* Creating An Object

```
const rectangle = {

    length : 1,
    breadth : 2,

    // a function that is created inside an object
    // is known as a method.
    draw : function() {
        console.log('lets draw');
    }
};
```

* Object Creation by —
    * Factory Function
    * Constructor Function

* Factory Function

```
function createRectangle (len, bre) {

    return rectangle = {
        length : len,
        breadth : bre,
        draw() {
```

```
        console.log('lets draw');
    };
};

// factory function calling
const rectangle0bj = createRectangle(5,4);
console.log(createRectangle(5,4));
```

Constructor Function

In this function we will us pascal notation.

Pascal Notation --> NumberOfStudents

Constructor function --> properties | methods -->

it initialize | define and doesn't return

```
function Rectangle() {
    // 'this' keyword represents the current
    // object we are working with.
    this.length = 1;
    this.breadth = 5;

    this.draw = function() {
        console.log('Drawing');
    };
}
```

Object creation using constructor

// 'new' keyword returns an empty object.

```
let rectangleObj = new Rectangle();
```

* Dynamic Nature of Objects

Addition / removal of any property in the object is possible.

```
// creation
retangleObj.color = 'yellow';
// deletion
delete rectangleObj.color;
```

* Constructor Properties

⇒ Functions are objects

In JS functions are also known as objects.

⇒ We can us '.' operator in any function and we will see many properties of it.

Types in JS

→ Primitive or value types
  —→ here copy is created

→ Reference or object types
  —→ here address is passed

* Primitive
  → Number
  → String
  → Boolean
  → Null
  → Symbol

* Reference
  → Functions
  → Objects
  → Arrays

* Iteration through Objects
  * For-of
  * For-in

* For-of

It is only used in iterables i.e. arrays and maps, etc.

Syntax

```
for (variable of iterable) {
    // statement
}
```

* For-in

→ It is only used in objects.

→ It is used to check whether any property is present or not in the object.

Syntax

```
for (key in object) {
    // statement
}
```

* Object Cloning

It is used to create a copy of original object with same properties included in it.

⟹ Using assign

Ex // original

```
const employee = {
    name : 'Rishabh',
    age : 20,
    height : 5.8
}
```

// cloning

```
const copyEmp = employee; Object.assign
                ({}, employee);

console.log(copyEmp);
```

=> Using iteration

```
const object = {a:1, b:2, c:3};
// using for-in loop
for (const property in object) {
    console.log(`${property}; ${object[property]}`);
}
```

// `` = backtick character

=> Using spread operator (...)

↓ It is created above

```
let copyEmp = {... employee};

console.log(copyEmp);
```

* Garbage Collector

→ It is used to deallocate the memory.

→ It runs ~~at~~ automatically in background.

→ We have no contol over it, mean we cannot contol the ~~start~~ / end of it.