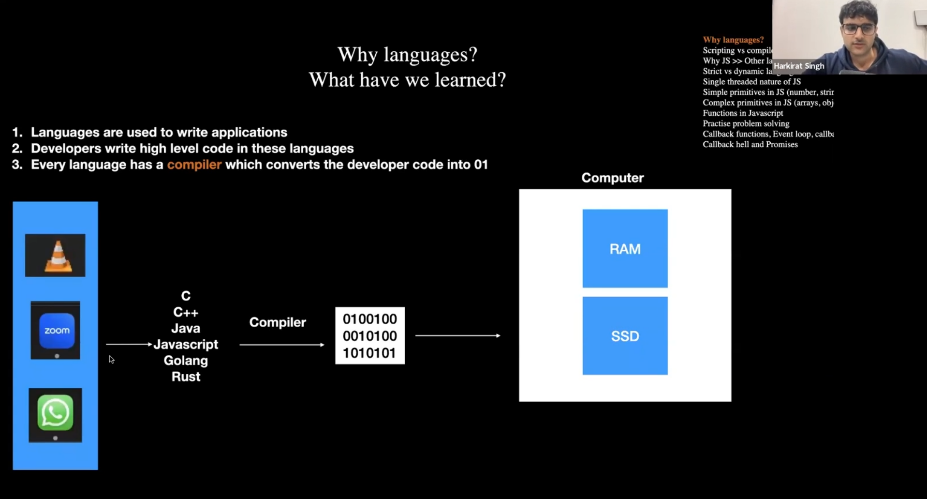
Why Languages?

Computers are dumb machines they only understand bits and bytes. But it will be too difficult for developers to write code into bits and bytes. (Zeroes and Ones). So, to solve this issue these programming languages are developed. So that developers can write the code into these languages which further converted into bits and bytes by **COMPILERS**. So yeah, purpose of Language is providing a method to make it easy for developers to understand the code and logic to build/read an application.



**Compilers**

Compilers are what convert high level languages to bits and bytes. For example, C++ language has this compiler called g++ which is famous out a few compilers.



**What is the difference between Interpreted vs Compiled Languages?**

For a machine to understand some code. It must be converted into bits and bytes. So for that there are two ways when a file consisting code can be converted into bits and bytes. First method is Compiled Languages. Like C++, Java, Rust, Golang how it works is once you have some code in C++. For the sake of example, we will say that we have written 10 lines of code in C++ and now we wanted to run that code on our machine. Now the first step will be the compiler will read through all those 10 lines and then convert them in binary code (0 & 1) and create a **TEMP** file out of that which is binary file and then will run it OR when you click on that file then your programme will be executed.

If in case of Compiled Language, the code consists an error at any place in the file. For example, here the line 9 consists some error in that case it won’t let you run the code at all will throw an error at you.

On the other hand, in Interpreted language like JavaScript, python same case a file consists 10 lines of code. The code here will also be converted into binary code but here the process will be line by line. Example if there are 10 lines the compiler will read first line then compile it & run it and then second line then third line, so on so forth. Also, if the same code file has some error in it in line 9 it will first 8 line and compile then run it. But when it will come to line 9 it will throw an error at you but up until then it will execute the programme.

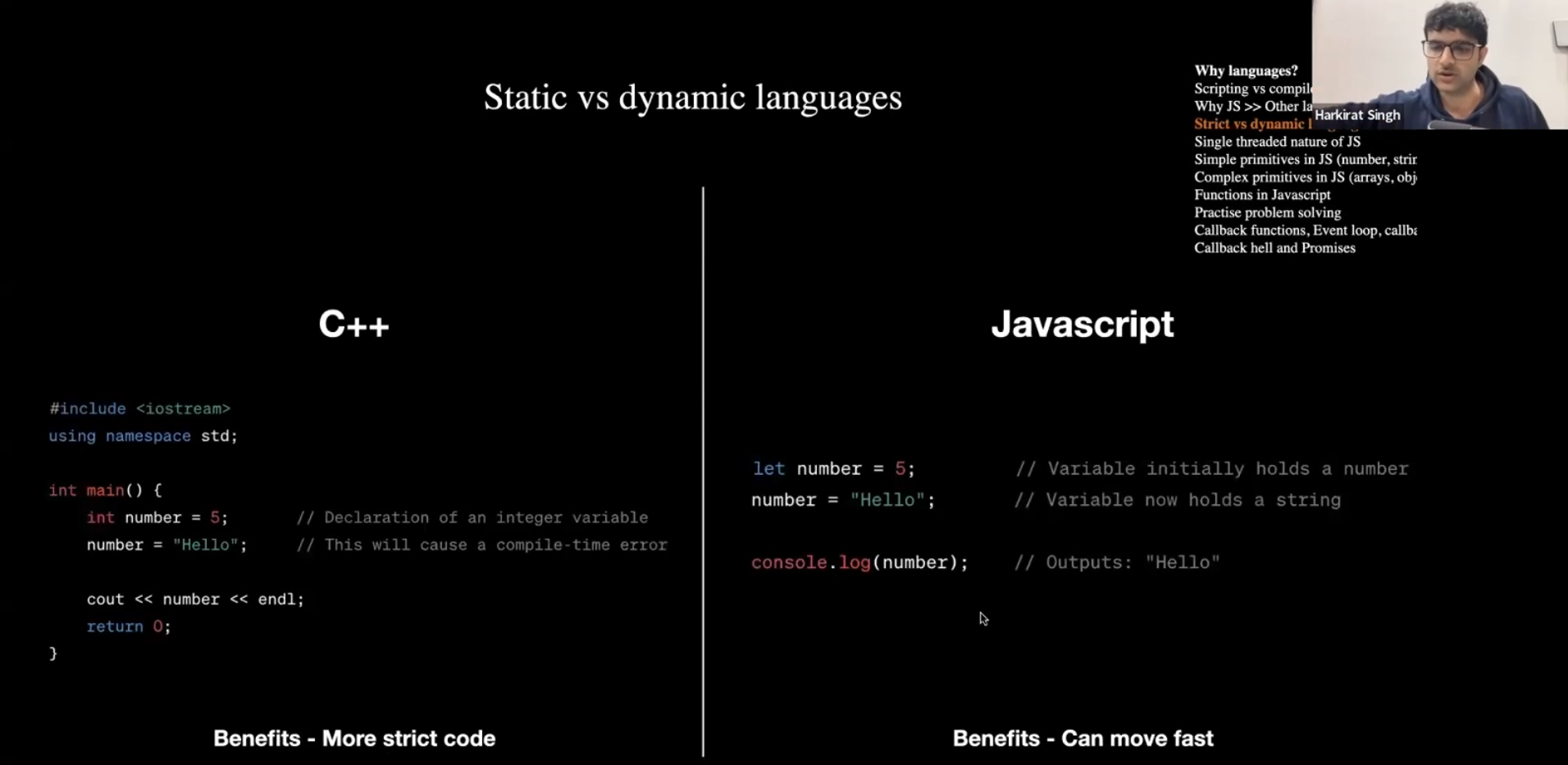
**Strict VS Dynamic Languages**

Here’s in below added image as you can see on the left side of the image that in C++. The syntax first you have the define the type of variable for example here we have first defined that (int number = 5). It says that it’s an **integer** with variable name **number** which has the value of **5**. But now later in the next line what we are trying to do is we are trying to convert that **Integer** to a  **String.**  That will throw an error because an integer can’t be a string. This type of error is called **Compile time error.**

In big code base it useful to use strict language as it upfront throws you errors and will let you know that there are problems in your code. Here you write the code slowly but more reliable code.

Now on right side. Same code written in JavaScript will run without error. Because it is a loosely type language. (or Dynamic Language). It will not throw an error here. That is also the reason we use **TypeScript** which provide the type to the variables as it helps us to write code with fewer errors.

In JavaScript it allows you change types. But as your project grow, you’ll see some errors later in your code which is called run time error and will create more difficulties because now you have bigger codebase and finding error there is a difficult task. And it’s where the typescript comes into picture and give you functionality of types which let’s you help catch the error upfront. But JavaScript also let you move fast as it is loosely written language & there are not compilation error.

****

**Single Threaded vs Multi Threaded Languages.**

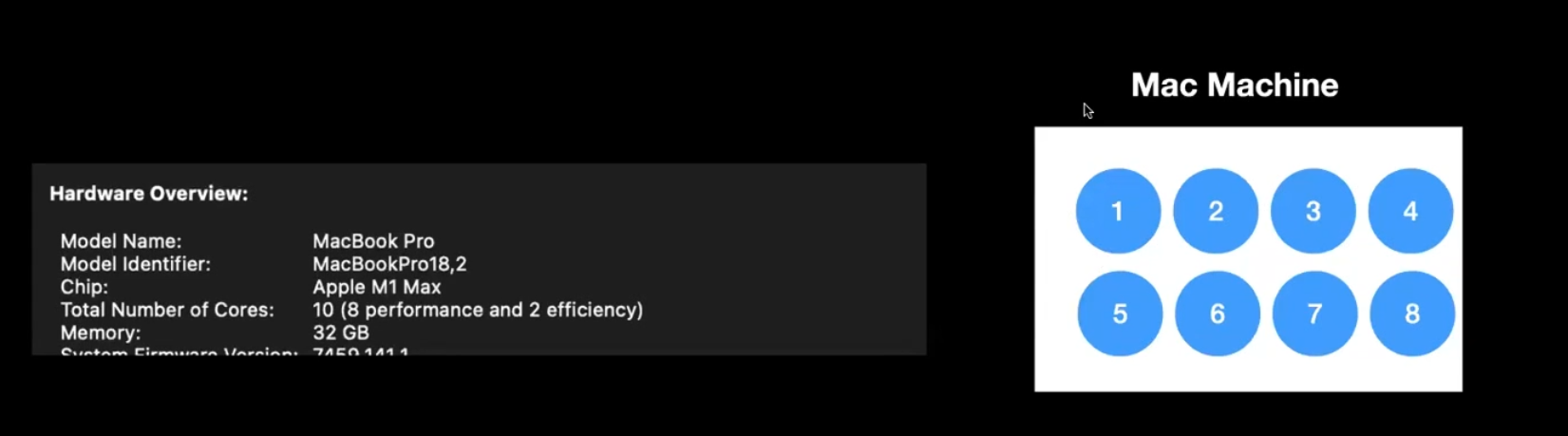
You must have heard of term called CORES while buying a computer or laptop for yourself. What those mean is one core can run one code at a single time. So when we say that JavaScript is a single Threaded language it means that it can only use one core at single time while running the code. So you can only run single code on single core at one time. Where as in other multi-threaded languages like C, C++, GO, Rust has this feature available which let you run your code on multiple cores at single time & let you reduce your run time while running the code & make the process faster. That is why it’s difficult to scale a programme on a single threaded language such as JavaScript.

For Example right now I’m using my VS code, Firefox browser, WordPress, Notepad at the same time. And if My machine has 10 cores. Does it mean that I can only run a 10 process like these at single moment. So, the answer is no because there exists something called context switch. Which basically means that everything that is running on your machine doesn’t need your CPU all the time. So, yeah you can use more process at single time than number of cores you have on your machine.

What single threaded means in JavaScript it that if your code has 10 lines it will be run line by line and will use only one core at a single time. But…(read next line)

**In JavaScript too you can also use multiple cores at same time by something called CLUSTER MODULE**

**[-1.47.52]**

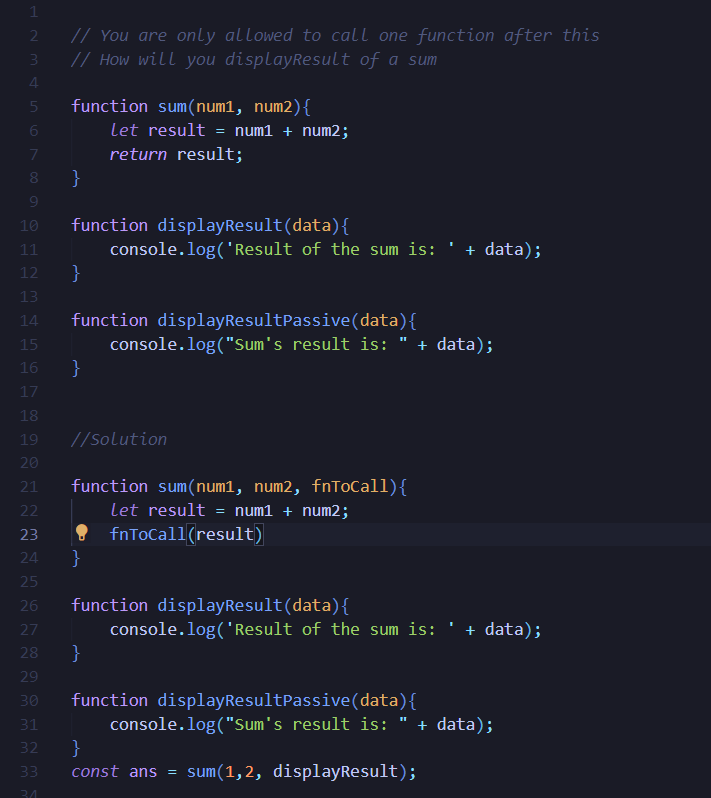


**callBack Function & Anonymous Function**

**CallBacks –** Call backs can be **defined** two ways. You can say it’s either a function where we call another function inside that function. Or we can say that callbacks are Asynchronous. Which means that it will be pushed to callBack Queue at some time in future.

CallBacks are if you actually see is just Asynchronous callback as it will be pushed to callback queue at some point of time in future. See there is not really a point of calling a function with a function unless until it’s blocking your main thread or taking quite a while to calculate the result of some function. What we usually do is when some thing is complex and time consuming, we hand it over to the webAPIs and they handle it and put them into the callback queue and run it once our call stack is empty.

Here below is a snipet of code. From line 1 to 16. You have to make changes in the code in a such way that you can call only a single function. And you can displayResult of a sum. What that means is you can



Solution: What we have done here that from line 21 to 33 is we have given a new argument to our function **Sum** here. (fnToCall = function to call). Then we call that function at line 23. So later at line 33 we are call a function of **Sum** also what we are doing is with the numbers that we want to add we are also giving it the other function **(displayResult or displayResultPassive)**  So, here we call only on function of **sum** but also we are giving another function as an argument. This what we call a **callBack hell.**

Another way of doing the same problem. First, we have defined the add and subtract function inside the calculateArithmetic function. But second time we have create two separate function to do the add & subtract and call it in the function calculateValues(). This way we are calling the function calculateValues() and inside that function it will call other function based on typeOfCalculation we provide to it.

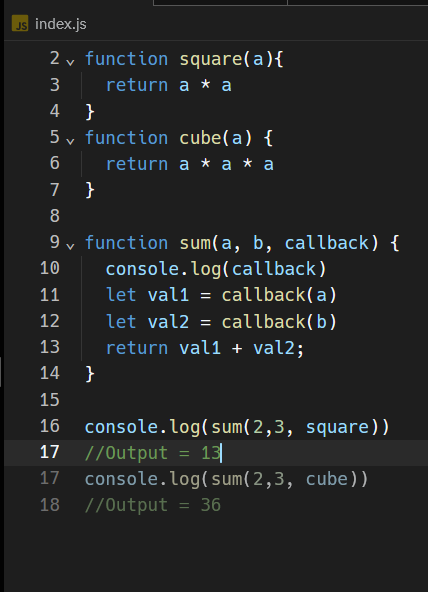
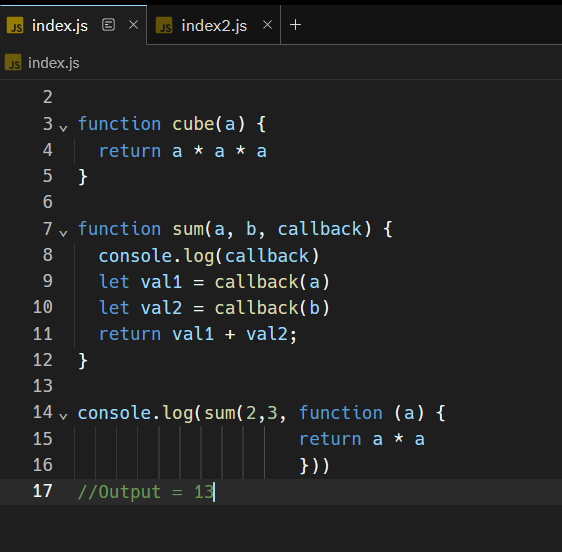
****

**Anonymous Function**

Anonymous function basically a function which doesn’t have a name. See in example below where when we use a callback function. We have to make sure the function we are calling should be already defined somewhere in code. But in case of Anonymous function, you don’t have to write the function before you call it in another function. You can just provide the whole function as an argument to another function. See the picture below you’ll have a better idea what does that mean.

In case of Anonymous function instead of calling a square function, I gave sum function a argument which itself is a function. So, here it became an anonymous function. If you’ll try to console.log the callback here it will also return you an output named as anonymous function.

**Callback function** **Anonymous Function**

**Functions** –

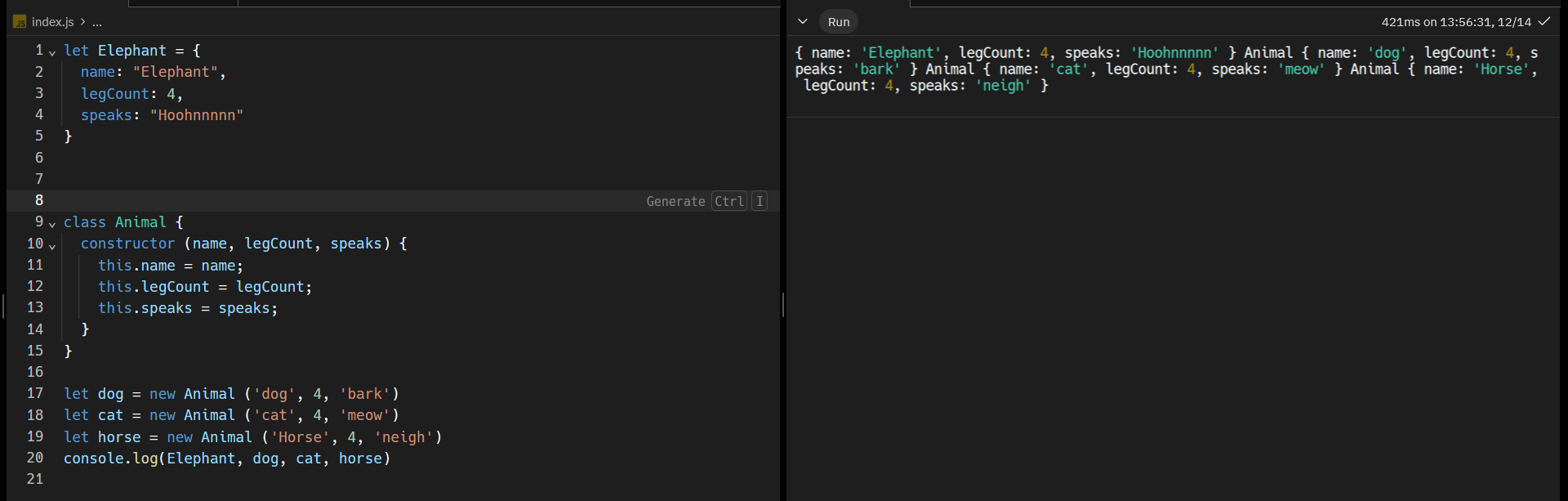
A function in JS is a set of statement that performs a task or calculates a value. It should take some input and return an output where there is some obvious relationship between the input and the output.

It basically let you write you code which you can use again and again without repeating the logic. Well you already know what functions are so not going to write it down again and again.

**Classes**

A class is **a type of object template**. The class statement initiates a JavaScript class. Properties and methods are assigned in the constructor() method. The constructor() method is called each time a class object is initialized.

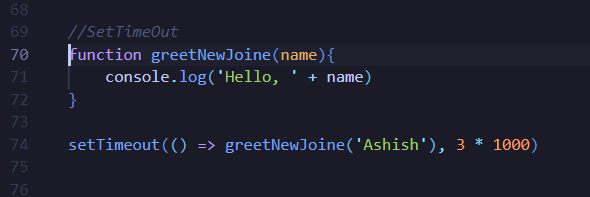
Here is below example you can see first we have tried to create an object named Elephant. But one way is to create such object is to repeat the logic and other way is uses classes where you just define the structure of such element and then you can just give it input for those properties in that structure and it will create new objects for you. That way you won’t have to repeat your code.



setTimeOut()

What setTimeOut() does it takes two argument as an input. First it takes the function which you wanted to run and later it takes the duration of time (after how long) you wanted to run that function.

In below given example it’s also a callBack function. As it first takes the a function as an argument, the function you wanted to call. Then it takes another function as an input where you put the time duration, i.e after how long you wanted to execute that first function.



**SetInterval()**

setInterval is a function where you can call a function again and again. You can also set the duration after how long you want to run that function. Additionally you can also set it that how many times you wanted to run a function using something called clearInterval().

setInterval() is also a callback function. As it also first takes first argument as input the function that you wanted to execute and then takes after how long you want to run the same first function again.



What is single thread non-blocking means?

What is the meaning of Asynchronous nature of JavaScript?

**Inbuilt methods of an Array. Some functions those are provided by JS to manipulate arrays.**

Array.length()

Array.indexOf()

Array.lastIndexOf()

Array.slice()

Array.substring() – This one is depreciated btw

Array.replace()

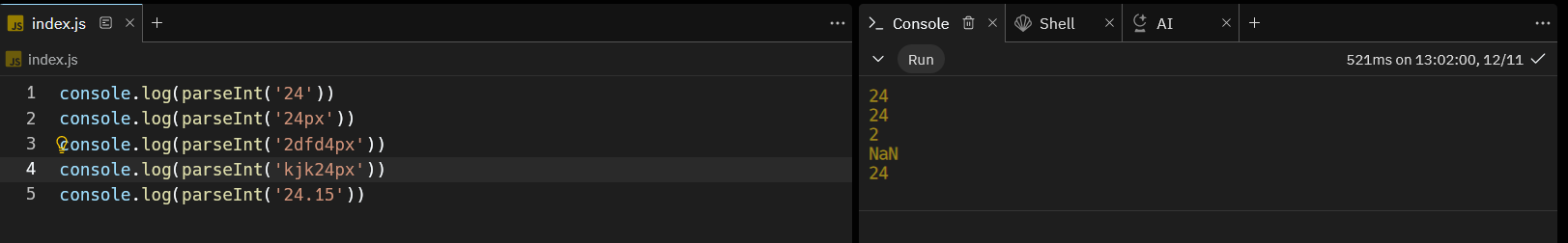
Array.split()

Array.trim()

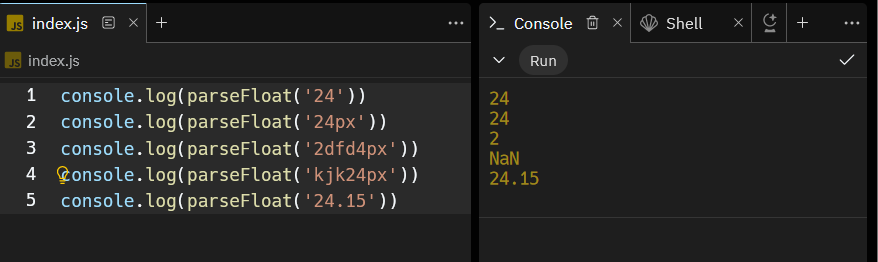
Array.toUpperCase()

Array.toLowerCase()

**Example usage for ParseInt()** – It keeps whole numbers and removes alphabet from at the end of string, can’t remove from at the beginning and from centre. Also doesn’t consider number consist decimals.

****

**Example usage of parseFloat()** – It let you have decimal values too.



**Other array methods**

Array.push() – add front

Array.pop() – remove front

Array.shift() – remove from front

Array.unshift() – add in front

Array.concat() - use for merge

Array.forEach() –

**Callbacks -**

Array.map()

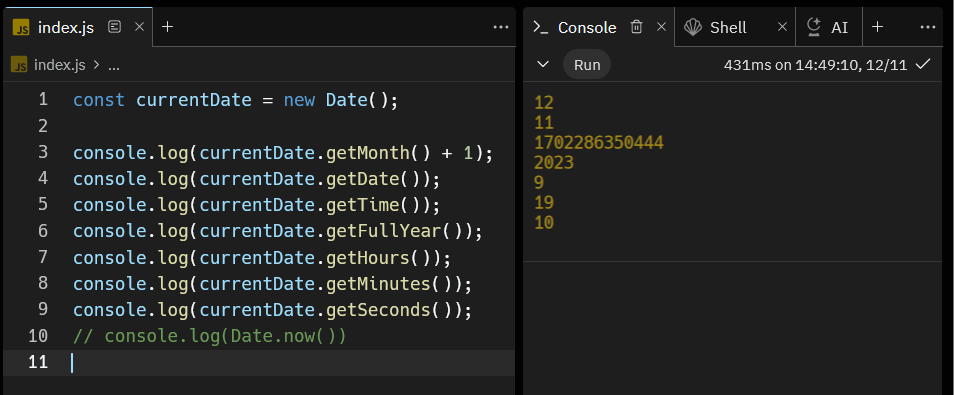
Array.filter()

Array.find()

Array.sort()

**Classes, Objects, Static method**

**Date** – Date features provided by JavaScript



**Some Math methods in JS**

Math.round()

Math.ceil()

Math.floor()

Math.randon()

Math.max()

Math.min()

Math.pow()

Math.sort()

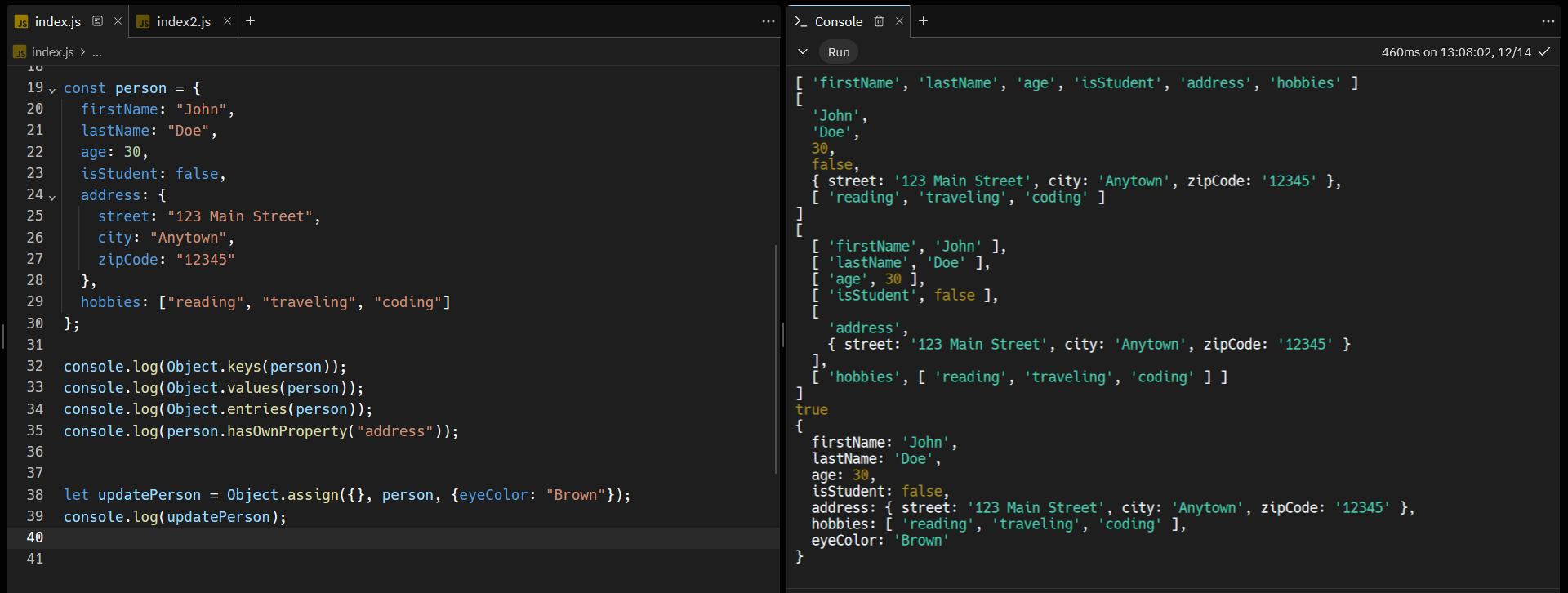
**Object Methods in JS**

Below is an object in JS. In objects the data is stored in Key: value fashion. As you can in below example we have created an object and named it as person. It has Keys like (firstName, lastName, age, isStudent) etc.

Also, it has values like (john, Doe, 30, false) etc.

Further we have some methods that are provided by JS to use on these objects.



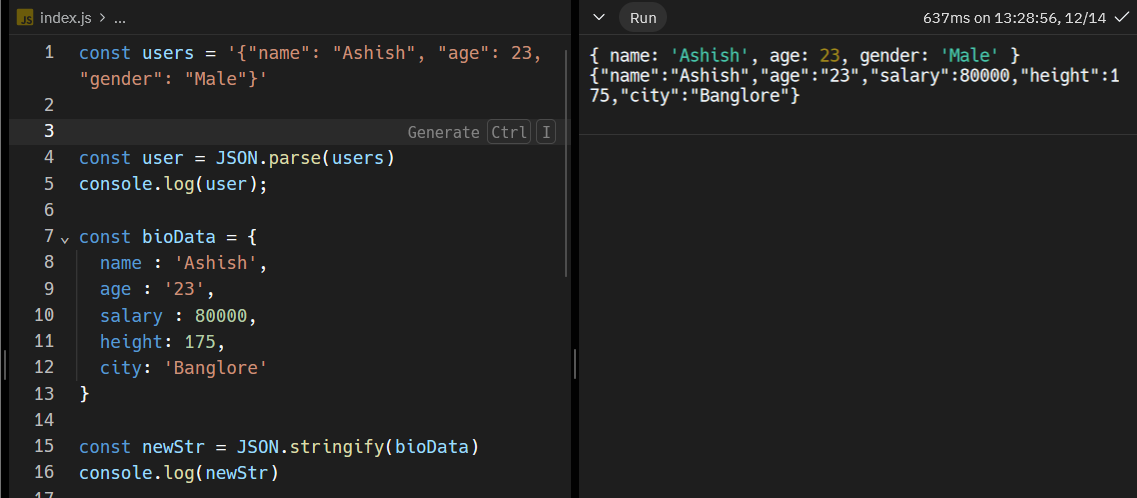
**Few methods we can use on Objects in JS**

**JSON** – JavaScript Object Notation.

JSON is text-based file format used to typically transferring data between a web servers and a client. It represents structured data based on JavaScript object syntax. That means JavaScript programme can convert JSON data to JavaScript object format without parsing and scenarizing the information.

JSON is also compatible other programming languages such as C++, PHP, RUBY

Below is an example that how easily you can convert a JSON file into JavaScript Object and vice-versa.



**Async function vs Sync Functions**

Question - What does the synchronous mean?

Answer: Together, one after the other, sequential. Only one thing is happening at a time.

In a synchronous program, tasks are executed one after the other in a sequential manner. Each task must finish before the program proceeds to the next one.

Question – What does asynchronous mean?

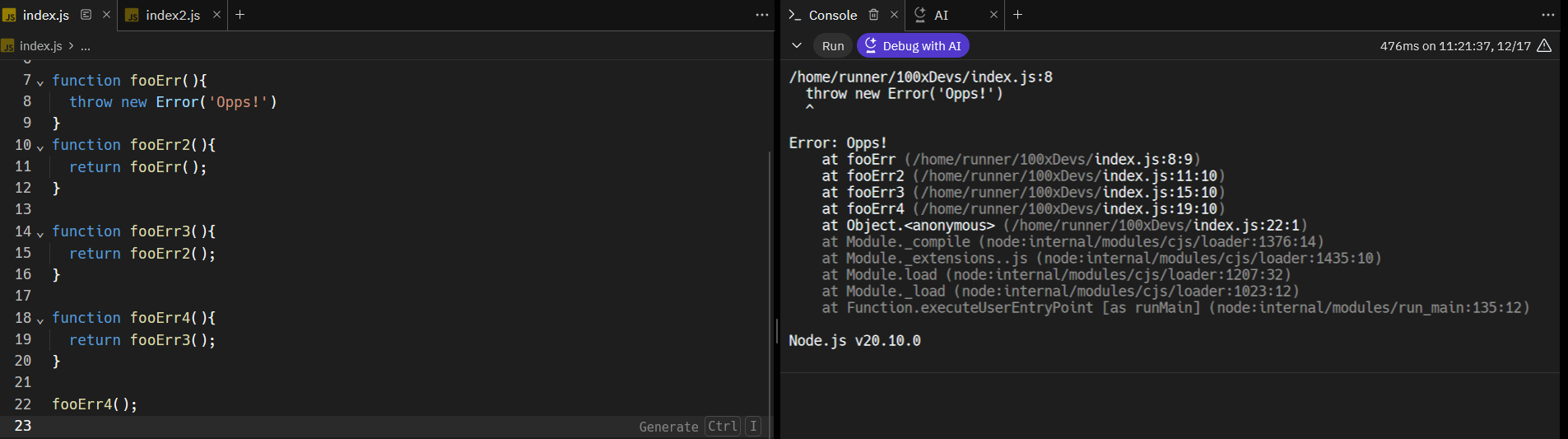
Answer: Asynchronous programming allows tasks to be initiated and run in the background, and the program can continue executing other tasks without waiting for the asynchronous tasks to complete. Multiple things are context switching with each other.

**Single-threaded, Non-blocking, Asynchronous, Concurrent**

One Thread == one call stack == one thing at a time.

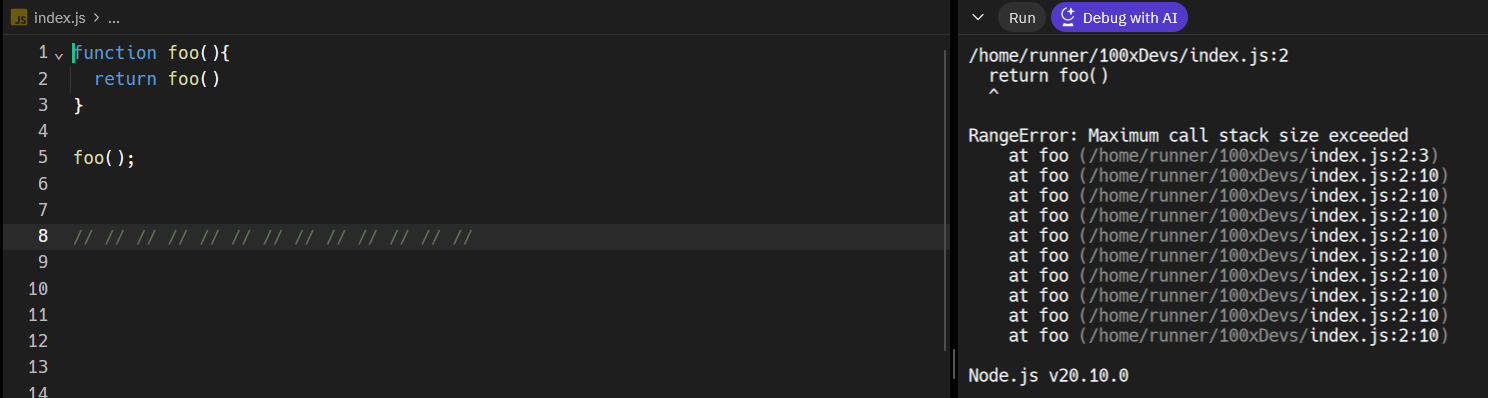
**JS has Call Stack, an event loop, callback queue and some other APIs and stuff  
Call Stack** -> is a data-structure which records where in a program you are. So, if you step into a function, you put something top of the stack and when you return a function you pop-off something top of stack. What that means is when you start a programmed all the function get stack above one another. First function will enter first in stack queue than second above it, so on so forth. Now as this function will return the result it will start deleting the function from top of the stack. The function which entered the stack last will return its value first and get out of stack first.

Here in below example you can see how we stacked function in a call stack and when returning the values how fooErr is called first then fooErr2 then fooErr3 and then so on so forth.



**Maximum Call Stack Size**

As the function gets stacked in call stack, the call stack has a limit too. That how many function can get stacked in call stack. Here’s below is an example of when a call-stack gets maximized and stack size exceeds it’s limit



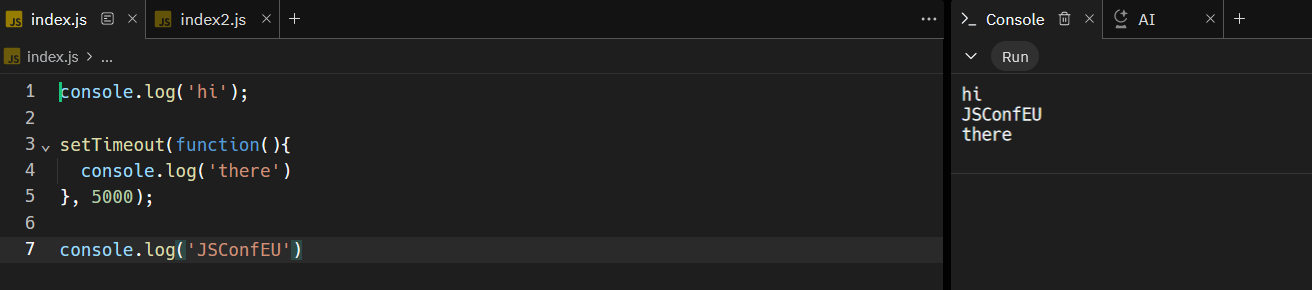
**BLOCKING** – there is definition of blocking, but blocking means that the code that is very slow. Some processes are slower than other process. For example, reading or writing a file in NODE is slow or we can say that image processing is a slow process or requesting something from other web-servers can be slow.

Let take an example – as we know the loop is a synchronous process- what does that means that if you’ll try to run a program which consists a loop (it has to count from 0 to 1 billion) and when you’ll run this program it will occupy the thread & rest of the program won’t be able to run until this loop is finished. So, that way you won’t be able to render anything on your website. Or say if a network request is taking too long in that case too your browser will freeze and you won’t be able to do anything. To tackle this problem, we use something called Asyn-Callbacks

**Asyn-Callback –** setTimeOut() is one those asyn-callback. As we have already discussed that JS has single stack and if something is blocking that stack. Then our whole website will freeze. So setTimeOut() will help us work around that and save ourself from that.

Question - How does that work?

Take an example of below code. Here as per usual flow of the code. The first thing that should shown up in console is (“hi” followed by “there” and then “JSConEU” at last) but it didn’t. The sequence was (‘hi’ > ‘JSConfEU’ > ‘there’). Here we have used setTimeOut() function what it did that if handed over the ‘there’ part to the browser and ran the rest of the code first. It kept the main thread from blocking for 5 second as the ‘there’ part of the code was supposed to be print after 5 second. Instead it dedicated that task to web API and itself ran the rest of the code.



**Heap –** It’s the part where memory allocation happens but for now, we can skip that part.

**Call Stack –** it’s an execution context –

**WebAPIs –** DOM(document), ajax (XMLHttpRequest), setTimeOut

Concurrency & the Event loop – One thing at a time, except not really.

See the below image it has 6 elements here

1. Code (where the code is written)
2. Stack (it is provided by JS part of V8 engine where JS runs)
3. WebAPIs – (is a part of browser – provided by the browser)
4. Task Queue
5. Event Loop
6. Console

When you run your code. Every task is stacked in STACK queue. Except the one we handover to webAPIs. How it work is if you put ‘there’ part in stack it will block the STACK for 5 second as it suppose to show up after 5 second. But to save ourself from that the when the setTimeOut() function is called it handovers that function to WebAPIs. WebAPIs are provided by the browser and they are part of the browser. So as soon as they get to the WebAPIs section it starts the timer of 5 seconds. In mean time the stack runs the rest of code and prints ‘Hi’ and ‘JSConfEU’ in the console. As soon as 5 second timer is over the WebAPIs push the function to Task Queue. Then there comes the EVENT LOOP in picture it checks if the STACK is empty. If yes in that case the EVENT LOOP runs and push the function in stack and prints ‘there’ in the console. (Point here to keep in mind that the task of event Loop is to check whether the STACK is empty or not and also if the STACK is empty and there is something waiting in TASK QUEUE it will push the first thing in the TASK QUEUE to the STACK)

Even if the if the setTimeOut() is set of 0 seconds. Even in that case the ‘there’ part will be printed in the end because the purpose of webAPIs is defer something unless the stack is clear.

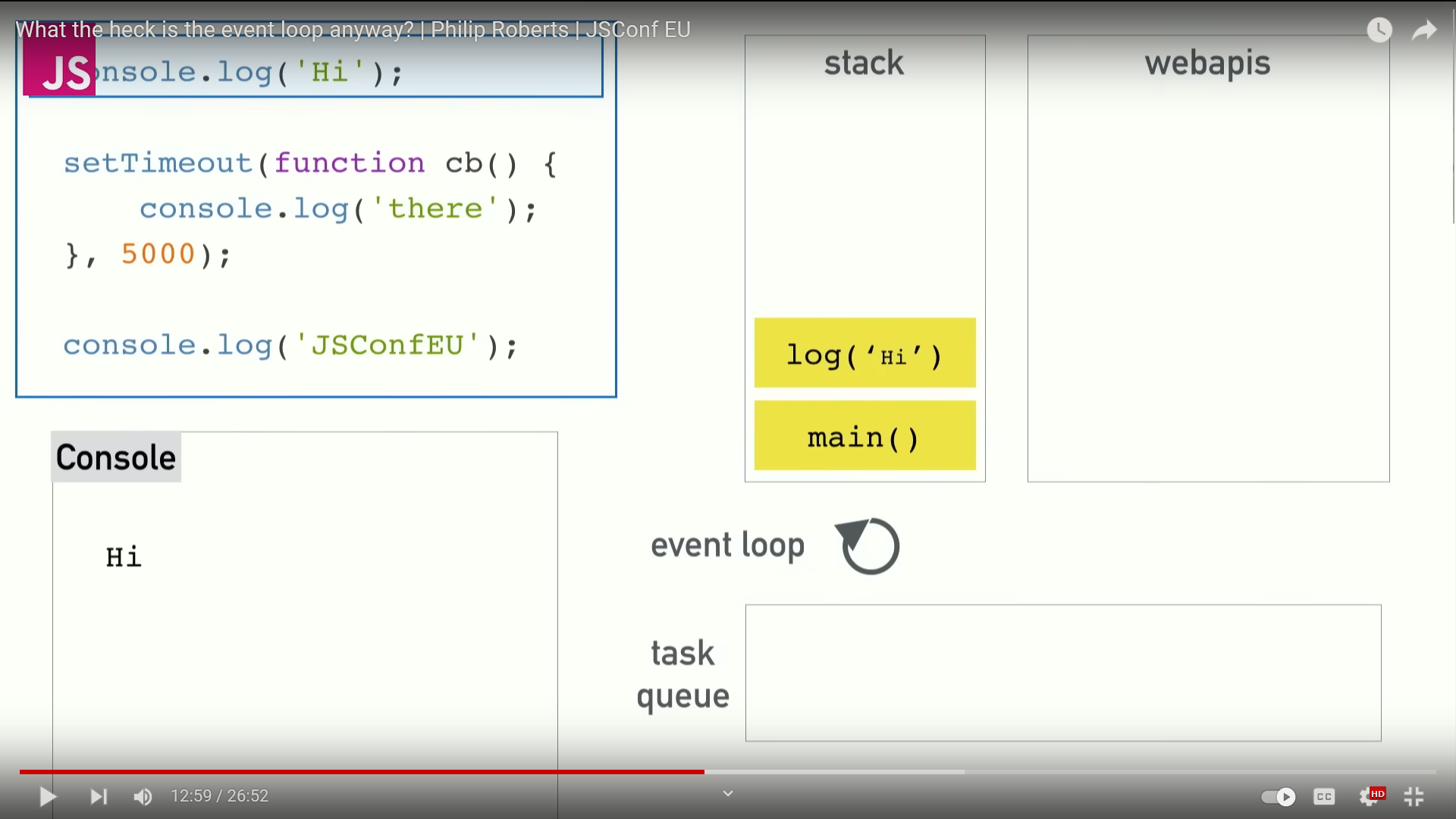
There is also this thing Image rendering in a browser. What it does that it renders the images/graphic on your screen. It happens almost every 16.4 milliseconds. Considering that we are using 60Hz display. Also, if you put a lot of tasks in your stack it will block the image rendering of your system. Which will lead freezing of your web-browser. To save yourself from that we can assign those tasks to web APIs. So that our main thread is not block.

Here’s the priority scale of a stack, image rendering and WebAPIs(task queue)

Stack > image Rendering > WebAPIs

That’s stack can block the image Rendering buy webAPIs won’t

Watch this video here last 10 mins - <https://youtu.be/8aGhZQkoFbQ?si=2JDVlJjtXV5T_41G>



setTimeOut() – Is not a guaranteed time of execution rather it is minimum time of execution. It doesn’t run the next-ish time. (whenever the call-stack is empty)

JavaScript is a common scripting language that's used to make websites interactive. JavaScript is an **asynchronous** and concurrent programming language that offers a lot of flexibility. It's single-threaded like synchronous but also non-blocking like asynchronous

JavaScript is single threaded Asynchronous Language. Whatever code you run in JavaScript engine. It runs inside the something call Stack. What happens as the JavaScript is single threaded language so if you’ll run some code in JS it will be able to execute only one task at a time. And the next code/programme won’t run unless until the previous programme ends running. But there is a problem in this way of executing code. That is what if you’ll try run something (maybe a popup on screen) after a certain time passes. It won’t work because JS doesn’t have that ability to do that. Or in case there is some task/calculation which takes some time to execute because it’s long or complex task, in that case it will block the thread and won’t let other part of the code run.

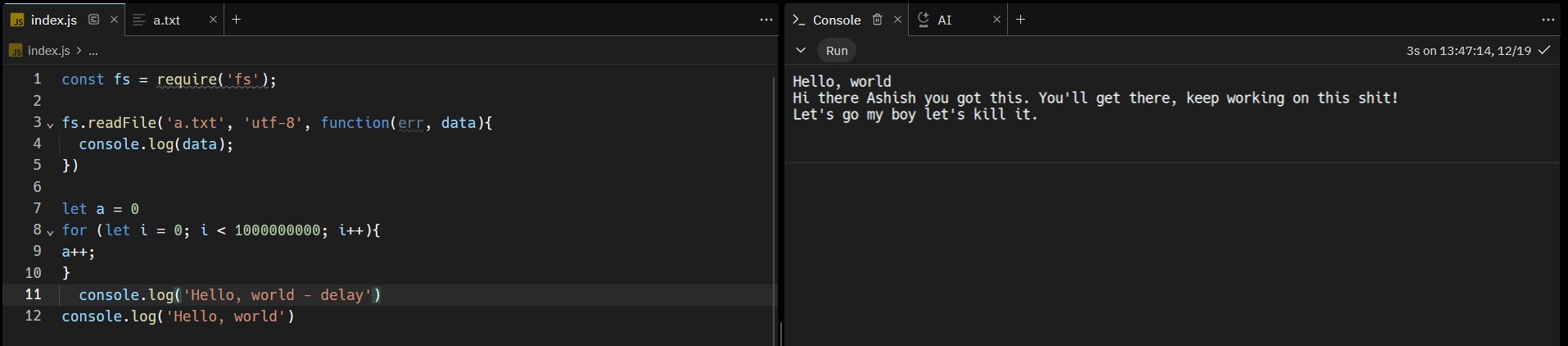
Now to solve this problem the developers has developed a few ways. Using them we can work around these problems. And those features help JS create an Async method of execution. These are not exactly the part of JS rather they are part/ feature of a web-browser. So, yeah you can think of them as web-API.

Other Asyn function in JS

* fs.readFile
* fetch
* setTimeOut()

Fs.readFile – Fs.readfile is a webAPI or it get’s put in task queue. Because while reading file from your system, there can be delay for many reasons (file is locked, big size files etc.) So you want it to hand it hover to the webAPIs so our main thread is not blocked.

Here’s in below example we are printing the data from an outside file(a.txt). See how it get printed in the end because the task queue have least priority here. Also when we try to print something which takes time to do the calculation, here in example it’s calculating upto <100000000. So it keep the main thread occupied and there for it takes 3 seconds to run this simple task. We should assign these type of tasks to our Asyn functions so that it won’t block the main thread.



**Promises**

The promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

It has three states

* Pending
* Fulfilled
* Rejected

Promises are syntactical sugar that makes our code slightly more readable.

Until now we have used other people’s asynchronous functions. How we can create an asynchronous function of our own (not really your own). But you can kind of create a wrapper on top of another async function.

Usually all async function you will write will be on top of JS provided async functions like setTimeOut or fs.readFile.

Difference between Non-Promisified & Promisified (Async function)

Non-Promisified -

* Returns Nothing
* Takes callback as input

Promisified

* Returns a promise
* Doesn’t take as input