Date: 01 - 06 - 2020

Morning Session: 9 am – 11 PM ;

By ~ Sundeep Charan Ramkumar Today

Topics: Arrays and Objects

Data types : are two types 1) primitive 2) reference

1) **Primitive:** are those whose value can be copied making sure that copied value will not get damaged basic data types(number , string , nul , undefined)
   Ex: var a = 10;
   Var b = a;
   B = 15;
   console.log(a,b) // 10, 15
2) **Reference:**
   Ex: var a = [1,2,3];
   Var b = a;
   B = [1,2,3,4];
   console.log(a,b); // [1,2,3,4] , [1,2,3,4]

Arrays: allow you to store collections of data and the data is organize in a sequential pattern inside your memory and finally hold memory referenced by single variable **(a = [5, "yodraj", ture, null])**

We can create an array with help of " [ ] " (square brackets)

Var sampleArray = [ ]; // Empty Array
Var sampleArray = [2210,, "yodraj"] //array

```
// Array
var sampleArray = [1, "hi there", 1.44, false, undefined];
var newArray = sampleArray;
newArray = [1, 2, 3];
console.log(newArray, sampleArray);
```

indexing: Access the Elements of an Array access an array element by referring to the index number.

Note : Array indexes start with 0. [0] is the first element. [1] is the second element. Or count with real world number and subtract with 1

```
//indexing

var a = [5,6,7] // a[0] = 5, a[1] = 6 , a[2] = 7
console.log(a[0]);
```

Replacing element:

```
//replacing element
var a = [5,6,7] // a[0] = 5, a[1] = 6 , a[2] = 7
a[0]  = 3;
console.log(a); // result = [3,6,7]
```

Nested Array:

```
> var matrix = [[1, -2, 4], [-6, 7, 8], [9, 2, 6]]
< undefined
> matrix
< ▼(3) [Array(3), Array(3), Array(3)]
  ▼0: Array(3)
     0: 1
     1: -2
     2: 4
     length: 3
   ▶__proto__: Array(0)
  ▼1: Array(3)
     0: -6
     1: 7
     2: 8
     length: 3
   ▶__proto__: Array(0)
  ▼2: Array(3)
     0: 9
     1: 2
     2: 6
     length: 3
   ▶__proto__: Array(0)
   length: 3
  ▶__proto__: Array(0)
```

Find length of array:

```
//length of array
var a = [1,2,3,4,5,6]
console.log(a.length); // length of array = 6
```

Traversing an array :

```
// Traversing an array
var oneDimensionalArray = [4, 6, 8, 10, 12, 14];

for (
  var index = 0;
  index <= oneDimensionalArray.length - 1;
  index = index + 1
) {
  console.log(oneDimensionalArray[index], "Current Index " + index);
}
```

For of loop :

```
// for of loop
var oneDimensionalArray = [4, 6, 8, 10, 12, 14];
for (value of oneDimensionalArray) {
    console.log(value);
  }
```

Object : another way to store collection of data(like array) It's defined as (" { } ") curly braces.

```
var user = {
      name:"yodraj",
      from: "hyderabad",
      pincode: 500009,
 };
 console.log(user);
```

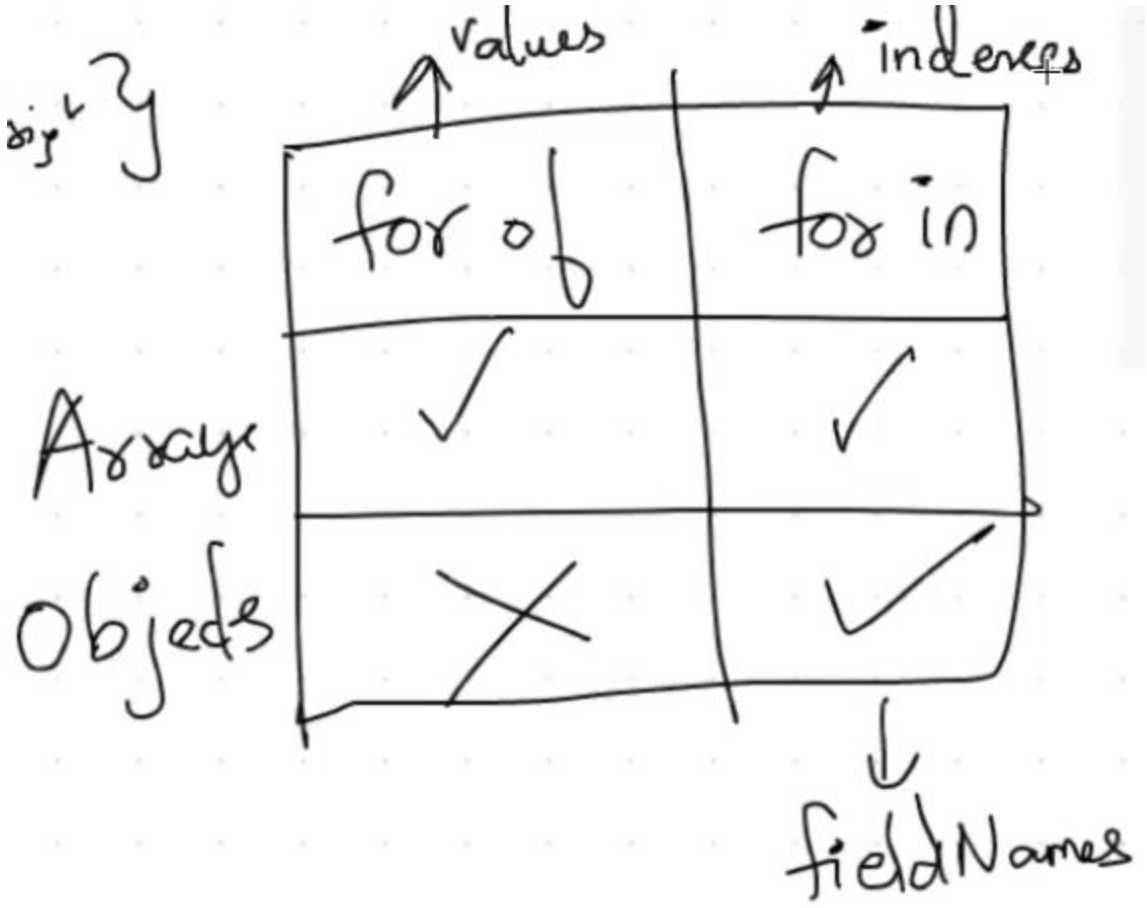
To Access elements in objects

```
// to access elements in objects
var user = {
    name:"yodraj",
    from: "hyderabad",
    pincode: 500009,
};
console.log(user.from); // result  =  hyderabad // array notation
// or u can access like this
console.log(user["from"]); // result = hyedrabad // dot notation
```

Traversing through an object:

```
// Traversing through an object
var sundeep = {
    name: "Sundeep Charan Ramkumar",
    age: 21,
    location: "Coimbatore",
    isMacHolder: true,
  };
for (fieldName in sundeep) {
    if (fieldName === "isMacHolder") {
      console.log("We have reached the limit");
      break;
    }
    console.log(typeof fieldName, sundeep[fieldName]); // sundeep["name"] // sundeep["age"]
    // sundeep ["location"]
  }

  for (index in oneDimensionalArray) {
    console.log(index, typeof index);
  }
```



Objects can not use for of but it can use for in and for in can give field name only

But

Arrays have super having both for of and for in ; for of will give values , for in will give indexes..