function AddArray()

     {

       var no =[1,2,3,4,5,6,7,8,9,10];

       var sum=0;

       for(var i=0;i<no.length;i++)

       sum+= no[i];

       console.log(sum);

       var res = no.map(item=> { sum+= item;})

       console.log(sum);

     }

const record = [{

     name: "Ajay",

     age: "34",

     address:

     {

         houseno: "8A",

         state: "Delhi",

         city: "New Delhi"

     }

  },

  {

     name: "Deepak",

     age: "32",

     address:

     {

         houseno: "8A",

         state: "Delhi",

         city: "New Delhi"

     }

  }

  ];

record.map(rec=>

  {

    console.log(rec.name);

    console.log(rec.address.houseno)

  })

const record = [{

     name: "Ajay",

     age: "34",

     address:

     {

         houseno: "8A",

         state: "Delhi",

         city: "New Delhi"

     }

  },

  {

     name: "Deepak",

     age: "32",

     address:

     {

         houseno: "8A",

         state: "Delhi",

         city: "New Delhi"

     }

  }

  ];

record.map(rec=>

  {

    console.log(rec.name);

    console.log(rec.address.houseno)

  })

  record.map(rec=>

    {

 const {name, age, address} = rec;

 console.log(name, age, address);

    });

**Object.assign()**

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/assign>

const target = { a: 1, b: 2 };

const source = { b: 4, c: 5 };

const returnedTarget = Object.assign(target, source);

console.log(target);

// expected output: Object { a: 1, b: 4, c: 5 }

console.log(returnedTarget);

// expected output: Object { a: 1, b: 4, c: 5 }

  var obj = Object.assign({} , a1, b1, c1);

      console.log(obj);

     // obj.a= 10;

      //console.log(obj);

      console.log(a1);

      console.log(b1); console.log(c1);

 //  var obj = Object.assign(a1,b1,c1);

      //  console.log(a1);

      //  console.log(b1);

      //  console.log(c1);

      //  console.log(obj);

const o1 = { a: 1, b: 1, c: 1 };

const o2 = { b: 2, c: 2 };

const o3 = { c: 3 };

const obj = Object.assign({}, o1, o2, o3);

console.log(obj); // { a: 1, b: 2, c: 3 }

What is Deep Copy & Shallow Copy

<https://developer.mozilla.org/en-US/docs/Glossary/Deep_copy>

734

ECMAScript 6 introduces the easily polyfillable [Object.assign](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/assign):

The Object.assign() method is used to copy the values of all enumerable own properties from one or more source objects to a target object. It will return the target object.

Object.assign({}, ['a','b','c']); // {0:"a", 1:"b", 2:"c"}

The own length property of the array is not copied because it isn't enumerable.

Also, you can use ES8 [spread syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax) on objects to achieve the same result:

{ ...['a', 'b', 'c'] }

For custom keys you can use [reduce](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce):

['a', 'b', 'c'].reduce((a, v) => ({ ...a, [v]: v}), {})

// { a: "a", b: "b", c: "c" }

var a1 = [1,2,3], a2 = [];

a2 = a2.concat(a1);

// or ES5 shorten alternative

// var a1 = [1,2,3], a2 = [].concat(a1);

// or better ES6 shorten alternative

// a2 = [...a1];

a2.push(4);

console.log(a1, a2);

// a1 -> [1,2,3], a2 -> [1,2,3,4]