GO IT

Node.JS #2

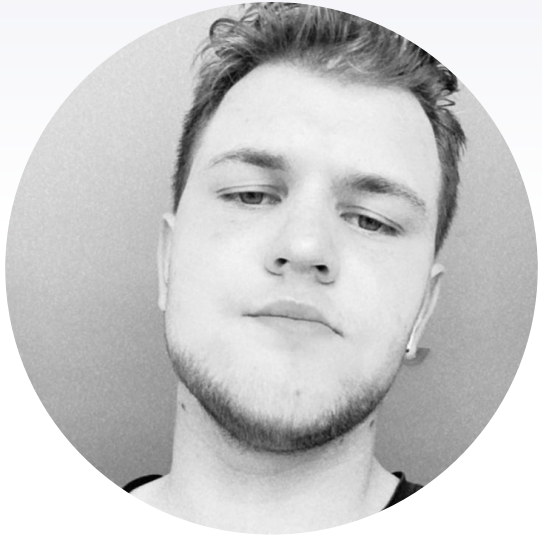
Yezhov Kyrylo

Software Engineer at EPAM systems

ezhov.kirill98@gmail.com

ezhov-kirill

ezhovkirill



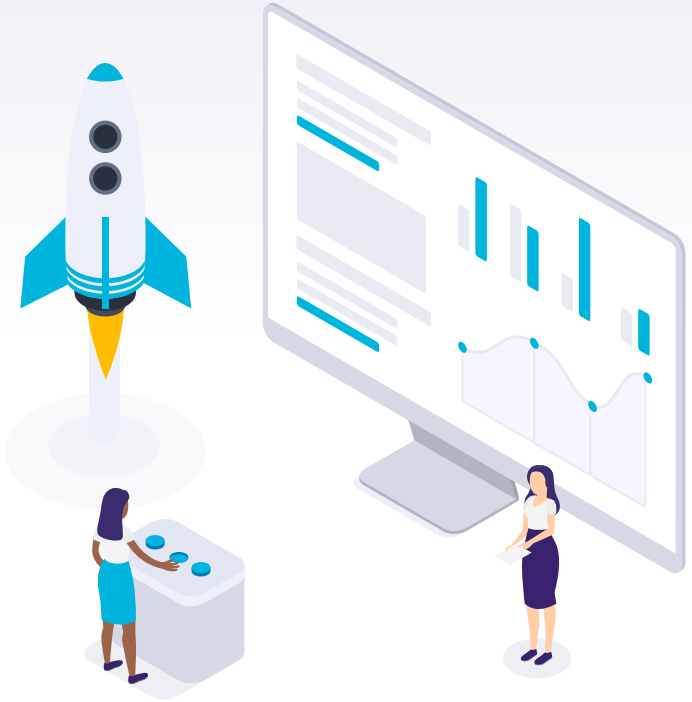
Regulations

- 12 lectures
- Questions anytime
- Initiative and involvement (Write to chat)

Agenda

- Questions
- Web architecture
- HTTP
- Express
- Questions

1 Web architecture

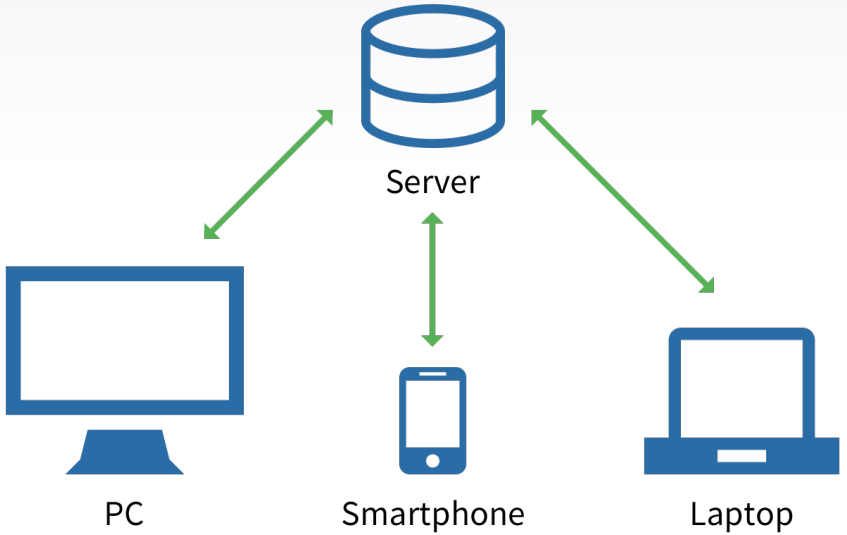


Client-server architecture

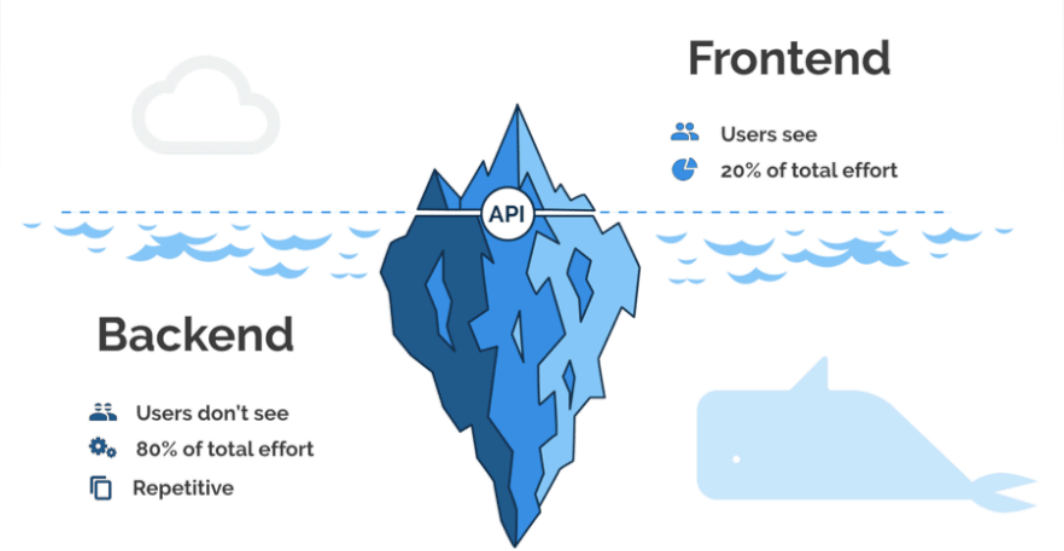
Client/server architecture is a producer/consumer computing architecture where the server acts as the producer and the client as a consumer.

The server houses and provides high-end, computing-intensive services to the client on demand.

These services can include application access, storage, file sharing, printer access and/or direct access to the server's raw computing power.



Client-server architecture

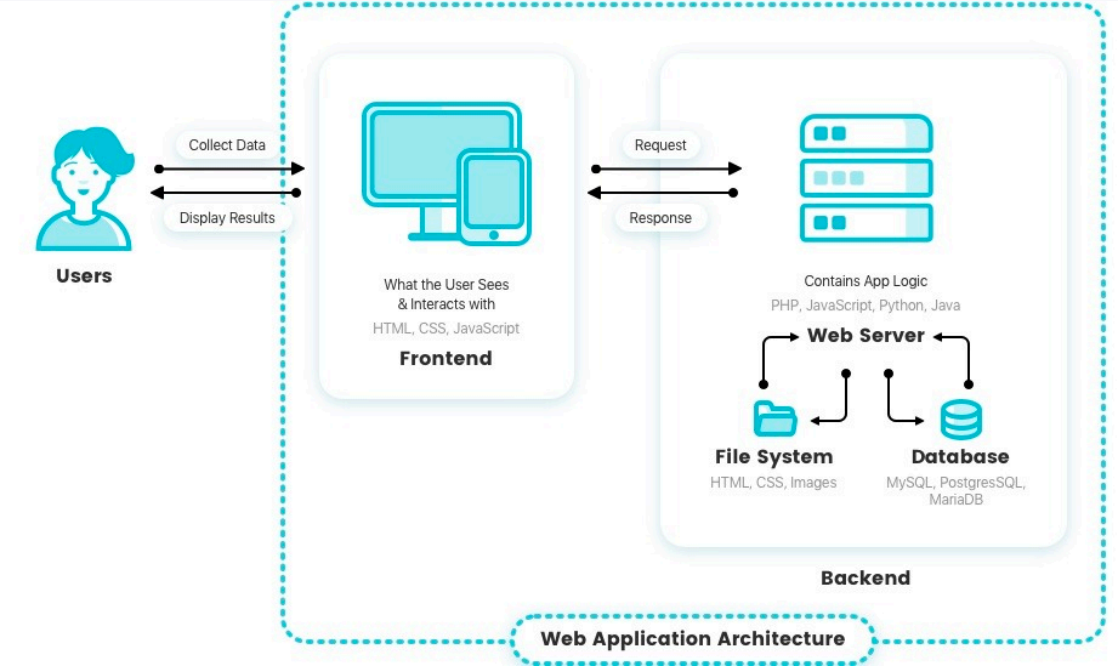


Client/server architecture works when the client computer sends a resource or process request to the server over the network connection, which is then processed and delivered to the client.

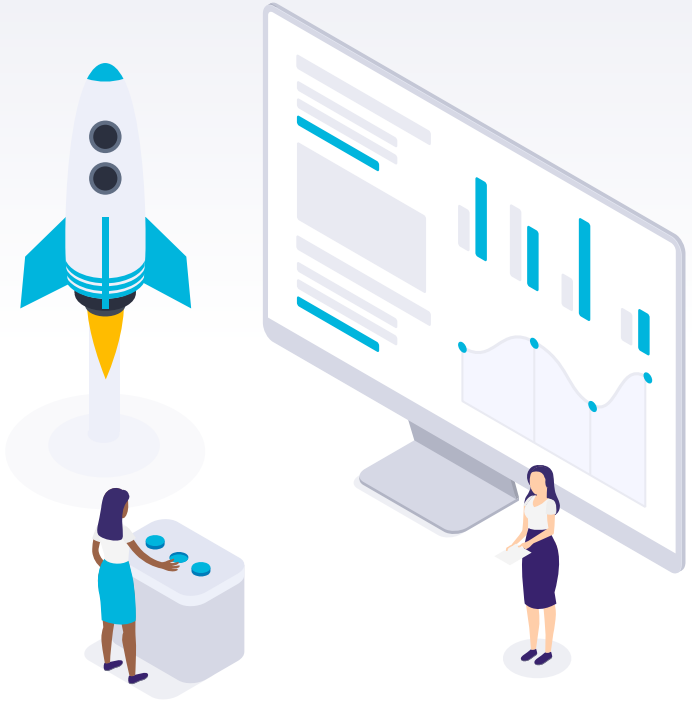
A server computer can manage several clients simultaneously, whereas one client can be connected to several servers at a time, each providing a different set of services.

In its simplest form, the internet is also based on client/server architecture where web servers serve many simultaneous users with website data.

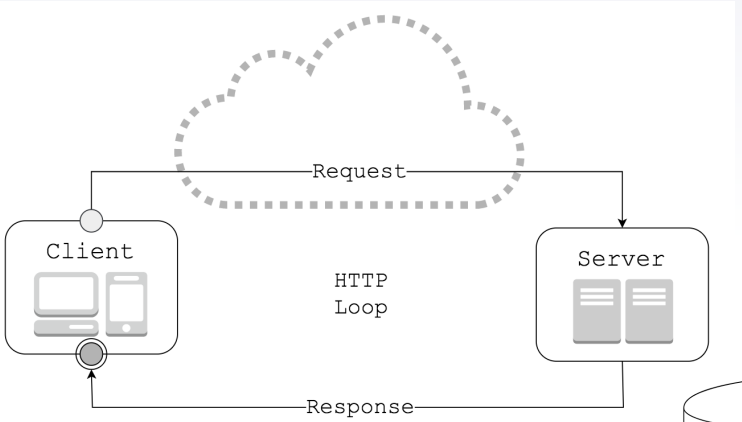
Modern web applications



1 HTTP



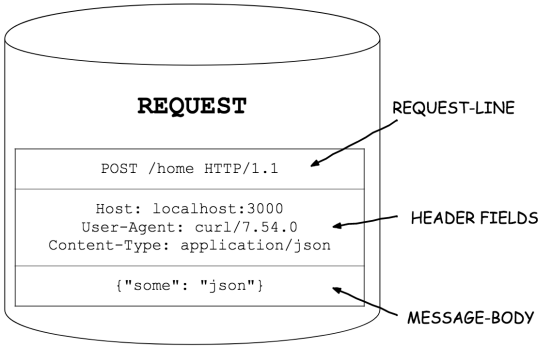
HTTP

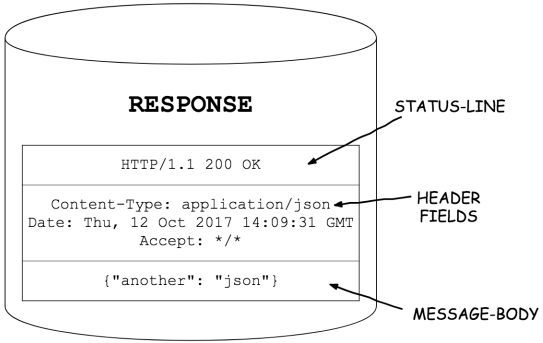


The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems.

This is the foundation for data communication for the World Wide Web (i.e. internet) since 1990.

HTTP is a generic and stateless protocol which can be used for other purposes as well using extensions of its request methods, error codes, and headers.





URI and URL

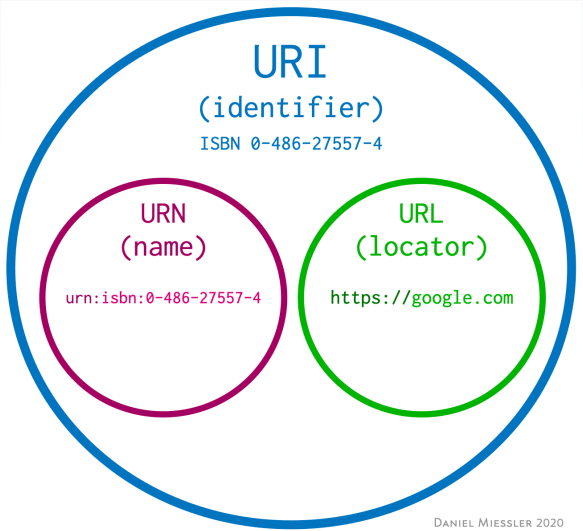
Uniform Resource Identifier

Uniform Resource Locator (Browser)

A Uniform Resource Identifier (URI) provides a simple and extensible means for identifying a resource (straight from RFC 3986). It's just an identifier; don't overthink it.

A URI is an identifier.

A URL is an identifier that tells you how to get to it.



HTTP

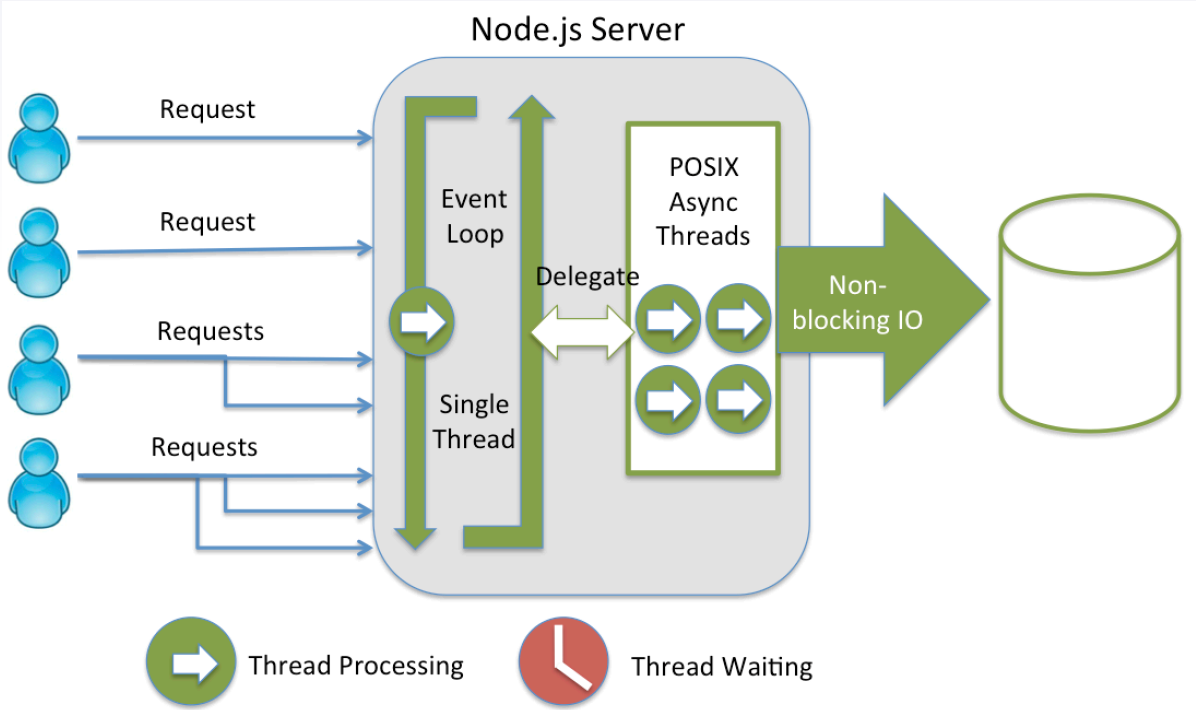
Built-in HTTP Module

```
var http = require('http');
```

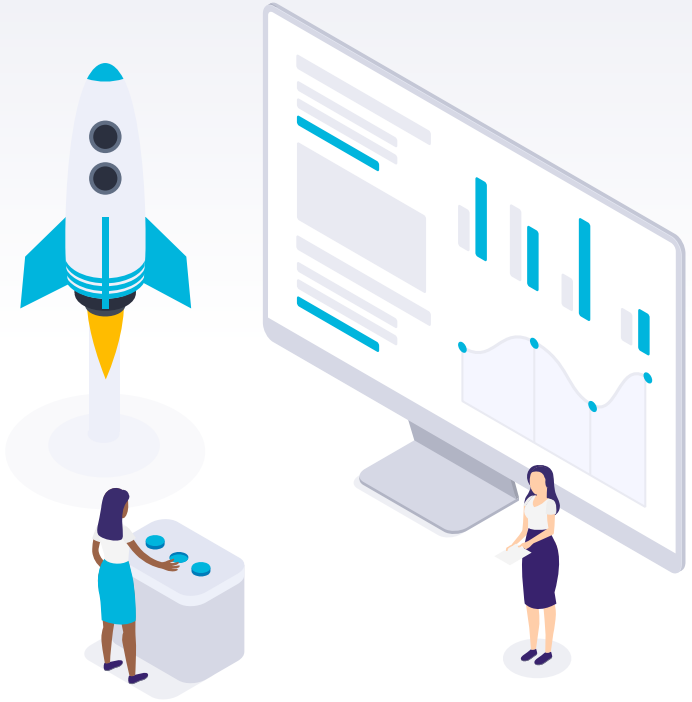
Now your application has access to the HTTP module, and is able to create a server:

```
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/
html'});
  res.end('Hello World!');
}).listen(8080);
```

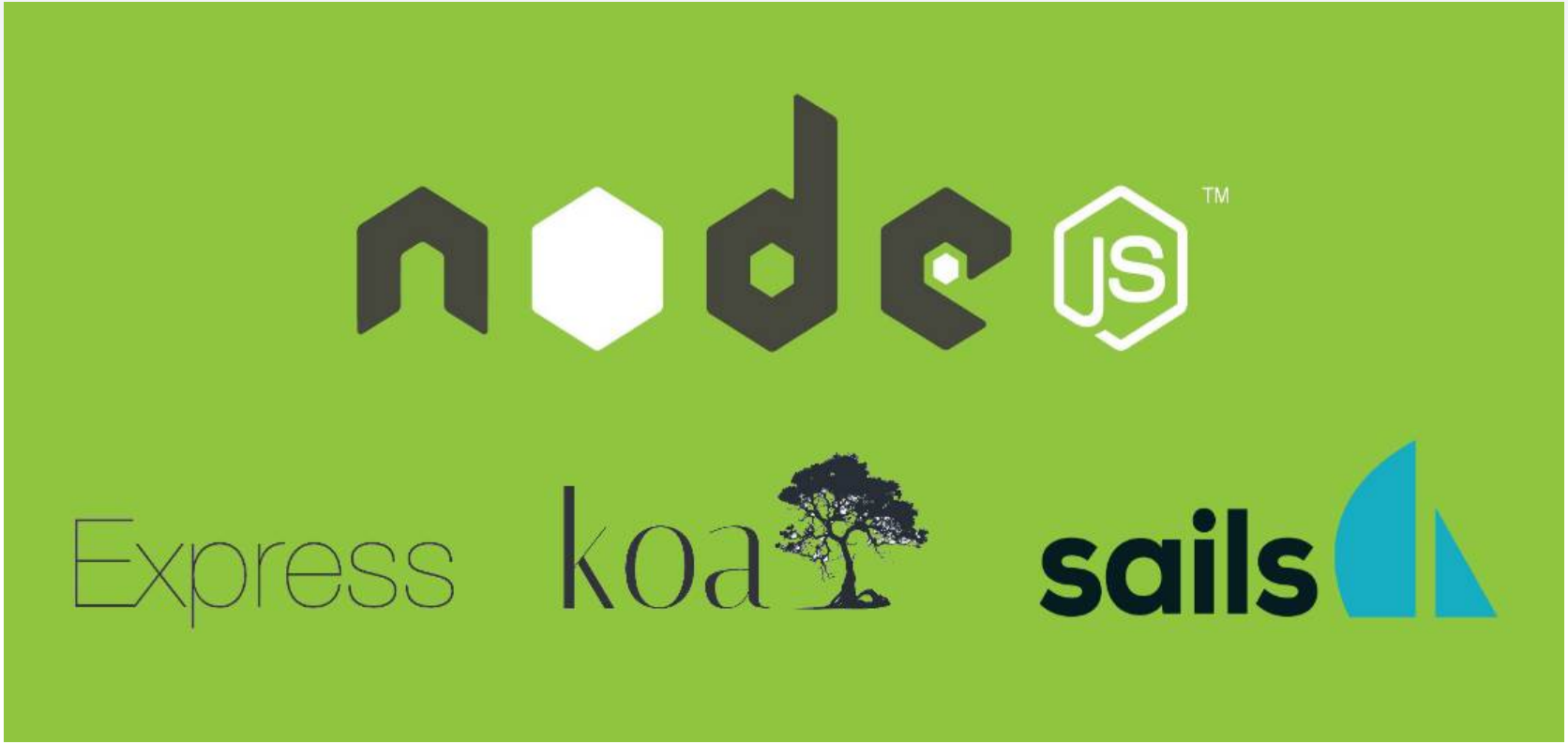
Webserver


Node.js Server
Request
Request
Requests
Requests
Event Loop
Single Thread
Delegate
POSIX Async Threads
Non-blocking IO
Thread Processing
Thread Waiting

1 Express



Frameworks



HTTP module in Node.JS

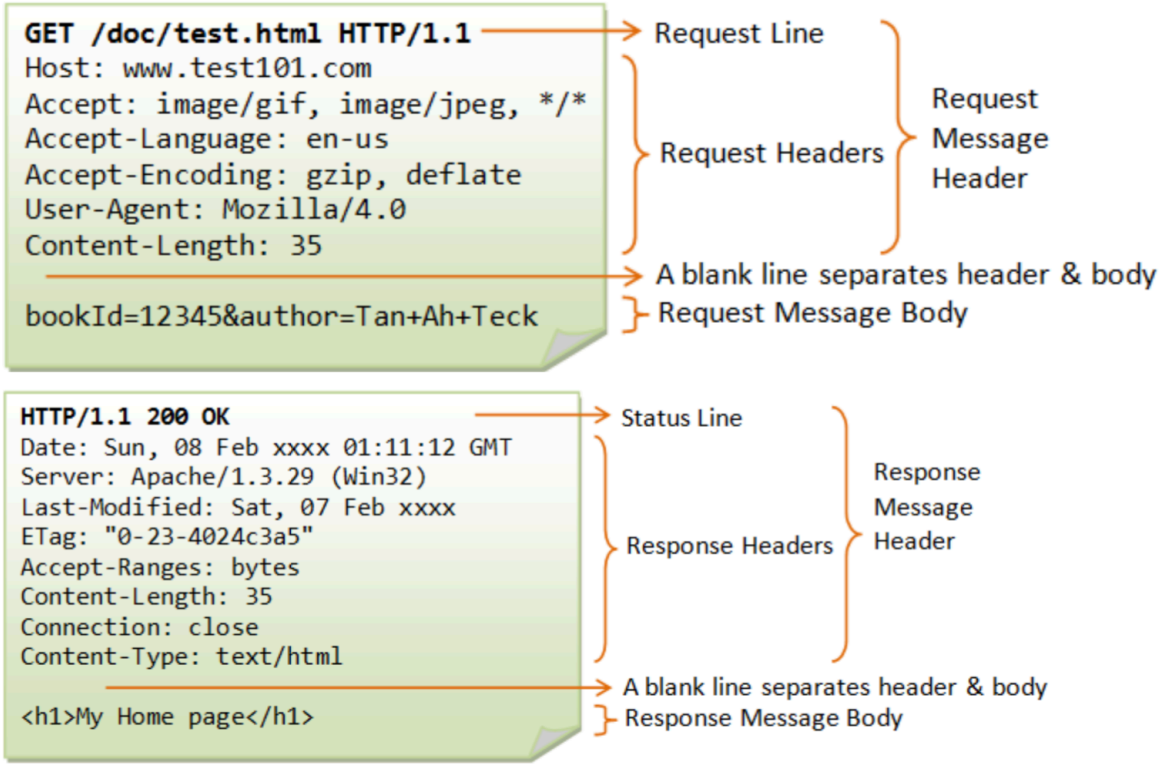
```
const http = require('http')
const port = 3000


const requestHandler = (request, response) => {
    console.log(request.url)
    response.end('Hello Node.js Server!')
}


const server = http.createServer(requestHandler)


server.listen(port, (err) => {
    if (err) {
        return console.log('something bad happened', err)
    }


    console.log(`server is listening on ${port}`)
})
```

Express.js

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

app.get('/', function (req, res) {
    res.json({ ok: true });
});

app.listen(3000);
```

Express.js 4.x API reference

- Express
- Application
- Request
- Response



RESPONSE

- Represents the HTTP response that an Express app sends for HTTP request
- Sending response:

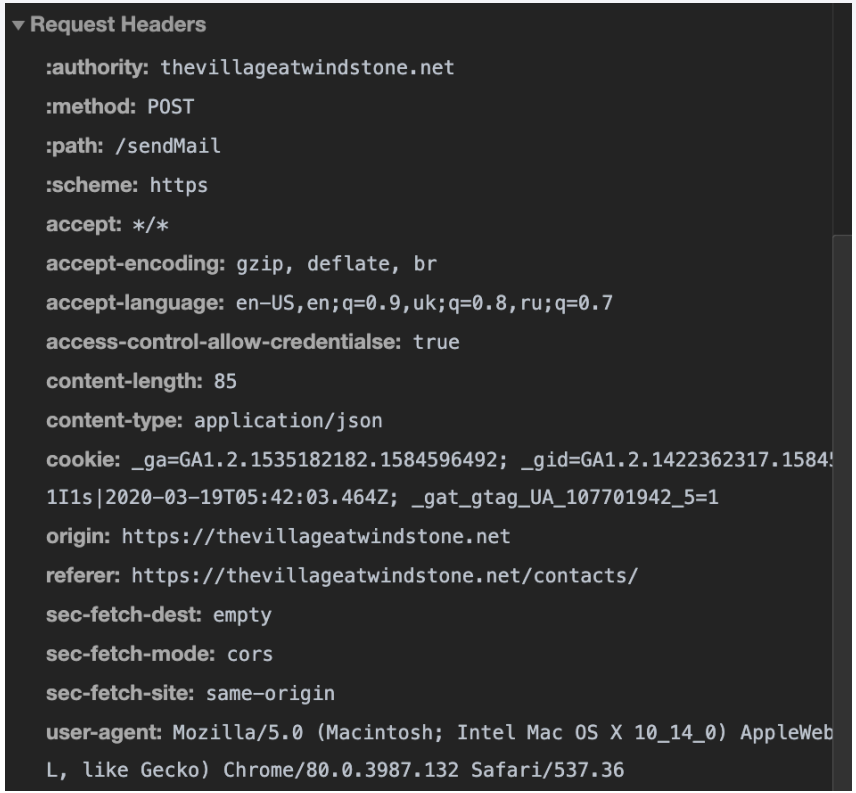
res.end()
res.sendStatus()
res.send()
res.sendFile()

res.json()
res.redirect()
res.render()

Mime types

```
.css - text/css
.csv - text/csv
.gif - image/gif
.html - text/html
.js - text/javascript
.json - application/json
.txt - text/plain
.xml - application/xml
Url encoded - application/x-www-form-urlencoded
```

Browser


Request Headers
:authority: thevillageatwindstone.net
:method: POST
:path: /sendMail
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9,uk;q=0.8,ru;q=0.7
access-control-allow-credentialse: true
content-length: 85
content-type: application/json
cookie: _ga=GA1.2.1535182182.1584596492; _gid=GA1.2.1422362317.1584
1I1s|2020-03-19T05:42:03.464Z; _gat_gtag_UA_107701942_5=1
origin: https://thevillageatwindstone.net
referer: https://thevillageatwindstone.net/contacts/
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: same-origin
user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_0) AppleWeb
L, like Gecko) Chrome/80.0.3987.132 Safari/537.36


Response Headers
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 10
Content-Type: application/json; charset=utf-8
Date: Tue, 02 Jul 2019 09:34:26 GMT

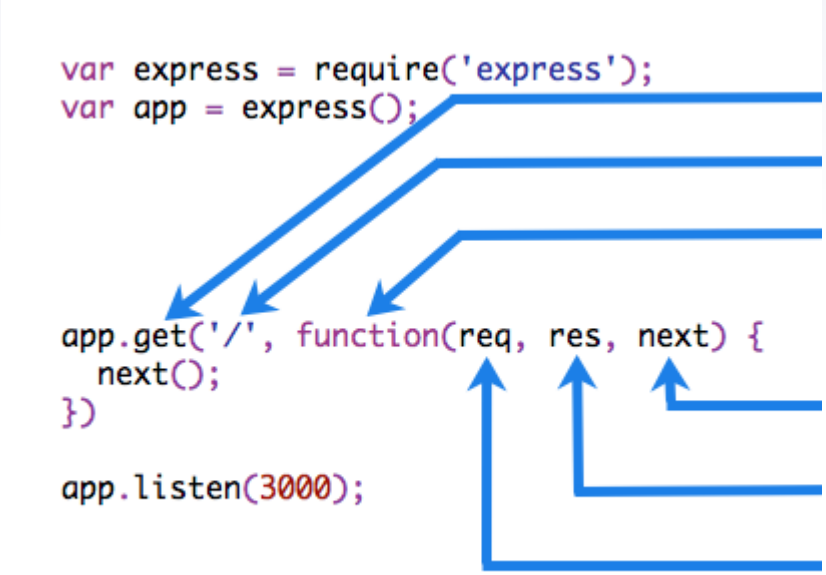
EXAMPLE

```
app.put('/employees/:id', async (req, res) => {
    if (!checkAuth(req.cookies.Authorization)){
        return res.status(401).send();
    }

    const employee = await getEmployeeById(req.params.id);

    if (employee === undefined) {
        return res.status(400).send();
    }

    await employee.update(req.body);
    res.json({ status: 'ok' });
});
```

What is Express

Express = Routing + Middlewares

Routing



HTTP method for which the middleware function applies

Path (route) for which the middleware function applies.

The middleware function.

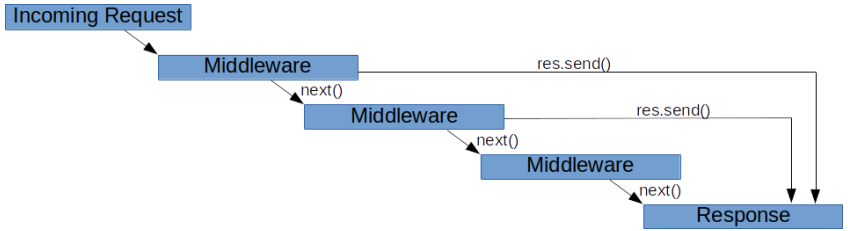
Callback argument to the middleware function

HTTP response argument to the middleware function, called "res" by convention.

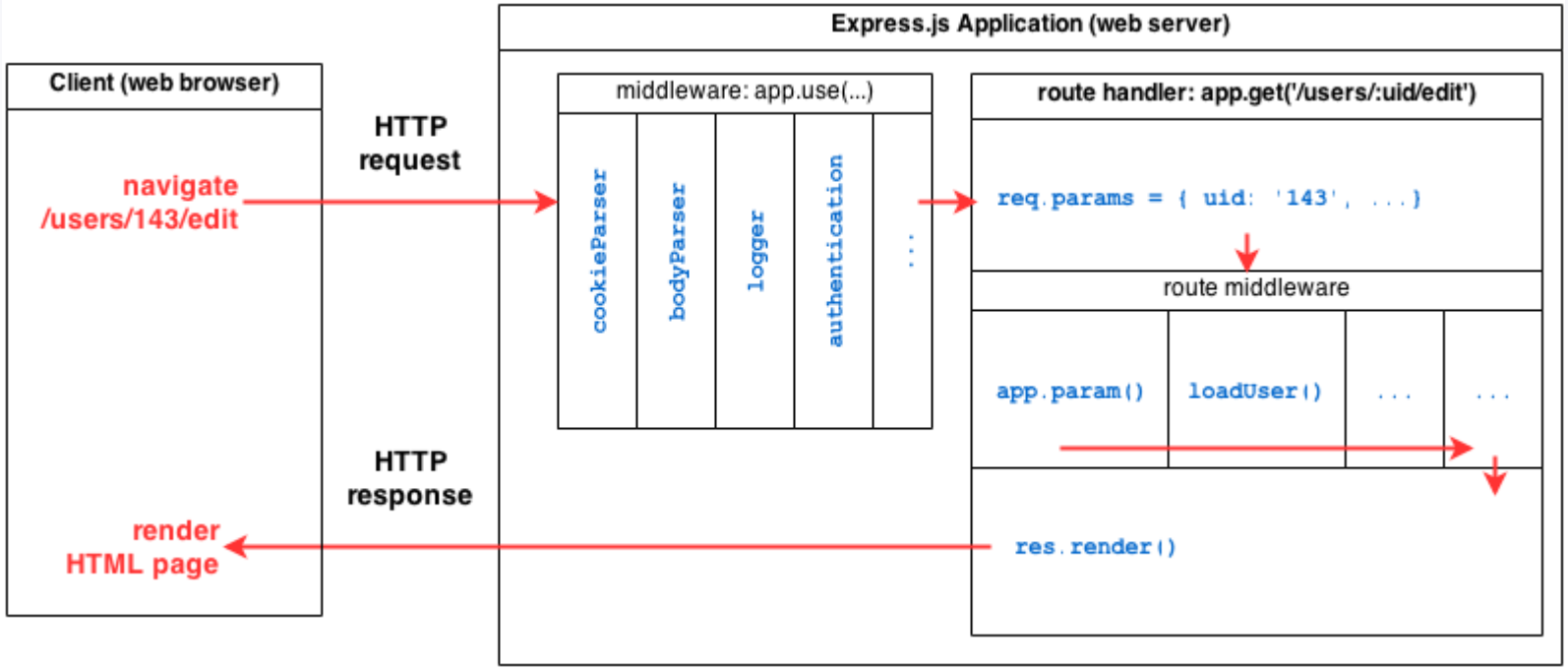
HTTP request argument to the middleware function, called "req" by convention.

Middleware

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



REQUEST -> RESPONSE CYCLE



Body, built-in middleware

- Contains submitted data as key-value pairs and undefined by default
- Use **express.json()** middleware to populate body from json

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

app.use(express.json());
```

- Use **urlencoded()** middleware to populate body from x-www-form-urlencoded

Static, built-in middleware

To serve static files such as images, CSS files, and JavaScript files, use the `express.static` built-in middleware function in Express.

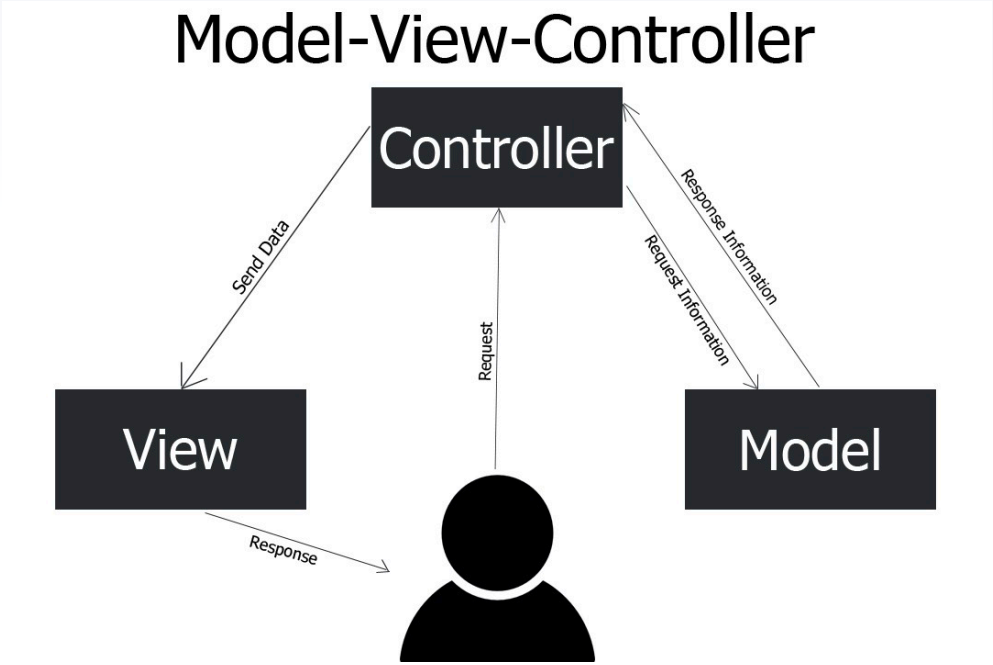
```
app.use(express.static('public'));
```

External middleware

Morgan - HTTP request logger middleware for node.js

```
var express = require('express')
var morgan = require('morgan')

var app = express()

app.use(morgan('combined'))

app.get('/', function (req, res) {
  res.send('hello, world!')
})
```

MVC


Model-View-Controller
Controller
Send Data
Request
Response Information
Request Information
View
Model
Response

Router

```
const express = require('express');
const app = express();
const router = express.Router();

router.use((req, res, next) => {
    // some middleware
    next();
});

router.get('/:id', (req, res) => {
    res.json({ id: req.params.id })
});

app.use('/users', router);
```

nodemon

Questions

Thanks!

ezhov.kirill98@gmail.com

ezhov-kirill

ezhovkirill