# React in Angular

slides at https://github.com/mvolkmann/talks

**R. Mark Volkmann**
Object Computing, Inc.
http://objectcomputing.com
Email: mark@objectcomputing.com
Twitter: @mark_volkmann
GitHub: mvolkmann

## Overview

- React components can be used in Angular apps!
- Even in ones created with the Angular CLI
  - `npm install -g @angular/cli`
  - `ng new my-app`
  - `cd my-app`
  - `ng serve`
  - browse `localhost:4200`

# npm Installs

- Core React library
  - `npm install react`
- React library for DOM rendering
  - `npm install react-dom`
- React library for specifying “prop types”
  - `npm install prop-types`
  - could also use Flow or TypeScript
- Babel for transpiling React code containing JSX
  - `npm install -D babel-cli`
- Babel preset plugin
  - `npm install -D babel-preset-env`
  - “automatically determines the Babel plugins you need based on your supported environments”
- Babel JSX plugin
  - `npm install -D babel-plugin-transform-react-jsx`

# Babel Setup

- Create **`.babelrc`** file at project root

.babelrc

```
{
  "presets": [
    ["env", {
      "targets": {
        "browsers": ["last 2 versions"]
      }
    }]
  ],
  "plugins": [
    "transform-react-jsx"
  ]
}
```

- Add npm script to run **`babel`** on all React component source files

```
"react-build": "babel 'src/react/**/*.jsx' -d .",
```

  - only runs on **`.jsx`** files under **`src/react`**
  - creates **`.js`** files in same directory that can be imported into Angular source files
    - don't check these into version control

## Example React Component

- In **`src/react`** directory

hello.jsx

```
import * as React from 'react';
import PropTypes from 'prop-types';

export class Hello extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}!</h1>;
  }
}

Hello.propTypes = {
  name: PropTypes.string.isRequired
};
```

## react-util.js

- Create **`src/react/react-util.js`**

```
import * as React from 'react';
import ReactDOM from 'react-dom';

export function render(component, props, targetId) {
  const target = document.getElementById(targetId);
  const element = React.createElement(component, props);
  ReactDOM.render(element, target);
}
```

  - avoids repeating this code every time a React component needs to be rendered

## Target Element

- Add elements with ids where React components will be rendered in the HTML of any Angular components
- For example, could all the following to **`src/app/app.component.html`**

```
<div id="react-hello"></div>
```

# Render With ReactDOM

- Modify the code of any Angular component to render React components
- For example, could do the following in **`src/app/app.component.ts`**
  - add **`OnInit`** to imports from **`'@angular/core'`**
  - add these imports

```
import {render} from '../react/react-util';
import {Hello} from '../react/hello';
```

  - add "**`implements OnInit`**" to the component class
  - add **`ngOnInit`** method

```
  ngOnInit() {
    render(Hello, {name: 'Mark'}, 'react-hello');
  }
```

where **`Hello`** is a React component,
**`{name: 'Mark'}`** is the props to be passed,
and **`'react-hello'`** is the id of the target element

## Steps to Run

- `npm run react-build`
    - to compile all the React .jsx files to .js files
- `npm start`
    - to start a local Angular web server
- browse `localhost:4200`