Junior phase

Express.js

An introduction to express.js

Duration: 30 minutes
Q&A: 5 minutes by the end of the lecture

Node is great!

Just Look!

```
var fs = require('fs'),http = require('http');
var server = http.createServer(function (request, response) {
    if (request.url === '/logo') {
        response.writeHead(200, { 'Content-Type': 'image/gif' });
        fs.readFile(__dirname + '/public/logo.png', function (err, data) {
            if (err) console.log(err);
            response.end(data);
        });
    } else if (request.url === '/') {
        response.writeHead(200, { 'Content-Type': 'text/html' });
        fs.readFile(__dirname + '/public/index.html', function (err, data) {
            if (err) console.log(err);
            response.end(data);
        });
    }
});  /// there's more that won't fit on this slide...
```

Node is great!

(except for the parts that aren’t !)

What's wrong with bare Node?

Node is complex! Because of that complexity, you will find yourself writing a lot of code, which means you're likely to introduce more errors.

What if we could do this instead?

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

var app = express();

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

app.listen(3000, () => {
      console.log(`Example app listening at
http://localhost:3000`);
});
```

Express is to Node
what
jQuery is to the Browser

What is Express?

Express is a web framework for Node that is lightweight and unopinionated which leverages middleware to help you create robust applications.

Express is a web framework for Node

A web framework provides an abstraction for all the common boilerplate code that is needed to develop dynamic web applications and services.

That is lightweight and unopinionated...

Express does a few useful things out-of-the-box.

Rather than bloat the framework with more features, Express' functionality is meant to be extended by middleware.

Which leverages middleware...

Middleware are, in essence, a chain of request handler functions.

They perform work on the request and response objects, then pass things along to the next thing.

Middleware: An Example

Let's say we want to write a simple logger middleware.

logger.js

```
var loggify = function (request, response, next) {
 console.log("Request: " + request.method + " at " +
request.url);
 next();
};
module.exports = loggify;
```

Middleware: An Example

Let’s say we want to write a simple logger middleware.

logger.js

```
var loggify = function (request, response, next) {
 console.log(“Request: ” + request.method + “ at ” +
request.url);
 next();
};
module.exports = loggify;
```

Middleware: An Example

Let’s say we want to write a simple logger middleware.

logger.js

```
var loggify = function (request, response, next) {
 console.log(“Request: ” + request.method + “ at ” +
request.url);
 next();
};
module.exports = loggify;
```

Middleware: An Example

Let’s say we want to write a simple logger middleware.

logger.js

```
var loggify = function (request, response, next) {
 console.log("Request: " + request.method + " at " +
request.url);
 next();
};
module.exports = loggify;
```

Middleware: An Example

Let’s say we want to write a simple logger middleware.

logger.js

Service.js

```
var express = require('express')
var loggify = require('./loggify');

var app = express();

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

Middleware: An Example

Let’s say we want to write a simple logger middleware.

logger.js

Service.js

```
var express = require('express')
var loggify = require('./loggify');

var app = express();

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

Middleware: An Example

Let’s say we want to write a simple logger middleware.

logger.js

Service.js

```
var express = require('express')
var loggify = require('./loggify');

var app = express();

app.use('/dashboard', loggify);
app.use(express.static('./public'));
```

Which leverages middleware…

Express comes with some middleware baked in, such as express.static to serve static files.

Typically, you'll find and install Express middleware via npm.

To help you create robust applications

You can leverage middleware to build out routes and endpoint API using Http Verbs

Service.js

```
var express = require('express'),loggify = require('./loggify');
var app = express();
app.get('/users', function(req, res, next){
// list all of the users
});
app.post('/users', function(req, res, next){ // add a new user };
app.use(loggify);
```

CodingLab
Powered by RBK

That’s it