JS Objects And Obj APIS

Pull from upstream!

Commit any changes first!

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

Agenda

- Objects
- Prototypes
- Classes
- Writing to the DOM
- Event Bubbling

Object Prototypes

- A better way of using Object.create()
- Will force inheritance of properties from a parent
- Allows us to have functions across all functions

ObjectName.prototype.methodName(){}

Note: ObjectName is analogous to class name

All objects will now have that method available.

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;
}
```

Classes in JavaScript

- Use keyword class
- Constructor is no longer using function, use constructor instead
- Methods can be defined with no other keywords necessary
- Not hoisted!

Let’s create an Object!

What is `this`?

- Outside of any object, it refers to the global object window **or** is undefined (if you “use script”)
- Arrow functions do not have their own “this” - rather the this value of the enclosing lexical scope is used;
- When in an object, it refers to the current Object
  - Works the same as in Java
  - This.data to access a data field in your object
- Caveat: in JS functions form closures
  - Every function creates its own “this”
  - Can lead to unexpected behavior

Inheritance in JavaScript

- Classes *extend* each other
- References to the superclasses' methods and constructors must use the *super* keyword
- If the superclass is not created using *class*, you must link the prototypes!

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
- Inheritance in ES6 syntax