**Inheritance and JavaScript**

Most modern object oriented programming (OOP) languages support the concept of inheritance. Think of this as a child inheriting the functionality of its parent, where the child can add its own specifics. This is done with the concept of classes. Classes maybe thought of as blueprints.

So for example in C#, C++ or Java we may have a Vehicle class, parent blueprint, which will have steering wheel, engine and wheels. This Vehicle can be specified further with the following classes of Car and Lorry, child blueprints, which can be specified further and so on. The important thing is that Car and Lorry both inherit from the Vehicle class.

Vehicle

3-Axled

2-Axled

Lorry

Saloon

Estate

Car

Hatchback

Another way to look at this is that Car “is A” Vehicle and Hatchback “is A” Car. Inheritance of a class from a parent class exemplifies the ISA relationship. This is also known as classical Inheritance.

However JavaScript DOES NOT SUPPORT classical inheritance, it has no concept of classes. This is very important especially if you have learned another class-based language. JavaScript is Class-Free Object Oriented Language. It uses prototypal inheritance, which is where an object inherits from another object. So objects are based on other objects.

This is done with the prototype property. A prototype may be thought of as an object, from which other objects inherit properties. This may inherit from another object, effectively this is a chain of objects. Therefore object properties, which are defined using the prototype object, are inherited by all instances, which reference it. This can be a bit confusing so it may be demonstrated as follows.

An object and its property

obj

In code it can be accessed by **obj.prop1**

All objects in JavaScript have a prototype property, also an object. This prototype can also have a property, and this prototype is a reference to another object. This can be shown in the example below.

obj

proto{}

Then if we call **obj.prop2**

We are actually calling the prototype property prop2. It looks like prop2 is on the obj, however is actually on proto{}

Now the prototype may also point to another prototype in turn pointing to another prototype, forming a prototype chain.

obj

proto{}

proto2{}

Now in this example calling **obj.prop3**

Will actually call the property from proto2{}. We are calling down the prototype chain. We are seeing where we have access to property amongst a sequence of objects.

We can also have another object referencing proto{}, obj2.

obj2

obj

proto{}

proto2{}

Now if we call **obj2.prop1** then **obj.prop1** is accessing the same property. There are effectively sharing the same property, not directly but via the prototype chain.

In code terms here is an example

var person = {

firstName: ‘Default’,

lastName: ‘Default’,

getFullName: function(){

return this.firstName + ‘ ‘ + this.lastName;

}

}

//Now another object called ben

var ben = {

firstName: ‘Ben’,

lastName: ‘Smith’,

}

//This below is for demonstration purposes only and should not be used

//Can be done in most modern browsers.

//We are setting the prototype of the ben object to the person object

ben.\_\_proto\_\_.person;

//so ben now inherits from person, if we try to access a property or method on //ben which does not exist it will go to person.

//So now executing

console.log(ben.getFullName()); //will return Ben Smith

console.log(ben.firstName); //will return Ben NOT Default

//Create a new object will only have a first name

var colin = {

firstName: ‘Colin’

}

colin.\_\_proto\_\_.person;

console.log(colin.getFullName()); //will return Colin Default

The above was for example purposes only to illustrate the point. So to conclude, Prototypal Inheritance happens when an objects prototype attribute points to another object to inherit it’s attributes and methods.

There are some very good examples online. Below is a brief list of a few of them

<https://www.youtube.com/watch?v=qMO-LTOrJaE>

<https://www.youtube.com/watch?v=doXpW5AD60Q>

<http://javascript.crockford.com/prototypal.html>

<http://javascript.crockford.com/inheritance.html>

<http://stackoverflow.com/questions/2064731/good-example-of-javascripts-prototype-based-inheritance>