Full Stack - Session 4

Express

Fast, unopinionated, minimalist web framework for Node.js

Topics

- Node as a Web Server - Revisited
- Express & Middleware
- Express Router & Generator
- Template Engines

Web Server and HTTP

TCP/IP

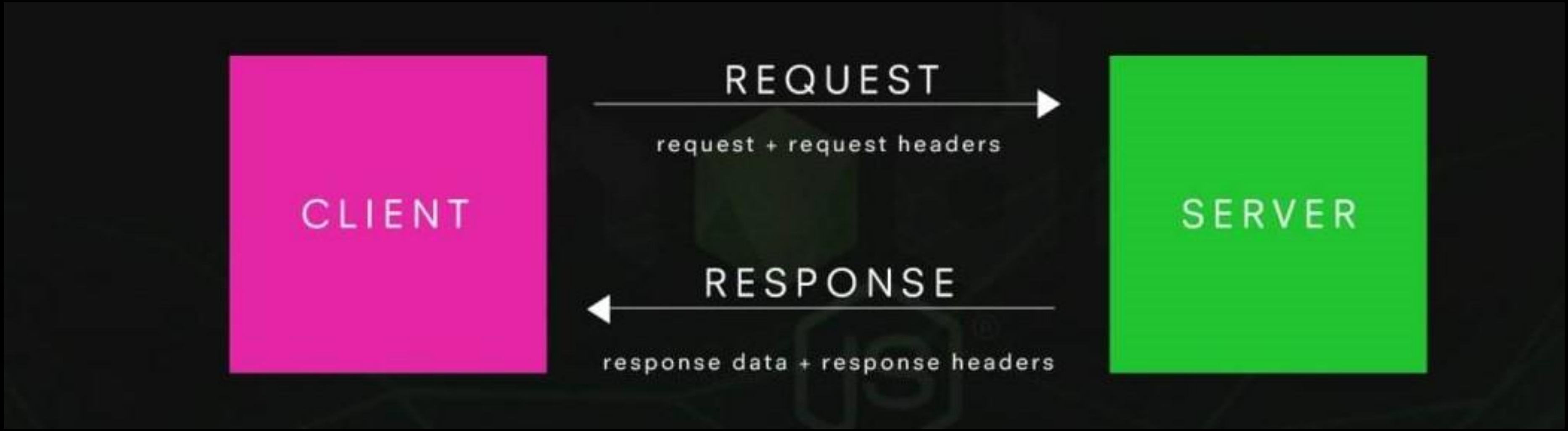
- Stands for Transmission Control Protocol/Internet Protocol
- These two protocols were developed in the early days of the internet by U.S Military
- IP refers to the moving of data packets between nodes.
- It is the foundation of the Internet

SOCKET
PACKETS

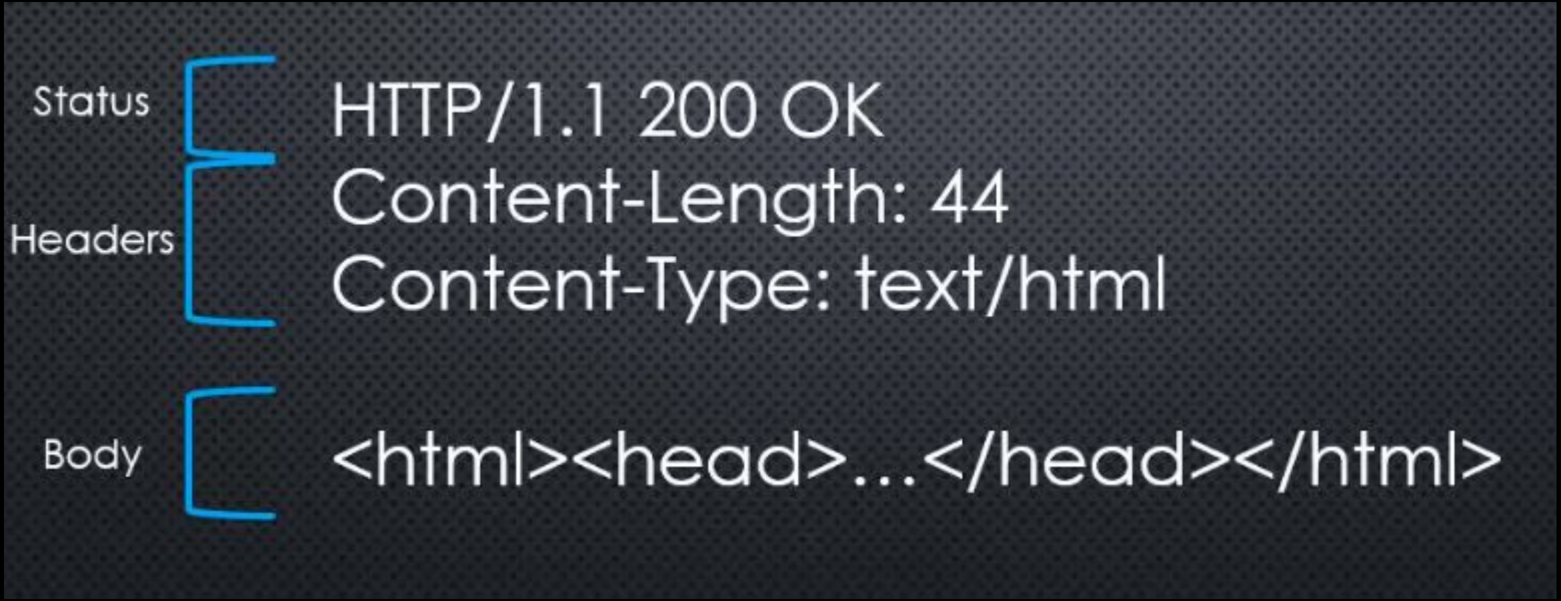
HTTP

- A set of rules (and format) for data being transferred on the web
- It stands for Hypertext Transfer Protocol
- It’s a format of defining data being transferred via TCP/IP

Response Headers

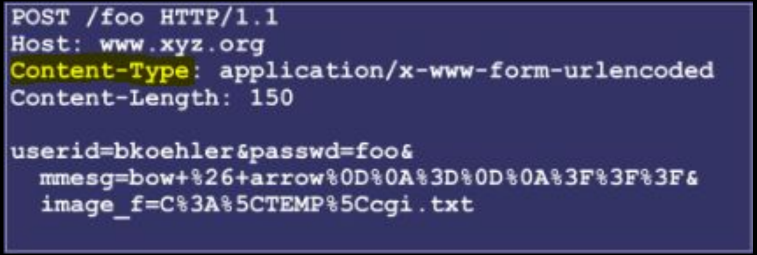


Response Header

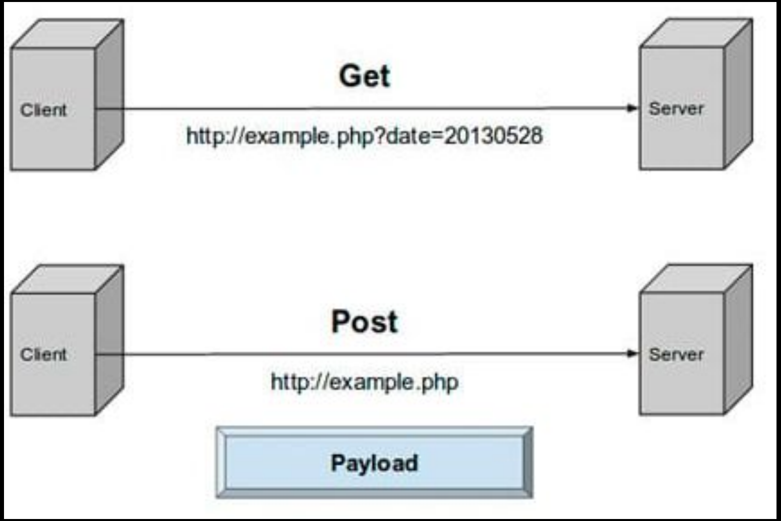


```
HTTP/1.1 200 OK
Content-Length: 44
Content-Type: text/html

<html><head>...</head></html>
```

POST vs GET HTTP Requests



```
POST /foo HTTP/1.1
Host: www.xyz.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 150

userid=bkoehler&passwd=foo&
  mmesg=bow+%26+arrow%0D%0A%3D%0D%0A%3F%3F%3F&
  image_f=C%3A%5CTEMP%5Ccgi.txt
```



- GET is a method that sends information by appending to page request
- POST is a method that transfers information via HTTP header
  - * Payload = QueryString is actually moved to the body of the message.

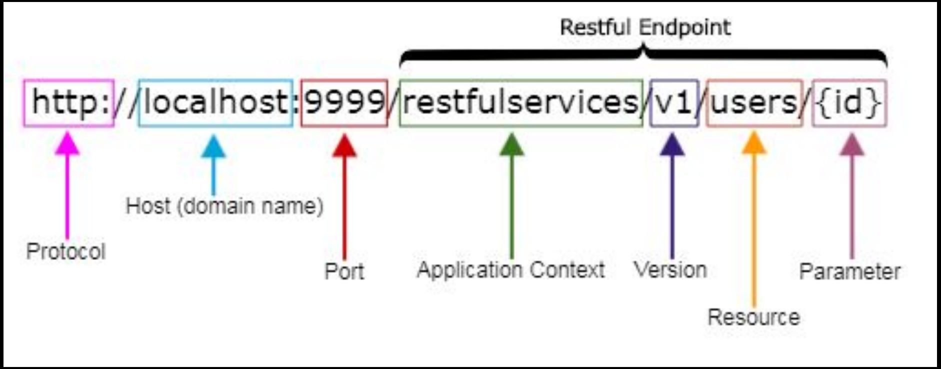
Video - Rest API

API

- Stands for Application Programming Interface
- It allows one piece of software to interact with another piece of software
- This interface is made available on the web via a set of URLs
- These URLs only accept and send data via HTTP and TCP/IP

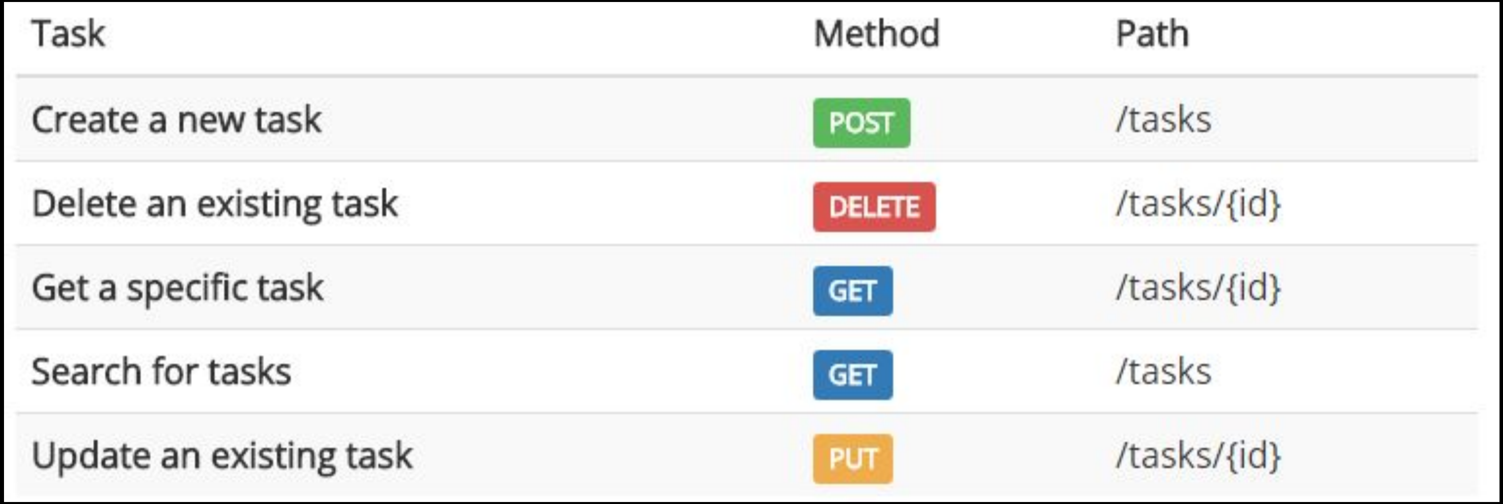
Endpoint

- One URL in a Web API
- Sometimes that endpoint (URL) does multiple things by making choices based on the HTTP request headers.
- Endpoints give and receive data in multiple formats, the most popular being JSON data.



REST - Representational State Transfer

- REST is an architectural style for building APIs
- The HTTP verbs and URLs have meaning.
- We organize and build our APIs to use HTTP verbs and URLs to match HTTP Requests in a meaningful way.



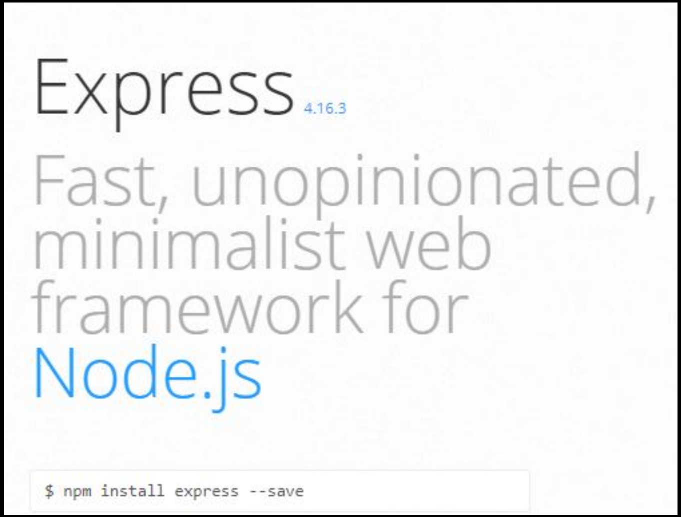
Task	Method	Path
Create a new task	POST	/tasks
Delete an existing task	DELETE	/tasks/{id}
Get a specific task	GET	/tasks/{id}
Search for tasks	GET	/tasks
Update an existing task	PUT	/tasks/{id}

Routing

- **Mapping HTTP Request to Content**
  - *(Whether actual files exist on server or not)*

Express

Express for Node.js



Express 4.16.3

Fast, unopinionated, minimalist web framework for Node.js

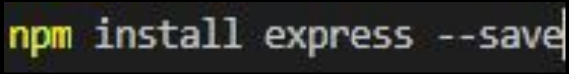
```
$ npm install express --save
```

http://expressjs.com/

- web application framework, designed for building single-page, multi-page and hybrid web applications
- minimalist, yet full featured
- Built-in support for routing & varous HTTP handlers, configuration, session management and middleware
- amazing community

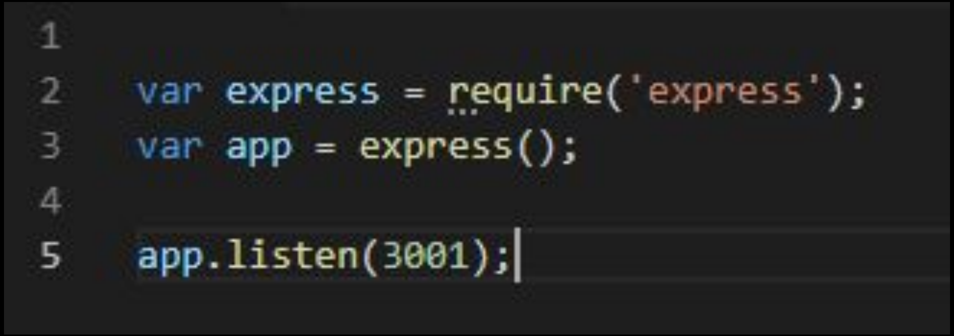
Installing & Setup Express

- Install Express on command line (development or global dependencies)



```
npm install express --save
```

- Create a new instance of the express object, set the port and it will begin listening



```

var express = require('express');
var app = express();

app.listen(3001);
```

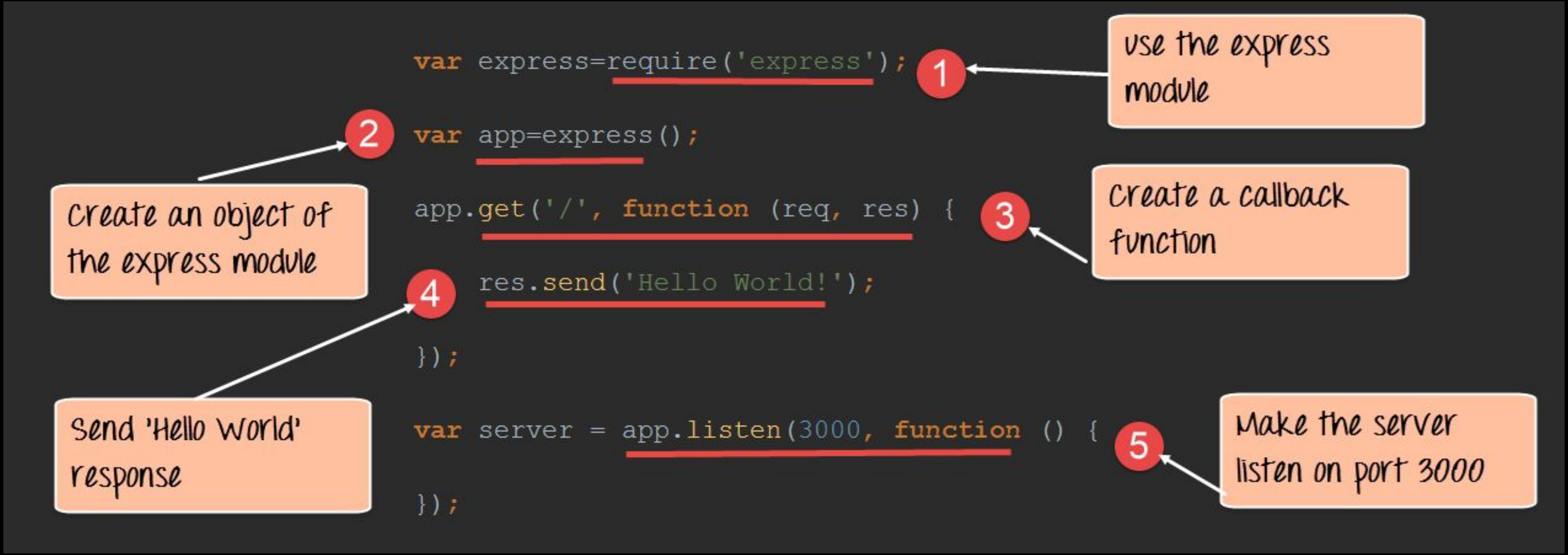
HTTP Method

- Specifies the type of action the request wishes to make

- **GET, POST, DELETE** etc. These are called verbs.

- Server responds to the verbs and takes action
  - GET - Fetches data
  - POST - Adds new data
  - PUT - Updates data
  - DELETE - Delete data

Using the Express module

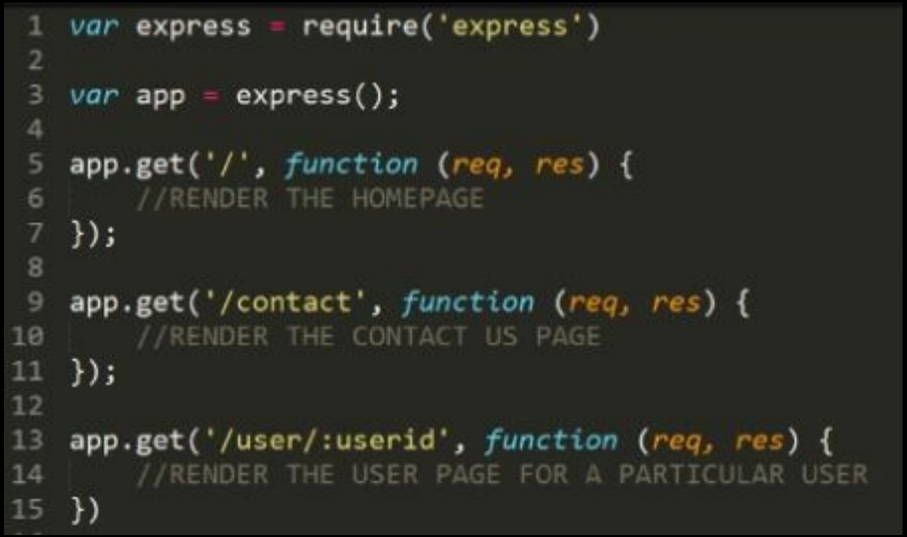

```
var express=require('express');

var app=express();

app.get('/', function (req, res) {

  res.send('Hello World!');

});

var server = app.listen(3000, function () {

});
```

1
use the express module
2
Create an object of the express module
3
Create a callback function
4
Send 'Hello World' response
5
Make the server listen on port 3000

Defining Routes

- Express apps can respond to various HTTP verbs as API methods

- Route can have the same name, as long as verb is different, it will be handled separately

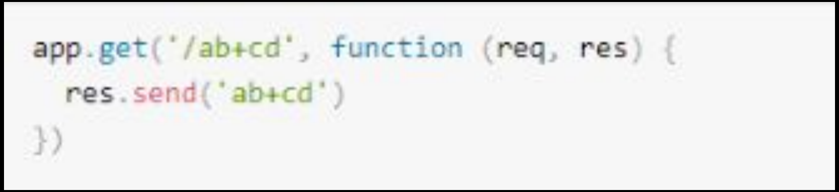


```
var express = require('express')

var app = express();

app.get('/', function (req, res) {
    //RENDER THE HOMEPAGE
});

app.get('/contact', function (req, res) {
    //RENDER THE CONTACT US PAGE
});

app.get('/user/:userid', function (req, res) {
    //RENDER THE USER PAGE FOR A PARTICULAR USER
})
```

Routing Paths

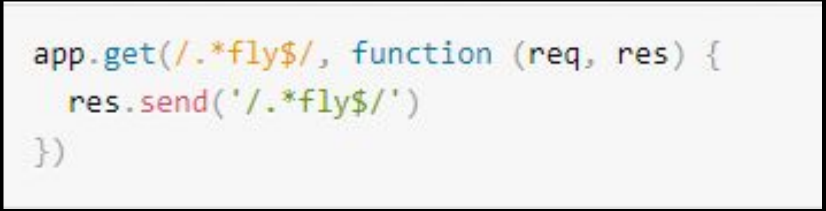
- Express will respond to HTTP verbs and routing, but will also provide routing matching
- Route Path Matching is utilities for pattern matches on the route path
  - http://expressjs.com/en/guide/routing.html
- Route paths can also be string patterns. String patterns use a subset of regular expression syntax to define patterns of endpoints that will be matched.

This route path will match abcd, abbcd, abbbcd and so on.



```
app.get('/ab+cd', function (req, res) {
  res.send('ab+cd')
})
```

This route path will match butterfly and dragonfly, but not butterflyman, dragonflyman, and so on.

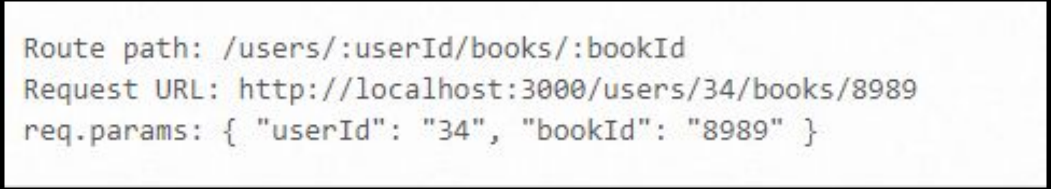


```
app.get(/.*fly$/, function (req, res) {
  res.send('/.*fly$/')
})
```

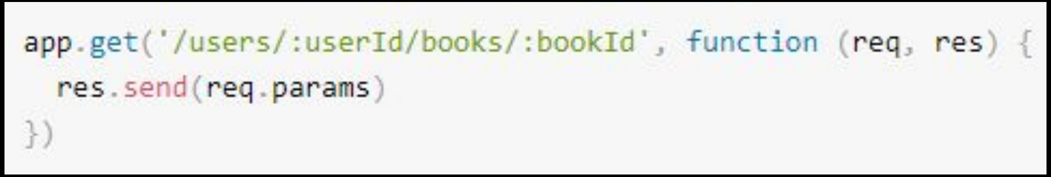
Video - Express

Route Parameters

- Route parameters are named URL segments that are used to capture the values specified at their position in the URL. The captured values are populated in the req.params object.



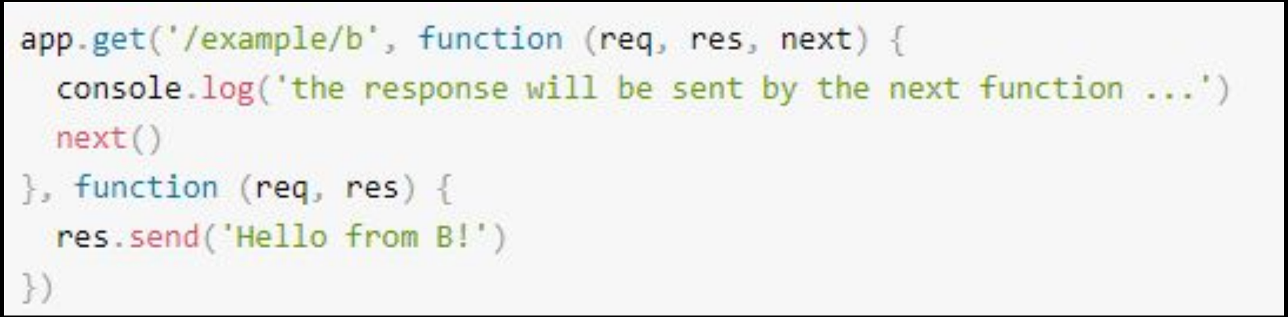
```
Route path: /users/:userId/books/:bookId
Request URL: http://localhost:3000/users/34/books/8989
req.params: { "userId": "34", "bookId": "8989" }
```



```
app.get('/users/:userId/books/:bookId', function (req, res) {
  res.send(req.params)
})
```

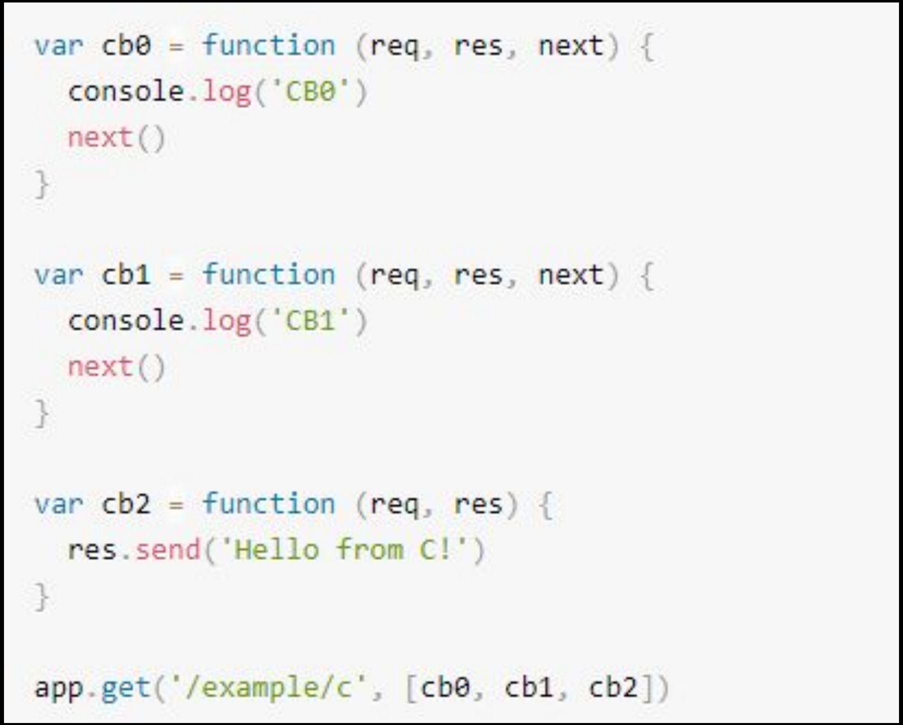
Route Handlers

- You can provide multiple callback functions that behave like middleware to handle a request. The only exception is that these callbacks might invoke next('route') to bypass the remaining route callbacks.



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

Route Handlers cont..

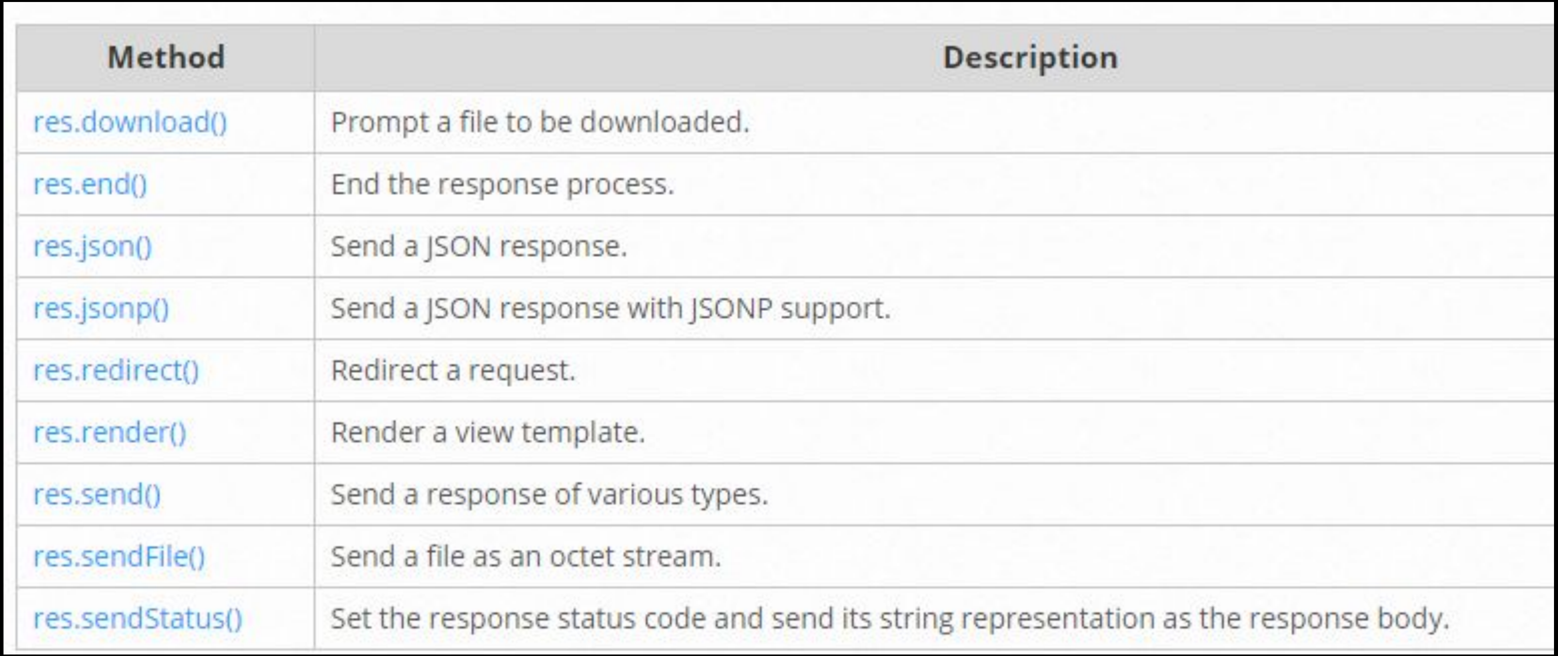


```
var cb0 = function (req, res, next) {
  console.log('CB0')
  next()
}

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])
```

- An array of callback functions can handle a route.

Response Methods

- The methods on the response object (res) can send a response to the client, and terminate the request-response cycle.



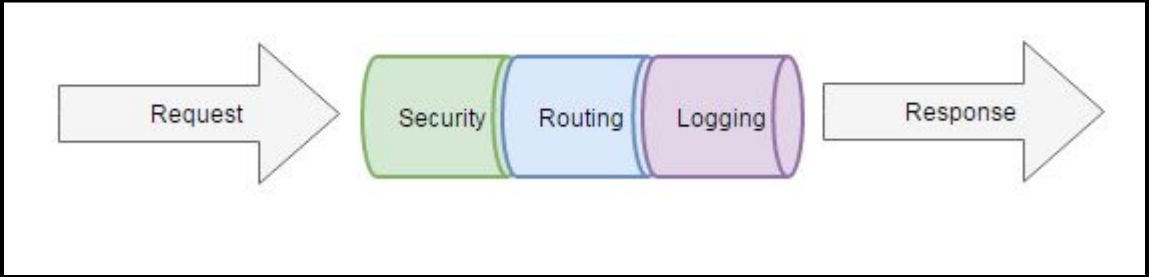
Method	Description
res.download()	Prompt a file to be downloaded.
res.end()	End the response process.
res.json()	Send a JSON response.
res.jsonp()	Send a JSON response with JSONP support.
res.redirect()	Redirect a request.
res.render()	Render a view template.
res.send()	Send a response of various types.
res.sendFile()	Send a file as an octet stream.
res.sendStatus()	Set the response status code and send its string representation as the response body.

Video - Express Route Params

Express & Middleware

Middleware

- Code that sits between two layers of software
- With Express, the middleware is sitting between the request and response



Middleware

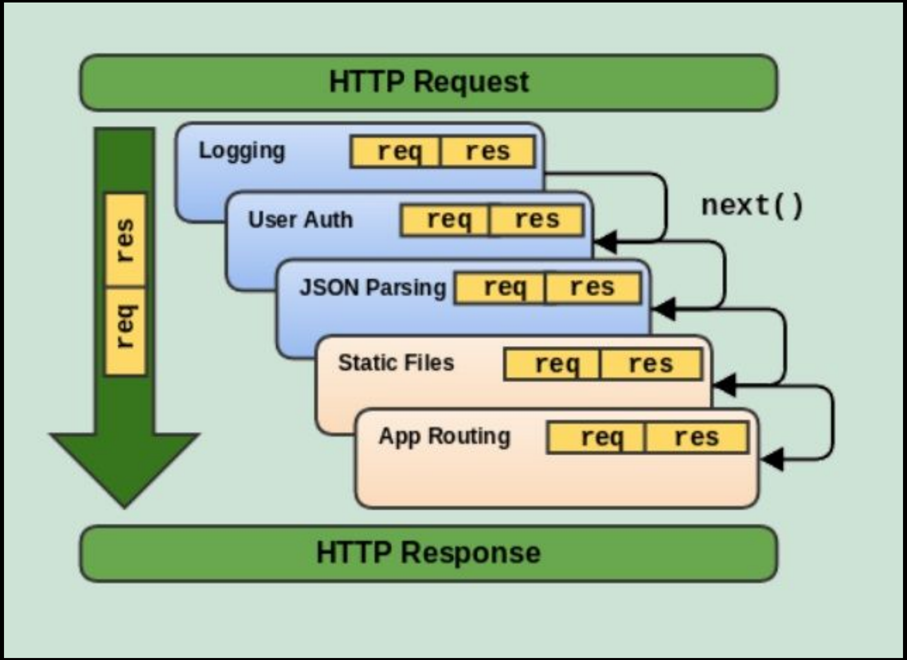
- Middleware is a pipeline of code that gets called before your request handler

- Express applications are basically a bunch of middleware calls

Middleware can:

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

Express Middleware



- Middleware is a function with access to the **request object (req)** and the **response object (res)**

- Also, has access to the **next middleware object (next)** in line in request-response cycle of Express application

Express Middleware modules

- Some useful Express maintained middleware and 3rd party middleware can be found here:
  - https://expressjs.com/en/resources/middleware.html

- Some popular middleware include:

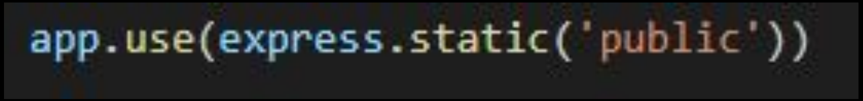
CookieParser	- parse cookie header
BodyParser	- parse the HTTP request body
Passport	- simple, unobtrusive authentication for Node.js

Serving Static Files with Express

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

- The function signature is **express.static**(root, [options]) where root is the root directory from which the serve the static assets.

- For example, to serve images, CSS files and JavaScript from the public directory use.



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

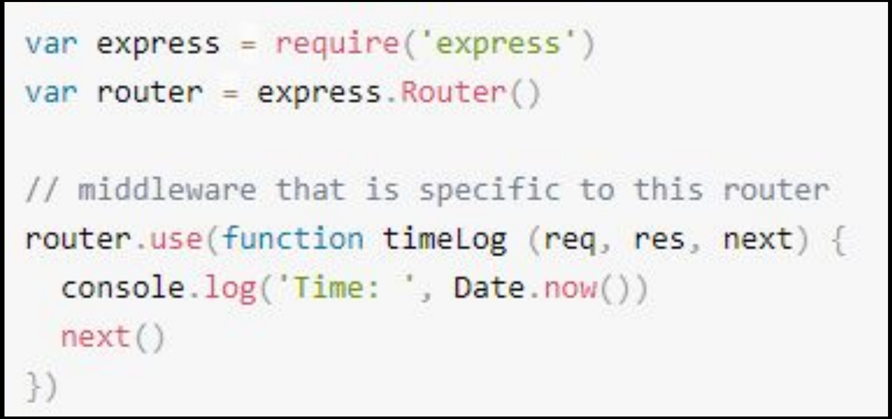
Video Express & Middleware

Express Router & Generator

Express Router

- Use the express.Router class to create modular, mountable route handlers.

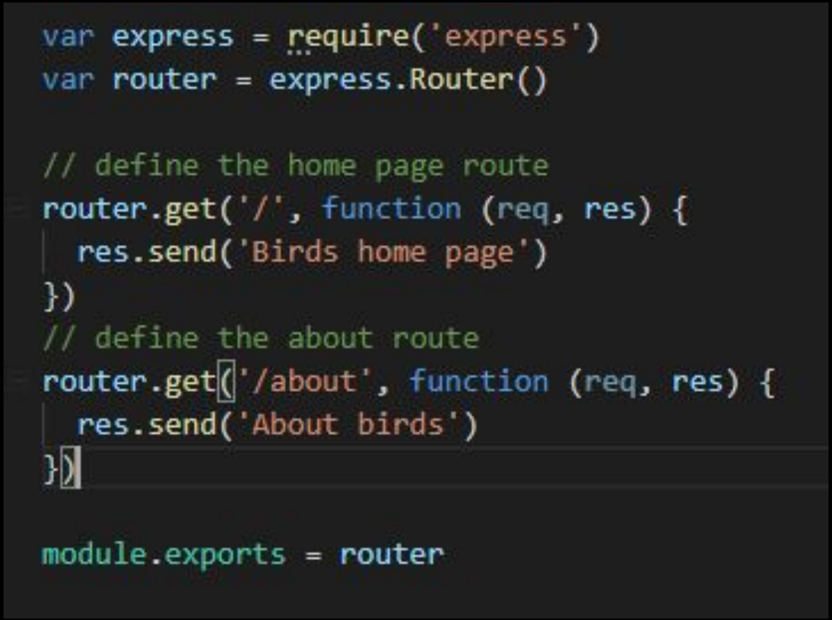
- A Router instance is a complete middleware and routing system; for this reason, it is often referred to as a “mini-app”.



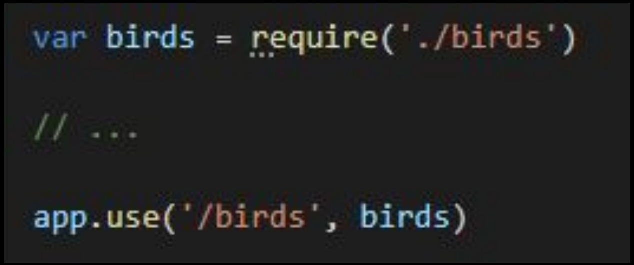
```
var express = require('express')
var router = express.Router()

// middleware that is specific to this router
router.use(function timeLog (req, res, next) {
  console.log('Time: ', Date.now())
  next()
})
```

express.Router

- Load the module in the main app.js, the app will now be able to handle /birds and /birds/about



```
var express = require('express')
var router = express.Router()

// define the home page route
router.get('/', function (req, res) {
  res.send('Birds home page')
})
// define the about route
router.get('/about', function (req, res) {
  res.send('About birds')
})

module.exports = router
```

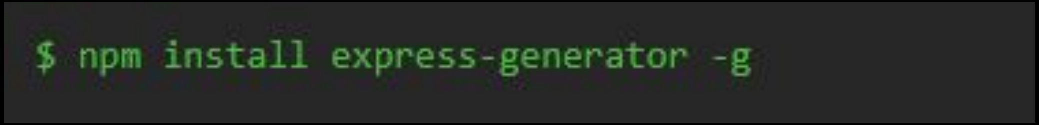


```
var birds = require('./birds')

// ...

app.use('/birds', birds)
```

Express application generator

- Use the application generator tool, express-generator, to quickly create an application skeleton.

- The express-generator package installs the express command-line tool. Use the following command to do so

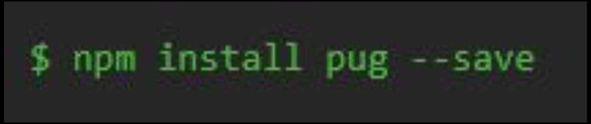


```
$ npm install express-generator -g
```

Using template engines with Express

- A template engine enables you to use static template files in your application. At runtime, the template engine replaces variables in a template file with actual values, and transforms the template into an HTML file sent to the client.

- Some popular template engines that work with Express are Pug, Mustache, and EJS. The Express application generator uses Jade as its default, but it also supports several others.



```
$ npm install pug --save
```