

## Twitter Thread by Pratham



**Pratham**

@Prathkum



### Some of my handmade JavaScript notes and cheat sheets that can help you ■■■

#### 1. JavaScript Function Anatomy

Function

Within curly braces you can write function code. You can write bunch of code as needed.

Return statement is used for returning some value from code. This is optional. After return statement you can't write any line of code.

```
function setName (name) {
  var str = "My name is" + name;
  return str;
}
setName("Pratham");
```

The word followed by function keyword defines the name of the function.

JavaScript is a camelcase language i.e, if function name is a combination of word then the first alphabet should be small and first alphabet of second word should be capital for ex: setName  
S small      N capital

We can pass parameters in function. This is optional. parameters are some information which is used in function code.

"function keyword defined that this is a function that binds a bunch of code"

Calling the func. when we call the f<sup>n</sup> the code inside function will be executed. At the time of calling the f<sup>n</sup> you have pass param if any.

#### 2. JavaScript Arrow Function

# Arrow function.

In arrow function you don't need to write the function keyword

```
(a) => {  
  return a+100;  
}
```

parameter

- place an arrow sign between arguments and function body (`{...}`)

You don't even need to write the function name as well.

```
(a) => a+100;
```

you can remove the braces and return keyword. It will work perfectly fine.

If in case there is a single parameter, then you can remove the argument parantheses.

```
a => a+100;
```

- \* If function body is there then return statement and parantheses (`{...}`) are required.

## 3. JavaScript Array Methods

```
[1,2,3].map(x => x * 2) // [2,4,6]
[1,2,3].filter(x => x < 2) // [1]
[1,2,3].indexOf(2) // 1
[1,2,3].reduce((x,y) => x * y) // 6
[1,2,3].reverse() // [3,2,1]
```

```
['P','R'].concat(['a']) // ['P','R','a']
['P','R','a'].slice(1) // ['R','a']
['P','A','T'].splice(1,0,'R') // ['P','R','A','T']
['P','R'].join('-') // 'P-R'
['P','R','A','T','H','A','M'].lastIndexOf('A') // 5
```

## Array Methods

```
[1,2,3,4].fill(0,2,4) // [1,2,0,0]
[1,2,3,4].find(x => x > 3) // 4
[1,2,3].findIndex(x => x > 1) // 1
[1,2,3].includes(2) // true
[1,2,3].some(x => x % 2 === 0) // true
```

```
[1,2,3].forEach(x => console.log(x))
// 1,2,3
[1,2].push(3) // [1,2,3]
[1,2,3].pop() // [1,2]
[1,2,3].shift() // [2,3]
[1,2,3].unshift(0) // [0,1,2,3]
[1,2,3].every(x => x < 5) // true
```

### 4. JavaScript Date Methods

## JAVASCRIPT DATE METHODS

```
const date = new Date();
```

```
date.getDate(); // 15  
date.getDay(); // 1 monday  
date.getFullYear(); // 2021  
date.getHours(); // 14  
date.getMinutes(); // 40  
date.getMonth(); // 1  
date.getSeconds(); // 13
```

```
date.toTimeString() // 14:49:53  
GMT+0530  
date.toDateString() // Mon Feb 15 2021  
date.toString() // Mon Feb 15 2021  
14:49:53 GMT+0530  
date.toLocaleDateString() // "2/15/2021".
```

```
Date.parse(date); // 1613380793000
```

→ parse a string *date* and it will return the no. of milliseconds since Jan 1, 1970

```
date.setDate(24);  
date.setFullYear(2022);  
date.setHours(12);  
date.setMinutes(45);  
date.setMonths(3);  
date.setSeconds(42);  
date.setTime(14);
```

```
Date.now(); // 1613381392120
```

→ returns the <sup>no.</sup> of milliseconds elapsed since Jan 1 1970 00:00:00 UTC

## 5. JavaScript Array Methods

## JAVASCRIPT ARRAY METHODS

```
[1,2,3].concat([4]) // [1,2,3,4]
[1,2,3].copyWithin(0,2,3) // [3,2,1]
[1,2,3].every(x => x < 3) // false
[1,2,3].fill(0,1,3) // [1,0,0]
[1,2,3].filter(x => x >= 2) // [2,3]
[1,2,3].find(x => x > 1) // 2
```

```
[1,2,3].map(x => x * 2)
// [2,4,6]
```

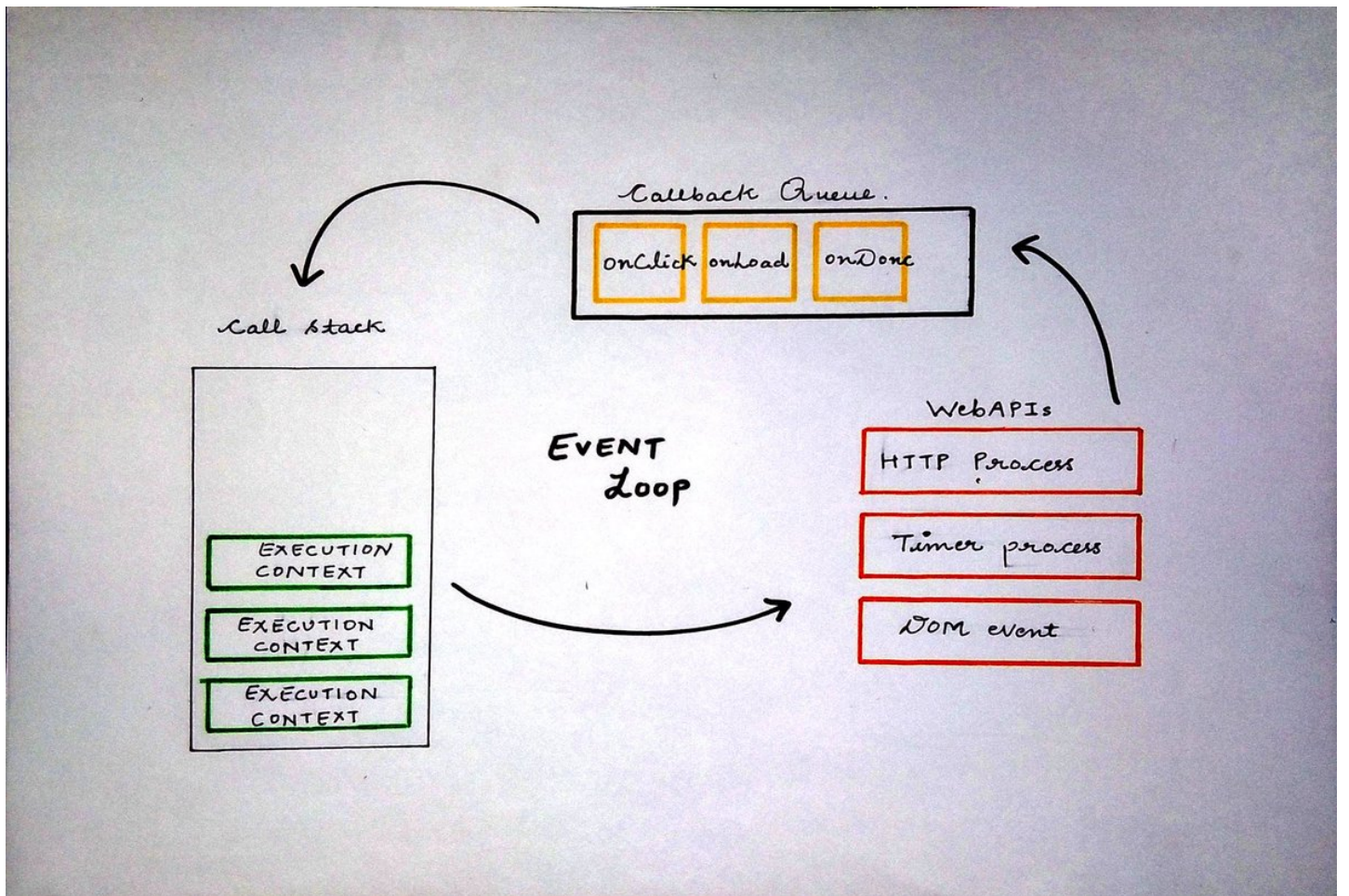
```
[1,2,3].forEach(x => console.log(x))
// 1 2 3
```

```
[1,2,3].push(4) // 4 return
                // length of arr
                // [1,2,3,4]
[1,2,3].shift() // 1
                // return shifted
                // element-
                // [2,3]
[1,2,3,4].slice(1,3) // [2,3]
```

```
[1,2,3].some(x => x > 2) // true.
[1,2,3].indexOf(2) // 1
[1,2,3].includes(2) // true.
[1,2,3].indexOf(2) // 1
[1,2,3].join("-") // "1-2-3"
[1,2,1].lastIndexOf(1) // 2
[1,2,3].pop(3) // 3 return popped element
                // [1,2]
```

```
[1,2,3].reduce((x,y) => x+y) // 6
                             // = 1+2+3
```

## 6. Event Loop



## 7. JavaScript DOM Methods

# DOM

## Getting elements.

```
document.getElementById("id");  
// return the element with  
id = "id".
```

```
document.getElementsByClassName("class");  
// returns the collections of element  
with class = "class"
```

```
document.getElementsByTagName("h1");  
// return the collection of h1 element
```

```
document.querySelector(".class");  
// return first element based on  
selector.
```

```
document.querySelectorAll(".class");  
// return collection of element  
with class = "class";
```

Returns all elements with specified class name or tag name as a NodeList (collection of nodes).

In order to access specific node, you can use indexing. for ex,

```
document.getElementsByClassName("class")[0]
```

## 8. DOM addEventListener Method

## DOM addEventListener() method

\* addEventListener method attaches an event to the specified element.

Calling event listener method on button

Getting button element from DOM

```
const btn = document.getElementById("btn")[0];  
btn.addEventListener("click", function() {  
    alert("Clicked");  
});
```

Passing "click" event as an argument. This event occurs when user clicks on the element.

This function will run when the event occurs. In this example, when button clicked, an alert will be displayed.