



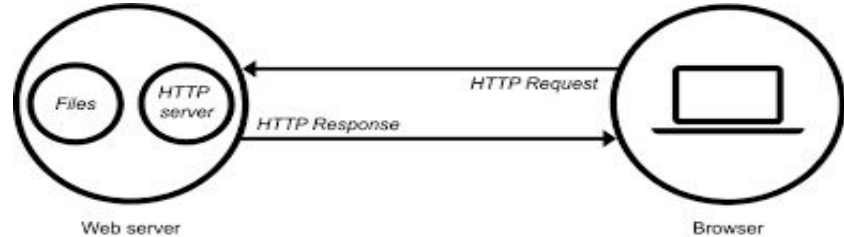
Coach: Arif, DCI  
Module: Backend

# What is **Server**?



- A machine with many hdds/storages
- Always live or active

# What is **server side code**?



```
const http = require('http')
```

```
const hostname = '127.0.0.1'
const port = 3000;
```

```
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain')
  res.end('Hello World')
});
```

```
server.listen(port, hostname, () => {
  console.log(`Server running a
http://${hostname}:${port}/`)
});
```

# What is **module**?

Background question:

What reactjs component can do?

Why we call it component?

- Reuse same code
- File sharing
- More structural

Types:

Build-in:

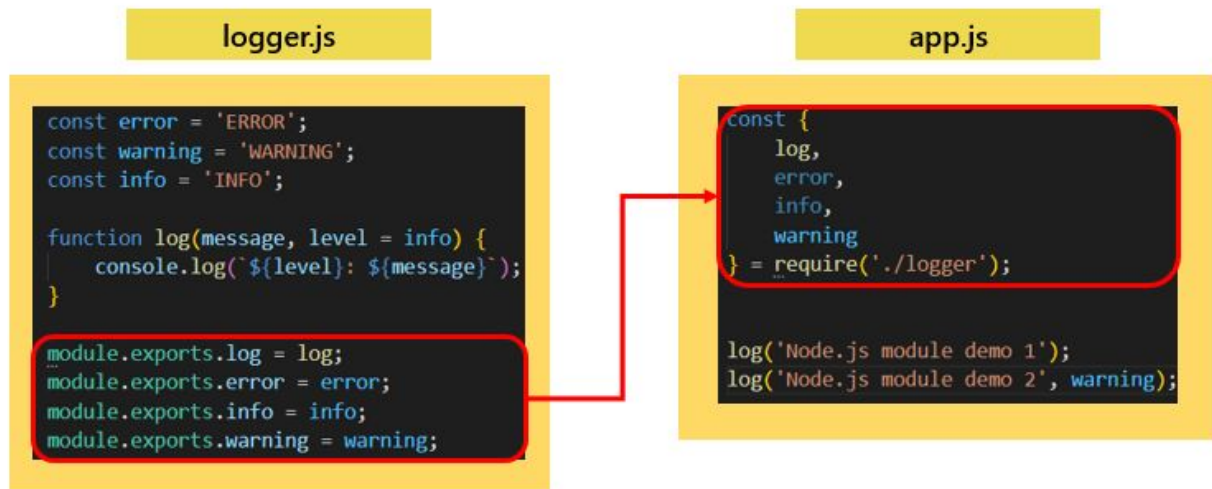
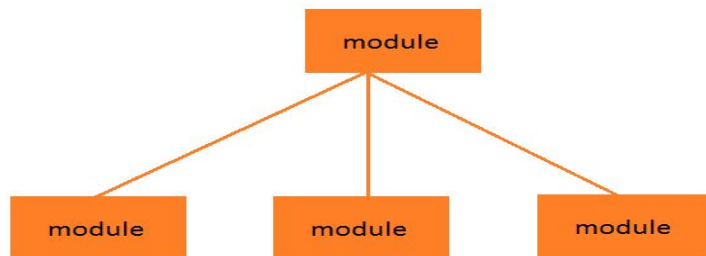
Http, fs, path, url etc.

3rd-party:

Cors, multer, passport

Custom:

Your code



# What is **framework**?

- Complete backend web server skeleton
- Like a library/set of code but can scale, maintain and extend your apps
- Make life easier

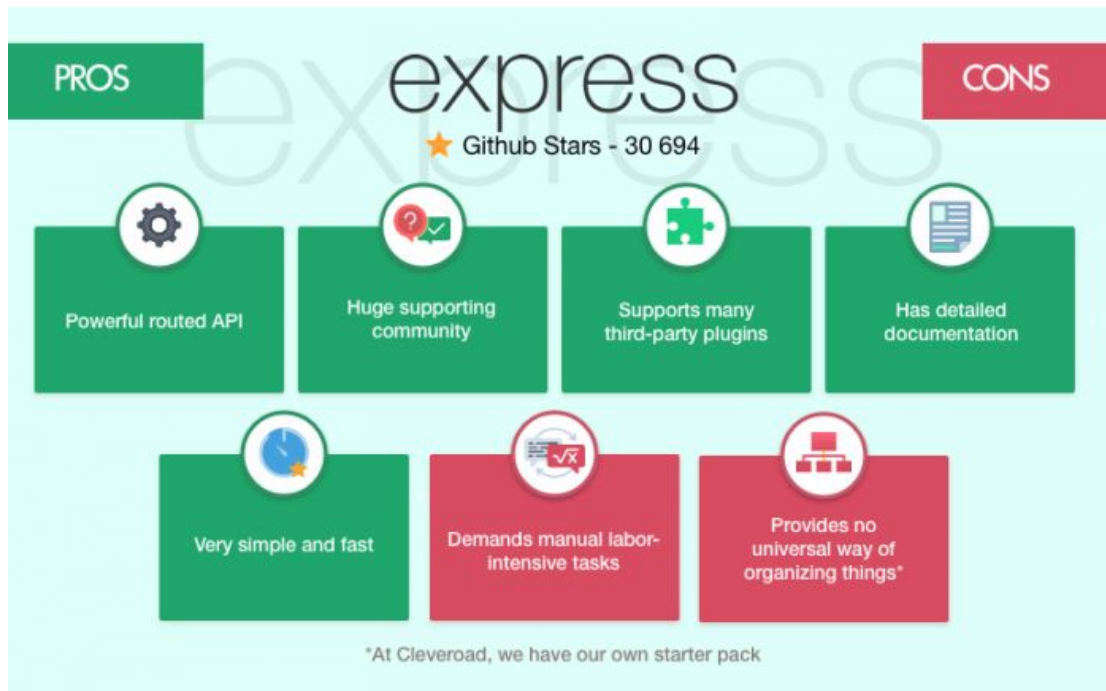
## Top NodeJS Frameworks



# Why Expressjs?

[Who are using expressjs?](#)

*No best way to do,  
Light-weight,  
Faster,  
Easy*



# Routing in expressjs

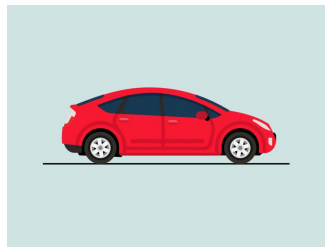
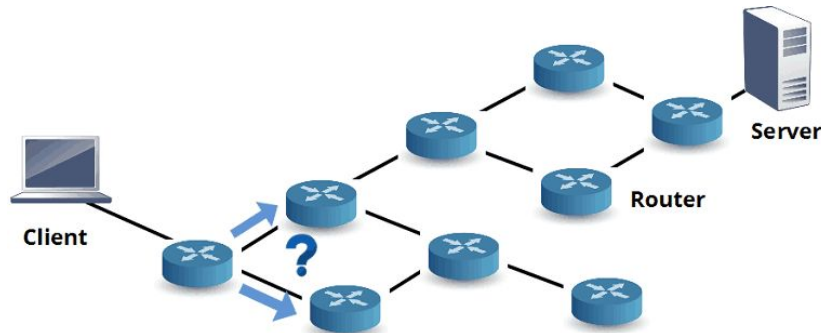
Where to Go? Which way?

- 1 route for 1 way
- Route with data/params or without
- Use http methods(get,post etc.)

Examples:

```
app.get('/user/goToBerlin', callback)
```

```
app.post('/create/user', callback)
```



## Render in Expressjs

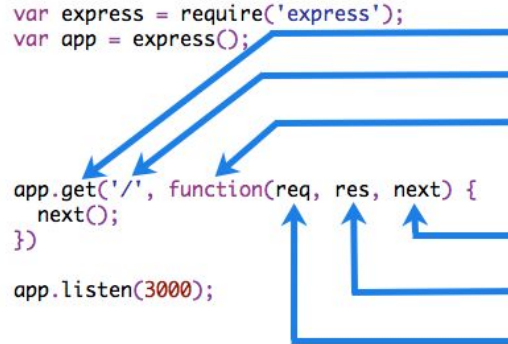
What to  
display/present/view?

```
res.send(), // return message/string  
res.sendFile(), // return a html file  
res.json(), // return a json object  
res.render() // return a template engine view
```

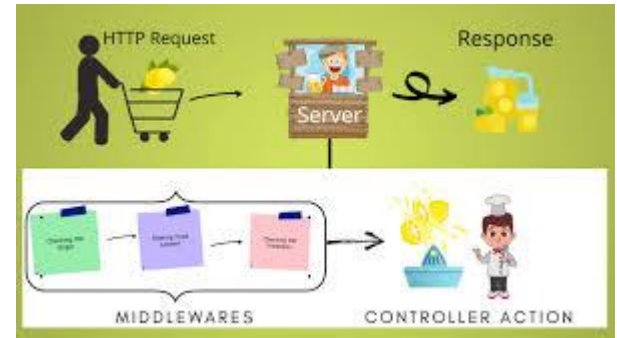
# Middleware in nodejs

- Something works in between request and response process(in middle)
- A function
- Can have 3 arguments (req, res, next)
- Next() is a callback and could be any name
- We can use middleware by using app.use() or in route(path,middleware,callback)

```
var express = require('express');  
var app = express();  
  
app.get('/', function(req, res, next) {  
  next();  
})  
  
app.listen(3000);
```



The diagram illustrates the execution flow of the provided code. Blue arrows show the sequence: an HTTP request enters the `app.get()` route handler, then passes through the `next()` call to the next middleware function (represented by the second arrow), and finally reaches the `app.listen()` function. The response is generated by the controller action (the third arrow) and is sent back to the client.



# Resources:

1. [Nodejs official](#)
2. [Expressjs](#)
3. [Module in nodejs](#)
4. [Tutorial for basic](#): w3School