The next frontier

Before I forget...
I'm Ari Lerner

And this is a technical talk

- Get on your keyboard
- Work with me

Coupons coupons coupons!

- 50% off Fullstack React
- 50% off ng-book 2

I have an idea

Now that we all have eaten lunch, it's time to start thinking about


Dinner

Let's get a list of places close by where we can all vote on our favorite restaurants...

How do we build this thing?

What do we need?

- Backend
- Front-end
- Deployment

Building our backend

Some options

- ExpressJS / NodeJS
- Golang / Gin
- Clojure / Pedestal
- Firebase

Custom backend?

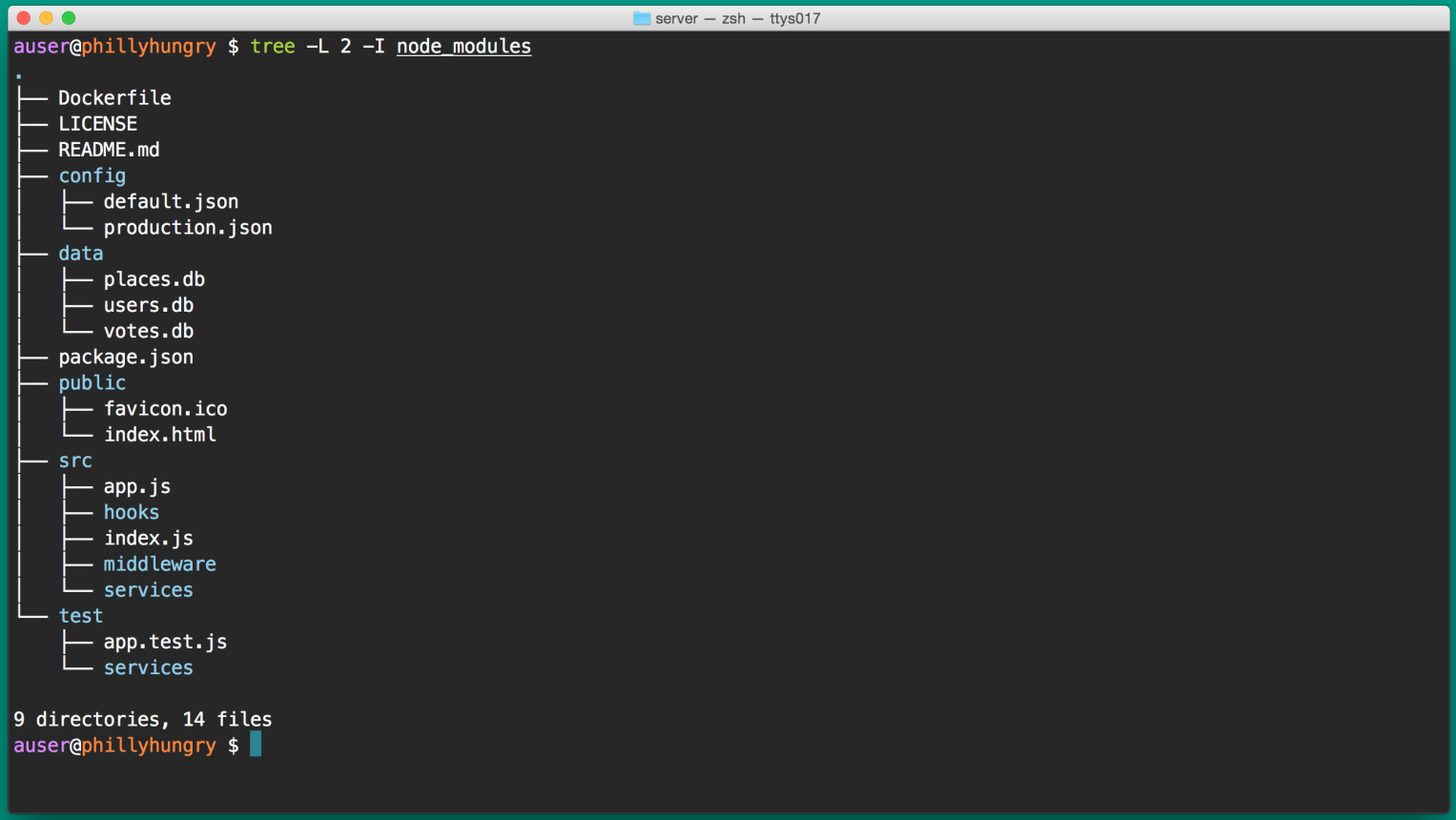
We'll need some authentication and a votes storage.

I propose we use

feathersjs

A real-time framework written in JavaScript with a big community, loads of plugins, open-source

To the code!


server — zsh — ttys017
auser@phillyhungry $ tree -L 2 -I node_modules
.
├── Dockerfile
├── LICENSE
├── README.md
├── config
│   ├── default.json
│   └── production.json
├── data
│   ├── places.db
│   ├── users.db
│   └── votes.db
├── package.json
├── public
│   ├── favicon.ico
│   └── index.html
├── src
│   ├── app.js
│   ├── hooks
│   ├── index.js
│   ├── middleware
│   └── services
└── test
    ├── app.test.js
    └── services
9 directories, 14 files
auser@phillyhungry $

/src/app.js

```
'use strict';

const path = require('path');
const serveStatic = require('feathers').static;
const favicon = require('serve-favicon');
const compress = require('compression');
const cors = require('cors');
const feathers = require('feathers');
const configuration = require('feathers-configuration');

const hooks = require('feathers-hooks');
const rest = require('feathers-rest');
const bodyParser = require('body-parser');
const socketio = require('feathers-socketio');

const middleware = require('./middleware');
const services = require('./services');
```

Services

`/src/services/index.js`

```
1. 'use strict';
2. const votes = require('./votes');
3. const places = require('./places');
4. const authentication =
require('./authentication');
5. const user = require('./user');
6.
7. module.exports = function() {
8.   const app = this;
9.
10.   app.configure(authentication);
11.   app.configure(user);
12.   app.configure(places);
13.   app.configure(votes);
14. };
15.
```

User service

`/src/services/user/index.js`

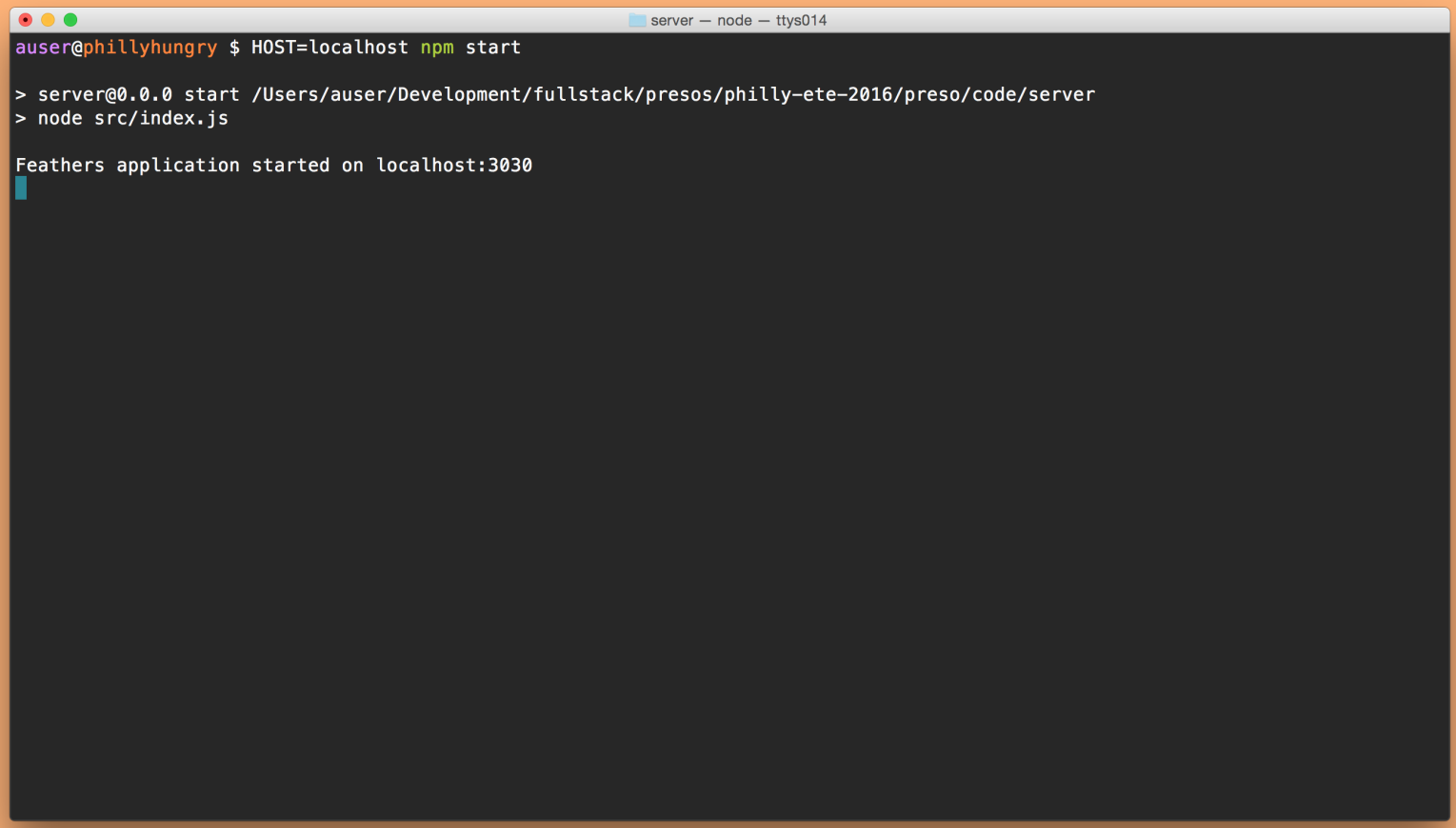
```
const NeDB = require('nedb');
const service = require('feathers-nedb');
const hooks = require('./hooks');

module.exports = function(){
  const app = this;

  const db = new NeDB({
    filename: path.join(app.get('nedb'), 'users.db'),
    autoload: true
  });

  let options = {
    Model: db,
    paginate: {
      default: 5,
      max: 25
    }
  };
```

Hooks

`/src/services/user/hooks/index.js`

```
1. 'use strict';
2.
3. const globalHooks = require('../../../hooks');
4. const hooks = require('feathers-hooks');
5. const auth = require('feathers-
authentication').hooks;
6.
7. exports.before = {
8.   all: [],
9.   find: [
10.     auth.verifyToken(),
11.     auth.populateUser(),
12.     auth.restrictToAuthenticated()
13.   ],
14.   get: [
15.     auth.verifyToken(),
16.     auth.populateUser(),
17.     auth.restrictToAuthenticated(),
```

Running


server — node — ttys014
auser@phillyhungry $ HOST=localhost npm start
> server@0.0.0 start /Users/auser/Development/fullstack/presos/philly-ete-2016/preso/code/server
> node src/index.js
Feathers application started on localhost:3030

Front-end

Lots of options too:

jquery

javascript

Aurelia

But let's be real

Angular 2

Spectacular community

Backed by Google

Ever-growing community resources

FAST (code through execution)

Has multiple renderers (native, browser, etc)

React

Spectacular community

Backed by Facebook

Ever-growing community resources

FAST (code through execution)

Has multiple renderers (browser, native, etc)

Why React?
Why Angular?

Why not *both*?

One more thing...

Typescript

No more chit-chat

Let's build our front-end

template.html.js

```
module.exports = function(data) {
  const package       = data.package;
  const webpackConfig = data.webpackConfig;
  const metadata      = data.metadata;

  return `
    <!DOCTYPE html>
    <html lang="">
    <head>

      <title>${metadata.title}</title>

      <!-- base url -->
      <base href="/">
      <link type="text/css" src="app.css" />
    </head>

    <body>

      <div id="root"></div>
      <app></app>

      ${Object.keys(webpackConfig.entry).map(f => {
        return `<script src="${f}.js"></script>`
      })}

    </body>
    </html>
    `
  }
```

/src/app.tsx

Connecting to feathersjs

```
20. const key = __GAPI_KEY__;
21. const apiUrl = __API_URL__;
22.
23. const feathers = require('feathers-client');
24. const io = require('socket.io-client');
25. const cookie = require('cookie');
26. ////////////////////////
27.
28. // setup app
29. const socket = io.connect(apiUrl);
30. let app = feathers()
31. app.configure(feathers.hooks())
32. app.configure(feathers.socketio(socket))
33. app.configure(feathers.authentication({
34.   storage: window.localStorage
35. }));
36.
37. let userData;
38. socket.on('connect', function(sock) {
39.   app.authenticate().then((res) => {
```

Authentication and connection

```
37. let userData;
38. socket.on('connect', function(sock) {
39.   app.authenticate().then((res) => {
40.     userData = {
41.       userId: res.data._id,
42.       token: res.token
43.     }
44.   });
45. })
46.
47. // Initial auth
48. const c = cookie.parse(document.cookie)
49. if (c && c['feathers-jwt']) {
50.   window.localStorage.setItem('feathers-jwt',
c['feathers-jwt']);
51. }
52.
53. // Get geo
54. const {geolocation} = window.navigator;
55.
```

Angular app

```
232.     </div>
233.     `
234. })
235. export class List {
236.     @Input() places:any;
237.     @Input() vote:any;
238.     @Input() votes:any;
239.     @Output('voteFor') voteFor = new
EventEmitter();
240. }
241.
242. @Component({
243.     selector: 'app',
244.     directives: [HeaderNode, List],
245.     providers: [VotesService],
246.     template: `
247.       <div class="${styles.wrapper}">
248.         <header-node title="Hungry Philly">
</header-node>
```

Bootstrapping component

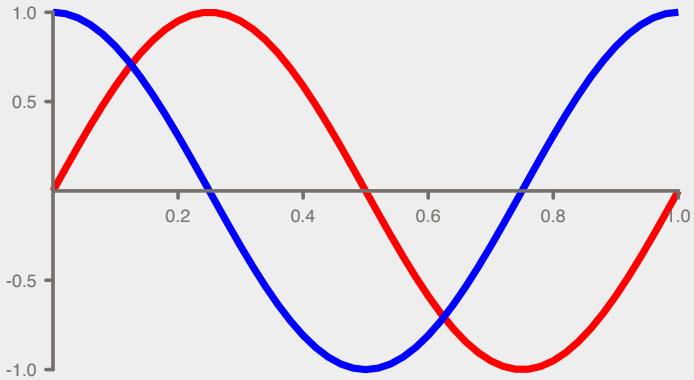
```
376.         user: userData,
377.         location: {
378.           currentLocation: {
379.             latitude: coords.latitude,
380.             longitude: coords.longitude
381.           }
382.         },
383.         places: [],
384.         votes: {}
385.       }),
386.     ])
387.   });
388. }).catch((err) => {
389.   bootstrap(Login, [])
390. });
391.
392. if ('production' === __NODE_ENV__)  {
393.   enableProdMode();
394. }
```

Let's code

React

Virtual DOM


Bird
Cat
Fish
Dog


1.0
0.5
-0.5
-1.0
0.2
0.4
0.6
0.8
1.0

VirtualDOM

React components

```
1. import React from 'react';
2.
3. class App extends React.Component {
4.   render() {
5.     return (<div>Hello world</div>)
6.   }
7. }
8.
9. const App = React.createClass({
10.   render: function() {
11.     return (<div>Hello world</div>)
12.   }
13. })
14.
15. const App = (props) => <div>Hello world</div>
```

I have this killer React component...

Let's integrate the two!

Let's integrate the two!

```
249.         <div class="${styles.sidebar}">
250.           <places-list
251.             [places]="places | async"
252.             [vote]="vote | async"
253.             [votes]="votes | async"
254.             (voteFor)="voteFor($event)"
255.           ></places-list>
256.         </div>
257.         <div class="${styles.content}">
258.           <div id='map'></div>
259.         </div>
260.       </div>`
261. })
262. export class App implements OnInit {
263.   places: Observable<any>;
264.   vote: Observable<any>;
265.   location: Observable<GeoLocation>;
266.   currentLocation: GeoLocation;
267.
268.   votes: Observable<{ [key: string]: Vote[]
```

Data between the two

Redux/ngrx

Short intro to Redux

- Single state tree
- Transformed with simple functions
- Driven by flux-like/central dispatchers

ngrx

```
53. // Get geo
54. const {geolocation} = window.navigator;
55.
56. import * as React from "react";
57. import * as ReactDOM from "react-dom";
58. import * as Redux from 'redux';
59.
60. const {Provider} = require('react-redux');
61.
62. import {Reducer, Action, provideStore, Store}
from '@ngrx/store';
63.
64. export const types = {
65.   'USER_LOGIN': 'USER_LOGIN',
66.
67.   'VOTE_FOR': 'VOTE_FOR',
68.
69.   'SET_LOCATION': 'SET_LOCATION',
70.   'GET_PLACES': 'GET_PLACES',
71.   'SET_PLACES': 'SET_PLACES'
```

Passing to react-redux

```
336.
337.       this.searchNearby(map, map.center, cb);
338.     }
339.
340.     ngOnInit() {
341.       const mountNode =
document.querySelector('#map')
342.       const {location} = this.store.getState();
343.       const {currentLocation} = location;
344.
345.       const render = () => {
346.         const {places} = this.store.getState();
347.
348.         const provider = (
349.           <Provider store={this.store}>
350.             <MapComponent
351.               apiKey={key}
352.               lat={currentLocation.latitude}
353.               lng={currentLocation.longitude}
354.               zoom={14}
```

CSS Modules

CSS sucks...

Mostly because it has all these damn *globals*

Can we get rid of the globals?

Sure

CSS Modules

```
ChangeDetectionStrategy} from 'angular2/core';
 10. import {NgClass} from 'angular2/common';
 11. import {bootstrap} from
'angular2/platform/browser';
 12. import {Observable, Subject} from 'rxjs';
 13.
 14. import {Vote} from './models';
 15. import {VotesService} from './services.tsx';
 16.
 17. const classnames = require('classnames');
 18. const styles = require('./styles.module.css');
 19.
 20. const key = __GAPI_KEY__;
 21. const apiUrl = __API_URL__;
 22.
 23. const feathers = require('feathers-client');
 24. const io = require('socket.io-client');
 25. const cookie = require('cookie');
 26. ////////////////////////
 27.
```

```
styles.container = "some-crazy__randomly_created-not-global__classname"
```

Getting up and running

webpack

build chains suck

really really suck

Good news is

You're not alone

Nobody likes build chains

But we have a lot of work done for us

Our webpack config

- hot reloading
- babel & typescript
- auto replace
- postcss
- css modules
- and more

```
const env = process.env;
const NODE_ENV = process.env.NODE_ENV;
const isDev  = NODE_ENV === 'development';
const isTest = NODE_ENV === 'test';

const dotenv       = require('dotenv');

const webpack = require("webpack");
const fs      = require('fs');
const path    = require('path'),

      join    = path.join,
      resolve = path.resolve;

const root  = resolve(__dirname);
const src   = join(root, 'src');
const modules = join(root, 'node_modules');
const dest  = join(root, 'dist');
```

Booting our app

Deployment

Short answer

Docker

```
FROM nginx
COPY ./dist /usr/share/nginx/html
```

Docker-compose

```
frontend:
  image: nginx
  volumes_from:
    - html
  ports:
    - "8000:80"

server:
  build: ./server
  command: npm run start

  environment:
    - NODE_ENV=production
```

```
13.       - PORT=3030
```

Launching

Docker Machine

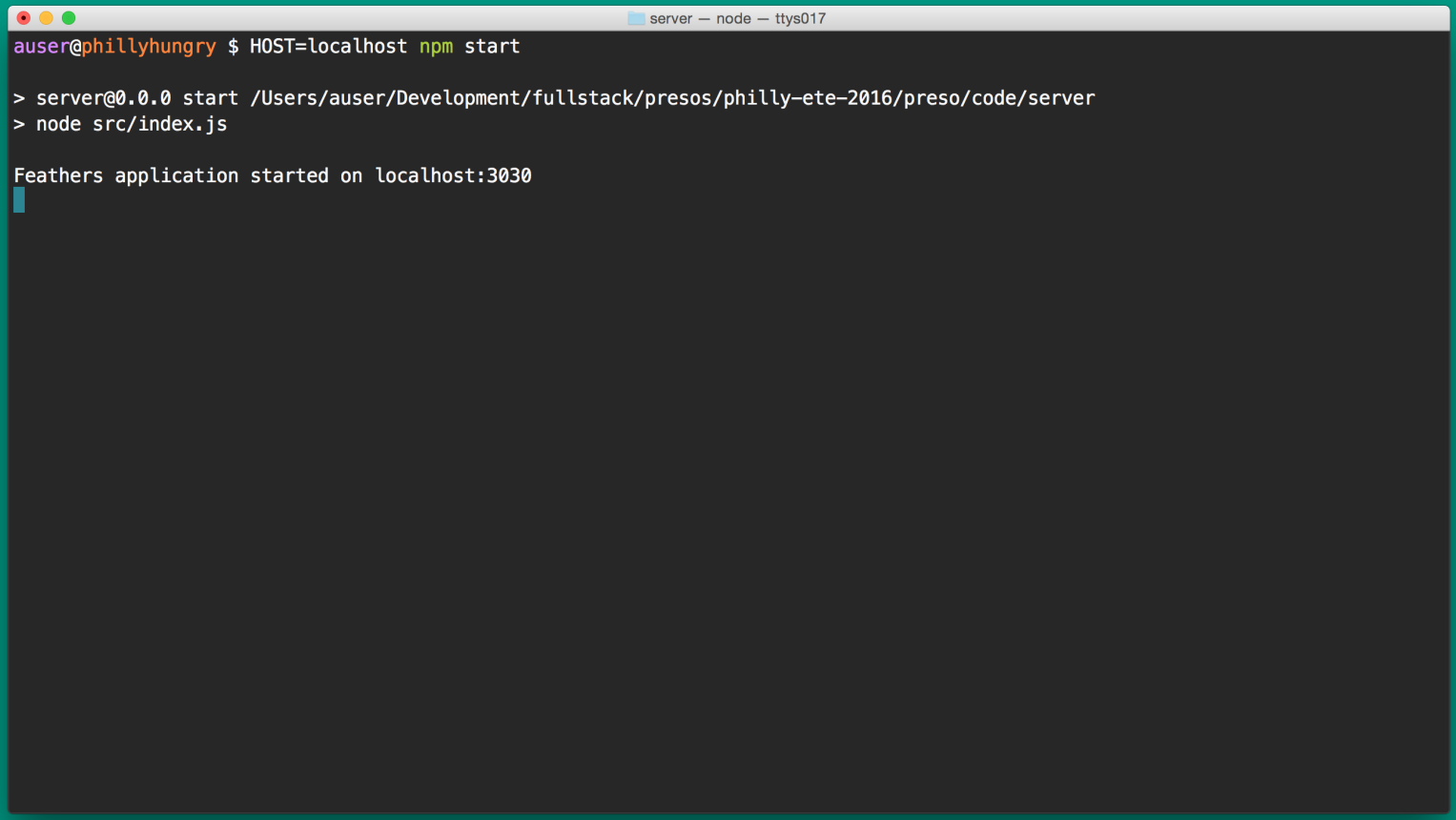
```
docker-machine create default
```

```
$ docker-compose build
```

Get some coffee...

```
$ docker-compose up
```

```
docker-machine create \ --driver amazonec2 \ --amazonec2-access-key your-aws-access-key \ --amazonec2-secret-key your-
```


server — node — ttys017
auser@phillyhungry $ HOST=localhost npm start
> server@0.0.0 start /Users/auser/Development/fullstack/presos/philly-ete-2016/preso/code/server
> node src/index.js
Feathers application started on localhost:3030


Thank You

Coupons coupons coupons!

- 50% off Fullstack React
- 50% off ng-book 2