Clarusway


# Backend Workshop -3-

## Workshop

Subject: Expressjs

### Learning Goals

Learning the basic structures we will use when doing a project with Express.js

### Introduction

Information about the framework we use can take shape in our minds. But when necessary, we must be able to express them verbally or in writing. This workshop aims to reinforce the basic information we may encounter during an interwiev.

### Pre-requirements

We will use the VSCode .

### Lets start

**1. What is Express?**

Answer:

Express is a minimalist web framework layer built on top of the NodeJS . That provides awesome features to build web and mobile applications. We can use Express to build single-page multi pages or hybrid web applications.

**2. Where does the popularity of Express come from?**

Answer:

Numerous Node.JS web application frameworks are available, but Express is extremely popular and serves as the "de facto" web application framework that all developers are expected to be familiar with. Since Express & NodeJS both are based on JavaScript, learning the language is not so difficult even for non-technical people. Due to the popularity of JS, developing the backend of applications is extremely easy.

We can list some of the reasons why it is popular as follows.

- Simple and easy to set up and customization.
- It supports MVC (Model-View-Controller), a very common architecture to design web applications.
- Enhanced performance with far fewer errors due to Google V8 Engine support.
- It is cross-platform and is not limited to any particular operating system.
- It lets you identify your application routes based on HTTP methods and URLs.
- It contains several middleware components that can be used on request and response to conduct extra duties.
- It helps you to identify a middleware handling error.
- It enables you to build a server with the REST API.
- Fast to link to databases like MongoDB, MySQL,SQlite,PostgreSQL.
- It comes with a default template engine and does support other template engines.
- Easily handle the application's static files and services.

**3. Have you ever taken a look, which famous companies use Express, what is Express place among other frameworks? Lets googling.**

Answer:

For example
https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/
https://www.apptension.com/blog-posts/top-backend-frameworks

**4. What are the differences between Express and Nodejs?**

Answer:

Express and NodeJS differ from each other in many ways. Below are some of the differences; Node JS is an open-source platform on which the JavaScript code is executed outside of a browser (server side). It is a platform acting as a web server and not a programming language or framework. Nodejs is used for a wider range of purposes. On the other hand, Express.js is a framework built on NodeJS. NodeJS focuses on providing a fast and efficient runtime for JavaScript, while Express focuses on simplifying the development of

server-side applications. Both NodeJS and Express are popular choices for web developers, but Express is the preferred choice for building web applications and RESTful APIs.

| Feature | Express | NodeJS |
|---|---|---|
| Purpose | Web framework | JavaScript runtime |
| Built on | NodeJS | Chrome's V8 JavaScript engine |
| Focus | Server-side development | General-purpose JavaScript programming |
| Key features | Routing, middleware, templating | Non-blocking I/O, event-driven programming |
| Use Cases | Building web applications, RESTful APIs | Server-side scripting, creating command line tools |

**5. What is Middleware, and what are its functions?**

Answer:

What is Middleware, and what are its functions? Middleware is the function that we invoke before the final request process through the express routing layer. Middleware basic functions are;

- Any code like setting headers, validation, etc., can be executed.
- Changes can be made to the response and request objects.
- The request-response cycle can also be ended by Middleware.
- The next middleware function can be called in the stack for proceeding and processing the final request.

**6. Explain what the keys we see in the package.json file mean.**

Answer:

Name: This field specifies the name of your package or project. It should be unique, lowercase, and not contain spaces or special characters.

Version: The version field indicates the current version of your package. It follows semantic versioning rules (e.g., 1.0.0) and helps others understand your package's release history.

Description: Here, you can provide a brief description of your package. This helps users and collaborators understand its purpose.

Main: This field specifies the entry point for your package. By default, it's typically set to "main": "index.js", assuming your main JavaScript file is named index.js.

Scripts: This section allows you to define custom scripts that can be executed using the npm run command. Common scripts include "start" for running your application and "test" for running tests.

Dependencies: This is where you list the packages your project depends on. These dependencies are essential for your project to work correctly. When someone else wants to use your project, they can run npm install, and it will install all the dependencies listed here.

DevDependencies: Similar to dependencies, but these packages are only needed during development. They may include testing libraries, build tools, or other utilities that are not required in a production environment.

Keywords: You can specify keywords that describe your package. These keywords help others discover your package when searching on the npm registry.

Author: Here, you can mention the author of the package along with their contact information.

License: The license field specifies under which terms your package is distributed. Common licenses include MIT, GPL, and Apache 2.0.

Repository: This field can point to the version control repository (e.g., GitHub) where your code is hosted. It helps users find and contribute to your project.

Browserslist: A configuration section that specifies which browsers and their versions your project supports. This is commonly used for frontend projects to ensure compatibility.

**7. What is the difference between res.send() and res.json() in Express.js?**

Answer:

res.send() is used to send a response with any type of data (string, object, buffer, etc.). While res.json() is used to send a JSON response. res.json() also sets the Content-Type header to application or JSON.

**8. What is the difference between app.use() and app.get() in Express.js?**

Answer:

app.use() is used to specify middleware that should be executed for all HTTP methods, while app.get() is used to specify middleware that should only be executed for GET requests.

**9. Write a middleware to log the incoming request method and URL to the console.**

Answer:

```
function logRequest(req, res, next) {
    console.log(`${req.method} ${req.url}`);
    next();
}
```

**10. Write a function in Express.js that calculates the sum of two numbers passed as query parameters.**

Answer:

```
app.get('/sum', (req, res) => {
const { num1, num2 } = req.query;
const sum = parseInt(num1) + parseInt(num2);
res.send(`The sum of ${num1} and ${num2} is ${sum}.`);
});
```

This function handles a GET request to the /sum endpoint and extracts the num1 and num2 query parameters from the request object using object destructuring. It then parses the parameters as integers using the parseInt function and calculates their sum.

---

☺ **Thanks for Attending** ✍

Clarusway 

---