CSC435: Web Programming

Lecture 9: JavaScript: Conditional and Loops, Event-driven programming

Bei Xiao

American University

Feb 15, Friday, 2019

Const versus Var versus Let

keyword	const	let	var
global scope	NO	NO	YES
function scope	YES	YES	YES
block scope	YES	YES	NO
can be reassigned	NO	YES	YES

global window object

```
var varVariable = “this is a var variable”;
let letVariable = “this is a let variable”;


console.log(window.varVariable); //this is a var variable

console.log(window.letVariable); //undefined
```

Redeclaration within the same scope not possible for let

// let variables cannot be re-declared while var variable can be re-declared in the same scope.

```
'use strict';
var temp = "this is a temp variable";
var temp = "this is a second temp variable"; //replaced easily

'use strict';

let temp = "this is a temp variable";

let temp = "this is a second temp variable" //SyntaxError: temp is already declared
```

Define Functions

```
function name() {
  statement ;
  statement ;
  ...
  statement ;
}
```

```
function myFunction() {
  alert("Hello!");
  alert("How are you?");
}
```

The above could be the contents of example.js linked to our HTML page.
Statements placed into functions can be evaluated in responses to user events.
To display results, you can use document.write()

Activity Outline

- Continue with basic JS data structure
- Semicolons
- Event Driven Programming
- Exercises

Exercise : converting length in cm to inches

- Write a .js that converts inches to cm.
- First start with a .html and put this in the head:

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>Conerting length in cm to inches</title>
  <script src="Exercise3.js" type="text/javascript"></script>
</head>
<body>

</body>
</html>
```

Exercise : converting length in cm to inches

- Write a .js that converts inches to cm.
- Write the result to browser as the following:
- “length in inches”, 10 inches
- “length in cm, 25.4 cm

- Hint: use document.write

Exercise : converting length in cm to inches

```
<html>
<head>
	<title>Calculate Inches to cm</title>
	<script type="text/javascript"
src="Exercise1.js"></script>
</head>
<body>

</body>
</html>
```

HTML

```
function cmtoinches() {
	let inches = prompt("tell me a number in
inces");
   let cm = inches * 2.54;
   document.write("length in Inches:"+ inches
+ " inches <br/>");
   document.write("length in centimeters:"+
cm + " cm <br/>");
}

// calling your function
cmtoinches()
```

.js

Math object

```
let rand1to10 = Math.floor(Math.random() * 10 + 1);
let three = Math.floor(Math.PI);
let power = Math.pow(Math.random(),2); // power of 2
Math.round(10*x)/10; // round to the tenth.
```

- methods: abs, ceil, cos, floor, log, max, min, pow, random, round, sin, sqrt, tan
- properties: E, PI
- http://www.w3schools.com/js/js_math.asp

Boolean type

```
let iLikeJS = true;
let ieIsGood = "IE6" > 0;    // false
if ("web dev is great") {  /* true */ }
if (0) {  /* false */ }
```

- any value can be used as a `Boolean`
  - "falsey" values: `0`, `0.0`, `NaN`, `""`, `null`, and `undefined`
  - "truthy" values: anything else
- converting a value into a `Boolean` explicitly:
  - `var boolValue = Boolean(`*otherValue*`);`

Logical operators

- Relational: > < >= <=
- Logical: && || !
- Equality: == != **=== !==**

  - Most logical operators automatically convert types.
  - These are all true:
    - 5 < "7"
    - 42 == 42.0
    - "5.0" == 5
  - The === and !== are strict equality tests; checks both **type and value**:
    - "5.0" === 5 is false

Typeof

```
typeof "John"                 // Returns string
typeof 3.14                   // Returns number
typeof NaN                    // Returns number
typeof false                  // Returns boolean
typeof [1,2,3,4]              // Returns object
typeof {name:'John', age:34}  // Returns object
typeof new Date()             // Returns object
typeof function () {}         // Returns function
typeof myCar                  // Returns undefined (if myCar is not declared)
typeof null                   // Returns object
```

More no type conversions:

http://www.w3schools.com/js/js_type_conversion.asp

If/else statement (same as java)

```
if (condition) {
  statements;
} else if (condition) {
  statements;
} else {
  statements;
}
```

- identical structure to Java's `if/else` statement
- JavaScript allows almost anything as a *condition*

If/else statement (same as java)

```
let name = "kittens";
If (name == "puppies") {
   name += "!";
} else if (name == "kittens"){
   name +="!!";
} else {
   name = "!" + name;
}
name == "kittens!!"
```

- identical structure to Java's `if/else` statement
- JavaScript allows almost anything as a *condition*

for loop

```
for (initialization; condition; update) {
  statements;
}
```

```
let sum = 0;
for (let i = 0; i < 100; i++) {
  sum = sum + i; // same as sum += i;
}
```

What is the output of this?

```
let s1 = "It's a-me, Mario!";
let s2 = "";
for (let i = 0; i < s.length; i++) {
  s2 += s1[i] + s1[i];
}
```

While loops (same as Java)

```
while(condition){
 statements;
}
```

```
do{
 statements;
} while (conditions);
```

Exercise: predicting outcome

```
function forloop() {
   for (var i = 1; i <= 10; i++) {
      let mystery = "";

     // inner loop 1

      for (var j = 1; j <= 10 - i; j++) {
         mystery += " ";
      }
     // inner loop 2
      document.write("<br>");

      for (var j = 1; j <= 2 * i - 1; j++) {
         mystery += "*";
      }
      document.write(mystery);
   }
}
```

Exercise: predicting outcome

```
function myFunction() {
    var text = ""
    var i = 0;
    do {
        text += "<br>The number is " + i;
        i++;
    }
    while (i < 5);
    document.getElementById("demo").innerHTML = text;
}
</script>
```

Arrays

```
let name = [];                                  // empty array
let name = [value, value, ..., value];          // pre-filled
name[index] = value;                            // store element
```

```
let ducks = ["Huey", "Dewey", "Louie"];

let stooges = [];           // stooges.length is 0
stooges[0] = "Larry";       // stooges.length is 1
stooges[1] = "Moe";         // stooges.length is 2
stooges[4] = "Curly";       // stooges.length is 5
stooges[4] = "Shemp";       // stooges.length is 5
```

- Two ways to initialize an array
- `length` property (grows as needed when elements are added)

Array Methods

```
let a = ["Stef", "Jason"];    // Stef, Jason
a.push("Brian");              // Stef, Jason, Brian
a.unshift("Kelly");      // Kelly, Stef, Jason, Brian
a.pop();                      // Kelly, Stef, Jason
a.shift();                    // Stef, Jason
a.sort();                     // Jason, Stef
```

- array serves as many data structures: list, queue, stack, ...
- methods: concat, join, pop, push, reverse, shift, slice, sort, splice, toString, unshift

  - push and pop add / remove from back
  - unshift and shift add / remove from front
  - shift and pop return the element that is removed
  - Lear more here:
  - http://www.w3schools.com/js/js_array_methods.asp

Examples: splice

```
let months = ['Jan', 'March', 'April', 'June'];
months.splice(1, 0, 'Feb');

// inserts at 1st index position
console.log(months);
// expected output: Array ['Jan', 'Feb', 'March', 'April', 'June']

months.splice(4, 1, 'May');
// replaces 1 element at 4th index
console.log(months);
// expected output: Array ['Jan', 'Feb', 'March', 'April', 'May']
```

Examples: shift()

```
var array1 = [1, 2, 3];

var firstElement = array1.shift();

console.log(array1);
// expected output: Array [2, 3]

console.log(firstElement);
// expected output: 1
```

Examples: unshift()

```
var array1 = [1, 2, 3];


console.log(array1.unshift(4, 5));
// expected output: 5


console.log(array1);
// expected output: Array [4, 5, 1, 2, 3]
```

Exercise

- What are the outputs?
  - let fruits = [‘banana’, ‘kiwi’,’melon’]
  - fruits[fruits.length] = ‘apple’
  - fruits.shift()
  - fruits.unshift(‘pear’)
  - fruits.splice(2, 0, ‘lemon’, ‘mongo’)
  - fruits.reverse()

Splitting strings

```
var s = "the quick brown fox";
var a = s.split(" "); // ["the", "quick", "brown",
"fox"]
a.reverse();                    // ["fox", "brown",
"quick", "the"]
s = a.join("!");                // "fox!brown!quick!the"
```

- split breaks apart a string into an array using a delimiter
- join merges an array into a single string, placing a delimiter between them
- s.splice () can add new items into the array.
- s.splice(position, howmany, item)
- s.splice(0,1); can remove the first element

Rules of semi-colon

- Normally, statements are terminated by semi-colons.
- The exception is statement ending with blocks.
  - Loops: for, while (but not do-while)
  - Branching: if, switch,try
  - Function declarations (not function expressions).

http://speakingjs.com/es5/ch07.html#_rules_for_using_semicolons

String methods

String concatenation:

```
var arr = [];
arr.push('Say hello');
arr.push (7);

arr.join('' )

'Say hello 7 times fast'
```

Extract Substrings:

> 'abc'. charAt(1)
'b'
> 'abc'.slice(1,2)
'b'
> 'ab'

http://speakingjs.com/es5/ch12.html

Prediction of output of the following code

```
string1 = "foobar";

document.write(string1.length + "<BR>");

for (i = 0; i < string1.length; i++) {
   document.write(string1.charAt(i));
}
document.write("<BR>");

string2 = "";
for (i = 0; i < string1.length; i++) {
   string2 = string2 + string1.charAt(i);
}
document.write(string2 + "<BR>");
```

Prediction of output of the following code

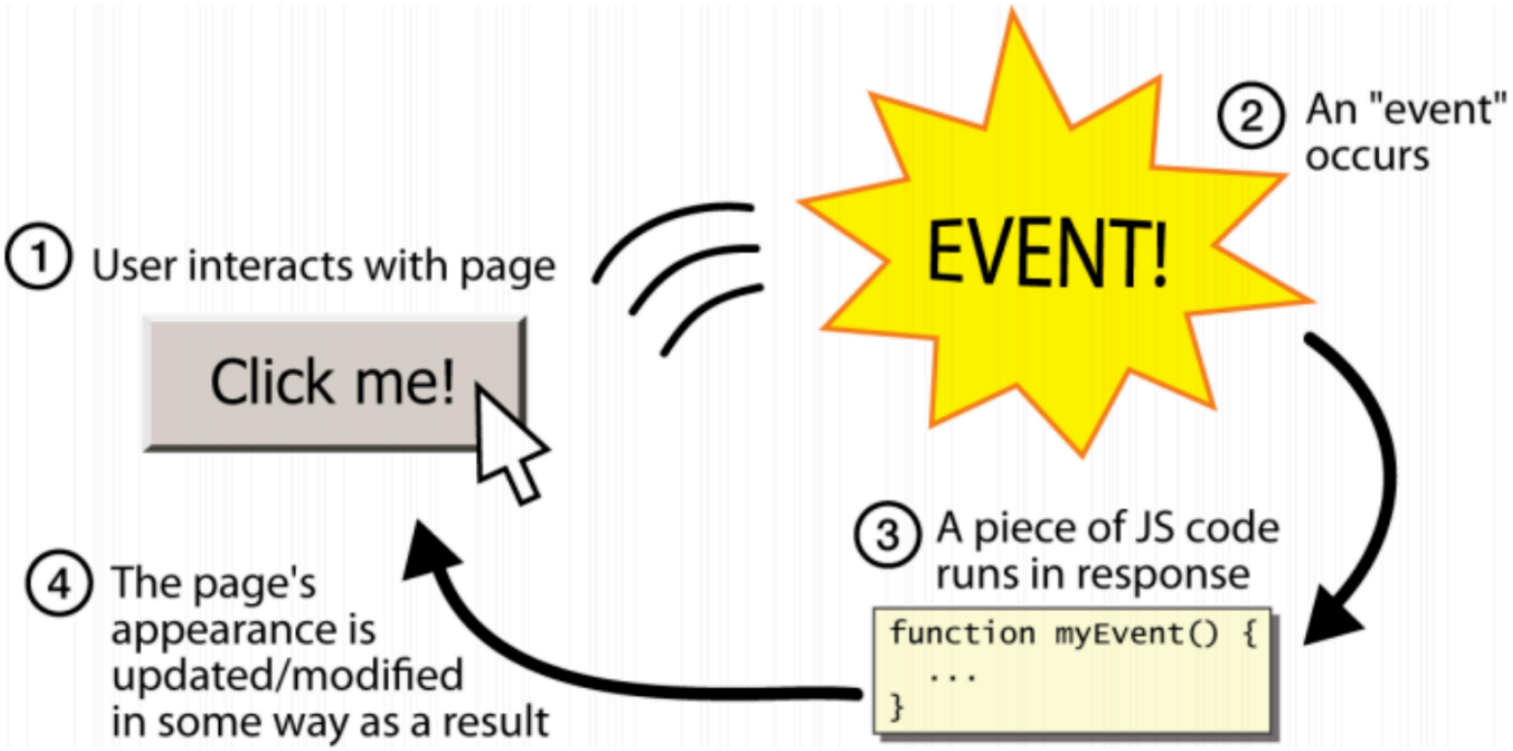
```
string1 = "foobar";

document.write(string1.length + "<BR>");

for (i = 0; i < string1.length; i++) {
   document.write(string1.charAt(i));
}
document.write("<BR>");

string2 = "";
for (i = 0; i < string1.length; i++) {
   string2 = string2 + string1.charAt(i);
}
document.write(string2 + "<BR>");
```

Event-Driven Programming



Unlike Java programs, JS programs have no main; they respond to user activation called events.

Event-driven programming: writing programs driven by user events

What is user events?

A. A user drags a page's scrollbar

B. A user accidentally spilling coffee on their computer.

C. A user visits a web page and the page load successfully.

D. A user typing their name into a text input box on a webpage.

E. A user clicks on hyperlink to visit a different page.

F. A user clicking a web pages submit button

Event Handlers

```
<element attributes onlick="function();">…..
```

html template

```
<div onlick="myFunction();">Click me! </div>
```

html example

```
Click me!
```

output

JavaScript functions can be set as event handlers

When you interact with the element, the function will execute

onclick is just one of many event HTML attribute we will see.

Event Handlers

```
<element attributes onlick="function();">…..
                                        html template
```

```
<img id="myImage" onclick="changeImage()"
src="pic_bulboff.gif" width="100" height="180">
```

Click the light bulb to turn on/off the light.

Buttons: <button>

```
<button onclick-"muFuction();">Click me! </button>
```

HTML

Click me!

Output

Button's text appears inside the tag; can also contain image

```html
<html>
▼<head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js"></script>
  </head>
▼<body>
    <button>Click Me!</button>
  </body>
</html>
```

```js
function onClick() {
  console.log('clicked');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```

document.getElemengById Example

```
<img id="pokeball" src="images/pokeball.jpg" alt="a pokeball" />
<button onclick="changeImage();">Click me!</button>
```

```
function changeImage() {
  let pokeballImg = document.getElementById("pokeball");
  pokeballImg.src = "images/mystery.gif";
}
```

Click me!

output

document.getElemengById Example

How do you make the code toggle between pikachu and the pokeball?

You can start with pokeball.html and pokeball.js

Hink:

Use image.src.match

Demo here:

https://www.w3schools.com/js/tryit.asp?filename=tryjs_lightbulb

Demo: click a button to toggle an image

Click me to change color

Exercise 0: move the ball

Click button below to move the image to right

Click Me

Hint: You can access to ball's position as this:

```
imgObj = document.getElementById('myImage');
 imgObj.style.position= 'relative';
 imgObj.style.left = '0px';

// to move the ball

parseInt(imgObj.style.left)
```

Exercise 1: smallest

Write a function named **findMin** that accepts an array of numbers as a parameter and returns the smallest number in the array. For example, if an array variable named nums stored the following values:

let nums = [-1, 3.2, 12, 15, -4, 1, -12.5, 1, 8];

Then the call of findMin(nums) should return -12.5 since that is the smallest numerical value in the array.

You may assume that the array passed to your function is non-empty and contains only number types.

To display results, you can use document.write()

Or you can create a button to trigger your function

```
<button onclick=”findmin();">Click me!</button>
```

Exercise 2: reverse a number

- Write a JavaScript function that reverse a number. E.g: 25368 -> 86352

- Step 1: Create a simple .html file.
- Step 2: Write a function and save it as ReverseString.js
- Step 3: link the .js into the <head></head> in your .html

- First change the number to string.
- Then n.split(“”).reverse().join(“”);

- Again, you can create a button to trigger the function or use alter.

Exercise 3: guess a number

- Write a JavaScript program where the program takes a random integer between 1 and 10, the user then prompted to input a guess number. If the user input matches the guess number, the program will display “good work”, otherwise, it will display “not matched”.

Hint: use alert() function to pop out text message.

Use Prompt() for user input. E.g.

```
var gnum = prompt('Guess the number between 1 and 10
```

Exercise 4: split string

- Write a JavaScript function to split a string and convert it to an array of words.

- alert(string_to_array(“ Monday is blue”) ;

Output: “Monday”, “is”, “Blue”

Exercise 5: show today's date

- Write a JavaScript function to display today's day in the following format:
- Today is Thursday. It is 5:30pm.

```
let  d = new Date();         //get the date
let  day = d.getDay();  // get the day of the date
let  hour = d.getHours(); // get hours
let  minuets = d.getMinuets(); //get minuets
```

You can use document.write or alert to display the message.

Consider makes the days into an array of strings. Again array starts with 0.

Exercise 5 (take home): show dates until Christmas

- Write a JavaScript to display how many dates until Christmas of 2018.
- You might find the following function useful:

```
var  d = new Date();        //get the date
var  n = d.getFulLyear();   // get the year of the date
var  d = d.getMonth();      // get the month of the
date
d.setFullYear(2020, 10,3); // Tue Nov 03 2020 11:17:37
GMT -500 (EST)
```

- Expected month from 0 to 11. 11 will be December, 12 will be start of the next year

- JS complete date references:
- http://www.w3schools.com/jsref/jsref_obj_date.asp

Take-home reading and exercise

Introduction to JavaScript (must read):

https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript

DOC model:

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction