FEWD - WEEK 7

WILL MYERS

Freelance Front End Developer

SLIDES

http://www.slideshare.net/wilkom/fewd-week7-slides

FINAL PROJECT MILESTONE 2

By week 7, you should have submitted pseudo code and a draft of the HTML/CSS for your application. This week, you'll focus on drafting the JavaScript and jQuery you'll need for your application.

YOUR WEEKLY FEWD GITHUB REPOSITORY

- Use the '+' button in the top-left of GitHub Desktop (*Create* tab)
- Create a new repository called *'FEWD_Week7'*
- Choose the [home]/FEWD folder for the local path
- Open this repo folder in your editor
- Commit the changes and publish the *FEWD_Week7* repository to github.com

YOUR WEEKLY WORKING FILES FROM ME

To get the *week7_working_files* you should just be able to select the *ga-fewd-files* repository in GitHub Desktop and press 'Sync'. This should pull in this weeks folder from github.com.

If you any difficulties you should just re-clone the *ga-fewd-files* repository.

Copy the whole *week7_working_files* into your *FEWD_Week7* repository and commit and publish to github.com

REVIEW OF LAST WEEK'S ASSIGNMENT

DEBUGGING

Why isn't this working?

DEBUGGING

Always start be defining the problem.

- "The image is not moving"
- "None of my code works"

DEBUGGING

This will tell you where to start your hunt

- Image not moving
    - find the code that makes the image move
- None of my code works
    - Syntax error, check console

DEBUGGING: LEVEL 1

Check for errors (red text, aligned right) in console To access debugging console

```
PC: CTRL+SHIFT+J
Mac: COMMAND+OPTION+J
```

Click the error

The location may not be correct but is a good place to start Ex: Unbalanced brackets or parentheses

DEBUGGING: LEVEL 2

So no red errors but not getting the right answer? Try console.log

Ex:

```
var stringOfNames="";
["Bob","Joe"].forEach(function(element){
    stringOfNames-=element+",";
    console.log(stringOfNames);
});
```

DEBUGGING: LEVEL 3

- Use the debugger in Chrome
- Set a breakpoint
- Run the code
- Step through the code until you get to the error
- Variable values display on the right
- You can switch to the console to run code or check value of variable

DEBUGGING: LEVEL 4

Get help!

1. Try "Your preferred search engine" search
2. Be ready to clearly articulate the problem (Write out what your problem is)
3. If nothing, ask instructor

</>
CODE ALONG

DEBUG

Open *week6_working_files/debug*

AGENDA

This week we are bringing HTML/CSS back into the equation. Let's return to responsive layouts and units as discussed in Week 3.

- Responsive Layouts
- REM/EM
- Media Queries

Let's read through the refresher notes for Week 3 again. Go to:

https://gist.github.com/wmyers/d2fedd6f2a52d272ad9e

A BIT MORE ON EMS VS REMS

em: a sized based on the width of the letter "m"

rem: a "root" em

Some browsers have issues with fonts sized in percents, `em` is better for fonts.

LAYOUTS - STATIC VS LIQUID VS ADAPTIVE VS RESPONSIVE

Did anyone read this web page?

http://blog.teamtreehouse.com/which-page-layout

LAYOUTS - ELASTIC

One other layout options is to use `em` for layout widths as well as for font-sizes. This means that the page will resize according to the end user's **preferred text size**.

This blog post from 2009 is still useful for explaining *fixed* versus *fluid* versus *elastic* layouts (though most of the example links are no longer correct): https://www.smashingmagazine.com/2009/06/fixed-vs-fluid-vs-elastic-layout-whats-the-right-one-for-you/

But remember Flexbox is slowing making these older ways of doing responsive websites obsolete.

BOXES EXERCISE

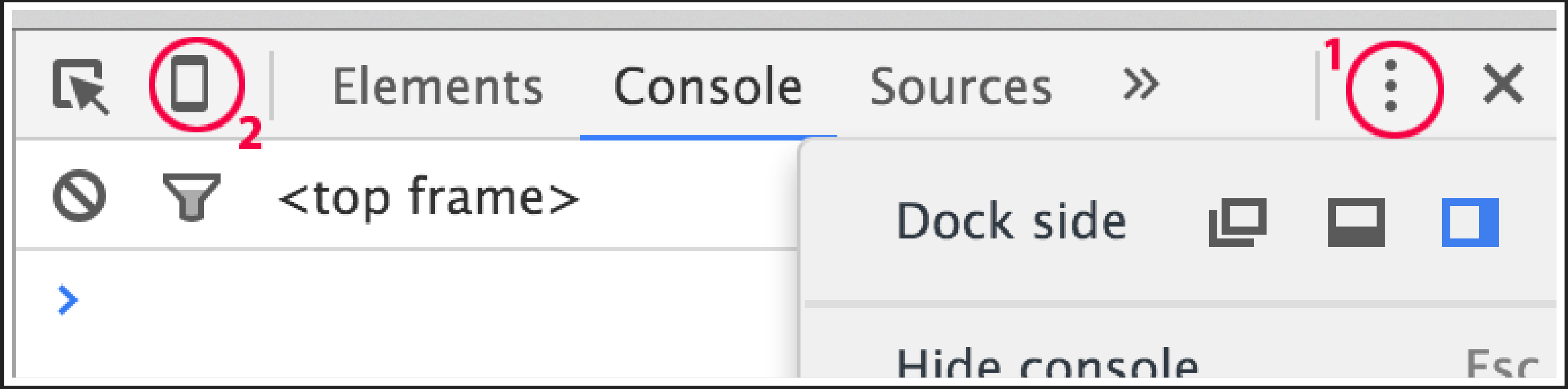
Let's start building a layout using boxes. We will eventually make this layout responsive. Open *week7_working_files/responsive_boxes/readme.pdf* in your browser. You can use `float` or `display:inline-block` or Flexbox, for columns.

Recreate *Boxing 1* using html and css. You can use the *week3_working_files/layout_challenge/5.magazine* files as a starting point if using Flexbox. Otherwise use one of the *1.two-column* layouts. Copy the files into *week7_working_files/responsive_boxes* folder.

Note that you need to have a **hidden element**.

RESPONSIVE DESIGN IS MORE DESIGN THAN CODE.

- Go to this website http://www.anderssonwise.com/
- Open Chrome Dev Tools (CTRL + Alt + I)
- Set the Dock Side to the right and then click on the mobile icon



Also look at the site on your phone browser. Discuss what is making the site responsive.

FIXED VS RESPONSIVE

Looking for comparisons of fixed versus responsive layouts.

Checkout these **Fixed** sites

- UPS.com
- Google.com

Checkout these **Responsive** Sites

- Generalassemb.ly
- Designmodo examples

MOBILE BOXES

Continuing with our boxing exercise. Keeping the same HTML, add/overwrite classes in your CSS, below your existing code, to make the boxes look like *Boxing 2*.

FIXED LAYOUT

- We have used fixed layouts in our designs up to this point - e.g in Relaxr pages
- Relies on a container of fixed width - usually 960px or 980px
- Has `margin:0 auto` (or similar) to center the container

RESPONSIVE LAYOUT

- Different styles for different screen widths
- Uses an elastic/fluid layout
- Fluid - Sized in percentages
- Elastic - Sized in ems

MEDIA QUERIES - USAGE

Please read the first part of this page regarding syntax.

For a list of `@media` types: https://developer.mozilla.org/en-US/docs/Web/CSS/@media#Media_types

For building responsive web pages you will use the following syntax:

```
@media only screen and
(max-width: Npx)  (min-width: Npx)  (max-device-width:Npx)  (min-device-wid
```

For tablets

```
(orientation: portrait)  (orientation: landscape)
```

MEDIA QUERIES - USAGE

Separate multiple clauses with "and"

```
@media only screen and (max-width: 769px)
and (orientation: landscape) {
...
}
```

Standard media queries sizes

- Small: up to 768px
- Medium: 769-991px
- Large: 992px+

MOBILE DISPLAY - VIEWPORT

The viewport meta tag (placed in the `<head>`) lets the browser know that it has to scale your webpage responsively on every device.

Combining the viewport meta tag with responsive CSS will make your web page work across different mobile browsers.

Otherwise the mobile browser will assume a fixed layout of between 800px to 1024px, and the user will have to zoom in to different sections of the page.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

VIEWPORT - DIFFERENT PIXELS

See this page to understand how *device independent pixels* are required to enable the viewport meta tag to scale a mobile browser page correctly:

https://developers.google.com/speed/docs/insights/ConfigureViev
hl=en

VIEWPORT - USER SCALING

There is an additional attribute for the viewport meta tag: `user-scalable=none`

```
<meta name="viewport" content="width=device-width,
initial-scale=1 user-scalable=none">
```

This prevents the user from using default browser zoom functionality (including pinch-zooming) - the implication is that responsive CSS and the standard use of the viewport meta tag should make zooming unnecessary.

However this is essentially anti-accessibility: http://codepen.io/absolutholz/post/user-scalable-no-evil-or-slightly-not-evil

VIEWPORT - 300MS TAP IN ANDROID

On the flip-side of the argument against using `user-scalable=none`, a couple of years ago, if you *did* use it then (in Android) you would remove a mandatory 300 millisecond delay caused by waiting to detect a user **double-tap**.

https://developers.google.com/web/updates/2013/12/300ms-tap-delay-gone-away

Following more recent changes, Chrome and Firefox now remove the 300ms tap delay if your web page contains the `content="width=device-width"` attribute instead. This means you can retain pinch zooming.

MOBILE DISPLAY - 300MS TAP IN IOS

Mobile Safari has a more complicated approach to solving the 300ms delay by detecting for *fast* taps and *slow* taps:

http://developer.telerik.com/featured/300-ms-click-delay-ios-8/

However there is a JavaScript library called FastClick which can be included which can be used to remove the 300ms click delay across all mobile browsers:

https://github.com/ftlabs/fastclick

MEDIA QUERY USAGE

```
/*inline boxes into columns*/
.box{
    display:inline-block;
}
@media only screen and (max-width:768px){
    /*insert responsive css here
    eg: stack inline boxes*/
    .box{
        display:block;
    }
}
```

If I put the media query before

```
.box{
    display:inline-block;
}
```

will this work as expected? See this link.

RESPONSIVE BOXES

Returning to our boxing layout, use media-queries to make your boxing web page responsive so that it changes from *Boxing 1* to *Boxing 2* when the browser window width is reduced.

TRANSITIONS TO MAKE MEDIA QUERY CHANGES SMOOTHER

We will look more at CSS animations and transitions in Week 9 and Week 10.

Feel free to research this further in the meantime.

LAYOUTS - GRIDS

Grids are ways of creating complex fixed or fluid layouts, involving nested rows and columns. Many grids have up to 12 columns and unlimited rows. NB Grids tend to use `floats` for creating columns.

This article shows you how to create a responsive fluid grid layout that uses media-queries to change to a single column layout when the page gets too small.

http://www.sitepoint.com/understanding-css-grid-systems/

http://codepen.io/SitePoint/pen/dPqqvN

GRIDS - BOOTSTRAP

Grids are often implemented using popular third party CSS *libraries*, that provide tested cross-browser complex layouts that developers can use out of the box.

The Bootstrap CSS library has grid layout classes (available when you include and link bootstrap.css in your `<head>`):

http://www.sitepoint.com/understanding-bootstrap-grid-system/

The Bootstrap grid system is also responsive. You apply different styles to your column `<div>`s and the relevant one will be automatically used as your window resizes.

GRIDS

It is not actually that hard to build your own grid. In fact we have already been doing so with things like the *magazine layout* in Week 3.

A simple column layout is still a grid, more complex grid layouts can still utilize simple CSS layout techniques: https://css-tricks.com/dont-overthink-it-grids/

(NB: this layout uses `floats`. We could use `display:inline-block` or Flexbox instead.)

LAYOUTS - THE NEW GRID DISPLAY PROPERTY

There is also a new `display:grid` property, which is a recent CSS addition like `display:flex`. At this time it is still better to use Flexbox (`display:flex`).

AGENDA

How to make a website responsive.

LEVEL ONE

- Convert widths to percentages
- Size fonts in ems
- Identify columns that can be stacked

LEVEL TWO

- Determine what content is extra
  - Does that h1 need an h2 under it?
  - Will that one liner get the visitor to understand my site or is it extra?
- Determine what content needs to be visible
  - Is it necessary to show my entire nav if there are other ways of getting to navigation?
- Identify extra styling
  - Do my columns need a border if they're stacked?

LAB

STARTUP MATCHMAKER