



## WEEK 2

35

## WEEK 1 REVIEW

---

1. What is the important step to connecting your domain to your hosting?
2. The request from the browser to the server is called a/an \_\_\_\_\_ Request
3. Client-side code runs at the browser- identify a reason to run code at the client.
4. Server side code (also back end) executes at the server. What is usually produced from server side code?
5. In a url, http or https are the \_\_\_\_\_



36

## REVIEW

---

1. The difference between an HTML tag and an attribute is?
2. The tag to create a hyperlink is \_\_\_\_\_
3. When the href of a link is set to “#”, it means \_\_\_\_\_
  - Why would that be desirable?
4. Which of the following tags, *by convention*, should NOT go in the body:  
[link]      [table]      [meta]      [title]
5. The tag to create a bulleted list is?
6. To specify the number of columns in a table do this: \_\_\_\_\_



37

## WEB DEVELOPMENT: THE PROCESS

---

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li>▪ Planning<ul style="list-style-type: none"><li>▪ “Begin with the end in mind”<br/>- Dr. Stephen R. Covey</li><li>▪ Users/Goals /Desired results</li><li>▪ Deliverable: Design Brief</li></ul></li><li>▪ Content<ul style="list-style-type: none"><li>▪ Calls to action</li><li>▪ Info architecture</li><li>▪ SEO Considerations</li><li>▪ Deliverable: Content document</li></ul></li><li>▪ Branding choices<ul style="list-style-type: none"><li>▪ Deliverable: Style Guidelines</li></ul></li><li>▪ Design<ul style="list-style-type: none"><li>▪ Photoshop, Canva, paper</li><li>▪ Deliverable: “Flat” design</li></ul></li></ul> | <ul style="list-style-type: none"><li>▪ Logistics<ul style="list-style-type: none"><li>▪ Tech choices / API's / Database</li><li>▪ Purchase domain</li><li>▪ Set up hosting for staging site</li><li>▪ Architect/set up database</li></ul></li><li>▪ Implementation<ul style="list-style-type: none"><li>▪ Build the front end (html/css)</li><li>▪ Add client-side code</li><li>▪ Add server-side code</li><li>▪ Test Offline and online</li><li>▪ Deliverable: Working draft of site</li></ul></li><li>▪ Launch<ul style="list-style-type: none"><li>▪ Move site if needed to permanent hosting and attach domain if needed</li><li>▪ Post launch testing including broken links, page speed optimization, mobile friendly assessment, SEO</li></ul></li></ul> |
|---|--|



38

## HTML FORMS

- `<form>` tag
- `<input>` tag for text, radio, checkbox, submit, reset
- Radio buttons of the same group all have the same name
- `<select>` for drop-down/list
  - `<option>` tag for each drop-down item
- `<textarea>` for multi-line text
  - It's a container tag
  - `rows`, `cols` attributes specify number of rows/columns
- Use CSS or a table to align elements (CSS is best practice)
- Use `id`, `name` to identify elements for use in script

### Pet Information Form

Name:

Create a password:

Where did you hear about us?

☐ Friend  
☐ Internet  
☐ Other

Type of pet: ☒ Dog ☐ Cat ☐ Hamster

Alternate type of pet:

Describe the problem:

39

## FORM METHOD & ACTION

`<form method='get' action='process.php'>`

On Browser:

Display form  
User clicks submit button

On Server:

Run process.php

Processing on server may include:

- Send mail
- Access database
- Return calculation to browser
- Etc.

40

# HTML 5

## Supersedes HTML4.01

Created in cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).

Originally for “catalog” type markup

- Enhanced mobile development
- Can be accessed with or without the internet - important for speed for applications such as online chat
- Can save data to databases locally
- Graphic and visual effects (formerly required Flash)
- Integrates seamlessly with CSS3
- Accessibility
- Video and audio support built in
- Simpler DocType declaration
- HTML 4.01 DOCTYPE
  - `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
- HTML5 DOCTYPE
  - `<!DOCTYPE html>`
- Tags are more descriptive



41

## STRUCTURE OF AN HTML5 FILE



- Aside
  - Somewhat related to the rest of the page.
- Header
  - header of the document
- Footer
  - footer for a document
- Nav
  - Navigation of the document
- Dialog
  - mark up a conversation.
- Figure
  - associate a caption with embedded content



42

## EXAMPLE HTML5 DOCUMENT

---

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset = "utf-8">
    <title> My HTML5 Page</title>
  </head>
  <body>
    <header>
      <h1>Title for My Page</h1>
    </header>
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
    <article>
      <section>
        <p> Sections can be within an article</p>
      </section>
    </article>
    <aside>
      <p>An aside may be styled as a sidebar</p>
    </aside>
    <footer> footer stuff here ... </footer>
  </body>
</html>
```

43

## HTML5: DEPRECATED TAGS

---

- font, center, b, i, u
- frame, frameset
- applet
- marquee, bgsound
- noscript
- See: <http://www.html-5.com/changes/deprecated/>

44

## HTML5: NEW FORM FEATURES

---

- Built-in validation
- Date/time (datetime, date, month, week, time)
- number
- range
- email
- url
- output



45

## CSS (CASCADING STYLE SHEETS)

---

- Define style instructions through style rules that apply to HTML elements
- Developed by the W3C
- Allows formatting to be separate from content
- Within the current page only but a style sheet file can be shared by all pages in a site.

CSS can control:

- |                            |                          |
|----------------------------|--------------------------|
| • Size                     | • Type of font           |
| • Color                    | • Hover effects          |
| • Border                   | • Positioning            |
| • Spacing                  | • Transitions/animations |
| • Variations (italic,bold) |                          |



46

## CSS RULES

---

- Each **rule** has a set of **property/value pairs**
- Rules can be inserted into an HTML command using the **style** attribute  
`<p style="style-property1:value; style-property2:value">`
- This is called an inline style and applies only to the tag to which it is attached
- You can also add style rules within a stylesheet
- A **property** is the style characteristic that are being modified.
- A **value** is the corresponding value for that property.
- Property/value pairs are separated by a semicolon (;)



47

## EXAMPLE

---

`h1 {text-align:right; color: #ff0000;}`

- Selector: h1
- Property: text-align, Value: right
- Property: color, Value: #ff0000
- The entire statement is a **rule** that states that all h1 tags should be right aligned on the page and colored red.



48

## THE “MAGIC” IS IN THE SELECTOR

Type of selector	Example
<b>An HTML tag</b> Applies to any instances of that tag on the page	h1
<b>A style class</b> (starts with a '.') Applies for: class="x" <tag class="x" ...	.my-class
<b>An id</b> (starts with a '#') Applies for: id="x" <div id="x" ...	#the-id
<b>A pseudo-element</b> (starts with a ':') modifies another selector	li:first-child



49

## STYLE RULE POSITIONING

### ▪ **Inline: highest precedence**

- “at the tag”
- <p style="color:#ff0000">

### ▪ **Internal: within <style> tag**

- Lower precedence than inline
- The *closer* a rule is to the selected element, the *stronger* the precedence

```
<style type= 'text/css'>
  h1 {text-align:center;
      color: #ff0000;}
</style>
```

### ▪ **External: In a separate file**

- By convention, style.css
- No style tag
- You may have several external stylesheets
- Include on a page using:  
<link rel="stylesheet" href="styles.css">



50



## SOME CSS TEXT PROPERTIES

---

- **font-family**
  - Serif vs sans serif fonts
- font-size (px or in)
- font-style (italic)
- color
- text-align
- text-transform (uppercase, lowercase)
- text-decoration (set to none to remove all, for example, underline for a link)
- line-height (%, numeric value, px)



51

## CSS STYLES VS HTML TAGS

---

- Using HTML tags only
  - `<p><strong>This text is bold</strong></p>`
- Using CSS:
  - font-weight: 700
- Using HTML tags only
  - `<strong><em>bold and italic </em></strong>`
- Using CSS:
  - font-weight: 700;  
font-style: italic

Use span to define a style without a line break

- Using HTML:
  - `<em>text1</em>`
- Using CSS
  - `<span style= 'font-style: italic'> text1</span>`



52

## METRICS FOR SIZES

---

- px, pt, pc
- in, cm, mm
- %
- auto
- inherit
- em
  - relative to the current font size  
(2em => 2x the size of the current font)

Used for  
height, max-height, min-height  
width, max-width, min-width

Must set display to *block* or  
*inline-block* for those to work



53

## FONTS

---

- Google fonts are easy to use and free! See [fonts.google.com](https://fonts.google.com)
- Fonts from other sources can be added to a site using @font-face if the referenced files are loaded into the site

```
@font-face {  
  font-family: 'RockwellStdBold';  
  src: url('fonts/RockwellStdBold.eot');  
  src: url('fonts/RockwellStdBold.eot') format('embedded-opentype'),  
        url('fonts/RockwellStdBold.woff2') format('woff2'),  
        url('fonts/RockwellStdBold.woff') format('woff'),  
        url('fonts/RockwellStdBold.ttf') format('truetype'),  
        url('fonts/RockwellStdBold.svg#RockwellStdBold') format('svg');  
}
```

- Several font face generator tools exist to make the job easier
  - The one I use: <https://everythingfonts.com/font-face>
- You **MUST** have a license for the font you want to use



54

## CSS COLORS

---

- Use hex, rgb or rgba colors (not color names)
- Hexadecimal colors
  - RGB: red green blue
  - Begin with #
  - #rrggbb
  - Range is 00 to FF (#FF0000 is red)
- rgb() color
  - rgb(12, 120, 255)
- rgba() adds transparency
  - rgba(12, 120, 255, .4)



Colors are used for ...

- Text color
  - color
- Backgrounds
  - background-color
- Border
  - border-color
  - border: 2px solid #123456;



55

## COMBINING SELECTORS: BOTH

---

- Use **a,b** for a rule to apply to 2 or more selectors.

- Example:

```
<h2>This is a heading 2</h2>
```

```
<h3>This is a heading 3</h3>
```

- Goal:

- h2 and h3 are both red

- CSS rule:

```
h2,h3 {color: #DD191C}
```



56

## COMBINING SELECTORS: FOLLOWS

---

- Use **a + b** for a rule to apply to b only when it comes after a.
- Example:

```
<h2>This is a heading 2</h2>
<h3>This is a heading 3</h3>
<h3>Here is another h3</h3>
```
- Goal:
  - When an h3 follows an h2 (perhaps it is a subheading) make its size, 30px
  - In this case, the rule should affect the first h3, not the second one
- CSS rule:

```
h2 + h3 {font-size: 30px}
```



57

## COMBINING SELECTORS: ALL

---

- Use **ab** for a rule to apply to an element with both a and b selectors. This can apply to a tag with a particular class, id or both.
- Example:

```
<h2>This is a heading 2</h2>
<h3>This is a heading 3</h3>
<h3 class='section-head'>Here is another h3</h3>
```
- Goal:
  - Make the h3 with a class of section-head be blue
- CSS rule:

```
h3.section-head {color: #3836CD}
```



58

## COMBINING SELECTORS: CONTAINED

- Use **a b** for a rule to apply to b when b is contained hierarchically within a

- Example:

```
<a href="https://accuweather.com">
  
</a>
<p></p>
```

- Goal:
  - Place a border around images that are contained within an `<a>` tag - but not any other images
- CSS rule:

```
a img {border: 1px solid #3836cd;}
```



59

## COMBINING SELECTORS: ATTRIBUTES

- Use **b [attr]** for a rule to apply only when the b tag has the attr attribute set
- Use **b [attr='value']** for a rule to apply only when the b tag has the attr attribute set to the value

- Example:

```
<form>
  <p>State: <input type='text' value='Massachusetts'></p>
  <p><input type='submit'></p>
</form>
```

A form with a text input field containing 'Massachusetts' and a blue 'Submit' button.

- Goal:
  - Set the font size of all input elements to 20px.
  - Change the submit button to have a blue background with a font color of white
- CSS rules:

```
input {font-size: 20px}
input[type="submit"] {background-color: #3836cd; color: #fff}
```



60

## COMBINING SELECTORS: PSEUDO-ELEMENTS

- last-child the last element hierarchically in a group
- first-child the first element hierarchically in a group
- nth-child(x) x is the index of the element positioned hierarchically in a group
- hover mouse is hovering over an element

- Example:

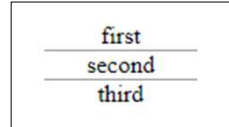
```
<ul>
```

```
<li>first</li>
```

```
<li>second</li>
```

```
<li>third</li>
```

```
</ul>
```



- Goal: Put a border below each item, except for the last one. Items are centered and 100px wide.

- CSS rules:

```
ul {list-style-type: none}
```

```
ul li {border-bottom: 1px solid #999; width: 100px; text-align:center}
```

```
ul li:last-child {border-bottom: none}
```



61

## COMBINING SELECTORS: !IMPORTANT

- **!important** add after any property value pair to give it priority. This will override the usual “cascade” effect which is especially important when working with WordPress themes

- Example:

```
<h2 style='font-size:12px'>This is tiny - or is it?</h2>
```

- Goal: All h2's have a font size of 30px

- CSS rule:

```
h2 {font-size: 30px !important}
```

This is tiny - or is it?

Without important

This is tiny - or is it?

With important



62

## CSS PRO TIPS

- To identify what element needs to change - “inspect element”
- Use !important only as needed.
- Use a private window if caching is an issue
- Use inline styles only for simple style changes that are needed one time only



63



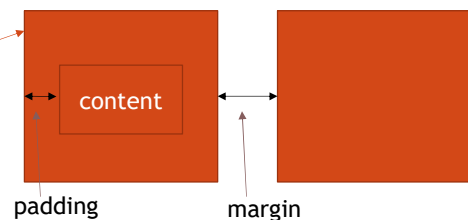
## HTML BLOCKS AND THE BOX MODEL

Blocks are often used to facilitate styling. They can have borders, height, width, margin and padding.

- `<p>` `<div>` `<span>` are containers that can create a plain block (Other tags are also blocks such as headings and lists)
- Other than line spacing, there is no default styling for `div`, `p`, `span`
- You can set display to **block** or **inline-block** on elements that are not already a block to make them a block

### The Box Model

- Border - around an item
- Padding - space between content and boundary of item
- Margin - space between items
- Use **box-sizing: border-box** to set exact width/height unaffected by margin/padding/border



64

## ADDING MARGIN AND PADDING

---

- Specify up to values - top, left, bottom, right (clockwise order)
  - padding: 3px                      3px on all sides  
margin: 3px
  - padding: 3px 5px                3px top and bottom, 5px left and right  
margin: 3px 5px
  - padding: 2px 3px 4px 5px;  
margin: 2px 3px 4px 5px;



65

## MORE BOX PROPERTIES

---

- float (left, right)
- display (inline, block, inline-block, none)
- height, width
- overflow (overflow-x, overflow-y)
- visibility
- z-index



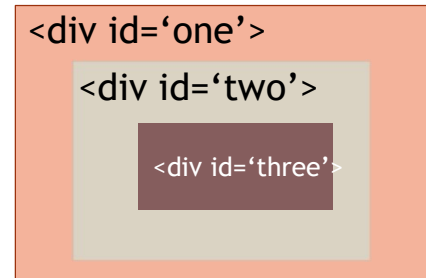
66



## POSITION

---

- Must already be a block (display:block or display:inline-block)
- position: absolute
  - Relative to the first parent element that has *position* set
- position: fixed
  - Relative to the page
  - May also want to use “z-index” property
- position: relative
  - Where the element would normally go on the page
  - position:relative is often used when position needs to be set, but should be neutral



one is parent for two  
two is parent for three



67

## CSS TRICKS

---

- |                           |  |
|---------------------------|--|
| ▪ box-sizing: border-box  | keeps margin/padding within the specified size |
| ▪ margin: 0 auto          | common technique to center a block             |
| ▪ Position:absolute       |  |
| ▪ Set left and right to 0 | centers a block                                |
| ▪ Set top and bottom to 0 | full column height                             |



68

## Transitions

---

- New to CSS3
- transition-property
- transition-duration
- transition-delay
- transition-timing-function

## WEBKIT

---

- Special extensions
- Prefixed with: -webkit-
- Some properties require webkit and non-webkit versions to work on all browsers



69

---

## RESPONSIVE DESIGN


- Responsive sites have all information accessible and easy to read , regardless of the browser width  
i.e., the site is responsive to the browser size.
- It should not be an afterthought
  - Responsive first design
  - Better, Mobile first design



70

## MOBILE CONSIDERATIONS

---

- Menus
  - Wide text-based menus may not be visible or will wrap
  - Best to use a “hamburger” menu 
  - Decide which menu items should be in mobile
- Also, think about ...
  - Logo size
  - Different headers
  - Buttons big enough for fingers



71

## RESPONSIVE DESIGN: HOW TO

---

- Viewport
- Media Queries
  - Inspect
- Adaptive elements
- Flexbox
- Duplicate items
- Emulators: [Mobiletest.me](http://Mobiletest.me)



72

## VIEWPORT

---

- Defines the area of the screen where the browser can render content
- `<meta name='viewport' content="width=device-width, initial-scale=1">`
- *Reflows content to match the device size*
- Put this in the `<head>` section



73

## MEDIA QUERIES

---

- @media queries override style rules based on browser width
- Syntax:  

```
@media (max-width: 900px) { /* styles go here */ }  
<link rel="stylesheet" href="handheld.css" media="only screen  
and (max-device-width:700px)"/>
```
- You can specify that styles are only for printed documents, screen readers, screens, etc  

```
@media only screen and (max-width: 900px) { /* styles go here */ }
```
- Determine breakpoints using “inspect”



74

## ADAPTIVE ELEMENTS & FLEXBOX

---

- With careful planning, it is possible to create a site that is responsive without doing any “extra work”.
- These techniques involve setting the size of your page elements to be proportional to the page size. This way when the page shrinks, so do the elements, creating a design that works well on the phone or on desktop.
- This MOSTLY works! There will be some elements that simply do not look good at a larger or smaller size. Therefore, you still need to examine every page on the site from top to bottom at significant browser widths to see what is not working.
- Where it doesn't work, your backup plan is to use media queries.



75

## ADAPTIVE ELEMENTS: TECHNIQUES

---

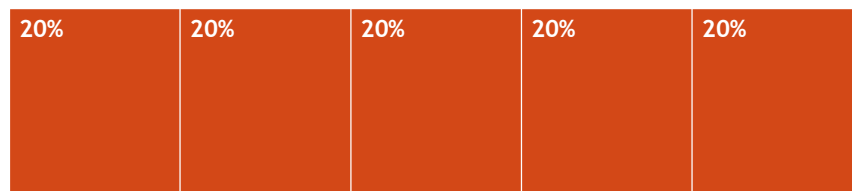
1. Use max-width
  - Set the width of an item to what you want it to be on mobile
    - Usually this is close to 100%
  - Set the max-width to what you want it to be on desktop
  - Don't forget ...
    - Center blocks that are less than 100%
    - Do not add left/right padding to blocks that are at 100%
2. Grid systems
  - Work in a grid of 4-8 columns across
  - Each element is set as a multiple of the column width
  - When the overall width is shrunk, the columns all shrink proportionally
3. Using Flexbox, items can expand or shrink to fit into a space. And it can work in rows or columns.



76

## GRID SYSTEMS

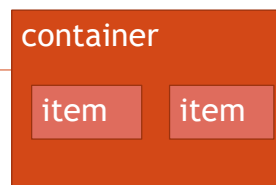
- Set up a grid layout and all elements must be a multiple of the grid width
- Adjust the grid sizes at various breakpoints and the size will automatically resize
- Bootstrap (<https://getbootstrap.com/>)



77

## FLEX BOX

- Flex display controls how items fit and are aligned within a container.
- **Container properties**
  - display:flex
  - flex-direction - rows or columns
  - flex-wrap - should it wrap
  - justify-content - distributes extra leftover space
  - align-items - can align vertically within a container
  - align-content - aligns rows against each other



- **Item properties**
  - order - order items should appear (if not the order they appear)
  - flex-basis - set size of an item
  - flex-grow, flex-shrink - allows some items to be larger or smaller than others
  - align-self - can override “align-items” for one item



78

## DUPLICATING CONTENT

- Sometimes the changes to a section from “desktop” to “mobile” are too drastic
- It doesn’t always make sense just to “squeeze” the page and see where things land
- Example- menus, headers, footers
- One option is to have two div’s with versions for different sizes.

```
<div class='no-mobile'>Full width stuff goes here</div>  
<div class='is-mobile'>Mobile friendly stuff goes here</div>  
  
.is-mobile {display:none;}  
@media (max-width: 600px) {  
  .is-mobile {display:block;}  
  .no-mobile {display:none;}  
}
```



79



WEEK 3

80