



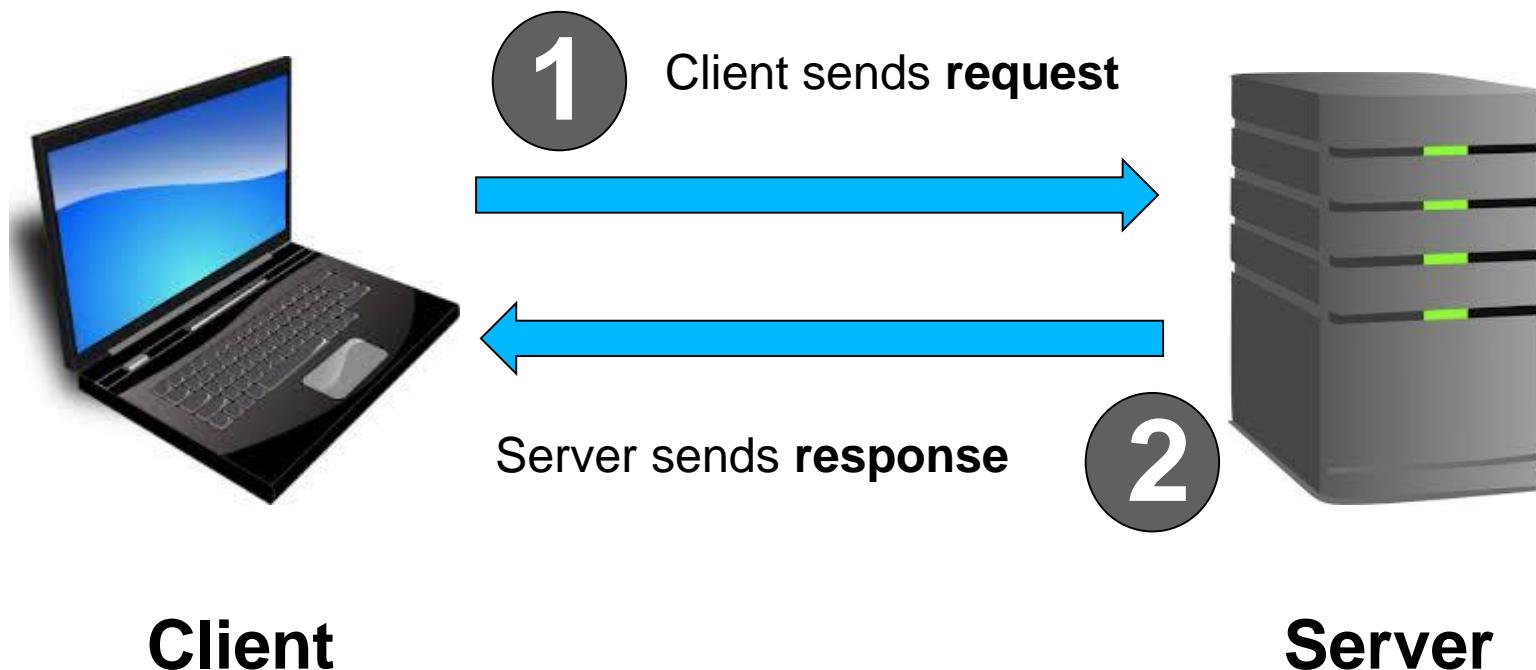
Video 4.1

Intro to Node.js Environment Setup

Chris Murphy

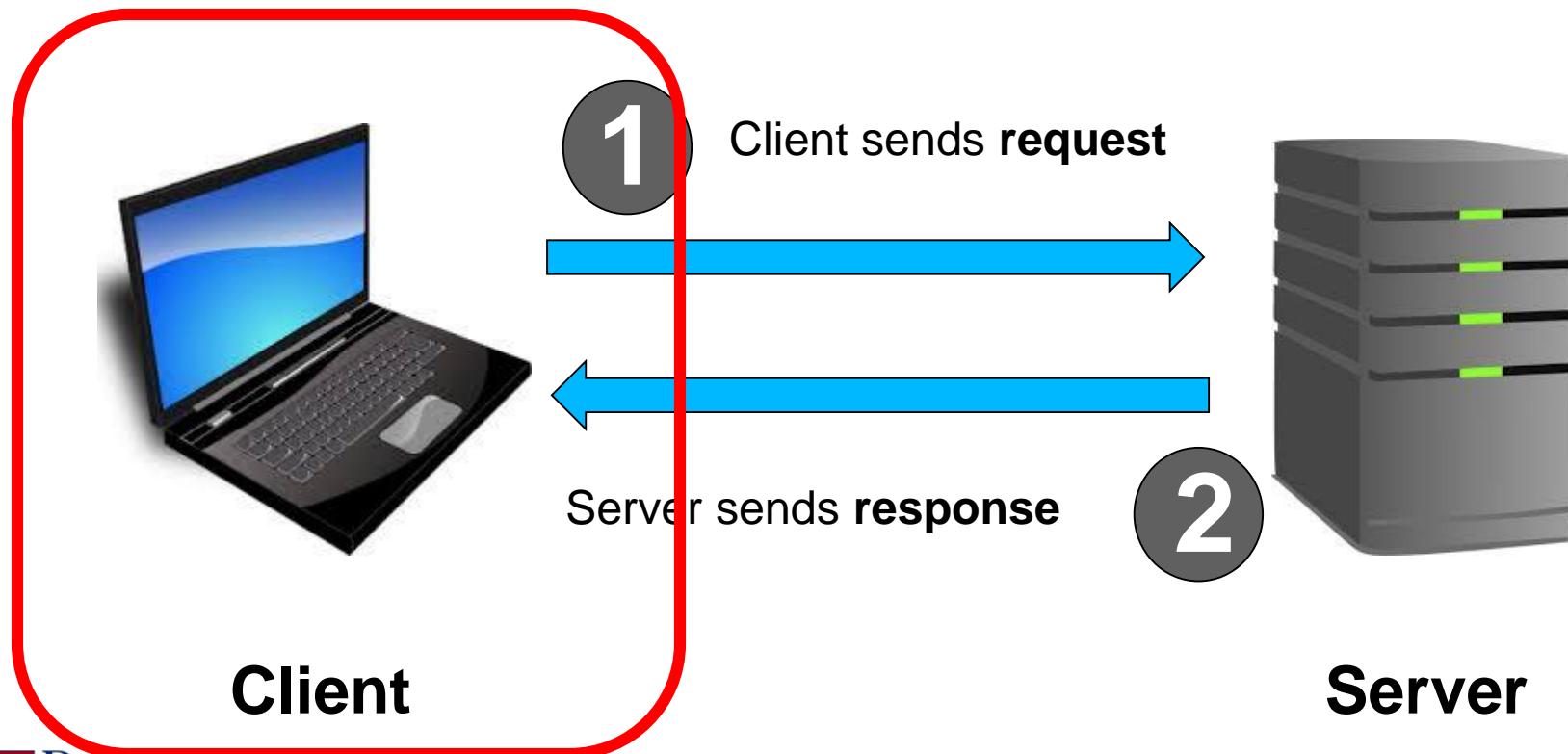
Review: How does a Web Browser Work?

- The World Wide Web utilizes **Hypertext Transfer Protocol (HTTP)** to transfer documents



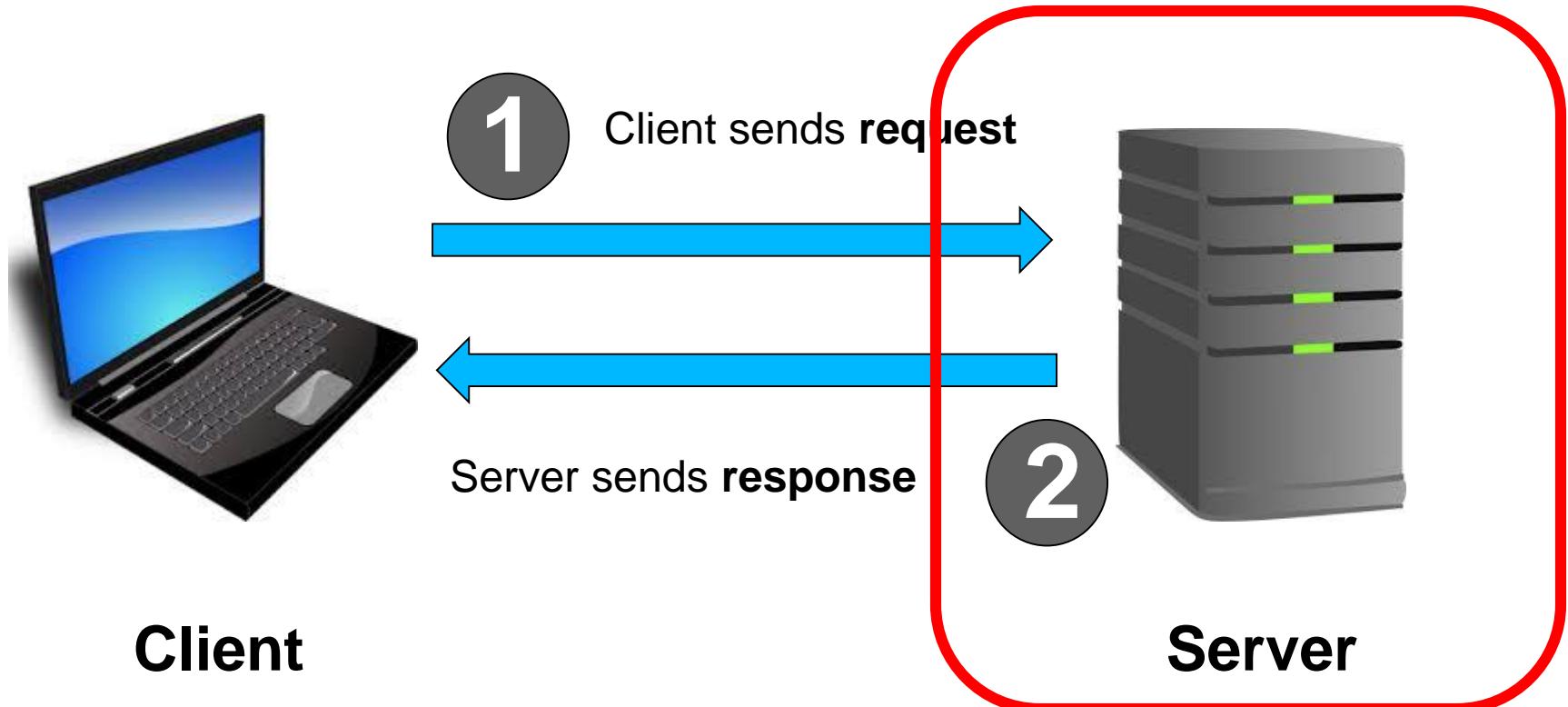
Review: How does a Web Browser Work?

- The World Wide Web utilizes **Hypertext Transfer Protocol (HTTP)** to transfer documents



Review: How does a Web Browser Work?

- The World Wide Web utilizes **Hypertext Transfer Protocol (HTTP)** to transfer documents



What does the Web Server do?

- Listen for and accept incoming HTTP requests
- Parse the HTTP request to determine what is being requested
- Locate (and/or create) the resource being requested
- Construct and send back the HTTP response

Node.js

- Asynchronous, event-driven JavaScript runtime environment for building web applications
- Treats HTTP requests as events that invoke callback functions/handlers that construct the HTTP response
- Also includes a package manager to simplify the deployment of JavaScript apps



Installing Node.js

- You can install Node.js by downloading, running, and finishing the package installer available here:
 - <https://nodejs.org/en/download/>
- Check that installation is correct using: `node -v`
- Update modules using: `npm install npm -g`

Setting up a new project

- Create a new folder for your project
- Use Terminal, Command Prompt, etc. to navigate to that folder
- Set up a new project by running: **npm init**
 - You will be prompted to enter some information about your project
 - Specify “index.js” as your entry point

Setting up a new project

- Your project folder should now have a **package.json** configuration file

```
{  
  "name": "helloworld",  
  "version": "1.0.0",  
  "description": "A basic hello world app",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "edX Learner",  
  "license": "ISC"  
}
```

Express

- Express is a web application framework that sits on top of a Node.js server
- Express helps you modularize and streamline your web application
- Within Express, you can organize your app in many ways:
 - Define separate modules that have different responsibilities
 - Handle requests via different routes and routers
 - Split each step in the processing of a request into Middlewares

Adding Express

- To use Express, run the following from the folder where you created your Node.js app:
npm install express --save
- The Express **package** will be downloaded to the project and added to your package.json file as a **dependency**
 - **Package**: a package is a module of JavaScript code, usually with a specific purpose, that can be re-used and assembled with other modules
 - **Dependency**: A dependency is a piece of code that your program relies on to work correctly

Express Configuration

- Your package.json file will now have a new section called dependencies
- npm can refer to this in the future and re-download or update your packages as needed

```
{  
  "name": "helloworld",  
  "version": "1.0.0",  
  "description": "A basic hello world app",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "edX Learner",  
  "license": "ISC",  
  "dependencies": {  
    "express": "^4.15.3"  
  }  
}
```



Express Configuration

- Your package.json file will now have a new section called dependencies
- npm can refer to this in the future and re-download or update your packages as needed

```
{  
  "name": "helloworld",  
  "version": "1.0.0",  
  "description": "A basic hello world app",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "edX Learner",  
  "license": "ISC",  
  "dependencies": {  
    "express": "^4.15.3"  
  }  
}
```



Express Configuration

- Your package.json file will now have a new section called dependencies
- npm can refer to this in the future and re-download or update your packages as needed

```
{  
  "name": "helloworld",  
  "version": "1.0.0",  
  "description": "A basic hello world app",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "edX Learner",  
  "license": "ISC",  
  "dependencies": {  
    "express": "^4.15.3"  
  }  
}
```



Express Configuration

- Your package.json file will now have a new section called dependencies
- npm can refer to this in the future and re-download or update your packages as needed

```
{  
  "name": "helloworld",  
  "version": "1.0.0",  
  "description": "A basic hello world app",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "edX Learner",  
  "license": "ISC",  
  "dependencies": {  
    "express": "^4.15.3"  
  }  
}
```



Express Configuration

- Your package.json file will now have a new section called dependencies
- npm can refer to this in the future and re-download or update your packages as needed

```
{  
  "name": "helloworld",  
  "version": "1.0.0",  
  "description": "A basic hello world app",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "edX Learner",  
  "license": "ISC",  
  "dependencies": {  
    "express": "^4.15.3"  
  }  
}
```



Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Hello World

- Create a file named **index.js** in your Node.js project root directory with the following contents:

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

app.use('/', (req, res) => {
    res.send('Hello World!');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

Running Express

- In the project folder, run: **node index.js**
- When the server starts, you should see “Listening on port 3000” written to the console/screen
- Open a browser on the same computer and go to **http://localhost:3000/**



localhost:3000

x

Guest



localhost:3000

⋮

Hello World!

Looking Ahead

- How can the server send different responses for different requests?
- How can the server dynamically generate responses?
- How does the server interact with external data sources?



Video 4.2

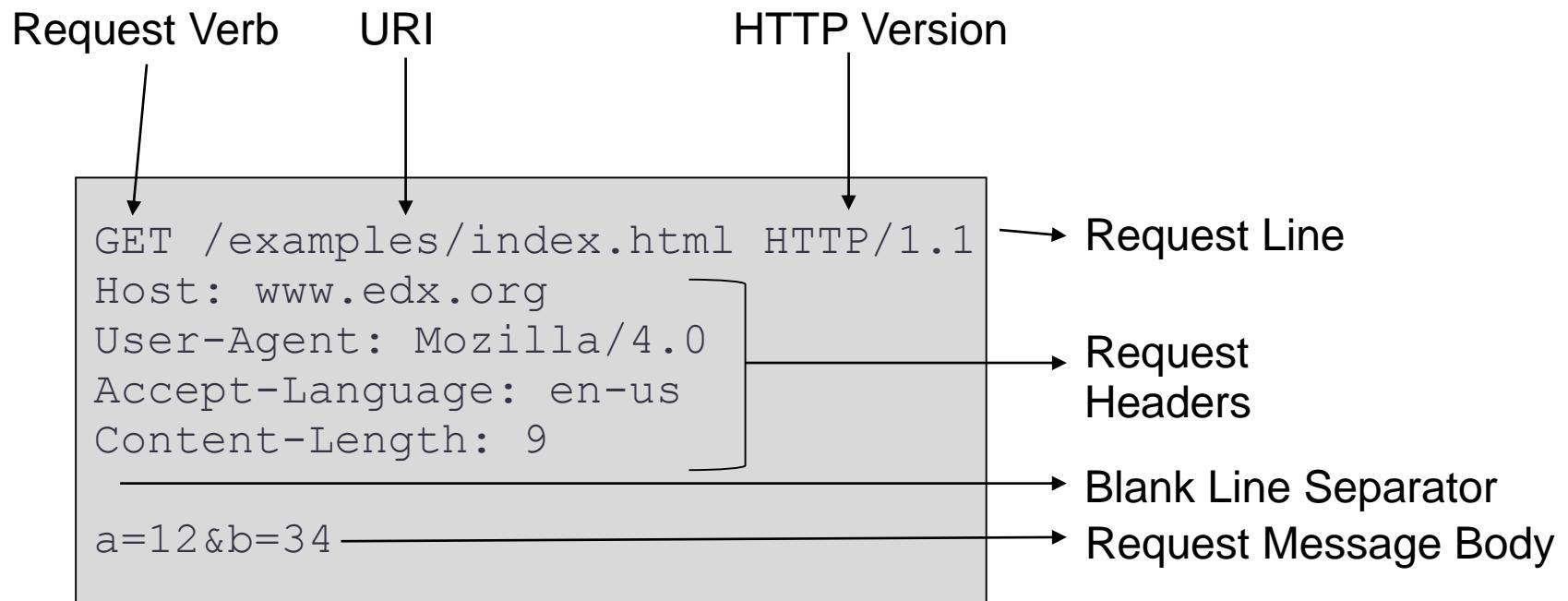
Node.js Request and Response Objects

Chris Murphy

Review

- Web browsers communicate with Web servers via HTTP requests and responses
- Node.js and Express simplify the development of Web servers to handle HTTP requests and create and return HTTP responses

Anatomy of an HTTP Request



Node.js/Express Request Objects

- An HTTP Request is represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Node.js/Express Request Objects

- An HTTP Request is represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

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

app.use ('/ ', (req, res) => {
  res.send('Hello World!');
} );

app.listen(3000, () => {
  console.log('Listening on port 3000');
} );
```

Node.js/Express Request Objects

- An HTTP Request is represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Node.js/Express Request Objects

- An HTTP Request is represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Request Object Properties/Functions

```
app.use('/', (req, res) => {  
  
    var method = req.method;  
    var url = req.url;  
    var agent = req.headers['user-agent'];  
    agent = req.get('User-Agent');  
  
    res.send('Hello World!');  
}) ;
```

Request Object Properties/Functions

```
app.use('/', (req, res) => {  
  
  var method = req.method;  
  var url = req.url;  
  var agent = req.headers['user-agent'];  
  agent = req.get('User-Agent');  
  
  res.send('Hello World!');  
}) ;
```

Request Object Properties/Functions

- **method**: the HTTP Request verb/action

```
app.use('/', (req, res) => {  
  
  var method = req.method;  
  var url = req.url;  
  var agent = req.headers['user-agent'];  
  agent = req.get('User-Agent');  
  
  res.send('Hello World!');  
}) ;
```

Request Object Properties/Functions

- **method**: the HTTP Request verb/action
- **url**: the resource that was requested

```
app.use('/', (req, res) => {  
  
    var method = req.method;  
    var url = req.url;  
    var agent = req.headers['user-agent'];  
    agent = req.get('User-Agent');  
  
    res.send('Hello World!');  
}) ;
```

Request Object Properties/Functions

- **method**: the HTTP Request verb/action
- **url**: the resource that was requested
- **headers**: object containing all headers

```
app.use('/', (req, res) => {

    var method = req.method;
    var url = req.url;
    var agent = req.headers['user-agent'];
    agent = req.get('User-Agent');

    res.send('Hello World!');
}) ;
```

Request Object Properties/Functions

- **method**: the HTTP Request verb/action
- **url**: the resource that was requested
- **headers**: object containing all headers
- **get(*field*)**: request header field

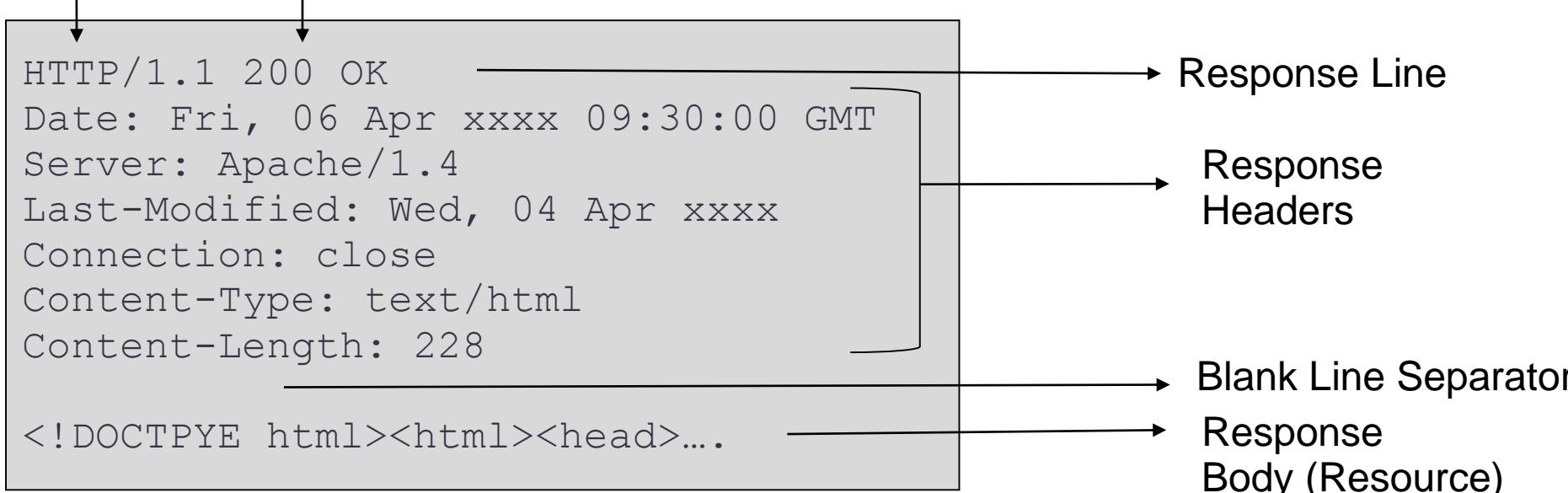
```
app.use('/', (req, res) => {

    var method = req.method;
    var url = req.url;
    var agent = req.headers['user-agent'];
agent = req.get('User-Agent');

    res.send('Hello World!');
}) ;
```

Anatomy of an HTTP Response

HTTP Version Status Code



Node.js/Express Response Objects

- An HTTP Response is also represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

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

app.use('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Node.js/Express Response Objects

- An HTTP Response is also represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

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

app.use('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(3000, () => {
  console.log('Listening on port 3000');
});
```

Response Object Functions

```
app.use('/', (req, res) => {  
  
  res.status(200);  
  res.type('html');  
  res.write('Hello world!');  
  res.write('<p>');  
  res.write('<b>Have a nice day</b>');  
  res.end();  
  
}) ;
```

Response Object Functions

```
app.use('/', (req, res) => {  
  
  res.status(200);  
  res.type('html');  
  res.write('Hello world!');  
  res.write('<p>');  
  res.write('<b>Have a nice day</b>');  
  res.end();  
  
}) ;
```

Response Object Functions

- **status**: set the HTTP status code

```
app.use('/', (req, res) => {  
  
  res.status(200);  
  res.type('html');  
  res.write('Hello world!');  
  res.write('<p>');  
  res.write('<b>Have a nice day</b>');  
  res.end();  
  
}) ;
```

Response Object Functions

- **status**: set the HTTP status code
- **type**: set the HTTP content type

```
app.use('/', (req, res) => {  
  
  res.status(200);  
  res.type('html');  
  res.write('Hello world!');  
  res.write('<p>');  
  res.write('<b>Have a nice day</b>');  
  res.end();  
  
}) ;
```

Response Object Functions

- **status**: set the HTTP status code
- **type**: set the HTTP content type
- **write**: add content to the body of the response

```
app.use('/', (req, res) => {  
  
    res.status(200);  
    res.type('html');  
    res.write('Hello world!');  
    res.write('<p>');  
    res.write('<b>Have a nice day</b>');  
    res.end();  
  
}) ;
```

Response Object Functions

- **status**: set the HTTP status code
- **type**: set the HTTP content type
- **write**: add content to the body of the response

```
app.use('/', (req, res) => {  
  
  res.status(200);  
  res.type('html');  
  res.write('Hello world!');  
  res.write('<p>')  
  res.write('<b>Have a nice day</b>');  
  res.end();  
  
}) ;
```

Response Object Functions

- **status**: set the HTTP status code
- **type**: set the HTTP content type
- **write**: add content to the body of the response

```
app.use('/', (req, res) => {  
  
  res.status(200);  
  res.type('html');  
  res.write('Hello world!');  
  res.write('<p>');  
  res.write('<b>Have a nice day</b>');  
  res.end();  
  
}) ;
```

Response Object Functions

- **status**: set the HTTP status code
- **type**: set the HTTP content type
- **write**: add content to the body of the response
- **end**: send the response and close the connection

```
app.use('/', (req, res) => {  
  
    res.status(200);  
    res.type('html');  
    res.write('Hello world!');  
    res.write('<p>');  
    res.write('<b>Have a nice day</b>');  
    res.end();  
  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
else {  
  res.write('Welcome, guest!');  
}  
  
res.end();  
  
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
  res.end();  
}) ;
```

Combining Requests and Responses

```
app.use('/', (req, res) => {  
  
  var name = req.query.name; // e.g. /?name=devesh  
  
  res.status(200).type('html');  
  
  if (name) {  
    res.write('Hi, ' + name + "it's nice to see you.");  
  }  
  else {  
    res.write('Welcome, guest!');  
  }  
  
res.end();  
  
}) ;
```

Summary

- Node.js and Express represent HTTP requests and responses using JavaScript objects
- We can use these objects' properties and functions to dynamically generate the content that is sent in response to a request



Video 4.3

Express Routing

Chris Murphy

Review

- Node.js and Express represent HTTP requests and responses using JavaScript objects
- We can use these objects' properties and functions to dynamically generate the content that is sent in response to a request

Express Routing

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

app.use('/about', (req, res) => {
    res.send('This is the about page.');
});

app.use('/login', (req, res) => {
    res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Express Routing

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

app.use('/about', (req, res) => {
    res.send('This is the about page.');
});

app.use('/login', (req, res) => {
    res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Express Routing

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

app.use('/about', (req, res) => {
  res.send('This is the about page.');
});

app.use('/login', (req, res) => {
  res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
  res.status(404).send('Not found!');
});

app.listen(3000, () => {
  console.log('Listening on port 3000');
});
```

Express Routing

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

app.use('/about', (req, res) => {
  res.send('This is the about page.');
}) ;

app.use('/login', (req, res) => {
  res.send('This is the login page.');
}) ;

app.use( /*default*/ (req, res) => {
  res.status(404).send('Not found!');
}) ;

app.listen(3000, () => {
  console.log('Listening on port 3000');
}) ;
```

Express Routing

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

app.use('/about', (req, res) => {
    res.send('This is the about page.');
});

app.use('/login', (req, res) => {
    res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Express Routing

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

app.use('/about', (req, res) => {
  res.send('This is the about page.');
});

app.use('/login', (req, res) => {
  res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
  res.status(404).send('Not found!');
});

app.listen(3000, () => {
  console.log('Listening on port 3000');
});
```

Express Routing

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

app.use('/about', (req, res) => {
    res.send('This is the about page.');
} );

app.use('/login', (req, res) => {
    res.send('This is the login page.');
} );

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

Express Routing

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

app.use('/about', (req, res) => {
    res.send('This is the about page.');
});

app.use('/login', (req, res) => {
    res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Express Routing

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

app.use('/about', (req, res) => {
    res.send('This is the about page.');
});

app.use('/login', (req, res) => {
    res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Express Routing

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

app.use('/about', (req, res) => {
    res.send('This is the about page.');
});

app.use('/login', (req, res) => {
    res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Express Routing

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

app.use('/about', (req, res) => {
    res.send('This is the about page.');
});

app.use('/login', (req, res) => {
    res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Express Routing

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

app.use('/about', (req, res) => {
    res.send('This is the about page.');
});

app.use('/login', (req, res) => {
    res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Express Routing

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

app.use('/about', (req, res) => {
    res.send('This is the about page.');
});

app.use('/login', (req, res) => {
    res.send('This is the login page.');
});

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Express Middleware

- A **middleware** is a function that is invoked in the handling of an HTTP request
- It is used in the “middle” between receiving a request and sending a response
- Multiple middlewares can be chained together on the same request

Middleware: Serving Static Files

- The simplest middleware is `express.static`, which serves static files that are locally stored

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

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

app.use( /*default*/ (req, res) => {
  res.status(404).send('Not found!');
});

app.listen(3000, () => {
  console.log('Listening on port 3000');
});
```

Middleware: Serving Static Files

- The simplest middleware is `express.static`, which serves static files that are locally stored

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

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

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

Middleware: Serving Static Files

- The simplest middleware is `express.static`, which serves static files that are locally stored

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

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

app.use( /*default*/ (req, res) => {
  res.status(404).send('Not found!');
});

app.listen(3000, () => {
  console.log('Listening on port 3000');
});
```

Middleware: Serving Static Files

- The simplest middleware is `express.static`, which serves static files that are locally stored

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

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

app.use( /*default*/ (req, res) => {
  res.status(404).send('Not found!');
} );

app.listen(3000, () => {
  console.log('Listening on port 3000');
} );
```

Middleware: Serving Static Files

- The simplest middleware is `express.static`, which serves static files that are locally stored

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

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

app.use( /*default*/ (req, res) => {
  res.status(404).send('Not found!');
} );

app.listen(3000, () => {
  console.log('Listening on port 3000');
} );
```

Middleware: Serving Static Files

- The simplest middleware is `express.static`, which serves static files that are locally stored

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

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

app.use( /*default*/ (req, res) => {
  res.status(404).send('Not found!');
});

app.listen(3000, () => {
  console.log('Listening on port 3000');
});
```

```
<!-- This is files/index.html -->

<html>
<body>
<h1>Hello!</h1>

<!-- File is files/images/kitty.jpg -->
</body>
</html>
```

```
<!-- This is files/index.html -->

<html>
<body>
<h1>Hello!</h1>

<!-- File is files/images/kitty.jpg -->
</body>
</html>
```

```
<!-- This is files/index.html -->

<html>
<body>
<h1>Hello!</h1>

<!-- File is files/images/kitty.jpg -->
</body>
</html>
```

```
<!-- This is files/index.html -->

<html>
<body>
<h1>Hello!</h1>

<!-- File is files/images/kitty.jpg -->
</body>
</html>
```

```
<!-- This is files/index.html -->

<html>
<body>
<h1>Hello!</h1>

<!-- File is files/images/kitty.jpg -->
</body>
</html>
```

```
<!-- This is files/index.html -->

<html>
<body>
<h1>Hello!</h1>

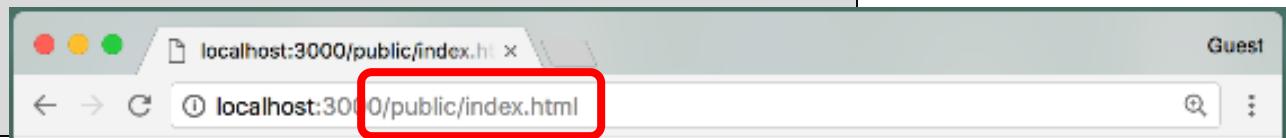
<!-- File is files/images/kitty.jpg -->
</body>
</html>
```



```
<!-- This is files/index.html -->

<html>
<body>
<h1>Hello!</h1>

<!-- File is files/images/kitty.jpg -->
</body>
</html>
```



Hello!



Middleware: Serving Static Files

- We can use the response object to send back specific HTML files as needed

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

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

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

Middleware: Serving Static Files

- We can use the response object to send back specific HTML files as needed

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

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

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
})

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

Middleware: Serving Static Files

- We can use the response object to send back specific HTML files as needed

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

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

app.use( /*default*/ (req, res) => {
    res.status(404).send('Not found!');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

Middleware: Serving Static Files

- We can use the response object to send back specific HTML files as needed

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

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Middleware: Serving Static Files

- We can use the response object to send back specific HTML files as needed

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

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```

Middleware: Serving Static Files

- We can use the response object to send back specific HTML files as needed

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

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

Middleware: Serving Static Files

- We can use the response object to send back specific HTML files as needed

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

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

Defining and Using Middleware

- Middleware functions can contain any amount of JavaScript code with any functionality
- They take three parameters: **req**, **res**, and **next**
- **next()** must be called at the end of the function to invoke the next middleware or the final response

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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

app.use(logger);

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

app.use(logger);

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

app.use(logger);

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

app.use(logger);

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

app.use(logger);

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

app.use(logger);

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

app.use(logger);

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

app.use(logger);

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

app.use(logger);

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

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

// app.use(logger);

app.use('/public', logger, express.static('files'));

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

// app.use(logger);

app.use('/public', logger, express.static('files'));

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

// app.use(logger);

app.use('/public', logger, express.static('files'));

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

// app.use(logger);

app.use('/public', logger, express.static('files'));

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



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

var logger = (req, res, next) => {
    var url = req.url;
    var time = new Date();
    console.log('Received request for ' + url +
        ' at ' + time);
    next();
};

// app.use(logger);

app.use('/public', logger, express.static('files'));

app.use( /*default*/ (req, res) => {
    res.status(404).sendFile(__dirname + '/404.html');
});

app.listen(3000, () => {
    console.log('Listening on port 3000');
});
```



Middleware Chaining

- Middleware functions are called in the order in which they are specified
- Each uses the same Request and Response objects
- A middleware function can modify the Request so that it can then be used by subsequent middleware functions “downstream” in the route

```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
          (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}
```

```
var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}
```

```
var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}
```

```
app.use('/welcome', nameFinder, greeter,
          (req, res) => { res.end(); } );
```

```
app.use('/admin', adminName, greeter,
          (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
        (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
        (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}
```

```
var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}
```

```
var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}}
```

```
app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );
```

```
app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



```
var nameFinder = (req, res, next) => {
  var name = req.query.name;
  if (name) req.username = name.toUpperCase();
  else req.username = 'Guest';
  next();
}

var greeter = (req, res, next) => {
  res.status(200).type('html');
  res.write('Hello, ' + req.username);
  next();
}

var adminName = (req, res, next) => {
  req.username = 'Admin';
  next();
}

app.use('/welcome', nameFinder, greeter,
         (req, res) => { res.end(); } );

app.use('/admin', adminName, greeter,
         (req, res) => { res.end(); } );
```



Middleware, Routes, and Routers

- We may find that the same combinations of middleware functions are being used in multiple routes
- We can combine middleware functions into “sub-routes” using **Routers** and then use those in our routes

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
       greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
       greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

app.use('/welcome', logger, nameFinder, header,
        greeter, footer, (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
       greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

app.use('/welcome', logger, nameFinder, header,
  greeter, footer, (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
  greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

app.use('/welcome', logger, nameFinder, header,
        greeter, footer, (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

app.use('/welcome', logger, nameFinder, header,
        greeter, footer, (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
       greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
       greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
       greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }
```

```
var greeter = (req, res, next) => { . . . }
```

```
var adminName = (req, res, next) => { . . . }
```

```
var logger = (req, res, next) => { . . . }
```

```
var header = (req, res, next) => { . . . }
```

```
var footer = (req, res, next) => { . . . }
```

```
app.use('/welcome', logger, nameFinder, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
app.use('/admin', logger, adminName, header,  
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, header,
        greeter, footer, (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, header,
        greeter, footer, (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, header,
        greeter, footer, (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, header,
        greeter, footer, (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, header,
        greeter, footer, (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, header,
        greeter, footer, (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, commonRoute,
        (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, commonRoute,
        (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, commonRoute,
        (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
        greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, commonRoute,
        (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, header,
       greeter, footer, (req, res) => { res.end(); } );
```

```
var nameFinder = (req, res, next) => { . . . }

var greeter = (req, res, next) => { . . . }

var adminName = (req, res, next) => { . . . }

var logger = (req, res, next) => { . . . }

var header = (req, res, next) => { . . . }

var footer = (req, res, next) => { . . . }

var commonRoute = express.Router();
commonRoute.use(header, greeter, footer);

app.use('/welcome', logger, nameFinder, commonRoute,
        (req, res) => { res.end(); } );

app.use('/admin', logger, adminName, commonRoute,
        (req, res) => { res.end(); } );
```

Summary

- **Routing** allows us to specify different functionality for different HTTP requests
- Routing uses **middleware** functions, each of which handles a different part of the functionality
- Middleware functions can be chained together and can pass values to each other by modifying the Request object
- **Routers** allow us to combine middleware functions into common “sub-routes”



Video 4.4

Getting User Data

Chris Murphy

Review

- Node.js and Express allow us to build server-side web apps in JavaScript
- HTTP Requests and Responses are represented as JavaScript objects
- We can display content based on different routes through the app

Getting Data from Users: HTTP Requests

- **Query parameters**
 - Key/value pairs that are part of the URL
 - Can be part of a static URL or be generated by an HTML form using the “GET” method
- **POST data**
 - Key/value pairs that are included in the body of the HTTP request
 - Result from an HTML form using the “POST” method

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {  
  
  var query = req.query;  
  console.log(query);  
  
  var name = query.name;          // 'Lydia'  
  var location = query.location; // 'United States'  
  
  var length = Object.keys(query).length; // 2  
  
  res.send('Hello World!');  
});
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {  
  
  var query = req.query;  
  console.log(query);  
  
  var name = query.name;          // 'Lydia'  
  var location = query.location; // 'United States'  
  
  var length = Object.keys(query).length; // 2  
  
  res.send('Hello World!');  
});
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {  
  
  var query = req.query;  
  console.log(query);  
  
  var name = query.name;          // 'Lydia'  
  var location = query.location; // 'United States'  
  
  var length = Object.keys(query).length; // 2  
  
  res.send('Hello World!');  
});
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {

  var query = req.query;
  console.log(query);

  var name = query.name;          // 'Lydia'
  var location = query.location; // 'United States'

  var length = Object.keys(query).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {  
  
  var query = req.query;  
  console.log(query);  
  
  var name = query.name;          // 'Lydia'  
  var location = query.location; // 'United States'  
  
  var length = Object.keys(query).length; // 2  
  
  res.send('Hello World!');  
});
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {  
  
  var query = req.query;  
  console.log(query);  
  
  var name = query.name;          // 'Lydia'  
  var location = query.location; // 'United States'  
  
  var length = Object.keys(query).length; // 2  
  
  res.send('Hello World!');  
});
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {

  var query = req.query;
  console.log(query);

  var name = query.name;           // 'Lydia'
  var location = query.location; // 'United States'

  var length = Object.keys(query).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {

  var query = req.query;
  console.log(query);

  var name = query.name;          // 'Lydia'
  var location = query.location; // 'United States'

  var length = Object.keys(query).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {

  var query = req.query;
  console.log(query);

  var name = query.name;           // 'Lydia'
  var location = query.location; // 'United States'

  var length = Object.keys(query).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {

  var query = req.query;
  console.log(query);

  var name = query.name;          // 'Lydia'
var location = query.location; // 'United States'

  var length = Object.keys(query).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {

  var query = req.query;
  console.log(query);

  var name = query.name;          // 'Lydia'
  var location = query.location; // 'United States'

  var length = Object.keys(query).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {

  var query = req.query;
  console.log(query);

  var name = query.name;           // 'Lydia'
  var location = query.location; // 'United States'

  var length = Object.keys(query).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {

  var query = req.query;
  console.log(query);

  var name = query.name;          // 'Lydia'
  var location = query.location; // 'United States'

  var length = Object.keys(query).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Query Properties

- An HTTP Request object can include **query** properties that come from the URL

http://localhost:3000/?name=Lydia&location=United+States

```
app.use('/', (req, res) => {

  var query = req.query;
  console.log(query);

  var name = query.name;          // 'Lydia'
  var location = query.location; // 'United States'

  var length = Object.keys(query).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/:location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.location; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;           // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;           // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;           // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

  var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

Request Object Parameters

- An HTTP Request object can include **param** properties that come from a parameterized URL

http://localhost:3000/name/Lydia/location/United States

```
app.use('/name/:userName/location/:userLocation',
  (req, res) => {

  var params = req.params;
  console.log(params);

  var name = params.userName;          // 'Lydia'
  var location = params.userLocation; // 'United States'

var length = Object.keys(params).length; // 2

  res.send('Hello World!');
}) ;
```

HTML Forms

- Forms allow users to enter or select data, e.g. via input boxes, checkboxes, radio buttons, etc.
- The form specifies the **action** and **method** that result when the user chooses to **submit** the form
 - Action: the URL to be requested
 - Method: the HTTP Request “verb,” e.g. GET or POST



← → ⌂



Name:

I like:

- Dogs
- Cats
- Birds

Submit form!

Name:

I like:

- Dogs
- Cats
- Birds

Submit form!

```
<html>
<body>

<form action="/handleForm" method="post">

Name: <input name="username">
<p>
I like:<br>
<input type=checkbox name="animal" value="dogs">Dogs <br>
<input type=checkbox name="animal" value="cats">Cats <br>
<input type=checkbox name="animal" value="birds">Birds <br>
<p>
<input type=submit value="Submit form!">

</form>

</body>
</html>
```

```
<html>
<body>

<b><form action="/handleForm" method="post"></b>

Name: <input name="username">
<p>
I like:<br>
<input type=checkbox name="animal" value="dogs">Dogs <br>
<input type=checkbox name="animal" value="cats">Cats <br>
<input type=checkbox name="animal" value="birds">Birds <br>
<p>
<input type=submit value="Submit form!">

<b></form></b>

</body>
</html>
```

```
<html>
<body>

<form action="/handleForm" method="post">

Name: <input name="username">
<p>
I like:<br>
<input type=checkbox name="animal" value="dogs">Dogs <br>
<input type=checkbox name="animal" value="cats">Cats <br>
<input type=checkbox name="animal" value="birds">Birds <br>
<p>
<input type=submit value="Submit form!">

</form>

</body>
</html>
```

```
<html>
<body>

<form action="/handleForm" method="post">

Name: <input name="username">
<p>
I like:<br>
<input type=checkbox name="animal" value="dogs">Dogs <br>
<input type=checkbox name="animal" value="cats">Cats <br>
<input type=checkbox name="animal" value="birds">Birds <br>
<p>
<input type=submit value="Submit form!">

</form>

</body>
</html>
```

```
<html>
<body>

<form action="/handleForm" method="post">

Name: <input name="username">
<p>
I like:<br>
<input type=checkbox name="animal" value="dogs">Dogs <br>
<input type=checkbox name="animal" value="cats">Cats <br>
<input type=checkbox name="animal" value="birds">Birds <br>
<p>
<input type=submit value="Submit form!">

</form>

</body>
</html>
```

```
<html>
<body>

<form action="/handleForm" method="post">

Name: <input name="username">
<p>
I like:<br>
<input type=checkbox name="animal" value="dogs">Dogs <br>
<input type=checkbox name="animal" value="cats">Cats <br>
<input type=checkbox name="animal" value="birds">Birds <br>
<p>
<input type=submit value="Submit form!">

</form>

</body>
</html>
```

```
<html>
<body>

<form action="/handleForm" method="post">

Name: <input name="username">
<p>
I like:<br>
<input type=checkbox name="animal" value="dogs">Dogs <br>
<input type=checkbox name="animal" value="cats">Cats <br>
<input type=checkbox name="animal" value="birds">Birds <br>
<p>
<b><input type=submit value="Submit form!"></b>

</form>

</body>
</html>
```

```
<html>
<body>

<form action="/handleForm" method="post"
```

Reading POST Data in Express

- When a form's method is “GET”, the data is sent in the URL query parameters
- When a form's method is “POST”, the data is sent in the **body** of the HTTP request
- To read the body of the HTTP request in Express, use the **body-parser** middleware
- To install it, run: **npm install body-parser**

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
} );

app.listen(3000, () => {
    console.log('Listening on port 3000');
} );
```

```
<html>
<body>

<form action="/handleForm" method="post">

Name: <input name="username">
<p>
I like:<br>
<input type=checkbox name="animal" value="dogs">Dogs <br>
<input type=checkbox name="animal" value="cats">Cats <br>
<input type=checkbox name="animal" value="birds">Birds <br>
<p>
<input type=submit value="Submit form!">

</form>

</body>
</html>
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = req.body.animal; // this is an array
  . . .
  res.send('Thank you!');
}) ;

app.listen(3000, () => {
  console.log('Listening on port 3000');
}) ;
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

```
<html>
<body>

<form action="/handleForm" method="post">

Name: <input name="username">
<p>
I like:<br>
<input type=checkbox name="animal" value="dogs">Dogs <br>
<input type=checkbox name="animal" value="cats">Cats <br>
<input type=checkbox name="animal" value="birds">Birds <br>
<p>
<input type=submit value="Submit form!">

</form>

</body>
</html>
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

```
<html>
<body>

<form action="/handleForm" method="post">

Name: <input name="username">
<p>
I like:<br>
<input type=checkbox name="animal" value="dogs">Dogs <br>
<input type=checkbox name="animal" value="cats">Cats <br>
<input type=checkbox name="animal" value="birds">Birds <br>
<p>
<input type=submit value="Submit form!">

</form>

</body>
</html>
```

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

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

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = req.body.animal; // this is an array
    . . .
    res.send('Thank you!');
}) ;

app.listen(3000, () => {
    console.log('Listening on port 3000');
}) ;
```

Summary

- HTTP Request **query** properties: key/value pairs that come from URL
- HTTP Request **param** properties: key/value pairs that come from parameterized URL
- HTTP Request **body** properties: input data from form submitting using POST method



Video 4.5

EJS

Chris Murphy

Review

- Node.js and Express allow us to build server-side web apps in JavaScript
- HTTP Requests and Responses are represented as JavaScript objects
- We can get data from user either via the URL or through form submissions



Name:

I like:

- Dogs
- Cats
- Birds

Submit form!



Name: Sahana

I like:

Dogs

Cats

Birds

Submit form!



Name: Sahana

I like:

Dogs

Cats

Birds

Submit form!



Hello, Sahana, nice to meet you.

Here are the animals you like:

- dogs
- birds

[Back to form](#)

Hello, Sahana, nice to meet you.

Here are the animals you like:

- dogs
- birds

[Back to form](#)

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html') .status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
    var name = req.body.username;
    var animals = [] .concat(req.body.animal);
    res.type('html') .status(200);
    res.write('Hello, ' + name + ', nice to meet you.');
    res.write('<p>Here are the animals you like:</p>');
    res.write('<ul>');
    animals.forEach( (animal) => {
        res.write('<li>' + animal + '</li>');
    });
    res.write('</ul>');
    res.write("<a href='/public/form.html'>" +
              "Back to form</a>");
    res.end();
}) ;
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html') .status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html') .status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html') .status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal) ;
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  }) ;
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
}) ;
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
}) ;
```



Hello, Sahana, nice to meet you.

Here are the animals you like:

- dogs
- birds

[Back to form](#)

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
}) ;
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html') .status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
}) ;
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html') .status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  ) ;
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
}) ;
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html') .status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
"Back to form</a>");
  res.end();
}) ;
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html') .status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
res.end();
}) ;
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html') .status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

What is EJS?

- EJS, or **EmbeddedJS**, is a **view engine** that uses data and embedded JavaScript to produce HTML
- This allows webpages to be developed statically and rendered dynamically server-side
- EmbeddedJS is a package that can be installed with the command: **npm install ejs**

Using EJS in an Express app

- Set EJS as the default rendering method in your app with **app.set('view engine', 'ejs');**

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

app.set('view engine', 'ejs');

app.get('/', (req, res) => {
    res.render('welcome', {username: 'CandyLover'});
}) ;
```

Using EJS in an Express app

- Set EJS as the default rendering method in your app with **app.set('view engine', 'ejs');**

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

app.set('view engine', 'ejs');

app.get('/', (req, res) => {
  res.render('welcome', {username: 'CandyLover'});
}) ;
```

Using EJS in an Express app

- Set EJS as the default rendering method in your app with `app.set('view engine', 'ejs');`
- Then generate and send the HTML from an .ejs file using the Response's `render` function

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

app.set('view engine', 'ejs');

app.get('/', (req, res) => {
  res.render('welcome', {username: 'CandyLover'});
}) ;
```

Using EJS in an Express app

- Set EJS as the default rendering method in your app with `app.set('view engine', 'ejs');`
- Then generate and send the HTML from an .ejs file using the Response's `render` function
- Arguments to the .ejs file are passed as objects

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

app.set('view engine', 'ejs');

app.get('/', (req, res) => {
    res.render('welcome', {username: 'CandyLover'});
});
```

Writing EJS files

- A .ejs file is just an HTML file that has JavaScript code embedded in it
- Anything between `<%=` and `%>` tags will be evaluated and incorporated into the HTML
- By default, the .ejs files should be in the `views/` subdirectory of the Express project

```
<!-- This is views/welcome.ejs -->

<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>

</body>
</html>
```

Writing EJS files

- A .ejs file is just an HTML file that has JavaScript code embedded in it
- Anything between `<%=` and `%>` tags will be evaluated and incorporated into the HTML
- By default, the .ejs files should be in the `views/` subdirectory of the Express project

```
<!-- This is views/welcome.ejs -->
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>

</body>
</html>
```

Writing EJS files

- A .ejs file is just an HTML file that has JavaScript code embedded in it
- Anything between `<%=` and `%>` tags will be evaluated and incorporated into the HTML
- By default, the .ejs files should be in the `views/` subdirectory of the Express project

```
<!-- This is views/welcome.ejs -->

<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>

</body>
</html>
```

Writing EJS files

- A .ejs file is just an HTML file that has JavaScript code embedded in it
- Anything between `<%=` and `%>` tags will be evaluated and incorporated into the HTML
- By default, the .ejs files should be in the **views/** subdirectory of the Express project

```
<!-- This is views/welcome.ejs -->

<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
  <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
  <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
  <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
  <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
  <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
    <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
    <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
  <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
  <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
    <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

EJS and JavaScript

- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server

```
res.render('welcome', {username: 'CandyLover', isAdmin: true});
```

```
<!DOCTYPE html>
<html>
<body>

<h1>Welcome, <%= username %>!</h1>
<% if (isAdmin) { %>
  <p> Remember to check your email every 24 hours! </p>
<% } %>

</body>
</html>
```

Hello, Sahana, nice to meet you.

Here are the animals you like:

- dogs
- birds

[Back to form](#)

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html') .status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
});
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.type('html').status(200);
  res.write('Hello, ' + name + ', nice to meet you.');
  res.write('<p>Here are the animals you like:</p>');
  res.write('<ul>');
  animals.forEach( (animal) => {
    res.write('<li>' + animal + '</li>');
  });
  res.write('</ul>');
  res.write("<a href='/public/form.html'>" +
            "Back to form</a>");
  res.end();
}) ;
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
    animals: animals }) ;
}) ;
```



```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals });
}) ;
```



```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
    animals: animals }) ;
}) ;
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals }) ;
}) ;
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals });
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% }); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals });
}) ;
```

<!-- This is views/showAnimals.ejs -->

```
Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% }); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals }) ;
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% }); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals }) ;
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% }); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name,
    animals: animals }) ;
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% }); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals }) ;
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
  <% animals.forEach( (animal) => { %>
    <li> <%= animal %> </li>
  <% } ); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals });
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( animal ) => { %>
  <li> <%= animal %> </li>
<% } ); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals }) ;
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% }); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
    animals: animals }) );
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% } ); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [] .concat(req.body.animal);
  res.render('showAnimals', { name: name,
    animals: animals }) );
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% } ); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals }) ;
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% } ); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals }) ;
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% } ); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals }) ;
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% }); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

```
app.use('/handleForm', (req, res) => {
  var name = req.body.username;
  var animals = [].concat(req.body.animal);
  res.render('showAnimals', { name: name,
                             animals: animals }) ;
}) ;
```

```
<!-- This is views/showAnimals.ejs -->

Hello, <%= name %>, nice to meet you.
<p>Here are the animals you like:
<ul>
<% animals.forEach( (animal) => { %>
  <li> <%= animal %> </li>
<% }); %>
</ul>
<a href='/public/form.html'>Back to form</a>
```

Summary

- **EJS** allows webpages to be developed statically and rendered dynamically server-side
- An Express app can generate and send the HTML from a .ejs file using the Response's **render** function
- EJS will execute any JavaScript that appears between `<%` and `%>` tags when generating the HTML page on the server



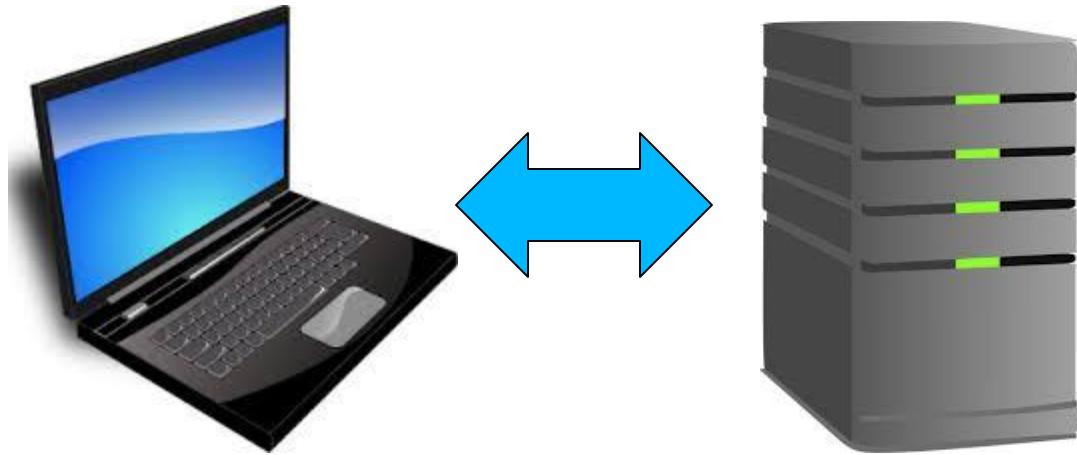
Video 4.6

MongoDB: Setup and Insert

Chris Murphy

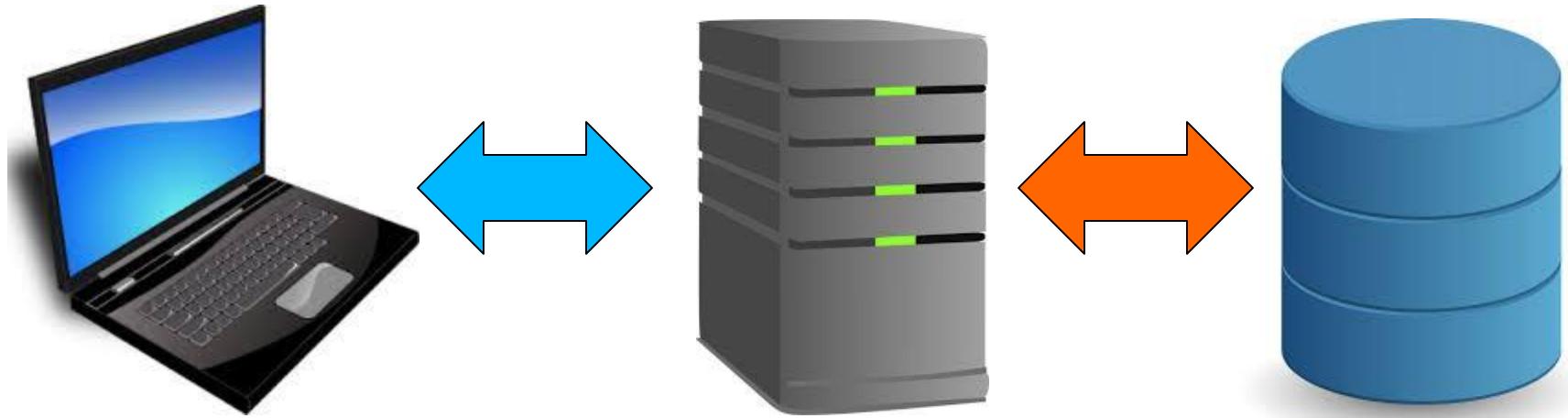
Review

- Node.js and Express allow us to build server-side web apps in JavaScript
- We can get data from the user via the URL query parameters or from form data



Client

Server



Client

Server

Database

What is a NoSQL Database?

- A **NoSQL Database** is a database that does not use SQL, the traditional method of storing data
- In SQL, data is stored in tables and rows. This is also known as a relational database
- NoSQL Databases attempt to address some of the shortcomings of SQL and other relational databases by organizing and storing data differently

What is MongoDB?

- **MongoDB** is one NoSQL Database that is designed for use with JavaScript apps
- MongoDB stores **collections of documents** rather than tables of rows



SQL Database

- A SQL table of Users might look like this:

Name	Age	Country	Occupation
Jane Doe	30	United States	Programmer
John Smith	25	Canada	Doctor
Kim Jones	27	France	Painter

MongoDB Documents

- A MongoDB **collection** of User documents might look like this:

```
{  
  name: 'Jane Doe',  
  age: 30,  
  country: 'United States',  
  occupation: 'Programmer'  
}
```

```
{  
  name: 'John Smith',  
  age: 25,  
  country: 'Canada',  
  occupation: 'Doctor'  
}
```

```
{  
  name: 'Kim Jones',  
  age: 27,  
  country: 'France',  
  occupation: 'Painter'  
}
```

Using MongoDB with a Node.js app

1. Install MongoDB locally or create an account on a remote service
2. Install packages locally to allow your JavaScript programs to access your MongoDB instance
3. Write JavaScript to describe the Schema (blueprint for Documents) that you will use in the Collection
4. Use the Schema to access MongoDB in your app

Installing MongoDB

- You can find download/installation instructions for MongoDB at <https://mongodb.com/download>
- Follow these instructions to create a new empty database and run the MongoDB server
- When you start it, it will tell you which port it is using

week4 — mongod --dbpath data/db/ — 105x38

```
[Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongod --dbpath data/db/
2017-07-11T17:53:24.349-0400 I CONTROL [initandlisten] MongoDB starting : pid=85366 port=27017 dbpath=da
ta/db/ 64-bit host=huntsman-ve565-2676.apn.wlan.upenn.edu
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] db version v3.4.6
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] git version: c55eb86ef46ee7aede3b1e2a5d184a7df4bf
b5b5
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] allocator: system
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] modules: none
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] build environment:
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten]     distarch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten]     target_arch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] options: { storage: { dbPath: "data/db/" } }
2017-07-11T17:53:24.350-0400 I - [initandlisten] Detected data files in data/db/ created by the 'w
iredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2017-07-11T17:53:24.351-0400 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=7680M,s
ession_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=
true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=
60,log_size=2GB),statistics_log=(wait=0),
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the
database.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** Read and write access to data and con
figuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.675-0400 I FTDC [initandlisten] Initializing full-time diagnostic data capture wi
th directory 'data/db/diagnostic.data'
2017-07-11T17:53:25.676-0400 I NETWORK [thread1] waiting for connections on port 27017
```

```
[Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongod --dbpath data/db/
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] MongoDB starting : pid=85366 port=27017 dbpath=da
ta/db/ 64-bit host=huntsman-ve565-2676.apn.wlan.upenn.edu
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] db version v3.4.6
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] git version: c55eb86ef46ee7aede3b1e2a5d184a7df4bf
b5b5
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] allocator: system
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] modules: none
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] build environment:
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten]     distarch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten]     target_arch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] options: { storage: { dbPath: "data/db/" } }
2017-07-11T17:53:24.350-0400 I -          [initandlisten] Detected data files in data/db/ created by the 'w
iredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2017-07-11T17:53:24.351-0400 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=7680M,s
ession_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=
true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=
60,log_size=2GB),statistics_log=(wait=0),
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the
database.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] **           Read and write access to data and con
figuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.675-0400 I FTDC [initandlisten] Initializing full-time diagnostic data capture wi
th directory 'data/db/diagnostic.data'
2017-07-11T17:53:25.676-0400 I NETWORK [thread1] waiting for connections on port 27017
```

week4 — mongod --dbpath data/db/ — 105x38

```
[Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongod --dbpath data/db/
2017-07-11T17:53:24.349-0400 I CONTROL [initandlisten] MongoDB starting : pid=85366 port=27017 dbpath=da
ta/db/ 64-bit host=huntsman-ve565-2676.apn.wlan.upenn.edu
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] db version v3.4.6
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] git version: c55eb86ef46ee7aede3b1e2a5d184a7df4bf
b5b5
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] allocator: system
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] modules: none
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] build environment:
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten]     distarch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten]     target_arch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] options: { storage: { dbPath: "data/db/" } }
2017-07-11T17:53:24.350-0400 I -          [initandlisten] Detected data files in data/db/ created by the 'w
iredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2017-07-11T17:53:24.351-0400 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=7680M,s
ession_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=
true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=
60,log_size=2GB),statistics_log=(wait=0),
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the
database.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] **           Read and write access to data and con
figuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.675-0400 I FTDC   [initandlisten] Initializing full-time diagnostic data capture wi
th directory 'data/db/diagnostic.data'
2017-07-11T17:53:25.676-0400 I NETWORK [thread1] waiting for connections on port 27017
```

Installing MongoDB

- You can find download/installation instructions for MongoDB at <https://mongodb.com/download>
- Follow these instructions to create a new empty database and run the MongoDB server
- When you start it, it will tell you which port it is using

Installing MongoDB

- You can find download/installation instructions for MongoDB at <https://mongodb.com/download>
- Follow these instructions to create a new empty database and run the MongoDB server
- When you start it, it will tell you which port it is using
- Alternatively, you may use an online service, e.g. MongoDB Atlas

Installing Drivers for MongoDB

- You can access MongoDB directly from your Node app using the **MongoClient**
- Alternatively, you can install helper packages such as **Mongoose** to simplify some tasks:

```
npm install mongoose --save
```



localhost:3000/public/personform.html

Guest

← → ⌂

localhost:3000/public/personform.html



Name:

Age:

Submit form!

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

```
<html>
<body>

<b><form action='/create' method='post'></b>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

<b></form></b>

</body>
</html>
```

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<b><input type=submit value='Submit form!'></b>

</form>

</body>
</html>
```

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

```
<html>
<body>

<form action='/create' method='post'
```



localhost:3000/public/personform.html

Guest

← → ⌂

localhost:3000/public/personform.html



Name:

Age:

Submit form!



Name: Alexis

Age: 17

Submit form!



← → ⚡



Successfully created new person:

Name: Alexis

Age: 17

[Create New Person](#)

[Show All](#)

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```



← → ⌂

ⓘ localhost:3000/create



Successfully created new person:

Name: Alexis

Age: 17

[Create New Person](#)

[Show All](#)

```
/* This is index.js */

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

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

app.use('/', (req, res) =>
{ res.redirect('/public/personform.html'); } );

app.listen(3000, () =>
{ console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

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

app.use('/', (req, res) =>
{ res.redirect('/public/personform.html'); } );

app.listen(3000, () =>
{ console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

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

app.use('/', (req, res) =>
{ res.redirect('/public/personform.html'); } );

app.listen(3000, () =>
{ console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

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

app.use('/', (req, res) =>
  { res.redirect('/public/personform.html'); } );

app.listen(3000, () =>
  { console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

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

app.use('/', (req, res) =>
{ res.redirect('/public/personform.html'); } );

app.listen(3000, () =>
{ console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

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

app.use('/', (req, res) =>
  { res.redirect('/public/personform.html'); } );

app.listen(3000, () =>
  { console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

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

app.use('/', (req, res) =>
  { res.redirect('/public/personform.html'); } );

app.listen(3000, () =>
{ console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

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

app.use('/', (req, res) =>
{ res.redirect('/public/personform.html'); } );

app.listen(3000, () =>
{ console.log('Listening on port 3000'); } );
```



← → ⚡



Successfully created new person:

Name: Alexis

Age: 17

[Create New Person](#)

[Show All](#)



Successfully created new person:

Name: Alexis

Age: 17

[Create New Person](#)

[Show All](#)

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  } );  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  } );  
} );
```

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  } );  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  } );  
} );
```

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  } );  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  } );  
});
```

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  } );  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  } );  
} );
```

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  }) ;  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  }) ;  
}) ;
```

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  } );  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  } );  
} );
```

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  } );  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  } );  
} );
```

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

```
app.use('/create', (req, res) => {  
  
    var newPerson = new Person ({ // defined in Person.js  
        name: req.body.name,  
        age: req.body.age,  
    } );  
  
    newPerson.save( (err) => {  
        if (err) {  
            res.type('html').status(500);  
            res.send('Error: ' + err);  
        }  
        else {  
            res.render('created', { person: newPerson });  
        }  
    } );  
});
```

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  } );  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  } );  
});
```

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  } );  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  } );  
} );
```

```
app.use('/create', (req, res) => {  
  
    var newPerson = new Person ({ // defined in Person.js  
        name: req.body.name,  
        age: req.body.age,  
    } );  
  
    newPerson.save( (err) => {  
        if (err) {  
            res.type('html').status(500);  
            res.send('Error: ' + err);  
        }  
        else {  
            res.render('created', { person: newPerson });  
        }  
    } );  
});
```

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  } );  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  } );  
} );
```

```
app.use('/create', (req, res) => {  
  
    var newPerson = new Person ({ // defined in Person.js  
        name: req.body.name,  
        age: req.body.age,  
    } );  
  
    newPerson.save( (err) => {  
        if (err) {  
            res.type('html').status(500);  
            res.send('Error: ' + err);  
        }  
        else {  
            res.render('created', { person: newPerson });  
        }  
    } );  
});
```

```
app.use('/create', (req, res) => {  
  
    var newPerson = new Person ({ // defined in Person.js  
        name: req.body.name,  
        age: req.body.age,  
    } );  
  
    newPerson.save( (err) => {  
        if (err) {  
            res.type('html').status(500);  
            res.send('Error: ' + err);  
        }  
        else {  
            res.render('created', { person: newPerson });  
        }  
    } );  
} );
```

```
app.use('/create', (req, res) => {  
  
    var newPerson = new Person ({ // defined in Person.js  
        name: req.body.name,  
        age: req.body.age,  
    } );  
  
    newPerson.save( (err) => {  
        if (err) {  
            res.type('html').status(500);  
            res.send('Error: ' + err);  
        }  
        else {  
            res.render('created', { person: newPerson });  
        }  
    } );  
});
```

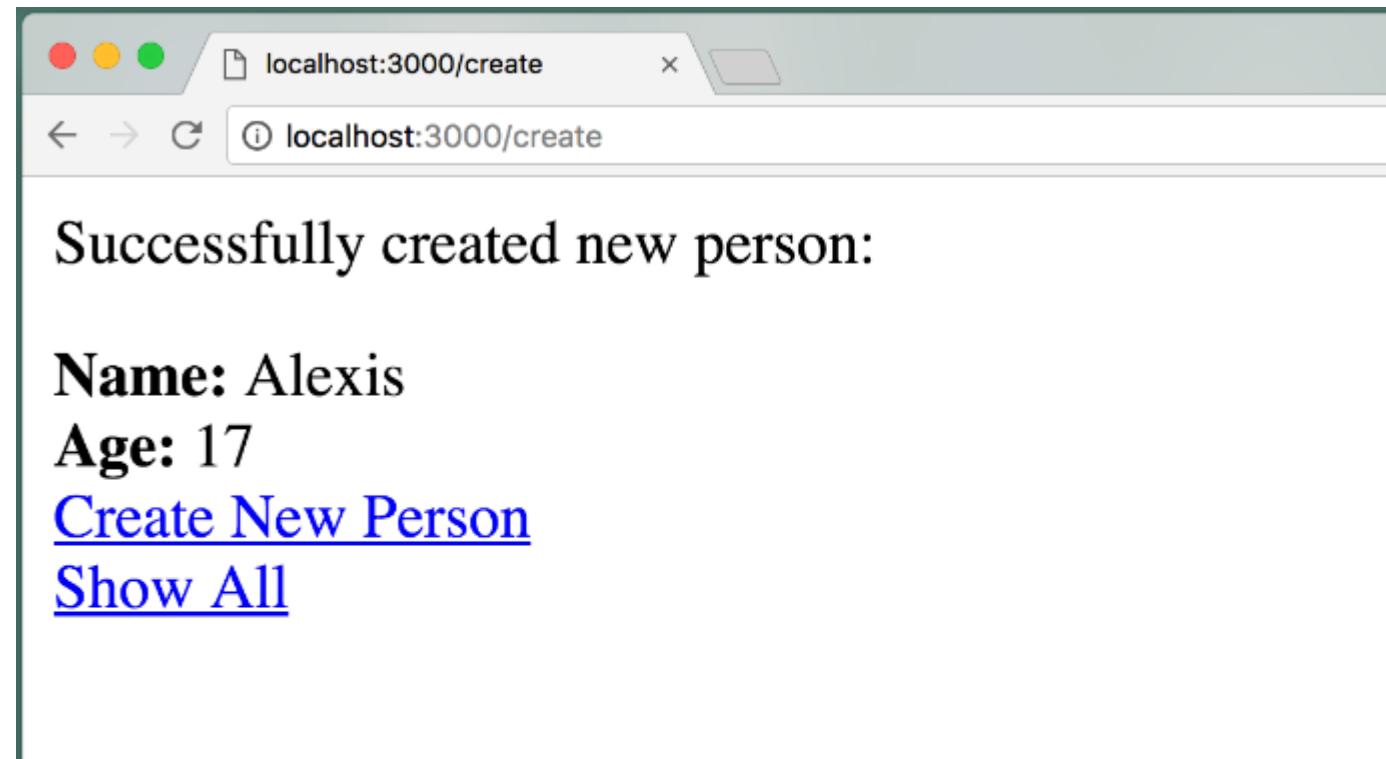
```
app.use('/create', (req, res) => {  
  
    var newPerson = new Person ({ // defined in Person.js  
        name: req.body.name,  
        age: req.body.age,  
    } );  
  
    newPerson.save( (err) => {  
        if (err) {  
            res.type('html').status(500);  
            res.send('Error: ' + err);  
        }  
        else {  
            res.render('created', { person: newPerson });  
        }  
    } );  
});
```

```
app.use('/create', (req, res) => {  
  
  var newPerson = new Person ({ // defined in Person.js  
    name: req.body.name,  
    age: req.body.age,  
  } );  
  
  newPerson.save( (err) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('created', { person: newPerson });  
    }  
  } );  
} );
```

```
<!-- This is views/created.ejs -->
```

```
Successfully created new person:
```

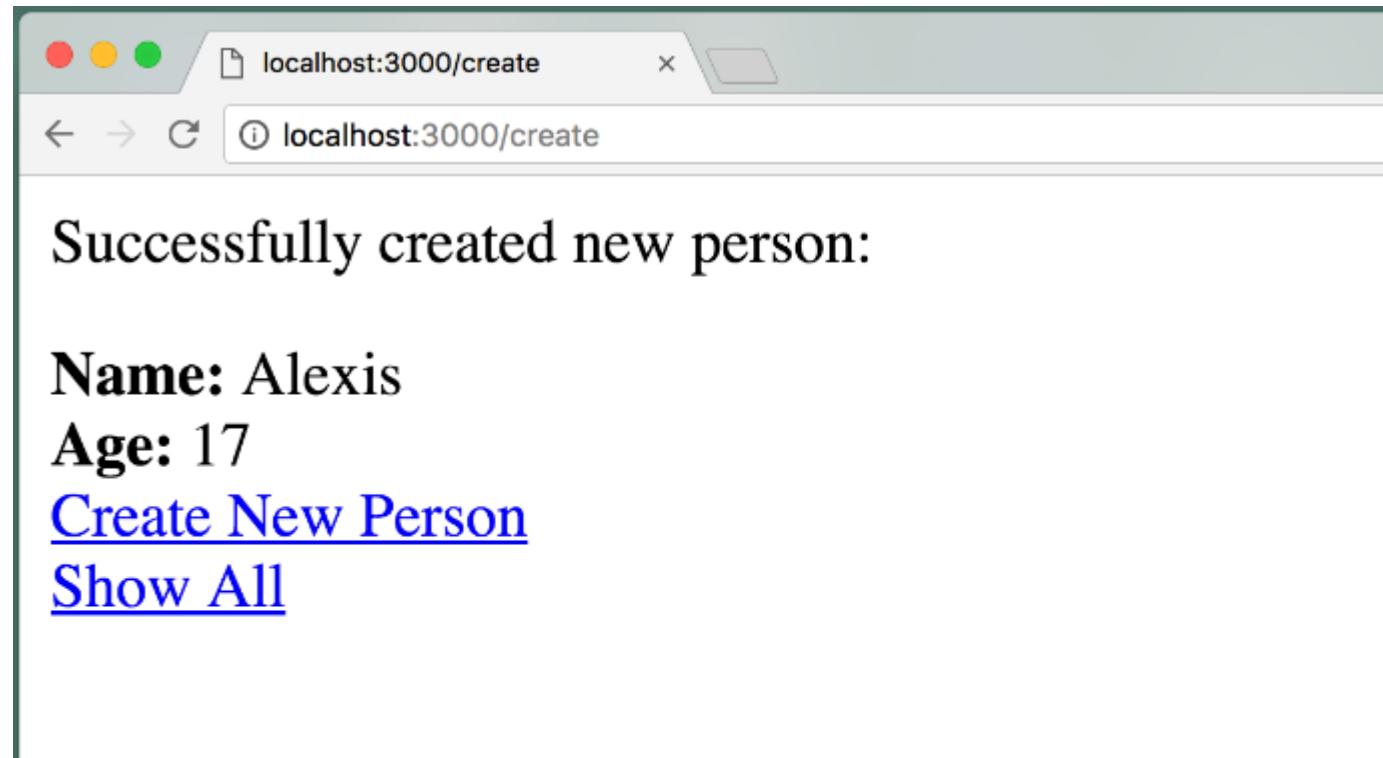
```
<p>
<b>Name:</b> <%= person.name %>
<br>
<b>Age:</b> <%= person.age %>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```



```
<!-- This is views/created.ejs -->
```

```
Successfully created new person:
```

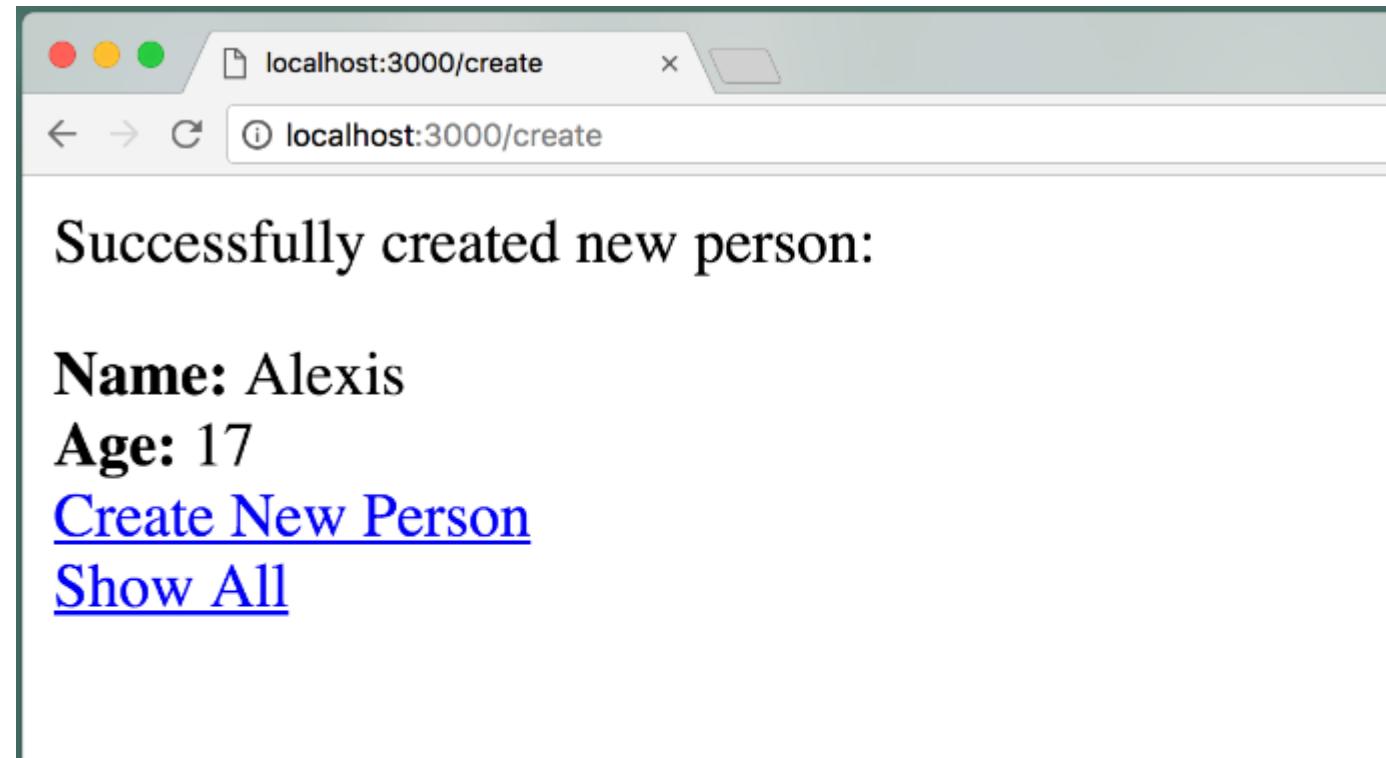
```
<p>
<b>Name:</b> <%= person.name %>
<br>
<b>Age:</b> <%= person.age %>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```



```
<!-- This is views/created.ejs -->
```

```
Successfully created new person:
```

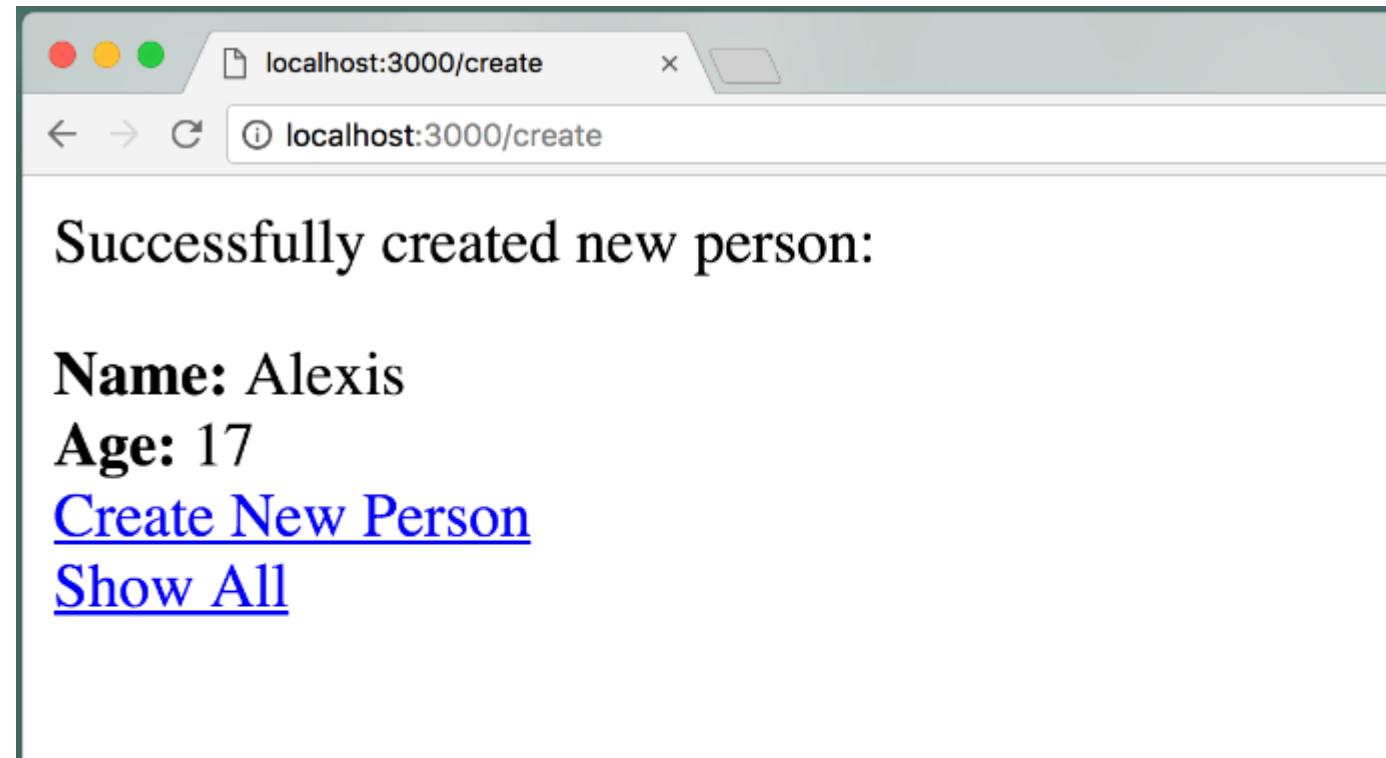
```
<p>
<b>Name:</b> <%= person.name %>
<br>
<b>Age:</b> <%= person.age %>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```



```
<!-- This is views/created.ejs -->
```

```
Successfully created new person:
```

```
<p>
<b>Name:</b> <%= person.name %>
<br>
<b>Age:</b> <%= person.age %>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```



```
[Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK  [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)
2017-07-11T18:18:06.603-0400 I NETWORK  [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
> ]
```

```
[Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
```

MongoDB shell version: 3.4.6

connecting to: mongodb://127.0.0.1:27017

2017-07-11T18:18:06.603-0400 I NETWORK [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)

2017-07-11T18:18:06.603-0400 I NETWORK [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }

MongoDB server version: 3.4.6

Server has startup warnings:

2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]

2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.

2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.

2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]

> █

```
[Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK  [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)
2017-07-11T18:18:06.603-0400 I NETWORK  [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
[> use myDatabase
switched to db myDatabase
>
```

```
[Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK  [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)
2017-07-11T18:18:06.603-0400 I NETWORK  [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
[> use myDatabase
Switched to db myDatabase
> db.people.find()
```

```
[Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK  [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)
2017-07-11T18:18:06.603-0400 I NETWORK  [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
[> use myDatabase
switched to db myDatabase
[> db.people.find()
{ "_id" : ObjectId("59654e6a0c7a004db2a20f86"), "name" : "Alexis", "age" : 17, "__v" : 0 }
```

```
[Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK  [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)
2017-07-11T18:18:06.603-0400 I NETWORK  [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
[> use myDatabase
switched to db myDatabase
[> db.people.find()
{ "_id" : ObjectId("59654e6a0c7a004db2a20f86"), "name" : "Alexis", "age" : 17, "__v" : 0 }
```

Summary

- **MongoDB** is a NoSQL Database that is designed for use with JavaScript apps
- MongoDB stores **collections** of **documents**
- We can access MongoDB from our Node/Express app using libraries such as Mongoose
- We define a **Schema** and then can create new documents using the **save** function



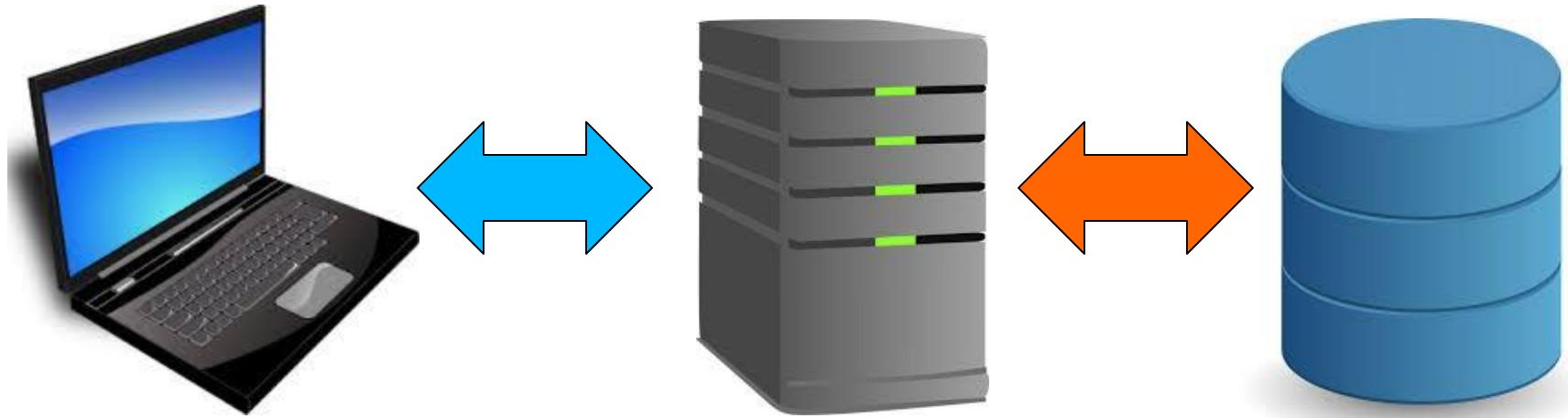
Video 4.7

MongoDB: Select and Update

Chris Murphy

Review

- **MongoDB** is a NoSQL Database that is designed for use with JavaScript apps
- MongoDB stores **collections** of **documents**
- We can access MongoDB from our Node/Express app using libraries such as Mongoose
- We define a **Schema** and then can create new documents using the **save** function



Client

Server

Database



localhost:3000/public/personform.html

Guest

← → ⌂

localhost:3000/public/personform.html



Name:

Age:

Submit form!



Name: Alexis

Age: 17

Submit form!



← → ⚡



Successfully created new person:

Name: Alexis

Age: 17

[Create New Person](#)

[Show All](#)



Successfully created new person:

Name: Alexis

Age: 17

[Create New Person](#)

[Show All](#)



← → ⌂

Here are all the people:

- Alexis: 17
- Olivier: 12
- Mesut: 11

[Create New Person](#)



Here are all the people:

- Alexis: 17
- Olivier: 12
- Mesut: 11

[Create New Person](#)

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) );
}) ;
```

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) );
}) ;
```

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) );
}) ;
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */
```

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

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
    name: {type: String, required: true, unique: true},
    age: Number
} );

module.exports = mongoose.model('Person', personSchema);
```

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) );
}) ;
```

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) ;
}) ;
```

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) );
}) ;
```

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) );
}) ;
```

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) );
}) ;
```

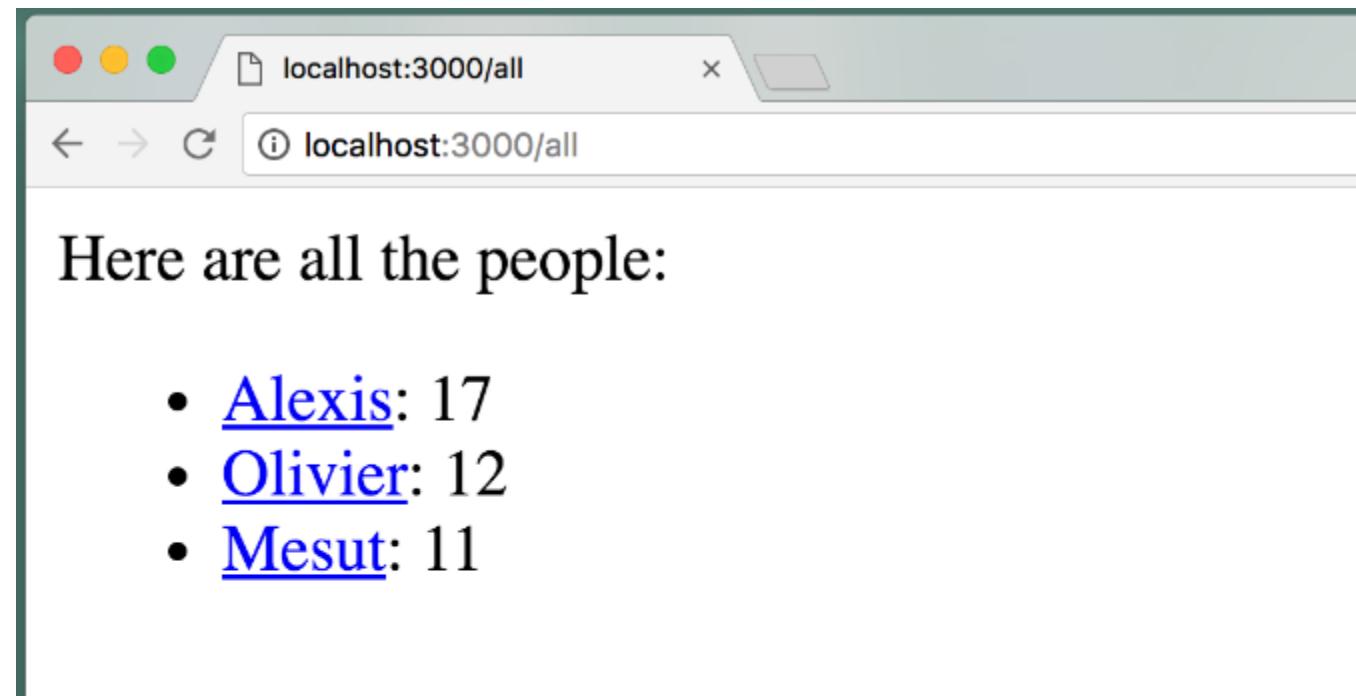
```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) );
}) ;
```

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  }) ;
}) ;
```

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) );
}) ;
```

```
app.use('/all', (req, res) => {
  Person.find( (err, allPeople) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (allPeople.length == 0) {
      res.type('html').status(200);
      res.send('There are no people');
    }
    else {
      res.render('showAll', { people: allPeople });
    }
  ) );
}) ;
```

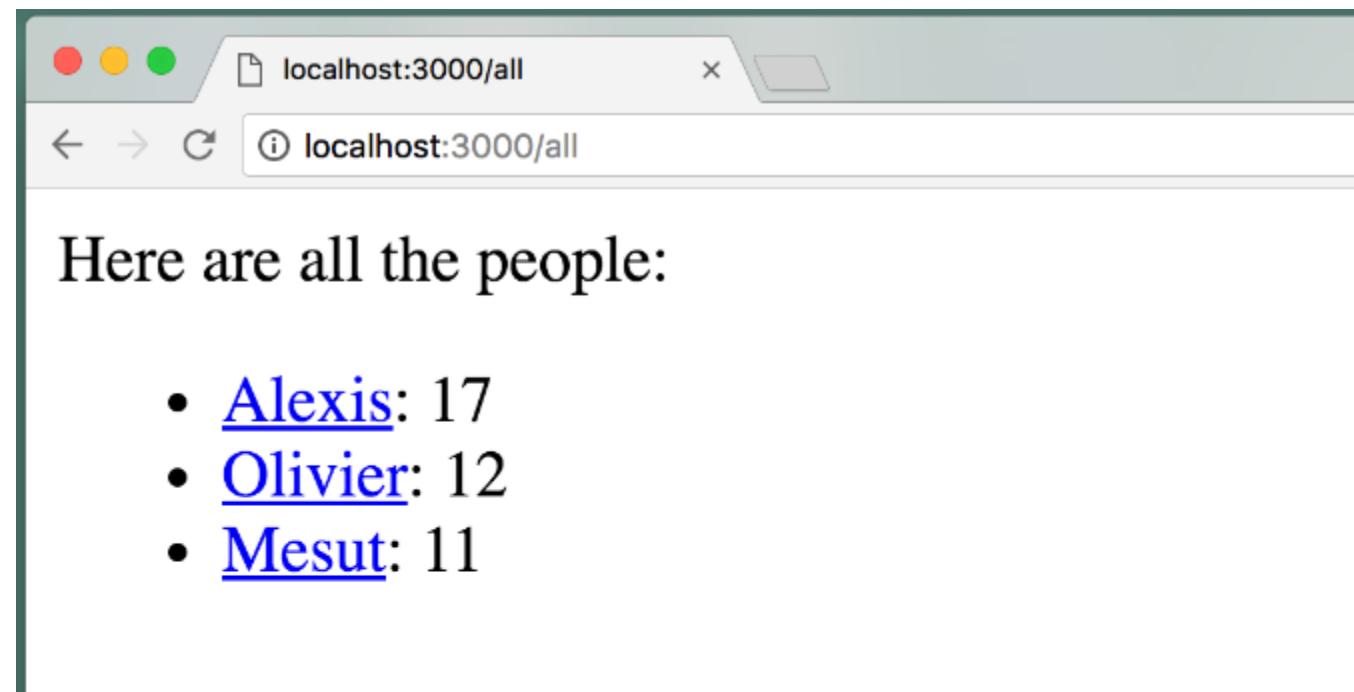
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



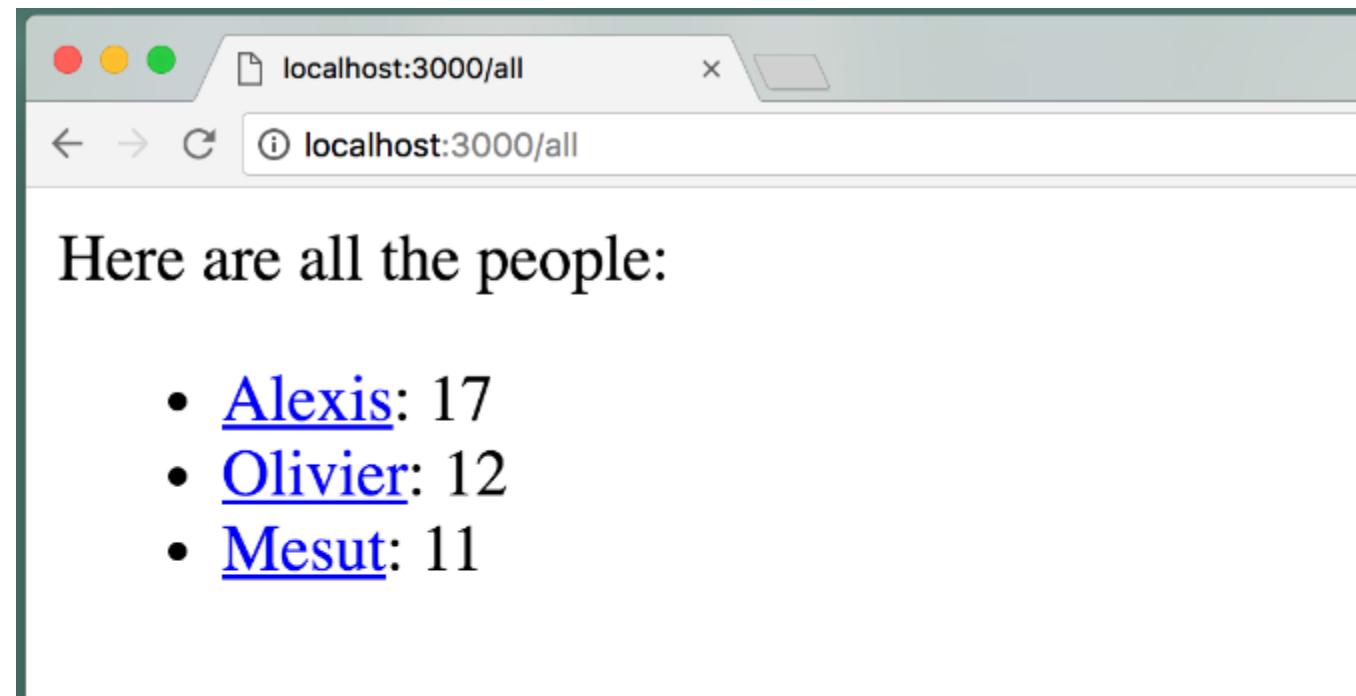
```
<!-- This is views/showAll.ejs -->
```

```
Here are all the people:
```

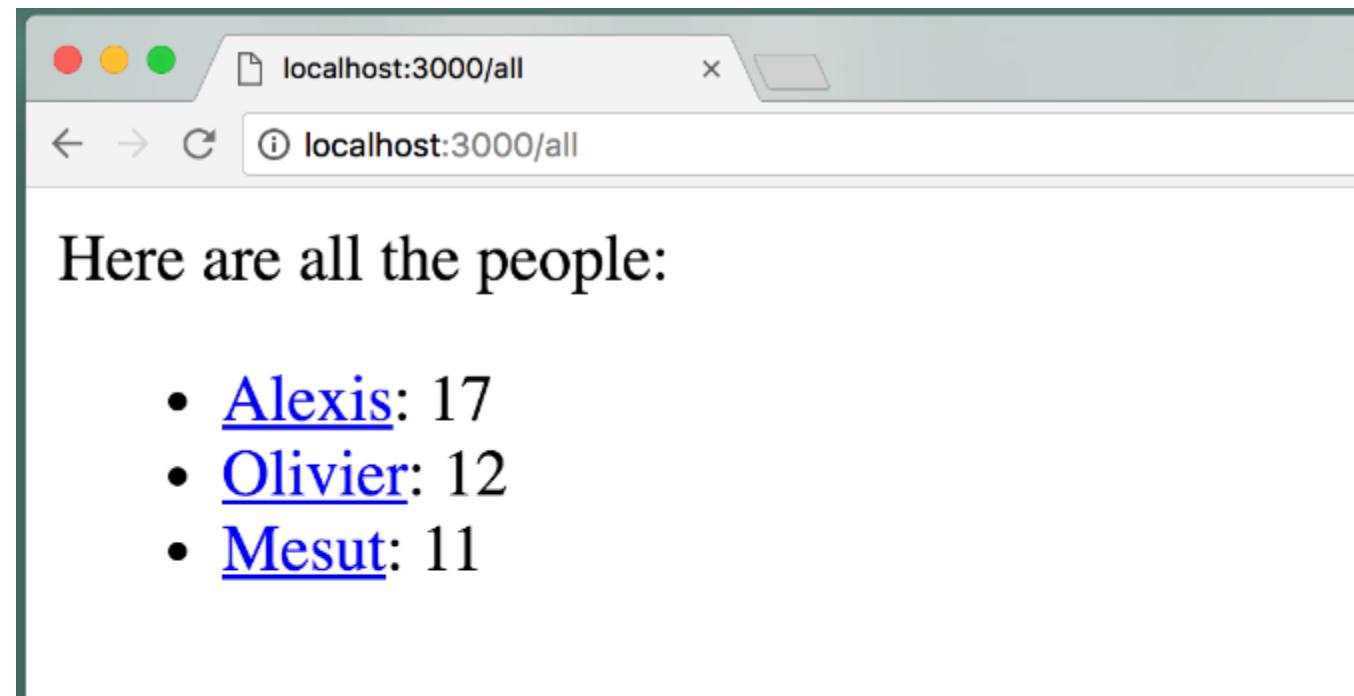
```
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



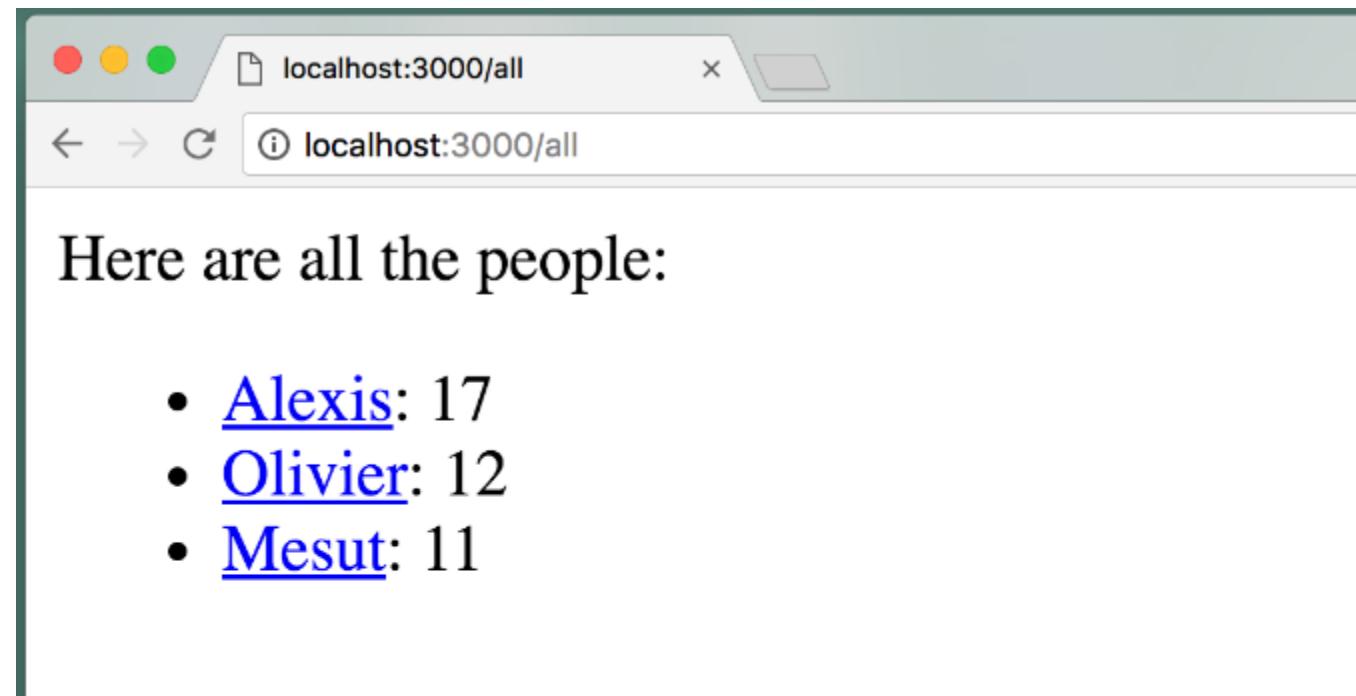
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



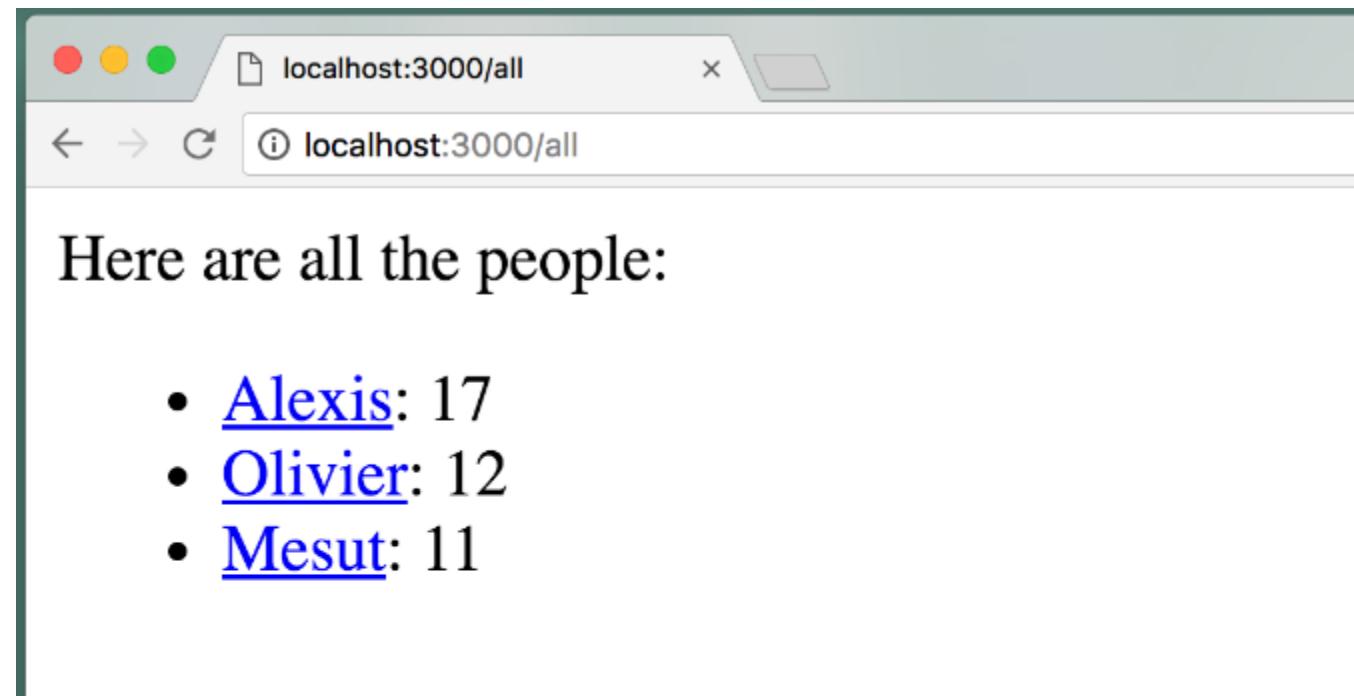
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



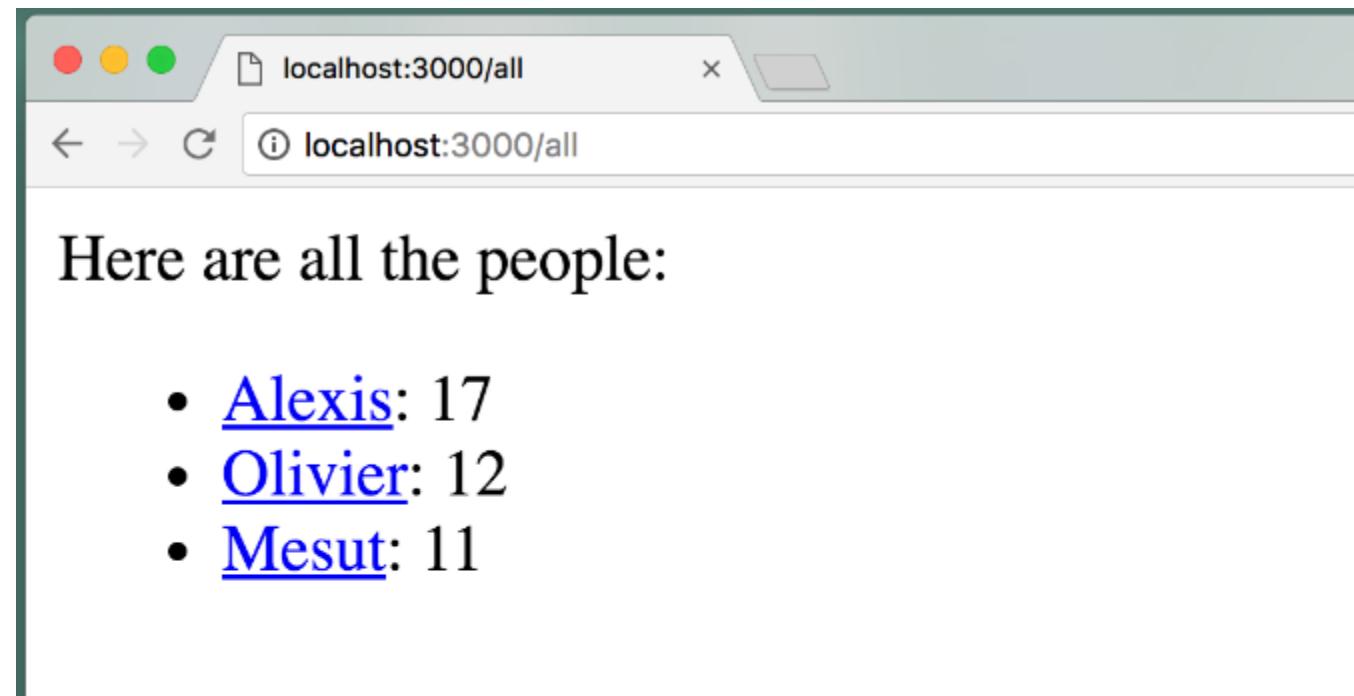
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



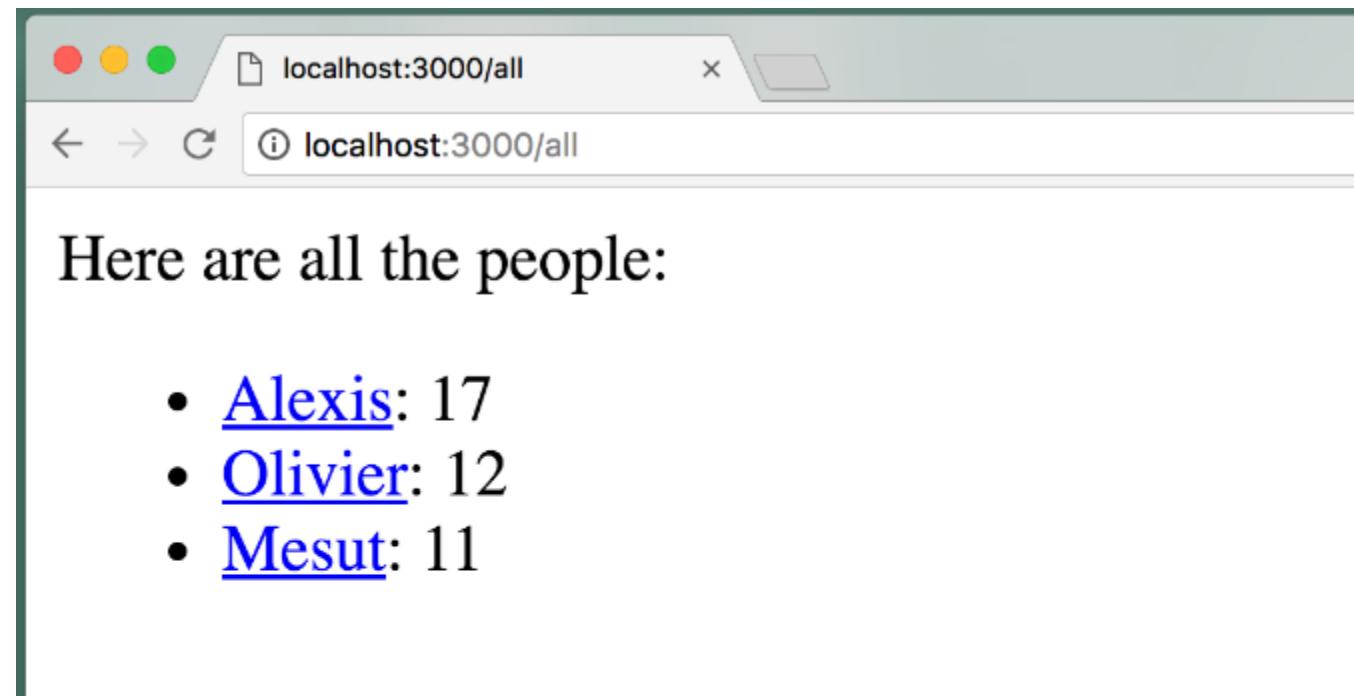
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



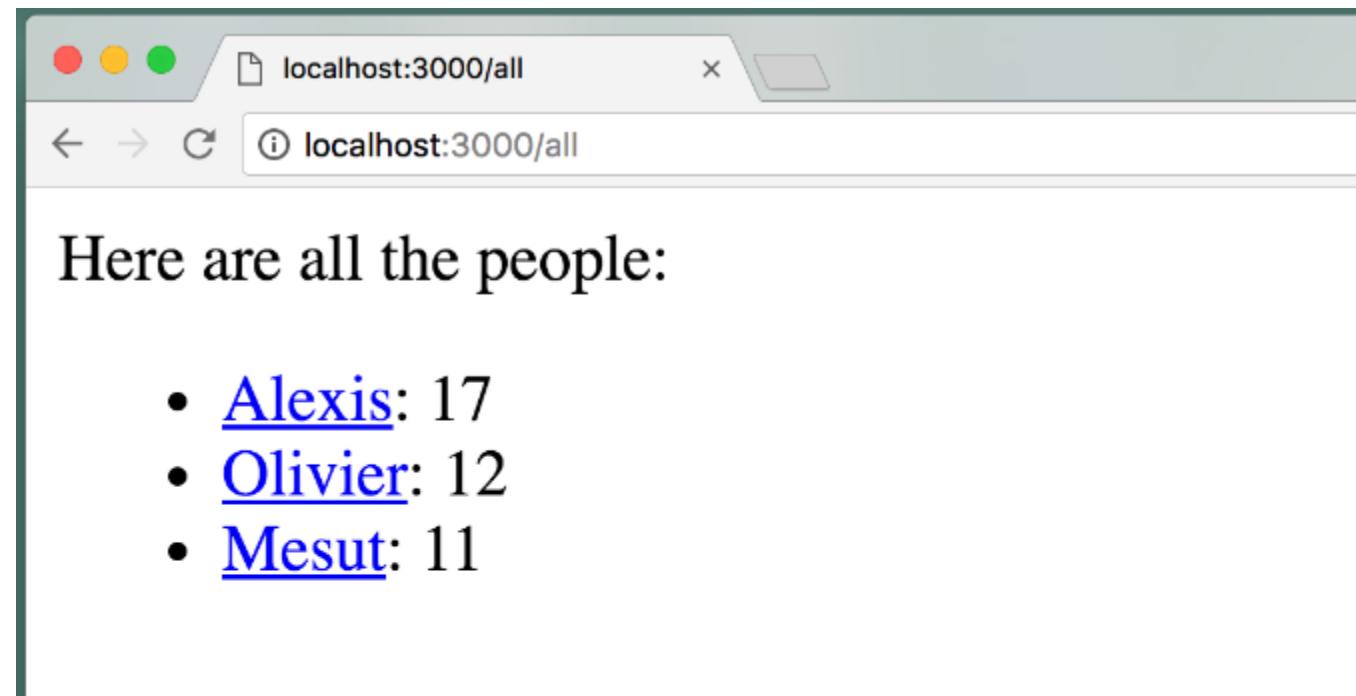
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



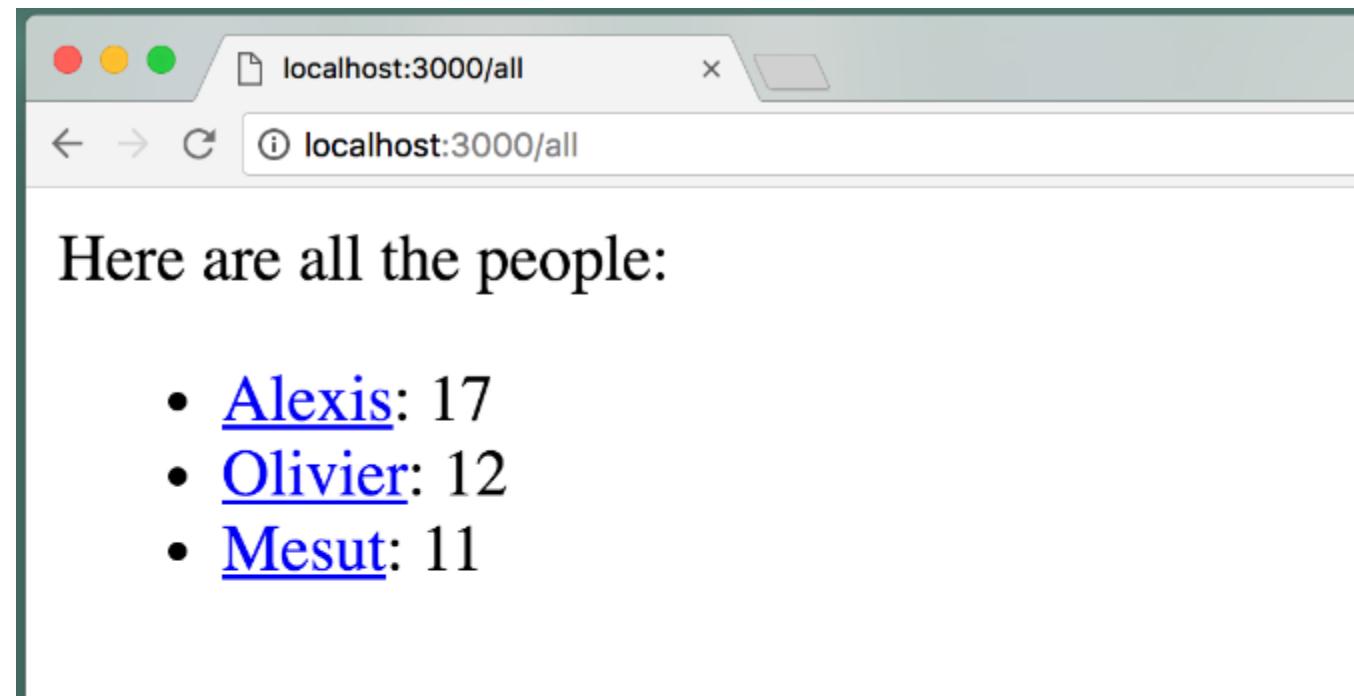
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



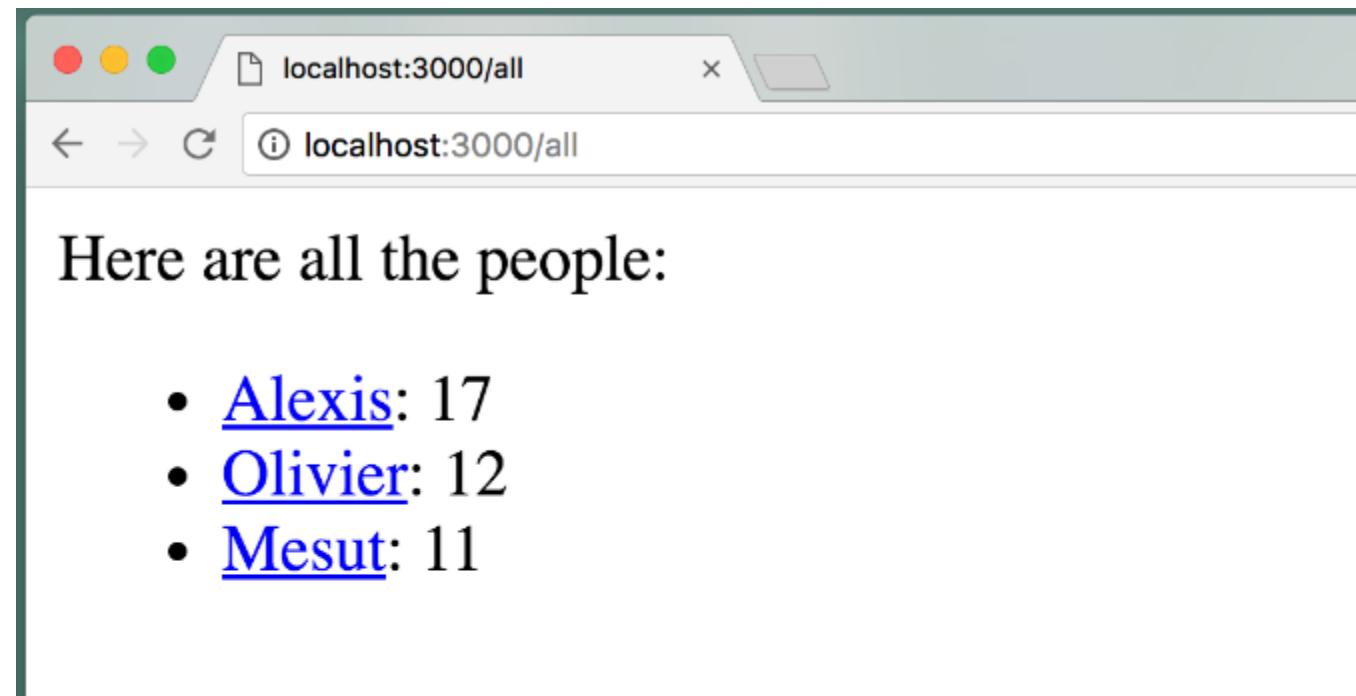
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



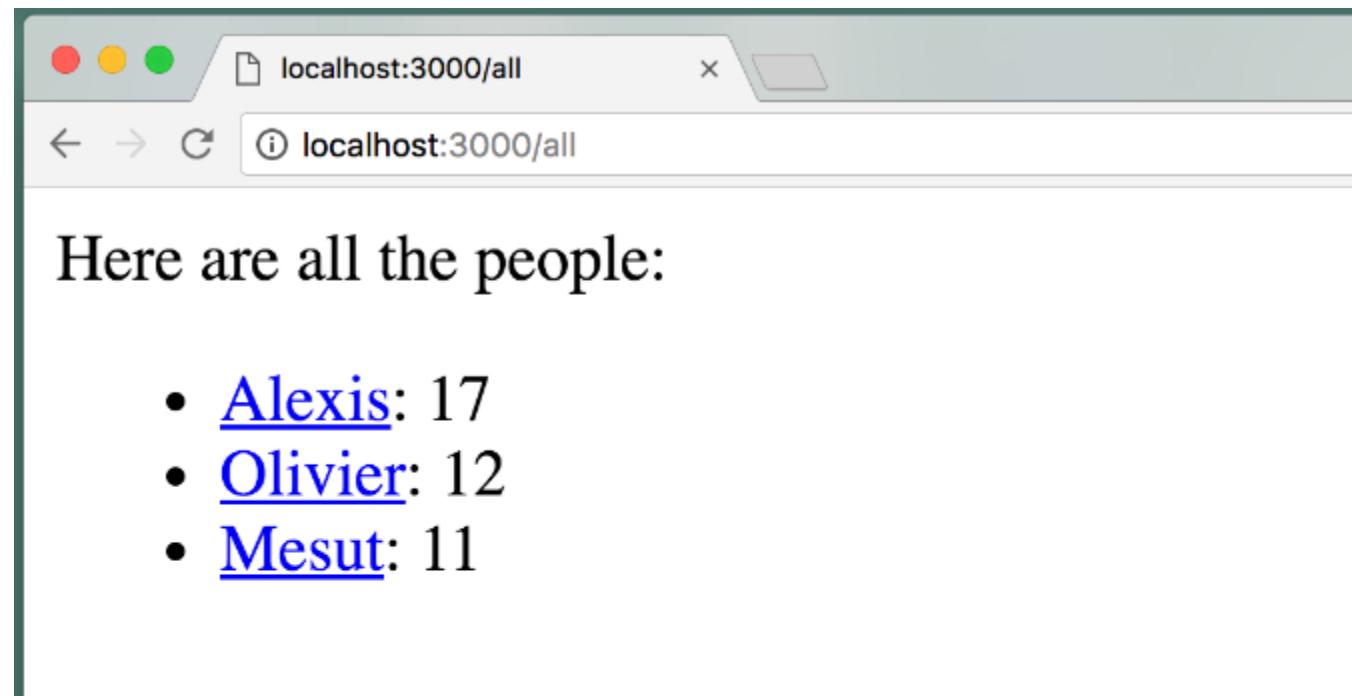
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



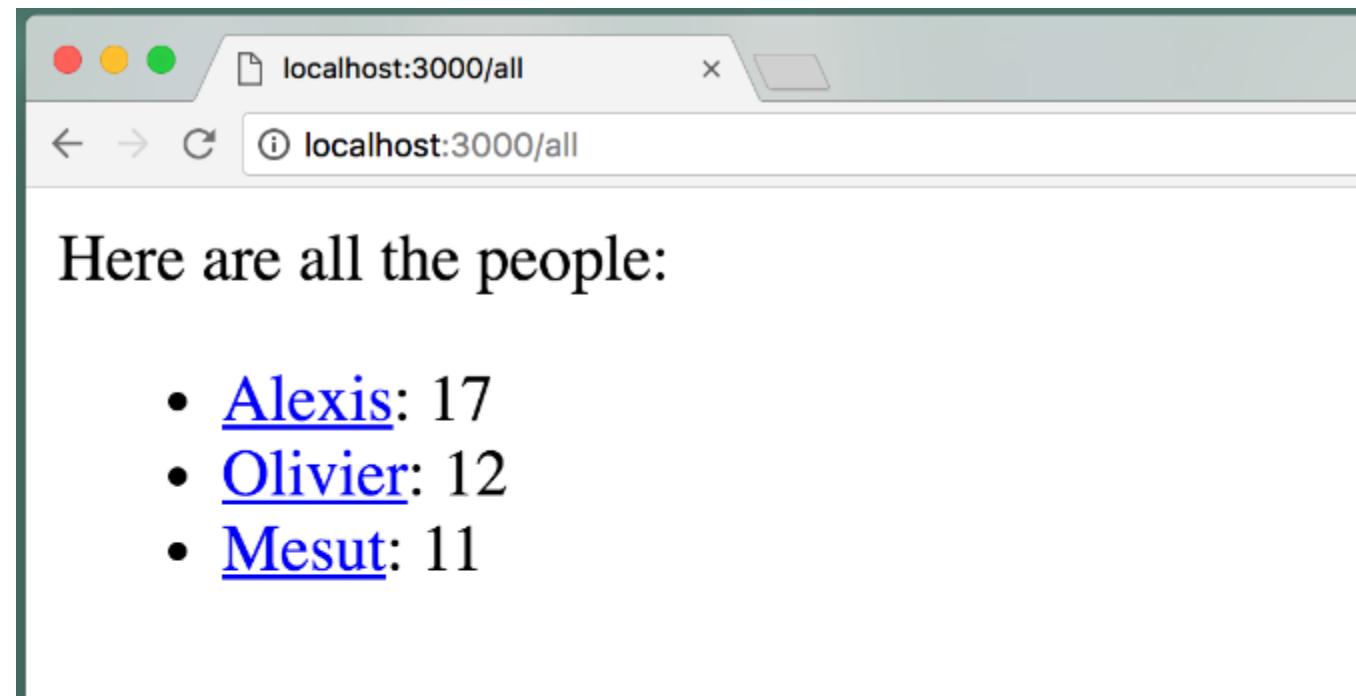
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



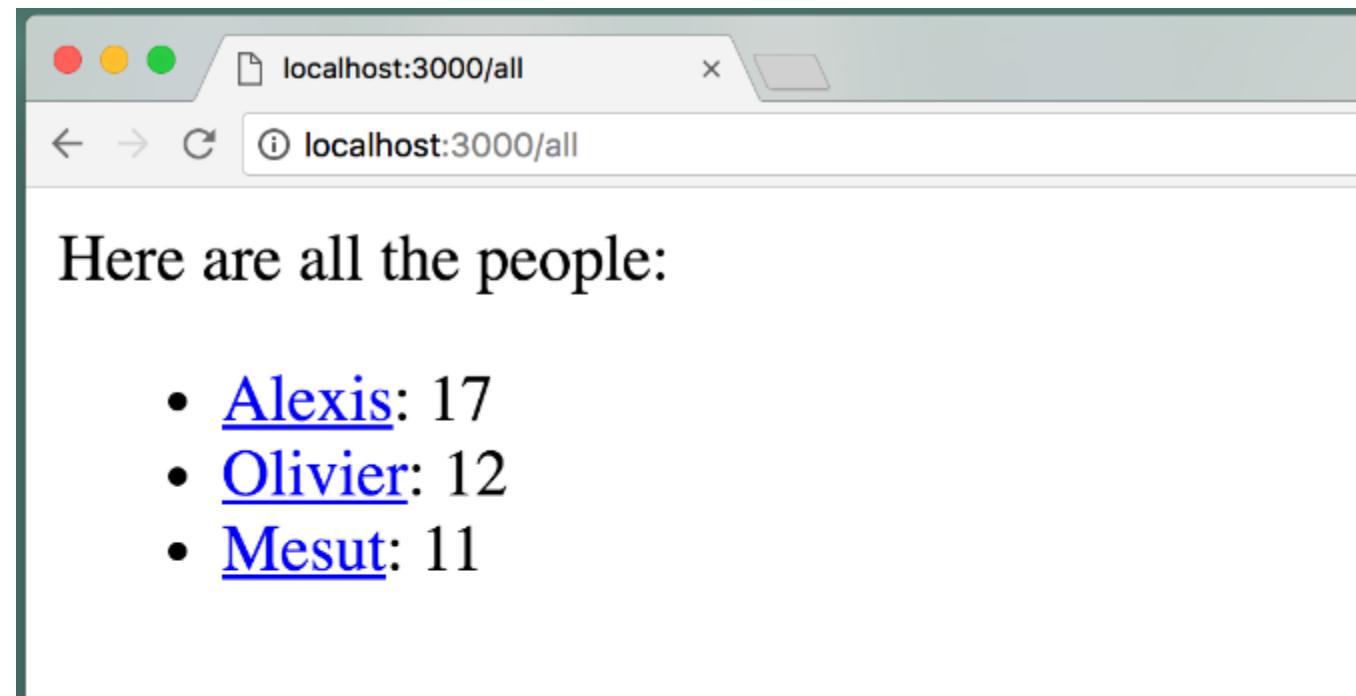
```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
    <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```



```
<!-- This is views/showAll.ejs -->
Here are all the people:
<ul>
<% people.forEach( (person) => { %>
  <li><a href="/person?name=<%= person.name %>">
    <%= person.name %></a>:
      <%= person.age %>
  </li>
<% } ); %>
</ul>
<br><a href='/public/personform.html'>Create New Person</a>
```





← → ⌂

Here are all the people:

- Alexis: 17
- Olivier: 12
- Mesut: 11

[Create New Person](#)



localhost:3000/all

x

Guest

← → ↶

localhost:3000/all



Here are all the people:

- Alexis: 17
- Olivier: 12
- Mesut: 11

[Create New Person](#)

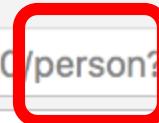
Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)



Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
} );
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
} );
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName }, (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
} );
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) ;
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
} );
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
}) ;
```

```
app.use('/person', (req, res) => {
  var searchName = req.query.name;
  Person.findOne( { name: searchName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + searchName);
    }
    else {
      res.render('personInfo', { person : person });
    }
  ) );
} );
```

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' type='hidden'>
Age: <input name='age' value='<%= person.age %>' type='text'>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The main content area displays a form for updating a person's information. The form has two fields: "Name:" followed by the value "Alexis" and "Age:" followed by the value "17". To the right of the age field is a grey button with the word "Update" in white. Below the form, there are two links: "Create New Person" and "Show All", both displayed in blue text.

Name: Alexis

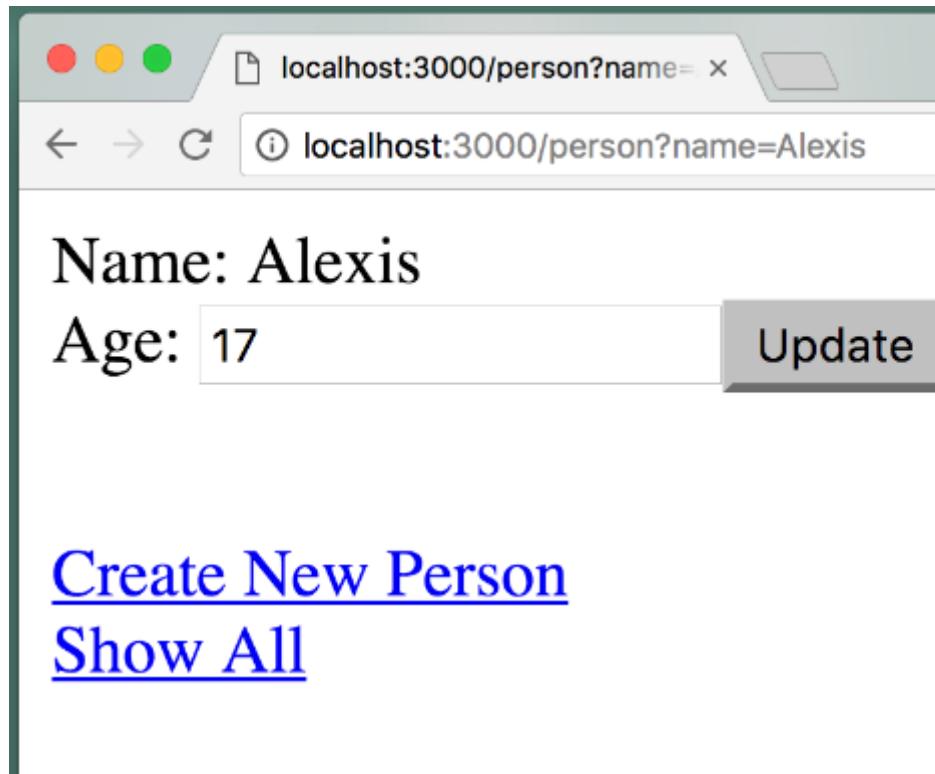
Age: 17

Update

[Create New Person](#)

[Show All](#)

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' type='hidden'>
Age: <input name='age' value='<%= person.age %>' type='text'>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```



```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' type='hidden'>
Age: <input name='age' value='<%= person.age %>' type='text'>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The page content displays a form for updating a person's details. The 'Name:' label is followed by the value 'Alexis'. Below it is an 'Age:' label with a text input field containing the value '17'. To the right of the input field is a grey 'Update' button. At the bottom of the page, there are two links: 'Create New Person' and 'Show All'.

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' type='hidden'>
Age: <input name='age' value='<%= person.age %>' type='text'>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The main content area displays a form for updating a person's information. The form has two fields: "Name:" followed by the value "Alexis" and "Age:" followed by the value "17". To the right of the age field is a grey "Update" button. Below the form, there are two links: "Create New Person" and "Show All", both displayed in blue text.

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' hidden>
Age: <input name='age' value='<%= person.age %>'>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The page content displays a form for updating a person's details. The 'Name:' label is followed by the value 'Alexis'. Below it is an 'Age:' label with a text input field containing the value '17'. To the right of the age input is a grey 'Update' button. At the bottom of the page, there are two links: 'Create New Person' and 'Show All'.

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' hidden>
Age: <input name='age' value='<%= person.age %>'>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The main content area displays a form for updating a person's information. The form has two fields: "Name:" followed by the value "Alexis" and "Age:" followed by the value "17". To the right of the age field is a grey button labeled "Update". Below the form, there are two links: "Create New Person" and "Show All", both displayed in blue text.

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<b><input name='username' value='<%= person.name %>' type='hidden'></b>
Age: <input name='age' value='<%= person.age %>'>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The main content area displays a form for updating a person's details. The form has two fields: "Name:" followed by the value "Alexis" and "Age:" followed by the value "17". To the right of the age field is a grey "Update" button. Below the form, there are two links: "Create New Person" and "Show All", both displayed in blue text.

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' type='hidden'>
Age: <input name='age' value='<%= person.age %>'>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The page content displays a form for updating a person's information. The 'Name:' label is followed by the value 'Alexis'. Below it is an 'Age:' label with a text input field containing the value '17'. To the right of the age input is a grey 'Update' button. At the bottom of the page, there are two links: 'Create New Person' and 'Show All'.

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' hidden>
Age: <input name='age' value='<%= person.age %>'>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The main content area displays a form for updating a person's details. The form has two input fields: one for Name containing "Alexis" and one for Age containing "17". To the right of the Age field is a "Update" button. Below the form, there are links for "Create New Person" and "Show All".

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' hidden>
Age: <b><input name='age' value='<%= person.age %>'></b>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The main content area displays a form for updating a person's information. The form has two fields: "Name" with the value "Alexis" and "Age" with the value "17". To the right of the age field is a "Update" button. Below the form, there are links for "Create New Person" and "Show All".

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' type='hidden'>
Age: <input name='age' value='<%= person.age %>' type='text'>
<input type='submit' value='Update'>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The page content displays a form for updating a person's information. The 'Name:' label is followed by the value 'Alexis'. Below it is an 'Age:' label with a text input field containing the value '17'. To the right of the input field is a grey 'Update' button. At the bottom of the page, there are two links: 'Create New Person' and 'Show All'.

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

```
<!-- This is views/personInfo.ejs -->
<form action='/update' method='post'>
Name: <%= person.name %><br>
<input name='username' value='<%= person.name %>' type='hidden'>
Age: <input name='age' value='<%= person.age %>'>
<b><input type='submit' value='Update'></b>
</form>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

A screenshot of a web browser window. The address bar shows the URL `localhost:3000/person?name=Alexis`. The page content displays a form for updating a person's information. The 'Name:' label is followed by the value 'Alexis'. Below it is an 'Age:' label with a text input field containing the value '17'. To the right of the age input is a grey 'Update' button. At the bottom of the page, there are two links: 'Create New Person' and 'Show All'.

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)

Name: Alexis

Age: 17

Update

[Create New Person](#)

[Show All](#)



localhost:3000/person?name=

Guest

← → ⌂ ⓘ localhost:3000/person?name=Alexis



Name: Alexis

Age: 18

Update

[Create New Person](#)

[Show All](#)



localhost:3000/person?name=

Guest



localhost:3000/person?name=Alexis



Name: Alexis

Age: 18

Update

[Create New Person](#)

[Show All](#)



localhost:3000/update



Guest



localhost:3000/update



Updated Alexis's age to 18

[Create New Person](#)

[Show All](#)



localhost:3000/update

Guest



localhost:3000/update



Updated Alexis's age to 18

[Create New Person](#)

[Show All](#)

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      });
    }
  });
});
```

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      });
    }
  });
});
```

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName }, (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      });
    }
  });
});
```

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      }) ;
    }
  }) ;
}) ;
```

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } ), (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      });
    }
  });
});
```

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      }) ;
    }
  }) ;
}) ;
```

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      });
    }
  });
});
```

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      });
    }
  });
});
```

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      });
    }
  });
});
```

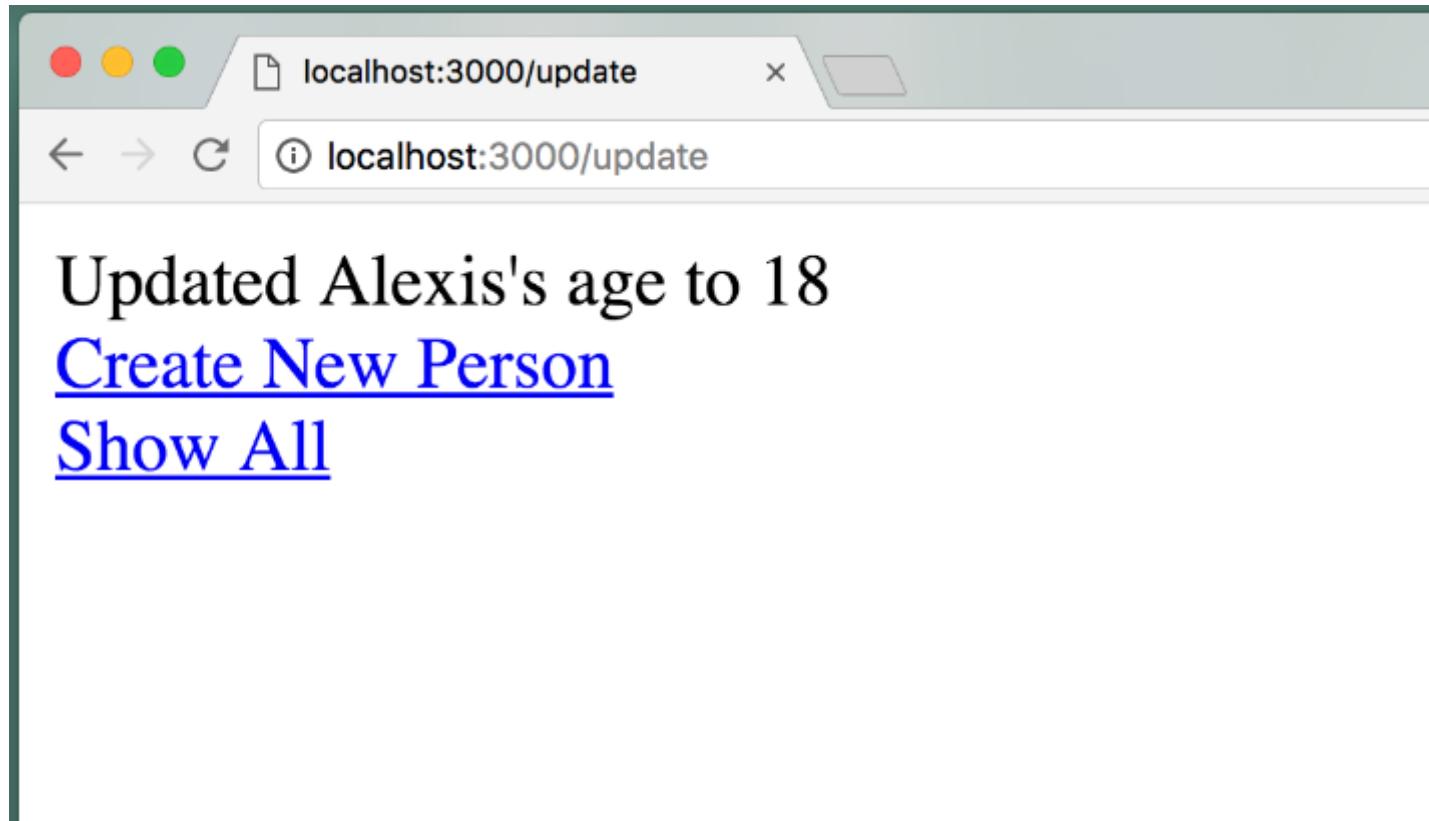
```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName }, (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      }) ;
    }
  }) ;
}) ;
```

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      });
    }
  });
});
```

```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      }) ;
    }
  }) ;
}) ;
```

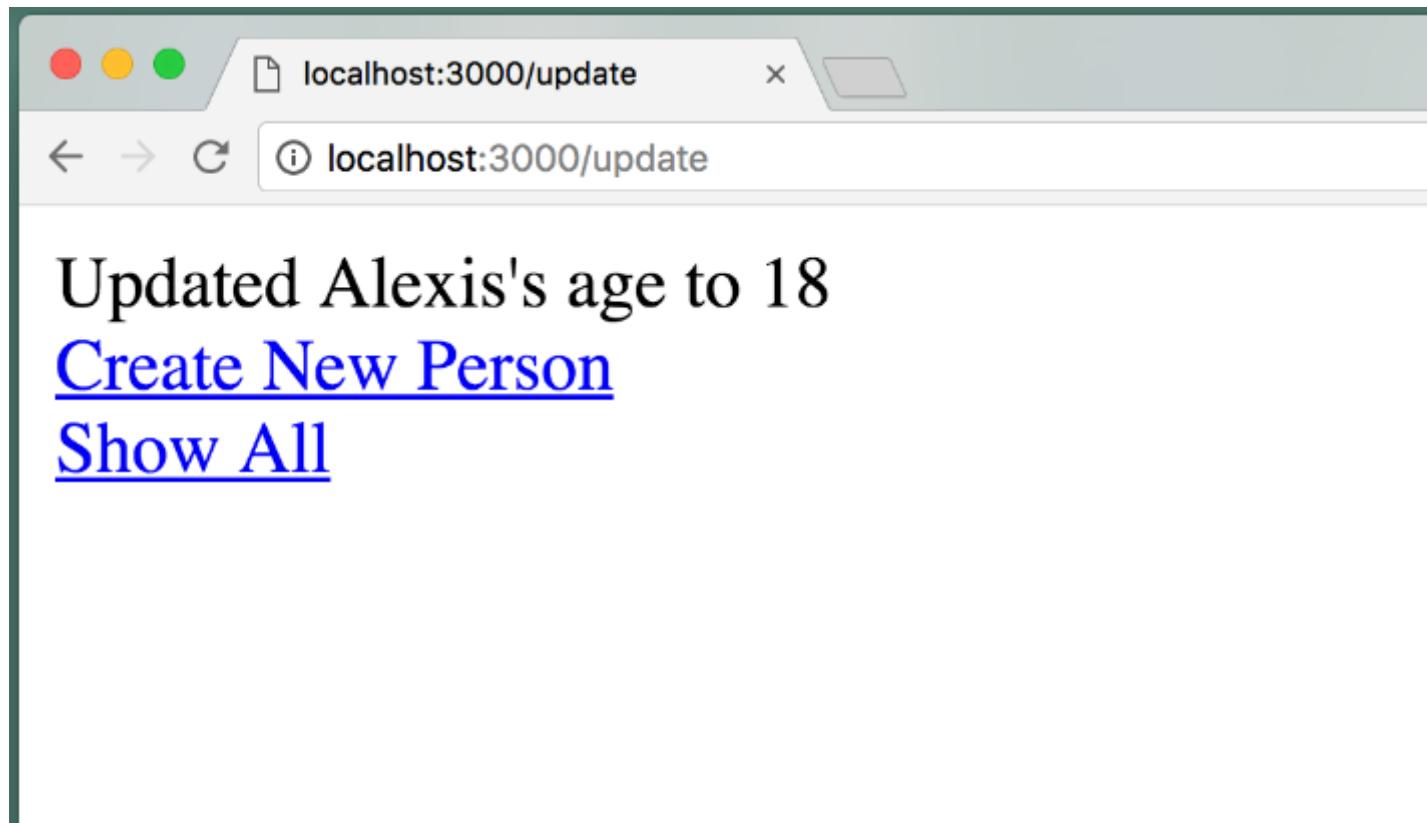
```
app.use('/update', (req, res) => {
  var updateName = req.body.username;
  Person.findOne( { name: updateName } , (err, person) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else if (!person) {
      res.type('html').status(200);
      res.send('No person named ' + updateName);
    }
    else {
      person.age = req.body.age;
      person.save( (err) => {
        if (err) {
          res.type('html').status(500);
          res.send('Error: ' + err);
        }
        else {
          res.render('updated', { person: person });
        }
      });
    }
  });
});
```

```
<!-- This is views/updated.ejs -->  
  
Updated <%= person.name %>'s age to <%= person.age %>  
  
<br><a href='/public/personform.html'>Create New Person</a>  
<br><a href='/all'>Show All</a>
```

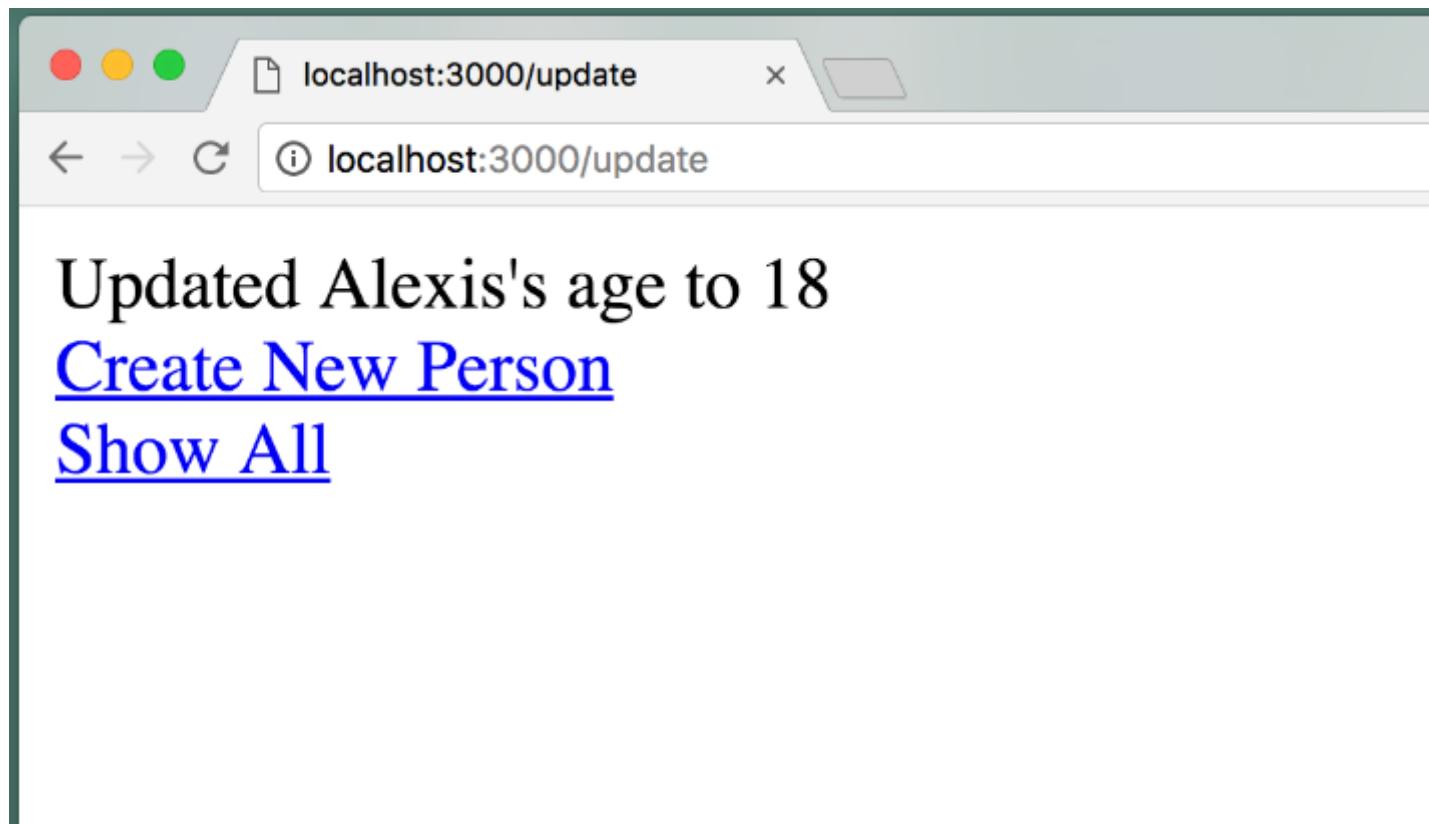


```
<!-- This is views/updated.ejs -->
```

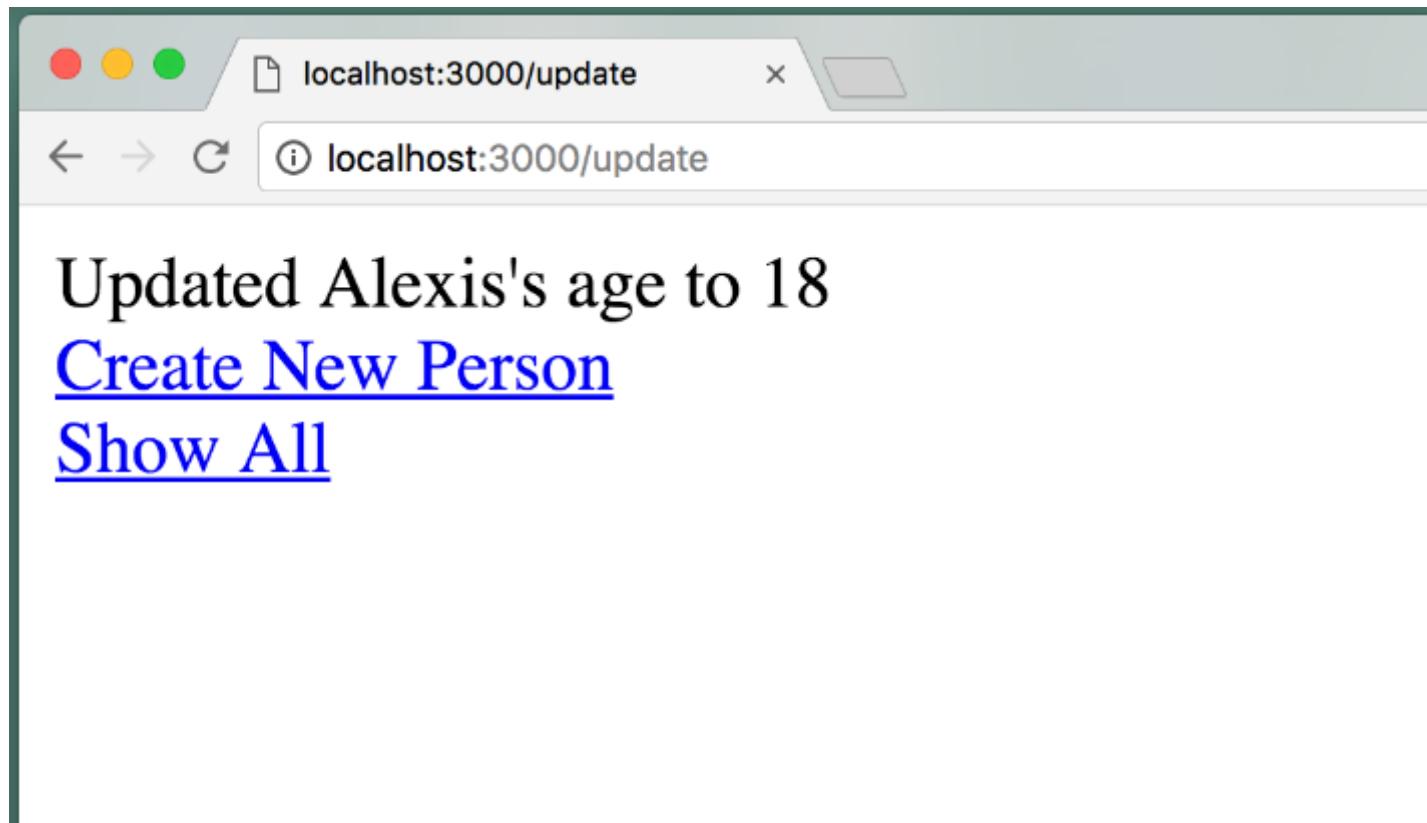
```
Updated <%= person.name %>'s age to <%= person.age %>  
<br><a href='/public/personform.html'>Create New Person</a>  
<br><a href='/all'>Show All</a>
```



```
<!-- This is views/updated.ejs -->  
  
Updated <%= person.name %>'s age to <%= person.age %>  
  
<br><a href='/public/personform.html'>Create New Person</a>  
<br><a href='/all'>Show All</a>
```



```
<!-- This is views/updated.ejs -->  
  
Updated <%= person.name %>'s age to <b><%= person.age %></b>  
  
<br><a href='/public/personform.html'>Create New Person</a>  
<br><a href='/all'>Show All</a>
```



Summary

- We can access MongoDB from our Node/Express app using libraries such as Mongoose
- We can use the **find** function to select all documents in a collection, or pass a query object to select only certain ones
- Once we have a document, we can update it using the **save** function



Video 4.8

MongoDB: Advanced Queries

Chris Murphy

Review

- MongoDB is a NoSQL Database that stores collections of documents
- Once we've defined a Schema we can use the **find** function to select all documents in a collection, or pass a query object to select only certain ones
- Once we have a document, we can update it using the **save** function

```
{  
  title: 'Introduction to Algorithms',  
  year: 1990,  
  authors: [  
    { name: 'Thomas Cormen', affiliation: 'Dartmouth' },  
    { name: 'Charles Leiserson', affiliation: 'MIT' },  
    { name: 'Ronald Rivest', affiliation: 'MIT' },  
    { name: 'Clifford Stein', affiliation: 'Columbia' }  
  ]  
}
```

```
{  
  title: 'Principles of Compiler Design',  
  year: 1977,  
  authors: [  
    { name: 'Alfred Aho', affiliation: 'Bell Labs' },  
    { name: 'Jeffrey Ullman', affiliation: 'Princeton' }  
  ]  
}
```

```
{  
  title: 'Introduction to Algorithms',  
  year: 1990,  
  authors: [  
    { name: 'Thomas Cormen', affiliation: 'Dartmouth' },  
    { name: 'Charles Leiserson', affiliation: 'MIT' },  
    { name: 'Ronald Rivest', affiliation: 'MIT' },  
    { name: 'Clifford Stein', affiliation: 'Columbia' }  
  ]  
}
```

```
{  
  title: 'Principles of Compiler Design',  
  year: 1977,  
  authors: [  
    { name: 'Alfred Aho', affiliation: 'Bell Labs' },  
    { name: 'Jeffrey Ullman', affiliation: 'Princeton' }  
  ]  
}
```

```
{  
  title: 'Introduction to Algorithms',  
  year: 1990,  
  authors: [  
    { name: 'Thomas Cormen', affiliation: 'Dartmouth' },  
    { name: 'Charles Leiserson', affiliation: 'MIT' },  
    { name: 'Ronald Rivest', affiliation: 'MIT' },  
    { name: 'Clifford Stein', affiliation: 'Columbia' }  
  ]  
}
```

```
{  
  title: 'Principles of Compiler Design',  
  year: 1977,  
  authors: [  
    { name: 'Alfred Aho', affiliation: 'Bell Labs' },  
    { name: 'Jeffrey Ullman', affiliation: 'Princeton' }  
  ]  
}
```

```
{  
    title: 'Introduction to Algorithms',  
    year: 1990,  
    authors: [  
        { name: 'Thomas Cormen', affiliation: 'Dartmouth' },  
        { name: 'Charles Leiserson', affiliation: 'MIT' },  
        { name: 'Ronald Rivest', affiliation: 'MIT' },  
        { name: 'Clifford Stein', affiliation: 'Columbia' }  
    ]  
}
```

```
{  
    title: 'Principles of Compiler Design',  
    year: 1977,  
    authors: [  
        { name: 'Alfred Aho', affiliation: 'Bell Labs' },  
        { name: 'Jeffrey Ullman', affiliation: 'Princeton' }  
    ]  
}
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
});

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
  name: String,
  affiliation: String
}) ;

var bookSchema = new Schema({
  title: {type: String, required: true, unique: true},
  year: Number,
  authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```

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

mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var authorSchema = new Schema({
    name: String,
    affiliation: String
}) ;

var bookSchema = new Schema({
    title: {type: String, required: true, unique: true},
    year: Number,
    authors: [authorSchema]
}) ;

module.exports = mongoose.model('Book', bookSchema);
```



localhost:3000/public/bookfor x

Guest

← → ⌂ ⓘ localhost:3000/public/bookform.html



Title:

Year:

Author #1:

Name:

Affiliation:

Author #2:

Name:

Affiliation:

Submit

```
<form action='/createbook' method='post'>  
  
Title: <input name='title'>  
<p>  
  
Year: <input name='year'>  
<p>  
  
Author #1: <br>  
Name: <input name='authors[0][name]'> <br>  
Affiliation: <input name='authors[0][affiliation]'>  
<p>  
  
Author #2: <br>  
Name: <input name='authors[1][name]'> <br>  
Affiliation: <input name='authors[1][affiliation]'>  
<p>  
  
<input type='submit' value='Submit'>  
  
</form>
```

```
<form action='/createbook' method='post'>  
  
Title: <input name='title'>  
<p>  
  
Year: <input name='year'>  
<p>  
  
Author #1: <br>  
Name: <input name='authors[0][name]'> <br>  
Affiliation: <input name='authors[0][affiliation]'>  
<p>  
  
Author #2: <br>  
Name: <input name='authors[1][name]'> <br>  
Affiliation: <input name='authors[1][affiliation]'>  
<p>  
  
<input type='submit' value='Submit'>  
  
</form>
```

```
<form action='/createbook' method='post'>  
  
Title: <input name='title'>  
<p>  
  
Year: <input name='year'>  
<p>  
  
Author #1: <br>  
Name: <input name='authors[0][name]'> <br>  
Affiliation: <input name='authors[0][affiliation]'>  
<p>  
  
Author #2: <br>  
Name: <input name='authors[1][name]'> <br>  
Affiliation: <input name='authors[1][affiliation]'>  
<p>  
  
<input type='submit' value='Submit'>  
  
</form>
```

```
<form action='/createbook' method='post'>  
  
Title: <input name='title'>  
<p>  
  
Year: <input name='year'>  
<p>  
  
Author #1: <br>  
Name: <input name='authors[0][name]'> <br>  
Affiliation: <input name='authors[0][affiliation]'>  
<p>  
  
Author #2: <br>  
Name: <input name='authors[1][name]'> <br>  
Affiliation: <input name='authors[1][affiliation]'>  
<p>  
  
<input type='submit' value='Submit'>  
  
</form>
```

```
<form action='/createbook' method='post'>  
  
Title: <input name='title'>  
<p>  
  
Year: <input name='year'>  
<p>  
  
Author #1: <br>  
Name: <input name='authors[0][name]'> <br>  
Affiliation: <input name='authors[0][affiliation]'>  
<p>  
  
Author #2: <br>  
Name: <input name='authors[1][name]'> <br>  
Affiliation: <input name='authors[1][affiliation]'>  
<p>  
  
<input type='submit' value='Submit'>  
  
</form>
```

```
<form action='/createbook' method='post'>  
  
Title: <input name='title'>  
<p>  
  
Year: <input name='year'>  
<p>  
  
Author #1: <br>  
Name: <input name='authors[0][name]'> <br>  
Affiliation: <input name='authors[0][affiliation]'>  
<p>  
  
Author #2: <br>  
Name: <input name='authors[1][name]'> <br>  
Affiliation: <input name='authors[1][affiliation]'>  
<p>  
  
<input type='submit' value='Submit'>  
  
</form>
```

```
<form action='/createbook' method='post'>  
  
Title: <input name='title'>  
<p>  
  
Year: <input name='year'>  
<p>  
  
Author #1: <br>  
Name: <input name='authors[0][name]'> <br>  
Affiliation: <input name='authors[0][affiliation]'>  
<p>  
  
Author #2: <br>  
Name: <input name='authors[1][name]'> <br>  
Affiliation: <input name='authors[1][affiliation]'>  
<p>  
  
<input type='submit' value='Submit'>  
  
</form>
```

```
<form action='/createbook' method='post'>  
  
Title: <input name='title'>  
<p>  
  
Year: <input name='year'>  
<p>  
  
Author #1: <br>  
Name: <input name='authors[0][name]'> <br>  
Affiliation: <input name='authors[0][affiliation]'>  
<p>  
  
Author #2: <br>  
Name: <input name='authors[1][name]'> <br>  
Affiliation: <input name='authors[1][affiliation]'>  
<p>  
  
<input type='submit' value='Submit'>  
  
</form>
```

```
<form action='/createbook' method='post'>  
  
Title: <input name='title'>  
<p>  
  
Year: <input name='year'>  
<p>  
  
Author #1: <br>  
Name: <input name='authors[0][name]'> <br>  
Affiliation: <input name='authors[0][affiliation]'>  
<p>  
  
Author #2: <br>  
Name: <input name='authors[1][name]'> <br>  
Affiliation: <input name='authors[1][affiliation]'>  
<p>  
  
<b><input type='submit' value='Submit'></b>  
  
</form>
```



Title:

Year:

Author #1:

Name:

Affiliation:

Author #2:

Name:

Affiliation:

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
    console.log(req.body);
    var newBook = new Book(req.body);
    newBook.save( (err) => {
        if (err) {
            res.type('html').status(500);
            res.send('Error: ' + err);
        }
        else {
            res.render('created', { book: newBook });
        }
    } );
} );
```

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
    console.log(req.body);
    var newBook = new Book(req.body);
    newBook.save( (err) => {
        if (err) {
            res.type('html').status(500);
            res.send('Error: ' + err);
        }
        else {
            res.render('created', { book: newBook });
        }
    } );
} );
```

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true })) ;

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
    console.log(req.body);
    var newBook = new Book(req.body);
    newBook.save( (err) => {
        if (err) {
            res.type('html').status(500);
            res.send('Error: ' + err);
        }
        else {
            res.render('created', { book: newBook });
        }
    } );
} );
```

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
    console.log(req.body);
    var newBook = new Book(req.body);
    newBook.save( (err) => {
        if (err) {
            res.type('html').status(500);
            res.send('Error: ' + err);
        }
        else {
            res.render('created', { book: newBook });
        }
    });
});
```

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
    console.log(req.body);
    var newBook = new Book(req.body);
    newBook.save( (err) => {
        if (err) {
            res.type('html').status(500);
            res.send('Error: ' + err);
        }
        else {
            res.render('created', { book: newBook });
        }
    } );
} );
```

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
    console.log(req.body);
    var newBook = new Book(req.body);
    newBook.save( (err) => {
        if (err) {
            res.type('html').status(500);
            res.send('Error: ' + err);
        }
        else {
            res.render('created', { book: newBook });
        }
    });
});
```

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
  console.log(req.body);
  var newBook = new Book(req.body);
  newBook.save( (err) => {
    if (err) {
      res.type('html').status(500);
      res.send('Error: ' + err);
    }
    else {
      res.render('created', { book: newBook });
    }
  } );
} );
```

```
{ title: 'JavaScript Programming',  
  year: '2017',  
  authors:  
    [ { name: 'Chris Murphy', affiliation: 'UPenn' },  
      { name: 'Swapneel Sheth', affiliation: 'UPenn' } ] }
```

```
{ title: 'JavaScript Programming',  
  year: '2017',  
  authors:  
    [ { name: 'Chris Murphy', affiliation: 'UPenn' },  
      { name: 'Swapneel Sheth', affiliation: 'UPenn' } ] }
```

```
var authorSchema = new Schema({  
  name: String,  
  affiliation: String  
});  
  
var bookSchema = new Schema({  
  title: {type: String, required: true, unique: true},  
  year: Number,  
  authors: [authorSchema]  
});
```

```
{ title: 'JavaScript Programming',  
  year: '2017',  
  authors:  
  [ { name: 'Chris Murphy', affiliation: 'UPenn' },  
    { name: 'Swapneel Sheth', affiliation: 'UPenn' } ] }
```

```
var authorSchema = new Schema({  
  name: String,  
  affiliation: String  
});  
  
var bookSchema = new Schema({  
  title: {type: String, required: true, unique: true},  
  year: Number,  
  authors: [authorSchema]  
});
```

```
{ title: 'JavaScript Programming',  
  year: '2017',  
authors:  
  [ { name: 'Chris Murphy', affiliation: 'UPenn' },  
    { name: 'Swapneel Sheth', affiliation: 'UPenn' } ] }
```

```
var authorSchema = new Schema({  
  name: String,  
  affiliation: String  
});  
  
var bookSchema = new Schema({  
  title: {type: String, required: true, unique: true},  
  year: Number,  
authors: [authorSchema]  
});
```

```
{ title: 'JavaScript Programming',  
  year: '2017',  
authors:  
  [ { name: 'Chris Murphy', affiliation: 'UPenn' },  
    { name: 'Swapneel Sheth', affiliation: 'UPenn' } ] }
```

```
var authorSchema = new Schema({  
  name: String,  
  affiliation: String  
});  
  
var bookSchema = new Schema({  
  title: {type: String, required: true, unique: true},  
  year: Number,  
authors: [authorSchema]  
});
```

```
{ title: 'JavaScript Programming',  
  year: '2017',  
  authors:  
    [ { name: 'Chris Murphy', affiliation: 'UPenn' },  
      { name: 'Swapneel Sheth', affiliation: 'UPenn' } ] }
```

```
var authorSchema = new Schema({  
  name: String,  
  affiliation: String  
});  
  
var bookSchema = new Schema({  
  title: {type: String, required: true, unique: true},  
  year: Number,  
  authors: [authorSchema]  
});
```

```
{ title: 'JavaScript Programming',  
year: '2017',  
authors:  
[ { name: 'Chris Murphy', affiliation: 'UPenn' },  
{ name: 'Swapneel Sheth', affiliation: 'UPenn' } ] }
```

```
var authorSchema = new Schema({  
  name: String,  
  affiliation: String  
});  
  
var bookSchema = new Schema({  
  title: {type: String, required: true, unique: true},  
  year: Number,  
  authors: [authorSchema]  
});
```

```
{ title: 'JavaScript Programming',  
year: '2017',  
authors:  
[ { name: 'Chris Murphy', affiliation: 'UPenn' },  
{ name: 'Swapneel Sheth', affiliation: 'UPenn' } ] }
```

```
var authorSchema = new Schema({  
    name: String,  
    affiliation: String  
});  
  
var bookSchema = new Schema({  
    title: {type: String, required: true, unique: true},  
    year: Number,  
    authors: [authorSchema]  
});
```

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
    console.log(req.body);
var newBook = new Book(req.body);
    newBook.save( (err) => {
        if (err) {
            res.type('html').status(500);
            res.send('Error: ' + err);
        }
        else {
            res.render('created', { book: newBook });
        }
    } );
} );
```

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
    console.log(req.body);
    var newBook = new Book(req.body);
    newBook.save( (err) => {
        if (err) {
            res.type('html').status(500);
            res.send('Error: ' + err);
        }
        else {
            res.render('created', { book: newBook });
        }
    });
});
```

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
    console.log(req.body);
    var newBook = new Book(req.body);
    newBook.save( (err) => {
        if (err) {
            res.type('html').status(500);
            res.send('Error: ' + err);
        }
        else {
            res.render('created', { book: newBook });
        }
    } );
} );
```

```
var express = require('express');
var app = express();
app.set('view engine', 'ejs');
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

app.use('/createbook', (req, res) => {
    console.log(req.body);
    var newBook = new Book(req.body);
    newBook.save( (err) => {
        if (err) {
            res.type('html').status(500);
            res.send('Error: ' + err);
        }
        else {
            res.render('created', { book: newBook });
        }
    } );
} );
```



Indicate the search criteria:

Title:

Author name:

Year:

All

Any

Search

Indicate the search criteria:

```
<form action='/search' method='post'>
```

Title: <input name='title'>

```
<p>
```

Author name: <input name='name'>

```
<p>
```

Year: <input name='year'>

```
<p>
```

```
<input type='radio' name='which' value='all' checked>All
```

```
<br>
```

```
<input type='radio' name='which' value='any'>Any
```

```
<p>
```

```
<input type='submit' value='Search'>
```

```
</form>
```

Indicate the search criteria:

<form action='/search' method='post'>

Title: <input name='title'>

<p>

Author name: <input name='name'>

<p>

Year: <input name='year'>

<p>

<input type='radio' name='which' value='all' checked>All

<input type='radio' name='which' value='any'>Any

<p>

<input type='submit' value='Search'>

</form>

Indicate the search criteria:

```
<form action='/search' method='post'>
```

Title: <input name='title'>

```
<p>
```

Author name: <input name='name'>

```
<p>
```

Year: <input name='year'>

```
<p>
```

```
<input type='radio' name='which' value='all' checked>All
```

```
<br>
```

```
<input type='radio' name='which' value='any'>Any
```

```
<p>
```

```
<input type='submit' value='Search'>
```

```
</form>
```

Indicate the search criteria:

```
<form action='/search' method='post'>
```

Title: <input name='title'>

```
<p>
```

Author name: <input name='name'>

```
<p>
```

Year: <input name='year'>

```
<p>
```

<input type='radio' name='which' value='all' checked>All

```
<br>
```

<input type='radio' name='which' value='any'>Any

```
<p>
```

<input type='submit' value='Search'>

```
</form>
```

Indicate the search criteria:

```
<form action='/search' method='post'>
```

Title: <input name='title'>

```
<p>
```

Author name: <input name='name'>

```
<p>
```

Year: <input name='year'>

```
<p>
```

```
<input type='radio' name='which' value='all' checked>All
```

```
<br>
```

```
<input type='radio' name='which' value='any'>Any
```

```
<p>
```

```
<input type='submit' value='Search'>
```

```
</form>
```

Indicate the search criteria:

```
<form action='/search' method='post'>
```

Title: <input name='title'>

```
<p>
```

Author name: <input name='name'>

```
<p>
```

Year: <input name='year'>

```
<p>
```

```
<input type='radio' name='which' value='all' checked>All
```

```
<br>
```

```
<input type='radio' name='which' value='any'>Any
```

```
<p>
```

<input type='submit' value='Search'>

```
</form>
```



Indicate the search criteria:

Title:

Author name:

Year:

All

Any

Search

```
app.use('/search', (req, res) => {  
    if (req.body.which == 'all') {  
        searchAll(req, res);  
    }  
    else if (req.body.which == 'any') {  
        searchAny(req, res);  
    }  
    else {  
        searchAll(req, res);  
    }  
} );
```

```
app.use('/search', (req, res) => {  
  if (req.body.which == 'all') {  
    searchAll(req, res);  
  }  
  else if (req.body.which == 'any') {  
    searchAny(req, res);  
  }  
  else {  
    searchAll(req, res);  
  }  
}) ;
```

```
app.use('/search', (req, res) => {  
  if (req.body.which == 'all') {  
    searchAll(req, res);  
  }  
  else if (req.body.which == 'any') {  
    searchAny(req, res);  
  }  
  else {  
    searchAll(req, res);  
  }  
} );
```

```
function searchAll(req, res) {  
  
var query = {};  
  
if (req.body.title) query.title = req.body.title;  
if (req.body.year) query.year = req.body.year;  
if (req.body.name) {  
    query['authors.name'] = req.body.name;  
}  
  
console.log(query);  
  
Book.find( query, (err, books) => {  
    if (err) {  
        res.type('html').status(500);  
        res.send('Error: ' + err);  
    }  
    else {  
        res.render('books', { books: books });  
    }  
}) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  // { 'authors.name': 'Chris Murphy', year: '2017' }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  // { 'authors.name': 'Chris Murphy', year: '2017' }  
  
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
});  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  // { 'authors.name': 'Chris Murphy', year: '2017' }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  // { 'authors.name': 'Chris Murphy', year: '2017' }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```



localhost:3000/search

x

Guest



localhost:3000/search



Here are the results of your search:

- *JavaScript Programming*. Chris Murphy, Swapneel Sheth, 2017.
- *Data Structures*. Chris Murphy, 2017.

```
<!-- This is views/books.ejs -->
```

```
Here are the results of your search:
```

```
<ul>
```

```
<% books.forEach( (book) => { %>
```

```
    <li>
```

```
        <i><%= book.title %></i>.
```

```
        <% book.authors.forEach( (author) => { %>
```

```
            <%= author.name %>,
```

```
        <% } ) ; %>
```

```
        <%= book.year %>.
```

```
    </li>
```

```
<% } ) ; %>
```

```
</ul>
```



<!-- This is views/books.ejs -->

Here are the results of your search:

<% books.forEach((book) => { %>

<i><%= book.title %></i>.

<% book.authors.forEach((author) => { %>

<%= author.name %>,

<% }) ; %>

<%= book.year %>.

<% }) ; %>



```
<!-- This is views/books.ejs -->
```

```
Here are the results of your search:
```

```
<ul>
```

```
<% books.forEach( (book) => { %>
```

```
    <li>
```

```
        <i><%= book.title %></i>.
```

```
        <% book.authors.forEach( (author) => { %>
```

```
            <%= author.name %>,
```

```
        <% } ) ; %>
```

```
        <%= book.year %>.
```

```
    </li>
```

```
<% } ) ; %>
```

```
</ul>
```



```
<!-- This is views/books.ejs -->
```

Here are the results of your search:

```
<ul>
```

```
<% books.forEach( book) => { %>
```

```
    <li>
```

```
        <i><%= book.title %></i>.
```

```
        <% book.authors.forEach( author) => { %>
```

```
            <%= author.name %>,
```

```
        <% } ); %>
```

```
        <%= book.year %>.
```

```
    </li>
```

```
<% } ); %>
```

```
</ul>
```



```
<!-- This is views/books.ejs -->
```

```
Here are the results of your search:
```

```
<ul>
```

```
<% books.forEach( book) => { %>
```

```
<li>
```

```
<i><%= book.title %></i>.
```

```
<% book.authors.forEach( author) => { %>
```

```
<%= author.name %>,
```

```
<% } ) ; %>
```

```
<%= book.year %>.
```

```
</li>
```

```
<% } ) ; %>
```

```
</ul>
```



```
<!-- This is views/books.ejs -->
```

```
Here are the results of your search:
```

```
<ul>
```

```
<% books.forEach( book) => { %>
```

```
<li>
```

```
<i><%= book.title %></i>.
```

```
<% book.authors.forEach( author) => { %>
```

```
<%= author.name %>,
```

```
<% } ); %>
```

```
<%= book.year %>.
```

```
</li>
```

```
<% } ); %>
```

```
</ul>
```



```
<!-- This is views/books.ejs -->
```

```
Here are the results of your search:
```

```
<ul>
```

```
<% books.forEach( book) => { %>
```

```
<li>
```

```
<i><%= book.title %></i>.
```

```
<% book.authors.forEach( author) => { %>
```

```
<%= author.name %>,
```

```
<% } ); %>
```

```
<%= book.year %>.
```

```
</li>
```

```
<% } ); %>
```

```
</ul>
```



```
<!-- This is views/books.ejs -->
```

```
Here are the results of your search:
```

```
<ul>
```

```
<% books.forEach( (book) => { %>
```

```
    <li>
```

```
        <i><%= book.title %></i>.
```

```
        <% book.authors.forEach( (author) => { %>
```

```
            <%= author.name %>,
```

```
        <% } ) ; %>
```

```
        <%= book.year %>.
```

```
    </li>
```

```
<% } ) ; %>
```

```
</ul>
```



```
<!-- This is views/books.ejs -->
```

```
Here are the results of your search:
```

```
<ul>
```

```
<% books.forEach( (book) => { %>
```

```
    <li>
```

```
        <i><%= book.title %></i>.
```

```
        <% book.authors.forEach( (author) => { %>
```

```
            <%= author.name %>,
```

```
        <% } ) ; %>
```

```
        <%= book.year %>.
```

```
    </li>
```

```
<% } ) ; %>
```

```
</ul>
```





Indicate the search criteria:

Title:

Author name:

Year:

- All
 Any

Search

```
app.use('/search', (req, res) => {  
  if (req.body.which == 'all') {  
    searchAll(req, res);  
  }  
  else if (req.body.which == 'any') {  
    searchAny(req, res);  
  }  
  else {  
    searchAll(req, res);  
  }  
}) ;
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);  
  
  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
if (req.body.year)  
  terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
if (req.body.name)  
  terms.push( { 'authors.name' : req.body.name } );  
  
var query = { $or : terms };  
console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
var query = { $or : terms };  
console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);  
  // { '$or': [ {title: 'Programming'}, {year: '2017'} ] }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year)  
    terms.push( { year: req.body.year } );  
  if (req.body.name)  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);
```

```
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
});  
}
```



Here are the results of your search:

- *JavaScript Programming*. Chris Murphy, Swapneel Sheth, 2017.
- *Data Structures*. Chris Murphy, 2017.
- *Intro to Java Programming*. Arvind Bhusnurmath, 2017.

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year) {  
    terms.push( { year: req.body.year } );  
  if (req.body.name) {  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);  
  // { '$or': [ {title: 'Programming'}, {year: '2017'} ] }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books } );  
    }  
  } );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year) {  
    terms.push( { year: req.body.year } );  
  if (req.body.name) {  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);  
  // { '$or': [ {title: 'Programming'}, {year: '2017'} ] }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  } );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: req.body.title } );  
  if (req.body.year) {  
    terms.push( { year: req.body.year } );  
  if (req.body.name) {  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);  
  // { '$or': [ {title: 'Programming'}, {year: '2017'} ] }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books } );  
    }  
  } );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: { $regex: req.body.title } } );  
  if (req.body.year) {  
    terms.push( { year: req.body.year } );  
  if (req.body.name) {  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);  
  // { '$or': [ {title: 'Programming'}, {year: '2017'} ] }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  } );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: { $regex: req.body.title } } );  
  if (req.body.year) {  
    terms.push( { year: req.body.year } );  
  if (req.body.name) {  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);  
  // { '$or': [ {title: [Object]}, {year: '2017'} ] }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books } );  
    }  
  } );  
}
```



localhost:3000/search

x

Guest



localhost:3000/search



Here are the results of your search:

- *JavaScript Programming*. Chris Murphy, Swapneel Sheth, 2017.
- *Data Structures*. Chris Murphy, 2017.
- *Intro to Java Programming*. Arvind Bhusnurmath, 2017.
- *The Art of Computer Programming*. Donald Knuth, 1968.

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: { $regex: req.body.title } } );  
  if (req.body.year) {  
    terms.push( { year: req.body.year } );  
  if (req.body.name) {  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);  
  // { '$or': [ {title: [Object]}, {year: '2017'} ] }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  } );  
}
```

```
function searchAny(req, res) {  
  
  var terms = [];  
  if (req.body.title)  
    terms.push( { title: { $regex: req.body.title } } );  
  if (req.body.year) {  
    terms.push( { year: req.body.year } );  
  if (req.body.name) {  
    terms.push( { 'authors.name' : req.body.name } );  
  
  var query = { $or : terms };  
  console.log(query);  
  // { '$or': [ {title: [Object]}, {year: '2017'} ] }  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) .sort( { 'title' : 'asc' } );
```



localhost:3000/search



Guest



localhost:3000/search



Here are the results of your search:

- *Data Structures*. Chris Murphy, 2017.
- *Intro to Java Programming*. Arvind Bhusnurmath, 2017.
- *JavaScript Programming*. Chris Murphy, Swapneel Sheth, 2017.
- *The Art of Computer Programming*. Donald Knuth, 1968.

Summary

- MongoDB allows us to have a Schema in which one document contains other documents
- We can then do queries for documents using the properties of the documents they contain
- We can also do “all” and “any” queries by passing objects to the **find** function
- And sort the results using **sort**



Video 4.9

Creating an API

Chris Murphy

Review

- Node.js and Express allow us to build server-side web apps in JavaScript
- MongoDB allows us to store data as documents and query them via JavaScript



Indicate the search criteria:

Title:

Author name:

Year:

All

Any

Search

```
function searchAll(req, res) {  
  
var query = {};  
  
if (req.body.title) query.title = req.body.title;  
if (req.body.year) query.year = req.body.year;  
if (req.body.name) {  
    query['authors.name'] = req.body.name;  
}  
  
console.log(query);  
  
Book.find( query, (err, books) => {  
    if (err) {  
        res.type('html').status(500);  
        res.send('Error: ' + err);  
    }  
    else {  
        res.render('books', { books: books });  
    }  
}) ;  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  
  
Book.find( query, (err, books) => {  
  if (err) {  
    res.type('html').status(500);  
    res.send('Error: ' + err);  
  }  
  else {  
    res.render('books', { books: books });  
  }  
} );  
}
```

```
function searchAll(req, res) {  
  
  var query = {};  
  
  if (req.body.title) query.title = req.body.title;  
  if (req.body.year) query.year = req.body.year;  
  if (req.body.name) {  
    query['authors.name'] = req.body.name;  
  }  
  
  console.log(query);  
  
  Book.find( query, (err, books) => {  
    if (err) {  
      res.type('html').status(500);  
      res.send('Error: ' + err);  
    }  
    else {  
      res.render('books', { books: books });  
    }  
  }) ;  
}
```



localhost:3000/search

x

Guest



localhost:3000/search



Here are the results of your search:

- *JavaScript Programming*. Chris Murphy, Swapneel Sheth, 2017.
- *Data Structures*. Chris Murphy, 2017.



Indicate the search criteria:

Title:

Author name:

Year:

- All
- Any



Indicate the search criteria:

Title:

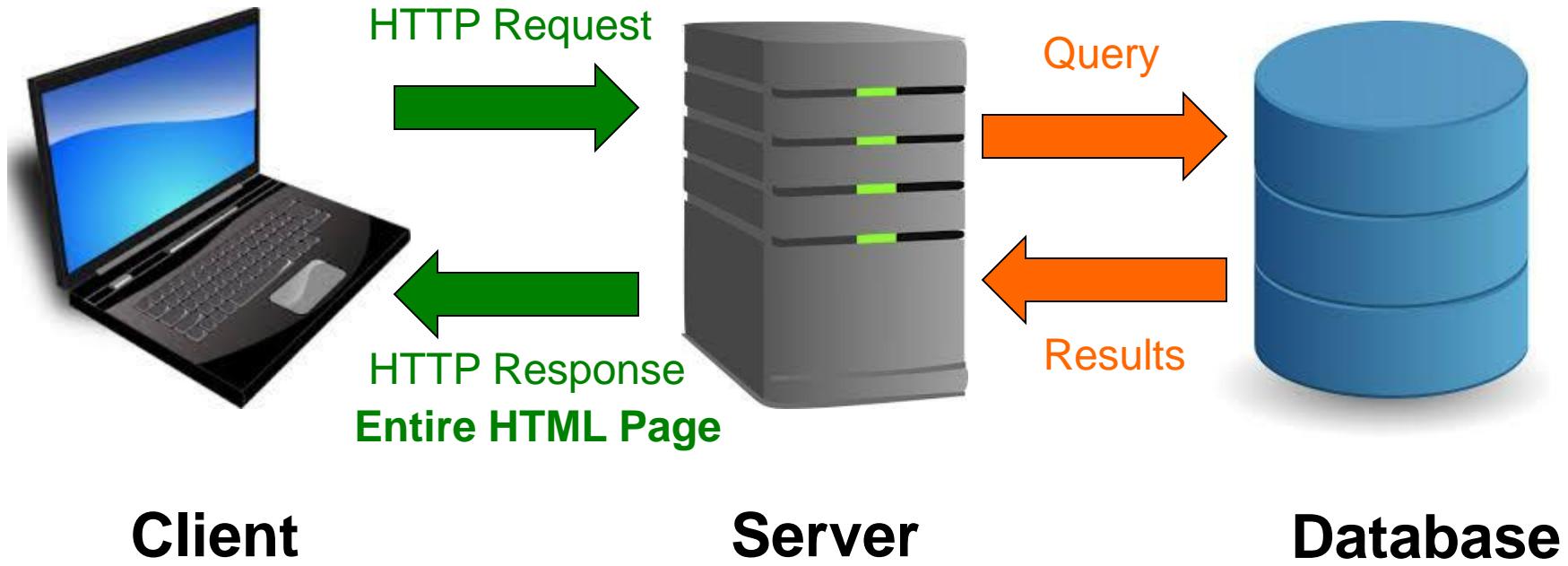
Author name:

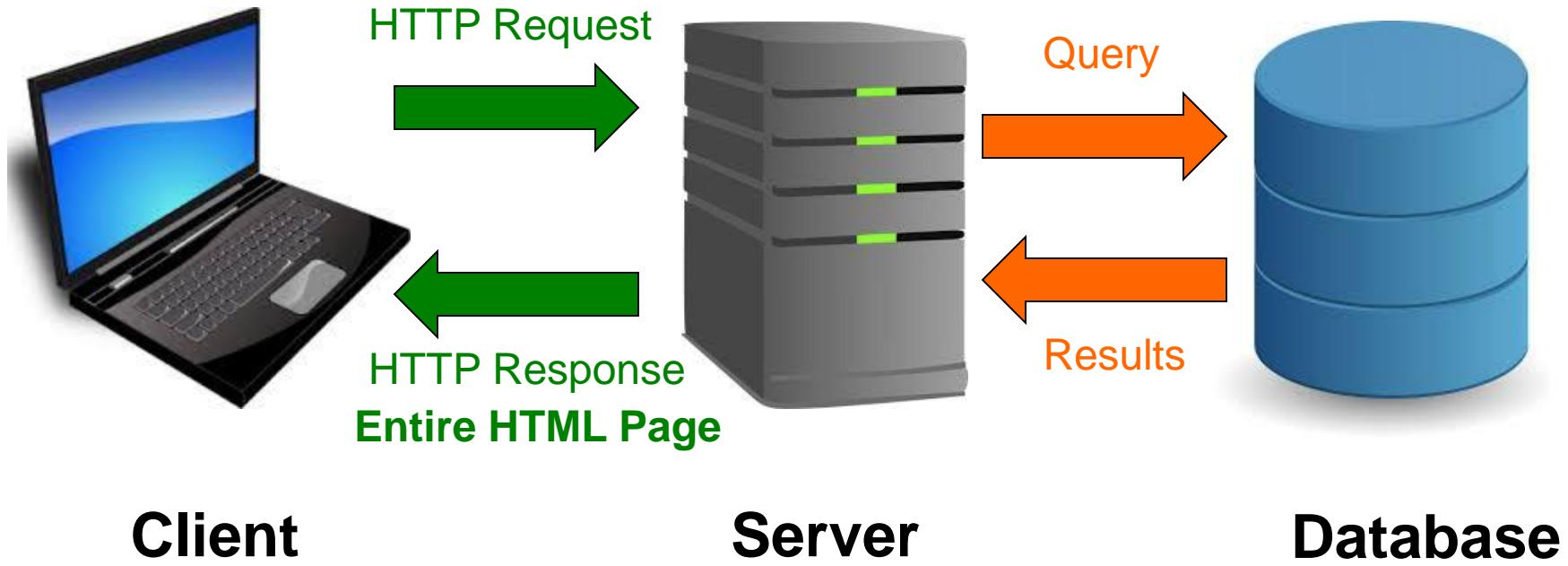
Year:

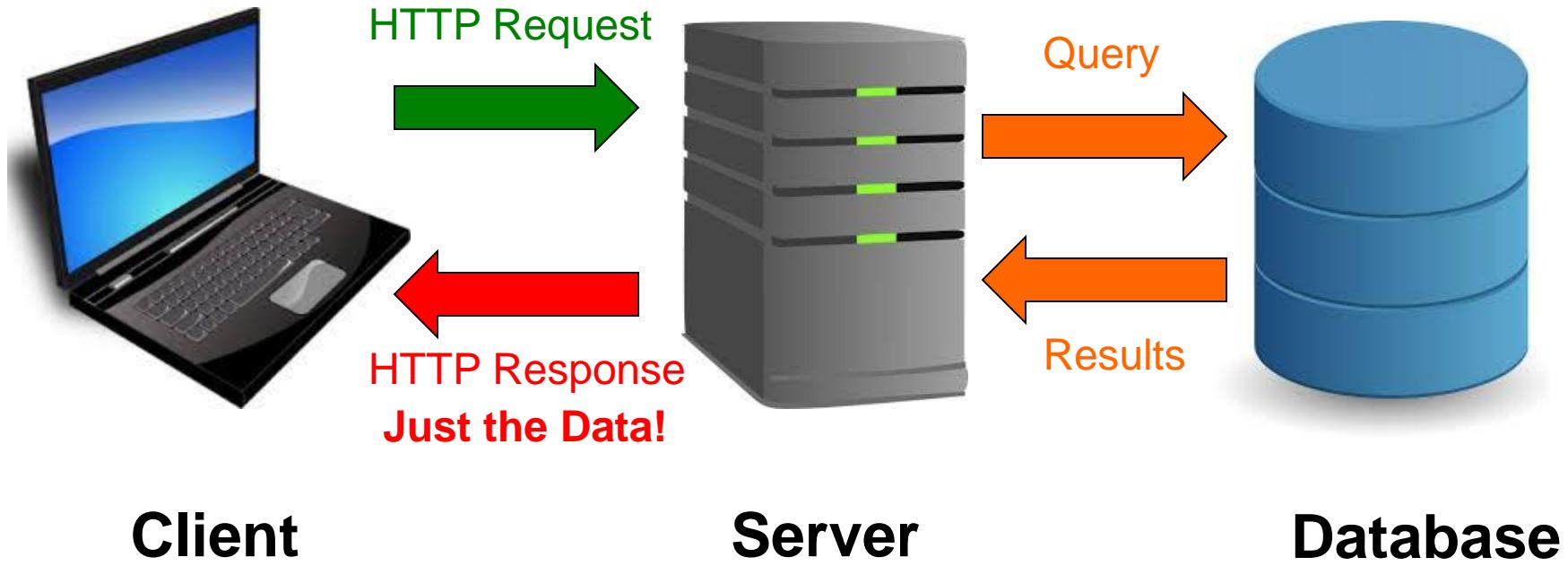
All

Any

Search







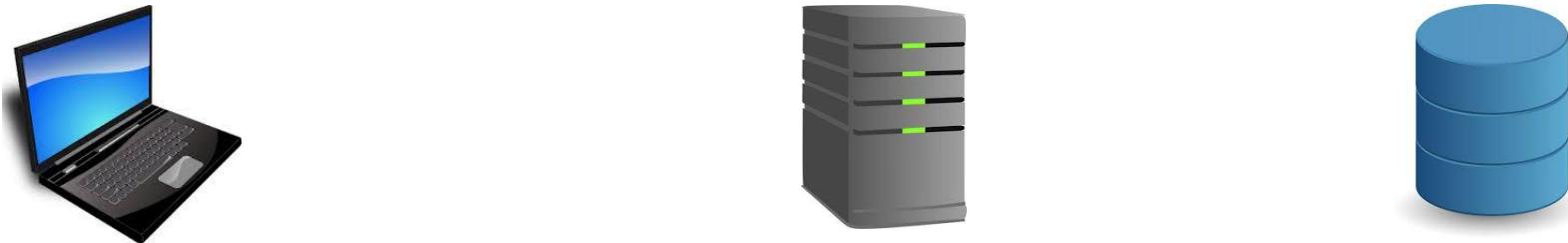
What is an API?

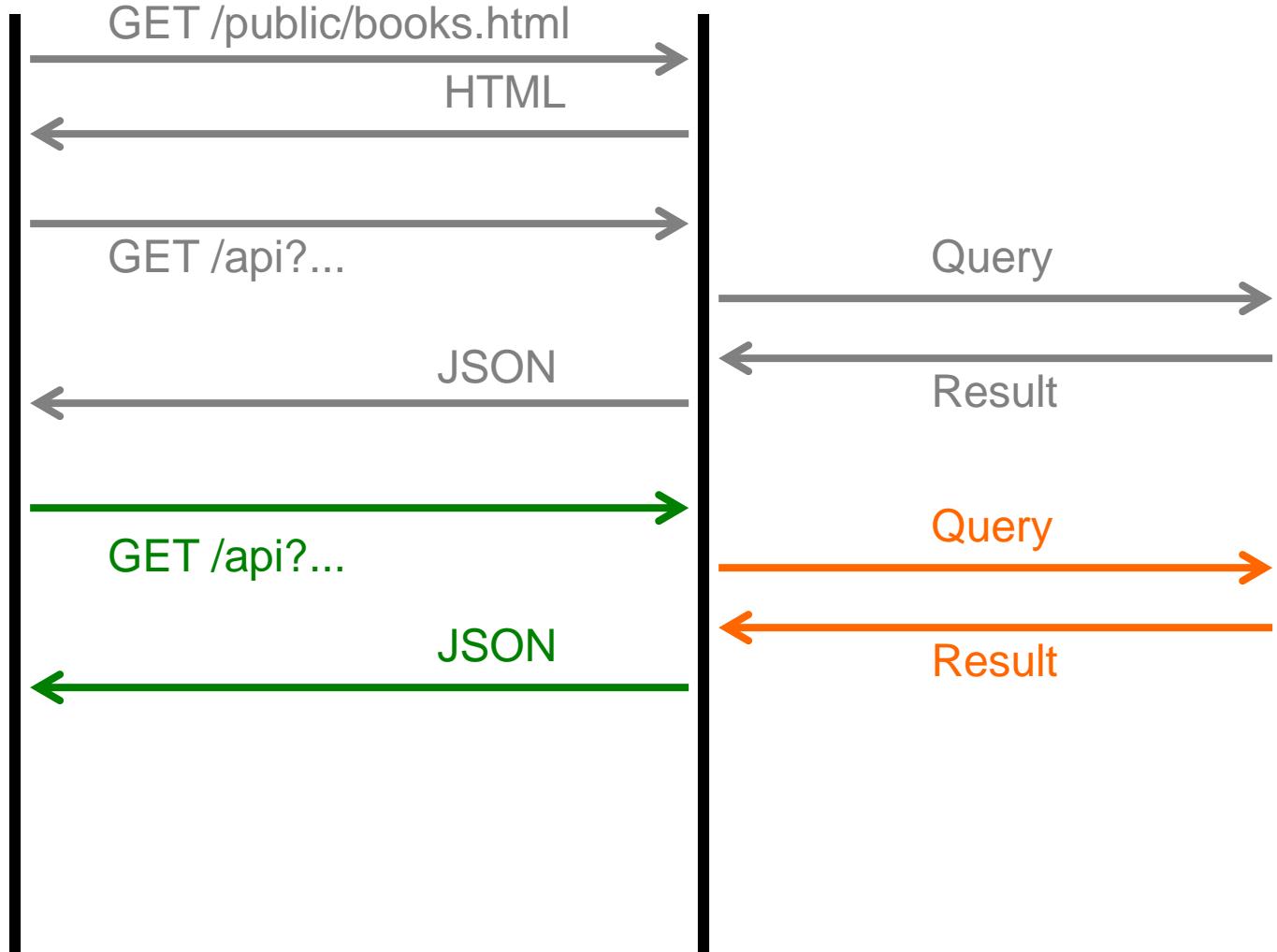
- An API is an **Application Programming Interface**
- It is a URL or a set of URLs that returns pure data to requests
- APIs can be used to incorporate data and functionality from other sources in your webapp

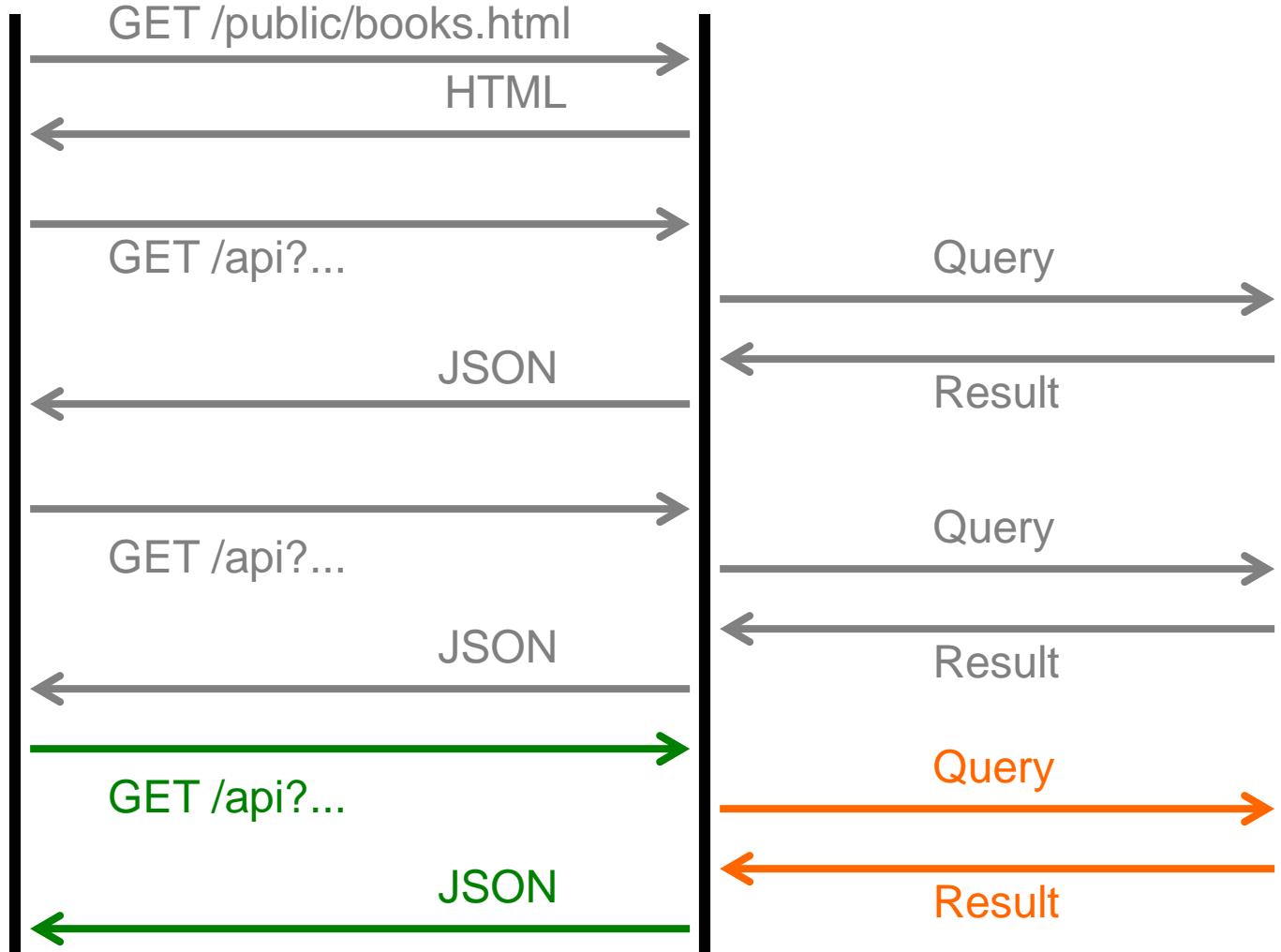
What is an API?

- An API is an **Application Programming Interface**
- It is a URL or a set of URLs that returns pure data to requests
- APIs can be used to incorporate data and functionality from other sources in your webapp
- **You can also create your own API using Node.js to return JSON data to HTTP requests**





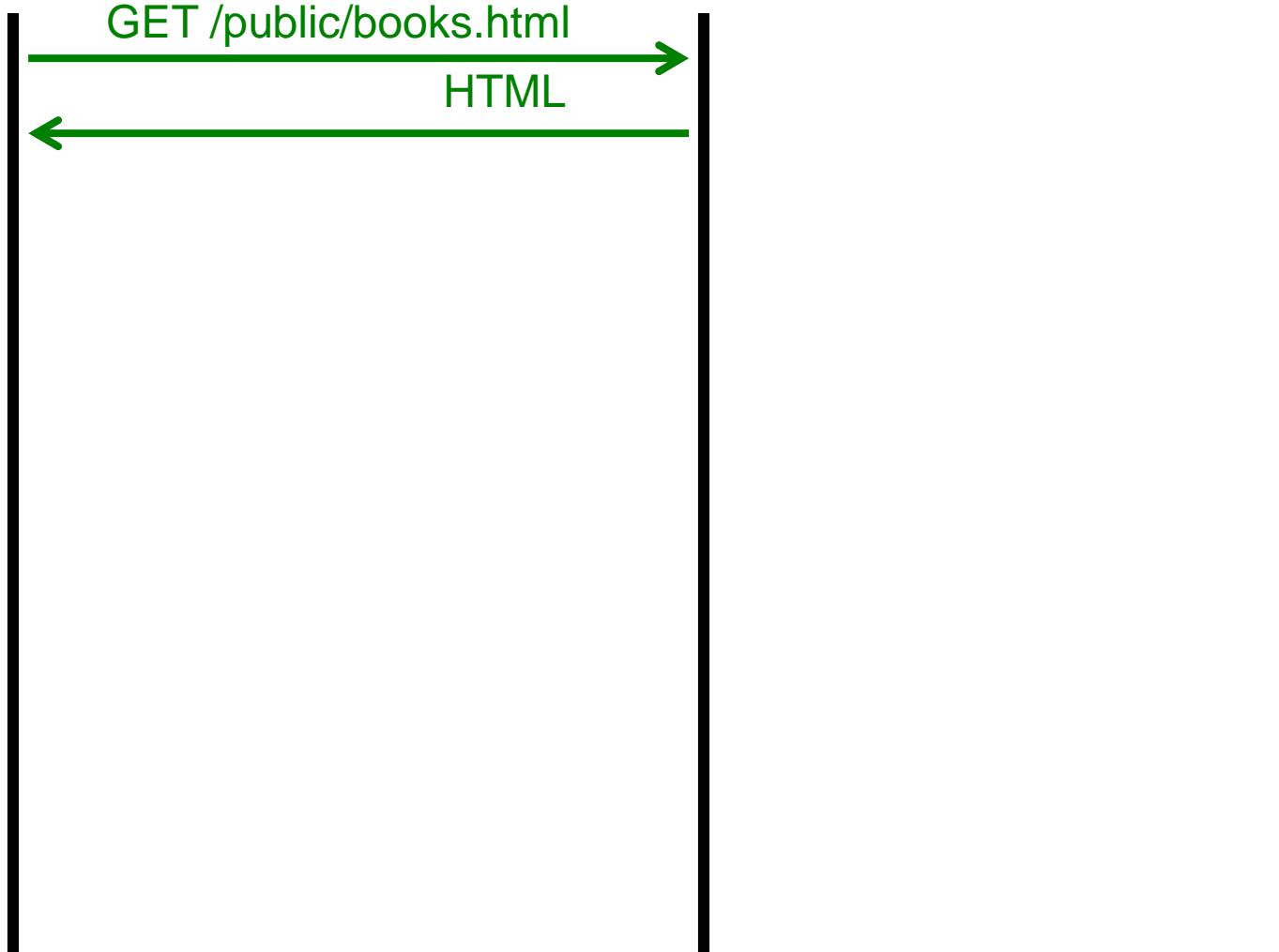




← → C

localhost:3000/public/books.html

**Title:** **Author:** **Year:**



← → C

localhost:3000/public/books.html



Title:

Author:

Year:

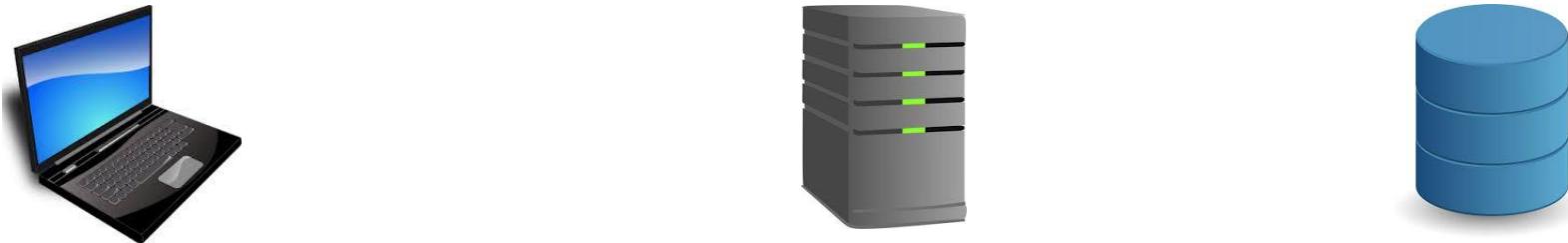


Title:

Author:

Year:

- *Intro to Java Programming*, Arvind Bhusnurmeh, 2017
- *JavaScript Programming*, Chris Murphy, Swapneel Sheth, 2017
- *The Art of Computer Programming*, Donald Knuth, 1968





Title:

Author:

Year:

- *Intro to Java Programming*, Arvind Bhusnurmeh, 2017
- *JavaScript Programming*, Chris Murphy, Swapneel Sheth, 2017
- *The Art of Computer Programming*, Donald Knuth, 1968

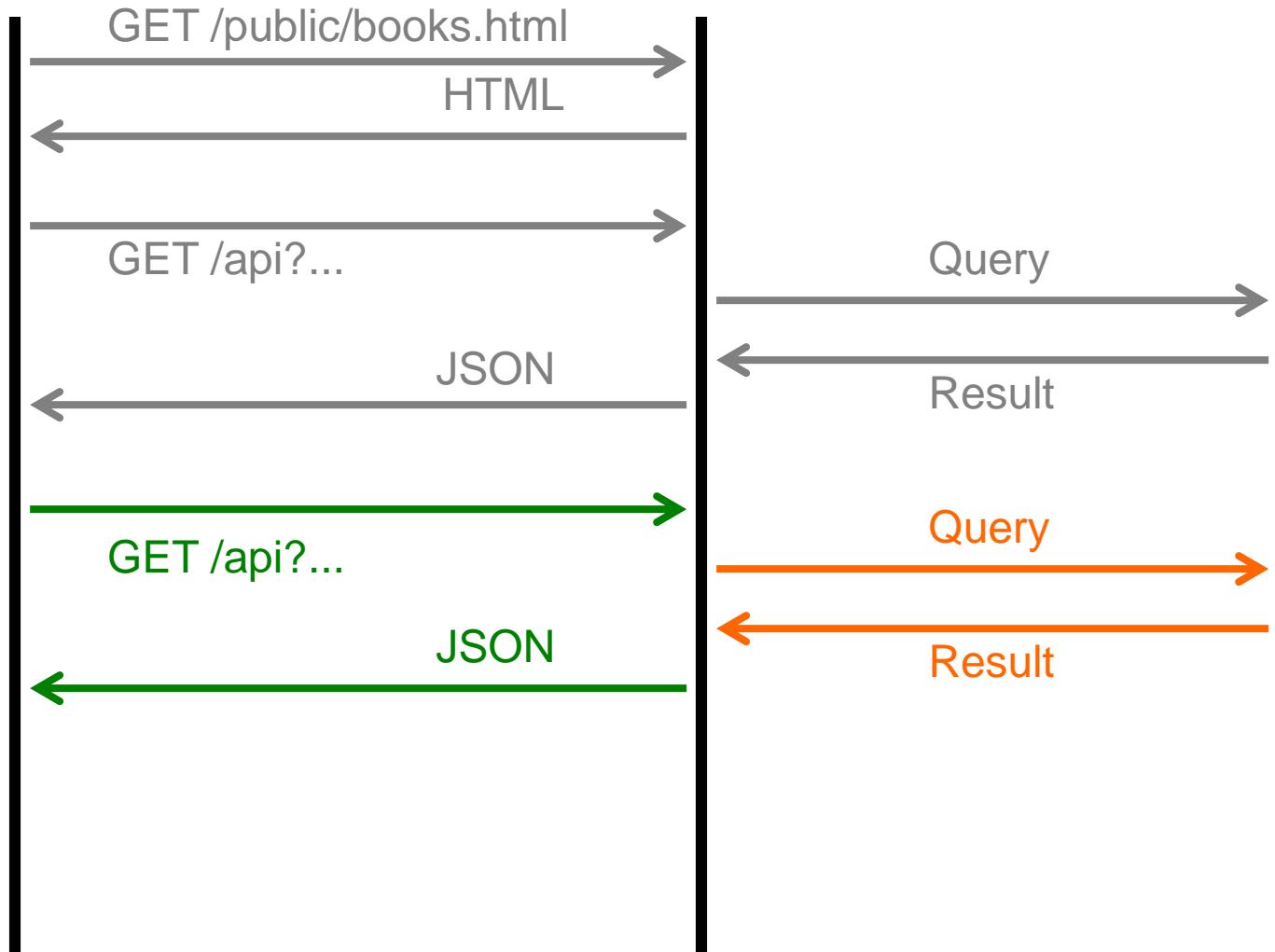
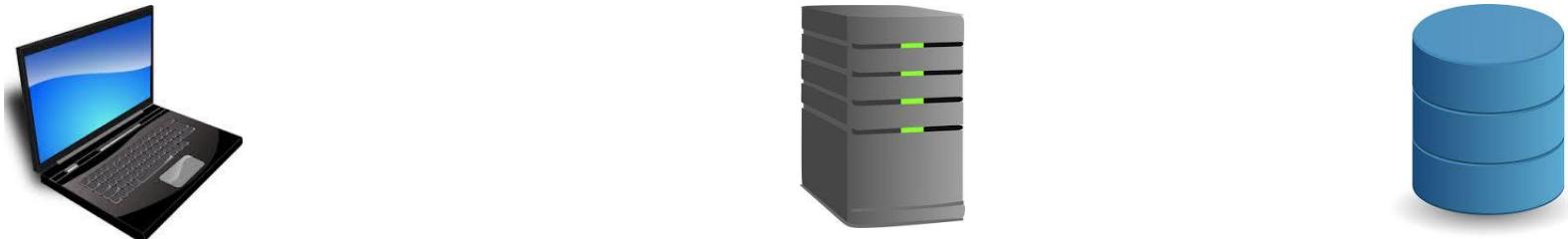


Title:

Author:

Year:

- *Intro to Java Programming*, Arvind Bhusnurmeh, 2017
- *JavaScript Programming*, Chris Murphy, Swapneel Sheth, 2017
- *The Art of Computer Programming*, Donald Knuth, 1968



← → C

localhost:3000/public/books.html

Title: Author: Year:

- *Intro to Java Programming*, Arvind Bhusnurmeh, 2017
- *JavaScript Programming*, Chris Murphy, Swapneel Sheth, 2017
- *The Art of Computer Programming*, Donald Knuth, 1968



Title:

Author:

Year:

- *Intro to Java Programming*, Arvind Bhusnurmeh, 2017
- *JavaScript Programming*, Chris Murphy, Swapneel Sheth, 2017
- *The Art of Computer Programming*, Donald Knuth, 1968



Title:

Author:

Year:

- *Intro to Java Programming*, Arvind Bhusnurmeh, 2017
- *JavaScript Programming*, Chris Murphy, Swapneel Sheth, 2017
- *The Art of Computer Programming*, Donald Knuth, 1968

← → C

localhost:3000/public/books.html



Title:

Author:

Year:



Title:

Author:

Year:

- *Intro to Java Programming*, Arvind Bhusnurmeh, 2017
- *JavaScript Programming*, Chris Murphy, Swapneel Sheth, 2017

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

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

. . .
```

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

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

. . .
```

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

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

. . .
```

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

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

. . .
```

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

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

. . .
```

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

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Book = require('./Book.js');

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

. . .
```

```
app.use( '/api', (req, res) => {

  var query = {};
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    });
  }
  else res.json({}); // empty query
}) ;
```

```
app.use( '/api', (req, res) => {

  var query = {};
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    });
  }
  else res.json({}); // empty query
}) ;
```

```
app.use( '/api', (req, res) => {

  var query = {};
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    });
  }
  else res.json({}); // empty query
}) ;
```

```
app.use( '/api', (req, res) => {

  var query = { };
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    });
  }
  else res.json({}); // empty query
}) ;
```

```
app.use( '/api', (req, res) => {

  var query = {};
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    });
  }
  else res.json({}); // empty query
}) ;
```

```
app.use( '/api', (req, res) => {

  var query = {};
  if (req.query.title)
    query.title = { $regex : req.query.title };
if (req.query.name)
  query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    }) ;
  }
  else res.json({}); // empty query
}) ;
```

```
app.use( '/api', (req, res) => {

  var query = { };
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
if (req.query.year)
  query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    }) ;
  }
  else res.json({ }) ; // empty query
}) ;
```

```
app.use( '/api', (req, res) => {

  var query = {};
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

if (Object.keys(query).length != 0) {
  Book.find( query, (err, books) => {
    if (!err)
      res.json(books);
    else {
      console.log(err)
      res.json({ });
    }
  ) );
}

else res.json({ }); // empty query
} ) ;
```

```
app.use( '/api', (req, res) => {

  var query = { };
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    });
  }
  else res.json({ }); // empty query
}) );
```

```
app.use( '/api', (req, res) => {

  var query = {};
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    });
  }
  else res.json({}); // empty query
}) ;
```

```
app.use( '/api', (req, res) => {

  var query = {};
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    });
  }
  else res.json({}); // empty query
}) ;
```

```
[ { _id: 5967e069359c635bb106d662,
  title: 'Intro to Java Programming',
  year: 2017,
  __v: 0,
  authors: [ [Object] ] },
{ _id: 5967d3b75d1bcf5afa49ca3a,
  title: 'JavaScript Programming',
  year: 2017,
  __v: 0,
  authors: [ [Object], [Object] ] },
{ _id: 5967e095359c635bb106d665,
  title: 'The Art of Computer Programming',
  year: 1968,
  __v: 0,
  authors: [ [Object] ] }
```

```
[ { _id: 5967e069359c635bb106d662,
  title: 'Intro to Java Programming',
  year: 2017,
  __v: 0,
  authors: [ [Object] ] },
{ _id: 5967d3b75d1bcf5afa49ca3a,
  title: 'JavaScript Programming',
  year: 2017,
  __v: 0,
  authors: [ [Object], [Object] ] },
{ _id: 5967e095359c635bb106d665,
  title: 'The Art of Computer Programming',
  year: 1968,
  __v: 0,
  authors: [ [Object] ] }
```

```
[ { _id: 5967e069359c635bb106d662,
  title: 'Intro to Java Programming',
  year: 2017,
  __v: 0,
  authors: [ [Object] ] },
{ _id: 5967d3b75d1bcf5afa49ca3a,
  title: 'JavaScript Programming',
  year: 2017,
  __v: 0,
  authors: [ [Object], [Object] ] },
{ _id: 5967e095359c635bb106d665,
  title: 'The Art of Computer Programming',
  year: 1968,
  __v: 0,
  authors: [ [Object] ] }
```

```
app.use( '/api', (req, res) => {

  var query = {};
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    });
  }
  else res.json({}); // empty query
}) ;
```

```
app.use( '/api', (req, res) => {

  var query = { };
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    }) ;
  }
  else res.json({ }) ; // empty query
}) ;
```

```
app.use( '/api', (req, res) => {

  var query = {};
  if (req.query.title)
    query.title = { $regex : req.query.title };
  if (req.query.name)
    query['authors.name'] = { $regex: req.query.name };
  if (req.query.year)
    query.year = req.query.year;

  if (Object.keys(query).length != 0) {
    Book.find( query, (err, books) => {
      if (!err)
        res.json(books);
      else {
        console.log(err)
        res.json({ });
      }
    }) ;
  }
  else res.json({}); // empty query
}) ;
```

← → C

localhost:3000/public/books.html



Title:

Author:

Year:



Title:

Author:

Year:

- *Intro to Java Programming*, Arvind Bhusnurmeh, 2017
- *JavaScript Programming*, Chris Murphy, Swapneel Sheth, 2017
- *The Art of Computer Programming*, Donald Knuth, 1968

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Book Finder</title>
    <script src="jquery.js"></script>
  </head>

  <body>

    <form>
      Title: <input name='title'><br>
      Author: <input name='author'><br>
      Year: <input name='year'><br>
    </form>

    <ul>
      <div id='results'></div>
    </ul>

    <script>
      // next slide -->
    </script>

  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Book Finder</title>
    <script src="jquery.js"></script>
  </head>

  <body>

    <form>
      Title: <input name='title'><br>
      Author: <input name='author'><br>
      Year: <input name='year'><br>
    </form>

    <ul>
      <div id='results'></div>
    </ul>

    <script>
      // next slide -->
    </script>

  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Book Finder</title>
    <script src="jquery.js"></script>
  </head>

  <body>

    <b><form></b>
      Title: <input name='title'><br>
      Author: <input name='author'><br>
      Year: <input name='year'><br>
    </b></form>

    <ul>
      <div id='results'></div>
    </ul>

    <script>
      // next slide -->
    </script>

  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Book Finder</title>
    <script src="jquery.js"></script>
  </head>

  <body>

    <form>
      Title: <input name='title'><br>
      Author: <input name='author'><br>
      Year: <input name='year'><br>
    </form>

    <ul>
      <div id='results'></div>
    </ul>

    <script>
      // next slide -->
    </script>

  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Book Finder</title>
    <script src="jquery.js"></script>
  </head>

  <body>

    <form>
      Title: <input name='title'><br>
      Author: <input name='author'><br>
      Year: <input name='year'><br>
    </form>

    <ul>
      <div id='results'></div>
    </ul>

    <script>
      // next slide -->
    </script>

  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Book Finder</title>
    <script src="jquery.js"></script>
  </head>

  <body>

    <form>
      Title: <input name='title'><br>
      Author: <input name='author'><br>
      Year: <input name='year'><br>
    </form>

    <ul>
      <div id='results'></div>
    </ul>

    <script>
      // next slide -->
    </script>

  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Book Finder</title>
    <script src="jquery.js"></script>
  </head>

  <body>

    <form>
      Title: <input name='title'><br>
      Author: <input name='author'><br>
      Year: <input name='year'><br>
    </form>

    <ul>
      <div id='results'></div>
    </ul>

    <script>
      // next slide -->
    </script>

  </body>
</html>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on("change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
if (author) query += 'name=' + author + '&;
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Book Finder</title>
    <script src="jquery.js"></script>
  </head>

  <body>

    <form>
      Title: <input name='title'><br>
      Author: <input name='author'><br>
      Year: <input name='year'><br>
    </form>

    <ul>
      <div id='results'></div>
    </ul>

    <script>
      // next slide -->
    </script>

  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Book Finder</title>
    <script src="jquery.js"></script>
  </head>

  <body>

    <form>
      Title: <input name='title'><br>
      Author: <input name='author'><br>
      Year: <input name='year'><br>
    </form>

    <ul>
      <div id='results'></div>
    </ul>

    <script>
      // next slide -->
    </script>

  </body>
</html>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
[ { _id: 5967e069359c635bb106d662,
  title: 'Intro to Java Programming',
  year: 2017,
  __v: 0,
  authors: [ [Object] ] },
{ _id: 5967d3b75d1bcf5afa49ca3a,
  title: 'JavaScript Programming',
  year: 2017,
  __v: 0,
  authors: [ [Object], [Object] ] },
{ _id: 5967e095359c635bb106d665,
  title: 'The Art of Computer Programming',
  year: 1968,
  __v: 0,
  authors: [ [Object] ] }
```

```
[ { _id: 5967e069359c635bb106d662,
  title: 'Intro to Java Programming',
  year: 2017,
  __v: 0,
  authors: [ [Object] ] },
{ _id: 5967d3b75d1bcf5afa49ca3a,
  title: 'JavaScript Programming',
  year: 2017,
  __v: 0,
  authors: [ [Object], [Object] ] },
{ _id: 5967e095359c635bb106d665,
  title: 'The Art of Computer Programming',
  year: 1968,
  __v: 0,
  authors: [ [Object] ] }
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```

```
<script>
  $("input").on( "change input textInput", () => {
    var title = $("input[name='title']").val();
    var author = $("input[name='author']").val();
    var year = $("input[name='year']").val();

    var query = '?';
    if (title) query += 'title=' + title + '&';
    if (author) query += 'name=' + author + '&';
    if (year) query += 'year=' + year;
    var url = 'http://localhost:3000/api/' + query;

    $.getJSON(url, (books, status) => {
      var results = $("#results");
      results.html('');
      books.forEach( (book) => {
        results.append('<li><i>' + book.title + '</i>, ');
        book.authors.forEach( (author) => {
          if (author.name) results.append(author.name + ', ');
        });
        results.append(book.year + '</li>');
      });
    });
  });
</script>
```



Title:

Author:

Year:

- *Intro to Java Programming*, Arvind Bhusnurmeh, 2017
- *JavaScript Programming*, Chris Murphy, Swapneel Sheth, 2017
- *The Art of Computer Programming*, Donald Knuth, 1968

Summary

- We can use **server-side** tools such as Node.js, Express, and MongoDB to develop APIs to make data available on the Web
- We can use **client-side** tools such as jQuery, React.js, and D3.js to access that data and display it in a Web page