Backend Development with NodeJS

AMIT DHOJU

Collaboration With :

WorldLink
CORPORATE SOCIAL RESPONSIBILITY

THE ALGORITHM
Tech for inclusive democracy

Mr. Amit Dhoju
Sr. Software Engineer

TOC

Day 1: JavaScript Fundamentals & Node.js Introduction

Section 1: JavaScript Basics

Javascript Scope

- scope
- closure
- hoisting
- variable declarations
- functions
- data Types

AMIT DHOJU

Javascript Scope

- global scope
- function scope
- block scope

```
let globalVar = "Global"; // Global scope

function test() {
  let localVar = "Function Scope"; // Function scope
  if (true) {
    let blockVar = "Block Scope"; // Block scope
    console.log(blockVar); // ✅ Accessible
  }
  // console.log(blockVar); ❌ Error
}

console.log(globalVar); // ✅
```

AMIT DHOJU

Javascript Closure

- A closure happens when a function remembers variables from its outer scope, even after that outer function has finished running.
- Why?
  - Because in JavaScript, functions carry their scope with them.

```
function outer() {
  let counter = 0; // Outer
  return function inner() {
    counter++;
    console.log(counter);
  };
}

const increment = outer();
increment(); // 1
increment(); // 2
increment(); // 3
```

AMIT DHOJU

Javascript Hoisting

- hoisting means JavaScript moves **declarations** (not initializations) to the top of their scope **before execution**.

For variables:

- var is hoisted **and** initialized with undefined.
- let and const are hoisted but **not initialized** (Temporal Dead Zone).

For functions:

- **Function declarations** are hoisted completely (you can call them before they're defined).
- **Function expressions** are hoisted like variables.

AMIT DHOJU

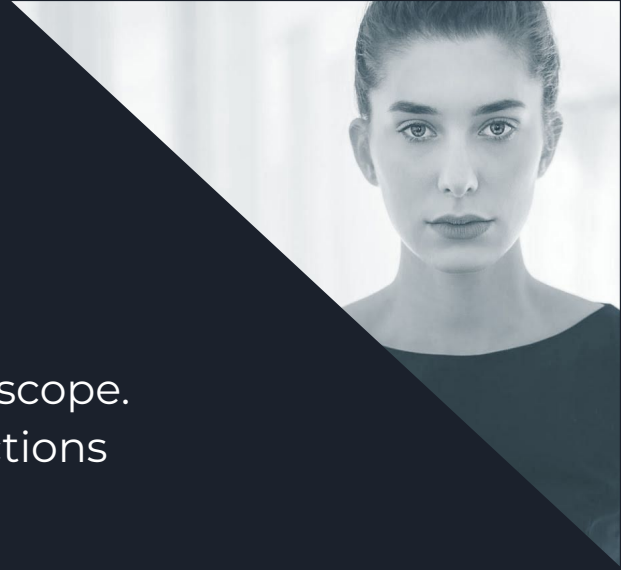
Javascript Hoisting Example

```
console.log(a); // undefined (var is hoisted)
var a = 5;

sayHi(); // ✅ Works
function sayHi() {
  console.log("Hi!");
}

// greet(); ❌ Error
const greet = function() {
  console.log("Hello!");
};
```

AMIT DHOJU

In Brief

- **Scope** decides where variables exist.
- **Closure** allows a function to remember its scope.
- **Hoisting** changes when variables and functions become available.



AMIT DHOJU

Variable declarations

- var
- let
- const
- which one to use?

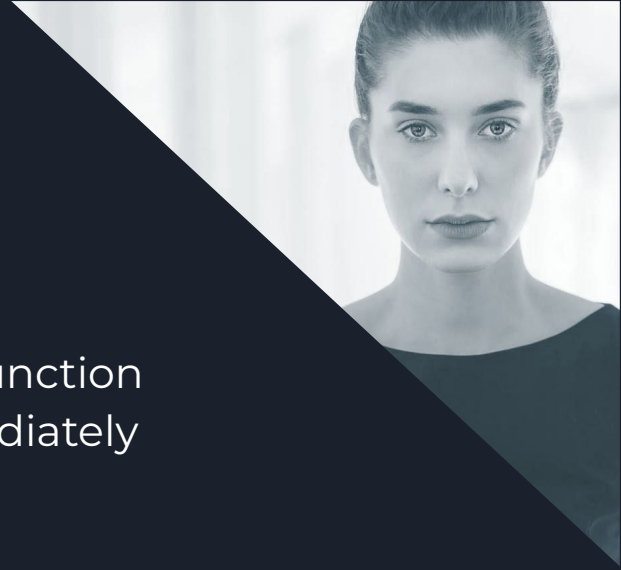
AMIT DHOJU

Javascript Functions

- regular functions (link)
- arrow functions (link)
- high order functions

AMIT DHOJU

Regular functions

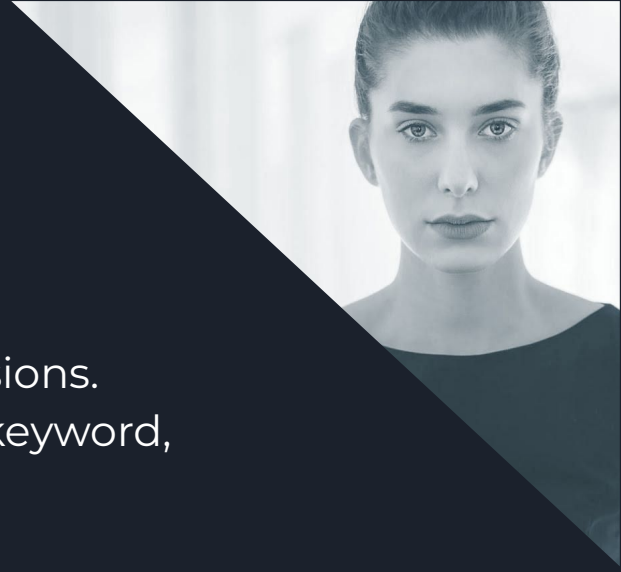


- javaScript functions are defined with the function
- declared functions are not executed immediately
- executed after its been called
- Link
- example

```
function functionName(parameters) {
  // code to be executed
}
```

AMIT DHOJU

Arrow functions



- allows a shorter syntax for function expressions.
- no need the function keyword, the return keyword, and the curly brackets.
- executed after its been called
- Link
- example

Before Arrow:

```
let hello = function() {
  return "Hello World!";
}
```

With Arrow Function:

```
let hello = () => {
  return "Hello World!";
}
```

AMIT DHOJU

High order functions

- A Higher-Order Function (HOF) in JavaScript is simply a function that does at least one of these
  - Takes another function as an argument
  - Returns a function as its result
- Example
  - Array methods: map, filter, reduce, forEach, sort

```
function multiplier(factor) {
  return function (number) {
    return number * factor;
  };
}

const double = multiplier(2);
console.log(double(5)); // 10
```

AMIT DHOJU

Asynchronous Programming

1. Callbacks: ***"I will call back later!"***
2. Asynchronous : “***I will finish later!***”
3. Promises: "***I Promise a Result!***"
4. Async/await: “***i will make promises easier to write***“

AMIT DHOJU

Callbacks Functions

- function passed as an argument to another function
- allows a function to call another function
- can run after another function has finished
- simplified use.
- Link: W3 schools Callbacks

AMIT DHOJU

Asynchronous Functions

- functions running in parallel with other functions are called asynchronous
- A good example is JavaScript setTimeout()
- Right: setTimeout(myFunction, 3000);
- LInk: [W3 schools Async]

AMIT DHOJU

Promises Functions

- "Producing code" is code that can take some time
- "Consuming code" is code that must wait for the result
- object that links Producing code and Consuming code
- A JavaScript Promise object can be:
  a. Pending
  b. Fulfilled
  c. Rejected
- Links: W3 schools Promise

AMIT DHOJU

Async/await Functions

- **async** makes a function return a Promise
- **await** makes a function wait for a Promise
- The await keyword makes the function pause the execution and wait for a resolved promise before it continues:
- Link: W3Schools Async/await

AMIT DHOJU


Any Questions??
AMIT DHOJU

Section 2: Introduction to **Node.js**

Some history facts.

- 1995 – Birth of JavaScript
  - Created by Brendan Eich at Netscape in just 10 days.
  - Originally called Mocha, then LiveScript, and finally JavaScript.
  - Designed to make web pages interactive inside the Netscape Navigator browser.
- 2009 – Creation
  - Created by Ryan Dahl.
  - Used Google's V8 JavaScript Engine (from Chrome) to run JS outside the browser.
  - Introduced non-blocking, event-driven I/O, perfect for scalable network apps.

What is Node.js

- Node.js is an open-source, cross-platform runtime environment that lets you run JavaScript outside the browser.

- Built on Google's V8 JavaScript Engine (the same engine powering Chrome).

- Uses an event-driven, non-blocking I/O model, which makes it lightweight and efficient — perfect for real-time applications.

Why Node.js?

- One language for full stack
  - JavaScript on both frontend and backend.
- High performance
  - Non-blocking I/O handles thousands of requests without slowing down.
- Rich ecosystem
  - npm has millions of packages ready to use.
- Scalability
  - Ideal for microservices, APIs, streaming apps, chats, and more.

AMIT DHOJU

Installing

- Visit https://nodejs.org
  - LTS (Long-Term Support) version → stable for most users
  - Current version → latest features (may be less stable)
- Verify installation
  - `node -v` # Check Node.js version
  - `npm -v` # Check npm version
  -

AMIT DHOJU

Running Node.js code

- One language for full stack
  - JavaScript on both frontend and backend.
- High performance
  - Non-blocking I/O handles thousands of requests without slowing down.
- Rich ecosystem
  - npm has millions of packages ready to use.
- Scalability
  - Ideal for microservices, APIs, streaming apps, chats, and more.

AMIT DHOJU

Running Node.js/Hello World

- Run in console
- Run a function in browser console
- Run a .js file

Understanding the Event Loop

- Node.js uses single-threaded, non-blocking architecture.
- The Event Loop is the heart of Node.js — it lets Node handle many tasks asynchronously.
- This means Node can handle thousands of requests without creating new threads.

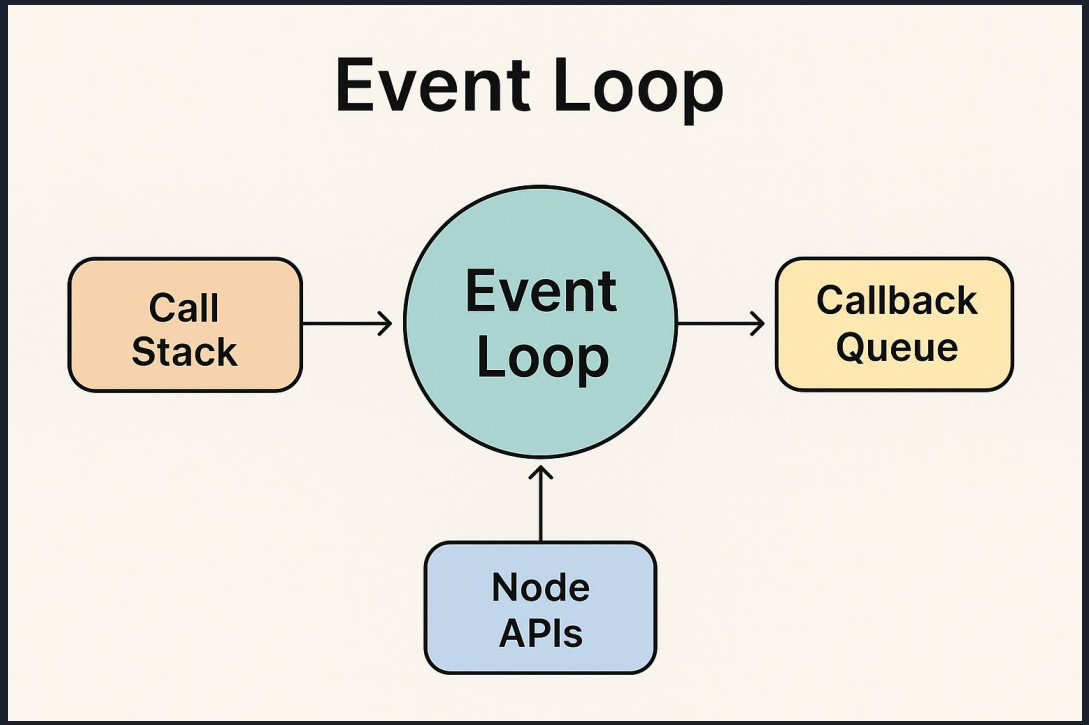
AMIT DHOJU

Event Loop

- The mechanism that allows Node.js to handle many tasks at once without creating multiple threads.
- Single thread runs JavaScript code.
- When you do something slow (like reading a file or calling an API), Node hands it off to the system to do in the background.
- Once the task is done, the callback (or promise) is queued.
- The Event Loop checks if the main thread (call stack) is free — if yes, it runs the callback.

AMIT DHOJU

Event Loop



AMIT DHOJU

How Event Loop Works?

- Call Stack executes JS code.
- When Node encounters async tasks (e.g., file read, HTTP request), it sends them to the Node APIs.
- When tasks complete, callbacks are moved to the Callback Queue.
- Event Loop checks if the Call Stack is free → moves queued callbacks into the stack to execute.

AMIT DHOJU

Examples

Code

```
console.log("Start");

setTimeout(() => {
  console.log("Timeout finished");
}, 2000);

console.log("End");
```

Output

```
Start
End
Timeout finished
```

AMIT DHOJU

Hands-On Exercise: Print triangle pattern


Any Questions??
AMIT DHOJU

Section 3: Node.js Core Modules

Node Core Modules

1. File System (fs)File operations like reading, writing and manipulating files
2. Path (path)Utilities for working with file and directory paths
3. HTTP (http)Create HTTP servers and make HTTP requests
4. HTTPS (https) – Secure version of HTTP using TLS/SSL

AMIT DHOJU

Node Core Modules 2

1. Console (console) – Provides simple logging methods like console.log, console.error, console.table

2. Timers (timers) – Functions for scheduling code execution: setTimeout, setInterval, setImmediate

3. Stream (stream) – Working with streaming data (readable, writable streams, files)

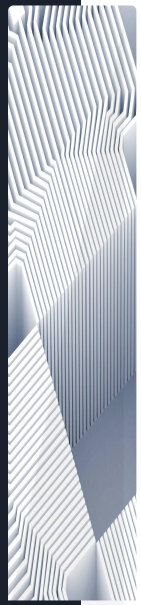
4. Buffer (buffer) – Handling binary data

AMIT DHOJU

Library VS Framework??

Library

- Pre-written code for specific tasks.
- You control when and how it’s used.
- Example: npm



Framework

- Defines overall structure
- It controls show things flow
- Examples: expressJs

AMIT DHOJU

Section 4:
Node.js Frameworks:
Express.js

Some Node js Framework

1. ExpressJS
2. NestJS
3. KoaJS
4. AdonisJS
5. FastifyJS
6. TotalJS

AMIT DHOJU

Express.js

1. Express.js is a minimalist, fast, lightweight and flexible web framework for Node.js.

2. It simplifies building web servers and APIs by providing a clean set of features like routing, middleware support, and more.

3. It’s the most popular framework in the Node.js ecosystem for backend development.

4. Link

AMIT DHOJU

Working

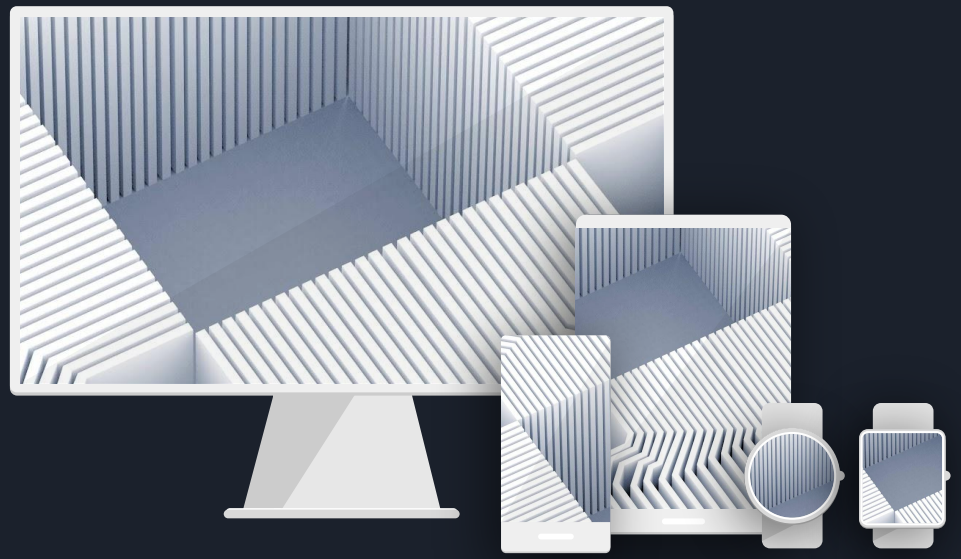
1. Listens for requests (http request from clients)
2. Parse requests (extract data from requests)
3. Match routes (find correct route like /home, /about)
4. Send response to client (returns a suitable response eg, JSON, html etc.

Installing Express JS

npm install express

Hands-On Exercise:

Express app setup



AMIT DHOJU

Overview

- ***express()-*** creates instance of the xpress application
- ***app.get(path, callback)*** - handles get requests
- ***app.post(path, callback)*** - handles post requests
- ***app.listen(port,callback)*** - start server at the the port

Section 5:
Git / Github

Git

- Version control system (VCS). It helps you track changes in your code or files over time.

- How does it work?
  You use Git on your local computer to save snapshots (commits) of your project as you work on it.

- What is it used for?
  Keeping track of code history
  Working with branches (different versions)
  Collaborating by merging changes

- Is it a website? - No, Git is software you install and run locally or on servers.

AMIT DHOJU

Github

- Web-based platform built on top of Git.

- What does it provide?
  A place to host your Git repositories online
  Tools for collaboration: Pull requests, issues, code reviews
  Social features like profiles, following, stars
  CI/CD integrations, project management, wikis

- Is it the same as Git?- No. GitHub uses Git for version control but adds a user-friendly interface, many services.

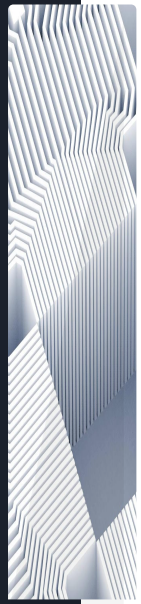
- Are there alternatives?- Yes! Examples: GitLab, Bitbucket, Azure DevOps — all are Git repository hosting platforms with their own features

AMIT DHOJU

Git vs Github

Git

- Version control system
- Track/manage code locally
- Locally on your computer
- Provides git commands & workflow
- Git is like docs file; write, edit docs



Github

- Online Git repository hosting
- Share, collaborate, manage code online
- Accessed through a web browser
- Provides web interface, collaboration tools
- Github is like google drive; you can upload, share and access from anywhere.

AMIT DHOJU


Any Questions??
AMIT DHOJU

That’s all for today🫡

Thank you for your time

Thank You!