Responsive Basics

Lesson 13

Why are we here today?

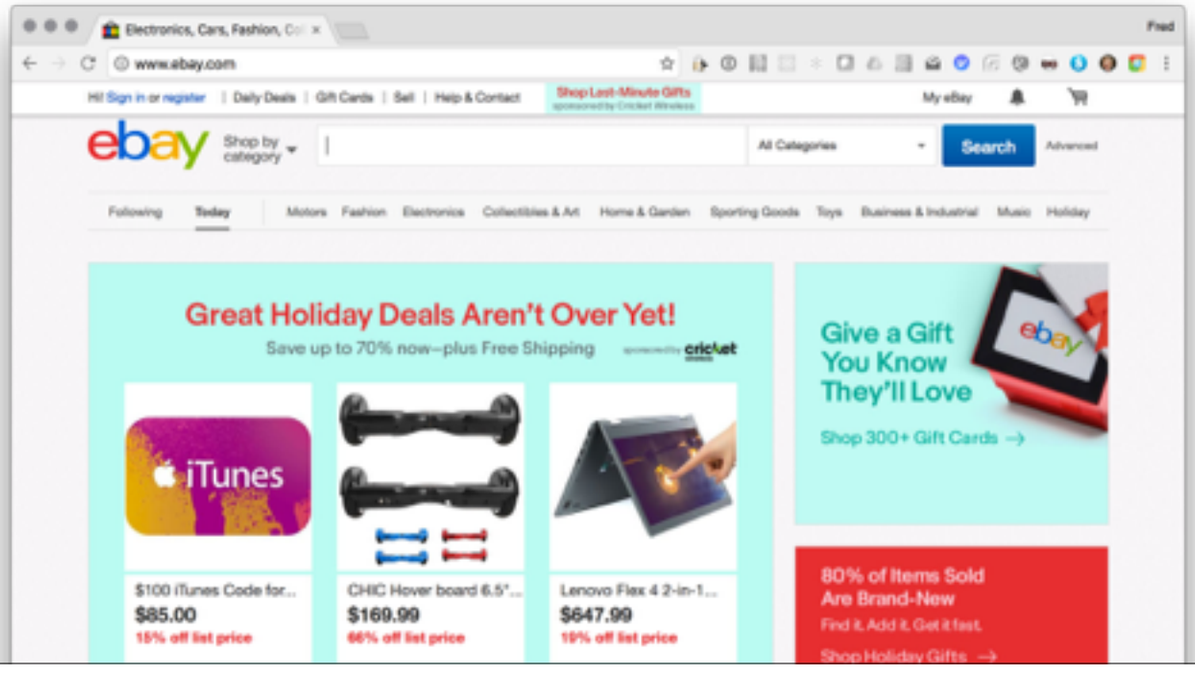
-For better or worse, the iPhone changed design
-Mobile browsers, made us have to change the way we designed
-78 percent of internet users access the web form their handset (i think my dad is the only holdout)

Objectives

- Describe responsive **design**
- Know the difference between fixed and responsive layouts
- Understand the difference between fluid and elastic layouts
- Apply media queries to achieve responsive and mobile layouts

In the Beginning…

- All websites were Fixed-Width Sites

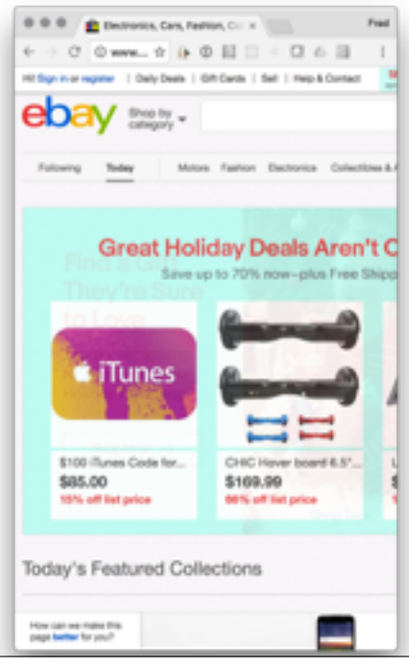


People had to decide how big their target audiences screens were.

Did your target have 640x480px(old laptop), 800x600px or 1024X768px (720px)
If I recall, I mostly designed for 800x600. Though I usually set my max width and then used percentages inside.

Not responsive

- When squeezed down, all columns squeeze down
- Half the page is missing when we make the browser smaller



eBay is using 980px as it width.

Non-responsive sites

- www.google.com
- www.huffingtonpost.com
- www.ebay.com

These sites in many cases do have a mobile version of their site passed in either through the route as m.ebay.com or as a parameter on the request www.ebay.com?layout=m.

These routing changes allow for a completely different layout to be shown on a mobile device.

Responsive Sites

- www.generalassemb.ly
- sweethatclub.com
- http://bradfrost.github.io/this-is-responsive/

- Let's take a note of the changes we see here.

What is happening as we make the site smaller. Let's look at what is happening in the console. We can see that different styling is applied as a page is made smaller. Font sizes may get larger to be better seen. Menus may disappear. Hamburger menu appears as a dropdown.

Other areas may stack on top of each other.

Let's take an inventory of the changes and what we see is happening. The html and css in these responsive sites are using the same layout and css to achieve the difference.

Responsive **Design**

Responsive Web Design

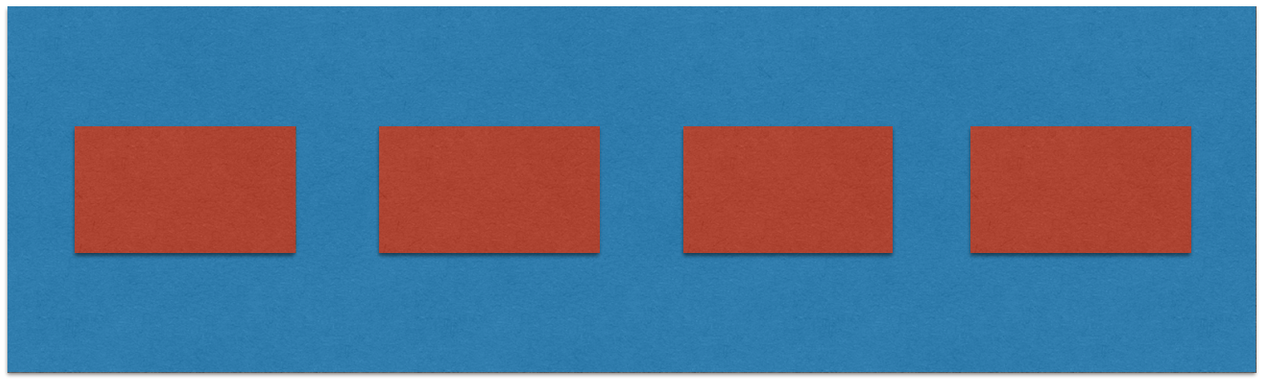
Responsive web design is an approach to web design aimed at allowing desktop webpages to be viewed in response to the size of the screen or web browser one is viewing with.

The key point is that the design needs to take into account the different screen sizes for that use case.

Anecdote -> We have one site at work that was designed solely for an ipad. One of the founders keeps going to it on his mobile phone and it doesn't look good that way because there was no design made up to handle it.

Why?

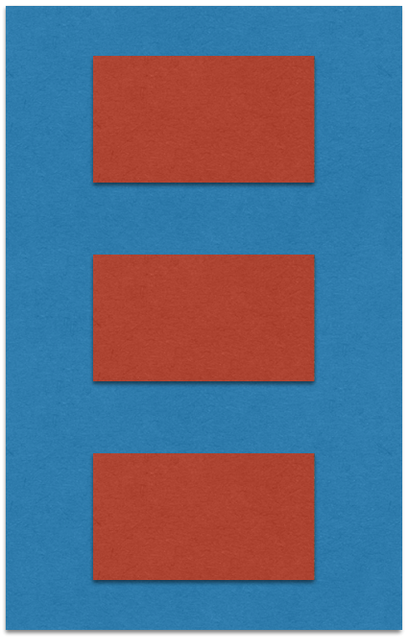
Desktop Sites have a horizontal orientation



On a desktop/laptop we generally have the screen stretched out to a wider layout.

Tablet/Mobile Screens

Have a vertical orientation



On a tablet/mobile site the screen is physically limited to a portrait mode.

We want to adapt our layout for smaller, taller screen size

3 Steps to Responsive

1. **Elastic/Fluid Foundation**
2. Flexible Content
3. Media Queries

1) elastic or fluid foundation to our layout
2) make sure the content is also flexible and fits changes to the layout
3) use media queries to make larger changes when screen sizes start breaking the harmony of the pages

Fluid foundation not fixed

- The web is not fixed-width
- Move from pixel-based grid to percentage based grids

Change pixels to percentages
Start with the math 200px/container size * 100 = percentage

Fluid foundation

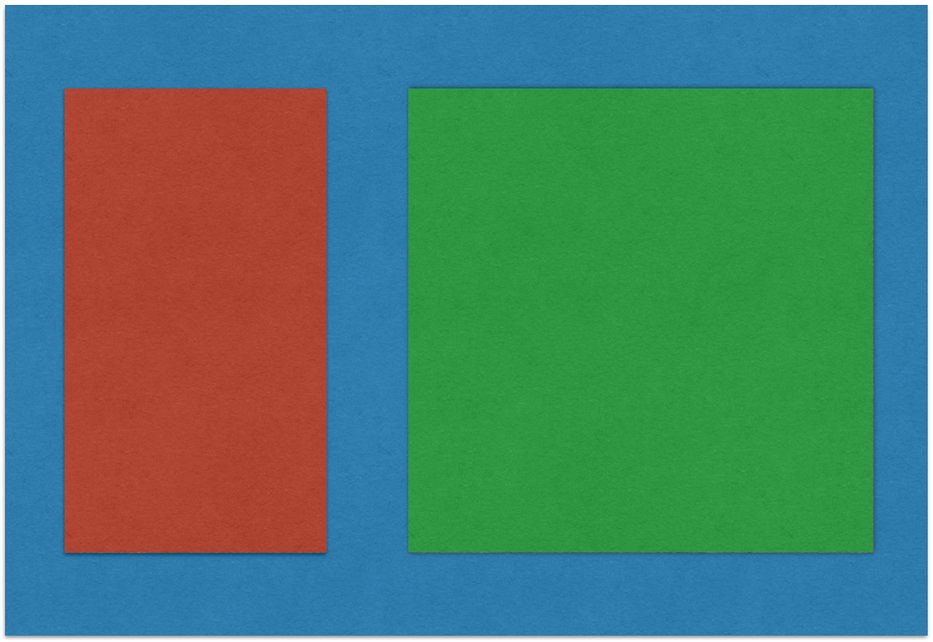
```
    //fixed-width
.item1{
  width: 60px;
}

   // fluid layout
.item1{
  width: ??;
}
```

if our container is 300px, width = 60/300*100= **20%**

Hands-on review

Container Width: 980px, Height: 400px



01_exercise_review

Relies on a container of fixed width(Usually 960px or 980px)

Holding two divs that are side by side

left div is 200px

right div is 700px

Code along

- Change our review code from fixed width to fluid layout
- Change fixed width to percentage

Steps to Responsive

1. Fixed to Fluid
2. **Flexible Content**
3. Media Queries

Once we have a flexible layout system, then the content needs to adjust to the sizes.

Flexible Content

- When we expand or contract our text. It tends to adjust to size of it's container(and can overflow it)
- images and video don't

image width?

- exercise 03 - code along

add max-width: 100%;
change to width: 100%;
finally show as a percentage

Exercise 04

- Use the base layout to create a flexible layout that looks like the boxes.png file
- Ignore the nav bar

Steps to Responsive

1. Fixed to Fluid
2. Flexible Content
3. **Media Queries**

Harmony

- When our content and design no longer work in harmony, we need to make a bigger shift in our layout

Screen sizes and widths

480 pixels // phones

768 pixels // tablets(portrait mode)

960 pixels // small screens

1200 pixels // big screens

—> As a general rule, target your styles against these breakpoints.

07_media_query_demo

How to target screen size

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

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

screen - Used for computer screens, tablets, smart-phones etc
min-width - The minimum width of the display area, such as a browser window

Other ways to include

- <link rel=“stylesheet” media=“screen and (min-width: 1024px)” href=“css/1024only.css”
- Can use a separate css sheet. I use media queries inline as they relate to one group of html objects

One more thing

<meta name="viewport" content="width=device-width, initial-scale=1.0”>

This is the responsive meta tag

Without this meta information, when we use different screens we won’t know the device screen size itself.

This was invented by apple, incidentally.

06_media_query_viewport

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.

This means that the browser will (probably) render the width of the page at the width of its own screen. So if that screen is 480px wide, the browser window will be 480px wide, rather than way zoomed out and showing 960px (or whatever that device does by default, in lieu of a responsive meta tag).

Exercise

07_media_queries

Add breakpoints for 768px and 480px.

Test using the chrome device simulator

I will walk through the first part. The second part is for you guys to do.

EM

- em is based on elements parent size
- The default text size in browsers is 16px. So, the default size of 1em is 16px.
- w3c recommends using em’s

The important thing to note is that the em is relative to the most recent item and not the root of the html

08_em

Make note that the html root is 16px. However, the calculations are at 15px which the the body’s elements

EM as computed value

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

header{
  font-size: 2em;  // 12*2 = 24px
  padding-left: 4em; // 12*4 = 48px
}

.item{
  font-size: 1.5em; // 12*1.5 = 18px
}
```

EM = Elastic Design

- A lot like our fluid foundation…but called elastic because it's based on font-size.

EM

```
.article{
  font-size: 2em;
}
```

If article is inside a div that has the font-size set to 20px. How big will the font-size be for article?

Answer: 40px

REM

- Based on the font-size of html element

09_rem

Less flexible inside of each grouping of items you are using.

REM

```
.article{
  font-size: 2em;
}
```

If the html has font-size set to 12px. And the div containing article above is 20px. What is the font-size for article?

Answer: 24px;

Caveats to em/rem

- Some browsers have issues with fonts sized in percents
- The idea that setting the font size for different media queries will just be automatic.

My own usage

- In my experience, I have used ems and rems sparingly.
- I can achieve and maintain everything using percentages and pixels
- Em’s and rem’s will hide size issues and obscure what is happening

Further info on em/rem

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

EM/REM Patience:
http://zellwk.com/blog/rem-vs-em/

Flexible boxes exercise

- Flexible boxes exercise
- Make this flexible layout fit an iphone and ipad