Lecture Materials – Week 2

Functions

- **function syntax**
  - format: `function name (parameters) {` (`function` ← keyword)
    `...`
    `}`

- function declaration & function call
  - → (eg. `function welcomeMsg(msg) { alert(msg); }`) : function declaration
  - → (eg. `var x = "Hello"; welcomeMsg(x);`) : function call

- **parameters**: sometimes functions need info in order to perform its "function"
  can have any name, just stay consistent

- **return values**: some functions return values
  can be used in assignment statements / conditional expressions

- use built in functions whenever possible
- don't be too specific when writing functions: don't hardcode too many values
  should be able to reuse multiple times in different scenarios

Code Placement

- JavaScript code can be placed in: head
  body
  external file

- place JavaScript function declarations in the head → separate from content
- place JavaScript function declarations in an external file → reuse code in multiple files
- call functions in the body

- debug code using the console

Folder Structure

- organize code into separate parts: html
  css
  images
  javascript

- **conventional structure**: project > css (html files)
  images
  js

- linking from an HTML file

  → (eg. <link rel="stylesheet" href="css/style.css">): link css style sheet

  → (eg. <script src="js/functions.js"></script>): link js code

  → (eg. <img src="images/picture.jpg">: link image

- linking from a css file

  → (eg. background: url("../images/picture.jpg"): link image

Events

- adding interactivity: call functions based on events via JavaScript API
  any element can react to an event

- events: onclick
  onmouseover
  onresize
  onload

- onclick: user clicks on HTML element
- onmouseover: user moves the mouse over an HTML element
- onresize: browser window is resized
- onload: browser finishes loading page

- how to: add the event to the tag
  include reaction/what you want to happen

  - format: <element event="function name()">...</element>

  → (eg. clicking this will invoke a JavaScript function)

- event result: use double quotes for outer
  use single quotes for string parameters

  → (eg.)

- events change the program flow: no more linear order (step-by-step)
  DOM is always listening for events → program "runs continuously"

- more events: mouse events
  keyboard events
  frame events

- mouse events: onclick
  ondblclick

onmousedown
onmouseenter
onmouseleave
onmousemove
onmouseout

- keyboard events: onkeydown
  onkeypress
  onkeyup

- frame events: onload
  onresize
  onscroll
  onerror

- use events sparingly

"this"

- this = a keyword that allows an element to reference itself
  allows you to access an element's info
  used in outside functions

  - every object in the DOM has an automatically generated this

  - makes it easy for functions to know what data to use