Welcome & FAQs

Introduction

- JavaScript: store variables
  set decision points
  loop
  reuse code w/ functions
  ] real programming language (> HTML & CSS)
  get data from the browser
  manipulate the DOM that browsers use to create webpages
  ] focused on web design

- **variables**: store data & refer back to it later

- **decision points**: use control statements to decide which code to run under different circumstances

- **looping**: avoid writing the same/similar code over & over
  determine how many times you want to run some code in code or at runtime

- **functions**: reuse code multiple times, but only write it once
  use code from others

- **manipulating the DOM**: find elements from the DOM
  delete " "
  add elements to the DOM
  react to mouseclicks
  " " page reloads
  " " other actions

Lecture Materials - Week 1

The Document Object Model (DOM)

- Javascript works well w/ the DOM structure used to create HTML documents
- "I want to grab that part of the webpage & change it" → easy w/ Javascript code

- every webpage can be broken down into DOM tree structure
- each HTML tag = node in the tree

  - nodes have all types of different attributes: text
    background-color
    width

DOM Review

- web pages are built upon the DOM: structures documents like trees
  every node has one parent & possibly many children
  nodes have properties, methods, & events

- the DOM & JavaScript: page content is represented by the DOM
  scripting languages (JS) use the DOM to interact w/ the document

- API = Application Programming Interface
- accessing the DOM is done w/ an API

  - no matter which browser or scripting language → the API is always the same

- DOM objects/elements: document
  element
  nodeList
  attribute

- document = the root of the page
  if there's an attribute used to style your page → you can find it using JavaScript in the API

  - eg: document.URI
    document.height
    document.links
    document.bgColor

- element = a node in the tree
  can be found using the API

- nodeList: an array (group) of elements

  → (eg. document.getElementByTagName('p')): would return a set of nodes (all p nodes together)

- attribute: another way to manipulate/change the document

- specific APIs

  → (eg. document.getElementById(id)): get a certain element using the id

  → (eg. document.getElementsByClassName(class)): grab all elements w/in given class

  → (eg. element.innerHTML): can use this to change the content (text) of any element

  → (eg. element.style): change element style

  → (eg. element.setAttribute(attribute, value)): add additional attributes to any element the DOM can grab

  → (eg. element.removeAttribute(attribute)): remove attributes

JavaScript Output

- HTML5 & CSS3 aren't really interactive
- JavaScript can: read & write HTML elements — ★ focus of lecture
  react to events
  validate data
  detect the visitor's browser
  create cookies

- JavaScript doesn't have a built-in print function
- data is displayed via:

an alert box	—	window.alert()
a prompt	—	window.prompt()
HTML output	—	document.write()
HTML element	—	element.innerHTML()
the browser console	—	console.log()

() = this is a function

- alert(): a pop-up window that displays info

  - format: alert("message");

- prompt(): very similar to alert, but requires input → a pop-up window that wants input

  → (eg. prompt("Enter your name: ");)

- document.write(): writes directly to the page → becomes part of the DOM
  permanent unlike alert() & prompt()
  not recommended method

  - format: document.write("message");

  - can include HTML inside quotes

  → (eg. document.write("<h1> My message </h1>");): will output My message as a heading

- **innerHTML**: to change the contents of the DOM
  combine function w/ element you want to change

  - format: element.innerHTML = "message";

  - element will come from API

- **console.log()**: writes directly to the browser console
  hidden from plain sight

  - format: console.log("message");

- **console** = a place to see what's going on during the execution of your program
  also provides debugging info for JavaScript, HTML & CSS

- debugging / access console:

  - Safari: 1) Preferences → Advanced
    2) Check Show development menu under menu box

  - Google Chrome: Developer → Javascript Console
    Inspect → Console

  ★ Firefox: Tools → Console (best browser for debugging)

  - Edge: F12

Variables

- storing data: data is stored in variables
  variable must be declared in order to use it

- **variable declaration**

  - format: var name;
    keyword — variable name

- variable names: can include letters
  " " digits
  " " underscores
  " " dollar signs
  can't start w/ a digit
  case-sensitive
  - name ≠ Name ≠ NAME
  should be **nmemonic** = meaningful

- **variable assignment**: assign values using the assignment operator (=)

- format: var name = value;
  LHS RHS

- if value isn't assigned → value = null

- left hand side (LHS) = the variable being updated
- right hand side (RHS) = the new value that will be stored in the variable

Data Types

- in JavaScript, a variable can take on many different types
- data types: number
  string
  boolean
  object
  array

- number = any numerical value w/ or w/o decimals

  → (eg. var width = window.innerWidth;)

  → (eg. var pi = 3.14;)

- string = a collection of characters (letters, #s, punctuation) w/in quotes

  → (eg. var location = window.location;)

  → (eg. var name = "Erin";)

- boolean = value that's either true or false

  → (eg. var windowStatus = window.closed;)

  → (eg. var status = false;)

- object = more complex variable/type

  → (eg. var topic = document.getElementById("id");): object = a node in the DOM
  nodes ≠ a single value → they have attributes

- array = a collection of multiple values
  elements are accessible by their index

  → (eg. var links = document.getElementsByTagName("a");)

  → (eg. document.write(links[0]);): accesses & writes 1st element from/in links array

Operators & Expressions

- operators: assignment operator
  arithmetic operators
  more operators
  string operators
  boolean operators
  logical operators

- **assignment operator**: =

- **arithmetic operators**:

+	addition
-	subtraction
*	multiplication
/	division
%	modulus

- **more operators**:

++	increment
--	decrement
+=	add then assign

- **string operators**:

+	concatenation
+=	concatenate then assign

- **boolean operators**:

==	is value stored in LHS equal to RHS value?
!=	is value stored in LHS <u>not</u> equal to RHS value?
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
===	equality w/ type → are these values the same <u>and</u> the same type?
!==	values are <u>not</u> the same or values are <u>not</u> the same type

- **logical operators**:

&&	and
||	or
!	not