WEB DEV 2 - WEEK THREE

MEDIA QUERIES

QUOTE

- “You must be shapeless, formless, like water. When you pour water in a cup, it becomes the cup. When you pour water in a bottle, it becomes the bottle. When you pour water in a teapot, it becomes the teapot.”

- Bruce Lee

WHAT ARE MEDIA QUERIES?

- Specific styles that are applied under certain circumstances.

MOBILE DEVICES

- Mobile devices aren’t a 1:1 pixel match
  - device width vs. pixel width
- Device width * pixel density = layout width

VIEWPORT

- Viewport - visible area of the browser
- http://screensiz.es/phone

VIEWPORT TAG

- <meta name=“viewport” content=“directive, directive”/>

VIEWPORT PROPERTIES: WIDTH

- Width: lets you set the viewport to a specific width, or to the width of the device.

- The ideal solution is:

- <meta name=“viewport” content=“width=device-width” / >

- Anytime the width of the viewport != the device width the device will scale it up or down to fit

VIEWPORT PROPERTIES: HEIGHT

- Height: lets you set the viewport to a specific height, or to the height of the device.
- Height can be used to prevent the page from scrolling vertically
- Best bet is to stick to the device width, and let the height flow naturally

VIEWPORT PROPERTIES: USER SCALABLE

- User scalable: whether or not the user can zoom in and out on the page.

- <meta name=“viewport” content=“user-scalable=no” />

- This can be detrimental to users with accessibility needs

VIEWPORT PROPERTIES: INITIAL SCALE

- Initial scale: sets the initial zoom level of the page
    - Number, between 0.1 (10%) and 10.0 (1000%)
- <meta name=“viewport” content=“initial-scale=1” />

VIEWPORT PROPERTIES: MAXIMUM SCALE

- Maximum-scale: sets how far the user can zoom in on a page
- Can go between 0.1 (10%) and 10.0 (1000%)
- As with user scalable, if you set the maximum scale declarative to 1.0, you disable the user's ability to zoom and limit your site's accessibility.

VIEWPORT PROPERTIES: MINIMUM SCALE

- Minimum-scale: sets how far the user can zoom out on a page

- Can go between 0.1 (10%) and 10.0 (1000%)

- As with user scalable, if you alter the minimum scale declarative, you disable the user's ability to zoom and limit your site's accessibility.

VIEWPORT:
BEST PRACTICE

- <meta name=“viewport” content=“width=device-width” />
- More reading:
  - http://dev.opera.com/articles/view/an-introduction-to-meta-viewport-and-viewport/

FUTURE OF VIEWPORTS

- To be written in your CSS file (already supported in IE10 and Opera)
    - @viewport { properties }

WHY ALL THIS TALK OF VIEWPORTS?

- **We need the viewport tag in order for media queries to work properly.**

- Ex:

  - A media query meant to kick in at a width of 360px, would never trigger on a Google Nexus 4. It's pixel width is 1080!!

MEDIA QUERY, STRUCTURE

- @media [not|only] type [and] (expression) { rules }

4 BASIC COMPONENTS

- **media types:** type of device to target
- **media expressions:** test against a feature and evaluate to ‘true’ or ‘false’
- **logical keywords:** let you create more complex expressions (and, or, not, only)
- **rules:** basic styles that adjust the display

MEDIA TYPES

- all (default)
- braille
- embossed
- handheld
- print
- projection
- screen
- speech
- tty
- tv

IMPLEMENTING MEDIA TYPES

- In CSS file:
  - @ media print { }
- In HMTL file:
  - <link rel=“stylesheet” media=“print” href=“print.css” />

SCREEN MEDIA TYPE

- Rather than use handheld, projection, tv, etc…
- Just use **screen**

IMPLEMENTING MEDIA EXPRESSIONS

- @media screen and (min-width: 320px) { rules }
- media expression = min-width: 320px
- rules are applied when expression returns true

TYPICAL MEDIA EXPRESSIONS

- width
- height
- orientation (portrait | landscape)
- resolution
- aspect-ratio

LOGICAL KEYWORDS

- **and** - test for more than one expression
- **not** - negate an expression
  - must preface the entire query (@media not…)
- **or** - used by comma (,) to join multiple queries. any can return true to apply rules
- **only** - older browsers support media types, but not queries. only will prevent styling errors

AND

```
@media screen and (min-width: 320px) {

    color: red;

}
```

NOT

```
@media not screen and (orientation: portrait) {
    float: left;
}
```

OR

- @media screen and (min-width: 640px), screen and (orientation: landscape)

ONLY

```
@media only screen and (min-width: 1024px) {

    float: left;

}
```

EMBEDDED VS EXTERNAL

- **embedded:** inside the css file
- **external:** reference to a css file
  - <link href=“style.css” media=“only screen and (min-width: 1300px)” />

MEDIA QUERY ORDER: DESKTOP DOWN

```
@media all and (max-width: 768px) { }

@media all and (max-width: 320px) { }
```

MEDIA QUERY ORDER: MOBILE UP

```
@media all and (min-width: 320px) { }

@media all and (min-width: 768px) { }
```

MOBILE UP IS PREFERRED

- Often mobile inherits ‘defaults’
- Can result in less code being written
- Supports the cascade, from simpler —> complex

DESKTOP DOWN EXAMPLE

```
aside {

    display: table-cell;
    width: 300px;

}

@media all and (max-width: 320px) {

    aside {
        display: block;
        width: 100%;          (necessary to override previous rule)
    }

}
```

MOBILE UP EXAMPLE

```
@media all and (min-width: 320px) {

    aside {
        display: table-cell;
        width: 300px;
    }

}
```

CREATE THE CORE EXPERIENCE

- One column layout
- http://www.implementingresponsivedesign.com/ex/ch3/ch3.1.html

NAVIGATION

- Shouldn’t take up precious real estate.
- Should be intuitive and simple.
- Should be usable across devices.
- http://bradfrostweb.com/blog/web/responsive-nav-patterns/

NAVIGATION OPTIONS

- Dont’ do anything.
  - easy way out, a lot of room for nag
- Select
  - compact, can be confusing.
- Toggle
  - Ideal, intuitive.