

Callback Functions

A Callback function is function that is performed after another function has completed its execution

It is typically supplied as an input to other function.

Callbacks are critical to understand, as they are used in array methods (such as map(), filter(), and so on), setTimeout, event listeners (such as click, scroll)

```
Eg> Function orderPizza(type, name, callback) {
    console.log('Pizza ordered.');
    console.log('Pizza is on preparation');
    setTimeout(function() {
        let msg = `Your ${type} ${name} Pizza is ready`;
        callback(msg);
    }, 3000);
}
```

Date: ___ / ___ / ___

Now Invocation of Order Pizza

```
orderPizza ('reg', 'cheese', function(message)  
{  
    console.log(message);  
});
```

Imp points to Note

→ Javascript fn can accept other fn as arg.
→ passing fn as argument is powerful
programming concept that can be
used to notify caller that something
happened.

It is also known as callback function.

→ Nesting too many callback fn
is not a great idea and it
creates Callback hell.

JavaScript Map

The `Array.Map()` allows you to iterate over array using loop.

This method allows you to iterate and modify its elements using a callback function.

The callback function will then be executed on each of array's element.

For. eg `let arr = [2, 3, 4, 5, 6];`

Now Imagine you have to multiply each element of array by 3

You can use for loop also like this

```
let arr = [2, 3, 4, 5, 6];
for (let i = 0; i < arr.length; i++) {
  arr[i] = arr[i] * 3;
}
console.log(arr);
```

|| [6, 9, 12, 15, 18]

By using map it will look like this:

```
let arr = [3, 4, 5, 6];
```

```
let modifiedArr = arr.map(function(el) {  
    return el * 3;  
});  
console.log(modifiedArr);  
// [6, 9, 12, 15, 18]
```

How to Use Map over ARRAY of OBJECT

```
let users = [  
    { firstName: 'Deepa', lastName: 'Chaurasia' },  
    { firstName: 'Devash', lastName: 'Chaurasia' },  
    { firstName: 'Jyoti', lastName: 'Chaurasia' }  
];
```

You can iterate as follow

```
let userFullnames = users.map(function(el) {  
    return ` ${el.firstName} ${el.lastName}`;  
});  
console.log(userFullnames);
```

```
// ['Deepa Chaurasia', 'Devash Chaurasia', 'Jyoti Chaurasia']
```

The Complete map() method syntax

The syntax of map() as follows

```
arr.map(function(element, index, array){  
    this);
```

The callback function() is called on each array, element, and the map() method always passes the current element, the index of current element and whole array object to it.

The this argument will be used inside callback function.

By default it's value is undefined

Eg— let arr = [2, 3, 5, 7]

```
arr.map(function(element, index, array){  
    console.log(this) // 80  
    3, 80);
```

Here you can see this value is 80 which is default value.

Reduce Method In JavaScript

Use it When : you have array of numbers, you want to add them all.

For eg - const nos = [29, 40, 30] ;
 const sum = nos.reduce ((total, amount)
 ⇒ total + amount) ;
 sum // 99

Filter() and Find() in JS

Filter() provides new array depending on certain criteria.

Unlike map(), it can alter size of new array, whereas find() return just a single instance.

For eg → let users = [
 { firstName: 'Ram', age: 14 },
 { firstName: 'Shyam', age: 17 },
 { firstName: 'Jacob', age: 25 }]
];

You could choose to sort these data by age groups, such as young (1-15) adult (15-50)

Like this:

```
const youngPeople = users.filter((person) => {
    return person.age <= 15;
});
```

```
const adult = users.filter((person) => {
    return person.age >= 50;
});
```

```
console.log(youngPeople);
console.log(adult);
```

And The Example of Find goes like this

```
const Ram = users.find((person) =>
    person.firstName === 'Ram');
```

```
console.log(Ram);
```

Unique Value - Set() in JS

```
let animals = [
```

{ ~~common~~

name: 'Lion',

category: 'wild'

},

{

name: 'dog'

category: 'pet'

},

{

name: 'cat'

category: 'pet'

},

If we loop through map, we will get
repeated value

But we don't want repeated value here

So we will use Unique value - Set()

For eg - let category = [...new Set(
animals.map(animal) ⇒

animal.category)];

console.log(category); // [wild, pet]