COMS/SE 319: Construction of User Interface
Spring 2021
LAB Activity 1 – JavaScript

Agenda:

- **Task 1: Play with JavaScript (Getting Started)**
- **Task 2: Play with JavaScript Variables**
- **Task 3: Play with JavaScript Functions**
- **Task 4: Functions as First-Class Objects**
- **Task 5: Closures**
- **Task 6: THIS**
- **Task 7: Event Handling**
- **Task 8: Interact with Prototypal Inheritance**

Task 1: PLAY WITH JAVASCRIPT (GETTING STARTED)

Assumptions:

- You already know HTML basics.
- If not – then browse through quickly https://www.w3schools.com/html/default.asp
  HTML is mostly about formatting --- so should be easy to grasp.

Learning Objectives:

- **Students will:**
  - learn how to embed js in HTML files
  - learn how to debug js code on browser
  - learn about variable types

Resource:

All the links shown in the snapshot below have a wealth of information. Please read first.
About JS in general: https://www.w3schools.com/js/default.asp
About JS variables: https://www.w3schools.com/js/js_variables.asp

Step 1:

READ https://www.w3schools.com/js/js_whereto.asp

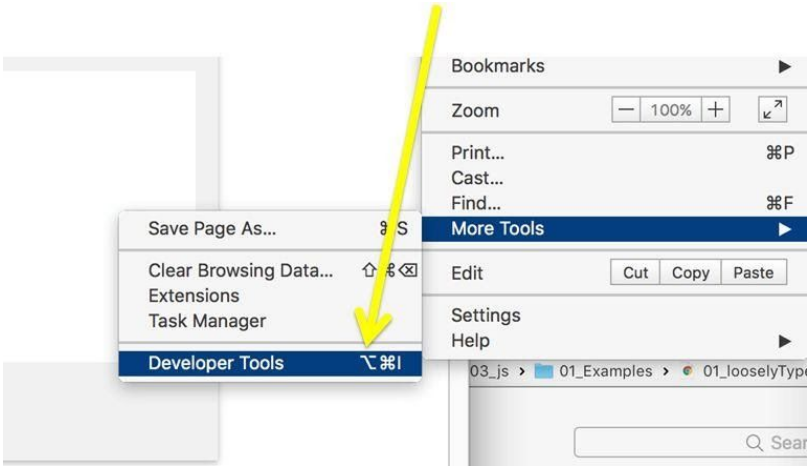
Step 2:

- READ 01_looselyTyped.html (provided inside zip folder)
- Double click on it (should run the file in the browser).
- We will assume you are using Google Chrome.

NOTE: Since the browser can interpret js (i.e. js runs on the client side), we do not need to have a server to play with js.

Step 3:

Start the debugger. Click on settings, more tools, and the developer tools (or use the shortcut)



Step 4:

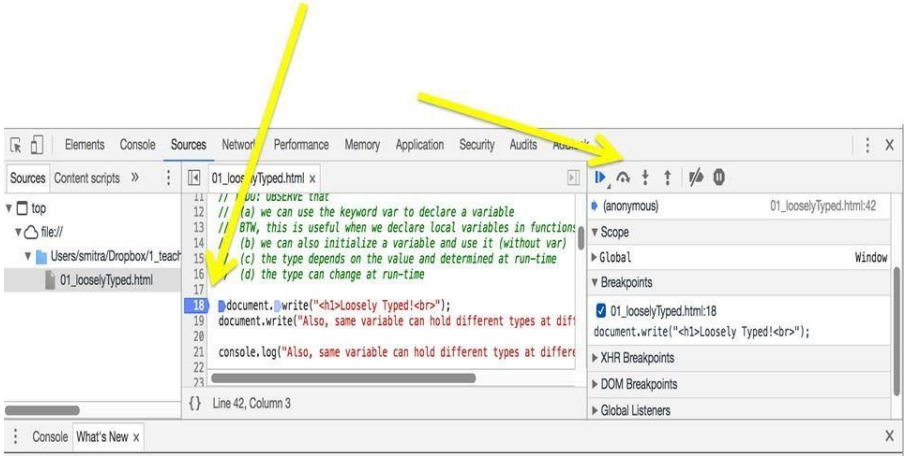
Play with the different tabs of the debugger. In particular (Elements, Console, Sources, and Network).

Step 5:

- Run the 01_looselyTyped.html (provided) by double-clicking on it.
- Think about the difference between document.write and console.log.
- Think about the different types of data that are demonstrated in this example.
- Think about how to find the type of data item.

Step 6:

Run the code in debug mode by

1. inserting a breakpoint (click on line number)
2. refreshing the browser (to reload the program)
3. stepping over a statement.
4. 

Task 2: PLAY WITH JAVASCRIPT VARIABLES

Learning Objectives:

- **Students will:**
  - learn about var, let, and const

Resource:
All the links shown in the snapshot below have a wealth of information. Please read first.
About JS variables: https://www.w3schools.com/js/js_variables.asp
About var, let, and const: http://wesbos.com/javascript-scoping/
More about let and const: http://wesbos.com/let-vs-const/

Step 1:

- Read about var, let, and const: http://wesbos.com/javascript-scoping/
- Read more about let and const: http://wesbos.com/let-vs-const/

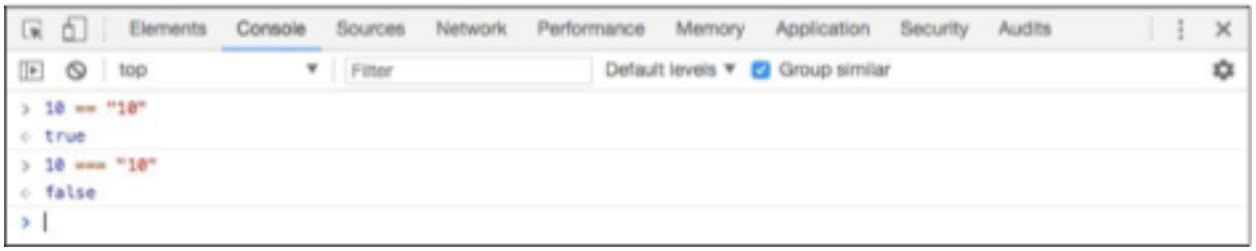
Step 2:

- READ 02_varsDeclarations.html (provided)
- Double click on it (should run the file in the browser).
- What do the following scopes mean?
    - Global
    - Function
    - Block
- How do var, let, and const differ from each other?
- Which one of those keywords below declares a variable in function scope?

Step 3:

- READ 03_equality.html(provided)
- Double click on it (should run the file in the browser).
- What is the difference between == and ===?

Note that you can type in JavaScript in the console window and see results!



Task 3: Play with Javascript FUNCTIONS

Learning Objectives

- Students will:
    1. learn about js functions:
        - function declarations vs function expressions
        - declaration hoisting
        - function names as pointers to function objects

Resource:
All the links shown in the snapshot below have a wealth of information. Please read first.
About JS functions: https://www.w3schools.com/js/js_functions.asp

Also, function scope: https://www.w3schools.com/js/js_scope.asp

Step 1:

1. Read 04_funcs_declaration.html
2. Run it by double clicking on the file
3. Think about the two ways that functions can be declared.
4. Think about the differences between the two in how they behave.
5. Think about what happens on lines 36 and 37.

Step 2:

1. READ 05_funcs_hoisting.html (provided)
2. Double click on it (should run the file in the browser).
3. Think about what function declaration hoisting means.

Step 3:

1. READ 06_funcs_pointers.html (provided)
2. Double click on it (should run the file in the browser).
3. Explain what happens in line 34?
4. Explain what happens on line 48 and why that is different from line 34.

Task 4: Functions as First-Class Objects

Learning Objectives:

- learn what is meant by first class objects
- learn that functions are first-class objects in javascript.

Resource:
https://en.wikipedia.org/wiki/First-class_function
Useful Blog:
https://hackernoon.com/effective-functional-javascript-first-class-and-higher-order-functions-713fde8df50a

Step 1:
READ https://en.wikipedia.org/wiki/First-class_function
When are functions said to be first class objects?
Step 2:
READ 07_funcs_firstClass.html (provided)
On which line is
a) a function being assigned to a variable?
b) a function being passed as a parameter?
c) a function being declared inside a function?
d) a function being returned from a function?
Step 3:
Double click on 07_funcs_firstClass.html. Make sure you understand the results!

Task 5: CLOSURES

Learning Objectives:

- **Students will:**
  Play with functions in order to see examples of closures.

Step 1:

- READ https://www.w3schools.com/js/js_function_closures.asp and http://javascriptissexy.com/understand-javascript-closures-with-ease/
- What is a javascript closure?

Step 2:

- READ 08_funcs_closures.html (provided)
- READ 09_1_funcs_useOfClosures.html
- READ 09_2_funcs_useOfClosures.html

Step 3:

- READ 10_0_funcs_closures3.html
- READ 10_1_funcs_closures3.html
- READ 10_2_funcs_closures3.html

If you uncomment these lines below in 10_2_funcs_closures3.html file, what is the output?

```
//document.write(x[0]() + "<br>");
//document.write(x[1]() + "<br>");
```

Step 4:

- READ 11_funcs_closuresReuseCode.html
- Note how we are able to use "higher order functions" (functions that take other functions as parameters) to abstract out common functionality. Makes the codes much easier to write (and read).

Task 6: THIS

Learning Objective:

- Students will:
  - learn about simple js objects.
  - understand "this" usage in js

Step 1:

- READ 12_1_funcs_objectsThis.html
- Run the code (by double-clicking the file) and then explain what happens.

Step 2:

- READ 13_objectsAsArrays.html
- Run the code (by double-clicking the file) and then explain what happens.

Step 3:

- Read the SampleProgram (match.js and lab.html).
- Run the code (by double-clicking the HTML) and then explain what happens.

Step 4:

NOTE that there are some other js files in the folder. You can play with them. As they include some concepts/tools we haven't gone over --- you can also ignore them for now.

Task 7: EVENT HANDLING

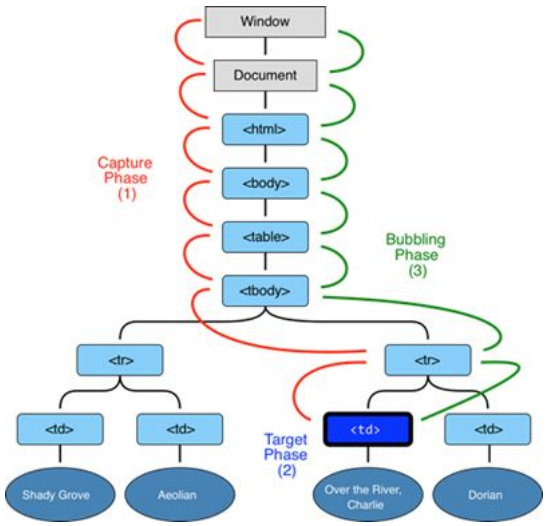
Learning Objective:

- Students will:

- Learn more about event handling (https://javascript.info/bubbling-and-capturing)
- how events bubble up
- how capture of events work
- how to stop event propagation

Step 1:

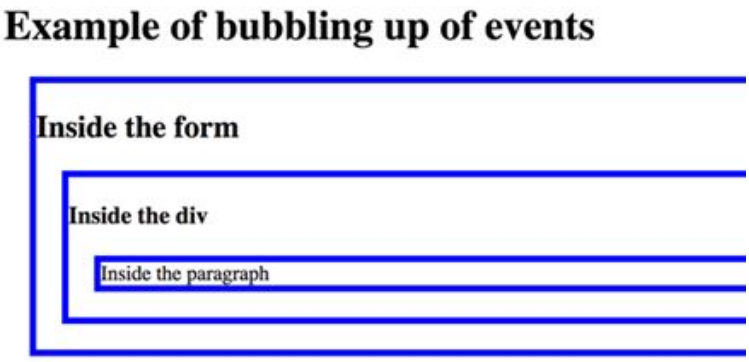
SCAN (i.e. read lightly and quickly) https://javascript.info/bubbling-and-capturing



Step 2 (bubbling):

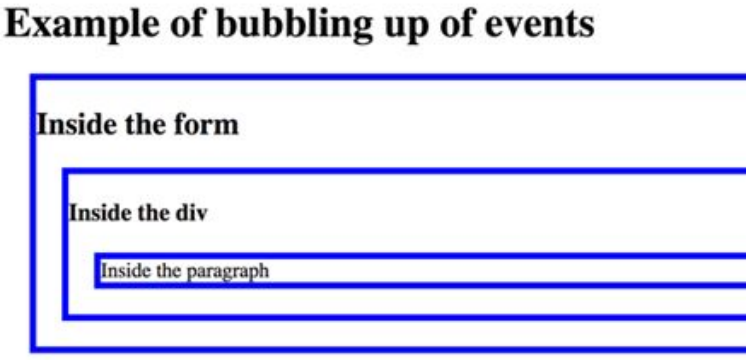
- Events bubble up from target DOM element and can be handled at each element.
- Read ***events01.html***

Click inside the div and other parts of the screen



Step 3 (capture):

- Events go downwards in capture phase from top to target DOM element and can be handled at each element.
- Read ***events02.html***
- **Click inside the div and other parts of the screen**



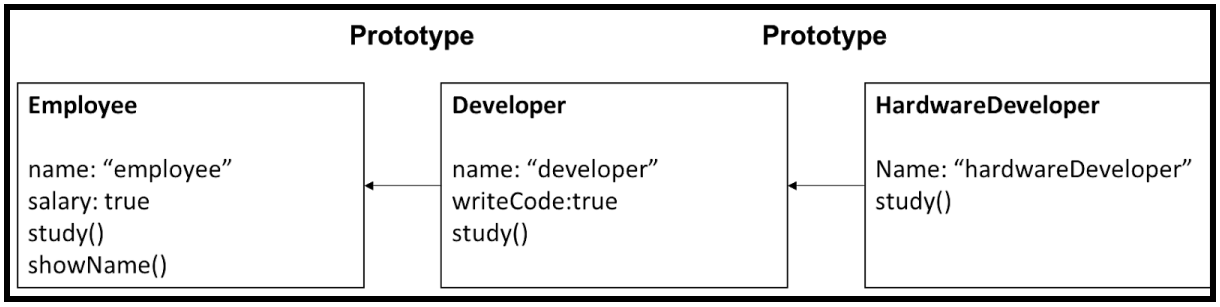
Task 8: Interact with Prototypal Inheritance

Learning Objective:

- What is prototypal inheritance and how it works.
- How to define, get and set a **[[Prototype]]**
- How to manage deep chain of prototype.
- How to use **this** in functions of Prototype.
- How to use **for...in** loop

Step 1: Warm up

- Reading this resource: https://javascript.info/prototype-inheritance
- *Prototypal inheritance* is a language feature that helps with inheriting properties/functions from another object when we create a new object.
- In this task, you will work with 3 instances: **employee, developer** and **hardwareDeveloper.** The diagram of 3 objects is shown below:



Step 2: prototype inheritance of objects.

A. Open the file prototype_inheritance_01.html on Text Editor (You can use Notepad on Windows or TexEdit in Mac).

B. Check each line to understand the code.

```
16  <script>
17  let employee = {
18    salary: true
19  };
20  let developer = {
21    writeCode:true
22  };
```

C. Double click on the file prototype_inheritance_01.html and check the output on Chrome.

D. Why there is **undefined** value on the output?

E. Now come back to the editor and uncomment line 23.

```
20  let developer = {
21    writeCode:true
22  };
23  // developer.__proto__ = employee;
```

F. The output of "Salary: " is no longer "undefined". Why?

Step 3: inherited function

A. Open the file prototype_inheritance_02.html and check the code.
B. Double click this file and check the output on Chrome.
C. Uncomment line 26 and double click the file again. Check the output.
D. Uncomment line 28-30 and double click the file again. Check the output.
    - Why the result after uncomment is different?
    - What is the relation between employee and developer?

```
24 let developer = {
25   writeCode:true,
26   // __proto__:employee
27 };
28 // developer.study = function() {
29 //   document.write("Study basic courses and programming language courses"+"<br/>");
30 // };
```

Step 4: deep prototype chain

A. Open the file prototype_inheritance_03.html and check the code.
B. Double click on the file and see the output on Chrome.
    - Why the third line show “Study basic courses”?
C. Uncomment line 28-30 and see the output on Chrome (refresh the web page on step 4B).
D. Continue uncomment line 35-37 and see the output on Chrome.
    - Which **study** function is actually run in this case?

```
24 let developer = {
25   writeCode:true,
26   __proto__:employee
27 };
28 // developer.study = function() {
29 //   document.write("Study basic courses and programming language courses"+"<br/>");
30 // };
```

```
32 let hardwareDeveloper = {
33   __proto__:developer
34 };
35 // hardwareDeveloper.study = function() {
36 //   document.write("Study basic courses,programming language courses and embeded system courses"+"<br/>");
37 // };
```

Step 5: this value

A. Open the file prototype_inheritance_04.html by text editor and check the code.
B. Double click on the file and see the output.
C. Now uncomment line 22-24, refresh the Chrome and see the output.
    - Which object does “this.name” refer to?

```
16  let employee = {
17    name:"employee",
18    salary: true,
19    study() {
20      document.write("Study basic courses"+"<br/>");
21    },
22    // showName(){
23    //   document.write("Name of position: "+this.name+"<br/>");
24    // }
```

Step 6: for...in

A. Open the file prototype_inheritance_05.html by text editor and check the code.

B. Uncomment line 36-44, open the file on Chrome to see the output.

- What is the purpose of **hasOwnProperty ?**

```
36  // for(let prop in developer) {
37  //   let isOwn = developer.hasOwnProperty(prop);
38  //
39  //   if (isOwn) {
40  //     document.write("Own properties/functions: "+prop+"<br/>");
41  //   } else {
42  //     document.write("Inherited properties/functions: "+prop+"<br/>");
43  //   }
44  // }
```

C. Now change the order of 2 fields in line 28-29:

Before	After
27 let developer = { 28 name:"developer", 29 writeCode:true, 30 __proto__:employee 31 };	27 let developer = { 28 writeCode:true, 29 name:"developer", 30 __proto__:employee 31 };

D. Refresh the page and see the output.

- Does the order of properties/functions change accordingly?

We have the quiz questions on canvas. Answer them on Canvas.

Please answer the Lab Activity Quiz in Canvas. No need to submit other questions mentioned in this pdf.
