**ES-6**

1. Spread operator.
2. Rest operator.
3. Array destructuring.
4. Object destructuring.
5. Primitive type.
6. Reference type.
7. Map function.
8. Filter function.
9. **Spread Operator**

The Spread operator is used to **copy one array** and object into another array and the object.

**Array**

<script>

       let arr= [12, "Nazim", true];

       console.**log** (arr [2])*//true*

       console.**log** (...arr)*//12 Nazim true*

       let newarr=[...arr, 'Dehradun']

*//newarr.push(35)*

       console.**log**(newarr)*// [ 12, "Nazim", true, "Dehradun", 35]*

    </script>

Spread operator with array.

**Object**

<script>

        let person= {

            name: "Nazim",

            add: "Dehradun",

            age:35

        }

        let nazim= {

            ...person,

            employment: "Ecole"

        }

        console.**log**(nazim);

    </script>

Spread operator with object

1. **Rest Operator**

When it (…) is used inside the parameter of a function, it is called rest operator.

    <script>

       function **sum** (a, b, c)

       {

        console.**log**(a+b+c)

       }

**sum** (10,20,30)*// mandatory to pass three arguments since we pass 3 parameter*

*//60*

    </script>

The problem is mandatory to pass three arguments since we pass 3 parameters

<script>

       function **sum** (...all)

       {

        console.**log**(all)*//[ 10, 20, 30 ]*

       }

**sum** (10,20,30)

</script>

1. **Array Destructuring**

Array

<script>

        function **myfun** () {

            return("hello")

        }

        function **greeting**(){

            console.**log**("goodmoring")

        }

       let arr=[12, "Nazim", **myfun**(), **greeting**];

       console.**log**(arr)*//[ 12, "Nazim", "hello", greeting() ]*

       let[val1, val2, val3, val4]=arr;

       console.**log**(val3)*//hello*

       console.**log**(**val4**())*//goodmoring*

   </script>

Array destructuring

1. **Object Destructuring**

**Object**

<script>

            let person = {

            name:'nazim',

            city: 'Dehradun',

            employment: 'Ecole',

**hello**: function (greeting) {

                console.**log**(greeting)

            }

        }

        person.**hello**('good morning')*// good morning*

        const {name, city, employment, **hello**} = person;

**hello** ("noon");*//noon*

        console.**log**(name);*//nazim*

    </script>

Object destructuring

let obj={

        name: "Nazim",

        age:35

    }

    let{name}=obj ;

    console.**log**(name);

object destructuring takes any property

<script>

        function **myfun**(n){

            return n;

        }

        let obj ={

            name:**myfun**("nazim"),

            age:35,

**add**:function(address){

                console.**log**(address)

            }

        }

        console.**log**(obj.name);*//nazim*

        console.**log**(obj.**add**("ajabpur"));*//ajapur*

        const {name,age,**add**}=obj;

        console.**log**(name)*//nazim*

        console.**log**(**add**("ajabpur"));*//ajabpur*

    </script>

Object destructuring

1. **Primitive type**

**Number and string**

<script>

            let num1= 100;

            let num2=num1;

            console.**log**(num1, num2)*//100 100*

            num2=150;

            console.**log**(num1, num2)*//100 150*

    </script>

Value copied into another variable both are treated as separate

1. **Reference type**

**Array and object**

<script>

           let person= {

            name:"nazim",

            age:35,

            city:'Dehradun'

           }

           let person1 = person;

        person.name="nazim khan"

        console.**log**(person)*//{ name: "nazim khan", age: 35, city: "Dehradun" }*

        console.**log**(person1)*//{ name: "nazim khan", age: 35, city: "Dehradun" }*

    </script>

With object

<script>

         let arr=[10, 20, 30];

         let arr1=arr;

         arr1[1]=40

         console.**log**(arr)*//[ 10, 40, 30 ]*

         console.**log**(arr1)*//[ 10, 40, 30 ]*

    </script>

With Array