[Rupesh Mishra](https://hackernoon.com/@happymishra66?source=post_header_lockup)Follow

Full Stack Developer

May 5, 2017

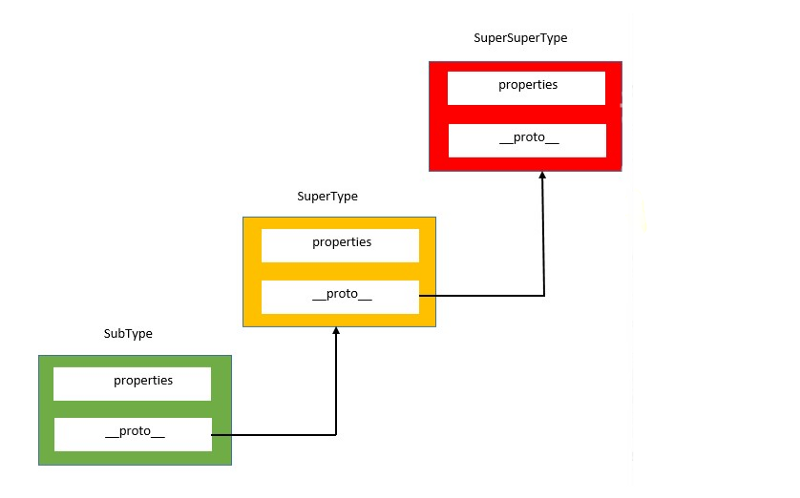
**Inheritance in JavaScript**

[Read on Github](https://github.com/happymishra/JavaScriptTutorials/blob/master/Part4/Inheritance%20in%20Javascript.md)

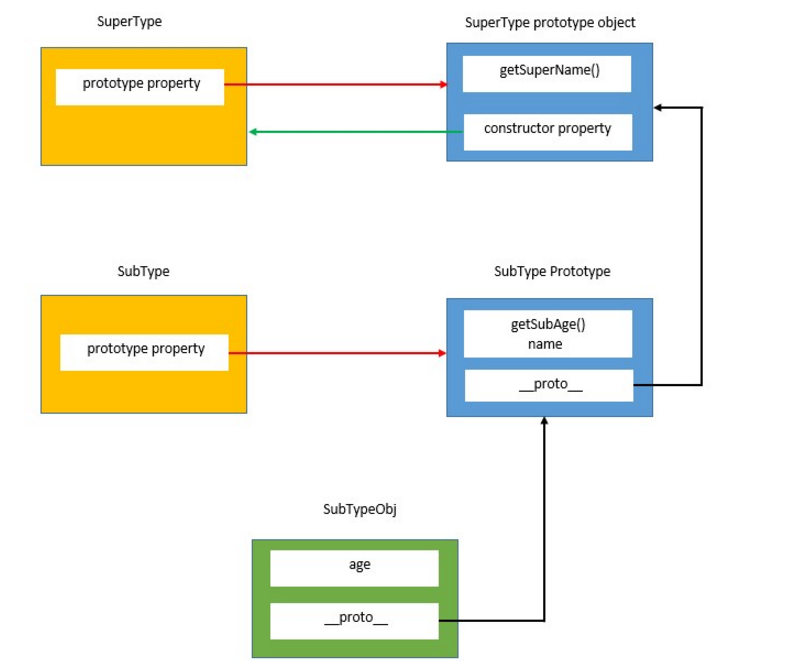
JavaScript does not have classes unlike other languages. It uses the concept of prototypes and prototype chaining for inheritance. If you do not know what is prototype, please go through [this article](https://medium.com/@happymishra66/prototypes-in-javascript-5bba2990e04b).

**Prototype Chaining**

Prototype chaining means an objects *dunder proto* or *proto* will point to another object instead of pointing to the constructor function prototype. If the other object’s *dunder proto* or *proto*property points to another object it will results into chain. This is prototype chaining.



Let’s implement prototype chaining



Above code defines two consructor functions, *SuperType* and *SubType*. By default, *SubType.prototype* has a *constructor*function which points to the *constructor function* itself and *proto* property which inherits the default object properties.

//Inherit the properties from SuperType  
SubType.prototype = new SuperType();

Above line rewrites the default prototype or he dunder proto property of the *SubType*constructon function and makes *SubType.prototype* to point to an object of *SuperType* constructor function.

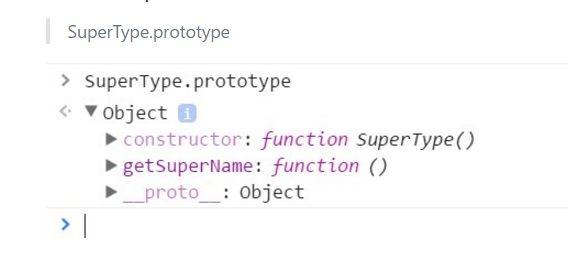
This means that all the properties and methods that typically exists on an instance of SuperType now also on *SubType.prototype*This means that now, *SubType*function has access to all the SuperType properties and methods.

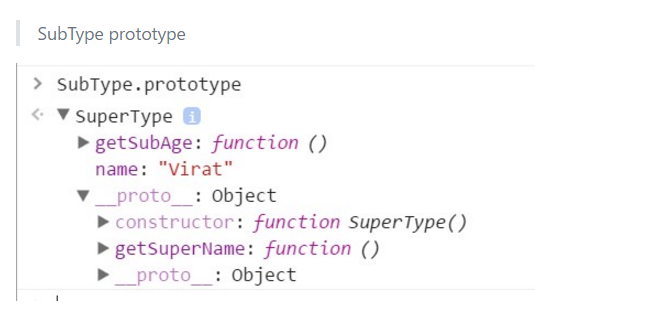
//Add new property to SubType prototype  
SubType.prototype.getSubAge = function(){  
 return this.age;  
}

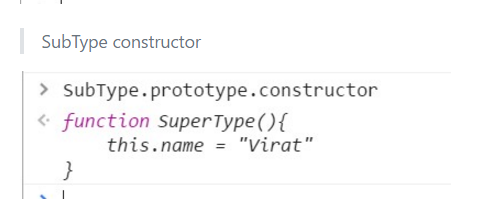
After the default prototype of *SubType*constructor function has been overwritten, by using the above line of code we add a new method *getSubAge()* on top of what was inherited from *SuperType*, to the prototype object of *SubType*constructor function.

**Note**: New methods must be added to the *SubType*after the inheritance because inheritance overwrites the existing prototype of *SubType*

**Console output**







***Note***: *getSuperName*() method remains on the *SuperType.prototype* object, but name property ends up on SubType.prototype. That’s because *getSuperName()* is a prototype method, and property is an instance property. *SubType.prototype* is now an instance of *SuperType*, so property is stored there. Also note that *SubType.prototype.constructor*points to *SuperType*, because the constructor property on the *SubType.prototype* was overwritten.

**Problems with prototype chaining**

As all the properties of the super type prototype are shared among the child objects, if one child modifies the property of the Super type prototype, other children also gets affected. This issue has been explained in great details [here](https://medium.com/@happymishra66/prototypes-in-javascript-5bba2990e04b)

To fix this issue, we use constructor to inherit the instance properties and prototype chaining to to inherit methods and share properties

Let’s try to understand the code First, we have defined a *SuperType*constructor function with *firstName*, *lastName* and friends as instance properties Then we defined a *superName*property on prototype of *SuperType*

Now, let’s look how we define the *SubType*constructor function

Here, we define a SubType constructor function. Inside the SubType constructor function, we call the SuperType constructor function with call. Call executes the SuperType constructor function in contecxt of the object begin created using the SubType constructor fucntion After inheriting the instance properties of the SuperType, we add one age property to the SubType constructor function

//Inherit methods and shared properties  
SubType.prototype = new SuperType();

So far we have just inherited all the instance properties of the *SuperType*constructor function, but the shared properties and methods of the *SuperType*constructor function are still not inherited. We inherit them using the above lines of code.

Once the above lines of code are executed, we have inherited all the properties of the *SuperType*constructor function

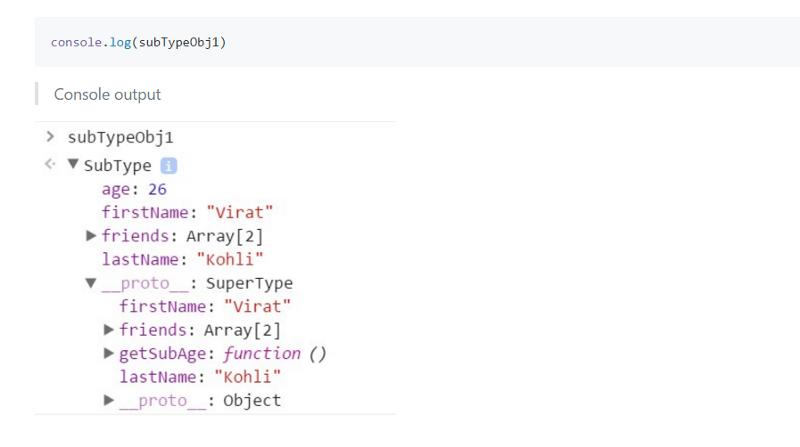
When we execute the above line of code, all the three parameters(Virat, Kohli and 26) are passed to the SubType constructor function. SubType constructor function, then calls SuperType constructor function using call *SuperType.call(this, firstname, lastName)* *this* here represent the *subTypeObj1*

SuperType constructor function is executed in the context of *subTypeObj1* and add propeties firstName, lastName, friends to the *subTypeObj1* object After return of *SuperType.call(this, firstname, lastName)*, SubType constructor function adds a *age* property to *subTypeObj1* object.

Thus as of now there are properties with the *subTypeObj1* object (firstName, lastName and age). Currently SubType constructor function has following methods and shared propertes in its prototype property:

1. getSuperName()
2. getSubAge

*subTypeObj1* inherits all these properties from SubType constructor function.



Further reads:

1. [Virtual DOM in ReactJS](https://medium.com/@happymishra66/virtual-dom-in-reactjs-43a3fdb1d130)
2. [Prototypes in JavaScript](https://medium.com/@happymishra66/prototypes-in-javascript-5bba2990e04b)
3. [Execution Context in JavaScript](https://medium.com/@happymishra66/execution-context-in-javascript-319dd72e8e2c)
4. [‘this’ in JavaScript](https://medium.com/@happymishra66/this-in-javascript-8e8d4cd3930)
5. [Create Objects in JavaScript](https://medium.com/@happymishra66/create-objects-in-javascript-10924cfa9fc7)
6. [Objects in JavaScript](https://medium.com/@happymishra66/objects-in-javascript-2980a15e9e71)