CSc 337

LECTURE 1: POST

Exercise - part 3

If there are no books that are in the category that the user supplies have your service return a 410 status and a message about the category not being found.

Set the status with the following code:

```
res.status(410);
```

GET vs POST

Two different types of requests you can send to a web service:

`GET` – asks the server for data

`POST` – sends data to the server

Why can't `POST` requests work the same way as `GET` requests?

Dealing with a `POST` request in a service

```
app.post('/', function (req, res) {
    res.header("Access-Control-Allow-Origin", "*");
    res.send('Hello world');
});
```

- instead of using `app.get` use `app.post`

- Send response as usual

Making a `POST` request from the client

```
fetch(url, {method : 'POST'})
    .then(checkStatus)
    .then(function(responseText) {

    })
    .catch(function(error) {

    });
}
```

- Add a second parameter to `fetch` specifying the method
- There are many methods `POST`, `PUT`, `PATCH`, `DELETE`, …
  - `Get` is the default method

Sending parameters

Send parameters as JSON!

By default parsing is really messy so we will install a helpful package - body-parser

Run this on the command line in your code directory:

```
npm install body-parser
```

Sending Parameters

```
const message = {name: "Allison",
                 email: "aeobourn@cs.arizona.edu"};
const fetchOptions = {
    method : 'POST',
    headers : {
        'Accept': 'application/json',
        'Content-Type' : 'application/json'
    },
    body : JSON.stringify(message);
}
fetch(url, fetchOptions)
    .then(checkStatus)
```

Add a stringified version of the JSON you want to send to an object containing the other options and send that.

Dealing with `POST` parameters on the server

```
const  bodyParser = require('body-parser');
const jsonParser = bodyParser.json();

app.post('/', jsonParser, function (req, res) {
      const name = req.body.name;
      res.send('Hello, ' + name);
});
```

Accessing parameters is similar to with a get request except you need to access `req.body` instead of `req.query`

Prefilght CORS error

The code on the previous slide produces an error because it is accepting complex parameter content and `Allow-Access-Control-Origin` isn't set.

- Adding this line in where we usually do doesn't fix this.
- Add the code below instead:

```
app.use(function(req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Headers",
               "Origin, X-Requested-With, Content-Type, Accept");
  next();
});
```

File Turn In Client

Write a page that allows the user to enter their name, email and select an assignment number from a dropdown menu. It should also include a text area where they can paste their code and a submit button.

When the user clicks the submit button the information should be sent to the server as a `POST` request.

File Turn In Service

Write a web service that accepts that data your client page posted. You service should save the code from the text area into a file named the student's name and section.

The service should send a success message back to the client if it was successful and a failure message back if it was not.

File saving information on the next slide.

Writing Files

appending to a file:

```
fs.appendFile(filename, filecontent, function(err) {
      if(err) {
            return console.log(err);
      }
      console.log("The file was saved!");
});
```

writing to a file:

```
fs.writeFile(filename, filecontent, function(err) {
      if(err) {
            return console.log(err);
      }
      console.log("The file was saved!");
});
```

File Turn In Confirmation

Have your page display a confirmation message stating whether the request was successful. This message should include the code the user submitted if the request came back successful.