Node.JS Questionnaire

What Are The Key Features Of Node.Js?

- **Asynchronous event driven IO helps concurrent request handling –** All APIs of Node.js are asynchronous. This feature means that if a Node receives a request for some Input/Output operation, it will execute that operation in the background and continue with the processing of other requests. Thus it will not wait for the response from the previous requests.
- **Fast in Code execution –** Node.js uses the V8 JavaScript Runtime engine, the one which is used by Google Chrome. Node has a wrapper over the JavaScript engine which makes the runtime engine much faster and hence processing of requests within Node.js also become faster.
- **Single Threaded but Highly Scalable –** Node.js uses a single thread model for event looping. The response from these events may or may not reach the server immediately. However, this does not block other operations. Thus making Node.js highly scalable. Traditional servers create limited threads to handle requests while Node.js creates a single thread that provides service to much larger numbers of such requests.
- **Node.js library uses JavaScript –** This is another important aspect of Node.js from the developer's point of view. The majority of developers are already well-versed in JavaScript. Hence, development in Node.js becomes easier for a developer who knows JavaScript.
- **There is an Active and vibrant community for the Node.js framework –** The active community always keeps the framework updated with the latest trends in the web development.
- **No Buffering –** Node.js applications never buffer any data. They simply output the data in chunks.

How node.js works?

Node.js works on a v8 environment, it is a virtual machine that utilizes JavaScript as its scripting language and achieves high output via non-blocking I/O and single threaded event loop.

What does event-driven programming mean?

In computer programming, event driven programming is a programming paradigm in which the flow of the program is determined by events like messages from other programs or threads. It is an application architecture technique divided into two sections 1) Event Selection 2) Event Handling

Where can we use node.js?

Node.js can be used for the following purposes

a) Web applications (especially real-time web apps)

b) Network applications

c) Distributed systems

d) General purpose applications

What is the advantage of using node.js?

a) It provides an easy way to build scalable network programs

b) Generally fast

c) Great concurrency

d) Asynchronous everything

e) Almost never blocks

What are the two types of API functions in Node.js ?

The two types of API functions in Node.js are

a) Asynchronous, non-blocking functions

b) Synchronous, blocking functions

What is control flow function?

A generic piece of code which runs in between several asynchronous function calls is known as control flow function.

Explain the steps how "Control Flow" controls the functions calls?

a) Control the order of execution

b) Collect data

c) Limit concurrency

d) Call the next step in program

Why Node.js is single threaded?

For async processing, Node.js was created explicitly as an experiment. It is believed that more performance and scalability can be achieved by doing async processing on a single thread under typical web loads than the typical thread based implementation.

Does node run on windows?

Yes – it does. Download the MSI installer from http://nodejs.org/download/

Can you access DOM in node?

No, you cannot access DOM in node.

Using the event loop what are the tasks that should be done asynchronously?

a) I/O operations

b) Heavy computation

c) Anything requiring blocking

Why node.js is quickly gaining attention from JAVA programmers?

Node.js is quickly gaining attention as it is a loop based server for JavaScript. Node.js gives user the ability to write the JavaScript on the server, which has access to things like HTTP stack, file I/O, TCP and databases.

What are the two arguments that async.queue takes?

The two arguments that async.queue takes

a) Task function

b) Concurrency value

What is an event loop in Node.js ?

To process and handle external events and to convert them into callback invocations an event loop is used. So, at I/O calls, node.js can switch from one request to another .

Mention the steps by which you can async in Node.js?

By following steps you can async Node.js

a) First class functions

b) Function composition

c) Callback Counters

d) Event loops

What are the pros and cons of Node.js?

Pros:

a) If your application does not have any CPU intensive computation, you can build it in Javascript top to bottom, even down to the database level if you use JSON storage object DB like MongoDB.

b) Crawlers receive a full-rendered HTML response, which is far more SEO friendly rather than a single page application or a websockets app run on top of Node.js.

Cons:

a) Any intensive CPU computation will block node.js responsiveness, so a threaded platform is a better approach.

b) Using relational database with Node.js is considered less favourable

How Node.js overcomes the problem of blocking of I/O operations?

Node.js solves this problem by putting the event based model at its core, using an event loop

What are the Challenges with Node.js ?

Emphasizing on the technical side, it's a bit of challenge in Node.js to have one process with one thread to scale up on multi core server.

What does it mean "non-blocking" in node.js?

In node.js "non-blocking" means that its IO is non-blocking. Node uses "libuv" to handle its IO in a platform-agnostic way. On windows, it uses completion ports for unix it uses epoll or kqueue etc. So, it makes a non-blocking request and upon a request, it queues it within the event loop which call the JavaScript 'callback' on the main JavaScript thread.

What is the command that is used in node.js to import external libraries?

Command “require” is used for importing external libraries, for example, “var http=require (“http”)”. This will load the http library and the single exported object through the http variable.

Mention the framework most commonly used in node.js?

“Express” is the most common framework used in node.js

What is ‘Callback’ in node.js?

Callback function is used in node.js to deal with multiple requests made to the server. Like if you have a large file which is going to take a long time for a server to read and if you don’t want a server to get engage in reading that large file while dealing with other requests, call back function is used. Call back function allows the server to deal with pending request first and call a function when it is finished.

When To Not Use Node.Js?

Node.js can be used for a variety of applications. But it is a single threaded framework, so we should not use it for cases where the application requires long processing time. If the server is doing some calculation, it won’t be able to process any other requests. Hence, Node.js is best when processing needs less dedicated CPU time.

What IDEs Can You Use For Node.Js Development?

Here is the list of most commonly used IDEs for developing node.js applications.

Cloud9.

It is a free, cloud-based IDE that supports, application development, using popular programming languages like Node.js, PHP, C++, Meteor and more. It provides a powerful online code editor that enables a developer to write, run and debug the app code.

JetBrains WebStorm.

WebStorm is a lightweight yet powerful JavaScript IDE, perfectly equipped for doing client-side and server-side development using Node.js. The IDE provides features like intelligent code completion, navigation, automated and safe refactorings. Additionally, we can use the debugger, VCS, terminal and other tools present in the IDE.

JetBrains InteliJ IDEA.

It is a robust IDE that supports web application development using mainstream technologies like Node.js, Angular.js, JavaScript, HTML5 and more. To enable the IDE that can do Node.js development we have to install a Node.js plugin. It provides features, including syntax highlighting, code assistance, code completion and more. We can even run and debug Node.js apps and see the results right in the IDE. It's JavaScript debugger offers conditional breakpoints, expression evaluation, and other features.

Komodo IDE.

It is a cross-platform IDE that supports development in main programming languages, like Node.js, Ruby, PHP, JavaScript and more. It offers a variety of features, including syntax highlighting, keyboard shortcuts, collapsible Pane, workspace, auto indenting, code folding and code preview using built-in browser.

Eclipse.

It is a popular cloud-based IDE for web development using Java, PHP, C++ and more. You can easily avail the features of Eclipse IDE using the Node.js plug-in, which is **<nodeclipse>**.

Atom.

It is an open source application built with the integration of HTML, JavaScript, CSS, and Node.js. It works on top of Electron framework to develop cross-platform apps using web technologies. Atom comes pre-installed with four UI and eight syntax themes in both dark and light colors. We can also install themes created by the Atom community or create our own if required.

Is Node.Js Entirely Based On A Single-Thread?

Yes, it's true that Node.js processes all requests on a single thread. But it's just a part of the theory behind Node.js design. In fact, more than the single thread mechanism, it makes use of events and callbacks to handle a large no. of requests asynchronously.

Moreover, Node.js has an optimized design which utilizes both JavaScript and C++ to guarantee maximum performance. JavaScript executes at the server-side by Google Chrome v8 engine. And the C++ lib UV library takes care of the non-sequential I/O via background workers.

To explain it practically, let's assume there are 100s of requests lined up in Node.js queue. As per design, the main thread of Node.js event loop will receive all of them and forwards to background workers for execution. Once the workers finish processing requests, the registered callbacks get notified on event loop thread to pass the result back to the user.

How To Get Post Data In Node.Js?

Following is the code snippet to fetch Post Data using Node.js.

```
app.use(express.bodyParser());
app.post('/', function(request, response){
console.log(request.body.user);
});
```

How To Make Post Request In Node.Js?

Answer.

```
var request = require('request');
request.post(
'http://www.example.com/action',
{ form: { key: 'value' } },
function (error, response, body) {
if (!error && response.statusCode == 200) {
console.log(body)
}
}
);
```

What Is Callback Hell?

Initially, you may praise Callback after learning about it. Callback hell is heavily nested callbacks which make the code unreadable and difficult to maintain.

Let’s see the following code example.

```
downloadPhoto('http://coolcats.com/cat.gif', displayPhoto)
function displayPhoto (error, photo) {
if (error) console.error('Download error!', error)
else console.log('Download finished', photo)
}
console.log('Download started')
```

In this scenario, Node.js first declares the “displayPhoto” function. After that, it calls the “downloadPhoto” function and pass the “displayPhoto” function as its callback. Finally, the code prints ‘Download started’ on the console. The “displayPhoto” will be executed only after “downloadPhoto” completes the execution of all its tasks.

How To Avoid Callback Hell In Node.Js?

Answer.

Node.js internally uses a single-threaded event loop to process queued events. But this approach may lead to blocking the entire process if there is a task running longer than expected.

Node.js addresses this problem by incorporating callbacks also known as higher-order functions. So whenever a long-running process finishes its execution, it triggers the callback associated. With this approach, it can allow the code execution to continue past the long-running task.

However, the above solution looks extremely promising. But sometimes, it could lead to complex and unreadable code. More the no. of callbacks, longer the chain of returning callbacks would be. Just see the below example.

With such an unprecedented complexity, it’s hard to debug the code and can cause you a whole lot of time. There are four solutions which can address the callback hell problem.

1. Make Your Program Modular.

It proposes to split the logic into smaller modules. And then join them together from the main module to achieve the desired result.

2. Use Async Mechanism.

It is a widely used Node.js module which provides a sequential flow of execution.

The async module has <async.waterfall> API which passes data from one operation to other using the next callback.

Another async API <async.map> allows iterating over a list of items in parallel and calls back with another list of results.

With the async approach, the caller's callback gets called only once. The caller here is the main method using the async module.

3. Use Promises Mechanism.

Promises give an alternate way to write async code. They either return the result of execution or the error/exception. Implementing promises requires the use of <.then()> function which waits for the promise object to return. It takes two optional arguments, both functions. Depending on the state of the promise only one of them will get called. The first function call proceeds if the promise gets fulfilled. However, if the promise gets rejected, then the second function will get called.

4. Use Generators.

Generators are lightweight routines, they make a function wait and resume via the yield keyword. Generator functions uses a special syntax <function* ()>. They can also suspend and resume asynchronous operations using constructs such as promises or <thunks> and turn a synchronous code into asynchronous.

Can You Create HTTP Server In Nodejs, Explain The Code Used For It?

Yes, we can create HTTP Server in Node.js. We can use the **<http-server>** command to do so.

Following is the sample code.

```
var http = require('http');
var requestListener = function (request, response) {
response.writeHead(200, {'Content-Type': 'text/plain'});
response.end('Welcome Viewers\n');
}
var server = http.createServer(requestListener);
server.listen(8080); // The port where you want to start with.
```

What Are Globals In Node.Js?

There are three keywords in Node.js which constitute as Globals. These are Global, Process, and Buffer.

Global.

The Global keyword represents the global namespace object. It acts as a container for all other <global> objects. If we type <console.log(global)>, it'll print out all of them.

An important point to note about the global objects is that not all of them are in the global scope, some of them fall in the module scope. So, it's wise to declare them without using the var keyword or add them to Global object.

Variables declared using the var keyword become local to the module whereas those declared without it get subscribed to the global object.

Process.

It is also one of the global objects but includes additional functionality to turn a synchronous function into an async callback. There is no boundation to access it from anywhere in the code. It is the instance of the EventEmitter class. And each node application object is an instance of the Process object.

It primarily gives back the information about the application or the environment.

- **<process.execPath>** – to get the execution path of the Node app.
- **<process.Version> –** to get the Node version currently running.
- **<process.platform> –** to get the server platform.

Some of the other useful Process methods are as follows.

- **<process.memoryUsage> –** To know the memory used by Node application.
- **<process.NextTick> –** To attach a callback function that will get called during the next loop. It can cause a delay in executing a function.

Buffer.

The Buffer is a class in Node.js to handle binary data. It is similar to a list of integers but stores as a raw memory outside the V8 heap.

We can convert JavaScript string objects into Buffers. But it requires mentioning the encoding type explicitly.

- **<ascii> –** Specifies 7-bit ASCII data.
- **<utf8> –** Represents multibyte encoded Unicode char set.
- **<utf16le> –** Indicates 2 or 4 bytes, little endian encoded Unicode chars.
- **<base64> –** Used for Base64 string encoding.
- **<hex> –** Encodes each byte as two hexadecimal chars.

Here is the syntax to use the Buffer class.

```
> var buffer = new Buffer(string, [encoding]);
```

The above command will allocate a new buffer holding the string with <utf8> as the default encoding. However, if you like to write a <string> to an existing buffer object, then use the following line of code.

```
> buffer.write(string)
```

This class also offers other methods like <readInt8> and <writeUInt8> that allows read/write from various types of data to the buffer.

How To Load HTML In Node.Js?

Answer.

To load HTML in Node.js we have to change the “Content-type” in the HTML code from text/plain to text/html.

Let’s see an example where we have created a static file in web server.

```
fs.readFile(filename, "binary", function(err, file) {
if(err) {
response.writeHead(500, {"Content-Type": "text/plain"});
response.write(err + "\n");
response.end();
return;
}

response.writeHead(200);
response.write(file, "binary");
response.end();
});
```

Now we will modify this code to load an HTML page instead of plain text.

```
fs.readFile(filename, "binary", function(err, file) {
if(err) {
response.writeHead(500, {"Content-Type": "text/html"});
response.write(err + "\n");
response.end();
return;
}

response.writeHead(200, {"Content-Type": "text/html"});
response.write(file);
response.end();
});
```

What Is EventEmitter In Node.Js?

Events module in Node.js allows us to create and handle custom events. The Event module contains “EventEmitter” class which can be used to raise and handle custom events. It is accessible via the following code.

```
// Import events module
var events = require('events');
```

```
// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

When an EventEmitter instance encounters an error, it emits an “error” event. When a new listener gets added, it fires a “newListener” event and when a listener gets removed, it fires a “removeListener” event.

EventEmitter provides multiple properties like “on” and “emit”. The “on” property is used to bind a function to the event and “emit” is used to fire an event.

What Is NPM In Node.Js?

NPM stands for Node Package Manager. It provides following two main functionalities.

- It works as an Online repository for node.js packages/modules which are present at <nodejs.org>.
- It works as Command line utility to install packages, do version management and dependency management of Node.js packages.

NPM comes bundled along with Node.js installable. We can verify its version using the following command-

```
$ npm --version
```

NPM helps to install any Node.js module using the following command.

```
$ npm install <Module Name>
```

For example, following is the command to install a famous Node.js web framework module called express-

```
$ npm install express
```

What Is Package.Json? Who Uses It?

What Is <Package.Json>?

- It is a plain JSON (JavaScript Object Notation) text file which contains all metadata information about Node.js Project or application.
- This file should be present in the root directory of every Node.js Package or Module to describe its metadata in JSON format.
- The file is named as "package" because Node.js platform treats every feature as a separate component. Node.js calls these as Package or Module.

Who Use It?

- NPM (Node Package Manager) uses <package.json> file. It includes details of the Node.js application or package. This file contains a no. of different directives or elements. These directives guide NPM, about how to handle a module or package.

Does Node.Js Support Multi-Core Platforms? And Is It Capable Of Utilizing All The Cores?

Yes, Node.js would run on a multi-core system without any issue. But it is by default a single-threaded application, so it can't completely utilize the multi-core system.

However, Node.js can facilitate deployment on multi-core systems where it does use the additional hardware. It packages with a Cluster module which is capable of starting multiple Node.js worker processes that will share the same port.

Which Is The First Argument Usually Passed To A Node.Js Callback Handler?

Node.js core modules follow a standard signature for its callback handlers and usually the first argument is an optional error object. And if there is no error, then the argument defaults to null or undefined.

Here is a sample signature for the Node.js callback handler.

```
function callback(error, results) {
// Check for errors before handling results.
if ( error ) {
// Handle error and return.
}
// No error, continue with callback handling.
}
```

How To Create A Custom Directive In AngularJS?

Answer.

To create a custom directive, we have to first register it with the application object by calling the <directive> function. While invoking the <register> method of <directive>, we need to give the name of the function implementing the logic for that directive.

For example, in the below code, we have created a copyright directive which returns a copyright text.

```
app.directive('myCopyRight', function ()
{
return
{
template: '@CopyRight MyDomain.com '
};
});
```

Note – A custom directive should follow the camel case format as shown above.

What Is A Child_process Module In Node.Js?

Node.js supports the creation of child processes to help in parallel processing along with the event-driven model.

The Child processes always have three streams <child.stdin>, child.stdout, and child.stderr. The <stdio> stream of the parent process shares the streams of the child process.

Node.js provides a <child_process> module which supports following three methods to create a child process.

- **exec – <child_process.exec>** method runs a command in a shell/console and buffers the output.
- **spawn – <child_process.spawn>** launches a new process with a given command.
- **fork – <child_process.fork>** is a special case of the spawn() method to create child processes.

What Are The Different Custom Directive Types In AngularJS?

AngularJS supports a no. of different directives which also depend on the level we want to restrict them.

So in all, there are four different kinds of custom directives.

- Element Directives (E)
- Attribute Directives (A)
- CSS Class Directives (C)
- Comment Directives (M)

What Is A Control Flow Function? What Are The Steps Does It Execute?

It is a generic piece of code which runs in between several asynchronous function calls is known as control flow function.

It executes the following steps.

- Control the order of execution.
- Collect data.
- Limit concurrency.
- Call the next step in the program.