

prototype:

```
> let heroPower = {
  thor: "hammer",
  spiderman: "sling",

  getSpiderPower: function () {
    console.log(`Spidy power is ${this.spiderman}`);
  },
};

< undefined
> heroPower
< {thor: 'hammer', spiderman: 'sling', getSpiderPower: f} i
  ▶ getSpiderPower: f ()
  spiderman: "sling"
  thor: "hammer"
  ▼ [[Prototype]]: Object
    ▶ constructor: f Object()
    ▶ hasOwnProperty: f hasOwnProperty()
    ▶ isPrototypeOf: f isPrototypeOf()
    ▶ propertyIsEnumerable: f propertyIsEnumerable()
    ▶ toLocaleString: f toLocaleString()
    ▶ toString: f toString()
    ▶ valueOf: f valueOf()
    ▶ __defineGetter__: f __defineGetter__()
    ▶ __defineSetter__: f __defineSetter__()
    ▶ __lookupGetter__: f __lookupGetter__()
    ▶ __lookupSetter__: f __lookupSetter__()
    __proto__: (...)
    ▶ get __proto__: f __proto__()
    ▶ set __proto__: f __proto__()

> Object.prototype.printJohn = function () {
  console.log(`John is everywhere`);
};

< f () {
  console.log(`John is everywhere`);
}

> heroPower
< {thor: 'hammer', spiderman: 'sling', getSpiderPower: f} i
  ▶ getSpiderPower: f ()
  spiderman: "sling"
  thor: "hammer"
  ▼ [[Prototype]]: Object
    ▶ printJohn: f ()
    ▶ constructor: f Object()
    ▶ hasOwnProperty: f hasOwnProperty()
    ▶ isPrototypeOf: f isPrototypeOf()
    ▶ propertyIsEnumerable: f propertyIsEnumerable()
    ▶ toLocaleString: f toLocaleString()
```

Initially heroPower object don't have the method **printJohn**, once we added it in prototype :

```
Object.prototype.printJohn = function () {
  console.log(`John is everywhere`);
};
```

Then we got the method **printJohn** in object

Now if we want to access it then :
<object_name>.<method_name()>

```
heroPower.printJohn();
```

output : John is everywhere

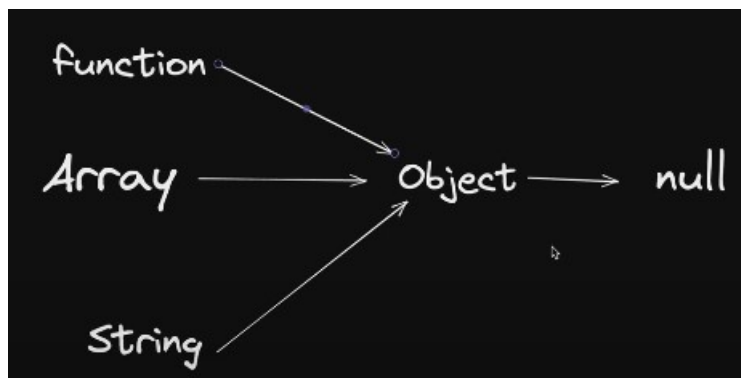
-As we know everything prototype is object. It means array pro. Is object only

so According to theory array should have have printJohn method, as as add it in object.

```
> let myHeros = ["thor", "spiderman"];  
< undefined  
> myHeros.printJohn();  
John is everywhere  
< undefined
```

Yes in array also it's available

what we did here is, we add method in top level hierarchy I.e, object & add method in it .



If we add anything in object, it will be automatically accessible in array as well.

CASE 2 : if we add the method in array, will it accessible in object?

```

> let myHeros = ["thor", "spiderman"];
< undefined
> myHeros
< ▶ (2) ['thor', 'spiderman']
> Array.prototype.sayMyName = function () {
  console.log("Your name comes from an Array !");
};
< f () {
  console.log("Your name comes from an Array !");
}
> myHeros
< ▼ (2) ['thor', 'spiderman'] ⓘ
  0: "thor"
  1: "spiderman"
  length: 2
  ▼ [[Prototype]]: Array(0)
    ▶ sayMyName: f ()
    ▶ at: f at()

```

```
myHeros.sayMyName();
```

If we try to access sayMyName from array, we are able to do that.
o/p: Your name comes from an Array !

Lets try if we can access from object also, like when we add in object we are access from array also :

```
heroPower.sayMyName(); //TypeError: heroPower.sayMyName is not a function
```

we are not able to access from object ,because we have added prototype method in array.

Parent to child is possible
child to parent not possible