

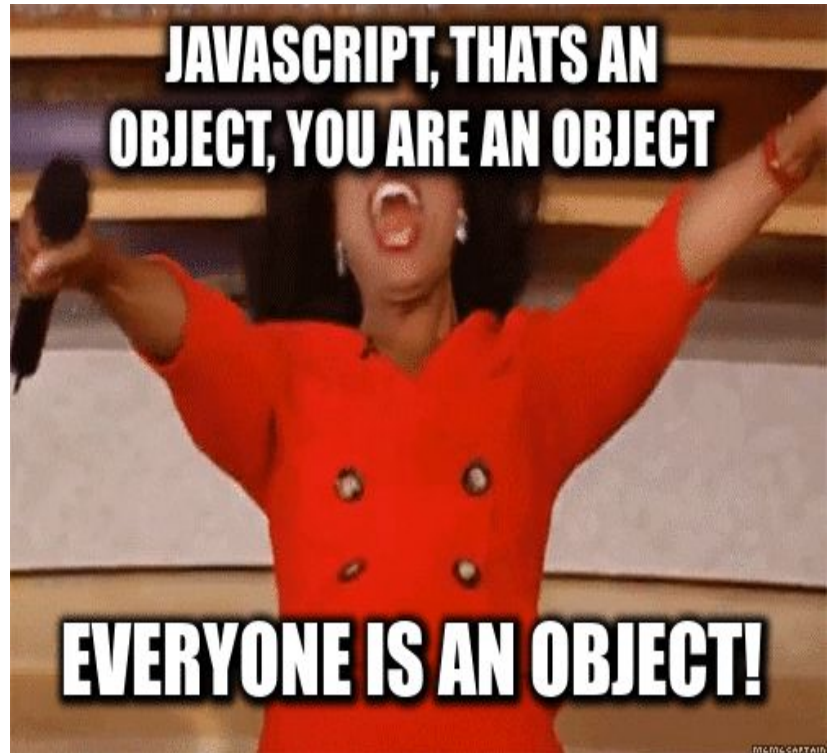


# Prototypal Inheritance

Key points to remember

# Everything in Js is an object

That is how JavaScript was built.



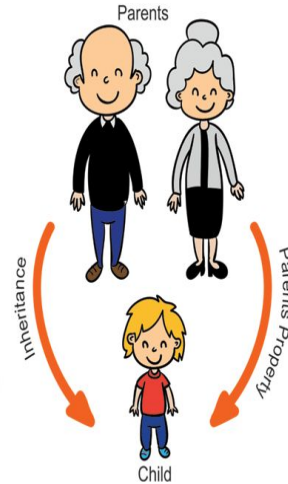


# What is prototypal inheritance.

To understand prototypal inheritance we need to understand these three key concepts, which are inheritance, prototype, prototype chain.

# What is inheritance ?

It is process where you get access to the properties and methods of another.





# What is a prototype ?

A prototype is just an object.

It allows properties and methods to be shared across all object.

```
var AppleInc = {
  name: "Apple",
  logo: "Apple fruit",
  operating_system: "Apple Software",
}

var iPhone = {
  name: "iPhone",
  operating_system: "ios"
}

console.log(iPhone)
```

output

```
▼ {name: 'iPhone', operating_system: 'ios'} ⓘ
  name: "iPhone"
  operating_system: "ios"
  [[Prototype]]: Object
    ▶ constructor: f Object()
    ▶ hasOwnProperty: f hasOwnProperty()
    ▶ isPrototypeOf: f isPrototypeOf()
    ▶ propertyIsEnumerable: f propertyIsEnumerable()
    ▶ toLocaleString: f toLocaleString()
    ▶ toString: f toString()
    ▶ valueOf: f valueOf()
```

```
var AppleInc = {
  name: "Apple",
  logo: "Apple fruit",
  operating_system: "Apple Software",
}


var iPhone = {
  name: "iPhone",
  operating_system: "ios"
}

Object.setPrototypeOf(iPhone, AppleInc)
console.log(iPhone)
```

output

```
▼ {name: 'iPhone', operating_system: 'ios'} ⓘ
  name: "iPhone"
  operating_system: "ios"
  [[Prototype]]: Object
    logo: "Apple fruit"
    name: "Apple"
    operating_system: "Apple Software"
    ▶ [[Prototype]]: Object
```

Notice, On the left `console.log()` iPhone does not have a `AppleInc` object property. But after we assigned `AppleInc` as its `prototype`, on the right `console.log()` we can see `AppleInc` object properties. Which means `iphone` object has inherited the properties of its `prototype` object (`AppleInc`).



As you see in the example `sample` string is split to turn that string into a array .but I did not write a function `split()`.

Instead `split` is linked to `sample` because of prototype inheritance.

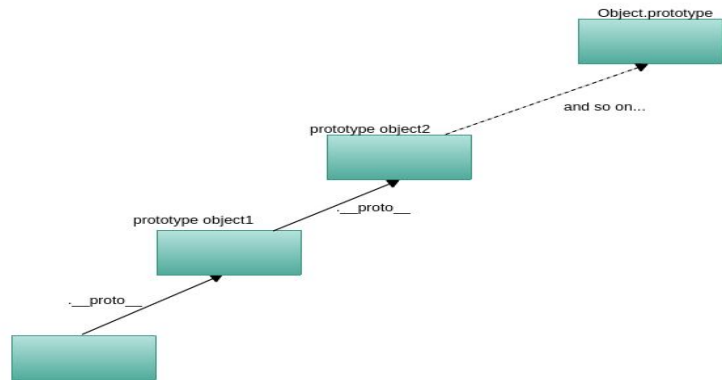
```
var sample= "hello, world";  
var Split= sample.split(",");  
  
console.log(Split);
```

```
▼ (2) ['hello', ' world'] ⓘ  
  0: "hello"  
  1: " world"  
  length: 2  
  ► [[Prototype]]: Array(0)
```

# What is an prototype chain ?

A prototype also keeps a reference to its own prototype object. And, prototype's prototype is also linked to its own prototype and so on. This is how it forms prototype chain.

JavaScript uses this link between an object and its prototype to implement inheritance which is known as prototypal inheritance.







## How to build a prototype inheritance?

We have `Object.create()` method to implement prototypal inheritance.

`Object.create()` takes an object as an argument and returns a new object with its `_proto_` set to the object that was passed as argument into `Object.create`.



Here in the example teacher creates a object using person by `Object.create()`

This teacher has properties of person.

In the example you can see `Object.getPrototypeOf(teacher)===person` which lets us know if teacher has person prototype or not.

```
const person = {
  name: "raju",
  greet: function(){
    console.log(`Hi, I'm ${this.name}`);
  }
}

const teacher = Object.create(person);

teacher.teach = function (subject) {
  console.log(`I can teach ${subject}`);
}

teacher.greet();
teacher.teach("JavaScript");
console.log(Object.getPrototypeOf(teacher) === person);
```

output

```
Hi, I'm raju
I can teach JavaScript
true
>
```