Responsive Basics

Lesson 13

Learning Objectives

- 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

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

HTML & CSS Review

Code Along

Open: **Lesson 13 HTML/CSS Layout Review Codepen**

Directions:

1. Open the **Codepen**
2. Let's build out the Marketing w/ Nav layout together using Flexbox

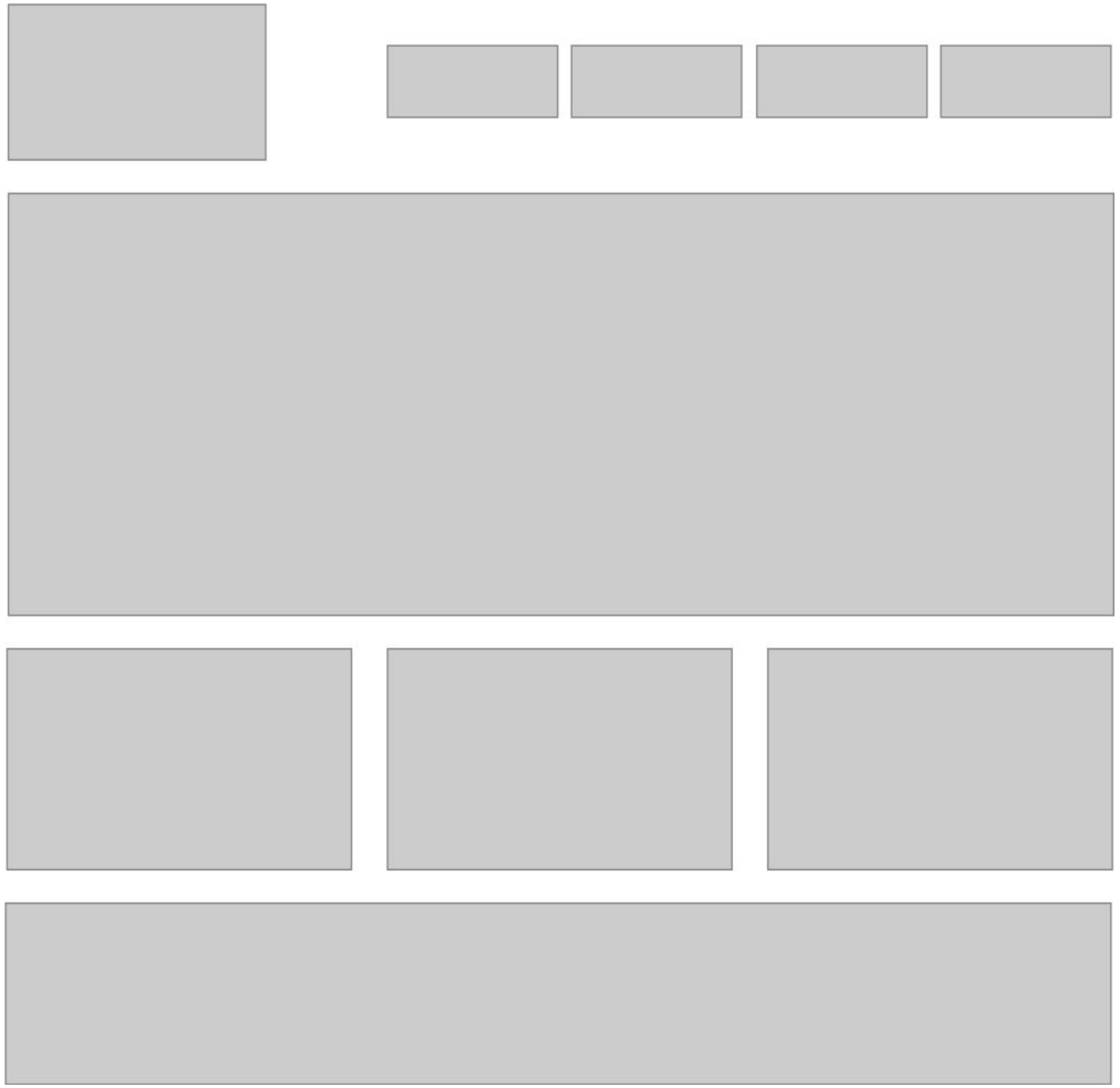
Timing:

- **10 minutes** - using flexbox, let's build out the Marketing w/ Nav layout

Tips:

- Take it one step at a time!
- Make sure you have the appropriate flex containers/items

Marketing with Nav



Exercise

Open: **starter_code/boxes**

Directions:

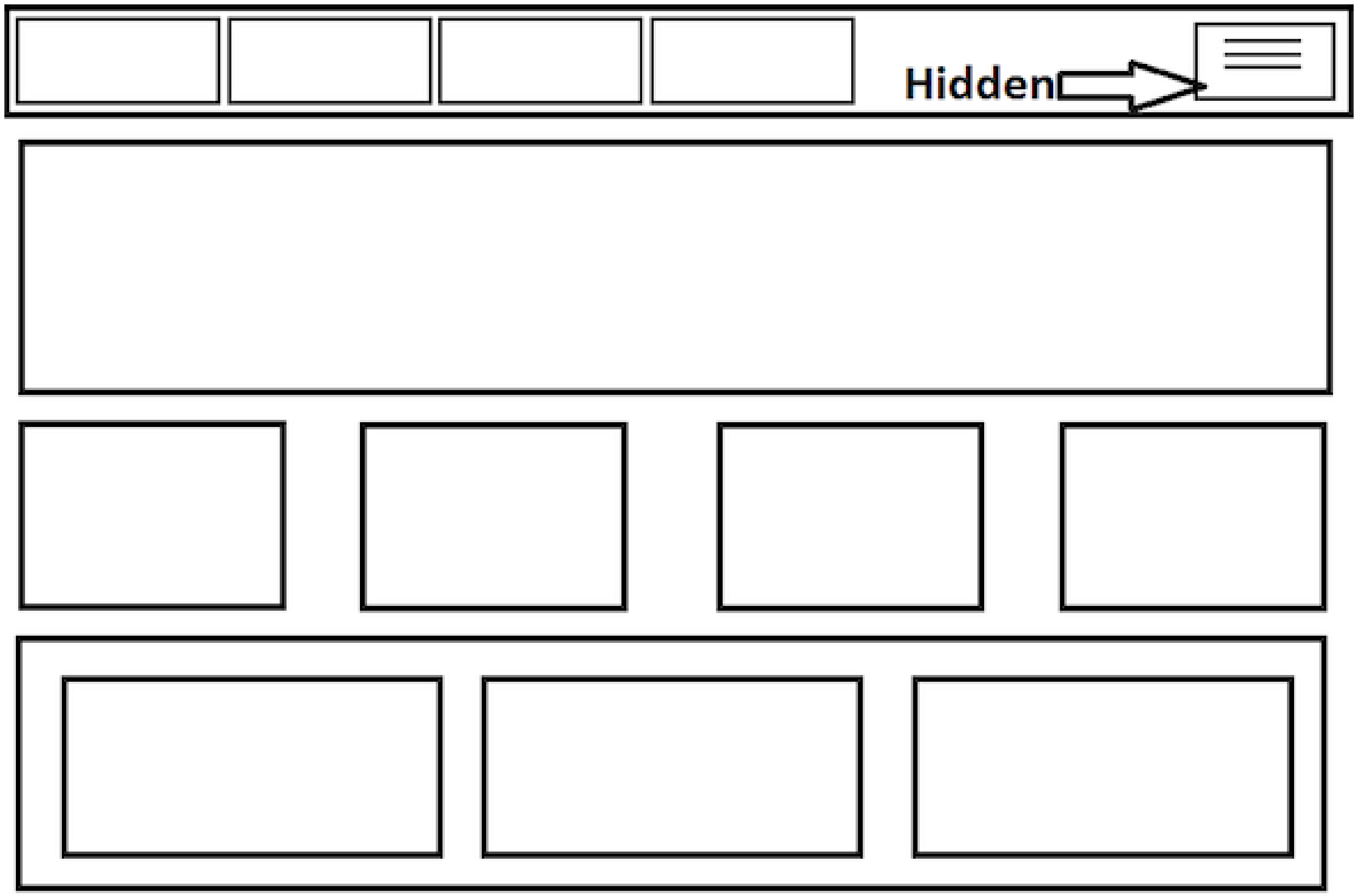
1. Open the **starter_code/boxes** folder
2. Have the students recreate the layout found in the *Boxes.png* image

Timing:

- **20 minutes** - using flexbox, students should create the layout from the *Boxes.png* image

Tips:

- Take it one step at a time!
- Remember your HTML structure, and be sure to include all of the required tags
- Hamburger icon HTML code: &#9776;


Hidden

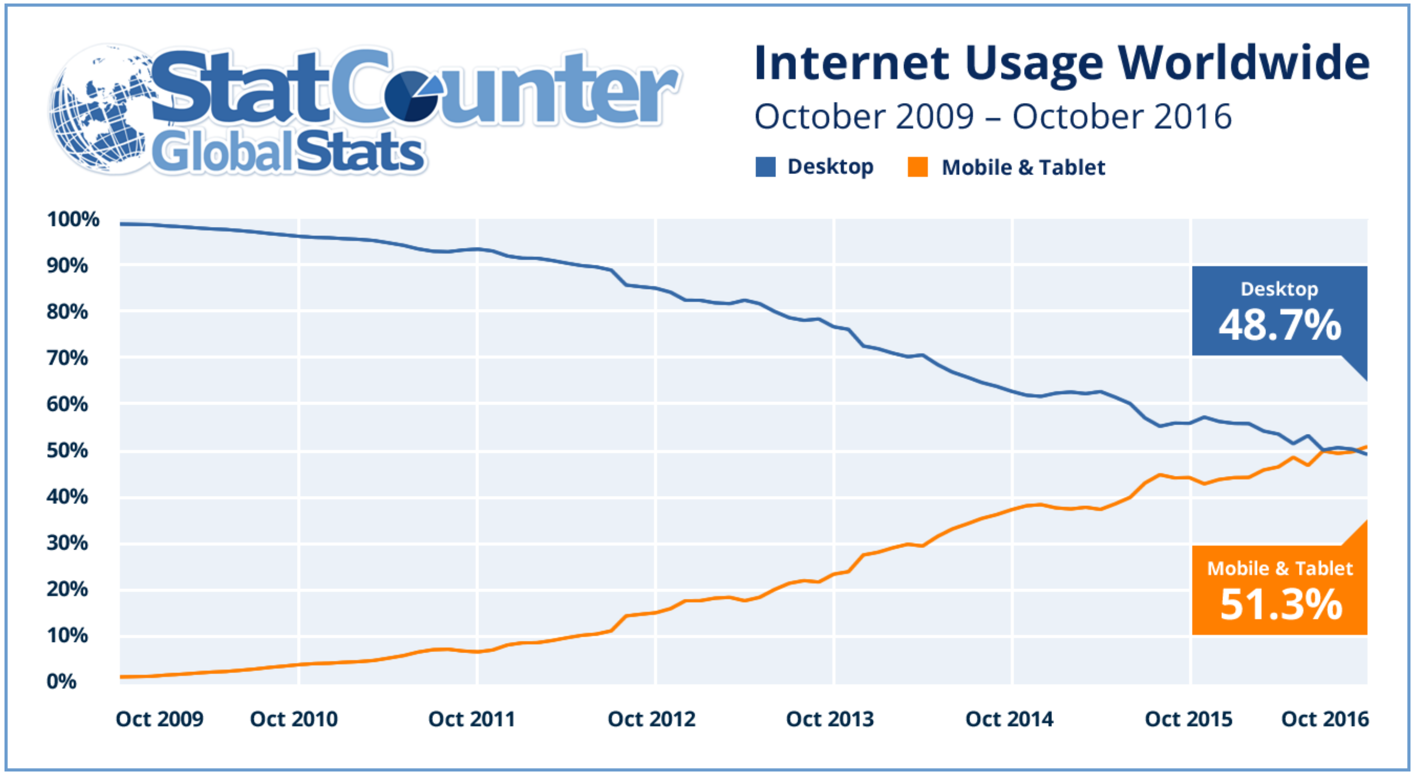
Responsive Layout

What is Responsive?

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)

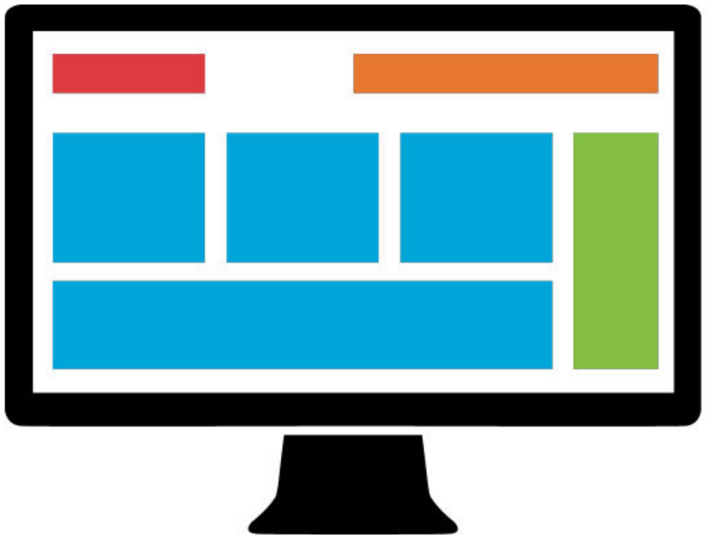
Web Usage Stats



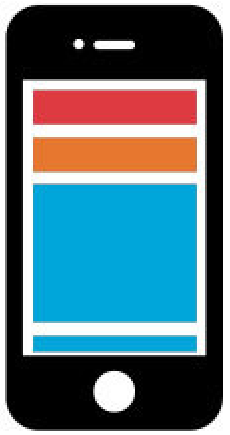
Around late 2009, when the first iPhones, iPads, etc. really started to take off, we began noticing a trend in the way users browsed the web...

Mobile-specific Websites

www.site.com



m.site.com



Initially, developers created 2 entirely separate websites -- one for desktop and another for mobile.

Single Responsive Website

www.site.com

With the advent of media-queries in CSS3, now developers can style the same site differently depending on screen-size, etc.

Website Layouts

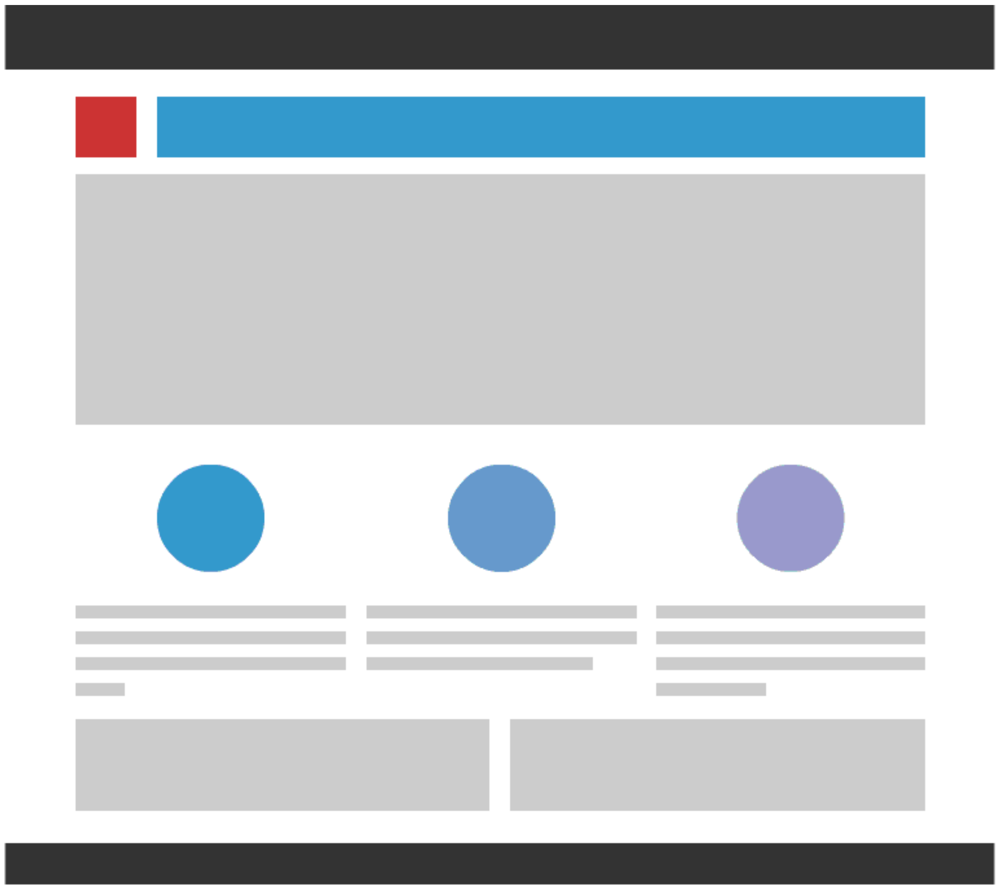
There are 3 general layout types in web design, each with their own pros & cons: **fixed, fluid, and responsive**.

Fixed Layout

A **fixed layout** has a wrapper that is a fixed (pixel) width, and the components inside it have either percentage widths or fixed widths.

The important thing is that the container (wrapper) element is not set to change. No matter the screen resolution, the website will display with the same width.

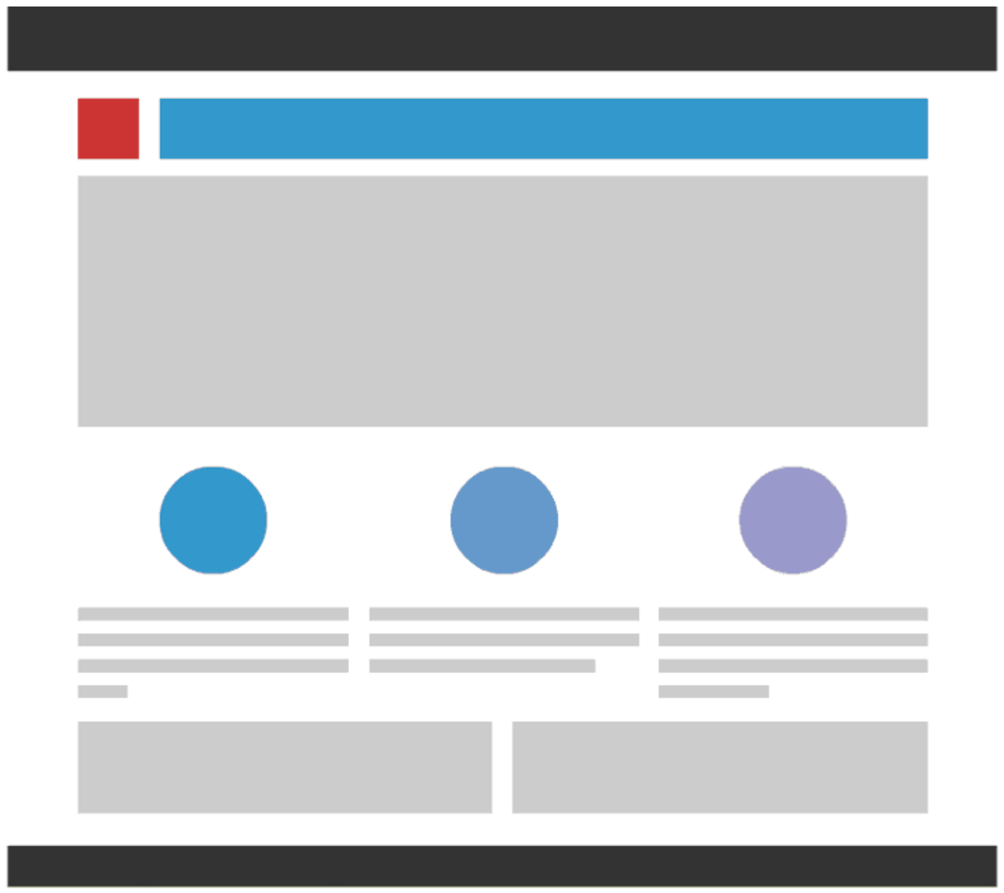
Fixed Layout Example



Fluid Layout

In a **fluid layout**, the container uses a percentage width, and the majority of the components inside have percentage widths, and thus adjust to the user's screen resolution.

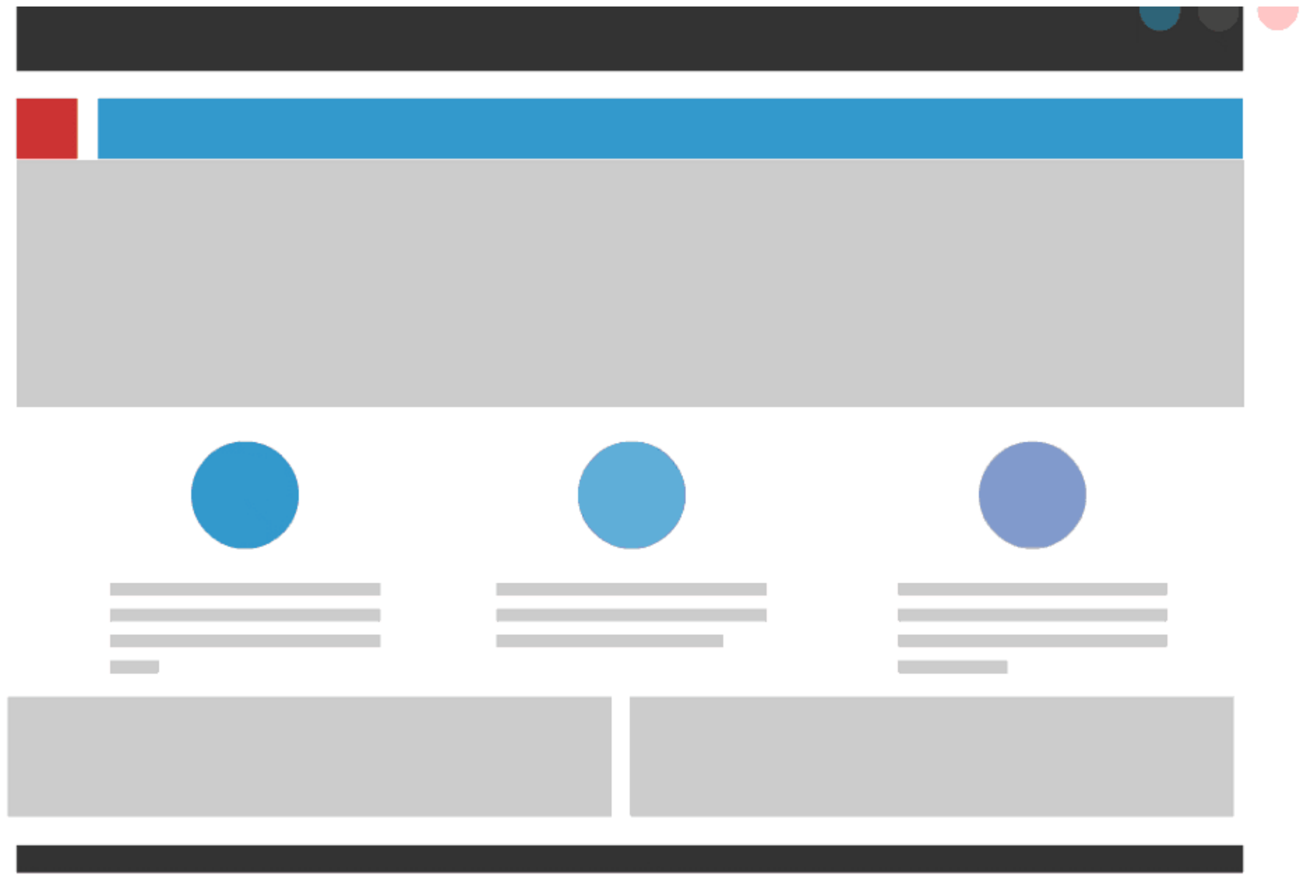
Fluid Layout Example



Responsive Layout

A **responsive layout** consists of a mixture of flexible grids and layouts, images, and an intelligent use of CSS media queries. This allows the website to adapt depending on the device, screen resolution and orientation, etc.

Responsive Layout Example



Pulse Check

Key Objective:

Let's explore examples of each type of website, and discuss the differences as a class.

- Fixed Layout - ProExtensions.com
- Fluid Layout - Startup Matchmaker (localhost)
- Responsive Layout
    - Msiddeeq.com (simple)
    - Generalassemb.ly (complex)

Timing:

- **15 minutes** - As a class, lets visit 3 different websites and analyze their layouts.

Tips:

- Use Chrome Dev Tools to toggle between different device sizes.

Break time!

Let's take 5-10 minutes to decompress...

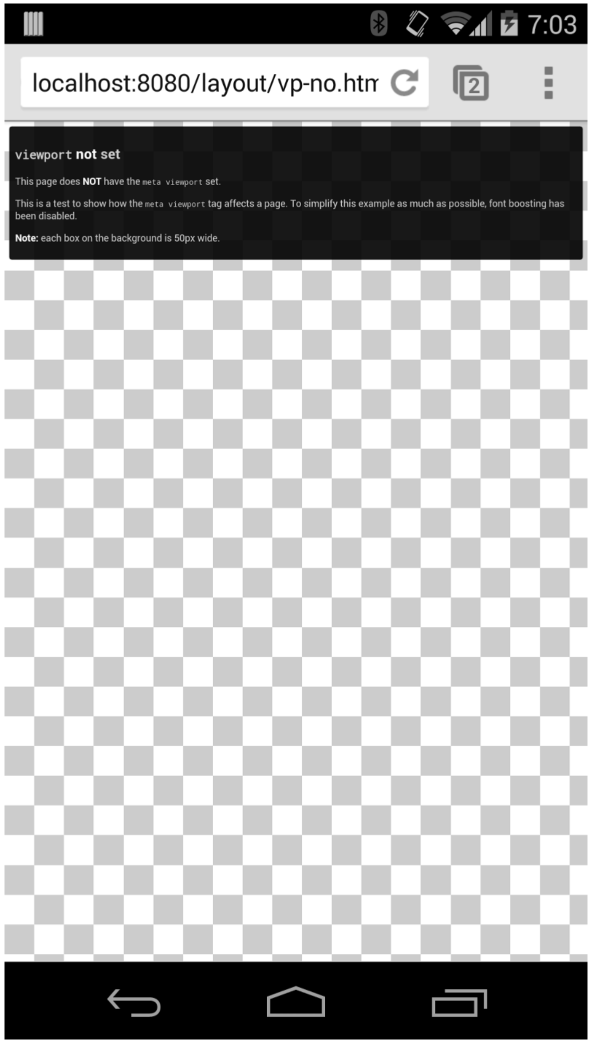
Media Queries

Viewport meta tag

```
<head>
    <!-- This goes in the head section of your site -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
```

- A **meta viewport** element gives the browser instructions on how to control the page's dimensions and scaling.
- The **width=device-width** part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- The **initial-scale=1.0** part sets the initial zoom level when the page is first loaded by the browser.

Without Viewport Tag


localhost:8080/layout/vp-no.htm

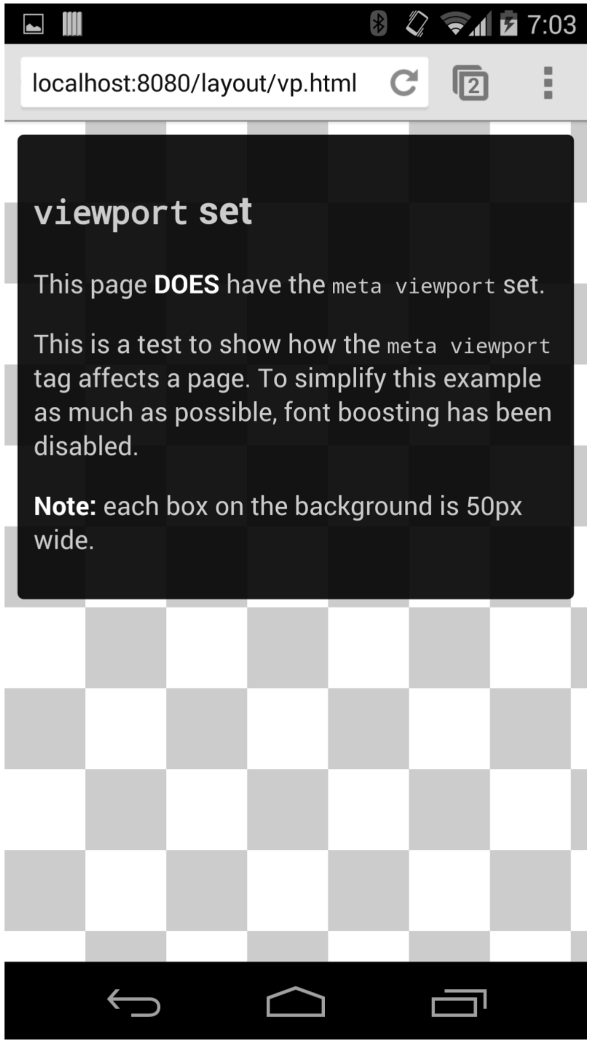
`viewport` **not** set

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.

With Viewport Tag


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.

What are Media Queries?

A **media query** is a CSS technique introduced in CSS3.

It uses the *@media* rule to include a block of CSS properties only if a certain condition is true.

```css
div.box {
    float: none;
}

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

@media Rule

The **@media rule** is used in CSS media queries to apply different styles for different media types/devices, such as:

- Viewport width/height
- Device width/height
- Device orientation
- Print styles
- Screenreaders
- and more...

Media Queries Order

Place your media queries at the **bottom of your stylesheet** so they can override code that comes before them.

If you have multiple media queries, they should go from smallest to largest (i.e. min-width: 400px would be placed above min-width: 768px, etc.), whenever possible.

This is referred to as "mobile-first".

Media Query Syntax - Media Types

Styles for computer screens

```
@media screen and (min-width: 960px){
    div {
        width: 75%;
    }
}
```

Styles for print

```
@media print {
    header, footer, nav {
        display: none;
    }
}
```

Media Query Syntax (cont'd)

min-width: screens 960px wide *or more*

```
@media screen and (min-width: 960px){
    div {
        width: 75%;
    }
}
```

max-width: screens 960px wide *or less*

```
@media screen and (max-width: 960px){
    div {
        width: 50%;
    }
}
```

Breakpoints

The point at which our media queries kick-in is commonly referred to as a **breakpoint**.

As you can see, this is a major component of responsive web design, and it allows us to create the right experience for the right screen size.

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 Breakpoints

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

... default CSS styles ...

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

    ... CSS for screens at least 480px wide ...

}

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

    ... CSS for screens at least 768px wide ...

}

/* Medium Devices, Desktops */
@media only screen and (min-width: 992px) {

    ... CSS for screens at least 992px wide ...

}

/* Large Devices, Wide Screens */
@media only screen and (min-width: 1200px) {

    ... CSS for screens 1200px and above ...

}
```

```
/*========  Non-Mobile First Method  ========*/

... default CSS styles ...

/* Large Devices, Wide Screens */
@media only screen and (max-width: 1200px) {

    ... CSS for screens 1200px and below ...

}

/* Medium Devices, Desktops */
@media only screen and (max-width: 992px) {

    ... CSS for screens up to 992px wide ...

}

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

    ... CSS for screens up to 768px wide ...

}

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

    ... CSS for screens up to 480px wide ...

}
```

Frameworks

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

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

Media Query Resources

- Responsive Web Design - Media Queries (W3Schools)
- CSS @media Rule (W3Schools)
- Media Queries for Standard Devices (CSSTricks)
- Using Media Queries (MDN)

Code Along

Open: **Basic Media Query Codepen**

Pulse Check

Key Objective:

Open the **Basic Media Query Codepen** link and add some media queries to create breakpoints in the code.

Timing:

- **10 minutes** - In a group, add a breakpoint at 768px and 992px
- **5 minutes** - Come together to discuss

Tips:

- Be sure to use "mobile-first" principles when adding your breakpoints

EM/REM

What is an em?

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

1em = 100% of the font-size (default 16px)

Em example

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

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

Em example (cont'd)

But... this could become an issue with nested ems:

```
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 13.5px */
}
```

Rem

The **rem** (or root em) is the same concept as **em**, except it references the "root element" (html element) rather than itself/the parent 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 */
}
```

Lab Time (part 1)

Open: **starter_code/boxes**

Directions:

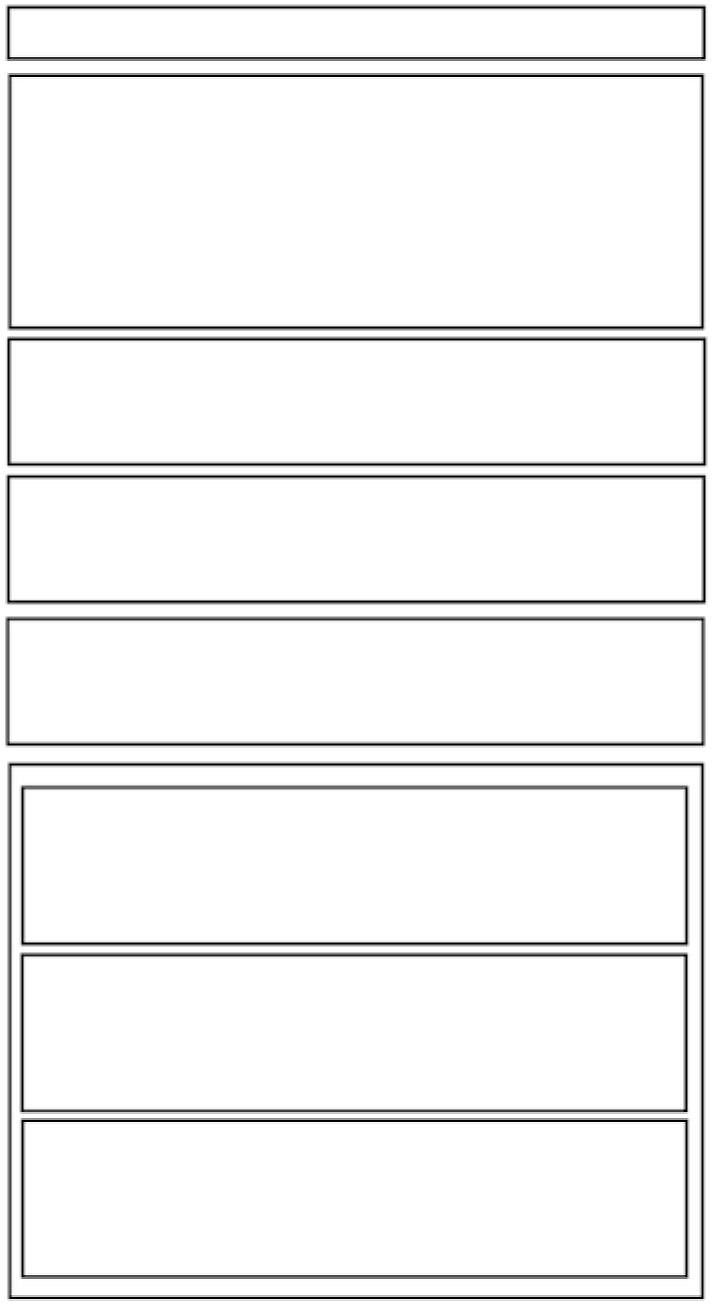
1. Open the CSS file in your **starter_code/boxes** folder
2. Place a comment at the bottom ("/* overwriting CSS for new layout goes here */")
3. Below this, add CSS to make original page look like *Mobile_boxes.png* image

Timing:

- **40 minutes** - add new CSS below default CSS to make page look like *Mobile_boxes.png* image

Tips:

- Should **NOT** need to edit the HTML; only use CSS to manipulate the existing elements



Lab Time (part 2)

Open: **starter_code/boxes**

Directions:

1. Add media-query around second set of CSS so the style will only apply when window is 768px wide or less

Timing:

- **20 minutes** - add CSS media query around second set of CSS

Tips:

- Should be able to do this on your own.
- This is NOT mobile-first design, so adapt your media query accordingly

Exit Tickets

Take 5-10 minutes to give us some

(Link is in Slack Room)

Learning Objectives Review

- We articulated that responsive design is more design than code.
- We know the difference between fixed, fluid, and responsive layouts.
- We applied media queries to web sites to achieve a responsive layout.

Week 6 HW Due Wednesday

- **Due Wednesday by 11:59pm ET**
- Once you upload your HW to your repo, please PM the TA with a link to the Assignment folder.
- This is how we will know whether an assignment has been completed or not
- Once graded, the TA will reply with feedback

Final Projects

- **Milestone 2 (due today):** HTML/CSS Prototype and pseudocode of JS
- **Milestone 3 (due next week):** First Draft of JS

Next Class...

Lesson 14 - Responsive Lab