# how not to suck at mp2

CS498RK
September 26th, 2017

# the stack

# node

What is node?

- A server-side platform built on top of Chrome’s V8 Javascript engine. (It’s actually written in C++)
- Enables client side programmers to write server-side applications without learning a new language!
- Most common use case (and a potential future MP?!) - writing a RESTful web-service as a middleware on your web-service layer, sitting between your client-side library and your backend pipeline.
- Deeper dive in coming lectures..

# npm

- Automated package and dependency manager for Node.js

Why use npm?

- Magically manages the dependency tree for your web-apps.
- Complex apps require a lot of functionalities which can be a pain to write from scratch over again, so we install packages written by other developers.
- How exactly we do that? Next slide...

# Working with npm

- Specify all your dependencies/devdependencies inside `package.json`
- Anytime you (or anyone else) need to get started with your project, they can simply run `npm install` to install all dependencies.
- To install a new dependency, run `npm install [package-name]`.
  - Use `-g` flag to install globally (unsafe, version updates would break this!)
  - Use `--save` to add the dependency to `package.json`
  - Use `--save-dev` to save as a dev dependency.
    - Why use dev dependencies? To separate production dependencies from non-prod ones.
    - Use `--production` flag with npm install to only install non devdependencies.
- It is also possible to specify what versions your project depends upon to prevent updates from breaking your project.

# take good care of your .gitignore (seriously)

- git tracks your files. Your .gitignore tells your local git to ignore certain files
- specifically, it makes sure we don’t track our node_modules folder
- node_modules may have installed dependencies specific to your computer
- we don’t want to track that folder since dependencies take up a lot of space
- we will rebuild the dependencies on our server when you submit!

# how to break the internet

leftpad.js package.json

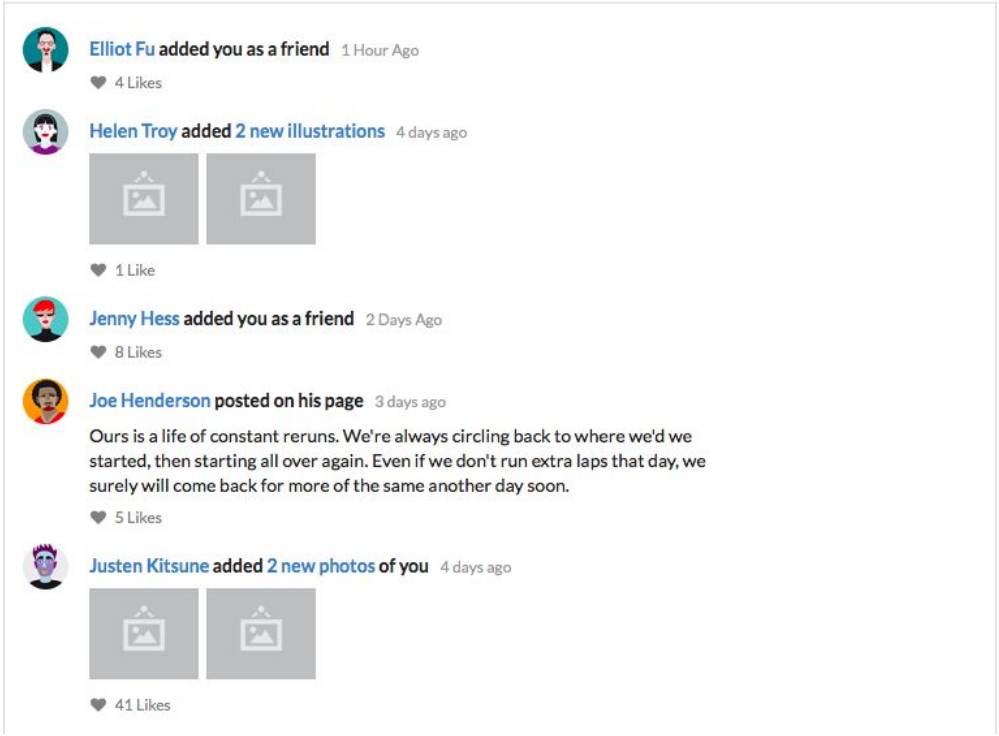
```
1  module.exports = leftpad;
2  function leftpad (str, len, ch) {
3    str = String(str);
4    var i = -1;
5    if (!ch && ch !== 0) ch = ' ';
6    len = len - str.length;
7    while (++i < len) {
8      str = ch + str;
9    }
10   return str;
11 }
12
13
14
15
16
17
18
19
20
```

https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code/

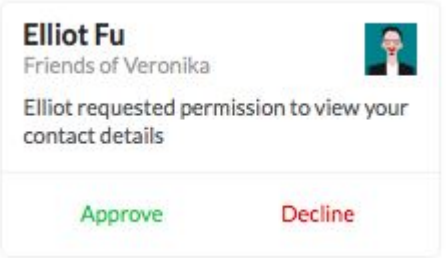
# semantic-ui

- Developmental framework that helps create responsive layouts
- Concise class names that are easy to use because of their “semantic” nature
- Many available UI components such as modals, tabs, accordions, etc. You can view all of them at once here - https://semantic-ui.com/kitchen-sink.html
- 20+ available themes
- Great documentation with a ton of examples
- Clean, modern design
- And there is a react version : https://react.semantic-ui.com/introduction

# Create beautiful components without writing any CSS







# axios

- Promise-based http client
- Make XMLHttpRequests from the browser
- Supports the Promise API
- Intercept request and response
- Transform request and response data
- Cancel requests
- Automatic transforms for JSON data
- Client side support for protecting against XSRF (Cross-site request forgery)

In short, it wraps a lot of native http methods for us!

# axios api

**Creating your axios request object**

```
axios({
  method:'get',
  url:'http://bit.ly/2mTM3nY',
  responseType:'stream'
})
```

or

```
axios.get('http://bit.ly/2mTM3nY')
```

**Request method aliases**

```
axios.request(config)

axios.get(url[, config])

axios.delete(url[, config])

axios.head(url[, config])

axios.options(url[, config])

axios.post(url[, data[, config]])

axios.put(url[, data[, config]])

axios.patch(url[, data[, config]])
```

# react

- version we're using:  15.4.2
- typecheck your props
  - PropTypes
- state
- lifecycle methods
  - componentDidMount
  - componentWillMount
  - componentWillReceiveProps

# react-router

- Routes -> different url paths on your website
- React Router -> lets you route different components to different urls on your website
- Your app is still a single page application (browser doesn't refresh)
- 2 parts
  - Router
  - NavLinks

```
<Router>
  <div className='container'>
    <Switch>
      <Route exact path='/' component={Create}/>
      <Route exact path='/landing' component={Landing} />
      <Route exact path='/projects' component={Projects} />
      <Route exact path='/progress' component={Progress} />
      <Route exact path='/contact' component={Contact} />
      <Route render={function() {
        return <p> Not Found </p>
      }} />
    </Switch>
  </div>
</Router>
```

```
<div className="nav">
    <NavLink exact className="nav-link" activeClassName='active' to='/'>
        <span>create</span>
    </NavLink>
    <NavLink className="nav-link" activeClassName='active' to='/projects'>
        <span>projects</span>
    </NavLink>
    <NavLink className="nav-link" activeClassName='active' to='/progress'>
        <span>progress</span>
    </NavLink>
    <NavLink className="nav-link" activeClassName='active' to='/contact'>
        <span>contact</span>
    </NavLink>
</div>
```

# your mission (should you choose to accept it)

Create a gallery using an API of your choice!

- API: Spotify, TMDB, pokeAPI
- A list view: where users can input a search query and the app returns a list of results that match the query (i.e. searching movies or albums).

- A gallery view: that displays some kind of image media from the chosen API (gallery of movie posters).

- A detail view: When an item in the search view or the gallery view is clicked, the app should display the different attributes of the selected item.

- Look at the docs for a complete list of requirements!

Questions?