**JavaScript Factory Functions with ES6+**

A factory function is any function which is not a class or constructor that returns a (presumably new) object. In JavaScript, any function can return an object. When it does so without the new keyword, it’s a factory function.

Factory functions have always been attractive in JavaScript because they offer the ability to easily produce object instances without diving into the complexities of classes and the new keyword.

For Example:

let person = {firstName:"John", lastName:"Doe", age:50};

console.log(person.firstName) // John

We can do the same like this

const firstName = "John";

const lastName = "Doe";

const age = "50";

var person = { firstName, lastName, age};

console.log(person) // {firstName:"John", lastName:"Doe", age:50}

**Prototypes**

Understanding Prototypal inheritance in JavaScript.

\_ What is constructor in javaScript.

- How to add properties to prototype.

- How to create classes

\_ ECMA6, ECMA2015

- The root object in javascript

In Javascript, by default every function has a property called prototype and this property object is by default empty so we can add properties and methods in the prototype property of the function. When we create an object from the function the object inherits the prototype.

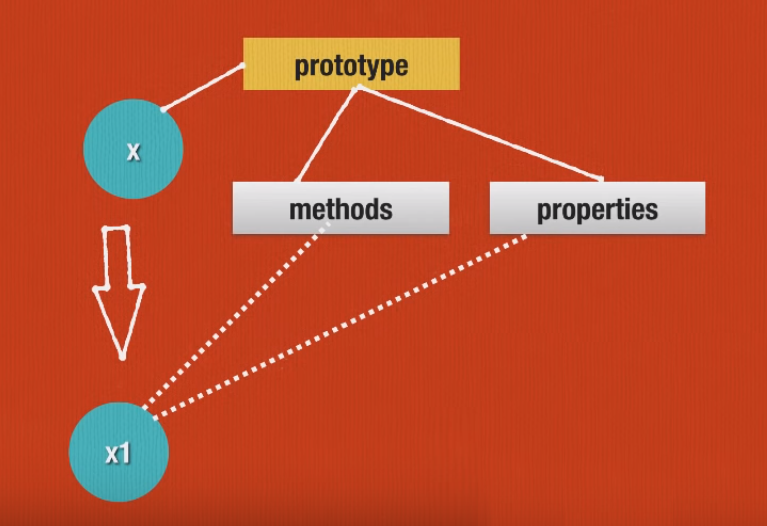
For example:

let Person = function (){

//

}

Person.prototype.propertyName;



**Constructor**:

Normally OOP Class and Javascript Class are not same actually the class in javascript is a constructor and it has prototype based inheritance. Every function expression is a constructor in javascript but we should use capital letter for separating constructor from the normal function.

For example:

let Person = function(age)

{

this.firstName="John";

this. lastName="Doe";

this. age=age;

this.fullName = function(){

return this.firstName +” ”+ this.lastName;

}

}

let p1 = new Person(50);

let p2 = new Person(40);

console.log(p1.fullName);

In the above constructor we can set prototype property like this later when an object will be created that will be inherited.

For example 02:

let Person = function(age)

{

this.firstName="John";

this. lastName="Doe";

this. age=age;

}

Person.prototype.fullName = function(){

return this.firstName +” ”+ this.lastName;

}

let p1 = new Person(50);

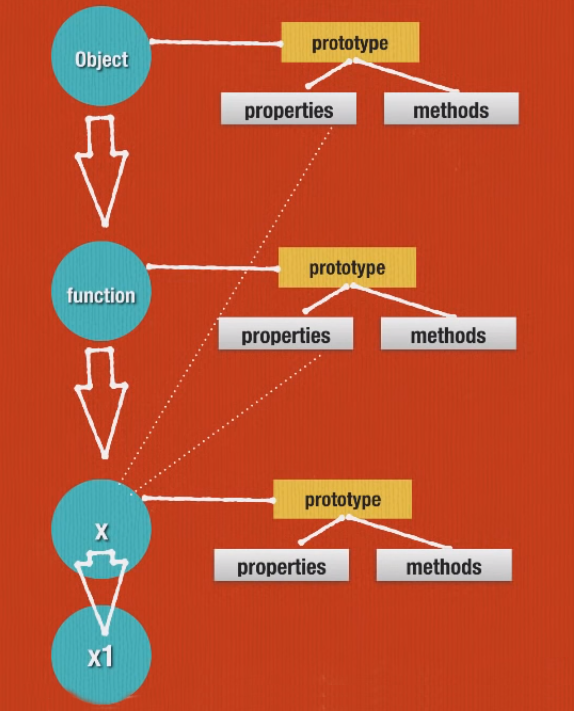
let p2 = new Person(40);

console.log(p1.fullName);

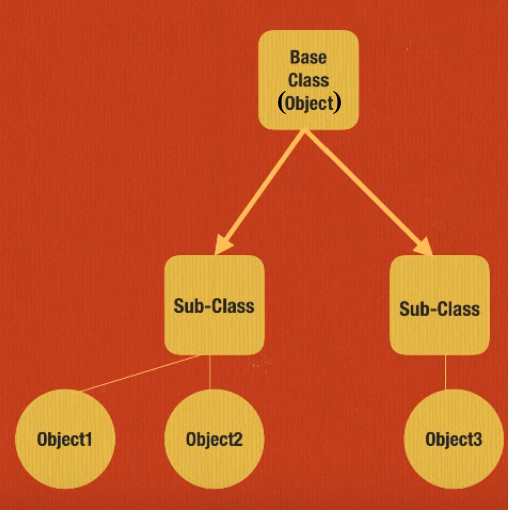
**Prototype Chain:**

console.dir(Person);

Then it will show details about prototype chain.



If we create an object from x then all of the properties and methods from the prototype chain will be inherited to that object like this

New topics to learn:

1. Creating Sub Class (Sub Constructor)
2. Overriding in Prototype Chain
3. Adding prototype to Master Object

**Base Class:**

//Base class

let Job = function ()

{

this.pays = true;

}

// prototype method

Job.prototype.print = function()

{

console.log(this.pays ? 'Please hire me' : 'no thank you');

}

// subclass

let TechJob = function(title,pays)

{

Job.call(this);

this.titile = title;

this.pays = pays;

}

TechJob.prototype = Object.create();