Express

A Node.js framework

Content

What is Express?

A routing and middleware Web framework.

Middleware is a function with access to the

- request object
- response object
- next middleware in app’s request-response cycle

So, an express callback might look like `function(req, res, next)`

Middleware capabilities

- Execution of code.
- Making changes to request and response objects.
- End the request-response cycle.
- Call the next middleware in the stack.

NOTE: If the current middleware does not end the request-response cycle, it must call next() to pass control, otherwise the request will be left hanging.

Installing Express in your Node app

1. Open your terminal
2. Navigate to the directory of your Node project
3. Type `npm install express --save`

NOTE: After installing node, you'll notice a new folder in your project named *node_modules*. This is where all the Node packages you install via npm typically reside.

PRO TIP: Add node_modules to your .gitignore file.

Hello World example

app.js

```
var express = require('express'); // import express
var app = express();

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

var server = app.listen(3000, function() {
    var host = server.address().address;
    var port = server.address().port;
    console.log('App listening at http://%s:%s', host, port);
});
```

The same request and response objects we used in our last Node app.

app.get

```
// GET requests to our server's root (/), will respond with
// "Hello World!"
// We call / a route
app.get('/', function (req, res) {
    res.send('Hello World!');
});
```

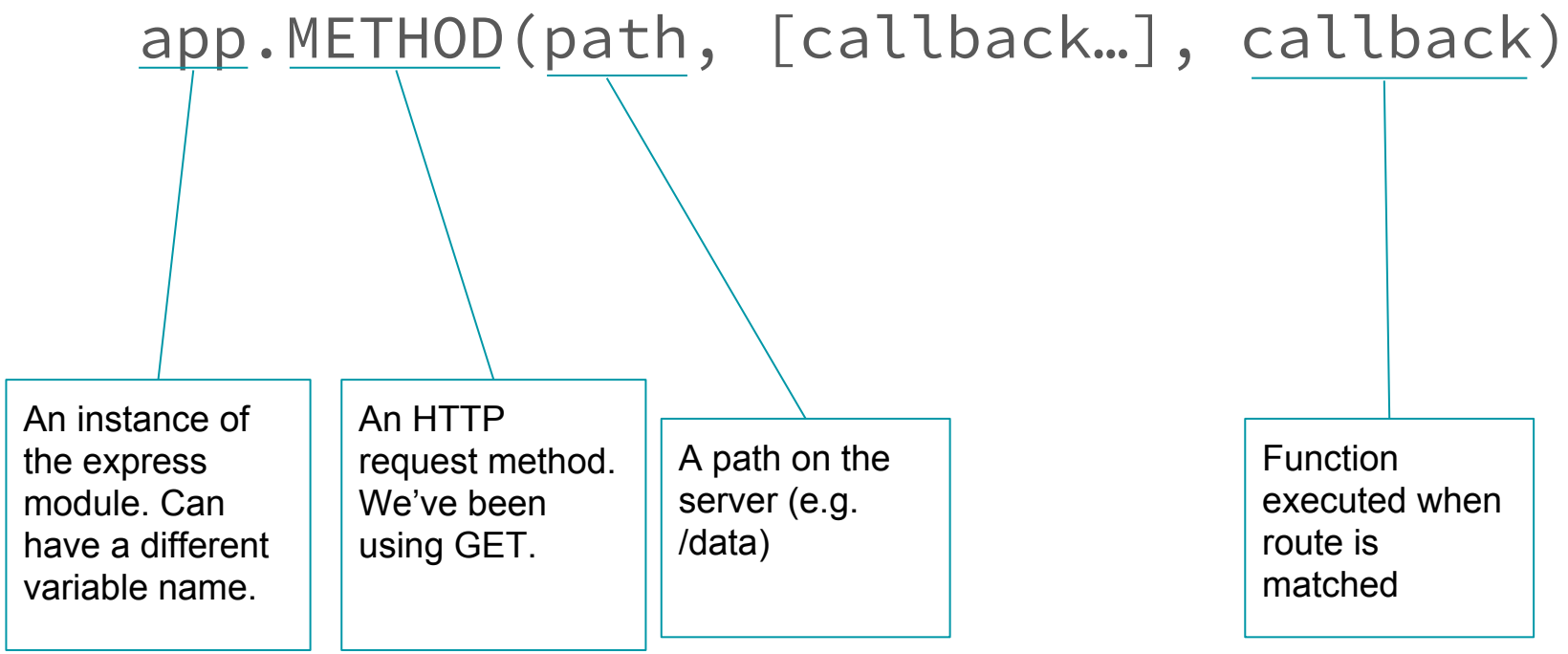
This is part of the code from the previous slide.

Routing

What is routing?

The definition of end-points (URIs) to an application and how it responds to client requests

Defining a route



Route path examples

```
// will match request to the root
app.get('/', function (req, res) {
  res.send('root');
});

// will match requests to /about
app.get('/about', function (req, res) {
  res.send('about');
});
```

Route path even more examples

```
// will match anything with an a in the route name
app.get('/a/', function (req, res) {
  res.send('/a/');
});

// will match butterfly, dragonfly, but not butterflyman or dragonflyman
// and other words that end in fly
app.get('/.*fly$/', function (req, res) {
  res.send('/.*fly$/');
});
```

Route handlers

How we use route handlers

- Provide multiple callback functions that behave like middleware to handle request
- To impose pre-conditions on a route
- To pass control to subsequent routes
- Can be in form of
  - function
  - array of functions
  - combinations of both

Route handler with >1 callbacks

```
app.get('/example/b', function (req, res, next) {
  console.log('response will be sent by the next function ...');
  next();
}, function (req, res) {
  res.send('Hello from B!');
});
```

Route with an array of callbacks

```
var cb0 = function (req, res, next) {
  console.log('CB0');
  next();
}

var cb1 = function (req, res, next) {
  console.log('CB1');
  next();
}

var cb2 = function (req, res) {
  res.send('Hello from C!');
}

app.get('/example/c', [cb0, cb1, cb2]);
```

Resources

- Express response methods
- Express Router object