**Objects in JavaScript :**

* basics

//   object

const circle = {

  radius : 1,

  location :{

    x : 1,

    y : 1

  },

  isvisible : true,

  draw: function() {

    console.log("draw function");

  }

};

circle .draw();    //  method

* Factory Function

In JavaScript, a factory function is a function that returns an object. It is a way of creating and returning objects in a more controlled and customizable manner. Factory functions are a form of design pattern that enables the creation of objects with specific properties and behaviors.

**Why it is useful?**

If we have complex logic, and we have to create multiple objects again and again that have the same logic, we can write the logic once in a function and use that function as a factory to create our objects. It’s the same as a real-world factory producing products.

function creteCircle (radius) {

   return {

    radius,

    draw (){

        console.log('draw');

       }

   };

};

 const circle1 = creteCircle (1);

 console.log(circle1);

 const circle2 = creteCircle (2);

 console.log (circle2);

* Constructor Function

Sometimes we need to create many objects of the same **type**.

To create an **object type** we use an **object constructor function**.

It is considered good practice to name constructor functions with an upper-case first letter.

  function Circle(radius) {

        this.radius = radius;

        this.draw = function(){

            console.log('draw');

        }

    }

    const circle = new Circle(1);

* Dynamic Nature of Objects

 const circle = {

    radius : 1

 };

 circle.color = 'yellow';

 circle.draw = function () {}

delete circle.color; // this well delete circle.color

 console.log(circle);

* Constructor property

The constructor property returns the function that created the Object prototype.

For JavaScript objects the constructor property returns:

* Functions are objects

Functions are aslo an object

* Value Vs Reference types

|  |  |
| --- | --- |
| values | Reference |
| Number | Object |
| String | Functions |
| Boolean | Array |
| Symbol |  |
| Undefined |  |
| Null |  |
|  |  |

Primitives are copied by their values

Objects are copied by their reference

// value

// let x = 10 ;

// let y = x ;

                // in this case the value of x is updated to 20 but the valaue of y is 10

// x = 20;

// reference

let x = { value : 20};

let y = x;

x.value = 30;

// another example

let obj = {value : 10};

function increase (obj) {

   obj.value++;

}

increase(obj);

console.log(obj)

* Enumerating properties of an object

const circle = {

    radius : 1,

    draw : function (){

        console,log(draw)

    }

};

// for in

for (let key in circle)

    console.log(key , circle[key]);

// for of is only use for array and maps

for (let key of Object.keys(circle))   // keys is a object constructor method

    console.log(key);

    //

    for (let entry of Object.entries(circle))

        console.log(entry);

    //

    if ('draw' in circle)

        console.log('yes');

* Cloning of an object

 Object cloning is a way to create an exact copy of an object. Means that inherting all the properties of an another object into itself.

    const circle = {

        radius : 1,

        draw () {

            console.log('draw')

        }

    }

    // first way

    // const another = {};

    // for (let key in circle)

    //     another[key] = circle[key];

    // console.log(another);

    // 2nd way

    //   use object.assign

    // const another = Object.assign ({} , circle);

    // console.log(another);

    // 3rd way

    // use ...

    const another = {...circle};

    console.log(circle);