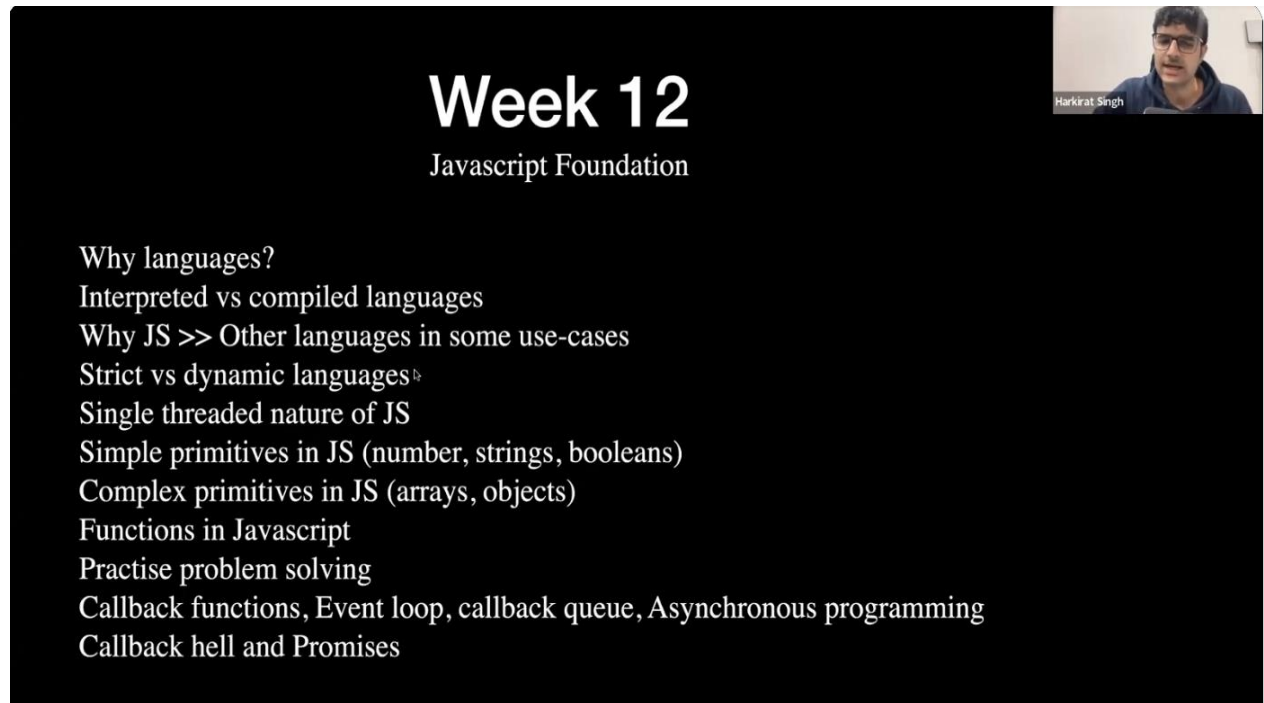


HARKIRAT DEVOPS

JS Foundations [1.2]



Week 12
Javascript Foundation

Why languages?
Interpreted vs compiled languages
Why JS >> Other languages in some use-cases
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, strings, booleans)
Complex primitives in JS (arrays, objects)
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callback queue, Asynchronous programming
Callback hell and Promises

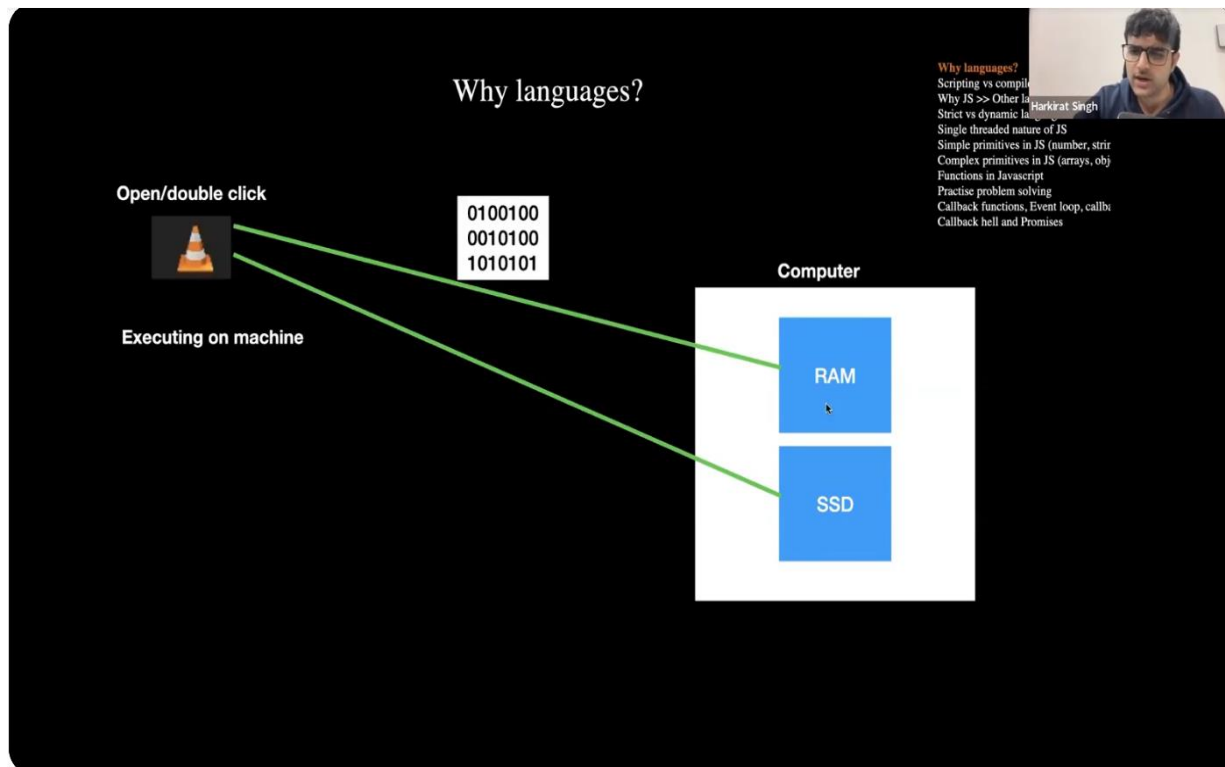
Harkirat Singh

Our first question is why do we need languages?

Before that, we will learn what is RAM and what is SSD.

SSD is a hard drive of a PC, every movie, photo will be stored in SSD.

But when we run something, for example playing movies or having a Zoom call are resides in RAM.



This music player was initially present in SSD, but when we double-clicked it, then it started running/playing and got stored (the thing that goes inside the RAM just the binary bits consisting of 0's and 1's) is inside the RAM.

Now, VLC is created by the use of Languages JS, C++, or any other language and then these languages are converted into 0's and 1's by the compiler so that this VLC can be understood by the machine/PC.

Why languages? What have we learned?

1. Languages are used to write applications
2. Developers write high level code in these languages
3. Every language has a **compiler** which converts the developer code into 01

Why languages?

Scripting vs compile

Why JS >> Other languages

Strict vs dynamic languages

Single threaded nature of JS

Simple primitives in JS (number, string)

Complex primitives in JS (arrays, objects)

Functions in Javascript

Practise problem solving

Callback functions, Event loop, callbacks

Callback hell and Promises

Compiled Vs Dynamic Language

Interpreted vs compiled languages

Compiler

Let's see it in action - The C++ compiler is called g++

Step 1 - write code

```
#include <stdio.h>
using namespace std;

int main() {
    cout << "hello world" << endl;
    return 0;
}
```

Step 2 - Compile code

```
100xdevs g++ a.cpp -o temp
```

Step 3 - Run the code (put it in ram)

```
→ 100xdevs ./temp
hello world
```

Why languages?

Scripting vs compile

Why JS >> Other languages

Strict vs dynamic languages

Single threaded nature of JS

Simple primitives in JS (number, string)

Complex primitives in JS (arrays, objects)

Functions in Javascript

Practise problem solving

Callback functions, Event loop, callbacks

Callback hell and Promises

Compiled Language (C++): In compiled languages like C++, the source code is translated into machine code (binary code) before execution. This translation process is done by a compiler, which converts the entire source code into an executable file that can be directly run by the computer's processor. This executable file contains machine code specific to the target platform. Compilation typically involves several

steps such as preprocessing, compilation, assembly, and linking. Once compiled, the resulting binary code can be executed independently of the original source code.

Interpreted vs compiled languages

Compiler

But JS is different (interpreted)

Step 1 - write code

```
console.log("Hello world");
```

Step 2 - Run code

```
→ 100xdevs node a.js  
Hello world
```

Why languages?

- Scripting vs compile
- Why JS >> Other languages
- Strict vs dynamic language
- Single threaded nature of JS
- Simple primitives in JS (number, string)
- Complex primitives in JS (arrays, objects)
- Functions in Javascript
- Practise problem solving
- Callback functions, Event loop, call stack
- Callback hell and Promises

Interpreted Language (JavaScript): In interpreted languages like JavaScript, the source code is not translated into machine code before execution. Instead, the source code is directly executed by an interpreter line by line. When a JavaScript program is run, each line of code is read, parsed, and executed by the interpreter in real-time. This means that there is no separate compilation step, and the source code is translated into machine code at runtime. Interpreted languages are often platform-independent since they rely on the interpreter to execute the code, making them more flexible in terms of deployment.

Interpreted vs compiled languages

Compiler

Compiled languages

1. First need to compile, then need to run
2. Usually don't compile if there is an error in the code
3. Example - C++, Java, Rust, Golang

Interpreted Languages

1. Usually go line by line
2. Can run partially if the error comes later
3. Example - Javascript, Python

Why languages?
 Scripting vs compile
 Why JS >> Other languages
 Strict vs dynamic languages
 Single threaded nature of JS
 Simple primitives in JS (number, string)
 Complex primitives in JS (arrays, objects)
 Functions in Javascript
 Practise problem solving
 Callback functions, Event loop, callbacks
 Callback hell and Promises

Why JS is better than other Languages?

Why is JS better than other languages

Browsers can only understand HTML/CSS/JS (not technically true)
Thanks to Node.js , Javascript can also be used for “Backend Development”





Why languages?
 Scripting vs compile
 Why JS >> Other languages
 Strict vs dynamic languages
 Single threaded nature of JS
 Simple primitives in JS (number, string)
 Complex primitives in JS (arrays, objects)
 Functions in Javascript
 Practise problem solving
 Callback functions, Event loop, callbacks
 Callback hell and Promises

Static Vs Dynamic Language

Static vs dynamic languages

C++

```
#include <iostream>
using namespace std;

int main() {
    int number = 5;    // Declaration of an integer variable
    number = "Hello";  // This will cause a compile-time error

    cout << number << endl;
    return 0;
}
```

Benefits - More strict code

Javascript


```
let number = 5;    // Variable initially holds a number
number = "Hello";  // Variable now holds a string

console.log(number); // Outputs: "Hello"
```

Benefits - Can move fast

Why languages?

- Scripting vs compile
- Why JS >> Other la
- Strict vs dynamic
- Single threaded nature of JS
- Simple primitives in JS (number, strir
- Complex primitives in JS (arrays, obj
- Functions in Javascript
- Practise problem solving
- Callback functions, Event loop, callb
- Callback hell and Promises



28:56/02:22:11

1.75x

Static Language (e.g., C++): In a static language like C++, the type of each variable is determined at compile-time and cannot change during runtime. This means that the type of every variable must be explicitly declared in the source code, and the compiler checks for type correctness before generating the executable code. Static languages often enforce strong type checking, meaning that type errors are caught at compile-time, reducing the likelihood of runtime errors related to type mismatches. Examples of static languages include C++, Java, and Rust.

Dynamic Language (e.g., Python): In a dynamic language like Python, the type of variables is determined at runtime, and variables can change their type during the execution of the program. This means that variables do not need to be explicitly typed in the source code, and their type can be inferred based on the value assigned to them. Dynamic languages typically perform type checking at runtime, allowing for more flexibility but potentially leading to runtime errors if types are incompatible. Examples of dynamic languages include Python, JavaScript, and Ruby.

Single Threaded Nature of JS

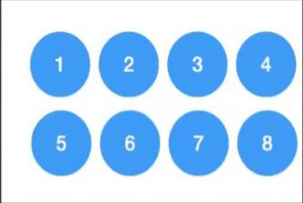
Single threaded nature of JS


Why languages?
Scripting vs compile
Why JS >> Other la
Strict vs dynamic la
Single threaded nature of JS
Simple primitives in JS (number, strir
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callb
Callback hell and Promises

Mac Machine

Hardware Overview:

Model Name:	MacBook Pro
Model Identifier:	MacBookPro18,2
Chip:	Apple M1 Max
Total Number of Cores:	10 (8 performance and 2 efficiency)
Memory:	32 GB
System Firmware Version:	7450.144.1





Each core will run a single process at a time, more the no. of cores we have more the no. of programs we can run at a time.

The single-threaded nature of JavaScript refers to how JavaScript code is executed within a single thread in the browser environment.

In web browsers, JavaScript is primarily used for client-side scripting, where it interacts with the Document Object Model (DOM) to manipulate web page content, respond to user actions, and perform other tasks. The JavaScript engine, which executes JavaScript code, operates within a single thread known as the main thread.

The JavaScript engine processes tasks in a single-threaded manner. It picks up tasks from the event queue one by one and executes them sequentially on the main thread. This means that only one piece of JavaScript code is executed at a time, and other tasks must wait in the event queue until the current task is completed.

Single threaded nature of JS

Why languages?
Scripting vs compile
Why JS >> Other la
Strict vs dynamic la
Single threaded nature of JS
Simple primitives in JS (number, strir
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callbs
Callback hell and Promises



JS can only use one of these at a time
It is **single threaded**
This is why it is considered to be a bad language for
scalable systems
There is a way to make it use all cores of your machine

Mac Machine



Single threaded nature of JS

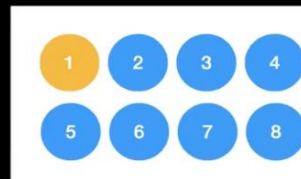
Why languages?
Scripting vs compile
Why JS >> Other la
Strict vs dynamic la
Single threaded nature of JS
Simple primitives in JS (number, strir
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callbs
Callback hell and Promises



More practically, JS runs line by line and only
One line runs at a time

```
index.js  [icon] [x] [+]  
index.js  
1 console.log("hi there");  
2 console.log(a);|
```

Mac Machine



Simple Primitives

Simple primitives

Variables (let, var, const)

Data types - strings, numbers and booleans

If/else

Loops - For loop

Let's write some code -

1. Write the program to greet a person given their first and last name
2. Write a program that greets a person based on their gender. (If else)
3. Write a program that counts from 0 - 1000 and prints (for loop)

Why languages?

Scripting vs compile

Why JS >> Other languages

Strict vs dynamic languages

Single threaded nature of JS

Simple primitives in JS (number, string, booleans)

Complex primitives in JS (arrays, objects)

Functions in Javascript

Practise problem solving

Callback functions, Event loop, callba

Callback hell and Promises

Complex Primitives

Complex primitives

1. Arrays

2. Objects

Let's write some code -

1. Write a program prints all the even numbers in an array
2. Write a program to print the biggest number in an array
3. Write a program that prints all the male people's first name given a complex object
4. Write a program that reverses all the elements of an array

Why languages?

Scripting vs compile

Why JS >> Other languages

Strict vs dynamic languages

Single threaded nature of JS

Simple primitives in JS (number, string, booleans)

Complex primitives in JS (arrays, objects)

Functions in Javascript

Practise problem solving

Callback functions, Event loop, callba

Callback hell and Promises

Functions

Functions

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, s
booleans)
Complex primitives in JS (arrays, ob
Functions in Javascript
Practise problem solving
Callback functions, Event loop, call
Callback hell and Promises

Functions let you

1. Abstract out logic in your program
2. Take arguments as an input
3. Return a value as an output
4. You can think of them as an independent program that is supposed to do something given an input
5. Functions CAN take other functions as input - this will confuse you (callbacks)

Let's write some code -

1. Write a function that finds the sum of two numbers
2. Write another function that displays this result in a pretty format
3. Write another function that takes this sum and prints it in passive tense

Functions

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, st
booleans)
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callb
Callback hell and Promises

Functions let you

1. Abstract out logic in your program
2. Take arguments as an input
3. Return a value as an output
4. You can think of them as an independent program that is supposed to do something given an input
5. Functions CAN take other functions as input - this will confuse you (callbacks)

<https://gist.github.com/hkirat/898ac1da32b6b347a8c0c3e73e1c0666>

```
index.js > ...
1 function sum(num1, num2) {
2   let result = num1 + num2;
3   return result;
4 }
5
6 function displayResult(data) {
7   console.log("Result of the sum is : " + data);
8 }
9
10 function displayResultPassive(data) {
11   console.log("Sum's result is : " + data);
12 }
13
14 // You are only allowed to call one function after this
15 // How will you displayResult of a sum
```

Synchronous and Asynchronous functions

Callback functions, event loops, callback queue

Synchronous vs Asynchronous functions

Synchronous

All the code we've written until now
All code running line by line (hence sync)

Asynchronous

Asynchronous functions in programming are those that allow a program to start a potentially long-running operation and continue executing other tasks without waiting for that operation to complete. This is particularly important in environments like web browsers or Node.js, where waiting for an operation to finish (like fetching data from a server or reading a large file) could make the application unresponsive.

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some t
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, st
booleans)
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

Callback functions, event loops, callback queue

Synchronous vs Asynchronous functions

Synchronous

```
function sum() {  
  let ans = 0;  
  for (let i = 0; i<1000; i++) {  
    ans = ans + i;  
  }  
  return ans;  
}
```

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some t
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, st
booleans)
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

Callback functions, event loops, callback queue

Synchronous vs Asynchronous functions

Asynchronous (setTimeout)

```
index.js > f fetchData > ...  
1 function fetchData() {  
2   console.log('Requesting data from the ChatGPT server...');  
3  
4   setTimeout(() => {  
5     console.log('Data received from the ChatGPT server: []');  
6   }, 3000);  
7 }  
8  
9 fetchData();
```

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, s
booleans)
Complex primitives in JS (arrays, ob
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callb
Callback hell and Promises

Callback Functions

```
main.js — class-1  
JS main.js  
JS main.js > [0] ans  
CodiumAI: Options | Test this function  
1 function sum(num1, num2, fnToCall) {  
2   let result = num1 + num2;  
3   fnToCall(result);  
4 }  
CodiumAI: Options | Test this function  
5 function displayResult(data) {  
6   console.log("Result of the sum is : " + data);  
7 }  
CodiumAI: Options | Test this function  
8 function displayResultPassive(data) {  
9   console.log("Sum's result is : " + data);  
10 }  
11  
12 // You are only allowed to call one function after this  
13 ⚠ How will you displayResult of a sum  
14 const ans = sum(1, 2, displayResult);
```

Ek functions argument ki tarah pass kia h bus ,
aur usko use krliya h , bus itna hi h callback
function.

