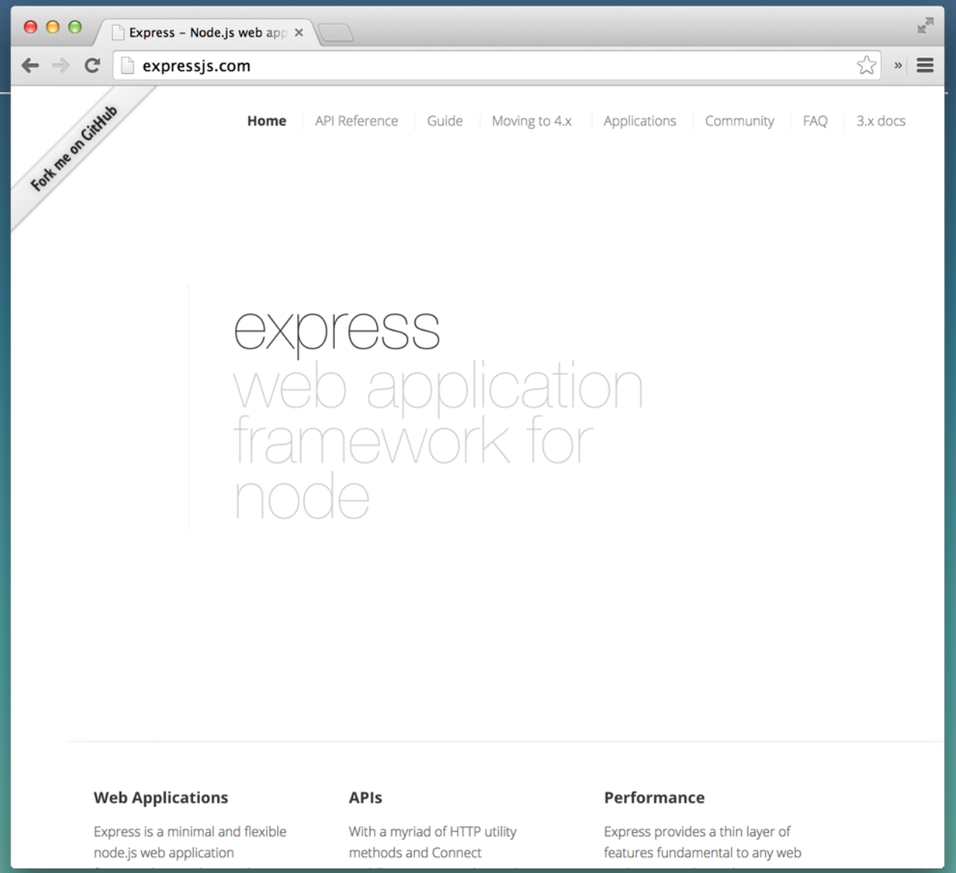
First Steps

Level 1

What Express Is

A web application framework for Node

- Minimal and flexible
- Great for building **Web APIs**
- Popular services built on Express
  i.e. **MySpace, Ghost** and more
- Foundation for other tools and frameworks, like **Kraken** and **Sails**



Installing Express

Node Package Manager

Use **npm** to install the latest stable version


```
$ npm install express
```

Use @ to install a specific version

```
$ npm install express@4.9
$ npm install express@3.15.2
```

installs latest version from the 4.9 branch

installs specific version

This course covers version 4.9.x

Code seen here should run on any version of Express which starts with 4.9 (i.e. 4.9.1, 4.9.2, 4.9.3, etc.)

Writing Hello World

Calling the **express** function gives us an **application instance**

application instance

app.js

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

Writing Hello World

The **app.get** function creates a route that accepts **GET** requests

app.js

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

app.get('/', function(request, response) {
  response.send('Hello world');
});

app.listen(3000);
```

binds application
to tcp port **3000**

sends back server response

built-in functions
named after HTTP verbs

```
app.post(...)
app.put(...)
app.patch(...)
app.delete(...)
...
```

Writing Hello World

The **app.listen** function takes an optional callback, which is called once the app is ready to start taking requests

app.js

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

app.get('/', function(request, response) {
  response.send('Hello world');
});

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

printed to the console

Running our Express app

Start the server with the **node** command

```
$ node app.js
Listening on port 3000
```

Changes to code require a **server restart.**

Requests with **curl**

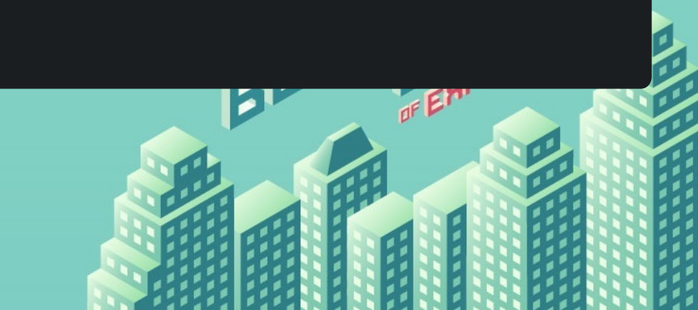
```
$ curl http://localhost:3000/

Hello world
```

server response

Control + C stops the server

```
$ node app.js
Listening on port 3000
^C
```

interrupts current process



The Request and Response objects

Express extends Node HTTP objects

app.js

```
app.get('/', function(request, response) {
  ...
});
```

Express **source code**

https://github.com/strongloop/express

lib/request.js

```
var req = exports = module.exports = {
  __proto__: http.IncomingMessage.prototype
};
...
```

objects from Node HTTP

inheritance in JavaScript

lib/response.js

```
var res = module.exports = {
  __proto__: http.ServerResponse.prototype
};
...
```

Calling Node's HTTP functions

We can respond from Express using Node's **write** and **end** functions

app.js

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

app.get('/', function(request, response) {
  response.write('Hello world');
  response.end();
});

app.listen(3000);
```

using Node API

…very useful when we start writing "extensions" for Express

same thing

```
response.send('Hello world')
```

using Express API

Response from both

```
$ curl http://localhost:3000/

Hello world
```

Responding with JSON

The send function converts **Objects and Arrays** to JSON

app.js

```
app.get('/blocks', function(request, response) {
  var blocks = ['Fixed', 'Movable', 'Rotating'];
  response.send(blocks);
});
```

use **-i** to print response headers

```
$ curl -i http://localhost:3000/blocks

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8

["Fixed","Movable","Rotating"]
```

sets proper **response headers**

Using the response.json function

The **json** function reads better when all we respond with is JSON

app.js

```
app.get('/blocks', function(request, response) {
  var blocks = ['Fixed', 'Movable', 'Rotating'];
  response.json(blocks);
});
```

Same response as **send**, for Objects and Arrays

```
$ curl -i http://localhost:3000/blocks

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8

["Fixed","Movable","Rotating"]
```

Responding with HTML

The **send** function converts strings to **HTML**

app.js

```
app.get('/blocks', function(request, response) {
  var blocks = '<ul><li>Fixed</li><li>Movable</li></ul>';
  response.send(blocks);
});
```

Responds with **text/html**

```
$ curl -i http://localhost:3000/blocks

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8

<ul><li>Fixed</li><li>Movable</li></ul>
```

For server-side templates, checkout **EJS** or **Jade**

Redirecting to relative path

The **redirect** function sets the proper response headers

app.js

```
app.get('/blocks', function(request, response) {
  response.redirect('/parts');
});
```

```
$ curl -i http://localhost:3000/blocks

HTTP/1.1 302 Moved Temporarily
X-Powered-By: Express
Location: /parts
Content-Type: text/plain; charset=utf-8

Moved Temporarily. Redirecting to /parts
```

Redirecting with custom status code

The **status code** can be passed as the first argument to **redirect**

app.js

```
app.get('/blocks', function(request, response) {
  response.redirect(301, '/parts');
});
```

optional status code

```
$ curl -i http://localhost:3000/blocks

HTTP/1.1 301 Moved Permanently
X-Powered-By: Express
Location: /parts
Content-Type: text/plain; charset=utf-8

Moved Permanently. Redirecting to /parts
```