```
npm init -y .
```

Set up webpack + dev server

```
npm install --save-dev express webpack webpack-dev-server
```

```
vim dev-server.js
var webpack = require('webpack');
var WebpackDevServer = require('webpack-dev-server');
var config = require('./webpack.config.dev');

new WebpackDevServer(webpack(config), {
  publicPath: config.output.publicPath,
}).listen(3000, 'localhost', function (err, result) {
  if (err) {
    return console.log(err);
  }

  console.log('Listening at http://localhost:3000/');
});
```

```
vim webpack/config.dev.js
var path = require('path');
var webpack = require('webpack');

module.exports = {
  devtool: 'eval',
  entry: [
    'webpack-dev-server/client?http://localhost:3000',
    './src/index'
  ],
  output: {
    path: path.join(__dirname, 'dist'),
    filename: 'bundle.js',
    publicPath: '/static/'
  },
  plugins: [
  ],
  module: {
    loaders: [{
      test: /\.js$/,
      loaders: [],
      include: path.join(__dirname, 'src')
    }]
  }
};
```

```
vim index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Sample App</title>
  </head>
  <body>
    <div id='root'></div>
    <script src="/static/bundle.js"></script>
  </body>
</html>
```

```
vim src/index.js
document.body.innerHTML = 'Hello World!';
```

Set up Babel

```
npm install --save-dev babel-core babel-loader babel-preset-es2015 babel-preset-react
```

- add `loaders: ['babel']` to webpack.config.dev.js

```
vim .babelrc
{
  "presets": ["es2015", "react"]
}
```

Restart server, show we can write some ES2015.

Set up React

```
npm install --save react react-dom
```

```
vim src/App.js
import React from 'react';

export default class App extends React.Component {
  render() {
    return <p>Hello World</p>;
  }
}
```

```
vim src/index.js
import App from './App';
import React from 'react';
import { render } from 'react-dom';

render(<App />, document.getElementById('root'));
```

Set up hot reloading

```
npm install --save-dev react-hot-loader
```

- add `hot: true` to `dev-server.js`

```
vim webpack.config.dev.js
  entry: [
    'webpack-dev-server/client?http://localhost:3000',
    'webpack/hot/only-dev-server',
    './src/index'
  ],

  plugins: [
    new webpack.HotModuleReplacementPlugin()
  ],



 loaders: ['react-hot', 'babel'],
```

Demonstrate hot reloading: build counter

```
vim src/App.js
import React from 'react';

export default class App extends React.Component {
  constructor() {
    super();
    this.state = { count: 0 };
  }

  increment() {
    this.setState({
      count: this.state.count + 1
    });
  }

  render() {
    return (
      <div>
        <p>{ this.state.count }</p>
        <button onClick={() => this.increment()}>Increment</button>
      </div>
    )
  }
}
```

Error handling

Show what happens with typo - stuff shown in browser console.

Testing

```
npm install --save-dev babel-register tape faucet react-addons-test-utils
enzyme jsdom
```

```
vim package.json
    "test": "tape -r babel-register -r ./test/setup-jsdom.js test/*-
test.js | faucet"
```

```
vim test/setup-jsdom.js
import jsdom from 'jsdom';

function setupDom() {
  if (typeof document === 'undefined') {
    global.document = jsdom.jsdom('<html><body></body></html>');
    global.window = document.defaultView;
    global.navigator = window.navigator;
  }
}

setupDom();
```

```
vim test/App-test.js
import test from 'tape';
import React from 'react';
import App from '../src/App';

import { mount } from 'enzyme';

test('App counter', t => {
  t.test('it has an initial count of 0', t => {
    t.plan(1);
    const wrapper = mount(<App />);
    t.equal(wrapper.find('p').text(), '0');
  });

  t.test('clicking increments the count', t => {
    t.plan(1);
    const wrapper = mount(<App />);
    wrapper.find('button').simulate('click');
    t.equal(wrapper.find('p').text(), '1');
  });
});
```

Bundling to Production

```
vim package.json
"build-prod": "webpack --config webpack.config.prod.js --progress"
```

```
vim webpack.config.prod.js
var path = require('path');
var webpack = require('webpack');

module.exports = {
  entry: './src/index.js',
  output: {
    path: path.join(__dirname, 'dist'),
    filename: 'bundle.js'
  },
  plugins: [
    new webpack.DefinePlugin({
      'process.env': {
        'NODE_ENV': JSON.stringify('production')
      }
    }),
    new webpack.optimize.UglifyJsPlugin({
      compress: {
        warnings: false
      }
    }),
    new webpack.optimize.OccurenceOrderPlugin(),
    new webpack.optimize.DedupePlugin()
  ],
  module: {
    loaders: [{
      test: /\.js$/,
      loaders: ['babel'],
      include: path.join(__dirname, 'src')
    }]
  }
};
```