JAVASCRIPT Basics

Object in JS

- Encapsulating the properties and function(behavior) in a single entity or multiple linked variables can be stored in a single entity can be stored in Object.
- Example:
    - const rectangle = {
      length: 1,
      breadth: 2
      };

```
const rectangle = {
    length: 1,
    breadth: 2,
    draw: function() {
        console.log('Area is : ' + this.length * this.breadth);
    }
};

console.log(rectangle);
```

    - if we are creating a function in object we call it as method.
- These are basically called as Object Oriented Programming
- Now if we want to create multiple objects of same types
- Two ways to do this
    - Factory Function

```
// Factory Function
function createRectange() {
    let rectangle = {
        length: 1,
        breadth: 2,
        draw: function() {
            console.log('Area is : ' + this.length * this.breadth);
        }
    };
    return rectangle;
}
// now we will call this funciton and change the values
```

        - Here but the main issue is that the values are constant and we want that the value should be updated at the time of creation then there we can use parameters
    - Constructor Function
        - Pascal Notation is used (NumberOfStudents)

```
// Constructor Function

function Rectangle(len, brea) {
    this.length = len;
    this.breadth = brea;
    this.draw = function() {
        console.log('Area is : ' + this.length * this.breadth);
    }
}
// here only initiallization of the variable are done no necessary to return it
let rec1 = new Rectangle(5, 5);
console.log(rec1.draw());
```

- Dynamic Nature of Object
    - We can add, update, remove the property after declaration

```
let rec1 = new Rectangle(5, 5);
console.log(rec1.draw());

rec1.height = 10;
console.log(rec1);
rec1.collor = 'Black';
console.log(rec1);

delete rec1.height;
console.log(rec1);
```

- Constructor Property
    - It is a predefined function which initialize the object or create the object.
    - Every object has a constructor function
    - If we have created our own constructor then that function will also have inbuilt constructor function.

```
let Rectangle1 = new Function(
    'length', 'breadth',
    `this.length = length;
    this.breadth = breadth;
    this.draw = function() {
        console.log('Area is : ' + this.length * this.breadth);
    }`);

let rect = new Rectangle1(4, 3);
console.log(rect.draw());
```

    - Using back-tick character to give the body of the constructor function.

Functions are Object?

- Yes, functions are also object.

```
let Rectangle1 = new Function(
    'length', 'breadth',
    `this.length = length;
    this.breadth = breadth;
    this.draw = function() {
        console.log('Area is : ' + this.length * this.breadth);
    }`);

let rect = new Rectangle1(4, 3);
console.log(rect.draw());

console.log(Rectangle1.length);
```

- Here we can see we can use directly length using the function name.

Difference between the Primitive Types and Reference Types

- Reference (address) is same in reference types but not in primitive types.

```
let check1 = {
    a: 10
};

let check2 = check1;
++check2.a;
console.log(check1.a + " " + check2.a); // it will print 11 11

let a1 = 10;
let a2 = a1;
++a1;
console.log(a1 + " " + a2); // it will print 11 10
```

```
let a3 = 10;
function inc(a) {
    a++;
}
inc(a3);
console.log(a3); // it will print 10 not 11
//because copy is passed not the reference
```

To check if any properties is present or not

- Simply use if else

```
// to check if any property is present in it or not
if ('color' in rectangle1) {
    console.log('Yes present');
}
else {
    console.log('Not present');
}
// it will print Not Present
```

Iterating through Object

- **For in Loop**
  - To iterate the property

```
// For in Loop
for(let key in rectangle1) {
    console.log(key);
}
// it is used to iterate to all keys present in that object
```

  - Here rectangle1 is an object.
- For of Loop
  - It is used on iterrable like arrays, maps etc.
  - It cannot be used on object it will give error
  - But if we want to use it on Object we need to convert that object into arrays.

```
// For of loop
let arr = [1, 2, 3, 4, 5];
for(let el of arr) {
    console.log(el);
}

// we have converted rectangle1 object into arrays of its keys
for(let key of Object.keys(rectangle1)) {
    console.log(key);
}
```

  - To convert its entries simply write Object.entries(rectangle1)

Object Cloning (different address or memory space for different variable)

- Iteration:
  - Create empty object
  - Use for in loop and copy the values

```
// Iteration
let src = {
    value: 10,
    firstName: 'Src1'
};

let dest = {};
for(let key in src) {
    dest[key] = src[key];
}
console.log(dest); // it will print same as src but not same address
```

- Assign:
  - Using assign keyword
  - let dest = Object.assign({}, src);

```
//Assign
let dest2 = Object.assign({}, src);
console.log(dest2);
```

  - here empty object is being filled by src object values.
  - We can use multiple src object like {}, src1, src2 and so on. It will copy all in that empty object.
- Spread:
  - Using …obj name

```
// Spread
let dest3 = {...src};
```

Garbage Collection

- Automatically free up the space which are not being used currently, done via garbage collector.
- We have no control on garbage controller. Runs in the background.