

# Library v/s Framework

## Library

A library is a collection of pre-written code that can be used to perform specific tasks.

eg - axios

## Framework

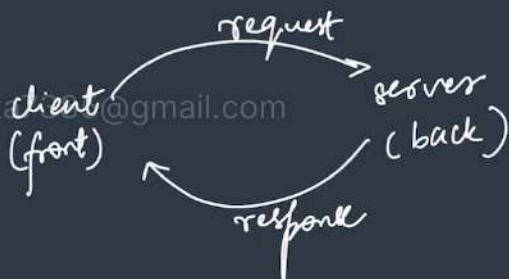
A framework is a set of pre-written code that provides a structure for developing software applications.

eg - express

# Express

A Node.js web application framework that helps us to make web applications  
It is used for **server** side programming.

1. listen for incoming request
2. parse
3. match response with routes
4. response



# Getting Started with Express

---

```
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app listening on port ${port}`);
});
```

**\*Ports** are the logical endpoints of a network connection that is used to exchange information between a web server and a web client.

```
[shradhakhapra@Shradhas-Air ExpressDir % npm init ]
This utility will walk you through creating a package.js
on file.
It only covers the most common items, and tries to guess
sensible defaults.

See `npm help init` for definitive documentation on thes
e fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package
and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
[package name: (expressdir) ]
[version: (1.0.0) ]
[description: ]
[entry point: (index.js) ]
[test command: ]
[git repository: ]
[keywords: ]
[author: Shradha ]
[license: (ISC) ]
```

```
const express = require('express')
const app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

## Installation

This is a **Node.js** module available through the **npm registry**.

Before installing, **download and install Node.js**. Node.js 0.10 or higher is required.

If this is a brand new project, make sure to create a `package.json` first with the **npm init** command.

Installation is done using the **npm install** command:

```
$ npm install express
```

Follow **our installing guide** for more information.

shradhakhapra@Shradhas-Air ExpressDir % node index.js



JS index.js X

JS index.js > ...


```
1  const express = require("express");
2  const app = express();
3
4  console.dir(app);
5
```

JS index.js ×

JS index.js > ...

```
1  const express = require("express");
2  const app = express();
3
4  console.dir(app);
5
6  let port = 3000;
7
8  app.listen(port, () => {
9    console.log(`app is listening on port ${port}`);
10  });
11
```

 localhost:3000

 localhost:3000

 localhost:3000 - Google Search



```
shradhakhapra@Shradhas-Air ExpressDir % node index.js  
app is listening on port 3000  
^C  
shradhakhapra@Shradhas-Air ExpressDir % node index.js  
app is listening on port 3000
```



localhost:3000

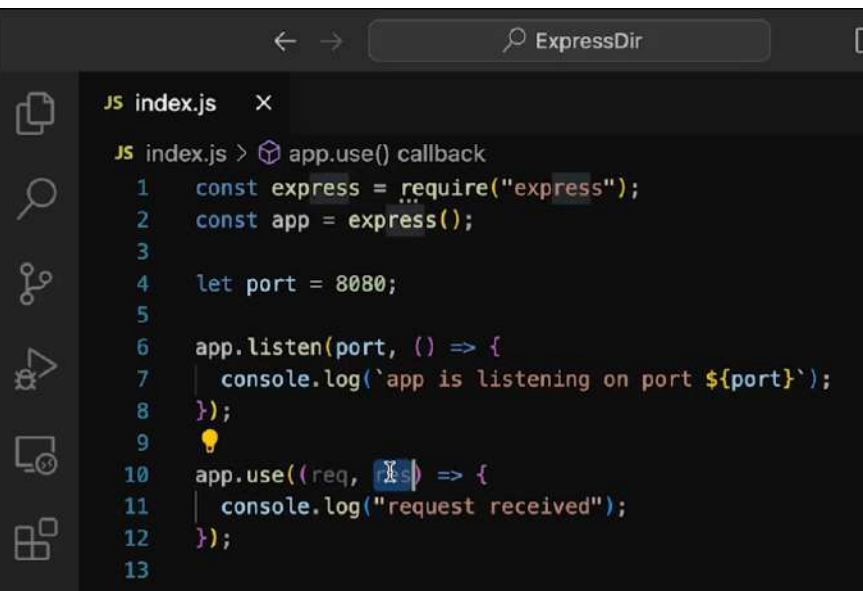
Cannot GET /

## app.use

```
app.use((req, res) => {  
  console.log("new incoming request");  
});
```

```
shradhakhapra@Shradhas-Air ExpressDir % node index.js  
app is listening on port 8080  
request received  
request received  
request received  
█
```

snehagupta7385@gmail.com

The image shows a VS Code editor window with a dark theme. The title bar at the top says "ExpressDir". The editor is open to a file named "index.js". The code in the file is as follows:

```
JS index.js > app.use() callback  
1  const express = require("express");  
2  const app = express();  
3  
4  let port = 8080;  
5  
6  app.listen(port, () => {  
7    console.log(`app is listening on port ${port}`);  
8  });  
9  
10 app.use((req, res) => {  
11   console.log("request received");  
12 });  
13
```

Line 9 has a yellow lightbulb icon next to it, indicating a suggestion. The left sidebar shows the Explorer, Search, Source Control, Run and Debug, and Extensions views. The Explorer view is active, showing the file structure of the project.

# Sending a Response



request  
(req)



response  
(res)

```
ended: false,
finished: false,
destroyed: false,
decodeStrings: false,
defaultEncoding: 'utf8',
length: 0,
writing: false,
corked: 0,
sync: true,
bufferProcessing: false,
onwrite: [Function: bound onwrite],
writecb: null,
writelen: 0,
afterWriteTickInfo: null,
buffered: [],
bufferedIndex: 0,
allBuffers: true,
allNoop: true,
pendingcb: 0,
constructed: true,
prefinished: false,
errorEmitted: false,
emitClose: false,
autoDestroy: true,
errored: null,
closed: false,
closeEmitted: false,
[Symbol(kOnFinished)]: []
},
allowHalfOpen: true,
_sockname: null,
_pendingData: null,
_pendingEncoding: '',
server: Server {
  maxHeaderSize: undefined,
  insecureHTTPParser: undefined,
  _events: [Object: null prototype],
  _eventsCount: 2,
  _maxListeners: undefined,
  _connections: 1,
```

snehagupta7385@gmail.com ExpressDir

JS index.js ×

JS index.js > app.use() callback

```
1  const express = require("express");
2  const app = express();
3
4  let port = 8080;
5
6  app.listen(port, () => {
7    console.log(`app is listening on p
8  });
9
10 app.use((req, res) => {
11   console.log(req);
12   console.log("request received");
13 });
14
```

## express()

### Application

### Request

### Response

#### Properties

res.app  
res.headersSent  
res.locals

#### Methods

res.append()  
res.attachment()  
res.cookie()  
res.clearCookie()  
res.download()  
res.end()  
res.format()  
res.get()  
res.json()  
res.jsonp()  
res.links()  
res.location()  
res.redirect()  
res.render()  
**res.send()**  
res.sendFile()  
res.sendStatus()  
res.set()

## res.send([body])

Sends the HTTP response.

The body parameter can be a Buffer object, a String, an object, Boolean, or an Array. For example:

```
res.send(Buffer.from('whoop'))  
res.send({ some: 'json' })  
res.send('<p>some html</p>')  
res.status(404).send('Sorry, we cannot find that!')  
res.status(500).send({ error: 'something blew up' })
```

This method performs many useful tasks for simple non-streaming responses: For example, it automatically assigns the Content-Length HTTP response header field (unless previously defined) and provides automatic HEAD and HTTP cache freshness support.

When the parameter is a Buffer object, the method sets the Content-Type response header field to "application/octet-stream", unless previously defined as shown below:


```
res.set('Content-Type', 'text/html')  
res.send(Buffer.from('<p>some html</p>'))
```

When the parameter is a String, the method sets the Content-Type to "text/html":

```
res.send('<p>some html</p>')
```

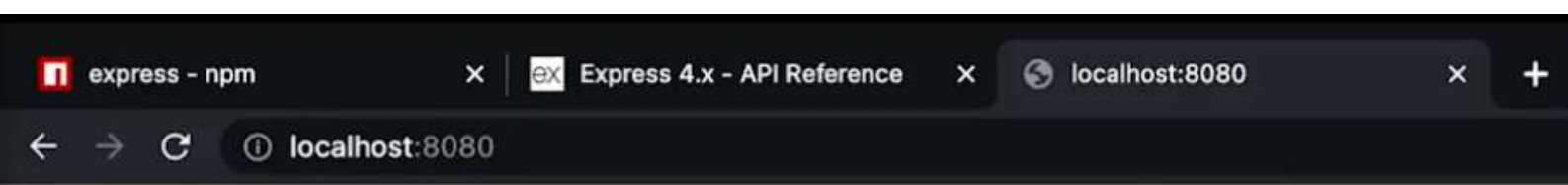
When the parameter is an Array or Object, Express responds with the JSON representation:

```
res.send({ user: 'tobi' })
```

JS index.js >  app.use() callback


```
1  const express = require("express");
2  const app = express();
3
4  let port = 8080;
5
6  app.listen(port, () => {
7    console.log(`app is listening on port ${port}`);
8  });
9
10 app.use((req, res) => {
11   // console.log(req);
12   💡 console.log("request received");
13   res.send("this is a basic response");
14 });
15
```






this is a basic response

JS index.js X

JS index.js >  app.use() callback

```
1  const express = require("express");
2  const app = express();
3
4  let port = 8080;
5
6  app.listen(port, () => {
7    console.log(`app is listening on port ${port}`);
8  });
9
10 app.use((req, res) => {
11    console.log("request received");
12   res.send({
13     name: "apple",
14     color: "red",
15   });
16 });
17
```



express - npm



Express 4.x - API Reference



localhost:8080



localhost:8080

```
{"name": "apple", "color": "red"}
```

JS index.js

JS index.js >  app.use() callback

```
1  const express = require("express");
2  const app = express();
3
4  let port = 8080;
5
6  app.listen(port, () => {
7    console.log(`app is listening on port ${port}`);
8  });
9
10 app.use((req, res) => {
11   console.log("request received");
12   let code = "<h1>Fruits</h1> <ul><li>apple</li><li>orange";
13   res.send(code);
14 });
15
```

# Fruits

- apple
- orange

# HOPPSCOTCH



Save My Work



REST



GraphQL



Realtime



Settings



My Wor... > Collections



Search



+ New



Collections are empty

Add new

GET Untitled



GET

http://localhost:8080

Send

Parameters

Body

Headers 1

Authorization

Pre-request Script

Tests

Query Parameters

Parameter 1

Value 1

Status: 200 • OK Time: 69 ms Size: 54 B

HTML

Raw

Headers 7

Test Results

Response Body

1 <h1>Fruits</h1> <ul><li>apple</li><li>orange</li></ul>

# Routing

it is process of selecting a path for traffic in a network or between or across multiple networks.

```
app.get("/apple", (req, res) => {  
  res.send({  
    name: "apple",  
    color: "red",  
  });  
});
```

**express()****Application***Properties*

app.locals

app.mountpath

*Events*

mount

*Methods*

app.all()

app.delete()

app.disable()

app.disabled()

app.enable()

app.enabled()

app.engine()

[app.get\(\)](#)

app.get()

app.listen()

app.METHOD()

app.param()

app.path()

app.post()

app.put()

app.render()

**app.get(name)**

Returns the value of `name` app setting, where `name` is one of the strings in the [app settings table](#). For example:

```
app.get('title')
// => undefined

app.set('title', 'My Site')
app.get('title')
// => "My Site"
```

**app.get(path, callback [, callback ...])**


Routes HTTP GET requests to the specified path with the specified callback functions.

**Arguments**

Argument	Description
path	The path for which the middleware function is invoked; can be any of: <ul style="list-style-type: none"><li>• A string representing a path.</li><li>• A path pattern.</li><li>• A regular expression pattern to match paths.</li><li>• An array of combinations of any of the above.</li></ul>



JS index.js ●

JS index.js >  app.get("/") callback

```
1  const express = require("express");
2  const app = express();
3
4  let port = 8080;
5
6  ✓ app.listen(port, () => {
7    |   console.log(`app is listening on port ${port}`);
8    | });
9
10 ✓ app.get("/", (req, res) => {
11   |   res.send("you contacted root path");
12   | });
13
14 ✓ // app.use((req, res) => {
15   | //   console.log("request received");
16   | //   let code = "<h1>Fruits</h1>";
17   | //   res.send(code);
18   | // });
19
20
```

JS index.js

JS index.js > ...

```
1  const express = require("express");
2  const app = express();
3
4  let port = 8080;
5
6  app.listen(port, () => {
7    console.log(`app is listening on port ${port}`);
8  });
9  💡
10 app.get("/", (req, res) => {
11   res.send("you contacted root path");
12 });
13
14 app.get("/apple", (req, res) => {
15   res.send("you contacted apple path");
16 });
17
18 app.get("/orange", (req, res) => {
19   res.send("you contacted orange path");
20 });
21
```

JS index.js >  app.get("\*") callback

```
7 | console.log(`app is listening on port ${port}`);  
8 | });  
9 |  
10 | app.get("/", (req, res) => {  
11 | | res.send("you contacted root path");  
12 | | });  
13 |  
14 | app.get("/apple", (req, res) => {  
15 | | res.send("you contacted apple path");  
16 | | });  
17 |  
18 | app.get("/orange", (req, res) => {  
19 | | res.send("you contacted orange path");  
20 | | });  
21 |  
22 | app.get("*", (req, res) => {  
23 | | res.send("this path does not exist");  
24 | | });  
25 |  
26 | // app.use((req, res) => {  
27 | //   console.log("request received");  
28 | //   let code = "<h1>Fruits</h1> <ul><li>apple</li><li>ora  
29 | //   res.send(code);  
30 | // });
```

GET



http://localhost:8080/

Parameters

Body

Headers

1

Authorization

Pre-

Query Parameters

Parameter 1

Status: 200 • OK Time: 83 ms Size: 23 B

HTML

Raw

Headers

7

Test Results

Response Body

1

you contacted root path

snehagupta7385@gmail.com

GET



http://localhost:8080/apple



Parameters

Body

Headers

1

Authorization

Query Parameters

Parameter 1

Status: 200 • OK

Time: 12 ms

Size: 24 B

HTML

Raw

Headers

5

Test Results

Response Body

1

you contacted apple path



GET



http://localhost:8080/pineapple

Parameters

Body

Headers 1

Authorization

Pre-requests

Query Parameters

Parameter 1

Value

Status: 200 • OK Time: 10 ms Size: 24 B

HTML

Raw


Headers 7

Test Results

Response Body

1 this path does not exist

JS index.js X

JS index.js >  app.post("/") callback

```
13
14   app.get("/apple", (req, res) => {
15     res.send("you contacted apple path")
16   });
17
18   app.get("/orange", (req, res) => {
19     res.send("you contacted orange path")
20   });
21
22   app.get("*", (req, res) => {
23     res.send("this path does not exist")
24   });
25
26   app.post("/", (req, res) => {
27     res.send("you sent a post request to")
28   });
29
30   // app.use((req, res) => {
31     //   console.log("request received")
32     //   let code = "<h1>Fruits</h1> <u"
33     //   res.send(code);
34     // });
35
```

POST Untitled



POST



http://localhost:8080/

Parameters

Body

Headers

1

Authorization

Pre-r

Query Parameters

Parameter 1

Status: 200 • OK

Time: 12 ms

Size: 31 B

HTML

Raw

Headers

7

Test Results

Response Body

1

you sent a post request to root



# Nodemon

To automatically restart server with code changes

Neat! Pickled Muskrat!

npm

Search packages

nodemon DT

3.0.1 • Public • Published 1 month ago

Readme

Code Beta

10 Dependencies

4,763 Dependents

## nodemon

nodemon is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected.

nodemon does **not** require *any* additional changes to your code or method of development. nodemon is a replacement wrapper for `node`. To use `nodemon`, replace the word `node` on the command line when executing your script.

Install

```
> npm i nodemon
```

Repository

[github.com/re](https://github.com/remy/nodemon)

Homepage

[nodemon.i](https://nodemon.io)

```
shradhakhapra@Shradhas-Air ExpressDir % npm install -g nodemon
```

```
added 34 packages, and audited 35 packages in 2s
```

```
3 packages are looking for funding  
run `npm fund` for details
```

```
found 0 vulnerabilities
```

```
shradhakhapra@Shradhas-Air ExpressDir % nodemon -v  
3.0.1
```

```
shradhakhapra@Shradhas-Air ExpressDir % nodemon index.js
```

```
[nodemon] 3.0.1
```

```
[nodemon] to restart at any time, enter `rs`
```

```
[nodemon] watching path(s): *.*
```

```
[nodemon] watching extensions: js,mjs,cjs,json
```

```
[nodemon] starting node index.js
```

```
app is listening on port 8080
```

```
[nodemon] restarting due to changes...
```

```
[nodemon] starting node index.js
```

```
app is listening on port 8080
```

```
█
```



JS index.js X

JS index.js > app.get("/") callback

```
1 const express = require("express");
```

```
2 const app = express();
```

```
3
```

```
4 let port = 8080;
```

```
5
```

```
6 app.listen(port, () => {
```

```
7   console.log(`app is listening on
```

```
8 });
```

```
9
```

```
10 app.get("/", (req, res) => {
```

```
11   res.send("hello, i am root");
```

```
12 });
```

```
13
```

```
14 app.get("/apple", (req, res) => {
```

```
15   res.send("you contacted apple pat
```

```
16 });
```

```
17
```

```
18 app.get("/orange", (req, res)
```

```
19   res.send("you contacted ora
```

```
20 });
```

```
21
```

```
22 app.get("*", (req, res) => {
```


```
23   res.send("this path does n
```

# Path Parameters

req.params

```
app.get("/ig/:username", (req, res) => {  
  let { username } = req.params;  
  res.send(`This account belongs to @${username}`);  
});
```

JS index.js X

JS index.js >  app.get("/:username") callback

```
10 app.get("/", (req, res) => {
11   res.send("hello, i am root");
12 });
13 
14 app.get("/:username", (req, res) => {
15   console.log(req.params);
16   res.send("hello, i am root");
17 });
18
```


```
app is listening on port 8080  
{ username: 'apnacollege' }
```

<http://localhost:8080/google/123>

```
app is listening on port 8080  
{ username: 'apnacollege', id: '123' }
```



JS index.js ×

JS index.js >  app.get("/:username/:id") callback

```
10 app.get("/", (req, res) => {
11   | res.send("hello, i am root");
12   | });
13
14 app.get("/:username/:id", (req, res) => {
15   | let { username, id } = req.params;
16   | res.send(`welcome to the page of @${username}.`);
17   | });
18
```

GET



http://localhost:8080/apple/123



Parameters

Body

Headers

1

Authorization

Pr

Query Parameters

Parameter 1

Status: 200 • OK

Time: 67 ms

Size: 30 B

HTML

Raw

Headers

7

Test Results

Response Body

1

welcome to the page of @apple.

GET



http://localhost:8080/google/123

Parameters

Body

Headers

1

Authorization

Query Parameters

Parameter 1

Status: 200 • OK Time: 23 ms Size: 31 B

HTML

Raw

Headers

7

Test Results


Response Body

1

welcome to the page of @google.



JS index.js ×

JS index.js >  app.get("/:username/:id") callback

```
10 app.get("/", (req, res) => {
11   res.send("hello, i am root");
12 });
13
14 app.get("/:username/:id", (req, res) => {
15   let { username, id } = req.params;
16   let htmlStr = `<h1>welcome to the page of @${username}<
17   res.send(htmlStr);
18 });
19
```

← → ↻ ⓘ localhost:8080/apnacollege/123

**welcome to the page of @apnacollege!**

# Query Strings

**req.query**

```
app.get("/search", (req, res) => {  
  let { q } = req.query;  
  if (!q) {  
    res.send("No search query");  
  }  
  res.send(`These are the results for: ${q}`);  
});
```

JS index.js X

JS index.js > ...

```
10 app.get("/", (req, res) => {
11   res.send("hello, i am root");
12 });
13
14 app.get("/:username/:id", (req, res) => {
15   let { username, id } = req.params;
16   let htmlStr = `<h1>welcome to the`;
17   res.send(htmlStr);
18 });
19 💡
20 app.get("/search", (req, res) => {
21   console.log(req.query);
22   res.send("no results");
23 });
24
```

GET



http://localhost:8080/search?q="apple"

Parameters

Body

Headers

1

Authorization

Pre-requests

Query Parameters

Parameter 1

Value

Status: 200 • OK Time: 82 ms Size: 10 B

HTML

Raw

Headers

7

Test Results

Response Body

1  
⚡

no results



pp is listening on port 8080

q: 'apple' }

GET



http://localhost:8080/search?q=apple&color=green



Parameters

Body

Headers

1

Authorization

Pre-request Sc

Query Parameters

Parameter 1

Value 1

Status: 200 • OK Time: 67 ms Size: 10 B

HTML

Raw

Headers

7

Test Results

Response Body

1 no results

```
{ q: 'apple' }  
{ q: 'apple', color: 'green' }
```

JS index.js > ...

```
10 app.get("/", (req, res) => {
11   res.send("hello, i am root");
12 });
13
14 app.get("/:username/:id", (req, res) => {
15   let { username, id } = req.params;
16   let htmlStr = `<h1>welcome to the page of @${username}`;
17   res.send(htmlStr);
18 });
19
20 app.get("/search", (req, res) => {
21   let { q } = req.query;
22   res.send(`search results for query: ${q}`);
23 });
```

GET



http://localhost:8080/search?q=orange

Parameters

Body

Headers

1

Authorization

F

Query Parameters

Parameter 1

Status: 200 • OK Time: 9 ms Size: 32 B

HTML

Raw

Headers

7

Test Results

Response Body

1 search results for query: orange

```
app.get("/search", (req, res) => {  
  let { q } = req.query;  
  res.send(`<h1></h1>Search results for query: ${q}`);  
});
```

← → ↻ ⓘ localhost:8080/search?q=orange

**search results for query: orange**

← → ↻ ⓘ localhost:8080/search?



# nothing searched