Express

CCAPDEV

What is Express?

- Express is a light-weight NodeJS framework that provides robust features for web and mobile applications

Installing Express

- Search for Express in the npm repository and follow the installation instructions
  - From a terminal, navigate to your project directory
  - Generate a package.json file
    - through `npm init`
  - Run the command `npm install express`
    - or `npm i express`
    - This will install express to that specific project.

NOTE

npm installs are per project, by default. Use `npm install -g <package name>` to install globally.

Creating an Express App

- Create your first Express App
  - Create a project folder
  - Generate a `package.json` file
  - Install Express
  - Create an `app.js` file
  - Require `express` inside `app.js`
  - Create an instance of an Express app

```
JS app.js > ...
1   const express = require('express');
2   const app = express();
3
```

Creating a Server

- Use your Express app instance to listen for requests

```
app.listen([port])
```

➥ `app`

variable containing your express app instance

➥ `.listen`

listen to requests, just like `server.listen()`

➥ `port`

the port number you wish to listen to

Creating a Server

- Use your Express app instance to listen for requests

```
app.listen(3000)
```

➥ `app`

variable containing your express app instance

➥ `.listen`

listen to requests, just like `server.listen()`

➥ `port`

the port number you wish to listen to

We are now currently listening to requests in port 3000

The Request Listener

- Our server is currently listening for requests. Now, we have to handle our responses to certain requests (much like our switch statement)
- Let’s try listening to GET requests

The Request Listener

- Use app.get() to listen to GET requests (more on GET & POST requests next meeting).
- Let's write a GET request listener:

```
app.get([url], [callback]);
```

➥ `.get`

specifies that we are listening to GET requests

➥ `url`

the value of the request URL

➥ `callback`

the function performed when a GET request for a certain URL is made

The Request Listener

- Use app.get() to listen to GET requests (more on GET & POST requests next meeting).
- Let's write a GET request listener:

```
app.get('/home', function(req, res){ });
```

➥ `.get`

specifies that we are listening to GET requests

➥ `url`

the value of the request URL

➥ `callback`

the function performed when a GET request for a certain URL is made

URL **Routing**

- Let’s handle requests to our web application

```
app.get([url], [callback]);
```

https://expressjs.com/en/guide/routing.html

URL **Routing**

- Let's handle requests to our web application

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

- Same as before, use the `res` object to reply to the request

`res.write()` and `res.end()`

or use express' new method:

`res.send()`

For new response methods provided by express:
https://expressjs.com/en/guide/routing.html#response-methods

URL **Routing**

- This same syntax can be used for other HTTP request methods as well:
  - `app.get([path], [callback])`
  - `app.post([path], [callback])`
  - `app.put([path], [callback])`
  - `app.delete([path], [callback])`

Route Paths

```
app.get([path], function (req, res) { ... });
```

- The route path can be strings, string patterns, and even regular expressions

Route Paths

```
app.get(`/`, function (req, res) { ... });
```

This route path will match requests to the root route, **/**.

Route Paths

```
app.get(`/about`, function (req, res) { ... });
```

This route path will match requests to **/about**.

Route Paths

```
app.get(`/random.text`, function (req, res) { ...
                        });
```

This route path will match to **/random.text**.

Route Paths

```
app.get(`/ab?cd`, function (req, res) { ... });
```

This route path will match /**acd** and /**abcd**.

Route Paths

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

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

Route Paths

```
app.get(`/ab*cd`, function (req, res) { ... });
```

This route path will match /**abcd**, /**abxcd**, /**abRANDOMcd**, and so on.

Route Paths

```
app.get(`/ab(cd)?e`, function (req, res) { ...
                      });
```

This route path will match **/abe** and **/abcde**

Route Paths

You can refer to the official documentation for other examples:

- https://expressjs.com/en/guide/routing.html#route-paths

Route Parameters

- **Route parameters** are named URL segments that are used to capture the values specified at their position in the URL.
- The captured values are populated in the `req.params` object, with the name of the route parameter specified in the path as their respective keys.

Route Parameters

Route path:

`/users/:userId/books/:bookId`

Request URL example:

`http://localhost:9090/users/34/books/8989`

req.params:

```
{ “userId”: “34”, “bookId”: “8989” }
```

Route Parameters

Route path:

`/profile/:username`

Request URL example:

`http://localhost:9090/profile/jimmyneutron`

req.params:

`{ “username”: “jimmyneutron” }`

Route Parameters

- Since the hypen(-) and the dot (.) are interpreted literally, they can be used along with the route parameters for useful purposes.

Route Parameters

Route path:

`/flights/:from-:to`

Request URL example:

`http://localhost:9090/flights/MNL-TPE`

req.params:

```
{ “from”: “MNL”, “to”: “TPE” }
```

Route Parameters

Route path:

`/class/:subject.:section`

Request URL example:

`http://localhost:9090/class/CCAPDEV.S15`

req.params:

```
{ “subject”: “CCAPDEV”, “section”: “S15” }
```

Route handling

- It is also possible to create chainable route handlers that points toward the same path.

```
app.route('/book’)
    .get((req, res) => { res.send('Get a random
    book') })
    .post((req, res) => { res.send('Add a book')
    })
    .put((req, res) => { res.send('Update the
    book') })
```

Static files

- Static files are the files that clients can download as they visit the website. They are made available to the public.
- To declare a directory along its files and subdirectories as static:

```
app.use([root] ,] express.static([folder name]));
```

➥ `root`

optional, specifies a virtual root URL for the static files

➥ `folder name`

the name of the directory in server to be made static

Static files

- public
  - img
    - profile.jpg `http://localhost:3000/img/profile.jpg`
    - instadog.jpg `http://localhost:3000/img/instadog.jpg`

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

Static files

- public
  - img
    - profile.jpg
      http://localhost:3000/static/img/profile.jpg
    - instadog.jpg
      http://localhost:3000/static/img/instadog.jpg

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

Other **Response** methods

- `res.status([code])` – sets the status code of the response in through the [code] parameter
- `res.get([header])` – gets the value of the header specified in the parameter
- `res.set([header],[value])` – sets an HTTP header for the response.
  - can also use a json object:
  - ```
    res.set({
        [header1]: [value1],
        ...
        [headerN]: [valueN]
    }
    ```

https://expressjs.com/en/4x/api.html#res
    ```

Other **Response** methods

- `res.json([json object])` – **sends** a JSON response
- `res.sendFile([filepath])` – **sends** a file from the given filepath.
  - can be called with an options JSON object parameter
  - can be called with a callback function
- `res.download([filepath]) ])` – **sends** a file from the given filepath, but prompts the user first.
  - can be called with an options JSON object parameter
  - can be called with a callback function

https://expressjs.com/en/4x/api.html#res

Other **Response** methods

- `res.redirect([code], [path])` – **redirects** to the URL from the specified `path`. Can also specify an optional status `code` to go along the redirect path.
  - `res.redirect('..')` – goes up one directory from current path
  - `res.redirect('back')` – requests back to the referrer.

https://expressjs.com/en/4x/api.html#res

Further Reading

- **Express middleware** – functions that can be used to preprocess requests/responses, generally handle errors, among others.
  - https://expressjs.com/en/guide/using-middleware.html
- Router object
  - https://expressjs.com/en/4x/api.html#router
- Response documentation
  - https://expressjs.com/en/4x/api.html#res
- Request documentation
  - https://expressjs.com/en/4x/api.html#req

Express

CCAPDEV