FEWD - WEEK 3

WILL MYERS

Freelance Front End Developer

SLIDES

http://www.slideshare.net/wilkom/fewd-week3-slides

YOUR WEEKLY FEWD GITHUB REPOSITORY

- Use the '+' button in the top-left of GitHub Desktop (*Create* tab)
- Create a new repository called *'FEWD_Week3'*
- Choose the [home]/FEWD folder for the local path
- Open this repo folder in your editor
- Commit the changes and publish the *FEWD_Week3* repository to github.com

YOUR WEEKLY WORKING FILES FROM ME

To get the *week3_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 *week3_working_files* into your *FEWD_Week3* repository and commit and publish to github.com

AGENDA

- Review
- Assignment preview
- Units of measurement
- Layouts
- Flexbox

ASSIGNMENT FOR TODAY

Continue with Relaxr!

Startup Matchmaker

UNITS OF MEASUREMENT

UNITS OF MEASUREMENT - PIXELS

Pixels are **fixed** units of measurement. They are intended to fix a layout width/height or font-size to a specific number of pixels in the screen resolution of the end user device.

Remember that some devices have much higher resolution than other devices.

UNITS OF MEASUREMENT - PERCENTAGES AND EMS

Percentages and ems are **relative** units. Percentages are *relative to the **size** of the parent container of an element*.

Ems (when used with the `font-size` property) are *relative to the current element's inherited **font-size***.

It is best practice to use `em` for font sizes and `%` for element sizes. Although you can use `em` for relative sizing of dimensions and `%` for fonts.

FONT-SIZE UNITS

Open *week3_working_files/font-sizes/page.html*

Open Chrome Dev Tools and select the *Computed* tab on the right.

https://developer.mozilla.org/en/docs/Web/CSS/font-size

https://css-tricks.com/css-font-size/

FONT SIZES WITH EMS

Using `em` with font sizes means that the fonts on a page will automatically scale to an end user's preferences (normally the default values of a browser).

`1em` is by default equal to the `font-size` of the page root `<html>` tag. The most common default `font-size` for the root tag is `16px`. (**if not using *reset.css***)

FONT SIZES WITH EMS

It is common practice to set the `<html>` or `<body>` tag `font-size` to `0.625em` or `62.5%`.

This will convert a default `16px` to `10px`, which makes it easier to apply relative `em` values in a decimal way.

`1em` now equals `10px`

`2.2em` now equals `22px` etc

COMPOUND RELATIVE UNITS OF MEASUREMENT

Remember that `em` values **cascade** through the DOM. So an `em` value on a child element will be relative to any `em` value on a parent element.

Nested relative units of measurement will **compound** each other. A child element with `%` or `em` style values will calculate relative to the font-size and dimensions of the parent element. If the parent element also has `%` or `em` values then a compound calculation will occur.

REMS INSTEAD OF EMS

`rem` gets around the compounding problem of `em`. It is only relative to the `font-size` of the root `<html>` tag.

Remember to set your root `font-size` on the `<html>` tag, not on the `<body>` tag.

FONT SIZES WITH KEYWORDS

You can also set font-sizes with **keywords**. There are seven **absolute** values ranging from *xx-small* to *xx-large*. There are two **relative** values - *smaller* and *larger*.

The root default absolute keyword value is *medium* (16px), which you can override. You can set relative values to any absolute parent container value, throughout the DOM.

Open *week3_working_files/font-keywords/page.html*

LAYOUTS

LAYOUTS - FLOATED COLUMNS VS INLINE-BLOCK COLUMNS

Do you remember `display:inline-block` from Week2 Refresher notes?

Open *week3_working_files/layout_challenge/1.two-column/inline-columns* and *float-columns*

Inline-block display requires `font-size` values to deal with inline-block `whitespace`. It also requires a `vertical:align:top` style declaration.

LAYOUTS - STATIC VS LIQUID VS ADAPTIVE VS RESPONSIVE

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

STATIC LAYOUTS

This is a "fixed width" layout across all platforms - commonly 960 pixels for a 1024x768 px resolution. Pixel units of measurement are **static** (fixed).

The dimensions of the page will not change, but modern browsers on mobile devices (like iPhones) will scale the page to fit the screen, the user then zooms in on different parts of the page.

LIQUID LAYOUTS

A liquid or "fluid" layout uses **relative units** rather than fixed units. Commonly percentages `%` are used. However you could also use `em`.

A fluid layout width (and/or) height commonly fills the screen, and automatically adapts to different platform/browser resolutions. Problems occur when a page is too wide or too narrow - content can be stretched too much or a multi-column layout can be too compressed.

ADAPTIVE LAYOUTS

An adaptive page layout is **static** but uses **CSS media queries** to roughly detect the width of the browser and alter the layout to a best-fit static layout.

Open *week3_working_files/media-query*

MEDIA QUERIES

Media queries are expression of logic inside a CSS style declaration:

```
.container{
  width: 960px;
}
@media all and (max-width: 500px) {
  .container{
    width: 400px;
  }
}
```

"If the browser 500px wide (or less), set container to be 400px wide. Otherwise container will be 960px wide."

`max-width: 500px` is the **breakpoint**

RESPONSIVE LAYOUTS

A responsive page layout combines a liquid layout and an adaptive layout.

A page will use relative units within certain page dimensions defined by media queries. When a media query breakpoint is reached (by changing the browser window size), the page layout will **significantly change**, e.g. items in a row will switch to become items in a column.

RESPONSIVE LAYOUTS

Responsive layouts are commonly the norm for web pages, so you can have one page that works correctly on both mobile and desktop devices.

Responsive designs are also mobile-first, meaning the mobile phone layout takes precedence and becomes wider for mobile tablets and desktop devices.

We will look further at media queries and responsive layouts later in the course.

FLEXBOX

Flexbox makes it easier to build responsive layouts. It reduces the need for multiple media queries and complex CSS.

However it is complex functionality that has gone through multiple revisions in recent years as different browser vendors agree on its functionality.

A good start is to play http://flexboxfroggy.com/ and to read https://css-tricks.com/snippets/css/a-guide-to-flexbox/

FLEXBOX

Open *week3_working_files/layout_challenge/2.three-column/flexbox-or-inline-columns*

Open *week3_working_files/flexbox-examples*

Flexbox can do things like:

- Vertically centering an item http://codepen.io/wilkom/pen/NGeyNg
- Layout of items in a row or column with `flex-wrap`
- Mix static and relative sizes of elements in a container with `flex-grow`