8/24/17

* Day 1
* NOTE: Day 1 should have been on 8/27 following the recommended schedule.
* Relative sizing: make elements adjust automatically as the browser changes sizes
* Relative units of measurement
* Responsive design is resizing and reorganizing content based on
  + The size of the screen used to view the page
  + And the size of other content on the page
* Using px (pixel) quantities to define size is a hard coded value and doesn’t work across all screen sizes. We can use relative measurements instead to allow for portions of a site to look the same regardless of screen size or layout
* Em
  + Em represents the size of the base font being used
    - So if the base font is 16px on a browser, then 2em would be 32px
  + Em can also be used not referring to the default base text size, if used in a CSS rule related to another defined font size:
    - .splash-section {
    - font-size: 18px;
    - }
    - .splash-section h1 {
    - font-size: 1.5em;
    - }
    - (NOTE – the h1 of splash-section would be 27px)
* Rem
  + Stands for “root em”
    - Similar to em, but instead of checking the parent element for the font size base, it checks the root element (that is <html>)
  + Most browsers set the default <html> font size to 16px
  + Convenient because all rem sized elements are compared to the same base font size
* Percentages for height and width
  + Used to size **non-text** elements on a page
  + Percentage sizes are relative to the parent container
  + NOTE: need to set the size of a parent container first, otherwise the percentage will have nothing to refer to
  + NOTE: 100% sizing should only be used when content will not have padding, borders, or margins
* Percentages for margins and padding
  + Calculated only based on the width of the parent element, **even** for vertical margins and padding
  + **Note**: When using relative sizing, ems and rems should be used to size text and dimensions on the page related to text size (i.e. padding around text). This creates a consistent layout based on text size. Otherwise, percentages should be used.
* Width: MIN and MAX
  + Min-width: ensures a minimum width
  + Max-width: ensures a maximum width
  + NOTE: The unit of pixels (px) is used with max and mins because it ensures a limit on the dimensions
* Same for min and max height
* Scaling images and video
  + Very common way to scale images and video proportionally:
    - .container {
    - width: 50%;
    - height: 200px;
    - overflow: hidden;
    - }
    - .container img {
    - max-width: 100%;
    - height: auto;
    - display: block;
    - }
* Scaling background images
  + body {
  + background-image: url('#');
  + background-repeat: no-repeat;
  + background-position: center;
  + background-size: cover;
  + }
  + **Common way^^^^**

**DAY 2**

* Quiz on sizing elements
  + Got a 16/16 on second try
* Lesson on responsive design with different size screens:
* Media Queries
  + CSS uses media queries to adapt to different screen sizes
  + CSS can use them to detect the size of the screen
* @media only screen and (max-width: 480px) {
  + Body{
  + Font-size: 12px;
  + }
* }
* ^^That’s how a media query is applied^^
* That example up there is a rule for screens smaller than 480 px wide
* @media is the keyword to begin a media query
* “only screen” is just the common media type for displaying any content, no matter they device
* Range
  + How to create a media query rule for a range of screen sizes:
  + @media only screen and (min-width: 320px) and (max-width: 480px) {
  + /\* ruleset for 320px - 480px \*/
  + }
* Dots Per Inch (DPI)
  + Sometimes you want to give higher-end device users higher-res pictures
  + We can target resolution with min-resolution and max-resolution
    - media only screen and (min-resolution: 300dpi) {
    - /\* CSS for high resolution screens \*/
    - }
  + That rule targets high resolution screens with only 300 or more DPI
* And Operator
  + The “and” operator can be used to chain multiple rules together (min resolution and min width, for example)
* Comma Separated List
  + (Instead of using “and”) Separate your list of media query targets with a comma to state that only one of those requirements needs to be met to trigger the ruleset
* Breakpoints are screen sizes at which your site doesn’t work properly

8/30

Day 3

* Quiz on media queries
* Got a 7/7 on first try
* User agent style sheet
  + A browser’s default styling for all of the html elements
  + Can be different across browsers
* You can reset the user agent style sheet with a CSS reset
  + This removes all browser-added styling so that all browsers start at the same point, helping compatibility
    - Create a reset.css file.
    - Copy and paste CSS reset rules into the reset.css file.
    - Link to reset.css in the HTML file (make sure reset.css is loaded first before other CSS files, otherwise reset.css may reset your custom rules by accident).
      * <html>
      * <head>
      * <title>My Title</title>
      * <link href="reset.css" type="text/css" rel="stylesheet">
      * <link href="styles.css" type="text/css" rel="stylesheet">
      * </head>
    - Popular reset sheet here
      * <http://meyerweb.com/eric/tools/css/reset/>
* Vendor prefixes
  + All browsers have their own implementations of many of the newer CSS stylings
  + To specify that you want to use a specific browser’s implementation
    - Use a vendor prefix like “-moz” for firefox
* Polyfills
  + Libraries that detect browser capabilities
  + Then you can detect if a user is using an old or non-functional browser and you write CSS rules for those cases to keep the site working for them
* Quiz on browser compatibility
  + Got a 5/5 first try
* Project: Tsunami Coffee
* Project: Secret Agent Supply