

Javascript Objects

Before we get into how the browser uses Javascript, we need to understand Javascript objects. By definition, a Javascript object is an unordered collection of related data in the form of key:value pairs. Consider a person as an object. A person may have a name, age, and telephone number. We can represent a person in the form of a Javascript object like this:

```
var personOne = {  
  "name": "Tom Jones",  
  "age": 32,  
  "phoneNumber": 8183228899  
};
```

The keys of the objects are the access points to the values, just like the index values are the access points in Arrays. We can access the values either using square bracket or dot notation.

```
console.log(personOne["name"]);  
// prints "Tom Jones" to the console  
// or  
console.log(personOne.age);  
// prints 32 to the console
```

The values in the objects can be any primitive or reference datatype. Above we have strings and numbers, but we could also use booleans, arrays, objects and others. We can also use a function as a value. Ever wonder what `Array.push(1)` and `Array.pop()` are? An array is a type of a Javascript object. Push and Pop are functions that belong to the Javascript Array object. Functions that belong to objects are called methods.

Let's see this in action. Put the following code in the console.

```
var myArr = [];  
console.log(typeof myArr);
```

Your result should look like this. `typeof` tells us the data type of whatever we pass in after it.

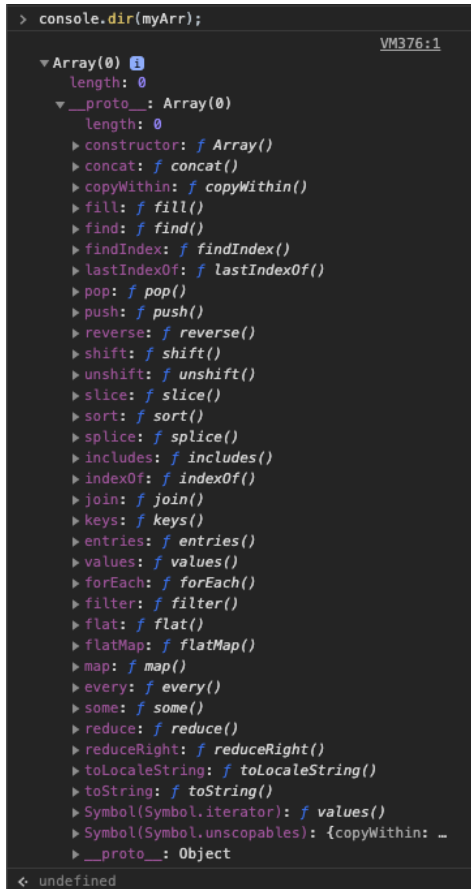
```
> var myArr = [];  
← undefined  
> console.log(typeof myArr);  
object VM202:1  
← undefined
```


Next, we'll use `console.dir()` to look into the object itself. `console.dir()` will provide us an interactive list of properties of an object.

```
console.dir(myArr);
```

Now, let's take a look at what we have. By clicking on the triangles next to the names, we can unpack the properties of the object. You will notice pop and push in the list. We can access any of these properties using either square bracket or dot notation, although dot notation is more commonly used.

```
myArr.push(8);  
console.log(myArr);  
// => [8]
```



```
> console.dir(myArr);  
  
VM376:1  
▼ Array(0)   
  length: 0  
  __proto__: Array(0)  
    length: 0  
    ▶ constructor: f Array()  
    ▶ concat: f concat()  
    ▶ copyWithin: f copyWithin()  
    ▶ fill: f fill()  
    ▶ find: f find()  
    ▶ findIndex: f findIndex()  
    ▶ lastIndexOf: f lastIndexOf()  
    ▶ pop: f pop()  
    ▶ push: f push()  
    ▶ reverse: f reverse()  
    ▶ shift: f shift()  
    ▶ unshift: f unshift()  
    ▶ slice: f slice()  
    ▶ sort: f sort()  
    ▶ splice: f splice()  
    ▶ includes: f includes()  
    ▶ indexOf: f indexOf()  
    ▶ join: f join()  
    ▶ keys: f keys()  
    ▶ entries: f entries()  
    ▶ values: f values()  
    ▶ forEach: f forEach()  
    ▶ filter: f filter()  
    ▶ flat: f flat()  
    ▶ flatMap: f flatMap()  
    ▶ map: f map()  
    ▶ every: f every()  
    ▶ some: f some()  
    ▶ reduce: f reduce()  
    ▶ reduceRight: f reduceRight()  
    ▶ toLocaleString: f toLocaleString()  
    ▶ toString: f toString()  
    ▶ Symbol(Symbol.iterator): f values()  
    ▶ Symbol(Symbol.unscopables): {copyWithin: _  
    ▶ __proto__: Object  
← undefined
```

The next module will go into how browsers transform our HTML into a Javascript object that we can access and manipulate the pages content.