TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Express.js

EXPERIENTIAL
learning factory

What is Express.js?

- Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
- It is an open source framework developed and maintained by the Node.js foundation.
- Express provides a minimal interface to build our applications. It provides us the tools that are required to build our app.
- It is flexible as there are numerous modules available on **npm**, which can be directly plugged into Express.
- Express was developed by **TJ Holowaychuk** and is maintained by the Node.js foundation and numerous open source contributors.

Installing Express.js

- Express is a Node.js module available through the npm registry.
- Before installing, download and install Node.js.
- Installation is done using the npm install command:

  “**npm install express**”

Listening to the server


```
const express = require('express');
const app = express();
app.get('/', function (req, res) {
        res.send('Hello World');
});
app.listen(3000)
```

- app.get(route, callback)

This function tells what to do when a get request at the given route is called. The callback function has 2 parameters, request(req) and response(res). The request object(req) represents the HTTP request and has properties for the request query string, parameters, body, etc. Similarly, the response object represents the HTTP response that the Express app sends when it receives an HTTP request.

- res.send()

This function takes any data as input and it sends the data to the requesting client. Here we are sending the string "Hello World!".

HTTP Methods

The HTTP method is supplied in the request and specifies the operation that the client has requested.

1. HEAD
2. TRACE
3. PUT
4. DELETE
5. OPTIONS
6. POST
7. GET
8. CONNECT

GET Method

- GET is used to request data from a specified resource.
- GET is one of the most common HTTP methods.
- The query string (name/value pairs) is sent in the URL of a GET request:

  eg: /test?name1=value1&name2=value2

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests are only used to request data (not modify)

POST Method

- POST is used to send data to a server to create/update a resource.
- The data sent to the server with POST is stored in the request body of the HTTP request.
- POST is one of the most common HTTP methods
- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

POST is not **idempotent**, so making a **POST** request more than one time may have additional side effects, like creating a second, third and fourth duplicates. But the key word here is may. Just because an endpoint uses **POST** doesn't mean that it must have side effects on every request. It just might have side effects.

PUT, DELETE

PUT Method

- PUT is used to send data to a server to create/update a resource.
- The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

Delete Method

- The DELETE method deletes the specified resource.

GET vs POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Routing

- **Routing** refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).
- Each route can have one or more handler functions, which are executed when the route is matched.
- Route definition takes the following structure:

app.METHOD(PATH, HANDLER)

Where:

app is an instance of express.
METHOD is an HTTP request method, in lowercase.
PATH is a path on the server.
HANDLER is the function executed when the route is matched.

Middleware Functions

Middleware functions are functions that have access to the request object (req), the response object (res), and the next function in the application's request-response cycle. The next function is a function in the Express router which, when invoked, executes the middleware succeeding the current middleware.

Middleware functions can perform the following tasks:

- Execute any code.
- Make changes to the request and the response objects.
- End the request-response cycle.
- Call the next middleware in the stack.

If the current middleware function does not end the request-response cycle, it must call next() to pass control to the next middleware function. Otherwise, the request will be left hanging.

Types of Middlewares

- Express is a routing and middleware web framework that has minimal functionality of its own: An Express application is essentially a series of middleware function calls.
- An Express application can use the following types of middleware:
    1. Application-level middleware
    2. Router-level middleware
    3. Error-handling middleware
    4. Built-in middleware
    5. Third-party middleware

Static Files

Static files are files that clients download as they are from the server. Create a new directory, public. Express, by default does not allow you to serve static files. You need to enable it using the following built-in middleware.

```
app.use(express.static('public'));
```

Note – Express looks up the files relative to the static directory, so the name of the static directory is not part of the URL.

The root route is now set to your public directory, so all static files you load will be considering public as root.

TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Thank You !!!

No.01, 3rd Cross Basappa Layout, Gavipuram Extension,
Kempegowda Nagar, Bengaluru, Karnataka 560019

praveen.d@testyantra.com

www.testyantra.com

EXPERIENTIAL
learning factory