JS

Pull from upstream!

Commit any changes first!

Slides created in conjunction with: Ilchul Yoon, Nelson Padua Perez

Agenda

- Office Hours Updates
- Symbols
- Maps/Sets
- Objects
- Writing to the DOM

Update #2

Wednesday April 22nd, we will **not** meet in person!

Logistics

Exercise 2 will be posted Thursday

Exercise 1 grades will be posted by the end of the week (hopefully)!

Example: instanceOf.html

Typeof vs instanceof

- **typeof**
  - Returns “object” for all reference types

- **instanceof** operator
  - Returns true if a value is an instance of the specified type and false otherwise
  - instanceof can identify inherited types

Every Object is an instance of an Object!

Two More Notable Array Methods

- reduce
  - Executes a reducer function (callback) on each element of the array, resulting in a single output value
  - First argument of the callback function is “accumulator”
  - Passes the result of callback (the accumulator) from one array element to the other
- filter
  - Creates a new array with all elements that pass the test implemented by the provided function

Debugging

- Select Inspect after loading the script, and Sources. This will open the debugger.
- Click on a source line to set a breakpoint.

- Alternatively, you can add in your code the statement

debugger;

which will invoke the debugger when you run the script

Symbols

- New primitive type in ES6
- Tokens that serve as unique ids
  - Create via the factory function Symbol()
  - new” keyword does not work

Let x = new Symbol(“Description”);

Symbols

- Can be used as special property keys
- Every symbol is unique
  - Symbol() === Symbol() is false
- Symbols can be used as property keys
  - Computed property key
  - Allows you to specify key of a property via an expression, by putting it in square brackets
- String value parameter is optional

Using Symbols

- Following operations ignore symbols
  - for-in loop
  - Object.keys()
  - Object.getOwnPropertyNames()
- Conversion of Symbol to Boolean returns true
- Can be used to represent concepts
  - const RED_COLOR = Symbol(‘red color’);

Sets

- Collection of keys
- Keys can be primitive or references
- The Set constructor has zero or more arguments. With no arguments an empty Set will be created
- If an argument is specified, it needs to be iterable (e.g., array)
- When iterating over sets, elements will be processed in the order they were inserted

Maps

- Collection of keys
- Keys can be primitive or references
- The Set constructor has zero or more arguments. With no arguments an empty Set will be created
- If an argument is specified, it needs to be iterable (e.g., array)
- When iterating over sets, elements will be processed in the order they were inserted

Creating Maps and Sets

Map:

- let m = new Map();
- m.set(key, value);

Set:

- let s = new Set();
- s.add(value);

Immediately Invoked Function Expression (IIFE)

- A JS function that runs as soon as it is defined
- A design pattern known as a Self-Executing Anonymous Function
- Two parts
  - anonymous function with lexical scope enclosed within the Grouping Operator ().
  - Prevents accessing variables within the IIFE idiom as well as polluting the global scope.
- Emulating block-scoped variables
- Not needed, if “let” is used instead of “var”

Objects

- Just a collection of properties
    - You can define your own; browser predefines a set of objects
    - A property can be seen as a variable associated with a value
    - Approaches to access and add properties
        - Using dot-notation
        - Using square brackets

Objects

- Property – association between a name and a value
  - When the value is a function the property is referred to as a method
  - Name can be any valid JavaScript string or anything that can be converted to a String (that includes empty string)
    - Any invalid property name can only be accessed using square bracket notation

How do we create Objects?

- Using Object Constructor
- Using Object Initializer/literal notation
- Using Object.create

Objects as Maps

- We can also view an object as an entity that associates values with strings.
    - Use the [] operator

Ex: myObj.value == myObject[“value”]

Object Type

- All objects in JavaScript are descended from Object
- All objects have a property called __proto__
- The __proto__ property points to an object (called prototype) from which properties are inherited
- Objects inherit methods and properties from Object.prototype
- Prototype chain
    - Set of objects defined by the __proto__ property
    - The end of the chain is a prototype with the null value (Object.prototype.__proto__)

Object Prototypes

- Methods:
  - Object.prototype.hasOwnProperty(prop)
    - prop is a direct property (not inherited through the prototype chain)
  - Object.prototype.isPrototypeof(obj)
  - Object.prototype.toString()
    - Returns a string representation of the object
  - Object.prototype.valueOf()
    - Returns the primitive value of the specified object
  - In ES6, Symbol.toPrimitive is a symbol that specifies a function valued property that is called to convert an object to a corresponding primitive value.

Object Constructors

- Rather than handwriting all values in an object, Javascript allows for Object Constructors

Ex:

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
}
```

Basics of Writing To Document from JavaScript

For now, we will only learn one way to dynamically write html from our JavaScript:

document.writeln(“html tags and text here”);

For example: document.writeln(“<p>Paragraph Text</p>”);

Basics of Writing To Document from JavaScript

You may also embed variables into your html now!

For example:

```
let x = “Station Wagons”;

document.writeln(“<p>My favorite cars are ” + x + “</p>”);
```

Most of the examples posted use this, so test it out!

WTWAW

After today make sure you know how to:

- Create a symbol (and know it’s use)
- Use and manipulate maps and sets
- Create Objects all 3 ways
- Create an object constructor
- Use document.writeln();