Object Oriented JavaScript – Video Lecture Notes

**\*\*Everything in JS is an Object**

var x = new Object();

* Don’t need to use new 🡪 using ‘{ }’ makes it obvious

var x = {};

x.Name = “John”;

* Creates a String Object on the Name property of x
* The value of Name is a String Object

x.ID = 123;

* Creates an Integer Object on the ID property of x
* The value of ID is an Integer Object

**\*\*It’s Important to Scope Things**

* Instead of declaring a bunch of global variables, scope them in a namespace

var Main = {}; 🡪 Scopes everything on the Main object (namespace)

Main.X = {};

Main.X.Name = “John”;

Main.X.ID = 123;

**\*\*Changing Object & Variable names to make more sense**

var Main = {};

Main.Marie = {};

Main.Marie.Name = “Marie”;

Main.Marie.age = 20;

**\*\*Creating Objects using Object Literals:**

* Define properties inline
* Known as Literal Object Instantiation or Inline Object Instantiation
* JS object notation - JSON

\*comma delimited list of name value pairs where name & value separated by “:”

Main.Marie =

{

Name: “Marie”;

Age: 20;

}

**\*\*Using a** **Constructor:**

* The piece of the class that defines how you create Objects

Main.Person = function(firstName, lastName, age) {

this.firstName = firstName;

this.lastName = lastName;

this.age = age;

}

🡪 Creates a person objects that is a function

🡪 Creates a person, & each person will have the same variables

* By creating this constructor:
* You’ve created reusable code, (can reuse the function to create other Person Objects)
* That allows you to invoke the Person Function, which will create the 3 properties specified w/ the 3 values passed in the parameters

Main.Nate = new Main.Person(“Nate”, “Kelling”, 25);

**\*\*Objects can point to the same reference**:

Main.OldestSibling = Main.Nate;

* w/ assignment operator, the item on the right is an already existing object:
* It does not create a new object
* It does not create a copy of the object
* Simply points both objects at the same reference

\*Points the variable OldestSibling to the memory address of Main.Nate

**PROTOTYPE:**

\*\*Methods

\*\*Class Prototype Methods

\*\*Methods you’re attaching to a Constructor or an Object Type

Main.Person = function(firstName, lastName, age) {

this.firstName = firstName;

this.lastName = lastName;

this.age = age;

this.GetFullName = function() {

Return this.firstName + “ “ + this.lastName;

} 🡪 usually don’t want to declare methods in constructor. Instead use prototype attribute

}

**Prototype Attribute:**

\*Makes JS close to being OO w/out having abstract classes polymorphism, interfaces, etc.

* Allows you to declare/add methods to the ‘Person Class’ outside of the constructor

Main.Person.prototype.GetFullName = function()

{

Return this.firstName + “ “ + this.lastName;

}

Main.Person.prototype.SetFirstName = function(name)

{

this.firstName = name;

}