GENERAL ASSEMBLY

LESSON 13 - RESPONSIVE BASICS

LEARNING OBJECTIVES:

AFTER TODAY, YOU SHOULD BE ABLE TO...

- Articulate that responsive design is more design than code.
- Know the difference between fixed, fluid, and responsive layouts.
- Apply media queries to web sites to achieve a responsive layout.

AGENDA

- Learning Objectives
- Review - HTML & CSS
- Responsive Layouts
- Media Queries
- REM/EM

REVIEW

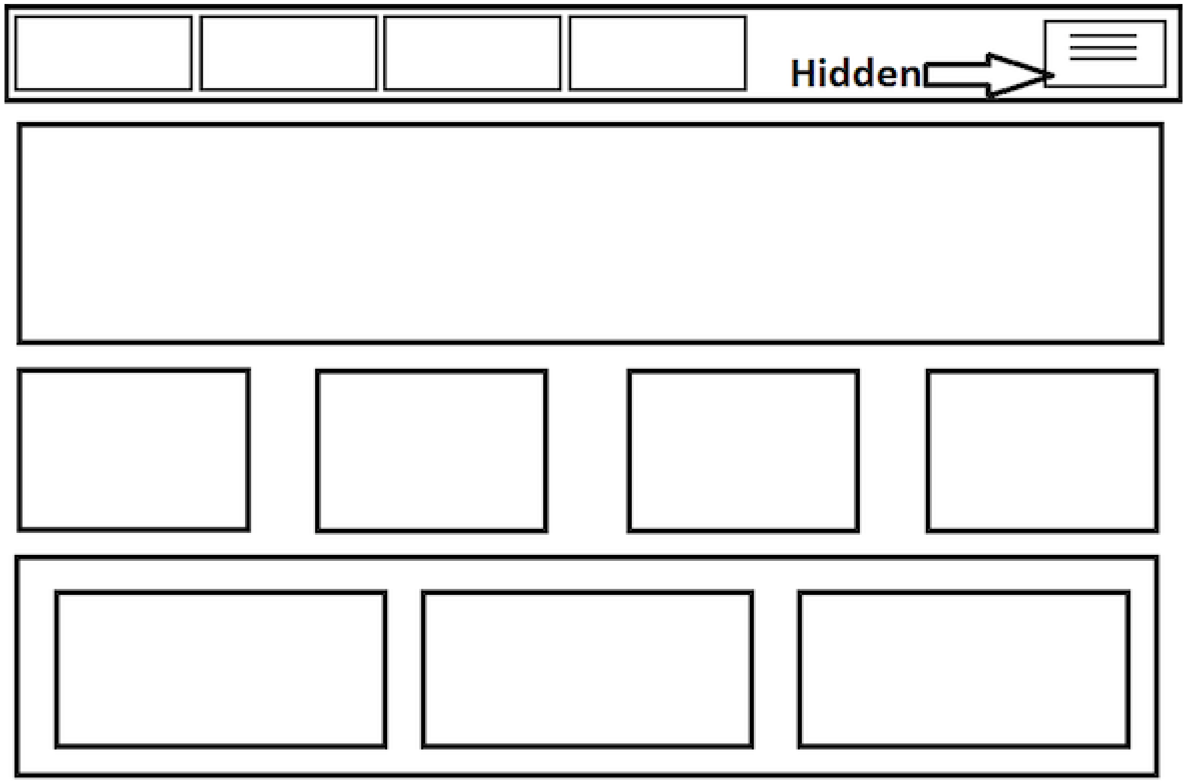
Let's bring HTML/CSS back into the equation...

Previously, we learned how we can use CSS + HTML to create unique layouts...

Let's review a few of them:

- Two-column
- Three-column
- Simple Marketing
- Marketing with Nav
- Magazine

YOU DO: BOXES LAYOUT


Hidden

RESPONSIVE LAYOUT

WHAT DOES "RESPONSIVE" MEAN?

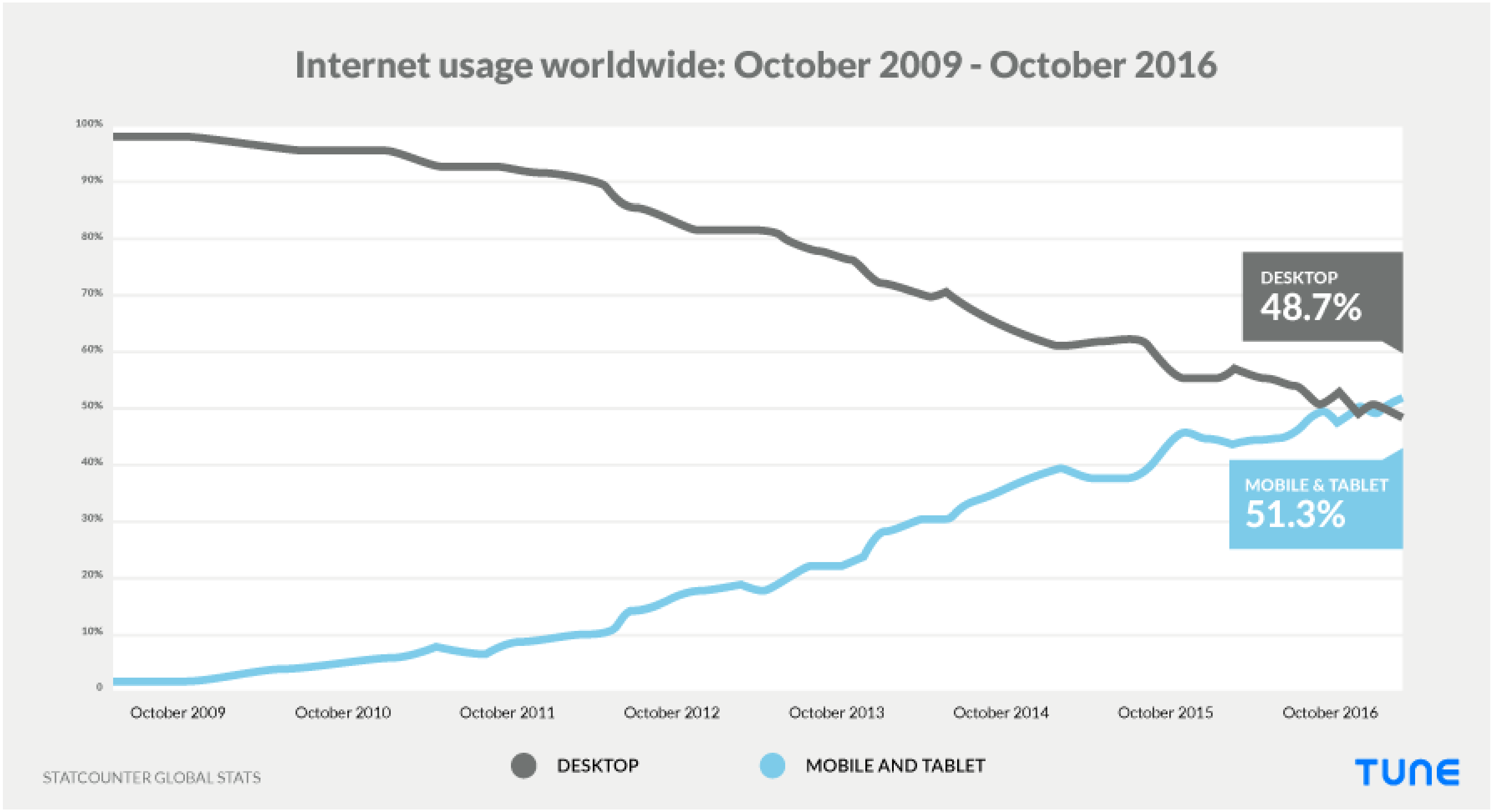
Responsive Web Design is about using HTML and CSS to resize, hide, shrink, enlarge, or move the content to make it look good on any screen.

This is more about the design than it is about any coding (though there is a bit of this involved)

BACKGROUND

Smart Phones.

Around late 2009, when the first iPhones, iPads, etc. really started to take off, we began noticing a trend in the way folks browsed the web...


Internet usage worldwide: October 2009 - October 2016
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0
October 2009
October 2010
October 2011
October 2012
October 2013
October 2014
October 2015
October 2016
DESKTOP
48.7%
MOBILE & TABLET
51.3%
DESKTOP
MOBILE AND TABLET
STATCOUNTER GLOBAL STATS
TUNE

This meant web designers/developers had to change how we thought about layout design...

The first wave of sites designed to address this was accomplished using mobile-specific websites (i.e. m.site.com).

But this meant you had to design an entirely new site for mobile!

WHY?

Because desktops had horizontal-oriented screens,
while mobile-devices had vertical-oriented screens

vs.

Thus far, we've been designing either fixed or fluid designs...

FIXED VS FLUID VS RESPONSIVE

So, let's explore the different types of web layouts...

FIXED LAYOUT

Originally, all of our websites were fized layouts. This worked since we didn't need to worry much about different screen widths...

- Relies on a container of fixed width
- Usually 960px or 1000px

FLUID LAYOUT

- Sized in percentages
- Proportions stay the same

RESPONSIVE LAYOUT

- Different styles for different screen widths
- Uses an elastic/fluid layout w/ breakpoints

CODE ALONG

Let's take a look at an example of each type of layout...

- Proextensions.com

Now, a fluid layout:

- Startup Matchmaker Lab

And let's take a look at several responsive sites:

- Msiddeeq.com
- Generalassemb.ly

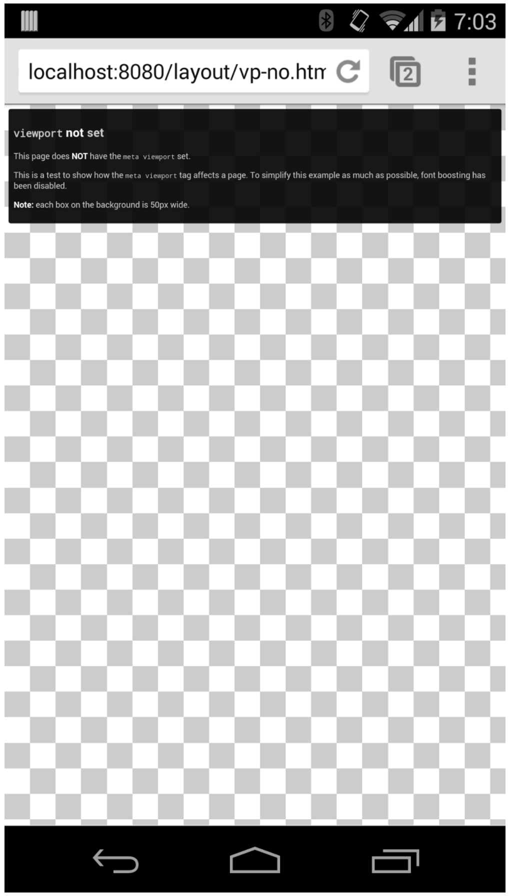
5 MINUTE BREAK

MEDIA QUERIES

GETTING STARTED

```
<meta name="viewport" content="width=device-width, initial-sca
```

- Use the meta viewport tag to control the width and scaling of the browser's viewport.
- Include width=device-width to match the screen's width in device-independent pixels.
- Include initial-scale=1 to establish a 1:1 relationship between CSS pixels and device-independent pixels.

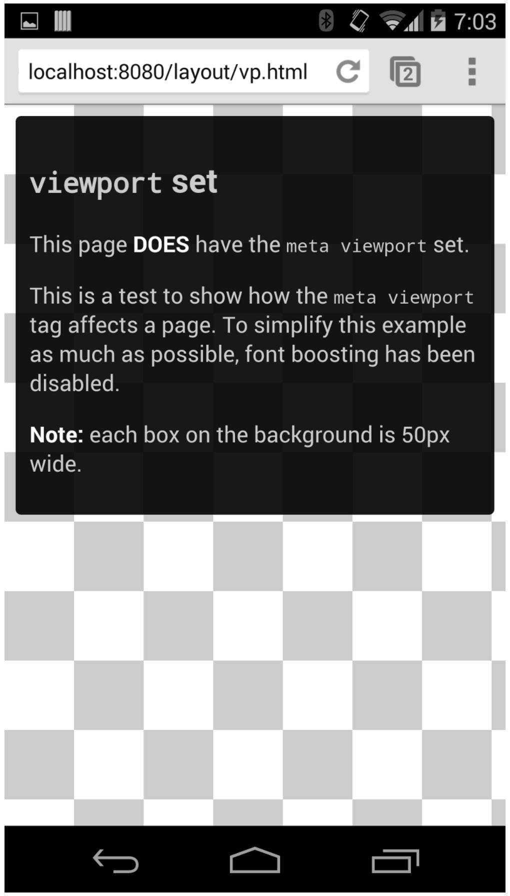

localhost:8080/layout/vp-no.htm

`viewport` **not set**

This page does **NOT** have the `meta viewport` set.

This is a test to show how the `meta viewport` tag affects a page. To simplify this example as much as possible, font boosting has been disabled.

Note: each box on the background is 50px wide.


localhost:8080/layout/vp.html

`viewport` **set**

This page **DOES** have the `meta viewport` set.

This is a test to show how the `meta viewport` tag affects a page. To simplify this example as much as possible, font boosting has been disabled.

Note: each box on the background is 50px wide.

More information on Viewport Meta Tag

@MEDIA RULE

The **@media rule** is used in media queries to apply different styles for different media types/devices.

The **@media rule** made it possible to define different style rules for different media types:

- all - used for all media types
- print - used for printers
- screen - used for computer screens, tablets, and smart phones
- speech - used for screenreaders

Media queries can also be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?) resolution

More Media Queries Resources:

- Mozilla Developer Network
- W3Schools

EXAMPLE CSS W/ MEDIA QUERIES

```
/* stack boxes */
.box{
    float: none;
}

@media screen and (min-width:768px){
    /* insert responsive css here ex: float boxes in columns *
    .box{
        float: left;
    }
}
```

BREAKPOINTS

A major component of responsive design is creating the right experience for the right screensize.

To achieve this, we typically define a specific width at which our media queries kick in -- these are out breakpoints.

If you're looking for a list of common breakpoints for the most common devices, check out this nifty article:

CSS Tricks

BE CAREFUL!

There are TONS of devices out there; each with their own specific screen resolutions (both portrait and landscape)

Don't design for devices! Design for experiences.

COMMON MEDIA QUERIES BREAKPOINTS:

```
/*==========
Mobile First Method
==========*/

... default CSS styles ...

/* Extra Small Devices, Phones */
@media only screen and (min-width : 480px) {

}

/* Small Devices, Tablets */
@media only screen and (min-width : 768px) {

}
```

FRAMEWORKS:

There are a number of frontend frameworks out there that handle most of this for you:

- Bootstrap 4
- Foundation 6
- Skeleton
- more...

PULSE CHECK - RESPONSIVE CODEPEN

Basic Media Query Example

EMS VS REMS

WHAT IS AN EM?

An `em` is a CSS unit relative to the font-size of the element.

1em = 100% font-size

- Relative unit sized based on the width of the letter 'm'
- Same as percentages
- Based on parent

```css
div {
    font-size: 20px;
}

div p {
    font-size: 2em; /* This font-size will result in 40px */
}
```

But... This could become an issue:

```
div {
    font-size: 24px;
}

div p {
    font-size: 0.75em; /* This font-size will be 18px */
}

div p span {
    font-size: 0.75em; /* However, this font-size will be 13.5
}
```

REM

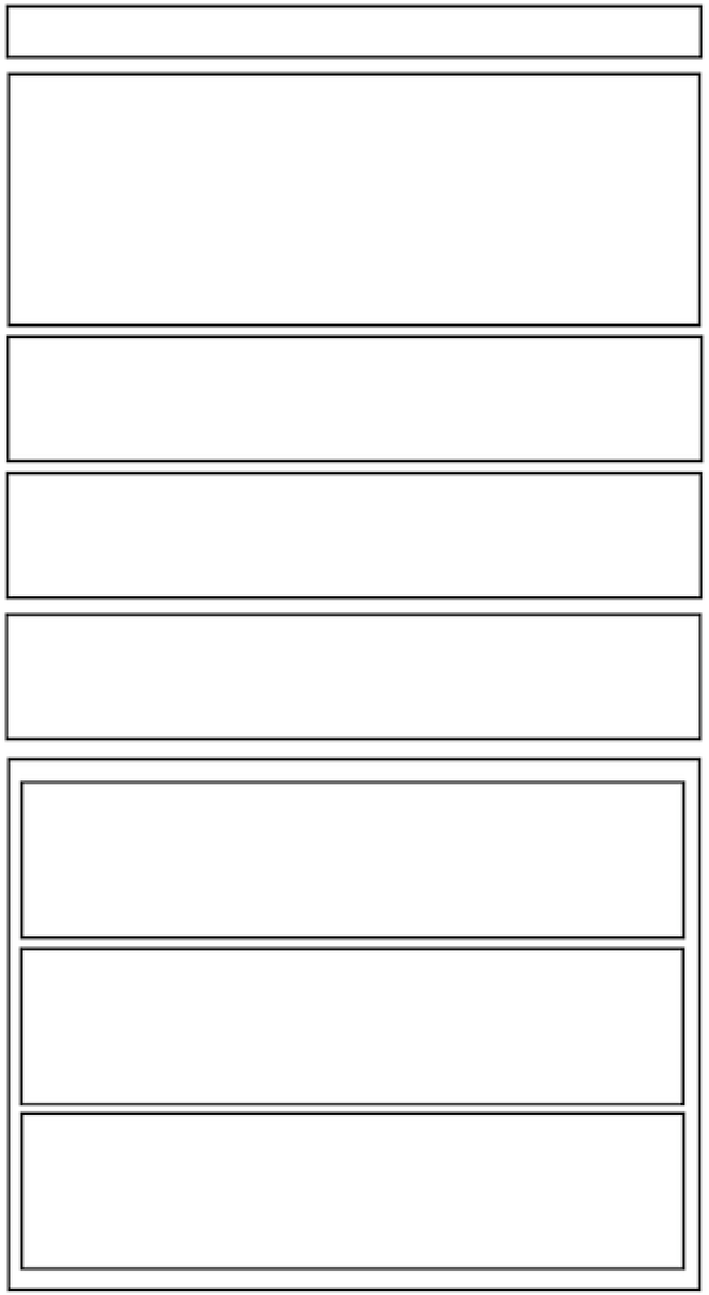
The `rem` (or root em) is the same concept as the `em`, except it references the "root element" (html element)

```
html {
    font-size: 24px;
}

div p {
    font-size: .75rem; /* This font-size will be 18px */
}

div p span {
    font-size: .75rem; /* This font-size will still be 18px */
}
```

YOU DO: MOBILE BOXES



YOU DO: RESPONSIVE BOXES - MEDIA QUERIES

LEARNING OBJECTIVES REVIEW

- We Articulated that responsive design is more design than code.
- We now Know the difference between fixed, fluid, and responsive layouts.
- We Applied media queries to web sites to achieve a responsive layout.

WEEK 6 HOMEWORK

Due: Wednesday, Feb 7