23. Introduction to NodeJS

Table of Content

1. Introduction to NodeJS
2. NPM Basics
3. Introduction to ExpressJS
4. API and HTTP Request Types

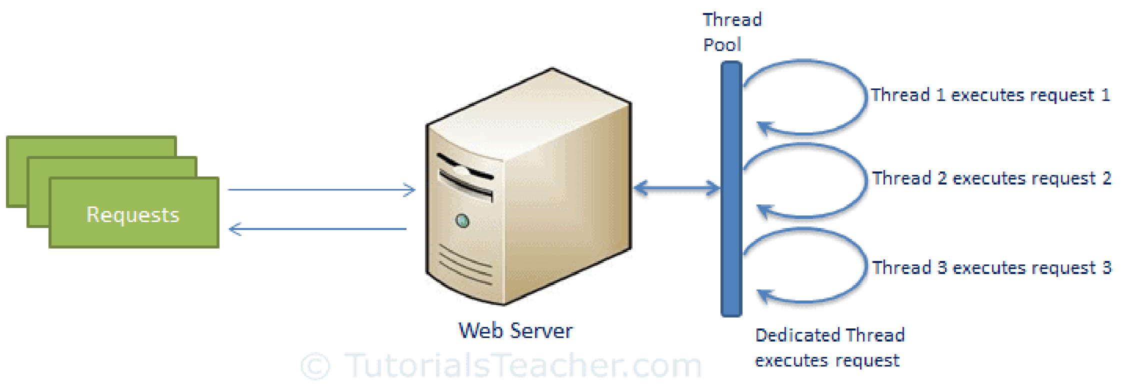
1. Introduction to NodeJS

Node.js is an open-source server-side runtime environment built on Chrome's V8 JavaScript engine. It provides an event-driven, non-blocking (asynchronous) I/O and cross-platform runtime environment for building highly scalable server-side applications using JavaScript.

Node.js can be used to build different applications such as command line, web, real-time chat, REST API server, etc.

▼ **NodeJS Process Model**

- **Traditional Web Server Model**
  In the traditional web server model, each request is handled by a dedicated thread from the thread pool. If no thread is available in the thread pool at any point of time then the request waits till the next available thread. Dedicated thread executes a particular request and does not return to the thread pool until it completes the execution and returns a response.

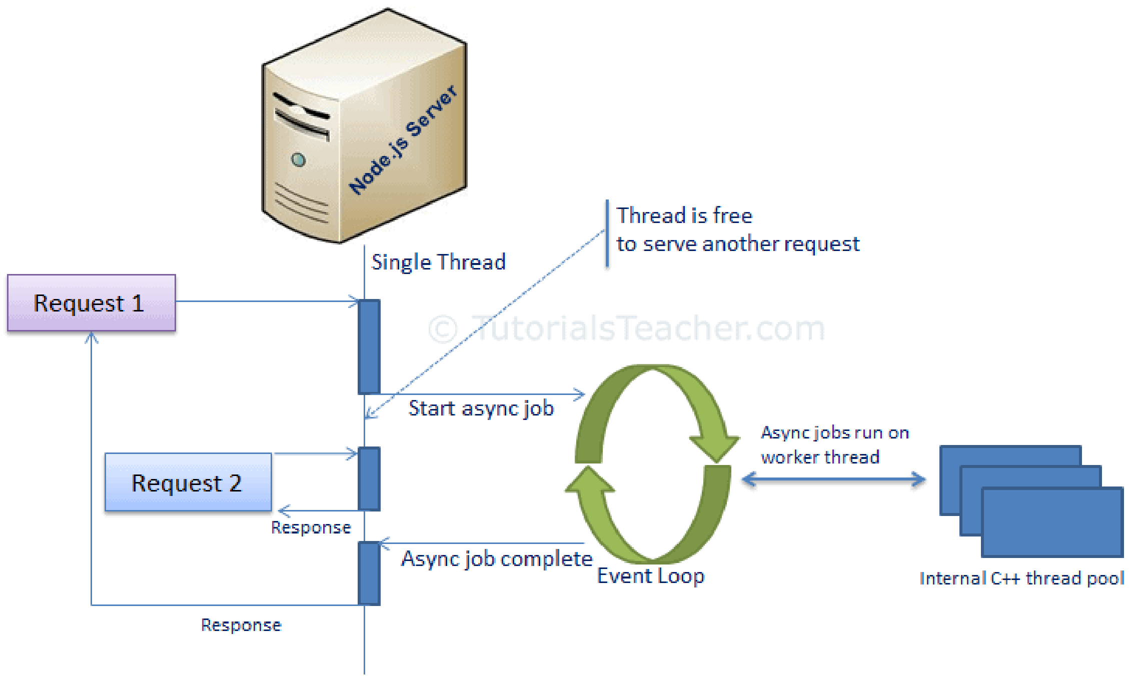

Thread Pool
Thread 1 executes request 1
Thread 2 executes request 2
Thread 3 executes request 3
Requests
Web Server
Dedicated Thread executes request
© TutorialsTeacher.com

- **Node.js Process Model**

  Node.js processes user requests differently when compared to a traditional web server model. Node.js runs in a single process and the application code runs in a single thread and thereby needs less resources than other platforms. All the user requests to your web application will be handled by a single thread and all the I/O work or long running job is performed asynchronously for a particular request. So, this single thread doesn't have to wait for the request to complete and is free to handle the next request. When asynchronous I/O work completes then it processes the request further and sends the response.

  An event loop is constantly watching for the events to be raised for an asynchronous job and executing callback function when the job completes. Internally, Node.js uses libev for the event loop which in turn uses internal C++ thread pool to provide asynchronous I/O.

  The following figure illustrates asynchronous web server model using Node.js.



▼ NodeJS Module

Each module in Node.js has its own context, so it cannot interfere with other modules or pollute the global scope. Also, each module can be placed in a separate .js file under a separate folder.

Node.js implements the CommonJS modules standard. CommonJS is a group of volunteers who define JavaScript standards for web server, desktop, and console application.

Types of Modules:

1. **Core Modules**

   Node.js is a light weight framework. The core modules include bare minimum functionalities of Node.js. These core modules are compiled into its binary distribution and load automatically when Node.js process starts. However, you need to import the core module first in order to use it in your application.

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.

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

var server = http.createServer(function(req, res){

  //write code here

});

server.listen(5000);
```

2. **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.

```
var log = {
            info: function (info) {
                console.log('Info: ' + info);
            },
            warning:function (warning) {
                console.log('Warning: ' + warning);
            },
            error:function (error) {
                console.log('Error: ' + error);
            }
    };

module.exports = log
```

```
var myLogModule = require('./Log.js');

myLogModule.info('Node.js started');
```

   The `module.exports` is a special object which is included in every JavaScript file in the Node.js application by default. The `module` is a variable that represents the current module, and `exports` is an object that will be exposed as a module. So, whatever you assign to `module.exports` will be exposed as a module.

2. NPM Basics

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.
Official website: https://www.npmjs.com

NPM is included with Node.js installation.

▼ Using NPM Commands

1. Install package: `npm install -g <package name>`
2. Update package: `npm update <package name>`
3. Uninstall package: `npm uninstall <package name>`

Use the `-D` flag to do the above changes with dev dependencies.

4. Create an empty Node project with npm: `npm -init`

3. Introduction to Express.JS

Express.js is a web application framework for Node.js. It provides various features that make web application development fast and easy which otherwise takes more time using only Node.js.

Advantages of Express.js

1. Makes Node.js web application development fast and easy.
2. Easy to configure and customize.
3. Allows you to define routes of your application based on HTTP methods and URLs.
4. Includes various middleware modules which you can use to perform additional tasks on request and response.

5. Easy to integrate with different template engines like Jade, Vash, EJS etc.
6. Allows you to define an error handling middleware.
7. Easy to serve static files and resources of your application.
8. Allows you to create REST API server.
9. Easy to connect with databases such as MongoDB, Redis, MySQL

▼ install express

`npm install -g express`

```
const express=require('express');
const app=express();

app.get('/',function(req,res)
{
res.send('Hello World!');
});

app.listen(3000,function() {});
```

4. API and HTTP Request Types

An API (Application Programming Interface), as the name suggests, is an interface that defines the interaction between different software components. Web APIs define what requests can be made to a component (for example, an endpoint to get a list of books), how to make them (for example, a GET request), and their expected responses.

There are a few types of HTTP methods that we need to grasp before building a REST API. These are the methods that correspond to the CRUD tasks:

- `POST`: Used to submit data, typically used to *create* new entities or edit already existing entities.
- `GET`: Used to request data from the server, typically used to *read* data.

- `PUT`: Used to completely replace the resource with the submitted resource, typically used to *update* data.
- `DELETE`: Used to *delete* an entity from the server.

Assignments:

1. Build a GET API service to return a random number between 1 to 6
2. Build an API service to do POST request with the following operations:
    a. sum of n numbers : POST `/sum`
    b. average of n numbers : POST `/average`