Built-in objects

Objects have properties and methods. JavaScript provides many standard **built-in** objects. In addition to that it also provides an option to create **user defined** objects.

Some of the commonly used built-in objects are:

* Array
* Date
* String

Let us look at each of them in detail.

An array is used to store multiple values in a single variable. We can easily access the values stored in an array using the index position of the data stored in the array.

**Array** built-in object is used to create arrays.

1. var arr1 = [element0, element1, ..., elementN];

Let's understand it with an example:

1. placesToVisit = ["Paris", "New York", "Switzerland"];
2. console.log(placesToVisit[0]);
3. *// Paris*
4. console.log(placesToVisit[2]);
5. *// Switzerland*

**NOTE: Indexing in array start with 0.**

Apart from constructing an array as shown above, we can also destructure an existing array. Just by assigning multiple variables to the array, we can access individual items. For example:

1. numArr=[100,200,300];
2. var [a,b,c]=numArr;
3. *// the numArr is now destructured and individual values are stored in the individual variables.*
4. console.log(a);
5. console.log(b);
6. console.log(c);

 Arrays comes with many built-in functions which makes working with array very easy in javascript.

**1. push()**

push() is used to insert a new element at the end of the array.

1. places = ["Paris", "New York"];
2. places.push("Switzerland");
3. console.log(places);
4. *// ["Paris", "New York", "Switzerland"]*

**2. pop()**

pop() is used to remove last element of array.

1. places = ["Paris", "New York", "Switzerland"];
2. places.pop();
3. console.log(places);
4. *// ["Paris", "New York"]*

**3. indexOf()**

indexOf() is used to find index of given elements.

1. places = ["Paris", "New York", "Switzerland"];
2. console.log(places.indexOf("New York"));
3. *// 1*

**Note**: indexOf() will return -1 if the value is not present

**4. splice(pos, n)**

splice(pos, n) will remove n elements from pos index position.

1. places = ["Paris", "New York", "Switzerland"];
2. places.splice(1, 2); *// This will remove 2 elements from index 1*
3. console.log(places);
4. *// ["Paris"]*

**5. forEach()**

forEach() is used to interate over an array. This is a Higher Order Function. It will take a function and invoke that function on all elements of array.

1. places = ["Paris", "New York", "Switzerland"];
2. places.forEach(function(place) {
3. console.log(place);
4. }
5. );
6. *// Paris*
7. *// New York*
8. *// Switzerland*

**Note:** [Learn more about Arrays](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array)

Let's look at some more built-in array functions.

**1. map()**

The ***map()*** array function generates a new array. It iterates over each element in the array, just like forEach. It invokes a function on each element, just like forEach.

But the difference, is forEach just invokes function on each item in the array. It does not create a new array. Hence, **map()**creates a new array based on what the function does.

1. placesToVisit= ["Paris", "New York", "Switzerland"];
2. function display\_uppercase(place) {
3. return place.toUpperCase();
4. }
5. placesUpparCase = placesToVisit.map(display\_uppercase);
6. console.log(placesUpparCase);

**2. filter()**

What if we want to get all words whose length is greater than 5?

For this we can use **filter()**. filter accepts a function. It iterates over each element and creates a sub array if the function returns true.

1. placesToVisit= ["Paris", "New York", "Switzerland"];
2. function filterPlaces(val) {
3. if (val.length > 5) {
4. return true;
5. }
6. }
7. filteredPlaces = placesToVisit.filter(filterPlaces);
8. console.log(filteredPlaces );

**3. find()**

Array has a method called **find()**. It returns the first element in the array which satisfies a given condition. It takes a callback. It executes the callback for each element in the array. If the callback returns true, then find returns the element for which the callback returned true and stops further iteration. If it was false for all elements, it returns undefined.

1. placesToVisit= ["Paris", "New York", "Switzerland"];
2. function findPlaces(val) {
3. if (val.length > 5) {
4. return true;
5. }
6. }
7. foundPlaces = placesToVisit.find(findPlaces);
8. console.log(foundPlaces);

**1. Arrow in *forEach()***

Consider the ***forEach()*** function of an array. This function takes another function as parameter and invokes the function for every item in the array.

1. placesToVisit= ["Paris", "New York", "Switzerland"];
2. placesToVisit.forEach(place => console.log("Trip to " + place));
3. *// Trip to Paris*
4. *// Trip to New York*
5. *// Trip to Switzerland*

**2. Arrow in *map()***

We know that an array object has a .*map()* function that creates a new array based on what the passed callback function does.

This *map()* can also be written using arrow function.

1. placesToVisit= ["Paris", "New York", "Switzerland"];
2. placesUpperCase = placesToVisit.map(place => place.toUpperCase());
3. console.log(placesUpperCase);
4. *// ["PARIS", "NEW YORK", "SWITZERLAND"]*

**3. Arrow in *filter()***

We know that an array object has a .*filter()* function that returns a filtered sub array based on what the passed callback function does.

This *filter()* can also be written using arrow function.

1. placesToVisit = ["Paris", "New York", "Switzerland"];
2. filteredPlace = placesToVisit.filter(place => place.length > 5);
3. console.log(filteredPlace);
4. *// [ 'New York', 'Switzerland' ]*

**4. Arrow in *find()***

We know that an array object has a .*find()* function that returns the first element in the array based on what is passed as callback function.

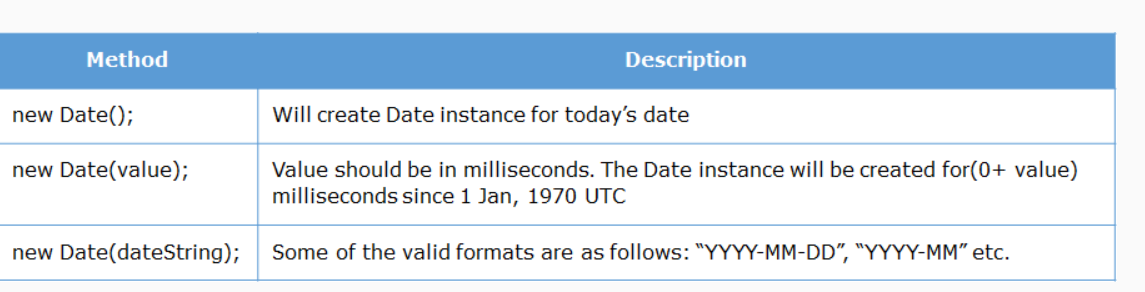
This *find()* can also be written using arrow function.

1. placesToVisit = ["Paris", "New York", "Switzerland"];
2. findPlace = placesToVisit.find(place => place.length > 5);
3. console.log(findPlace);
4. *// "New York"*

# Date Object

**Date**is a built-in object which is used to create a **Date**instance. **Date**instance can be created using **new Date();**

Some methods of object are:



**Note**: Month in Date is 0 based.

* JS has many useful built-in objects
* Some of the key array methods are : foreach, find, map, filter, push, pop, splice
* Arrow functions ease the use of array functions
* new Date() will create a date object with current date