# Objects predefined methods

## 1- Object.create()

Objects can be created using the Object.create().

```
var person = {
            name: "Rowan",
            displayName() { console.log(this.name)},
        };
        var person1 = Object.create(person);
        person1.displayName();
```

## 2-Object.prototype.valueOf()

converts this value to an object.

```
function MyNumberType(n) {
            this.number = n;
        }
        MyNumberType.prototype.valueOf = function () {
            return this.number;
        };
        var object1 = new MyNumberType(4);
        console.log(object1 + 3);
```

## 3- toLocaleString()

toLocaleString() method  calls toString().

```
var obj = {
            toString() {
                return "My Object"; }
        };
        console.log(obj.toLocaleString());
```

## 4- __defineGetter__()

method binds an object's property to a function, __defineGetter__(prop, func)
prop is a name of prototype that getter function bound

```
var obj = {};
        obj.__defineGetter__("gimmeFive", function () {
            return 5;
        });
        console.log(obj.gimmeFive);
```

## 5- get()

get look like __defineGetter__() but I doesn't take function and prop it's merge them

```
const obj = {
            get gimmeFive() {
                return 5;
            },
        };
        console.log(obj.gimmeFive);
```

## 6- Object.getPrototypeOf()

It's return prototype of object

```
 var prototype1 = {};
        var object1 = Object.create(prototype1);
        console.log(Object.getPrototypeOf(object1) ===
prototype1);
```

## 7- Object.hasOwn()

returns true if the object has the indicated property as its own property. Otherwise false

```
var object1 = {
      prop: ''
  };
  console.log(Object.hasOwn(object1, 'prop'));
  //output: true

  console.log(Object.hasOwn(object1, 'toString'));
  //output: false
```

## 8-9-call() and apply()

The call() method calls the function with a given this value and. it's almost like apply() except that call() accepts argument list.

apply() accepts a single array of arguments.

```
func.apply(this, ['eat', 'bananas'])
func.call(this, 'eat', 'bananas').
```

## 10- isPrototypeOf()

checks if an object exists in another object's prototype chain.

```
function Foo() { }
      function Bar() { }
      var bar = new Bar();
      Bar.prototype = Object.create(Foo.prototype);
      console.log(Foo.prototype.isPrototypeOf(bar));
      // Expected output: true
```