NodeJS Training - Day 1

By : LAKSHMIKANT DESHPANDE (M.Tech)

Few Lines About Lakshmikant Deshpande..!

Lakshmikant Deshpande is a solutions architect with 12 years of experience in Java and advanced Java technologies such as Spring, Spring Boot, and Hibernate.

In the JavaScript paradigm, he excels in both MERN and MEAN stacks. His expertise extends to cloud technologies such as AWS and GCP, along with containerization using Docker and Kubernetes.

Lakshmikant is currently exploring Golang and Rust technologies, showcasing his skills as a polyglot programmer.

In addition to his technical abilities, he is a passionate trainer, committed to sharing knowledge and helping others grow.

Outside of technology, Lakshmikant enjoys traveling, biking, hiking, trekking in the Himalayan ranges, and is an avid reader and web series enthusiast.

Day 1 : Overview

- IDE for writing your code (Editor)
- Install Node.js using NVM
- NPM (Node Package Manager)
- GitHub Account Creation
- Introduction to JavaScript
- Understanding JSON
- Advanced Javascript
- ES 6 concepts
- What is Node.js?
- Simple JavaScript Program Execution with Git Commands
- Handling Exceptions

Session - 1

- IDE for writing your code (Editor)
- Install Node.js using NVM
- NPM (Node Package Manager)
- GitHub Account Creation
- Basic git commands
    - Initializing a Repository
    - Cloning a Repository
    - Staging Changes
    - Committing Changes
    - Pushing

Session - 2

- Introduction to JavaScript
- Basics of Javascript
  - Variables
  - Data Types
  - Operators
  - Conditions / Decision making
  - Loops / Iterations
  - Data Storage
  - Function
- Understanding JSON

Session - 3

- Advanced Javascript
  - Hoisting
  - Closures
  - Callback Functions
  - Fetch
  - Promise
  - Async/Await
  - Event Loop

Session - 4

- ES 6 concepts
  - Arrow Functions
  - Template Literals
  - Destructuring Assignment
  - Default Parameters
  - Rest and Spread Operators
  - Class Syntax
- What is Node.js?
- REPL

Session - 4 contd….

- Simple JavaScript Program Execution with Git Commands
- Handling Exceptions Theory
  - Operational Errors & Functional Errors
  - Try.. Catch..

Visual Studio Code

- **Visual Studio Code** is a powerful editor for Node.js development with built-in support for debugging, version control, and terminal integration.
- Install relevant extensions like Node.js Extension Pack, ESLint, and Prettier to enhance your development experience.
- Configure debugging, use the integrated terminal for running scripts, and leverage Git integration for version control.

NVM : Node Version Manager

- **NVM** is a versatile tool for managing multiple versions of Node.js.
- It simplifies switching between versions and setting default versions for different projects.
- Installation is straightforward on Unix-based systems, and a separate version is available for Windows.

NPM (Node Package Manager)

- **NPM** is an essential tool for managing Node.js packages and dependencies.
- It helps in installing, updating, and removing packages, as well as running scripts and publishing packages.
- NPM simplifies dependency management and is a key part of the Node.js development workflow.

GitHub

- **GitHub** is a powerful platform for version control and collaboration in software development.
- It integrates with Git to manage code changes and supports collaboration features like pull requests and issue tracking.
- With GitHub, you can create repositories, clone them, make changes, and use various tools to manage and automate your development workflow.

Introduction to Javascript

What is JavaScript?

- JavaScript is a versatile, high-level programming language.
- It enables dynamic website content and interactive features.

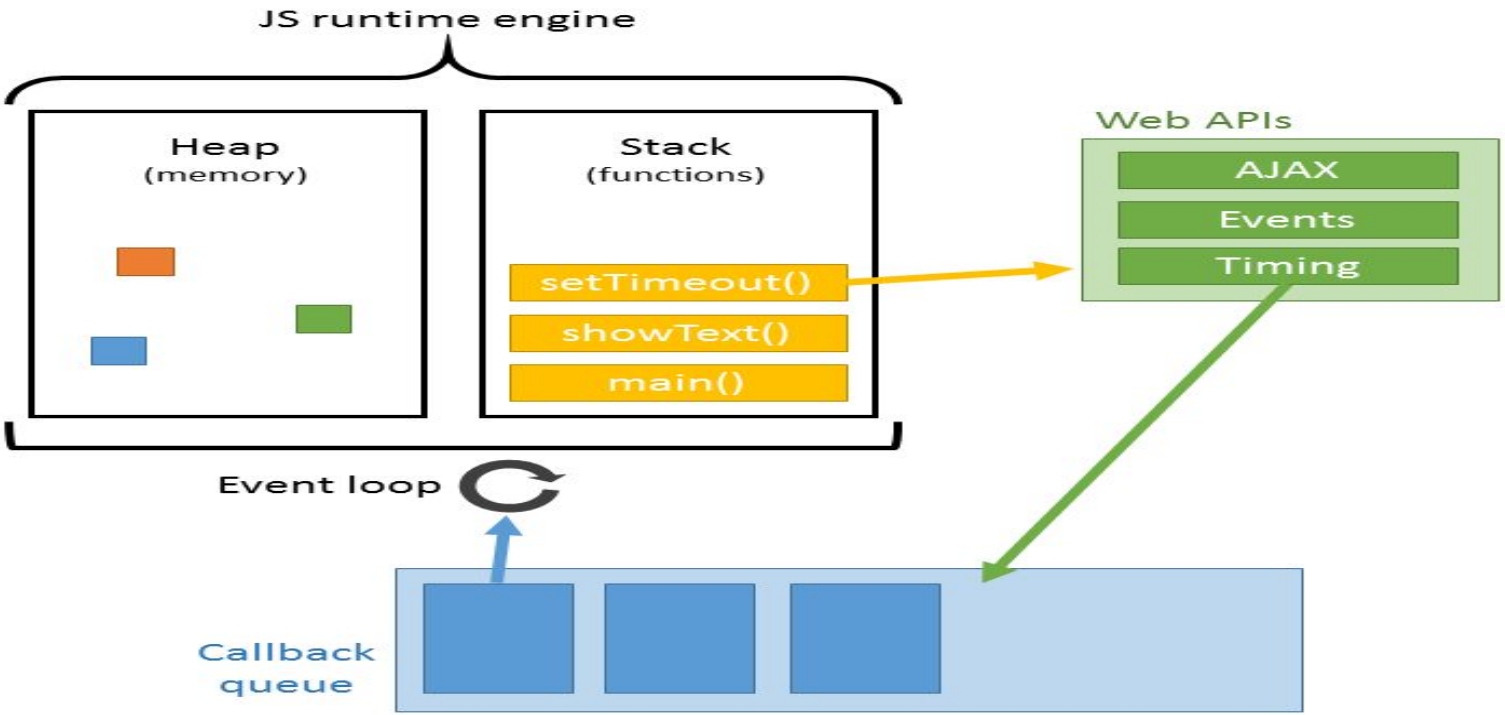
Use of JavaScript

- Enhances web pages, supports server-side (Node.js), and mobile apps.
- Used in game development, desktop apps (Electron), and more.

Trend of JavaScript

- Continues to be one of the most popular programming languages.
- Expands beyond web development into IoT, AI, and other fields.

JavaScript Engine



Javascript In Browser and Node Environment

- **Console**
  - log
  - error
  - warn
  - info
  - clear
  - assert
  - time
  - timeLog
  - timeEnd

- **JavaScript REPL - Read Eval Print Loop**
  - 2 + 3
  - 10 - 4 * 2
  - Math.sqrt(16)

- **Variables**
  - Variables are declared using the var, let, or const keyword.
  - Variables declared with var and let can be reassigned to new values.
  - The var keyword has function scope or global scope, while let has block scope (limited to the nearest enclosing block).
  - Variables declared with var or let can be declared without an initial value and assigned later.
- **Constants**
  - Constants are declared using the const keyword.
  - Constants cannot be reassigned to a new value once they are declared. They are immutable.
  - Constants must be assigned a value at the time of declaration and cannot be declared without an initial value.
  - Constants also have block scope like let.

Data Types

- **Primitive data types.**
  - Number
  - String
  - Boolean
  - Null
  - Undefined
  - Symbol

- **Non-primitive data types.**
  - Object
  - Function
  - Array
  - Date
  - RegExp

Operators

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators
- Conditional (Ternary) Operator
- Type Operators
- String Operators

Loops

- **for loop**: It executes a block of code for a specified number of times, based on a defined initialization, condition, and iteration expression.
- **while loop**: It repeatedly executes a block of code as long as a specified condition is true.
- **do-while loop**: It is similar to the while loop, but it executes the block of code at least once before checking the condition.
- **for...in loop**: It iterates over the properties of an object, providing access to each key.
- **for...of loop**: It iterates over iterable objects (arrays, strings, etc.), providing access to each element.
- **Entry controlled vs exit controlled loops**

Conditions / Decision making

- **Boolean Conditions**
- **Comparison Conditions**
- **Logical Conditions**
- **Ternary Operator**
- **Nested Conditions**
- **Switch Statement**

Function

- **Function Declaration**
- **Function Expressions (Anonymous)**
- **Arrow Functions**
- **Function Expression (Named)**
- **Method Definition (Within an Object)**
- **Function as a Constructor**
- **Immediately Invoked Function Expression (IIFE)**
- **Generator Function**
- **Object Prototype Method**

Understanding JSON (JavaScript Object Notation)

- **JSON** is a simple, text-based format for data interchange.
- **Objects** and **arrays** are the core structures in JSON.
- **Parsing** and **stringifying** JSON are essential for working with JSON data in JavaScript.
- **JSON** is widely used for transmitting data between a client and server, and in various other applications for its simplicity and ease of use.

Advanced Javascript

- Hoisting
- Closures
- Callback Functions
- Fetch
- Promise
- Async/Await
- Event Loop

ES 6 Concepts

- **Arrow Functions:** Shorter syntax for writing function expressions.
- **Template Literals:** Enhanced string literals that support interpolation and multiline strings.
- **Destructuring Assignment:** Extracts values from arrays or objects into individual variables.
- **Default Parameters:** Allows default values to be specified for function parameters.
- **Rest and Spread Operators:** Collects elements into arrays or objects (rest) or spreads them out (spread).
- **Object Literal Extensions:** Shorthand syntax for defining methods and computed property names.
- **Class Syntax:** Simplified syntax for creating and working with classes and objects.
- **let and const:** Block-scoped variables that offer better control over variable declarations.
- **Enhanced for...of Loop:** Iterates over iterable objects (arrays, strings, etc.) with a simpler syntax.
- **Symbol:** A new primitive type that allows the creation of unique and non-enumerable property keys.

What is Node.js?

- **Node.js** is a powerful platform that enables developers to use JavaScript on the server-side, providing efficient, scalable solutions for web applications and other network-based applications. Its asynchronous nature and large ecosystem make it a popular choice for modern development.

Key Features of Node.js:

1. **Event-Driven Architecture**:
    - Node.js uses an event-driven, non-blocking I/O model. This means it can handle many connections simultaneously without being blocked by slow operations, such as reading files or network requests.
2. **Single Programming Language**:
    - With Node.js, developers can use JavaScript on both the client-side and server-side of an application, enabling a more unified development experience.

Key Features of Node.js contd…

- **Asynchronous and Non-Blocking I/O**:
  - Node.js handles I/O operations asynchronously. Instead of waiting for operations like file reads or database queries to complete, it continues executing other code and processes the results when the operations finish.
- **V8 JavaScript Engine**:
  - Node.js uses the V8 JavaScript engine developed by Google, which compiles JavaScript to native machine code for fast execution.
- **NPM (Node Package Manager)**:
  - Node.js comes with a built-in package manager called NPM, which provides access to a vast ecosystem of libraries and modules that can be easily included in projects.
- **Scalability**:
  - Node.js is designed to build scalable network applications. Its event-driven architecture allows it to efficiently handle a large number of simultaneous connections.
- **Community and Ecosystem**:
  - Node.js has a large and active community, contributing to a rich ecosystem of libraries, frameworks, and tools.

Typical Use Cases

Web Servers:

- Node.js is often used to create web servers and APIs. Frameworks like Express.js simplify the creation of web applications and RESTful APIs.

Real-Time Applications:

- Ideal for real-time applications like chat applications, gaming servers, and collaborative tools due to its non-blocking nature and WebSocket support.

Microservices:

- Node.js is frequently used in microservices architectures to build lightweight, modular services.

Command-Line Tools:

- Developers can use Node.js to create command-line tools and scripts due to its ability to handle file system operations and execute scripts efficiently.

Exceptions

- **Exceptions** in Node.js are used to handle errors and manage unexpected conditions that arise during the execution of your application.
- Operational Errors & Functional Errors
- Error Propagation
- Exceptions in Node.js are handled using standard JavaScript mechanisms (try-catch, .catch(), etc.) and specific practices for asynchronous code and event emitters.
- **Custom errors** can be created by extending the Error class.
- Handling **unhandled promise rejections** and errors in **Express middleware** helps maintain application stability and provide meaningful error messages.

Common Exception Types in Node.js

- **SyntaxError**:
  - Occurs when there is a syntax error in your code.
  - Example: Missing or incorrect use of punctuation in code.
- **TypeError**:
  - Occurs when a value is not of the expected type.
  - Example: Calling a non-function variable as a function.
- **ReferenceError**:
  - Occurs when code references a variable that is not defined.
  - Example: Using a variable before declaring it.
- **RangeError**:
  - Occurs when a value is not within the expected range.
  - Example: Passing an invalid range to a function.
- **Error**:
  - The base class for all errors. Custom errors are usually created by extending this class.