

CSC435: Web Programming

Lecture 11: JavaScript: Unobtrusive, Objects

Bei Xiao

American University

Feb 26, Tuesday, 2018

Future lecture plan

Lectures	Content	homework
Feb 23 (today)	Unobtrusive JS OOP in JavaScript Functions	Creative Project 2 out
Feb 26 (Tuesday)	OOP, Form validation.	Homework 3 due
March 1 (Friday)	DOM, Events and Timers	Homework 4 (UI control is out
March 5 (Tuesday)	JavaScript Animations Mid-term review	Creative Project 3 out
March 12-17	Spring break	Homework 4 due
March 19 (Tuesday)	Mid-term exam	In-class exam (HTML, CSS, JS)
March 22 (Friday)	Ajax, Fetch, Json	Homework 5 out
March 26	JavaScript and Ajax	

Activity Outline

- JavaScript Functions Demo
 - Unobtrusive JavaScript
 - Function exercise
 - JavaScript Objects
 - OOP Exercises.
-
- Next class: Quiz 2 (in-class requires turn in on blackboard).

Take-home reading and exercise

Introduction to JavaScript (must read):

JavaScript functions:

http://eloquentjavascript.net/03_functions.html

Objects and arrays:

http://eloquentjavascript.net/04_data.html

DOC model:

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

Is web dev a basket of tricks?

- One can never know enough.
- Anyone can know just enough to do something (ugly).
- Frameworks comes and go all the times.
- There 10 different ways to do the same thing. However, there is perhaps a best way.
- You can always perfect your web page.
- Vanilla JavaScript (not framework) is still absolutely worth to know. Reason? It is so much easier to learn the frameworks after mastering basic JS.

Declarative versus Procedural

- **Declarative:** a style of building the structure and elements of computer programs- that expresses the logic of a computation without describing its control flow.
 - Declarative programming requires a more descriptive style. The programmer must know *what* relationships hold between various entities.
 - Examples: HTML, SQL, XML
- **Procedural:** A procedural language is a type of computer programming language that specifies a series of well-structured steps and procedures within its programming context to compose a program. It contains a systematic order of statements, functions and commands to complete a computational task or program.
 - Examples: Python, C/C++, Java, JavaScript, PHP

JavaScript function Example: display the smallest number from an array

```
<!DOCTYPE html>
<html>
<head>
  <script src="smallest.js"
  type="text/javascript"></script>
</head>
  <body>
  </body>

</html>
```

HTML

```
function findsmallest(numbers) {

    let numlen = numbers.length;
    let smallest = numbers[0];

    for(i = 1; i < numlen; i++){
        if (smallest >
numbers[i]){

            smallest = numbers[i];
        }
    }
    alert(smallest);
}
let numbers = [-2,100,4,-1,-10];
findsmallest(numbers)

smallest.js
```

Document Object Model (DOM)

A set of JavaScript objects that represent each element on the page

Each tag in a page corresponds to a JavaScript DOM object

JS code can talk to these objects to examine elements' state

- e.g. see whether a box is checked

We can change state

- e.g. insert some new text into a div

We can change styles

- e.g. make a paragraph red

How to get DOM elements in JS:

1. Ask for them by

id: `document.getElementById(...)`

1. Query for them with CSS style selectors:

1. `document.querySelector(...)`

2. `document.querySelectorAll(...)`

2. Make new ones! `document.createElement(...)`

Getting a DOM elements in JS:

```
<p id="october"></p>
```

html

```
let pTag = document.getElementById("october");
```

JS

What's inside a DOM object?

```
<p id="october"></p>
```

html

```
let pTag = document.getElementById("october");
```

JS

What's inside a DOM object?

For starters, the HTML attributes. This HTML:

```

```

Has a DOM object (let's call it `puppyImg`) with these two properties:

- `puppyImg.src` -- set by the browser to `images/puppy.png`
- `puppyImg.alt` -- set by the browser to `"A fantastic puppy photo"`

Exercise 1: unobtrusive JS event

- Write a HTML with a button “OK”
- Write a unobtrusive JavaScript that when the user click the OK button, then the page pop out “Booyah”.
- Check with your neighbor to see if your code is truly Unobtrusive.
- `<button id=“ok”>OK </button>`

Hint: wrote two functions

```
function pageLoad(){} // load the page and event handler
```

```
Function okayClick(){} // alert
```

Obtrusive Event Handler (bad)

```
<button onclick="okayClick();">OK</button>
```

HTML

```
// called when OK button is clicked
```

```
function okayClick() {  
    alert("booyah");  
}
```

JavaScript



OK

This is a bad style (HTML is cluttered with JS code)

Goal: remove all JavaScript code from HTML body

Solution: attach an event handler in JavaScript Code

```
let objectName.onevent = function()
```

JS

```
<button id="ok">OK</button>
```

HTML

```
let okButton = document.getElementById("ok");  
okButton.onclick = okayClick;
```

JS

It is legal to attach event handlers to elements' DOM objects in your JavaScript code.

Notice that you do not put parentheses after the function's name

This is better style than attaching them in the HTML.

When does my code run?

```
<html>
  <head>
    <script src="myfile.js" type="text/javascript"></script>
  </head>
  <body> ... </body>
</html>
```

HTML

```
let x = 3;
function f(n) { return n + 1; }
function g(n) { return n - 1; }
x = f(x);
```

JavaScript/myfile.js

- Your file's JS code **runs the moment** the browser loads the script tag
 - Any variables are declared immediately
 - Any functions are declared but not called, unless your global code explicitly calls them
- At this point in time, the browser has not yet read your page's body
 - None of the DOM objects for tags on the page have been created yet

A failed attempt at being unobtrusive

```
<html>
<head>
  <script src="myfile.js" type="text/javascript"></script>
</head>
<body>
  <div><button <em>id="ok"</em>>OK</button></div>
  (... more html ...)      HTML
```

```
let btn = document.getElementById("ok");
btn.onclick = okayClick;           // this is bad: btn is null at this point
                                   JavaScript/myfile.js
```

- Problem: global JS code runs the moment the script is loaded.
- Script in head is processed before page's body has loaded.
- No elements are available yet or can be accessed yet via the DOM
- We need a way to attach the handler after the page is loaded.

The window.onload Event

```
function functionName() {  
    // put code to initialize the page here  
}  
  
// instruct window to run the function when the page has loaded:  
window.onload = functionName; // notice no () after function name
```

- There is a global event called window.onload event that happens once everything in the page is loaded.
- If you attach a function as a handler for window.onload, it will run at that moment.

Exercise 1: unobtrusive JS event

- Write a HTML with a button “OK”
- Write a unobtrusive JavaScript that when the user click the OK button, then the page pop out “Booyah”.
- Check with your neighbor to see if your code is truly Unobtrusive.
- `<button id=“ok”>OK </button>`

Hint: wrote two functions

```
function pageLoad(){} // load the page and event handler
```

```
Function okayClick(){} // alert
```

Exercise 1: unobtrusive JS event

```
<body>  
  <button id="ok"> Ok!</button>  
</body>
```

(1)

```
function pageLoad() {  
  let ok = document.getElementById("ok"); // (3)  
  ok.onclick = okayClick;  
}  
  
function okayClick() {  
  
  alert("booyah");          // (4)  
}  
  
window.onload = pageLoad;    // (2)
```

Common Unobtrusive JS Errors

Event names are all in lower cases, not capitalized like most variable.

```
window.onLoad=pageLoad;  
window.onload = pageLoad;
```

You shouldn't write () when attaching the handler
(if you do, it calls the function immediately, rather than setting it up to be called later)

```
ok.onclick = okayClick();  
ok.onclick = okayClick;
```

Can't directly call the function alert() must encloses in your own functions.

```
ok.onclick = alert("booyah");  
ok.onclick = okayClick;  
  
function okayClick() { alert("booyah"); }
```

Anonymous Functions

```
Function (<var> parameters <var>) {  
    .....statements.....;  
}
```

- JavaScript allows you to declare anonymous functions
- Quickly creates a function without giving it a name
- Can be stored as a variable, attached as an event handler, etc.

Anonymous Function Example

```
window.onload = function() {  
  let ok = document.getElementById("ok");  
  ok.onclick = okayClick;  
};  
  
function okayClick() {  
  alert("booyah");  
}
```

- Or, the more concise, but perhaps harder to read:

```
window.onload = function() {  
  document.getElementById("ok").onclick = function() {  
    alert("booyah");  
  };  
};
```

Unobtrusive styling

```
function okayClick() {  
  this.style.color = "red";    // <-- bad style  
  this.className = "highlighted"; // <-- better style  
}
```

JS

```
.highlighted{color:red;}
```

CSS

- Well-written JavaScript code should contain as little CSS as possible
- Use JS to set CSS classes/IDs on elements
- Define the styles of those classes/IDs in your CSS file
- We will discuss this in another class

Global variables can be bad

```
let count = 0;  
function incr(n) {  
  count += n;  
}  
function reset() {  
  count = 0;  
}  
  
incr(4);  
incr(2);  
console.log(count);  
JS
```

- globals can be bad; other code and other JS files can see and modify them.
- How many global symbols are introduced by the above code?

Enclosing code in a function

```
function everything() {  
  let count = 0;  
  function incr(n) {  
    count += n;  
  }  
  function reset() {  
    count = 0;  
  }  
  
  incr(4);  
  incr(2);  
  console.log(count);  
}  
  
everything(); // call the function to run the code
```

- Solution: wrap everything in a function. Variables and functions declared in a function are local to it.
- How many global are there: 1 global (everything).
- Can we get it down to 0?

Module pattern example

```
(function() {  
  let count = 0;  
  function incr(n) {  
    count += n;  
  }  
  function reset() {  
    count = 0;  
  }  
  
  incr(4);  
  incr(2);  
  console.log(count);  
})();
```

- How many global symbols so far?
- 0 global symbols

JavaScript “strict” mode

```
“use strict”;
```

```
..... Your code .....
```

- Writing "use strict"; at the very top of your JS file turns on strict syntax checking:
- Shows an error if you try to assign to an undeclared variable
- Stops you from overwriting key JS system libraries
- Forbids some unsafe or error-prone language features
- You should *always* turn on strict mode for your code in this class!

JS skeleton

```
<!-- in the <head> block -->  
<script src="path/to/javascript/file.js"  
type="text/javascript"></script>
```

HTML

```
(function() {  
  
    // set-up code that doesn't involve the DOM  
    // (e.g. setting up initial values, arrays, etc.)  
  
    window.onload = function() {  
        // phew! your code goes here  
    };  
  
    //function definitions go here  
  
})();
```

JS

JS example

```
(function() {  
    window.onload = function() {  
        alert("This alert was fired from inside a  
window.onload handler. All of the javascript  
files have been run, so let's try to see what x is  
set to.");  
  
        alert("This is inside the module, so let's  
check the value of x. x = " + x);  
  
        alert("It should be undefined, because the  
module has given us a 'private' namespace,  
where we don't inherit the symbols from the  
global namespace.");  
  
        var x = 5;  
  
    };  
})();
```

Modifying DOM Elements (Example

```
<a id="fb-link" href=http://www.facebook.com> Facebook </a>  
HTML
```

Before the JavaScript runs, we'd see:

Facebook

And after we run this JavaScript:

```
let link = document.getElementById("fb-link");  
link.innerHTML = "MySpace is back in a really big way.";
```

We 'd see

My space is back in big way.

Exercise 1:

Display a random number

- Create a .html with a button:
- `<button id="ok">Lucky Number </button>`
- Write a .js function that display a random number between 0 and 10 when the user click the button.

Lucky Number! your lucky number is: 9

- Hint: please write unobtrusive JS
- Use `element.innerHTML` to write the number next to the button.

Exercise 2:

simple computations (download the starter html from blackboard)

- Write a dropdown menu use

- `<input class="numberInput" type="text">`

- `<select id = 'input1t">
<option value ="square">square
<option value ="cube">cube
<option value ="factorial">factorial
</select>`

```
input.onchange =  
function() {  
    var num =  
    input.value;
```

Use `<input id ="input1">` to ask users to input a number.

- To allow users to compute various functions of the number entered in the input area such as square, cube, or factorial.
- Display the results in the browser when they select the method.
- Hint: `let input = document.querySelector('.numberInput')`
- `let output = document.getElementById('output')`

Exercise 2

- How do we convert this code to unobtrusive?

```
<!DOCTYPE html>
<html>
<body>

<h2 class="example">A heading with class="example"</h2>
<p class="example">A paragraph with class="example".</p>

<p>Click the button to add a background color to the first element in the
document with class="example".</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
    document.querySelector(".example").style.backgroundColor = "red";
}
</script>

</body>
</html>
```

JavaScript Properties and methods

Properties:
example

`array.length`

`myString.length`


`Math.max`

Methods: example

`array.push()`

`string.toUpperCase()`

JavaScript objects

Object	Properties	Methods
	<code>car.name = Fiat</code> <code>car.model = 500</code> <code>car.weight = 850kg</code> <code>car.color = white</code>	<code>car.start()</code> <code>car.drive()</code> <code>car.brake()</code> <code>car.stop()</code>

```
var car {  
  name: "Fiat",  
  model: "500",  
  color: "white",  
  Weight: "850kg"};
```

```
// retrieval  
car.name // "Fiat"  
  
car[name] // "Fiat"
```

Object: construction and retrieval

- An object is a container of properties, where a property has a name and a value.*

Construction

```
Var flight {  
  airline: "Oceanic",  
  Number: 815,  
  Departure: {  
    IATA: "SYD",  
    time: "2004-09-22 14:55",  
    city: "Sidney"  
  };  
  Arrival: {  
    IATA: "LAX",  
    time: "2004-09-23 10:42",  
    city: "Los Angeles"  
  }  
};
```

Retrieval

```
flight.departure.IATAL // "SYD"  
  
flight[airline] // "Oceanic"  
  
// use || to fill in default  
value  
  
Var status = flight.status ||  
"unknown";  
  
flight.equipement //undefined  
  
flight.equipment.model //throw  
"TypeError"
```

Object: update

- A value in an object can be updated by assignment. If the property name already exist in the object, the property value is replaced:*

Construction

```
Var flight {  
  airline: "Oceanic",  
  Number: 815,  
  Departure: {  
    IATA: "SYD",  
    time: "2004-09-22 14:55",  
    city: "Sidney"  
  };  
  Arrival: {  
    IATA: "LAX",  
    time: "2004-09-23 10:42",  
    city: "Los Angeles"  
  }  
};
```

update

```
flight['airline'] = 'wow'  
// if the object doesn't have  
the property name, the object  
is augmented:  
  
flight.equipment = {  
  model: 'Boeing 777'  
};  
  
flight.status = 'overdue'
```

Object: reference

```
var Stooge = {  
  "first-name": "Jeremy",  
  "second-name": "Howard"  
}  
var x = stooge;  
x.nickname = 'Curly';  
var nick = stooge.nickname;  
  
//nick is 'Curly' because x and stooge  
are references to the same object
```

Object: function construct with “this”

```
function person(firstname,lastname,age,eyecolor)
{
    this.firstname=firstname;
    this.lastname=lastname;
    this.age=age;
    this.eyecolor=eyecolor;
}
```

```
// new instance
```

```
myFather=new person("John","Doe",50,"blue");
```

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this>

Object: adding method

```
myFather.name = function () {  
    return this.firstName + " " + this.lastName;  
};
```

Object: quiz

Which of the following is a valid way to create a direct instance of an object?

- a. `myObject.create ();`
- b. `myObject = new Object;`
- c. `myObject = new Object();`

Object: quiz

- What is the the output of the following code after “alert”?

```
function person (firstname, lastname, age, eyecolor)
{
  this.firstname=firstname;
  this.lastname=lastname;
  this.age=age;
  this.eyecolor=eyecolor;
}
```

```
myFather = new person("John", "Doe", 50, "blue");
var x =myFather;
x.job = "Teacher";
var profession = myFather.job;
alert(profession);
document.writeln("father's firstname is ",
  myFather.firstname, "<br>");
```

Using “reference”

- Add code to the code in the last slide and print:
- `my father 's nickname is Johnny
using document.writeln`

Demo: show info

In a JavaScript, create an object.

Create a property called “info” and assign a string.

Write a function (object method) myFunct() that alert the “info” value of the .info property to the browser.

you can say: “I am a new shinny object”

Create a instance of the method of the object by calling myFunct()

Create a button uses onClick to evoke the method. How do you display the “info” to the browser?

Enumeration of object

```
for (var key in object ) {  
    print(object[key]);  
}
```

```
var obj = {first: "prop1", second:  
    "propr2", 3: "proper3"}
```

```
for (var key in obj) {  
    s += key + ":" + obj[key] + " ";  
}  
document.write(s);
```

Object: exercise 1

- Write a JavaScript program to list the property of the following sample object:
- ```
var student = {
 Name: "Jenny Klein"
 Class: " Senior"
 AUID: " 31635"
 Hobby: "writing code"
};
```

Sample output: name, Class, AU ID, Hobby

Hint: write a function to output the list of property. E.G. you can use `string.push()` to append to an empty array and then print out the array.

# Object: exercise 2

- Write a JavaScript program to display the reading status (i.e. display book name, author name, and reading status) of the following books.

```
var library = [
 {
 title: 'Bill Gates',
 author: 'The Road Ahead',
 readingStatus: true
 },
 {
 title: 'Steve Jobs',
 author: 'Walter Isaacson',
 readingStatus: true
 },
 {
 title: 'Mockingjay: The Final Book of The Hunger Games',
 author: 'Suzanne Collins',
 readingStatus: false
 }
];
```



# Exercise: input number

- Create a simple UI input field
- Ask the user to input numbers between 1-10
- If the input number is not within the range,
- Tell them the input is not valid.
- If the input number is within the range,
- Tell them the input is valid.

# Exercise: input number

- Create HTML element with ID:

```
<p>Please input a number between 1 and 10:</p>
<input id="numb">
<button type="button"
onclick="myFunction(">Submit</button>
<p id="demo"></p>
```

- In JavaScript, create a function to validate the number:
  - a ) Get the element of the input field by id.

```
var numb = document.getElementById("num").value
```

- b) see if the number is a number `isNaN()` and whether it is between 1 and 10.
  - c) report the results to the browser using  
`document.getElementById("demo").innerHTML = text;`