### 1) What is Express.js?

Express.js, or simply Express, is a free, open-source, lightweight, and fast backend web application framework for Node.js. It is released as open-source software under the MIT License.

It is designed for building single-page, multi-page, and hybrid web applications and APIs. It is called the de facto standard server framework for Node.js. It was founded and developed by **TJ Holowaychuk** in 2010 and written in JavaScript.

### 2) What are some distinctive features of Express?

As Express is a lightweight, minimal and flexible Node.js web application framework, it provides a robust set of features for web and mobile applications. Following is the list of some distinctive features of this framework:

* js can be used to design single-page, multi-page, and hybrid web applications and APIs.
* It allows to set up middleware to respond to HTTP/RESTful Requests.
* It defines a routing table to perform different HTTP operations (method and URL).
* It allows to dynamically rendering HTML Pages based on passing arguments to templates.
* It provides high performance because of its ultra-fast I/O. It prepares a thin layer; therefore, the performance is adequate.
* Its MVC-like structure makes it organize the web application into MVC architecture.
* It provides good database support. It supports RDBMS as well as NoSQL databases.
* It is asynchronous and single-threaded.
* Its robust API makes routing easy.

3) Is Express.js front-end or backend framework?

Express.js or Express is a JavaScript backend framework. It is mainly designed to develop complete web applications (single-page, multi-page, and hybrid web applications) and APIs. Express is the backend component of the **MEAN stack** where **M stands for MongoDB**, which handles database; **E stands for Express,** which handles backend; **A stands for AngularJS**, which is for the front-end, and **N stands for Node**.

4) Why do we use Express.js?

Express.js is an automatically prebuilt Node.js framework that facilitates us to create server-side web applications faster and smarter. The main reason for choosing Express is its simplicity, minimalism, flexibility, and scalability characteristics.

### 5) What is the difference between Express.js and Node.js?

Node.js is an open-source, cross-platform run-time environment used for executing JavaScript code outside of a browser. Node.js is not a framework or a programming language; it is a platform that acts as a web server. Many big companies such as Paypal, Uber, Netflix, Wallmart, etc., are using this. On the other hand, Express is a small framework based on the functionality of Node.js.

7) Write a code to get post a query in Express.js.

1. var bodyParser = require('body-parser')
2. app.use( bodyParser.json() );       // to support JSON-encoded
3. app.use(bodyParser.urlencoded({     // to support URL-encoded
4. extended: **true**
5. }));

**Express application generator:** This is used to create an application skeleton quickly. Use the following command to install the Express application generator.

1. npm install express-generator -g
2. express myApp

By using the above command, a project named "myApp" will be created along with following the files/folders in the project.

* **Bin:** The bin folder contains one file called www is the main configuration file of the app.
* **Public:** The public folder contains JavaScript, CSS, and images, etc.
* **Routes:** This folder contains the routing files.
* **Views:** The view folder contains the view files of the application.
* **js:** The app.js file is the main file of the application.
* **json:** The package.json file is the manifest file. It contains all metadata of the project, such as the packages used in the app (called dependencies) etc.

10) Which are the arguments available to an Express JS route handler function?

Following are the arguments that are available to an Express.js route handler-function:

* **Req:** the request object
* **Res:** the response object
* **Next (optional):** It is a function employed to pass management to one of the above route handlers.

### 14) How can you deal with error handling in Express.js? Explain with an example.

Error handling is much easier in the Express versions over Express 4.0. Use the following steps to do the error handling:

Create an Express.js application. There is no built-in middleware like error handler in express 4.0, so you have to either install a middleware or create a custom one.

**Create a Middleware:**

Create a middleware as following:

1. // error handler
2. app.use(function(err, req, res, next) {
3. // set locals, only providing error in development
4. res.locals.message = err.message;
5. res.locals.error = req.app.get('env') === 'development' ? err : {};
6. // render the error page
7. res.status(err.status || 500);
8. res.render('error');
9. });

**Install Error Handler Middleware:**

**Install the errorhandler as following:**

1. npm install errorhandler --save

**Create a variable:**

1. var errorhandler = require('errorhandler')

**Use the middleware as following:**

1. **if** (process.env.NODE\_ENV === 'development') {
2. // only use in development
3. app.use(errorhandler({log: errorNotification}))
4. }
5. function errorNotification(err, str, req) {
6. var title = 'Error in ' + req.method + ' ' + req.url
7. notifier.notify({
8. title: title,
9. message: str
10. })
11. }

15) Write the code to start serving static files in Express.js.

See the Example:

1. app.use(express.**static**('public'))
2. app.use('/static', express.**static**(path.join(\_\_dirname, 'public')))

16) What is Middleware in Express.js? What are the different types of Middleware?

Middleware is a function invoked by the Express routing layer before the final request handler.

Middleware functions are used to perform the following tasks:

* It is used to execute any code.
* It is also used to make changes to the request and the response objects.
* It is responsible for ending the request-response cycle.
* It can call the next middleware function in the stack.

**Type of Middleware**

Following are the main types of Middleware:

* Application-level Middleware
* Router-level Middleware
* Error-handling Middleware
* Built-in Middleware
* Third-party Middleware

**Application-level middleware:** The application-level middleware method is used to bind to the app object using app.use() method. It applies on all routes.

1. //This middleware will execute for each route.
2. app.use(function (req, res, next) {
3. console.log('Current Time:', Date.now())
4. next()
5. })

**Router-level Middleware:** The router-level Middleware is used to bind to a specific instance of express.Router().Built-in Middleware: The built-in Middleware was introduced with version 4.x. It ends the dependency on Connect.

There are the following built-in middleware functions in Express.js:

* **static:** It is used to serve static assets such as HTML files, images, etc.
* **json:** It is used to parse the incoming requests with JSON payloads. It is available with Express 4.16.0+
* **urlencoded:** It is used to parse the incoming requests with URL-encoded payloads. It is available with Express 4.16.0+

**Third-party Middleware:** There are many third-party middleware available such as:

* Body-parser
* Cookie-parser
* Mongoose
* Sequelize
* Cors
* Express-validator

To handle HTTP POST requests in Express.js version 4 and above, we have to install a middleware module called body-parser. Body-parser extracts the entire body portion of an incoming request stream and exposes it on req.body, The Middleware was a part of Express.js earlier, but now you have to install it separately. You can install it by using the following command:

1. npm install MODULE\_NAME

You can load it by using requires and used later:

**See the Example:**

1. var bodyParser = require('body-parser');
2. app.use(bodyParser.json());
3. app.use(bodyParser.urlencoded({ extended: **false** }))

#### **Note: You can use multiple middleware as an array on a single route.**

**See the Example:**

1. var middlewareArray = [middleware1, middleware2]
2. app.get('/home', middlewareArray, function (req, res, next) {
3. //Code snippets
4. })

### 17) Which template engines do Express support?

Express.js supports any template engine that follows the (path, locals, callback) signature.

### 18) How can we render a pain HTML?

There is no need to "render" HTML with the res.render() function. If you have a specific file, you can use the res.sendFile() function, but you should use the express if you serve many assets from a directory.static() middleware function.