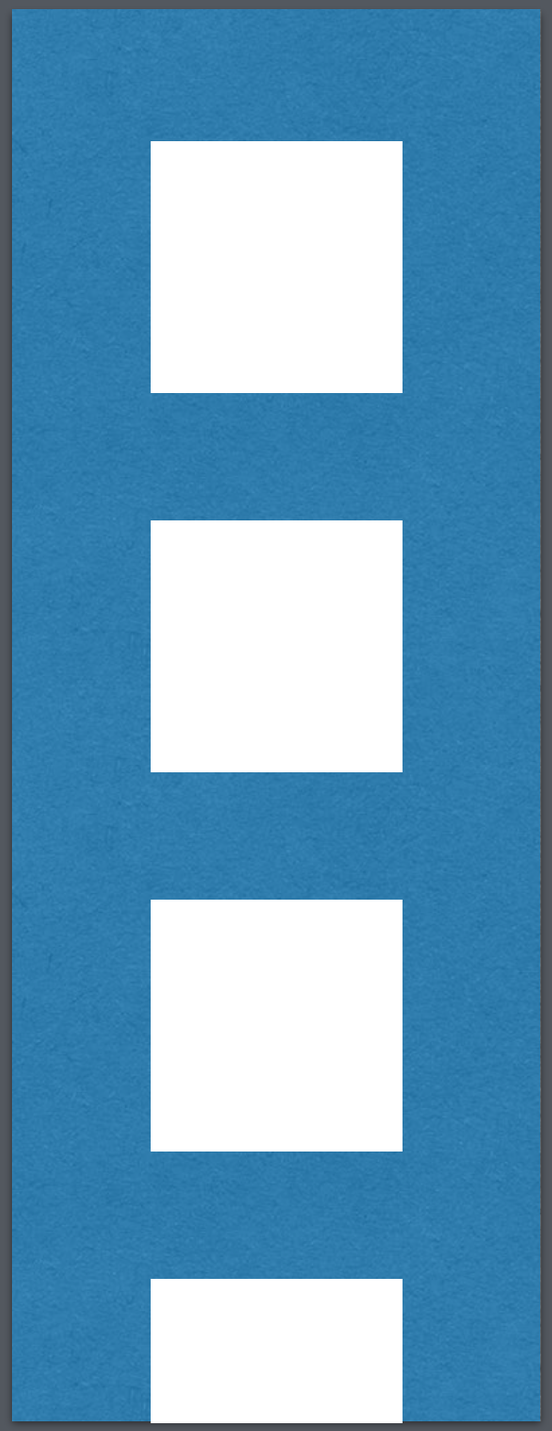
-Literally one guy (Ethan Marcotte) invented this technique with an article on the subject: http://alistapart.com/article/responsive-web-design

WHY?



Laptop/desktops are horizontal orientations

WHY?



Tablet/mobile phones are vertical orientations

LET'S EXPLORE

-Responsive design is so common now you probably don't even notice it:
Acceptable: http://www.cnn.com/
Good: https://www.ftc.gov/

But you notice it when it's not there:
https://twitter.com/

SCREEN SIZE

iPhone SE / iPhone 5S = 320px
Galaxy / iPhone 6 = 380- 400px
iPhone 6+, Galaxy Note = 420px
Phablet / tablet = 500px - 650px
iPad / iPad mini = 768px
Laptops = 960px - 1200px
Desktop = 1024px - 1800px

Think about device widths + CSS cascading. How would we target these?
http://screensiz.es/phone

DEVICE GROUPING

Phone >
- iPhone SE / iPhone 5S = 320px
- Galaxy / iPhone 6 = 380- 400px
- iPhone 6+, Galaxy Note = 420px

Tablet >
- Phablet / tablet = 500px - 650px
- iPad / iPad mini = 768px

Laptops >
- Laptops = 960px - 1200px

Desktops
- Desktop = 1024px - 1800px

Point in between groups = breakpoint

GOOD BREAKPOINTS

```
480 pixels  // phones
768 pixels  // tablets
960 pixels  // small screens
1200 pixels // big screens
```

Target your styles against these breakpoints - you can make others but these are versatile.

SIMPLEST FORM

```
@media (min-width: 480px) {
   .my-class {
     property: value;
   }
}
```

Target a width w/ `@media` and provide property/value pairs inside of current stylesheet - it is really that easy. What does this target?

TYPE AND WIDTH

```
@media screen and (min-width: 480px) {
  .my-class {
    property: value;
  }
}
```

What's happening here?

MEDIA TYPES

```
all          // Suitable for all devices
handheld     // mobile only
print        // styles for printed docs
screen       // color computer screens
tv           // intended for TVs (30"+)
```

Typically, all you would use is the screen type. This spec is a bit outdated, pre-responsive sites.

THINKING TIME

Why are my breakpoints AFTER the devices they are targeting? (ie phone breakpoint is 480px)

MOBILE-FIRST

This means your styles start with mobile and cascade to larger screens (preferred way of doing responsive design because mobile screens are harder to accommodate).

How would we structure media queries to do that?

Separate Stylesheet

```
<link rel="stylesheet " media="screen and
(min-width: 1024px)" href="css/
1024only.css">
```

Some put all mobile/tablet styles in a separate sheet - also doable. I prefer inline responsive styles because I can better track what's going on a per-style basis. Both methods work.

CODE ALONG

Assignment #1

EM TYPOGRAPHY

```
.class-cool {
  font-size: 2em;
}
```

-Em is a unit of measurement from print design (based on size of M, which is the largest letter)
-Sizing unit based on parent element's size = IMPORTANT!

PARENT SIZING!

```
.class-cool {
  font-size: 2em;
}
```

If `.class-cool` is inside a container with `font-size: 16px;` what size is `2em` in this case?

SENSIBLE USAGE

1) Set font-size in px on body

```
body {
   font-size: 12px;
}
```

2) Base other styles on it

```
h1 { font-size: 2.5em; }   // 30px
h2 { font-size: 2em; }     // 24px
```

This is called a vertical rhythm if you do it right.

THINKING THROUGH IT

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

div {
   height: 5em; // 100px
   width: 3em; // 60px
   margin: 0.5em; // 10px
   border: 0.05em; // 1px
}
```

People use em for layout properties but I think they're crazy - you'll soon see why.

EM COMPUTED VALUES

```
body {
   font-size: 12px;
}

div {
   font-size: 1.6em;      // 19.2px
}
```

```
h2 {
   font-size: 1.1em;      // 21.12px
   padding: 0.55em;       // 10.56px
   margin: 0.2em;         // 3.84px
}

<div class="container">
   <h2>How big is this title?</h2>
</div>
```

EM COMPUTED VALUES

```
body {
   font-size: 12px;
}

div {
   font-size: 1.6em;      // 19.2px
}

h2 {
   font-size: 1.1em;      // 13.2px
   padding: 0.55em;       // 6.6px
   margin: 0.2em;         // 2.4px
}

<h2>How big is this title?</h2>
```

REM TYPOGRAPHY

```
.class-cool {
  font-size: 2rem;
}
```

Not a mediocre band or a phase of sleep but rather Root EM.

It fixes the parent inheritance problem of EM: the sizing won't change depending on nesting.

REM COMPUTED VALUES

```
/* Wont change based on HTML nesting */

body {
   font-size: 12px;
}

div {
   font-size: 1.6rem;     // 19.2px
}

h2 {
   font-size: 1.1rem;     // 13.2px
   padding: 0.55rem;      // 6.6px
   margin: 0.2rem;        // 2.4px
}
```

WHY

The Em holy grail is book smart: set a font-size on the body and all layout sizes relative to it, then you can just reset the body size for responsive viewports and everything else follows accordingly.

WHY NOT

In reality, this has worked for me zero times. Even if your markup is perfect, somebody will come in and ask you to move something a little bit. At that point, you'll be making Ems of Ems and wondering what you've done with your life.

THOUGHTS ON EM

Some devs love EM. I'm not one of them because it usually makes your code harder to follow, especially big codebases.

Using percentages and pixels is much more semantic and legible to other humans who will come in contact with your work.

ADDITIONAL READING

Pixel + Em + REM technique:
https://css-tricks.com/rems-ems/

Pitfalls of Em and Rem but it can work given enough patience:
http://zellwk.com/blog/rem-vs-em/

CODE ALONG

Assignment #1