JavaScript OOP

1)

<http://javascriptissexy.com/oop-in-javascript-what-you-need-to-know/>

exercises

<http://www.w3resource.com/javascript-exercises/javascript-conditional-statements-and-loops-exercises.php>

books to get

[**https://www.google.com/search?q=javascript+algorithms+book&oq=javascript+algorithms+book&aqs=chrome..69i57.12633j0j4&sourceid=chrome&ie=UTF-8**](https://www.google.com/search?q=javascript+algorithms+book&oq=javascript+algorithms+book&aqs=chrome..69i57.12633j0j4&sourceid=chrome&ie=UTF-8)

[**https://mgechev.github.io/javascript-algorithms/**](https://mgechev.github.io/javascript-algorithms/)

**VERY IMPORTANT**

[**https://www.freecodecamp.com/challenges/sum-all-numbers-in-a-range**](https://www.freecodecamp.com/challenges/sum-all-numbers-in-a-range)

[**http://www.objectplayground.com/**](http://www.objectplayground.com/)

**Check out books from these guys….**

**Douglas Crockford**

**nicholas c. zakas**

**Namespace:** blakhb blah

1)sd

Object.prototype:

1)

Constructor:

1. It is important to reset the pointer of the constructor after you set your Object.prototype to a parent class because otherwise, when a new instance of your new object is created, its constructor will be pointing to the constructor of the parent and not of your child class!!!\*

EXP:

var Person = function(name){

this.firstName = name;

}

var Child = function(name){

Person.call(this, name);

}

Child.prototype = Object.create(Person.prototype);

Var myChild = new Child(‘javier’); 🡨 THAT WILL POINTING TO THE PARENT CONSTRUCTOR unless you do this.

Child.prototype.constructor = Child; 🡨 Resets the constructor pointer to the class and not the parent.

Encapsulation:

1. When a sub-class or child class does not need to redefine one of the methods of its parent unless with the intent to change what it does.

Abstraction

1)

Polymorphism

1. When sub class define the same methods as their parents but doing something different.
2. The object.prototype object bears all the properties and methods of what gets inherited by sub classes.

Object.create() Method

1)

Methods of Implementing OOP in JavaScript

1. The Combination Constructor/Prototype Pattern is one of the best way to implement OOP in JavaScript in terms of encapsulation. As for inheritance, we implement the parasitic combination inheritance.
2. A lot of practice is needed in mastering how to override functions (methods) of parent classes in an effective way.

Parasitic Combination Inheritance: OOP Method (BEST)

1. Out of all the methods to implement inheritance and encapsulation in JavaScript, this method is the best because it avoids creating duplicate properties and methods up the prototype chain!!! Especially even more so if you have a grandchild class inheriting from a child which that class inherits from a parent class.
2. This methods makes use of the Object.create() so make sure that it is supported otherwise use a shim!! Which is pretty simple to implement.
3. This method layouts out the prototype chain nicely so each instance inheriting from a parent class has the correct constructor up the prototype chain.
4. This method makes use of the .call() method inside the constructor function of the each class so any instance of that class will bear those properties as their own. Without this .call() only the methods and properties part of the prototype ‘object’ will be chained to its instances.
5. Over all it is the best way to implement inheritance.

<http://javascriptissexy.com/oop-in-javascript-what-you-need-to-know/>

**Check out books from these guys….**

**Douglas Crockford**

**nicholas c. zakas**