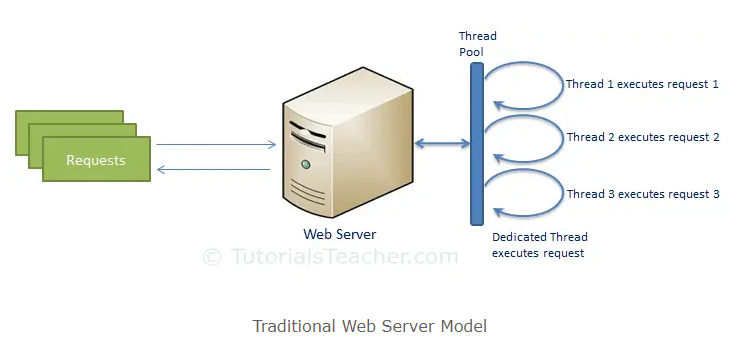
**Node JS**

**1.What is Node JS?**

Node JS is a run time environment. It uses JavaScript on server. It runs on V8 JavaScript engine.

**2.What is process model or Traditional server vs Node JS process server?**

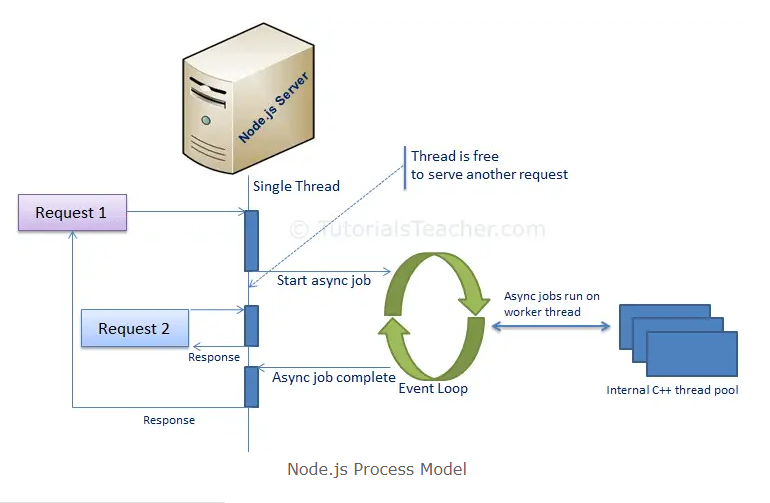
**Traditional Web Server Model**

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 thread pool until it completes the execution and returns a response.  


**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.



Node.js process model increases the performance and scalability with a few caveats. Node.js is not fit for an application which performs CPU-intensive operations like image processing or other heavy computation work because it takes time to process a request and thereby blocks the single thread.

**3.Explain event loop. (Call Stack, APIs, call back que, micro task que)**

The event loop is the secret behind JavaScript asynchronous programming. JS executes all operations on a single thread, but using a few smart data structures it gives us the illusion of multi-threading.

The event loop is what allows Node JS to perform non-blocking I/O operations – despite the fact that JavaScript is single thread application – by offloading operations to the system kernel whenever possible.

**4.What are Globals?**

Globals are predefined objects in JavaScript which have set of functions like setInterval, clearInterval, setTimeout, clearTimeout, setImmediate & clearImmediate.

**5.What is chaining?**

Chaining is mechanism whereby the output of one stream is connected to another stream as input to create a chain of multiple operations.

**6.When to use Node JS and when not to?**

Simple APIs, chat apps, small calculations but not CPU dedicated apps and heavy calculations.

**7.Is Node JS single threaded? How can you use it like multithread?**

Node JS is a single threaded application. Using process, we can use it like multithreaded.

**8.How Node JS achieve asynchronous programming?**

Using promise, asyn-await, callbacks, event listener & event loop.

**9.What is middleware in Node JS?**

Middleware functions are functions that have access to the request object(req), the response object(res) and the next middleware function in the application’s request-response cycle.

Middleware can process request objects multiple time before the server works for the request.

It can be used for logging, authentication & authorization

**10.What are ES6 features?**

Let, const, arrow function, the spread operator, for/of. Map, objects, set objects.

Class. Promise, Symbol, Default parameter, , function rest parameter,

String.includes(), string.startswith(), string.endswith()

Array.from(), array.keys(), array.find(), array.findIndex();

**11.What is callback?**

A callback is a function passed as an argument to another function. This technique allows a function to call another function. A callback function can run after another function has finished.

**12.What is callback hell? How to save it.**

Nesting of callbacks can lead to an unreadable and not easy to manageable codebase commonly known as callback hell in Node JS. Node JS callback hell can be avoided using promise and async-await.

Splitting of functions and by writing comments can also be used to avoid callback hell.

**13.Why to use Express when we can use Node JS?**

Express.JS is a small framework that works on the top of Node JS web server functionality to simplify its APIs and add helpful new features. It makes it easier to organize your application’s functionality with middleware and routing.

**14.Writ a simple code to create server in node.**

var http = require(‘http’); //import module

var server = http.createServer(function(){ // set connection

//handle request

})  
server.listen(8080) // define port

**15.Write a code accept get/post request and return hello world.**

Let app = require(‘express’).router();

App.get(‘/getDetails’,requires(‘controller.js’));

Let controller = async(req,res,next)=>{

Res.send({message:’Hello World’})

}

module.export = controller

**16.What is the difference between promise and async-await and callback?**

<https://blog.devgenius.io/promise-vs-callback-vs-async-await-ac30d1801eaa>

**Promise**:

Promise is a Javascript object that represents the eventual completion(or failure) of an asynchronous operation and its resulting value. It allows you to handle async operations in a more elegant way and avoid callback hell problem.

**Callback**:

Callbacks are a way of passing a function as an argument to another function, and it gets executed after the first function has completed. It is one of the most common ways of handling async operations in JavaScript. Callback functions are commonly used as a means of continuing processing after an asynchronous operation has completed.

**async-await**:

Async/await is a more recent addition to JavaScript, and it’s built on top of Promises. It allows you to write asynchronous code that looks and behaves like synchronous code, using the keywords “async” and “await”. The “async” keyword is used to define an async function, and the “await” keyword is used to pause the execution of an async function until a promise is resolved.

With async/await, the code inside an async function is executed in a synchronous-like way, making it more readable and easier to understand. It also helps to avoid the callback hell problem.

**17.Difference between RestAPI vs SoapAPI.**

Soap is protocol/ rest is an architectural style

Soap stands for Simple Object Access Protocol/ Rest stands for Representational State Transfer.

Soap can’t use Rest because it is a protocol/Rest can use Soap because it is a concept.

Soap strictly follows rules/ Rest does not follow any strict rule

**18.Why to use PUT for update and not POST?**

In general, the HTTP PUT method is used to update an existing resource, while the POST method is used to create a new resource. PUT requests are used to update a resource, while POST requests are used to create a new resource