ACROPOLIS INSTITUTE OF TECHNOLOGY & RESEARCH, INDORE

ACROPOLIS
Enlightening Wisdom

Department of Computer Science & Information Technology

Name of the Student	**: Pooja Patidar**
Branch & section	**:CSIT-2**
Roll No.	**:0827CI201134**
Year	**:2020-24**

Submitted to- Prof. Taresh Ayaspure
Signature-_____________

Department of Computer Science & Information Technology

Introduction

Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a web browser. Node.js is a popular, lightweight web framework for beginners, and it is used by many big companies like Netflix and Uber.

When we typically think of JavaScript, our mind tends to go to web development. Until Node.js came along, there was really no way to run JavaScript outside of a browser. When we write a backend server and database, Node.js is a popular choice because we can run our code as a standalone application rather than something that can only be evaluated in a browser environment.

Fundamentals of Node.js

Now that we know what Node.js is, let's explore the fundamentals of this tool..

1. **Console**
2. **Buffer**
3. **File System**
4. **Event Loop**
5. **Globals**

Advantages of NodeJS: Here are the benefits of using Node.js

1. **Easy Scalability:** Developers prefer to use Node.js because it is easily scaling the application in both horizontal and vertical directions. We can also add extra resources during the scalability of the application.
2. **Real-time web apps:** If you are building a web app you can also use PHP, and it will take the same amount of time when you use Node.js, But if I am talking about building chat apps or gaming apps Node.js is much more preferable because of faster synchronization. Also, the event loop avoids HTTP overloaded for Node.js development.
3. **Fast Suite:** Node.js runs on the V8 engine developed by Google. Event loop in Node.js handles all asynchronous operation so Node.js acts like a fast suite and all the operations can be done quickly like reading or writing in the database, network connection, or file system
4. **Easy to learn and code:** Node.js is easy to learn and code because it uses JavaScript. If you are a front-end developer and have a good grasp of JavaScript you can easily learn and build the application on NodeJS
5. **Advantage of Caching:** It provides the caching of a single module. Whenever there is any request for the first module, it gets cached in the application memory, so you don't need to re-execute the code.

6. **Data Streaming:** In Node.js HTTP request and response are considered as two separate events. They are data stream so when you process a file at the time of loading it will reduce the overall time and will make it faster when the data is presented in the form of transmissions. It also allows you to stream audio and video files at lightning speed.
7. **Hosting:** PaaS (Platform as a Service) and Heroku are the hosting platforms for NodeJS application deployment which is easy to use without facing any issue.
8. **Corporate Support:** Most of the well-known companies like Walmart, PayPal, Microsoft, Yahoo are using NodeJS for building the applications. NodeJS uses JavaScript, so most of the companies are combining front-end and backend Teams together into a single unit.

Application of NodeJS: NodeJS should be preferred to build:

- Real-Time Chats,
- Complex Single-Page applications,
- Real-time collaboration tools,
- Streaming apps
- JSON APIs based application

1. Console:-

The **console** is a module provided by Node.js that is akin to the JavaScript console in the browser when you inspect a webpage. The console has methods that are available for us to use for debugging purposes.

- console.log(): Frequently used to log some sort of output.
- console.warn(): Explicitly delivers a warning to the console.
- console.error(): Explicitly delivers an error message to the console. You can log an error as a string or as an object. If logged as a new Error(), a traceback will be included as part of the message.
- console.trace(): Logs a traceback when an error occurs in your code. Gives line number and column number of the file that the error probably occurred.

2. Buffer

At its core, the **Buffer class** in Node.js is a temporary storage solution for file systems. Due to its low-level nature, as web developers we will rarely actually use the Buffer class directly. The main purpose of this class is to allocate memory.

Let’s take a look at a few methods that the Buffer class provides.

3. File System

The **file system (fs) module** allows us to interact with files in Node.js. There are synchronous and asynchronous methods that can be used to read or write to a file using the fs module. In contrast to using console or the Buffer class, we need to import the fs module into the file that we would like to use in order to get it to work.

4. Event Loop

Much of Node.js is built to be **event-driven**. When a user clicks on an interface or types in a form, an event is triggered to happen and then something occurs as a result. To attach a function or set of functions to a specific event is emitting an event.

These functions, called event listeners, are one part of an overall journey called the Event Loop

5. Globals

Global objects are available in every module, so they can be used without importing a specific module. The `Buffer` class, for example, class is defined as a global in Node.js. Some other common global objects are:

- The `console` object is used to print to `stdout` and `stderr`.
- Timers, such as `setImmediate`, `setInterval`, and `setTimeout`, are also globals.
- The `process` object is also global.

How to build a basic Node.js project

Let's learn how to get started with Node.js by creating a simply Node.js file.In this example, we will be setting up our computer to work as a server!

If you would like to learn how to create a Node.js app, please see Educative beginner course.

Install Node.js and NPM

First, you need to Go to the site Node.js site and download the files.

Follow the installation prompts and restart your machine for best results.

Another way you can install Node.js is to use a package manager.

Then, test that it's working by printing the version using the following command:

```
> node -v
```

You should also test npm by printing the version using the following command:

```
> npm -v
```

Create a file:-

Once you have installed Node.js properly, create a Node.js file. In this example, we have named it named “first.js”. We then add the following code and save the file on your computer like so: C:\Users\Your Name\first.js

```
var http = require('http');

http.createServer(function       (req,       res)       {
res.writeHead(200,  {'Content-Type':  'text/html'});
res.end('Hello World!'); }).listen(8080);
```

This code is essentially telling the computer to print “Hello World!” when accessed on port 8080.

Command line interface:-

Node.js files must be initiated in your computer’s “Command Line Interface” program. Navigate to the folder that contains the file “first.js”.

```
C:\Users\Your Name>_
```

Initiate your file:-

This file needs to then be initiated by Node.js. Do this by starting your command line interface, writing node first.js, and clicking enter:

```
C:\Users\Your Name>node myfirst.js
```

Awesome! Now your computer is set up as a server, so when you accesses the computer on port 8080, the “Hello World!" message will print.

To see this in real time, open your browser, and type: http://localhost:8080

Node.js Modules

In Node.js, Modules are the blocks of encapsulated code that communicates with an external application on the basis of their related functionality. Modules can be a single file or a collection of multiples files/folders. The reason programmers are heavily reliant on modules is because of their re-usability as well as the ability to break down a complex piece of code into manageable chunks.

Modules are of three types:

Core Modules

local Modules

Third-party Modules

Core Modules: Node.js has many built-in modules that are part of the platform and comes with Node.js installation. These modules can be loaded into the program by using the require function.

Syntax:

var module = require('module_name');

The require() function will return a JavaScript type depending on what the particular module returns. The following example demonstrates how to use the Node.js Http module to create a web server.

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('Welcome to this page!');
  res.end();
}).listen(3000);
```

In the above example, the require() function returns an object because the Http module returns its functionality as an object. The function http.createServer() method will be executed when someone tries to access the computer on port 3000. The res.writeHead() method is the status code where 200 means it is OK, while the second argument is an object containing the response headers.

The following list contains some of the important core modules in Node.js:

Core Modules	Description
http	creates an HTTP server in Node.js.
assert	set of assertion functions useful for testing.
fs	used to handle file system.
path	includes methods to deal with file paths.
process	provides information and control about the current Node.js process.
os	provides information about the operating system.
querystring	utility used for parsing and formatting URL query strings.
url	module provides utilities for URL resolution and parsing.

Local Modules: Unlike built-in and external modules, local modules are created locally in your Node.js application. Let's create a simple calculating module that calculates various operations. Create a calc.js file that has the following code:

Filename: calc.js

```
exports.add = function (x, y) {
    return x + y;
};

exports.sub = function (x, y) {
    return x - y;
};

exports.mult = function (x, y) {
    return x * y;
};

exports.div = function (x, y) {
    return x / y;
};
```

Since this file provides attributes to the outer world via exports, another file can use its exported functionality using the require() function.

Filename: index.js

```
var calculator = require('./calc');

var x = 50, y = 10;

console.log("Addition of 50 and 10 is "
                + calculator.add(x, y));

console.log("Subtraction of 50 and 10 is "
                + calculator.sub(x, y));

console.log("Multiplication of 50 and 10 is "
                + calculator.mult(x, y));

console.log("Division of 50 and 10 is "
                + calculator.div(x, y));
```

Step to run this program: Run **index.js** file using the following command:
node index.js

Output:

```
Addition of 50 and 10 is 60

Subtraction of 50 and 10 is 40

Multiplication of 50 and 10 is 500

Division of 50 and 10 is 5
```

Note: This module also hides functionality that is not needed outside of the module.
Third-party modules: Third-party modules are modules that are available online using the Node Package Manager(NPM). These modules can be installed in the project folder or globally. Some of the popular third-party modules are mongoose, express, angular, and react.
Example:

- npm install express
- npm install mongoose
- npm install -g @angular/cli

Conclusion

Node.js lets developers use JavaScript to write command line tools and for server-side scripting. The functionality of running scripts server-side produces dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[6] unifying webapplication development around a single programming language, rather than different languages for server-side and client-side scripts.

Reference

https://www.fortinet.com/