

JS WORKSHOP

Just another workshop on JS or is it ? 😊

What is Javascript

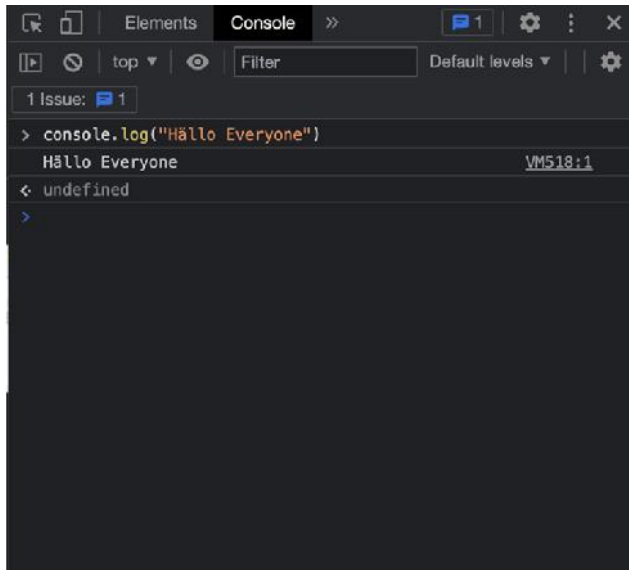
JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting

1. Event-driven
2. Functional

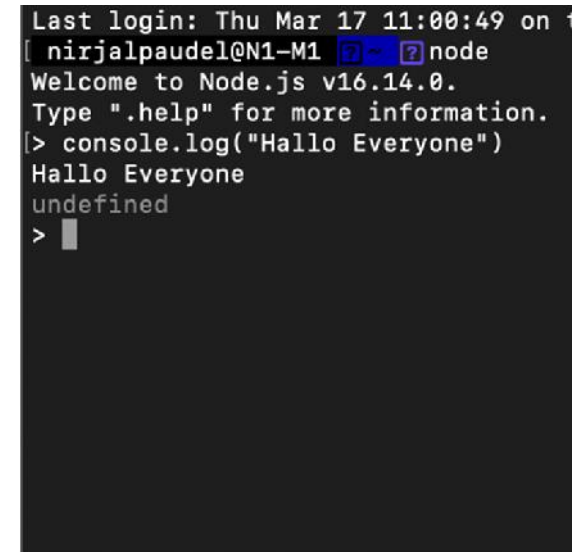


Quickest way to run JS?

Open inspect element in browser and go to console section, you will see REPL or in a terminal using node.



A screenshot of a web browser's developer console. The 'Console' tab is selected, showing a single log entry. The entry is a message 'Häillo Everyone' in orange text, with the source 'VM518:1' on the right. Above the message, the command '> console.log("Häillo Everyone")' is visible. Below the message, the return value '< undefined' is shown. The console interface includes a filter bar at the top and a '1 Issue' indicator.



A screenshot of a terminal window. The prompt is 'nirjalpaudel@N1-M1'. The user has entered 'node', and the terminal displays 'Welcome to Node.js v16.14.0. Type ".help" for more information.' The user then enters 'console.log("Häillo Everyone")', and the terminal outputs 'Häillo Everyone' followed by 'undefined' on the next line. The prompt '>' is visible at the bottom.

Where is JS used ?

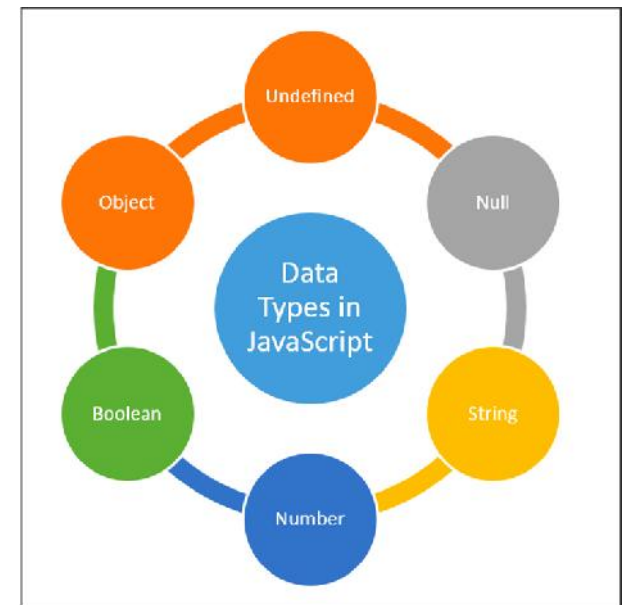
1. Web application with vanilla JS
2. Web server with NodeJS
3. Android development with React Native
4. Computer app development with electron, etc

[What is JavaScript Used For?: A Complete Guide | Career Karma](#)

Data types

Different types of data types in JS are listed below

1. Number → floating point, integer → 1, 2.33, etc
2. String → “Hola”, “Hello”, “LOL”, “YOLO”, etc
3. Boolean → true or false
4. Symbol → Symbol(“Symbol here”)
5. Objects → Array and Objects
6. Undefined
7. Null



Objects

Objects are key value pairs. It takes a key and a value. Every thing is wrapped around { and }. It can represent real world things.

```
> const newhuman = {  
  "key": "value",  
  "kaccha": "badam",  
  "lucky_number": 7,  
  "boy": true  
}
```

```
const data_types = {  
  "number": 3.33,  
  "string": "21 guns",  
  "boolean": true,  
  "array": [1,2,4,5],  
  "object": { hello:"HI" }  
}
```

Array

An array is a special variable, which can hold more than one value. Instead of saving values like

NOTE:

Unlike other languages, array can
grow to any length

```
> let car1 = "Saab";  
let car2 = "Volvo";  
let car3 = "BMW";  
  
// We can do this for better use  
let cars = [ "BMW", "Volvo", "BMW" ];  
: undefined  
> |
```

Null

JavaScript uses the null value to represent the intentional absence of any object value.

Undefined

Undefined means a variable has been declared but has not yet been assigned a value.

<https://flexiple.com/undefined-vs-null-javascript/>

Function

A function is a block of code designed to do a specific task. Inspired by digestion system.

May have a return statement that return a value.

```
function addTwoNumber(a,b){  
    let sum = a+b;  
    return sum;  
}
```

```
undefined
```

```
addTwoNumber(4,5)
```

```
9
```

Fun fact - Template string

In js, we can declare string using template string methods. This is just like using email templates.

```
> a= `Hello ${name}`  
< 'Hello '  
  
> name = "Nirjal Paudel"  
< 'Nirjal Paudel'  
  
> a= `Hello ${name}`  
< 'Hello Nirjal Paudel'  
  
>
```

= VS == VS ===

Apart from spelling, their differences are

1. = is used to assign a value to a variable
2. == is used to compare between values or variables but

Data type based comparison is not

1. === is used to strictly compare between values or variables.

```
a= 12;  
12  
a=="12"  
true  
a==="12"  
false
```

Let vs const vs var

| var | let | const |
|--|---|--|
| The scope of a <i>var</i> variable is functional scope. | The scope of a <i>let</i> variable is block scope. | The scope of a <i>const</i> variable is block scope. |
| It can be updated and re-declared into the scope. | It can be updated but cannot be re-declared into the scope. | It cannot be updated or re-declared into the scope. |
| It can be declared without initialization. | It can be declared without initialization. | It cannot be declared without initialization. |
| It can be accessed without initialization as its default value is "undefined". | It cannot be accessed without initialization, as it returns an error. | It cannot be accessed without initialization, as it cannot be declared without initialization. |

Hoisting

JavaScript Hoisting refers to the process whereby the interpreter appears to move the declaration of functions, variables or classes to the top of their scope, prior to execution of the code. Hoisting allows functions to be safely used in code before they are declared.

What we see

```
var gender = "F";
sayHello(gender);

function sayHello(gender) {
  if (gender === "M") {
    console.log("Wonderwall - Oasis");
  } else {
    console.log("Don't took back in anger - Oasis");
  }
}
```

What interpreter interprets as

```
// moved to top
function sayHello(gender) {
  if (gender === "M") {
    console.log("Wonderwall - Oasis");
  } else {
    console.log("Don't took back in anger - Oasis");
  }
}
var gender = "F";

sayHello(gender);
```

Arrow Function

Discussed earlier, a function what a function is

But arrow function is a compact alternative to the traditional functions. But it does have things like

Differences & Limitations:

- Does not have its own bindings to this or super, and should not be used as methods.
- Does not have new.target keyword.
- Not suitable for call, apply and bind methods, which generally rely on establishing a scope.
- Can not be used as constructors.
- Can not use yield, within its body.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

<https://betterprogramming.pub/when-to-use-bind-call-and-apply-in-javascript-1ae9d7fa66d5>

Custom Errors

Almost every things you see in red is error. Error shows that there is something wrong

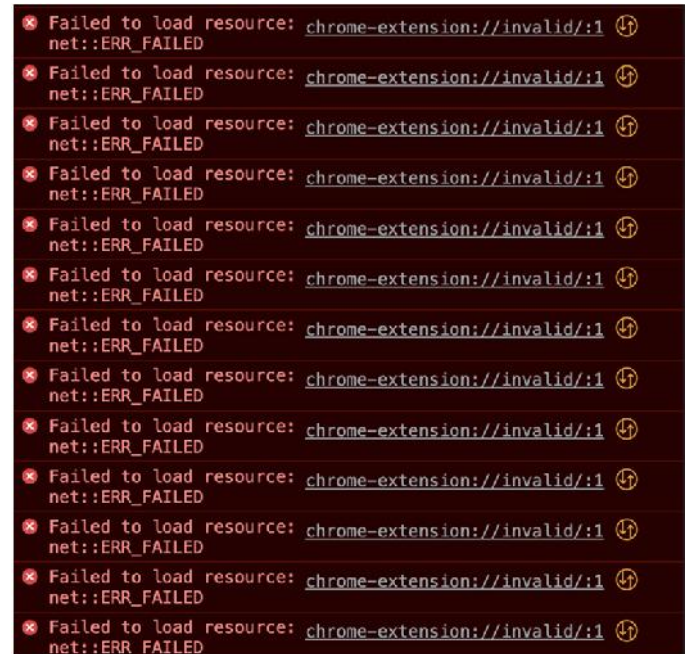
In your code.

We can write custom error in JS, it is done using throw statement

Example

```
throw 3;
```

```
throw 55;
```



Document Object Model

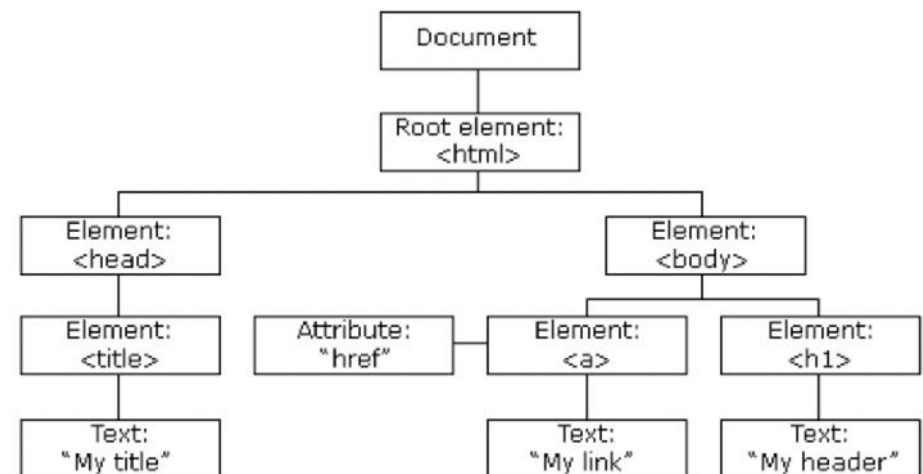
When a web page is loaded, the browser creates a Document Object Model of the page. Each element in DOM represents what we see on the screen. DOM elements have various properties for style, class-list, id, etc

The HTML DOM model is constructed as a tree of Objects:

Using JavaScript, we can

- change all the HTML elements in the page
- change all the HTML attributes in the page
- change all the CSS styles in the page
- remove existing HTML elements and attributes
- add new HTML elements and attributes
- react to all existing HTML events in the page
- create new HTML events in the page

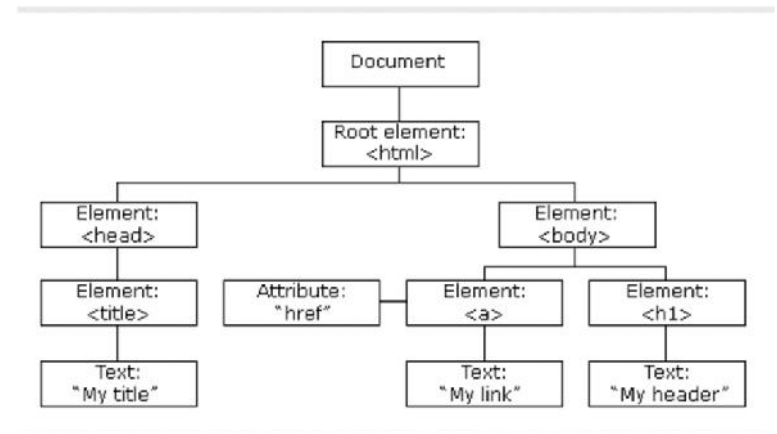
https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model



DOM manipulation

- Change title of a website
- Add a div to an element in website
- Change style of an element
- Change content of an element using innerHTML
- Change content of an element using innerText
- Add or remove class to an element
- Change id to an element

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents



DOM events and listeners

Various events take place in the DOM. Events like click, double click, drag, blur, change, etc. As a JS developer, you must learn how to react to these events.

All the DOM events: [HTML DOM Event Object](#)

Listeners are attached with a function that gets trigger every time the specified events takes place. Lets see an example.

Who is in for day 2



BUT



End of day 1

Cheat Sheet:

[Javascript Cheat Sheet](#)

Docs:

[JavaScript and HTML DOM Reference](#)

[JavaScript | MDN](#)

[JavaScript - GeeksforGeeks](#)

Videos:

[JavaScript Programming - Full Course](#)

JS Array and array methods

Talked earlier, an array is a collection of dynamic elements. An array has many methods like

Suppose an array of numbers as

But when talking about array methods in Javascript, we generally talk about, methods like

find(), findIndex(), filter(), map(), reduce()

```
> const numbers = [3,2,0,1,5,6]
< undefined
> numbers.length // to get length of array
< 6
> numbers.includes(0)
< true
> numbers.includes(-11)
< false
> |
```

find()

find() : Find is an array method which take a function as an argument (**also known as callbacks**), returns one of two things

1. The array element that matches your condition.
2. **undefined** when the condition is not matched by any of the element.

[👉, 🏆, 🙌, 👁️, 🏀, 😡].find(emoji=> emoji.isAngry()) → 😡

Learn more about find() method here:

[Array.prototype.find\(\) - JavaScript | MDN](#)

```
[2,1,3,5,4,7].find((num)=>num>3)  
5
```

findIndex()

`findIndex()` : Find is an array method which take a function as an argument (**also known as callbacks**), returns one of two things

1. The index of first array element that matches your condition.
2. **-1** when the condition is not matched by any of the element.

`[🏀,👉,🎓,🙌,👀,🏈].find(emoji=> emoji.isKickAble()) → 0`

Learn more about `findIndex()` method here:

[Array.prototype.findIndex\(\) - JavaScript | MDN](#)

```
[1,2,3,4,5,1,1].findIndex((num)=>num===1)
// it read the 1st content which has 0 index
0
```

map()

map() : Map is an array method which take a function as an argument (**also known as callbacks**), and returns a new array with transformation applied.

`[1,2,3,4,5].map(num=>num*2) → [2,4,6,8,10]`

Learn more about map() method here:

[Array.prototype.map\(\) - JavaScript | MDN](#)

```
const a = [2,4,6,8,10]
undefined
a.map(num=>num/2)
▶ (5) [1, 2, 3, 4, 5]
```


filter()

`filter()` : filter is an array method which take a function as an argument (**also known as callbacks**), and returns a new array which satisfies the given condition

`[😄, 😞, 😐, 😡].map(emoji=>emoji.isHappy())` → `[😄, 😡]`

Learn more about `filter()` method here:

[Array.prototype.filter\(\) - JavaScript | MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter)

```
> a
< ▶ (5) [2, 4, 6, 8, 10]
> a.filter(num=>num>5)
< ▶ (3) [6, 8, 10]
>
```

reduce()

reduce() : reduce is an array method which take a function as an argument (**also known as callbacks**), and returns a single element that is **aggregated by the callback**. One of the feared thing in JS

Lets do a hands on

Learn more about reduce() method here:

[Array.prototype.reduce\(\) - JavaScript | MDN](#)



Lets revise

- find -> search and get 1st element that matches the condition
- findIndex() -> get index of 1st element that matches the condition
- map() -> transform every element of the array.
- filter() -> get a new array of all the elements of that array which matches the condition
- reduce() -> get a single element that is formed by using all the elements of the given message



Steven Luscher
@steveluscher

Map/filter/reduce in a tweet:

```
map([🥦, 🐮, 🍷], cook)
=> [🍿, 🍔, 🍳]
```

```
filter([🍿, 🍔, 🍳], isVegetarian)
=> [🍿, 🍳]
```

```
reduce([🍿, 🍳], eat)
=> 🤮
```

7:53 am · 10 Jun 16 · [Twitter for iPhone](#)

SYNCHRONOUS CALLS VERSUS ASYNCHRONOUS CALLS

SYNCHRONOUS CALLS

A callback in which the code execution waits for an event before continuing

Code execution waits for the event before continuing

Programmer can use synchronous callbacks when it is necessary to execute tasks in a sequence and when it does not require much time for execution

ASYNCHRONOUS CALLS

A callback that does not block the execution of the program

Do not block the program from the code execution

Programmer can use asynchronous callbacks when the tasks are not dependent on each other and when it takes time for execution

Visit www.PEDIAA.com

Synchronous Code

Asynchronous Code

YOU ARE LOOKING IN THIS MEME

INSTEAD OF THOSE ANIMATIONS

imgflip.com

Fetch API

The Fetch API provides an interface for fetching resources (including across the network). It allows programmers to get content of a url. The content can be

- JSON
- Text
- Blob

The response of fetch is not right away so, we will have to react to its response. Fetch returns a promise (like human beings doing promise). It has 3 states

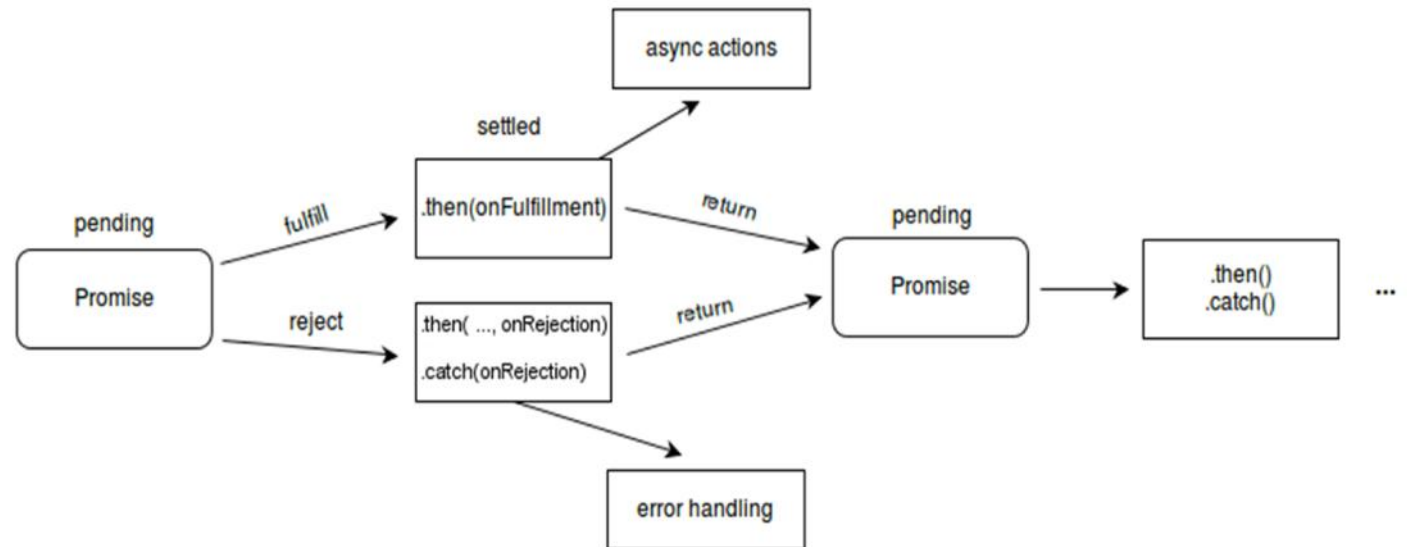
1. Pending: You make your promise - Here is a hope that it may finish
 2. Fulfilled: You keep your promise - you will react positively.
 3. Rejected: You break your promise - you will pick yourself up.
-

Fetch API Hands On

Let us try to fetch data from a URL. Lets try it on <https://jsonplaceholder.typicode.com/users>

Please install this extension

[JSON Formatter - Chrome](#)

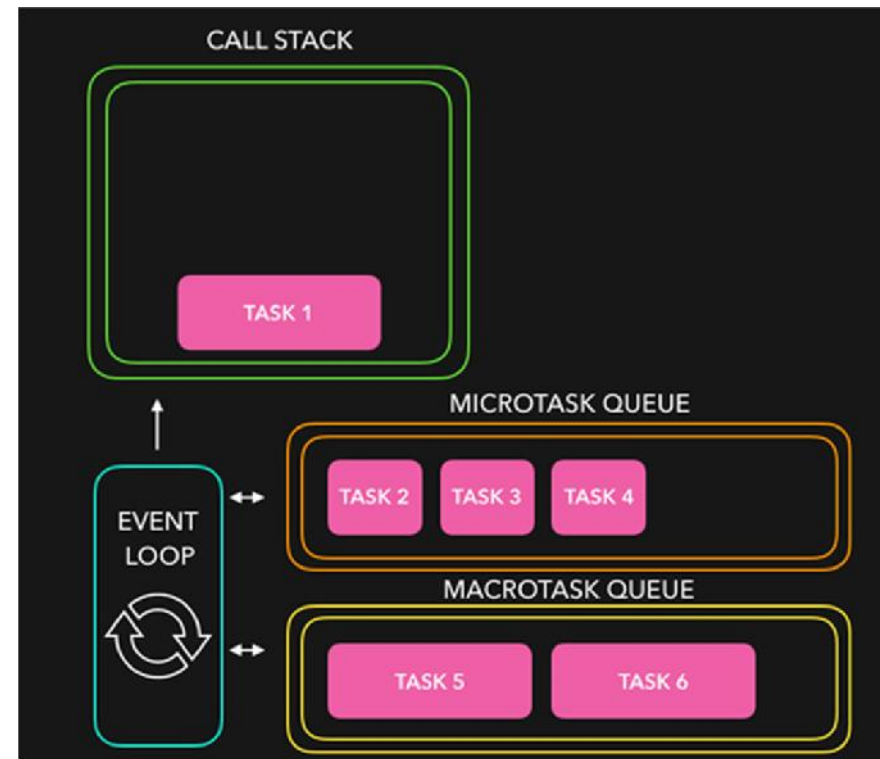


What really happens inside JS ?

There is a concept called event loop that runs Promises or any Asynchronous code in JS, this is one of the advanced topic in JS,

Go through this and learn more about event loop

[What the heck is the event loop anyway?](#)



End of day - 2

Videos:

[Asynchronous JavaScript Course \(Async/Await, Promises, Callbacks\)](#)

[Event loop | Will javascript wait?](#)

[8 Must Know JavaScript Array Methods](#)

Docs:

[JavaScript | MDN](#)

<https://www.youtube.com/c/WebDevSimplified>

So what now ?

“ If this workshop is a lion, then JS is a dinosaur. It’s huge ”

So keep learning, keep doing projects and keep learning more.

Want to talk about JS?

- LinkedIn: <https://www.linkedin.com/in/nirjalpaudel/>
- Email: me@nirjalpaudel.com
- GitHub: <https://github.com/n1rjal>
- Facebook: <https://www.facebook.com/n1rjal/>

