A simple HTTP server

Node HTTP module

```javascript
const http = require('http');
const server = http.createServer((request, response) => {
  response.writeHead(200, { 'Content-Type': 'text/plain' });
  response.end("Don't Panic");
}).listen(5042);
console.log('Listening on port %d', server.address().port);
```

Manually construct the response

In action:
```
$ curl http://localhost:5042/
Don't Panic
```

A simple HTTP server with Express

```javascript
const http = require('http');
const express = require('express');
const app = express();
app.get('/', (request, response) => {
  response.send("Don't Panic");
});
app.get('/:name', (request, response) => {
  response.send("Don't panic " + request.params.name);
});
const server = http.createServer(app).listen(5042);
console.log('Listening on port %d', server.address().port);
```

There is a one-to-one mapping between API routes and Express routes

Express architecture

Middleware modify or respond to request
Aspect-oriented Programming (AOP)

- Design pattern for implementing “cross-cutting” concerns
  Middleware is an example of AOP
- “Cross cutting” concerns are those that affect many parts (or concerns) of the code
  Many requests require body parsing
- AOP is a general set of techniques for DRYing up “cross cutting” concerns

Film model (M in MVC)

Film “resource” is a simple JavaScript object

Good enough for now, but what about?
- Validate user rating is 0-5?
- Express associations between models
- Support different persistence layers (e.g. databases)

We can use ORMs and other libraries to provide this “cross cutting” functionality

The models are typically the RESTful resources

<table>
<thead>
<tr>
<th>Route</th>
<th>Controller Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/api/films</td>
</tr>
<tr>
<td>GET</td>
<td>/api/films/:id</td>
</tr>
<tr>
<td>PUT</td>
<td>/api/films/:id</td>
</tr>
<tr>
<td>DELETE</td>
<td>/api/films/:id</td>
</tr>
<tr>
<td>GET</td>
<td>/api/films</td>
</tr>
</tbody>
</table>

A single model: Film

Can't trust the client!
Lo-fi OO modeling: CRC cards*

<table>
<thead>
<tr>
<th>User</th>
<th>Responsibility</th>
<th>Collaborator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows user’s name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Knows films I rated</td>
<td>Rating</td>
</tr>
<tr>
<td></td>
<td>Knows user’s name</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Film</th>
<th>Responsibility</th>
<th>Collaborator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows its title</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Knows its plot overview</td>
<td></td>
</tr>
<tr>
<td>Knows which genres it is</td>
<td></td>
<td>Genre</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rating</th>
<th>Responsibility</th>
<th>Collaborator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows rating</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Knows its owner</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>Knows its film</td>
<td>Film</td>
<td></td>
</tr>
</tbody>
</table>

“many to many” “has many”

CRCs and user stories

Independently rate a film
As a user
I want to rate a film
So that I can save my opinions of films

Show average ratings
As a user
I want to view average ratings of a film
So that I can know if it is a good film

*Kent Beck & Ward Cunningham, OOPSLA 1989*