Web Front-End Development
Week 1: Remembering HTML, CSS, JavaScript

World Wide Web

What are the key components of the WWW?

Browsers

- Use HTTP to request files from a web server
- Use refresh to request file from the server even if it's cached in the browser
    - Sometimes need a hard refresh: command shift 5 (mac)
- Browsers have 3 different engines
    - Browser engine - handles communication between the rendering engine and the user interface
    - Rendering engine – renders documents (HTML, XML, etc) in the browser
        - WebKit – used in Safari
        - Blink –used in Chrome, Opera, and Microsoft Edge
            - based on WebKit but developed separately by Google
        - Gecko – open sourced to Mozilla and used in Firefox
    - JavaScript engine
        - Runs JavaScript in the browser
- Local storage can store pages and cookies
    - browser cache
- Document Object Model
    - The browsers object model of the web page

Servers

- Computer with web server software installed
- IP address registered with DNS so its site can be found

HTTP

- HyperText Transport Protocol
- Stateless protocol – once a request has been sent and received, it's forgotten. So tasks such as logins, browsing sessions or shopping carts aren't remembered from request to request
    - Cookies -- small text files stored in the browser
    - JavaScript programs run in the browser
    - Server-based session management

HTML

https://repl.it/@aileenjp/HTML

It's important to have well structured and semantic HTML

- Conveys meaning
- Helps comprehension
- Enhances ease of use
- Used by search engines
- Improves accessibility
- Improper structure can make content unreachable and cause errors

An HTML document has three basic parts:

- <html> HTML element
- <head> HEAD element (about the document)
- <body> BODY element (document text)

HTML elements have three parts:

- Opening tag, content, closing tag
- Tags act as containers for their content
- All tags should be closed . . .
- Tags need to be properly nested
    - <body><section><h1>text</h1></section></body>

Attribute values must be in quotation marks

-

Valid characters in HTML are A-Z, a-z, 0-9 and some symbols and punctuation

- Reserved symbols: & " < >

Valid HTML

- Make sure your HTML is valid
    - W3C Validator http://validator.w3.org
- Always check all your links after you publish
    - W3C Link Checker http://validator.w3.org/checklink
- You should run every page you publish through the HTML validator.
- Just because your page looks ok in one browser does not mean it's valid HTML.
- Invalid HTML can cause problems in other browsers, search engines, or screen readers.

Hyperlinks

Hyperlinks link web pages using a web address (URL).

https://repl.it/@aileenjp/HTML-withlinks

How would you link to the official star wars page? `http://www.starwars.com/`

- anchor tag: <a> link label
- href attribute specifies the link destination
- Official website is the link label
- http://www.starwars.com is the link destination
- Absolute URL
    - Linking to other web sites use the full web address
    - Specifies entire location

How would you link to the FAQ page?

- Relative URLs link to other pages on the same site
    - the location is relative to the URL of the current page
    - Don't need to specify the full URL

What could you do to get the second link on a new line?

Instead of the <h1>Star Wars tag how would you link to the starwarslogo400.png image in the images folder?

From index.html how would I make Leia a link to the leia.html page?

From leia.html how would I link to characters.html?

From leia.html how would I link to index.html?

Relative URLs make a web site portable – can move pages to a new server without changing all the links, since they are relative

- Move all pages to the new site
- Replicate same directory structure

More efficient – DNS doesn't need to look up the server location again, just the file being requested
Relative links are shorter in your HTML

CSS
CSS controls the style and presentation of elements
CSS associates style rules with HTML elements

https://repl.it/@aileenjp/CSS-Intro-external

- There are 3 ways to implement CSS on your site:
  - Create an external style sheet to hold all your CSS rules that all your pages can use
  - Define CSS rules in the <head> section of a page
  - Use the style attribute to put CSS rules directly into a HTML element
- Cascading style sheets mean that your rules can cascade.
  - Rules from your external style sheet will be applied first
  - Rules in the <head> section of a document will override those
  - Rules in the style attribute will override all others

How do we link to the external stylesheet? (repl)
How do we give our whole page a background color?

- selector {property: value; property: value; property: value; }
  - The selector is the element the style is for
  - The property is a style effect
  - The value is being assigned to that property
- Be careful of making your background-color black, black text won't show up!

What if you also wanted the text in the body of the page to be sized 120% of the default font size?
How do we make our Star Wars heading a different color (pick a color)
How would I make all the trilogy names a different color? (pick a color)
What if I had other h4 tags on the page?
What if I wanted to make my favorite character bold?

Identifying Elements
How do the id and class attributes differ?
The id attribute is used to uniquely identify an element

- Can be used with any element
- No two elements can have the same id in a page
- Use an id when you want to be able to refer to that element

The class attribute lets you group together several elements on a page

- Can be used with any element

Identifying elements on page is useful for

- styling
- interactivity
- links within a document(ids)

HTML and CSS work together and affect each other so they need to be planned together as well
What if I wanted all the trilogy movie names to be italic?

Add another rule for the same selector.

You can also group selectors by creating a list using commas
You can be more specific by chaining them using a .
Descendent selectors don't use a comma just a space
Selectors with a higher specificity ranking take precedence

- IDs (highest)
- Class
- Type (lowest)

For identical selectors order matters, the later rule applies
Guideline - start with more general selectors then get more specific as needed

- Type selectors
- Class selectors
- ID selectors
- Descendant selectors

CSS Box model (slide)
The CSS box model describes how the size of elements are calculated.
To the browser, every HTML element is a rectangular box.
An element's default box size is based on its content.
The developer tools shows you the box model for each element.

Block vs inline elements

- Block elements start on a new line so their box model is the full width of the window
- Inline elements stay on the same line so their box model is the size their content needs

There are five CSS properties used to determine the size and spacing of the elements.

- width and the height are used to set a specific width or height value to the content box
    - The display property is required for inline elements in order to apply width and height properties
- min-width and max-width enable a box's width to be flexible
- Border adds a border between the padding and the margin of an element
    - Every box has a border that separates one box from another
    - controls the border width, style, and color
    - Shorthand lists these width style color
        - border: 2px dashed yellow
- Padding adjusts the space inside of the element.
    - Padding is the space between the border of the box and the content in the box
    - Use padding when you don't want the content within an element to go to the edge of the box
- Margin controls the amount of space around the outside of the element
    - Block elements stack on top of each other by default
    - Use margins to add space between boxes

Padding and margin shorthand are in clockwise order starting at the top

- Top right bottom left
    - padding: 10px 5px 3px 2px
    - margin: 1px 2px 3px 4px

Padding and border add to the overall size of an element's box.

Units

- Pixels are fixed units and offer control, but they're not very flexible.
  - Doesn't take into consideration screen size or user font size
- Percentages are relative to the size of the component's containing element
  - 100% is the browser's default type size
  - Needed to design and develop for different screen sizes
  - Used in responsive web design
- Em
  - 1em is the browser's default type size
- 16px is the default font size in most cases
  - 1.5em would be equivalent to 24px
  - 0.875em would be equivalent to 14px

To visualize the box model let's put a border around all of our li elements.
Padding adds space between the element and its box

- shorthand: top right bottom left
  padding: 20px 20px 5px 5px;

Margin adds space between boxes

- If we add margin-bottom: 20px; we don't see a difference
- The top and bottom margins of blocks are combined (collapsed) into a single margin whose size is the largest of the individual margins

CSS Layout
CSS was originally for style but not really layout.
The float property was added for simple layouts involving an image floating inside a column of text, with the text wrapping around the left or right of it. It was not meant to handle the layout of an entire page.
Flexbox was added to CSS in 2016 and is a one-dimensional layout method for laying out items in rows or columns. Items flex to fill additional space and shrink to fit into smaller spaces.
You set an element up as a flexbox container by assigning display to be flex.
All children of a flexbox container are flex items that you can set flex properties for.

There's a link to a great flexbox tutorial on canvas.

Grid was added to CSS in 2017 and handles two-dimension layout using rows and columns.
It already has a good adoption rate https://caniuse.com/
You set display to grid and the first-level descendants get applied to the grid.
Developer tools will show you the grid lines.
And it works with flexbox.

Flexbox vs grid:
Flexbox is good for smaller components, single access layout.
Grid handles two-dimensional web pages better.
Use grid for full page layout, use flexbox for individual components

Responsive Web Design

If your web site doesn't look and work well on a mobile device, you're losing over half your viewers. Responsive Web Design lets you have one web site that adapts to different screen sizes and resolutions

- Can't make any assumptions about the device your site will be viewed on
    - Screen size and resolution
    - Browser window width
    - Default font size
    - Portrait or landscape
- Same content, different presentation (usually)
- Avoids major changes

Design considerations:

- Smaller screen and display size
- Lower resolution
- Interface limitations
    - Harder to click on links, form elements, hovers, etc
- Many different size devices with varying capabilities and features (slides)
- Interaction on mobile devices is different
- Bandwidth limitations
    - Keep total page size under 1 MB

Responsive web design is a set of techniques and ideas that enable the technology to respond to the user's environment

- Fluid grids (slide)
    - Design using proportions instead of fixed pixels
    - Express an element's size in relationship to its container
    - Flexbox and Grid layouts are responsive in nature
- CSS3 Media queries conditionally apply CSS styles
    - min-width sets a minimum browser/screen width
    - max-width sets a maximum width

```
@media screen and (max-width: 60em){ styles}
@media screen and (min-width: 60em) and (max-width: 85em){ styles}
```

- Conditionally load a different CSS stylesheet

```
<link rel="stylesheet" media="screen and (max-width: 60em)" href="small.css" />
<link rel="stylesheet" media="screen and (min-width: 60em)" href="large.css" />
```

    - Can also target orientation
- Flexible images and media
    - Don't set the height and width properties
    - Use CSS or dynamic sizing
    - img {max-width: 100%} lets the browser resize the image as needed
        - Full size image is still downloaded
        - Use the <picture> tag to specify different image sources based on the page layout

Viewport

- On many mobile devices web sites are automatically scaled to fit the smaller screen so a full-sized design shrinks and lets the user choose to zoom in and out. If you have a responsive web site you don't want this.
- Override the default to resize images

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

These concepts are implemented using open web standards like HTML, CSS and JavaScript.

Examples
Flexbox http://creative.colorado.edu/~apierce/samples/alice_flexbox.html

- <meta name="viewport" content="width=device-width, initial-scale=1.0">
- display: flex
- percentages instead of fixed size
    - font
    - width
- If you assume the default font size to be 16px then 16px = 1em. 16x30=480 so that would be targeting 480 pixels
- max-width for image

Flexbox w/media queries http://creative.colorado.edu/~apierce/samples/alice_flexbox_media.html

- picture tag https://developer.mozilla.org/en-US/docs/Web/HTML/Element/picture
    - https://caniuse.com/#search=picture
- media queries

JavaScript
JavaScript is the official programming/scripting language of HTML5

- Works with HTML and CSS to add interactivity to web pages
    - Access content of a page
    - Modify content of a page
    - Instructions the browser can follow
    - React to events in the browser
- Runs entirely client-side in the web browser

<script> </script> tag tells the browser all code within this tag is in a scripting language
All code within the script tags must be valid JavaScript.
To better organize your files it's best to separate your JavaScript into separate files

- Similar to using an external stylesheet for CSS
- JavaScript files should use a .js file extension
- The src attribute tells the browser where to find the JavaScript file, just like with images

<script src="js/script.js"></script>

https://repl.it/@aileenjp/JavaScript-basics

Variables
Naming variables

- JavaScript is case sensitive
    - FIRSTNAME and firstName are two different variables
- Don't use spaces or punctuation
    - Use capitalization or underscores for multiword names
    - Many of the punctuation characters have special meanings
- Use descriptive names
- Don't use really long names

Strings

- A string is a series of characters and are always enclosed in quotes
- Can use single or double quotes, but the pair must match.
- Some text editors will let you insert curly quotes around a string, “like this.” JavaScript will not recognize strings surrounded by curly quotes; it only recognizes straight quotes, "like this."

Example:

```
var test = 42; //number
console.log(test);
test = "forty two"; //string
console.log(test);
```

Data types https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures
If a value is not provided it will have value of “undefined”

- “undefined” means the variable exists but its value is not defined
- Not the same as “null” which is an empty value that must be assigned

```
console.log(me); //error, not defined

Add var me; above the console.log and now it’s defined.
```

JavaScript is a loosely typed language

- When defining variables you don’t specify a type
- typeOf will return the type of the variable or object
    - does not throw an error if the variable has not been declared.

JavaScript uses dynamic typing to determine the type of the variable

- the type can change over time

Add the following to see the type of the variable test after each assignment.

```
console.log(typeof(test));
```

Sometimes a variable’s type in JavaScript is not what you expect
typeOf will return the type of the variable or object
The isNaN function tells you if something is a number. Returns false if it’s a number

- isNaN(“four”) returns true
- isNaN(“4”) returns false

JavaScript has functions that handle type conversion

- parseInt(x) converts the parameter to an Integer
- parseFloat(x) converts the parameter to a real number/floating point
- convert to a string using .toString()

Example:

```
//strings
var test2 = 30;
test = 42;
var sum = test + test2;
```

```
console.log(sum);
console.log(typeof (sum));
test = "forty two";
sum = test + test2;
console.log(sum);
console.log(typeof (sum));
```

Scope

Where and how a variable is defined will affect where it can be used. This called the variable's scope. JavaScript has three ways of declaring variables:

- var - declares a variable with the scope of its current *execution context*
    - Variables created outside a function are global variables and can be used anywhere in your script.
        - Convenient for accessing data throughout a program.
        - Dangerous because it's hard to keep track of where they're being changed.
        - Use global variables sparingly and intentionally.
    - Variables that are defined inside a function using the var keyword will have a local scope.
        - Can only be used in that function
        - Function parameters are automatically local variables
        - Always declare variables by defining variables with the 'var' statement so that they have a local scope in a function.

Example:

```
//var scope
console.log("var");
var hot=90;
var temp=80;
    if (temp<hot){
      var hot=85;
      console.log(hot);
    }
console.log(hot);
```

- let - declares a block-scoped local variable that are limited in scope to the block, statement, or expression on which it is used.
    - This is unlike the `var` keyword, which defines a variable globally, or locally to an entire function regardless of block scope
    - Try to use let whenever possible

Example:

```
//let scope
console.log("let");
let hot2=90;
var temp2=80;
    if (temp2<hot2){
      let hot2=85;
```

```
      console.log(hot2);
    }
console.log(hot2);
```

- const - declares a block-scoped constant that can't be changed
    - Constants are block-scoped, like variables defined using let but cannot change through re-assignment, and it can't be redeclared.

Example:

```
//constants
const freezing = 32;
console.log(freezing);
freezing = 0;
```

A variable that is initialized inside a function without the var keyword will have a global scope and should be avoided.

Conditionals

If statements allow us to respond differently based on a condition.
They use Boolean expressions to test a condition

- Every condition will evaluate to either **true** or **false**
- All computer decisions reduce to Boolean decisions
- In JavaScript Boolean decisions are formed using relational operators

Relational operators

= = Abstract Equality

- double equals (= =) will perform a type conversion when comparing two things

= = = Strict Equality

- triple equals (= = =) will do the same comparison as double equals but without type conversion; if the types differ, `false` is returned

(no spaces, I only have them for clarity)
JavaScript equality
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Equality_comparisons_and_sameness

!= Not equal to
< Less than
<= Less than or equal to
> Greater than
>= Greater than or equal to

- Relational operators can be combined with arithmetic operators
    - 5+3 < 4
- Relational operators are always performed last

Logical Operators

! NOT

- The NOT operator, **!**, is used to negate or invert a Boolean value.

|| OR

- Only one needs to be **true** to result in **true**!

&& AND

- Both must be **true** to result in **true**!

if/else statements allow the program to take one action if the given condition is **true** but a different action if the condition is **false**.

Example:

```
//conditionals
let temp3=25;
if (temp3 < 70)
{
  if (temp3 < 30)
  {
      console.log("It's " + temp3 + " degrees outside - " + "wear a
jacket!");
  } //end 30 if
    else
    {
      console.log("It's " + temp3 + " degrees outside - " + "wear a
sweater!");
    } //end 30 if else
} //end 70 if
  else
  {
      if (temp3 > 85)
    {
        console.log("It's " + temp3 + " degrees outside - " + "wear a
tank top!");
      } //end 85 if
    else
    {
        console.log("It's " + temp3 + " degrees outside - " + "wear
short sleeves");
      } // end 85 if else
  } // end 70 if else
```

Run with values 25, 50, 86, 85

Using Conditionals:

- Plan out what you want your program to do.
- Put your test condition in ( )
- Type both { } for your conditional statements so you don't forget the closing bracket
- Indent code in brackets so it's easier to read
- Remember = = = for testing equality

- No semicolon after the test condition because it's not the end of the if statement

Iteration
JavaScript supports the most common types of loops so our programs can repeat a set of instructions until a given condition changes.

While Loop
While loops repeat a set of instructions

- While the test condition is true the statements in the curly brackets are run
- When it reaches the end curly bracket, JavaScript jumps back to the beginning of the while loop and evaluates the test condition again.
- When the test condition evaluates to false the body of the while loop is skipped and it continues with the rest of the program.
- We must provide something to test before the loop is executed the first time.
- The loop body has a chance of never executing if the test condition is **false** the first time.
- The while loop is excellent for error checking

JavaScript also supports the do-while loop which puts the test condition at the end so the body always executes at least once.

Example:

```
//iteration
let guess;
let result;
let target;
let turns = 0;
target = Math.floor(Math.random() * 100) + 1;
alert("I'm thinking of a number between 1 and 100. Guess my number,
and I'll tell you if it's too high, too low, or correct.  Try to
guess in the fewest turns.");
guess = prompt("What is your guess?");
turns = turns + 1;
while (guess != target)
{
  if (guess > target)
    {
      guess = prompt ("Too high!! Next guess?");
    turns = turns + 1;
    } // end if
    if (guess < target)
    {
      guess = prompt ("Too low!! Next guess?");
    turns = turns + 1;
    } // end if
    if (guess == target)
```

```
    {
      console.log("YOU WIN!!! The number was " + target + ". It took
you " + turns + " turns! ");
    } // end if
} // end while
if (turns<= 10)
{
  if (turns <= 5)
  {
    result = "Nicely done!";
  }
  else
    {
      result = "Not bad";
    }
}
else
  {
    result = "That took a lot of guesses";
  }
console.log(result);
```

JavaScript Math reference https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math

For Loop

The for loop is usually used for repeating a set of instructions a certain number of times.

- the number of repetitions can be calculated

There are three parts within the for loop:

1. Initialize the counter variable
   - Only happens the first time
2. Test condition
   - Boolean expression
3. Update the counter
   - increment/decrement statement
   - Ensures the test condition eventually is false

Initialize counter to 0 or 1

- Will change number of repetitions

If the test condition is false, the loop will never execute
Test condition must be true for the loop to execute
Don't change the counter within the loop
The counter will automatically be changed at the beginning of each loop iteration

Infinite loops

- Iteration will continue as long as the test condition is **true**.

- Something within the loop body ***MUST*** change causing the test condition to become **false**.
- If a **false** condition is never reached, you will have an infinite loop.
- Carefully plan out your program before writing code.
- Initialize your test condition variable(s) before a while loop.
- Make sure the test condition will eventually become false and the loop will end.
- Handle incorrect user input.
- Desk check your program.

Arrays

An array is a data structure that holds a group of values that are related to each other

- An array is an indexed collection of data
- The ***index*** is the reference number used to access the stored data
  - The index is a location, similar to a PO box number.
  - The index always starts at 0
- The ***element*** is the data stored at a given index location, similar to the mail in the PO box.
  - The first element is located at index [0]
  - The last element is located at index [length-1]
- Arrays are ordered
- Array can hold duplicate data
- The square brackets tell JavaScript to create an array
- Create an empty array when the data will be assigned later
- for loops are a great way to iterate through an array because you can calculate how many items are in the array
- The .length property stores the number of items in the array
- The forEach() method executes a function once for each array element https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach

Example:

```
//arrays and for loop
let scores = [];
let num;
num = prompt("How many grades do you have?");
for (i=0; i<num; i++)
{
  scores[i]=prompt("Enter grade #" + parseInt(i+1));
}

console.log("Your grades are:");
let total=0;
scores.forEach(function(score, index, myscores){
  console.log(score);
  total=total+score;
});
console.log("Your average is " + total/num);
```

What type do you think scores is?

```
console.log(typeof(scores));
```

How about the items in scores?

```
console.log(typeof(scores[0]));
```

We want the items in scores to be treated as integers.

```
total=total+parseInt(score);
```

Other array functions
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

- .push() adds an item to the end of the array
- .pop() removes an item from the end of the array
- .shift()removes an item from the front of the array
- .unshift() adds an item to the front of the array

JavaScript also has Sets which are unordered keyed collections of data

- Keyed collections are collections which use keys; these contain elements which are iterable in the order of insertion.
- Cannot have duplicate data

Functions

A function consists of multiple instructions that are grouped together so they can be run together because they perform a specific task

- Enables modularization
- Makes code reusable
- Reduce size and complexity of code
    - Efficiency by avoiding repetition
- Ensures consistency
    - Modifications are only made once
- Functions may:
    - Perform a task
    - Accept data
    - Return a value
    - Or any combination of the three

You can define functions in the <head> section or separate your JavaScript functions into their own file so they can be used in multiple pages.

- JavaScript files should use a .js file extension
- To run a function you call it within the **<body> </body>** portion of an HTML page.
- Functions are not run until they are called in the body portion of an HTML page.

Parameters

Some functions need data to achieve its task.
Ex: The alert() and prompt() functions need data sent to them.

- Parameters in the function header act as variables to hold data sent to the function
- If a function is expecting data you must send the data it needs when you call the function
    - Arguments represent the data sent to a function when it's called

- Arguments and parameters are matched 1-to-1 according to their order

Return values

A function can return data back to where the function is being called from.
Ex: The prompt() function returns what the user entered.

- Return statements return a single value
- Functions without a return statement have a specific purpose that does not require a value to be sent back to the calling program
    - Output
- When calling a function that returns a value you must do something with the returned value
    - assign it to a variable
    - have it be a part of output

Example:

```
//functions
function FahrToCelsius(tempFahrenheit)
{
  let tempCelsius;
  tempCelsius = (5/9) * (tempFahrenheit - 32);
  return tempCelsius;
}

let tempF;
let tempC;
tempF = prompt("Enter the temperature (in Fahrenheit):");
tempC = FahrToCelsius(tempF);
console.log("You entered " + tempF + " degrees Fahrenheit.");
console.log("That's equivalent to " + tempC + " degrees Celsius.");
```

Comments

As your scripts get larger it's easy to lose track of what all your variables are for, and what different parts of the code do.
When you, or another programmer, returns to your code weeks or months later, comments help you remember what each part of the program does.
Comments are notes only visible in the code, not seen in the browser by the user
Single-line comments

- // is a single line comment
- The browser ignores everything after the //

Multi-line comments span more than one line

- /* This is a multi-line
- comment */

Good places to use comments

- Variables - what they are for
- Different algorithms or parts of your program
- Anything you want help remembering
- Document the source of your code

Any JavaScript commands in the comments will be ignored

- Also useful to temporarily "comment out" parts of your code that you don't want to run

Creative server

Create a web page that will be the index page for your class portal

- Files must have a .htm or .html extension
- File names
    - no spaces, use camel case, underscore or dash
    - descriptive but not too long
    - File names are case sensitive
    - Home page default is index.htm
    - Use a consistent file naming scheme
- Connect to the creative server
    - The public_html folder holds your public files
    - Create a fwd folder on the server in public_html
    - http://creative.colorado.edu/~*identikey*/fwd
- Publish to the server
    - File Transfer Protocol (FTP) transfers files from your local computer to the server
        - Fetch, FUGU, Cyberduck
- Always keep a backup of your site on a flash drive
- Always work from your local file in the Workspace
- Never edit directly on the server; make a local copy first