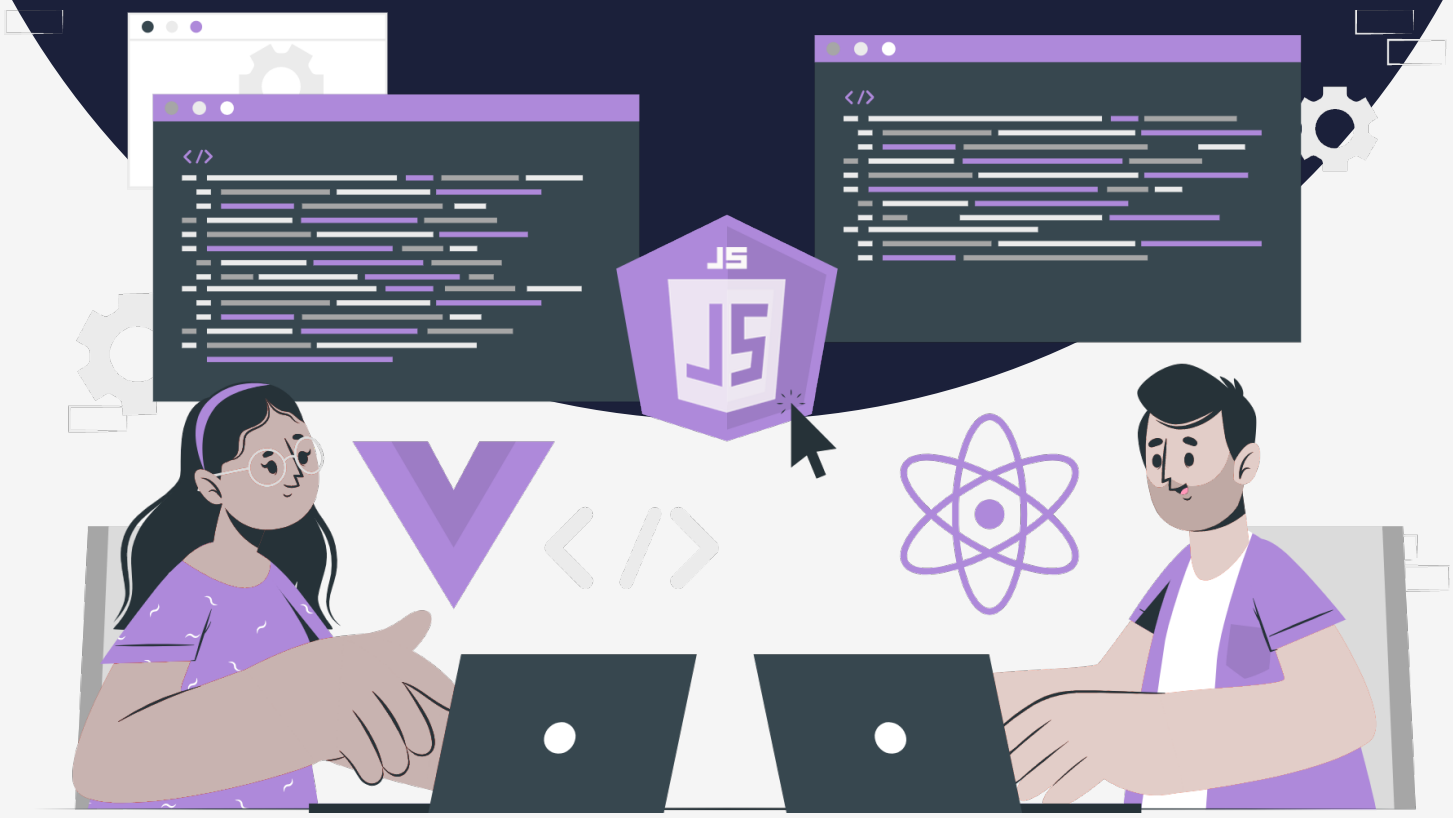
Lesson:

Intro to Node JS (Runtime, V8, Libuv)



PW SKILLS

Topics Covered :

1. What is Node JS?
2. Features of Node JS.
3. Applications of Node JS
4. How to install Node JS?
5. What is Runtime?
6. Runtime in JavaScript.
7. How does JavaScript Runtime work?
8. What is V8?
9. Key features of V8.
10. What is Libuv?
11. Key features of Libuv

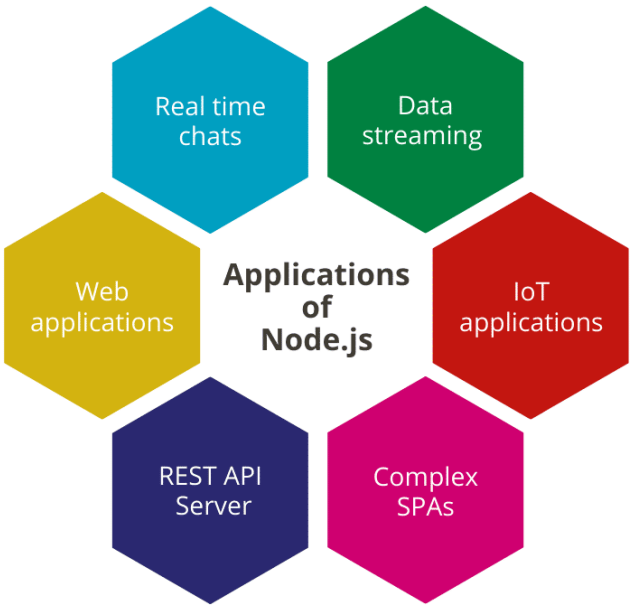
What is Node JS?

Node.js is an open-source, cross-platform JavaScript runtime environment that enables developers to run JavaScript code on the server side. It allows developers to create concurrent connections and real-time applications such as chat, gaming, and other types of applications needing real-time processing. It also has a large and active developer community that supports and maintains a vast ecosystem of open-source libraries and modules, making it easy to add functionality and integrate with other technologies.

Features of Node JS

1. **Easy:** Node.js is quite easy to start with. It's a go-to choice for web development beginners. With a lot of tutorials and a large community getting started is very easy.
2. **Scalable:** Node.js is a single-threaded program, which means that it can only handle one task at a time. However, it can handle a large number of connections at the same time without any delay. This is what makes Node.js very scalable. Because it can handle many connections simultaneously, it can handle a large number of requests and provide a fast response time, making it a great option for building scalable applications that can handle a lot of traffic.
3. **Speed:** Non-blocking thread execution means that Node.js can run multiple tasks at the same time without having to wait for one task to finish before starting another. This makes Node.js faster and more efficient because it can work on multiple tasks simultaneously, rather than having to wait for one task to finish before starting another. This allows Node.js to handle a large number of tasks with fewer resources and less delay.
4. **Packages:** A vast set of open-source Node.js packages is available that can simplify your work. There are more than one million packages in the NPM ecosystem today.
5. **Multi-platform:** Cross-platform support allows you to create SaaS websites, desktop apps, and even mobile apps, all using Node.js.
6. **Maintainable:** Node.js is an easy choice for developers since both the frontend and backend can be managed with JavaScript as a single language.

Applications of Node JS:



How to install Node JS

To install Node.js, you can follow these steps:

1. Go to the official Node.js website **(https://nodejs.org/)**
2. Download the installer for your operating system (Windows, macOS, or Linux)
3. Run the installer and follow the prompts to install Node.js on your system
4. Verify that Node.js is installed by opening a terminal or command prompt and typing "node -v". This should display the version of Node.js that you have installed.

Alternatively, you can use a package manager like apt or brew to install Node.js on Linux or macOS. You can refer to the Node.js official website for more information.

What is Runtime?
Runtime is a piece of code that implements portions of a programming language's execution model. Doing this, allows the program to interact with the computing resources it needs to work. Runtimes are often integral parts of the programming language and don't need to be installed separately.

Runtime is also when a program is running. That is, when you start a program running on a computer, it is a runtime for that program. In some programming languages, certain reusable programs or "routines" are built and packaged as a "runtime library." These routines can be linked to and used by any program when it is running.

Runtime in JavaScript.
In JavaScript, runtime refers to the period during which a JavaScript program is executed by a JavaScript engine. When a JavaScript program is loaded into a web page or runs using a command-line tool, the JavaScript engine starts to execute the code. The JavaScript engine follows a set of rules, known as the JavaScript specification, to interpret and execute the code.

During runtime, the JavaScript engine creates an execution context, a data structure containing the variables, functions, and other resources needed to execute the code. The engine then starts to execute the code, line by line, in the order it appears in the program.

JavaScript is a single-threaded language, which means that only one task can be executed at a time. However, JavaScript provides mechanisms such as callbacks and promises, to handle asynchronous operations and avoid blocking the execution of the code.

It's worth noting that JavaScript's runtime is different from other languages' runtime as when it is used for developing web applications, it runs on the browser on client side and for server side programming , it is run with Node.js. So the environment and the runtime are different.

How does JavaScript Runtime work?

JavaScript runtime works by interpreting and executing JavaScript code. When a JavaScript program is loaded into a web page or runs using a command-line tool, the JavaScript engine starts to execute the code. The engine follows a set of rules, known as the JavaScript specification, to interpret and execute the code.

The JavaScript engine creates an execution context for each function call and a global execution context for the entire program. The execution context contains the variables, functions, and other resources needed to execute the code. The engine then starts to execute the code, line by line, in the order it appears in the program.

JavaScript is a single-threaded language, which means that only one task can be executed at a time. However, JavaScript provides mechanisms such as callbacks and promises, to handle asynchronous operations and avoid blocking the execution of the code.

The JavaScript engine also includes a garbage collector, which constantly monitors the heap and frees up memory that is no longer being used. This is done in a way that does not interrupt the program execution.

JavaScript runtime in a browser is executed in a sandboxed environment, which means that it is limited in the resources it can access and what it can do. This is to protect the user's computer from malicious code. For example, JavaScript code running in a browser cannot access the user's file system or make arbitrary network connections.

JavaScript runtime on the server side, with Node.js, has more freedom and can access the file system, network resources, and more. The runtime is also different as it uses the V8 JavaScript engine developed by Google instead of the browser's JavaScript engine.

In summary, JavaScript runtime works by interpreting and executing JavaScript code in the order it appears. The JavaScript engine creates execution contexts, manages the execution flow, and handles memory management through garbage collection. The runtime also has some limitations in a browser environment to protect the user's computer.

What is V8?

V8 is an open-source JavaScript engine developed by Google which is used in the Google Chrome browser and also used in the runtime environment for Node JS. It's responsible for executing JavaScript code within the browser, parsing and compiling JavaScript code into machine code for faster execution.

V8 is known for its high performance, scalability, and compatibility with modern JavaScript features and APIs. The engine is implemented in C++ and is designed to handle complex, large-scale JavaScript applications, making it ideal for use in high-performance web browsers and server-side JavaScript environments.

The source code for V8 is freely available, allowing developers to contribute to its development and use it in their own projects. The V8 project has been actively maintained and developed by Google since its creation, and it continues to be a critical component of the Google Chrome browser and other web-based applications.

Key features of V8

- **Written in C++:** V8 is implemented in C++, making it fast and efficient. The C++ code is compiled into machine code, which can be executed directly by the computer's hardware, resulting in the fast and efficient execution of JavaScript code.
- **Dynamic Optimization:** V8 uses dynamic optimization techniques to continuously optimize the execution of JavaScript code, making it faster over time. The engine uses a Just-In-Time (JIT) compiler that generates optimized machine code for the code that is executed most frequently.
- **Heap Allocation:** V8 uses a compact, high-performance heap to allocate memory for JavaScript objects and data. The heap is designed to minimize fragmentation and maximize available memory, allowing V8 to handle large and complex JavaScript applications with ease.
- **Garbage Collection:** V8 uses a garbage collector to automatically reclaim memory that is no longer in use. This helps prevent memory leaks and improves the stability of JavaScript applications.
- **Modern JavaScript Features:** V8 supports modern JavaScript features, including ECMAScript 6 and later, as well as various APIs, such as Web Assembly, the Document Object Model (DOM), and more.
- **Open-Source:** V8 is an open-source project, with the source code available for anyone to use, modify, and contribute to. This makes it an attractive option for developers building web-based applications, as it provides access to a powerful and well-maintained JavaScript engine.

What is Libuv?

libuv is a cross-platform library for asynchronous I/O, developed for use in Node.js. It provides a uniform API for working with asynchronous I/O operations, such as file system operations, network communication, and timers.

Libuv abstracts away the differences between various operating systems and provides a consistent, high-level API for performing asynchronous I/O operations. This allows developers to write asynchronous code that works seamlessly across different platforms, without having to worry about the underlying operating system details.

In addition to its core I/O functionality, libuv also provides other features, such as thread pooling, process management, and signaling. These features make it a versatile and powerful library for building scalable and efficient applications.

Libuv is an open-source project and is used by a number of popular projects, including Node.js, and several other applications and frameworks. The library is actively maintained and has a large and active community of contributors and users, making it a well-supported and widely used tool for asynchronous I/O.

Key features of Libuv

- **Cross-Platform:** libuv abstracts away the differences between various operating systems and provides a consistent, high-level API for performing asynchronous I/O operations. This allows developers to write asynchronous code that works seamlessly across different platforms, without having to worry about the underlying operating system details.

- **Asynchronous I/O:** The core of libuv is its support for asynchronous I/O. The library provides APIs for working with file system operations, network communication, and other I/O operations that can take time to complete. These APIs allow developers to perform I/O operations in an asynchronous manner, which can improve the performance and scalability of their applications.

- **Event Loop:** libuv provides an event loop that allows developers to write asynchronous code in a simple and straightforward manner. The event loop listens for events, such as the completion of an I/O operation, and dispatches a callback function to handle the event. This allows developers to write code that is responsive and efficient, without having to deal with complex multi-threading or blocking I/O operations.

- **Thread Pool:** libuv provides a thread pool that allows developers to perform blocking I/O operations in a separate thread, freeing up the main thread to continue processing other events. This can improve the performance and scalability of applications, particularly when dealing with large amounts of data.

- **Process Management:** libuv provides APIs for working with processes, including process creation, process control, and process signals. This makes it easy to build multi-process applications, such as servers, that can take advantage of multiple cores or processors.

- **Open-Source:** libuv is an open-source project and is freely available for use. The source code is available on GitHub, and the library is actively maintained by a large and active community of contributors and users.