* All the browsers use JavaScript engines to run these JavaScript codes.
* In order to execute a JavaScript program outside the browser environment, **Ryan Dahl** created a run-time environment called **Node.js**.
* Thus,**Node.js** can be used as the server-side technology as well.

Let us learn about the **features**of Node.js.

**Features of Node.js:**

**1. Reduced resources**:

Using Node.js as the backend technology makes the application single language dependent (**One Code Base**) i.e. JavaScript is used in both client-side as well as server-side. This helps in -

* Building faster application with less number of Human resources
* Building an application with fewer lines of code and less number of files

**2. Library support**:

NPM **(Node Package Manager)** is a default package manager provided by the Node.js. NPM offers the worlds largest software registry. You can also [explore other libraries](https://www.npmjs.com/).

**3. Asynchronous and Event-driven**:

The servers which are created using Node.js handles all the request asynchronously, which means it will never wait for a return value from the APIs

**4. Very Fast**:

Node.js is a run-time-environment which helps us code JavaScript, which is an interpreted language. Since Node.js is built on Google Chrome's V8 JavaScript Engine, it helps in executing the JavaScript code faster.

**5. Single threaded but highly scalable**:

A traditional web server creates limited threads for handling the incoming request. The usage of **Event handling mechanism**in Node.Js web server sends a response in a non-blocking way. Hence Node server can handle a large number of the request than the commonly used traditional servers.

To know more on non-blocking, refer [Node docs](https://nodejs.org/en/docs/guides/blocking-vs-non-blocking/).

**6. No buffering**:

Node.js APIs' are asynchronous, hence they do not buffer the entire stream of data before processing it. These APIs' send the incoming stream of data in chunks for further processing.

**7. Streaming of data is easy**:

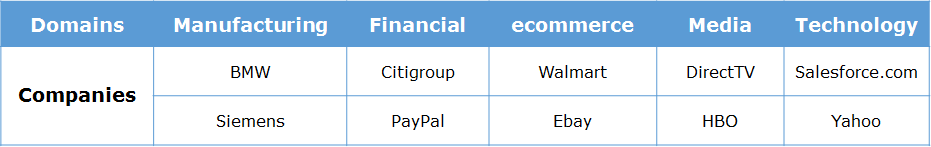
Node has built-in Stream API available, which can stream the data very fast. This API is useful in applications like a Twitter stream, video streaming etc.

**8. Wide client side and database connectivity :**

Node.js has absolutely no dependencies and also jells perfectly with any possible client-side technology like **Angular**, **Reactjs** etc. and any database like **MySQL** or **MongoDB**.

Because of all these features Node.js has grabbed attention of various organisations across different domains.

Node.js is used in applications developed for a wide range of domains. The below table lists out few domains and the companies that have migrated to Node.js.



The below table lists the reasons for moving from current technology to Node.js technology.

# ****Node.js :****

* Is an open-source server side run-time environment built on **Chrome's V8 JavaScript engine**.
* Has a model which is event-driven and non-blocking, which makes it effective and lightweight

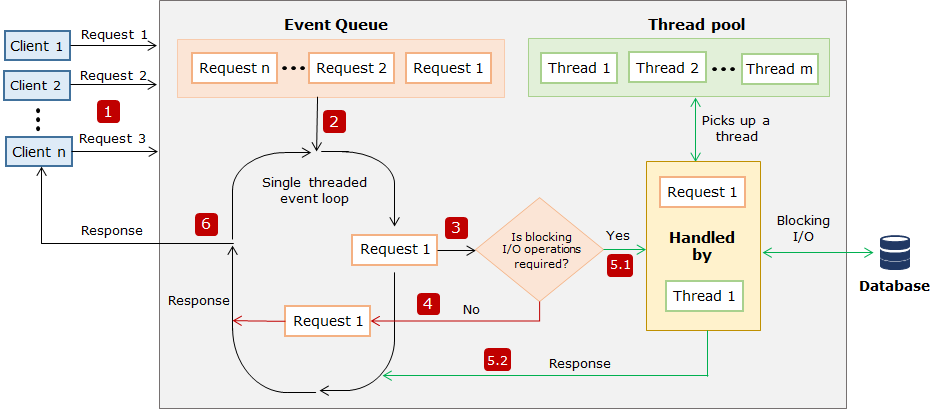
# ****Chrome's V8 JavaScript engine :****

* V8 is Google's open source high-performance JavaScript engine, written in C++ and used in Google Chrome, designed to improve the performance of the JavaScript execution in browsers.
* At the time of execution, it converts the JavaScript code into machine code using **Just-In-Time (JIT) Compiler**, to achieve high speed. Therefore byte code or intermediate code will not be created

Node.js is said to be highly scalable, because it handles the client request using **Single threaded model with event loop.**

Let us now take a look at this model structure.

Node.js environment is created based on **Single Threaded with Event Loop Model** which is built using JavaScript's callback mechanism.



**Single threaded event loop model processing steps:**

**Step 1:**   Assume **multiple** clients, send a request to access the web application **concurrently**from a **web server.** Node.js web server receives those requests from clients and places them into a queue known as **"Event Queue"**. The Node.js web server internally maintains a **limited thread pool** to provide service to the client. Let us assume **'**m**'** number of threads can be created and maintained.

**Step 2:**   The Node.js web server internally has a component, known as **"Event Loop"**. It receives a request and process them in an indefinite loop. But the Event loop component uses a **"Single Thread"** to process all the request.

**Step 3:**Event Loop component checks for any client request that is placed in Event Queue. If no requests are present, then it waits for incoming requests. If the requests are present, then it picks up one client request from Event Queue and starts processing that request.

**Step 4:**   If that client's request has no blocking I/O operations involved, then the request is processed till completion and the response is sent back to the client.

**Step 5.1:**If the client request requires some blocking I/O operations like file operations, database interactions, any external services then it checks the availability of threads from internal thread pool.

**Step 5.2:**One thread is assigned from the internal pool of threads and assigned to the client request. That thread is responsible for taking that request, processing it and performing blocking I/O operations.

**Step 6:** Once the request is processed, it prepares the response and sends it back to the Event Loop, which in turn, sends that response to the corresponding client.

Let us get started by coding in node Js

Let us test if the Node has been properly installed and configured in our system. To verify,

a. Open Visual Studio code IDE and press [Ctrl + `] keys to open the integrated console.

b. Type the following command.

1. node -v

If Node.js is installed correctly, we will observe the Node version in our terminal

Now that the required software is installed, let us go ahead and create our first node application.

**Note:** At the time of this course creation latest stable version for the nodejs is 12.18.3, you can always visit [nodejs.org](https://nodejs.org/en/) to get the stable and current version of nodejs.

**Highlights:**

* Creating a javascript file
* Executing a javascript file

Now As we know that Node environment is installed in our machine, let us create our first Node program.

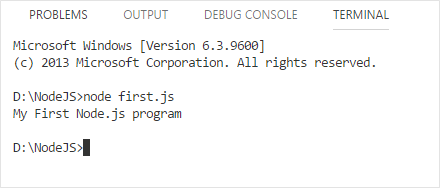
**Step 1:** Create a new JavaScript file, **first.js** and type the below code to be executed inside the file.

1. console.log("My First Node.js program");

**Step 2:** Execute the file using the node command.

1. node first.js

**Step 3:** After successful interpretation of the code, we can see the output in the terminal.

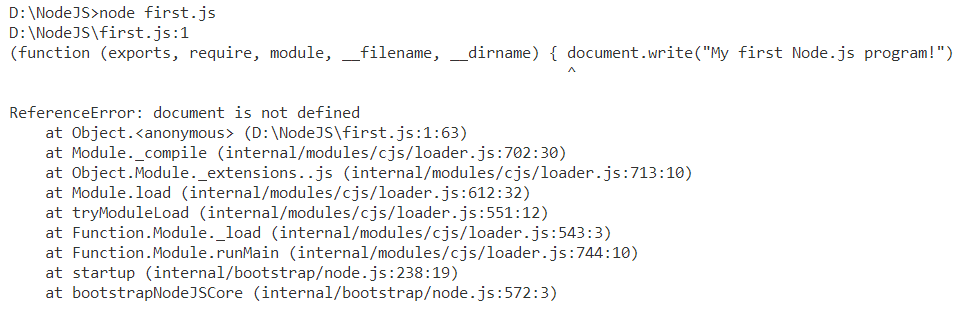


Let us see what happens when the JavaScript file which contain code for browser interaction is executed through Node.js

Let us make a small modification in our previous code.

1. document.write("My first Node.js program!");

We have used the **document.write()** function to display an output in the browser. But while interpreting the same in Node, we encounter an error as shown.



So any JavaScript file which doesn't contain codes for browser interaction will execute successfully.

# Highlights:

* Creating a Node application
* Executing first Node application

The below steps shows how to execute a **Node.js**application.

**Step 1:** Open VS Code and create a new JavaScript file named ***firstApp.js*** and type the following code.

1. let myFunction = (message) => {
2. console.log(message);
3. }
4. myFunction("I have begun my journey through Node.js");

**Step 2:** In the integrated terminal of VS code IDE, execute the file using the node command.

1. node firstApp.js

**Step 3:** Observe the output in the terminal.

https://academy.onwingspan.com/common-content-store/Shared/Shared/Public/lex_11742273089797390000_shared/web-hosted/assets/noderunres.PNG