XCCELERATE

D23 - Node Packages

•••

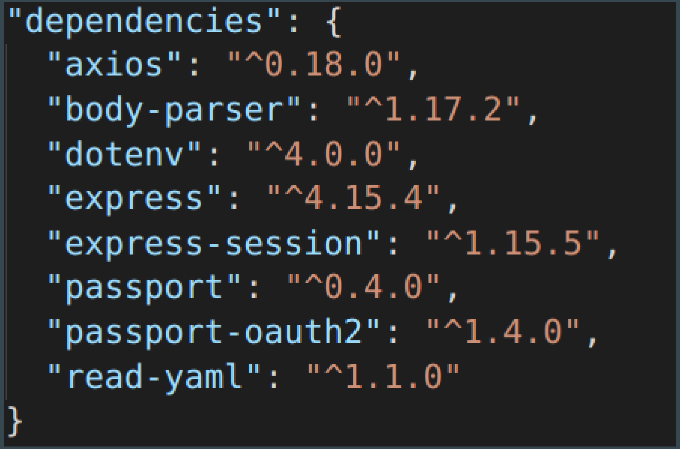
Agenda

- NPM
- Express
- Express Middleware
- HTTP Request & Response Cycle
- Cookie & Session

NPM

NPM manages the dependency of our project by recording the versions and the name of the packages we used in the project.

Whenever we are developing our application , we are also developing a package on our own.



```
"dependencies": {
  "axios": "^0.18.0",
  "body-parser": "^1.17.2",
  "dotenv": "^4.0.0",
  "express": "^4.15.4",
  "express-session": "^1.15.5",
  "passport": "^0.4.0",
  "passport-oauth2": "^1.4.0",
  "read-yaml": "^1.1.0"
}
```

npm init

npm init initializes the project to be a NPM repository such that you can run npm commands within that folder by creating a package.json file in the folder.

There are two advantages of using npm init for each projects:

- Separate packages dependencies of different repositories.
  - Each Project has its own package.json
  - For example:
    - 1st Project use axios@1.0.0,
    - 2nd Project use axios@2.0.0
- Install packages from NPM repositories without searching and downloading packages manually.

This is normally the command being run after running git init.

npm install

There are 3 types of npm install

1. **Install local packages**

   npm install <package-name>

1. **Install global packages**

   npm install -g <package-name>

3. **Install packages according to the package.json**

   npm install

Global packages are shared by all npm projects in your machine. Local packages are limited to be used locally.

Express

Express is a web framework that helps us building modern web applications. Here is the architecture of a typical Express Application

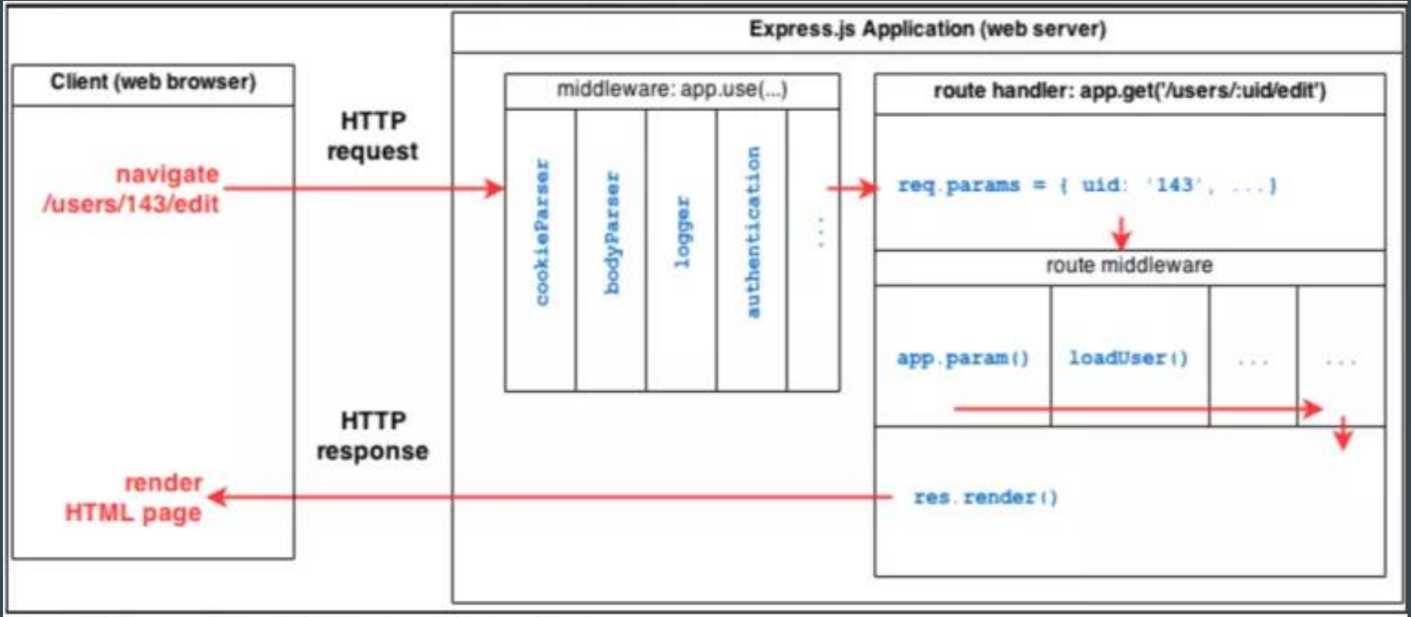
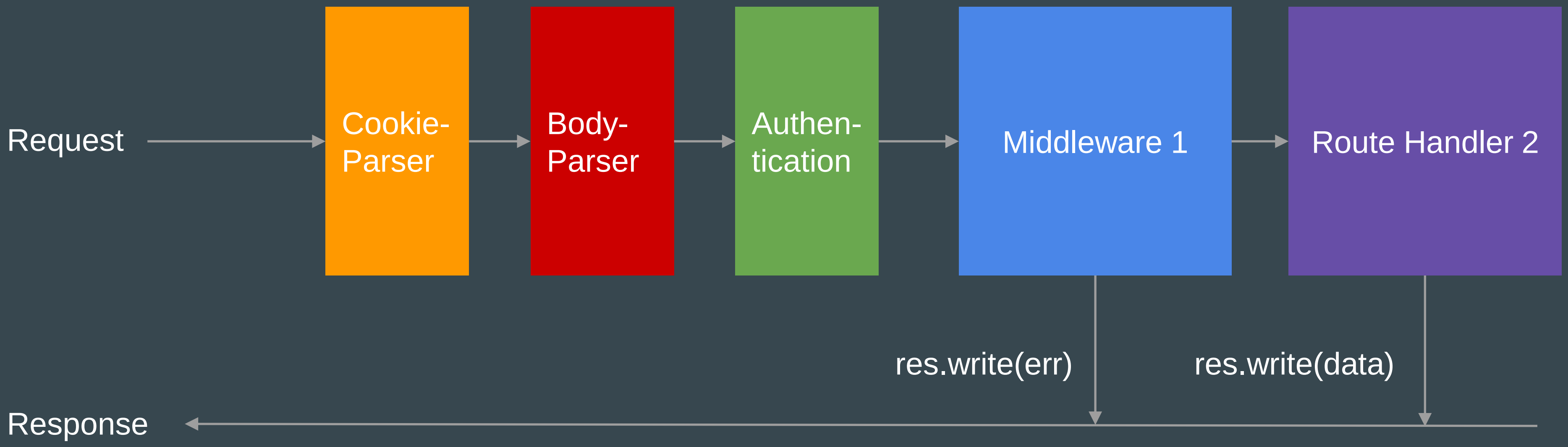


Figure 6 ExpressJS and Middleware Entire Stack

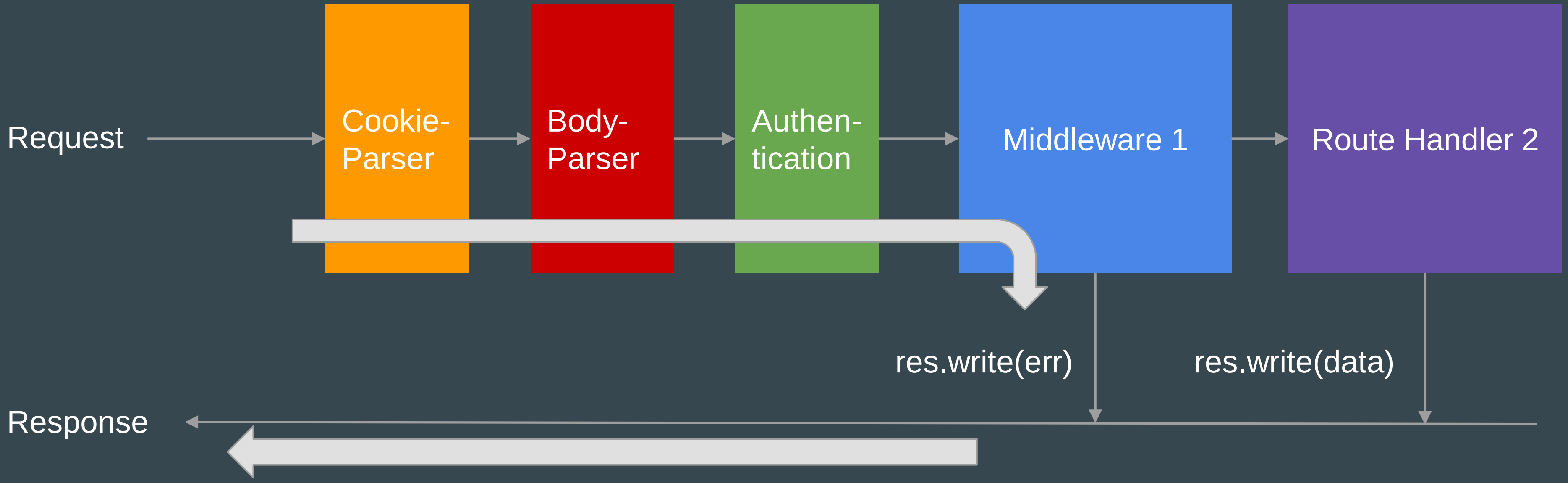
Express architecture explained

Express makes use of middleware architecture to perform tasks.
Essentially **route handlers** itself is also just a middlewares.



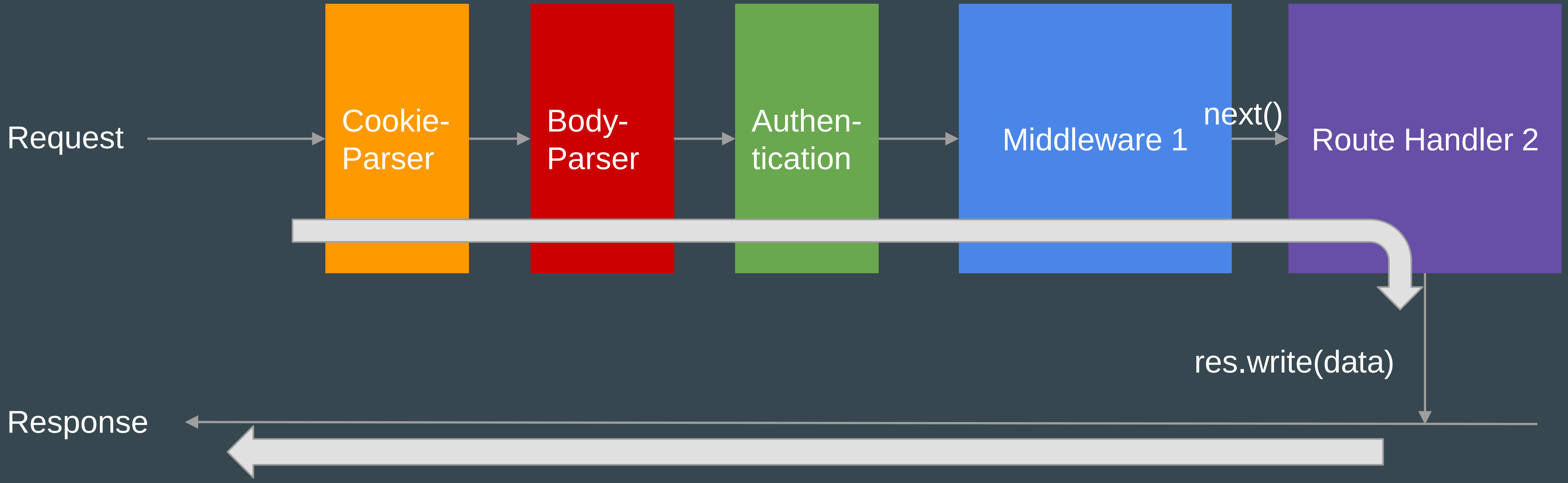
Express architecture explained

If the developer starts to write to the response in Middleware 1, Router handler is going to be omitted in this case.



Express architecture explained

If the developer calls the method **next()** in the middleware 1 , the req is going to continue to Route Handler 2 .

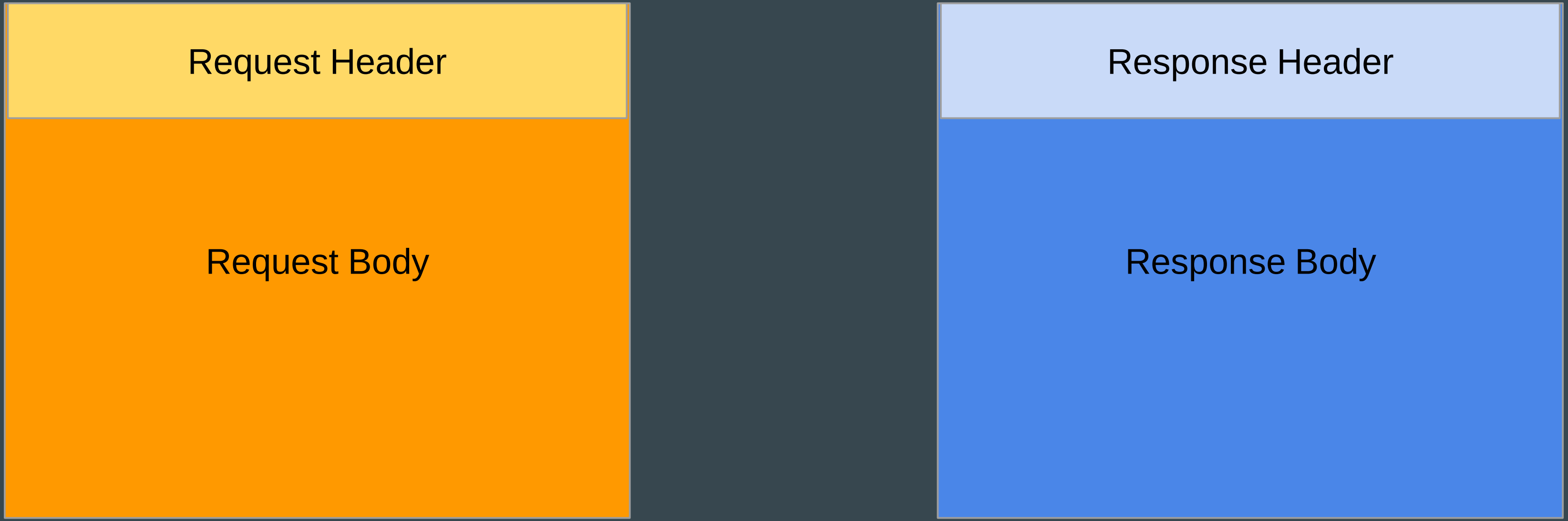


Getting Data from Request

	app.get('/user)	**app.post('/user)**	**app.get('/user/:id')**
Method	GET	POST	GET
How to get data	req.query	req.body (with the help of body-parser)	req.params
How to test in browser	Type in the address bar or form submission	Create a form and set the method to POST	Type in the address bar or form submission
How to test in Postman	Click the Params button next to the URL and input it to query string	Add to the request body and choose x-www-form-urlencode	Type directly as part of the URL
Example of url	/user**?id=1234**	/user	/user/**1234**

HTTP Request & Response Cycle

Both HTTP requests and responses are in plain text by default.
Any additional security has to be added by layers like SSL/TLS.



Content of HTTP request and response

The headers contains the metadata of request and responses and the body contains the content.

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: curl/7.47.0
Accept: */*
```

(The POST parameters goes to here)

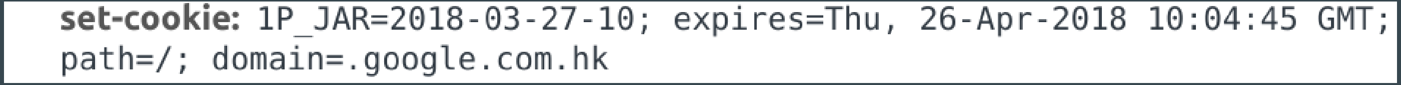
```
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Referrer-Policy: no-referrer
Location:
http://www.google.com.hk/?gfe_rd=cr&dcr=0&ei
=p67JWaiROJGQ4QLD-IiwDQ
Content-Length: 272
Date: Tue, 26 Sep 2017 01:34:31 GMT
```

```
<HTML><HEAD><meta http-
equiv="content-type"
content="text/html;charset=utf-8">
<TITLE>302
Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
</BODY></HTML>
```

Cookies

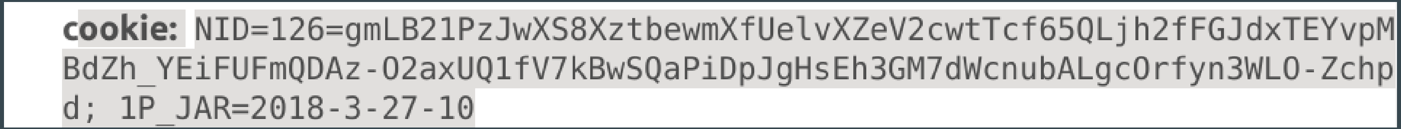
Cookies are key-values pairs of data stored on the **client side**. Browsers store cookies based on the hostname or domains.

For example, accessing google.com, here is the response header from Google



```
set-cookie: 1P_JAR=2018-03-27-10; expires=Thu, 26-Apr-2018 10:04:45 GMT;
path=/; domain=.google.com.hk
```

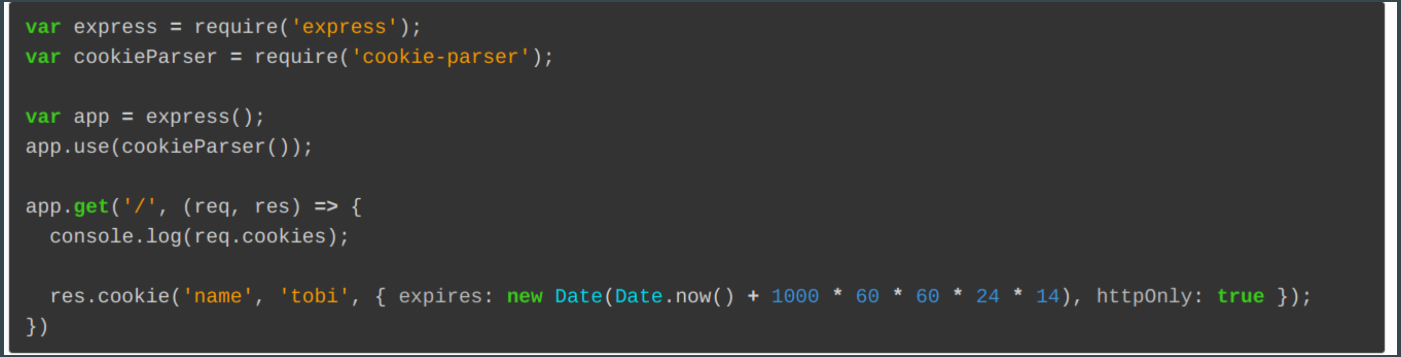
When you access google.com again, the following is set in your request headers.



```
cookie: NID=126=gmLB21PzJwXS8XztbewmXfUelvXZeV2cwtTcf65QLjh2fFGJdxTEYvpM
BdZh_YEiFUFmQDAz-O2axUQ1fV7kBwSQaPiDpJgHsEh3GM7dWcnubALgcOrfyn3WLO-Zchp
d; 1P_JAR=2018-3-27-10
```

Cookie-Parser

Cookie-Parser is a npm package that helps the developer to read and set the cookies by using the values req.cookies.

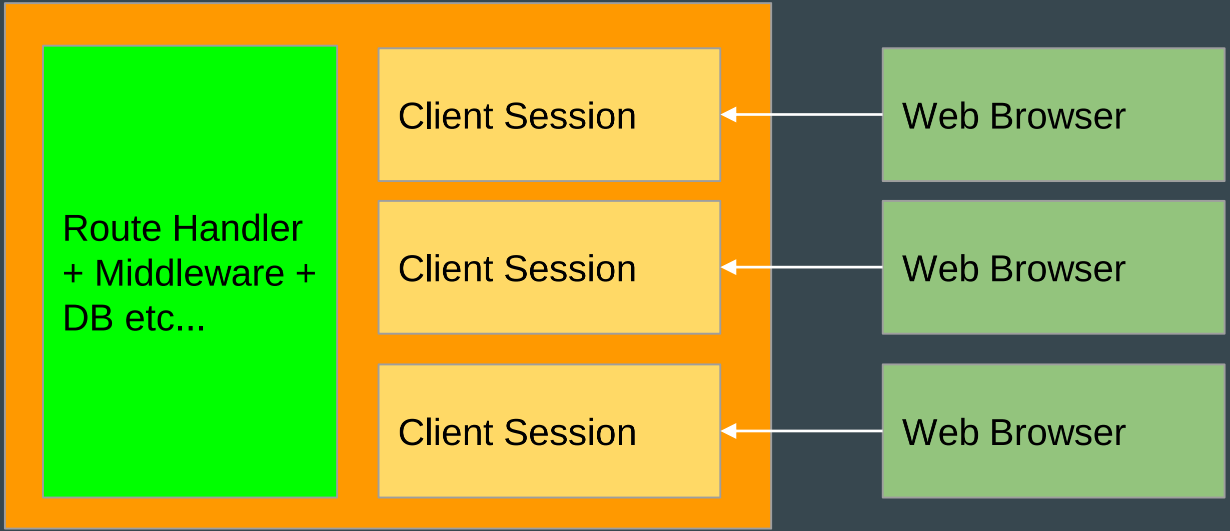


```
var express = require('express');
var cookieParser = require('cookie-parser');

var app = express();
app.use(cookieParser());

app.get('/', (req, res) => {
  console.log(req.cookies);

  res.cookie('name', 'tobi', { expires: new Date(Date.now() + 1000 * 60 * 60 * 24 * 14), httpOnly: true });
})
```

Please always remember that cookies only stores on the client-side , not server-side.

Session

Session is an important concept in modern web.
When we login to a website, we are initiating a new session **on server side**.
Each user (browser) would have a separate session to avoid accessing the wrong data.
All of the data related to a specific client should be stored in its session instead of ANY local variables in your route handler.



Cookie Session

Because cookies are stored in client side only. Thus it is ideal to identify the user to access the corresponding session of its own. To achieve this, you have to install **cookie-session** which already includes cookie-parsers.

```
var express = require('express');
var cookieSession = require('cookie-session');

var app = express();
app.use(cookieSession({
  name: 'session',
  secret: 'a hard to guess secret' ,

  // Cookie Options
  maxAge: 24 * 60 * 60 * 1000 // 24 hours
}))

app.get('/', (req, res) => {
  console.log(req.session);

  if(!req.session.id) {
    req.session.id = 'unique identifier'
  }

  findUser(req.session.id) // return information about the user
})
```

Q & A