INDEX

BACK-END TRAINING CONTENTS

What is back-end?

Back-end development means **working on server-side software, which focuses on everything you can't see on a website**. Back-end developers ensure the website performs correctly, focusing on databases, back-end logic, Application Programming Interface (APIs), architecture, and servers.

What are the components of back-end development?

1. Servers
2. Logic
3. Frameworks
4. Databases
5. APIs

What is JavaScript?

JavaScript is a lightweight programming language that web developers commonly used to create more dynamic interactions when developing web pages, applications, servers, and or even games.

How is JavaScript related to Java?

Java is an OOP programming language while Java Script is an OOP scripting language. Java creates applications that run in a virtual machine or browser while JavaScript code is run on a browser only. Java code needs to be compiled while JavaScript code are all in text.

How JavaScript works?

JavaScript is a single-threaded programming language, which means it has a single Call Stack. Therefore it can do one thing at a time.

How does JavaScript work in web pages?

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.

Variables:

Variables are **containers for storing data values, like numbers and characters**.

Rules:

1. A variable name must only contain alphabets, digits, and underscore.

2. A variable name must start with an alphabet or an underscore only.

It cannot start with a digit.

3. No whitespace is allowed within the variable name.

4. A variable name must not be any reserved word or keyword.

What is Data type?

A data type, in programming, is **a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error**.

There are,

1. Strings,

2. Numbers,

3. Booleans,

4. Undefined,

5. Null.

What is Keyword?

Keywords are **reserved words that are part of the syntax in the programming language**. Keywords cannot be used to name identifiers.

There are,

1. Let,

2. var,

3. Const.

What is operators?

In JavaScript, an operator is **a special symbol used to perform operations on operands.**

1. Arithmetic operators,

2. Comparison operators,

3. Logical operators.

Functions:

A function is **a block of code which only runs when it is called**. You can pass data, known as parameters, into a function. Functions are used to perform certain actions, and they are important for reusing code.

Arguments and Parameters:

- The values that are declared within a function when the function is called are known as an argument.
- The variables that are defined when the function is declared are known as a parameter.

Return Statements:

The return statement **ends function execution and specifies a value to be returned to the function caller**.

Arrow functions:

Arrow functions allows a short syntax for writing function expressions. You don't need the function keyword, the return keyword, and the curly brackets.

Callback function:

A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.

Async/Await:

- **Async:** It simply allows us to write **promises-based** code as if it was synchronous and it checks that we are not breaking the execution thread.
- It operates asynchronously via the event loop.
- Async functions will always return a value. It makes sure that a promise is returned and if it is not returned then JavaScript automatically wraps it in a promise which is resolved with its value.

- **Await:** Await function is used to wait for the promise. It could be used within the async block only. It makes the code wait until the promise returns a result. It only makes the async block wait.

JavaScript Errors Throw and Try to Catch:

try{} statement: The code which needs possible error testing is kept within the try block. In case any error occur, it passes to the catch{} block for taking suitable actions and handle the error. Otherwise, it executes the code written within.

catch{} statement: This block handles the error of the code by executing the set of statements written within the block. This block contains either the user-defined exception handler or the built-in handler. This block executes only when any error-prone code needs to be handled in the try block. Otherwise, the catch block is skipped.

Object:

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Array:

JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

Array Destructuring:

Destructuring in JavaScript is a simplified method of extracting multiple properties from an array by taking the structure and deconstructing it down into its own constituent parts through assignments by using a syntax that looks similar to array literals.

Object Destructuring:

Object destructuring assigns the properties of an object to variables with the same names by default.

Alias:

Creates a reference to a PCI Variable allowing it to be accessed with a different name.

Spread Operator:

The Spread operator allows an iterable to expand in places where 0+ arguments are expected. It is mostly used in the variable array where there is more than 1 value is expected. It allows us the privilege to obtain a list of parameters from an array.

Server:

A server is a software program or a hardware device that provides services to the other connected devices in the network. The other devices are called hosts or clients.

Database:

A database is an organized set of related data. It is a difficult task to develop a database for a complex application. Therefore, the developers can design an ER diagram. It provides a visual representation of the database. Afterward, they can implement the real database using DBMS. DBMS stands for Database Management System. It is capable of communicating with the end users, applications and the database. It mainly helps to manage the data stored in databases.

What is API and How it works?

API stands for “Application Programming Interface.” An API is **a software intermediary that allows two applications to talk to each other**. In other words, an API is the messenger that delivers your request to the provider that you're requesting it from and then delivers the response back to you.

JSON Communication:

JSON (JavaScript Object Notation) is defined as a file format used in object-oriented programming that uses human-readable language, text, and syntax to store and communicate data objects between applications.

Thunder Client:

Thunder Client is an alternative to the famous Postman tool used for testing client APIs. The Thunder Client VS Code extension is lightweight and **lets you test APIs on the fly within the editor**. You might not want to download another tool to test the APIs you're building.

What is NodeJS?

Node. JS is **a JavaScript runtime environment that achieves low latency and high throughput by taking a “non-blocking” approach to serving**

requests. In other words, Node. js wastes no time or resources on waiting for I/O requests to return.

How to install Express in NodeJS?

Create a directory to hold your application, and make that your working directory. Use the npm init command to create a package.json file for your application.

Named Exports:

As the title suggests, named exports use the name of the function or class as their identifier. When you want to import a named component, you use the same name it was exported with. Names must be imported inside curly brackets.

Default Exports:

Default exports are created by including a `default` tag in the export. Usually, you see default exports happen at the bottom of a file, but it’s possible to define them when your component is declared. When importing a default export, you don’t use curly brackets.

Node.js Module Types:

Node.js includes three types of modules:

1. Core Modules
2. Local Modules
3. Third Party Modules

1.Core Module

Core Module	Description
http	http module includes classes, methods and events to create Node.js http server.
url	url module includes methods for URL resolution and parsing.
querystring	querystring module includes methods to deal with query string.
path	path module includes methods to deal with file paths.
fs	fs module includes classes, methods, and events to work with file I/O.
util	util module includes utility functions useful for programmers.

Node.js Local Module:

Local modules are modules created locally in your Node.js application. These modules include different functionalities of your application in separate files and folders. You can also package it and distribute it via NPM, so that Node.js community can use it. For example, if you need to connect to MongoDB and fetch data then you can create a module for it, which can be reused in your application.

NPM - Node Package Manager:

Node Package Manager (NPM) is a command line tool that installs, updates or uninstalls Node.js packages in your application. It is also an online repository for open-source Node.js packages. The node community around the world creates useful modules and publishes them as packages in this repository.

Nodemon:

Nodemon is **a command-line tool that helps with the speedy development of Node. js applications**. It monitors your project directory and automatically restarts your node application when it detects any changes. This means that you do not have to stop and restart your applications in order for your changes to take effect.

What is Package.json?

The package. json file is **the heart of any Node project**. It records important metadata about a project which is required before publishing to NPM, and also defines functional attributes of a project that npm uses to install dependencies, run scripts, and identify the entry point to our package.

NPM commands:

Install:

npm install <packagename>

There's a shorthand for installing the new packages.

npm i <packagename>

Uninstall:

npm uninstall <packagename>

Shorthand for uninstalling the new packages.

npm un <packagename>

Update:

npm update <packagename> or npm update

Shorthand for updating the packages.

npm up <packagename>

List:

```
npm list
```

Version:

```
npm view <packagename> <versions>
```

If version is not specified, default version is 'latest'

Help:

```
npm help <term>
```

Install/Update the package globally:

This command will install or update the package globally in your local system.

```
npm install -g nodemon
npm update -g nodemon
```

-g specifes global. If -g is not specified, package will be installed in local by default which can't be accessed outside the project directory.

Install a package as production/development dependency:

This command will install the package which will be available in the specified environment.

```
npm install -P <packagename> P for Production
npm install -D <packagename> D for Development
```

Init:

If you don't have package.json in a directory, and you trigger npm install moduleName at that directory path then module will be installed globally.

npm init or npm init –y

build:

npm build and npm run build are entirely different.
npm run build - This command runs the build field from the package.json scripts field.
npm build - It is an internal command. If you run it you'll get: npm WARN build npm build called with no arguments.

Start:

npm start

Stop:

npm stop <filename>

Sync and Async programming:

Synchronous tasks happen in order—you must complete the current task before moving on to the next. Asynchronous tasks are executed in any order or even at once.

What is CRUD?

CRUD is the acronym for **CREATE, READ, UPDATE and DELETE**. These terms describe the four essential operations for creating and managing persistent data elements, mainly in relational and NoSQL databases.

Route handling:

As a generic term, a route handler is code that is looking for a request to a specific incoming URL such as /login and often a specific HTTP verb such as POST and has specific code for handling that precise URL and verb.

Route controllers:

A Controller is **routable object which receives a single property from the Route – model** – which is the return value of the Route's model() method. The model is passed from the Route to the Controller by default using the setupController() function.

Middleware:

Middleware is software that different applications use to communicate with each other. It provides functionality to connect applications intelligently and efficiently so that you can innovate faster. Middleware acts as a bridge between diverse technologies, tools, and databases so that you can integrate them seamlessly into a single system. The single system then provides a unified service to its users.

HTTP Methods:

GET

The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.

POST

A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.

PUT

Replaces all current representations of the target resource with the uploaded content.

DELETE

Removes all current representations of the target resource given by a URI.

What is a NoSQL Data Model?

NoSQL or ‘Not Only SQL’ is a data model that starkly differs from traditional SQL expectations.

The primary difference is that NoSQL does not use a relational data modeling technique and it emphasizes flexible design. The lack of requirement for a schema makes designing a much simpler and cheaper process. That isn’t to say that you can’t use a schema altogether, but rather that schema design is very flexible.

CRUD Operations in MongoDB:

The basic methods of interacting with a MongoDB server are called CRUD operations. CRUD stands for **Create, Read, Update, and Delete**. These CRUD methods are the primary ways you will manage the data in your databases.

Hashing and Salting:

Password Hashing is a one-way process that converts a password to ciphertext using hash algorithms. A hashed password cannot be decrypted, but a hacker can try to reverse engineer it.

Password salting adds random characters before or after a password prior to hashing to obfuscate the actual password.

Mongoose library:

Mongoose is **an Object Data Modeling (ODM) library for MongoDB and Node. js**. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

What is Schema and Model?

A schema is a collection of database objects, including tables, views, indexes, and synonyms. There is a variety of ways of arranging schema objects in the schema models designed for data warehousing. One data warehouse schema model is a star schema.

MongoDB Datatypes:

- **String** − This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
- **Integer** − This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- **Boolean** − This type is used to store a boolean (true/ false) value.
- **Double** − This type is used to store floating point values.
- **Min/ Max keys** − This type is used to compare a value against the lowest and highest BSON elements.
- **Arrays** − This type is used to store arrays or list or multiple values into one key.
- **Timestamp** − ctimestamp. This can be handy for recording when a document has been modified or added.
- **Object** − This datatype is used for embedded documents.
- **Null** − This type is used to store a Null value.

- **Symbol** − This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.
- **Date** − This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
- **Object ID** − This datatype is used to store the document’s ID.
- **Binary data** − This datatype is used to store binary data.
- **Code** − This datatype is used to store JavaScript code into the document.
- **Regular expression** − This datatype is used to store regular expression.