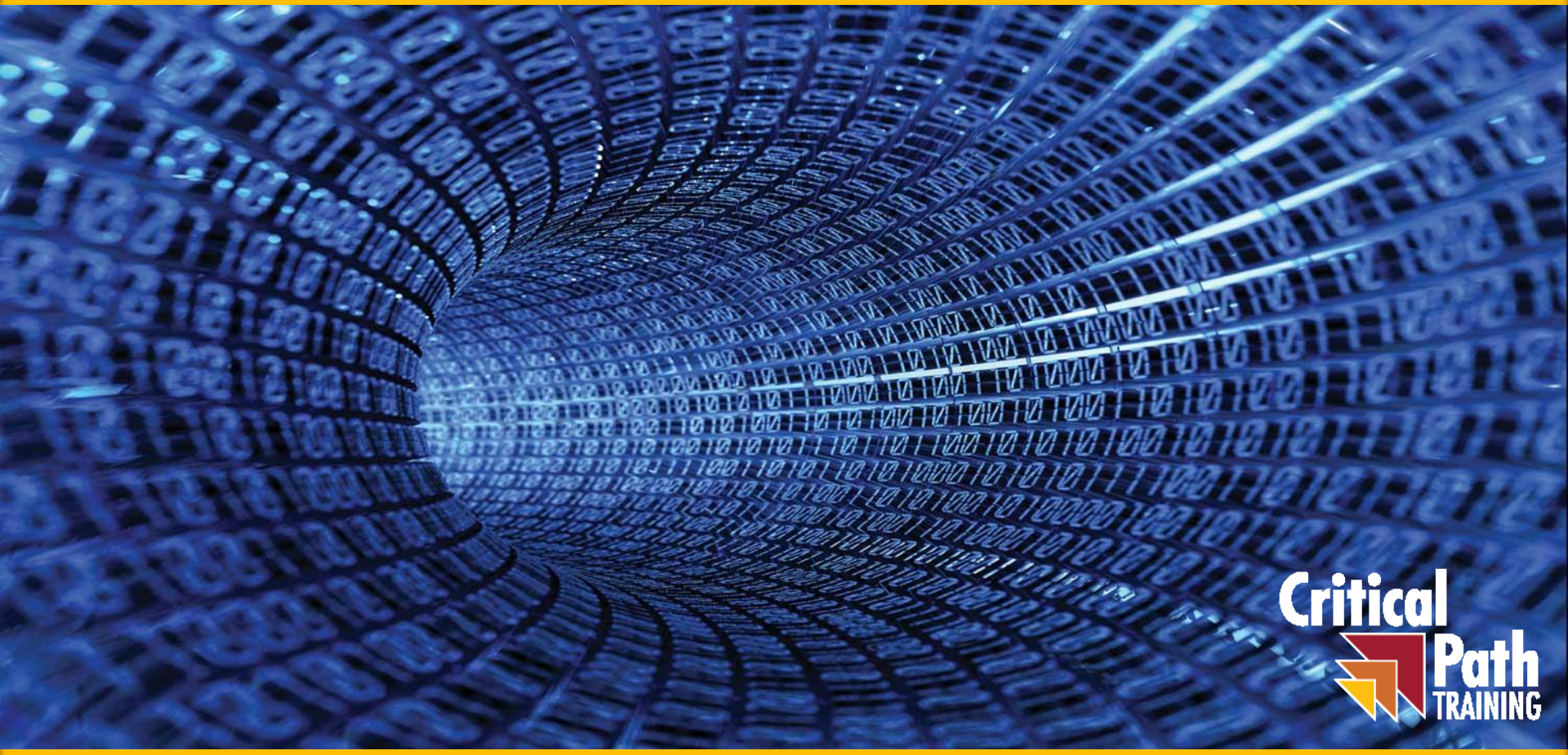
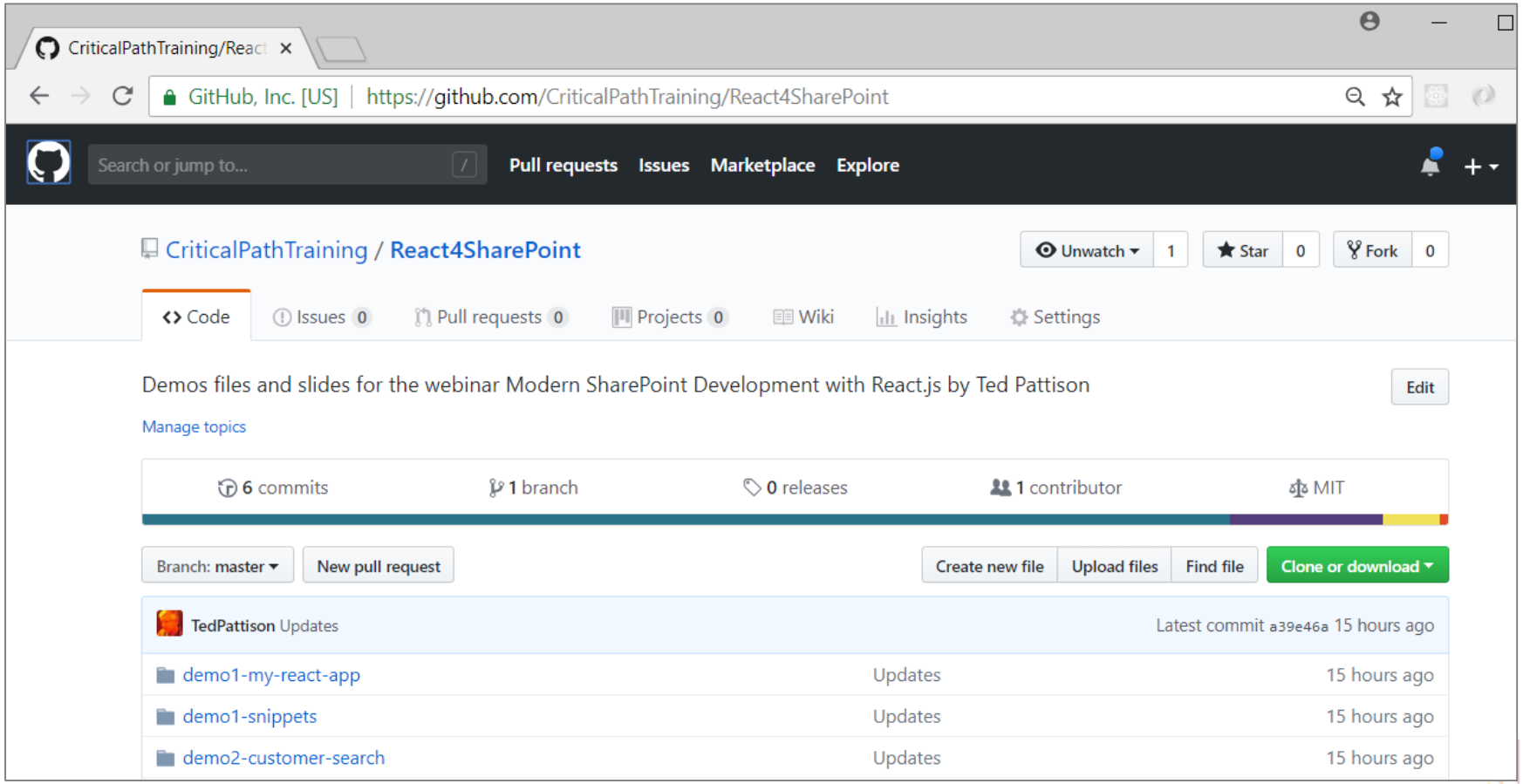
Modern SharePoint Development with React.js



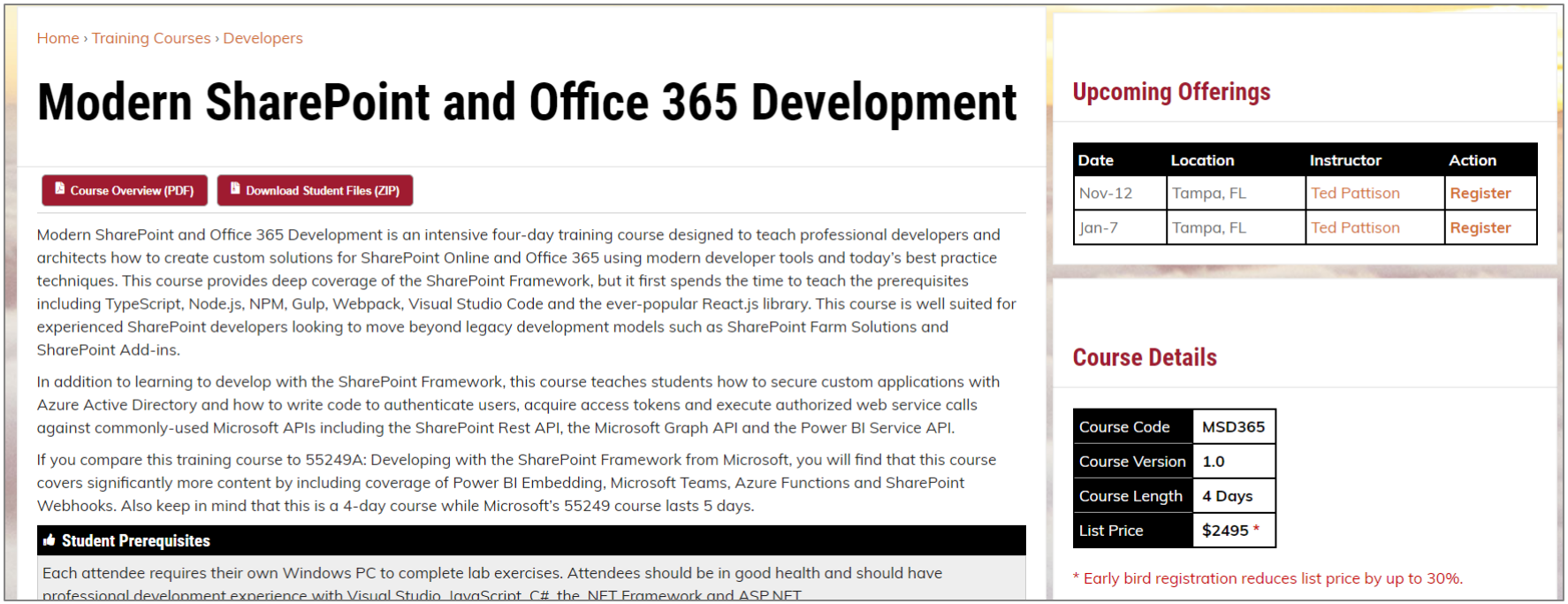
GitHub Repo

- All sample code and slides available for download
  - https://github.com/CriticalPathTraining/React4SharePoint



Effective SharePoint Framework Training

- MSD365: Modern SharePoint and Office 365 Development
  - 4-day of training with lots of hands-on labs
  - Learn how to develop with SPFx the right way right from the start
  - https://www.criticalpathtraining.com/courses/sharepoint/modern-sharepoint-office-365-development/



Home › Training Courses › Developers

Modern SharePoint and Office 365 Development

Course Overview (PDF) | Download Student Files (ZIP)

Modern SharePoint and Office 365 Development is an intensive four-day training course designed to teach professional developers and architects how to create custom solutions for SharePoint Online and Office 365 using modern developer tools and today's best practice techniques. This course provides deep coverage of the SharePoint Framework, but it first spends the time to teach the prerequisites including TypeScript, Node.js, NPM, Gulp, Webpack, Visual Studio Code and the ever-popular React.js library. This course is well suited for experienced SharePoint developers looking to move beyond legacy development models such as SharePoint Farm Solutions and SharePoint Add-ins.

In addition to learning to develop with the SharePoint Framework, this course teaches students how to secure custom applications with Azure Active Directory and how to write code to authenticate users, acquire access tokens and execute authorized web service calls against commonly-used Microsoft APIs including the SharePoint Rest API, the Microsoft Graph API and the Power BI Service API.

If you compare this training course to 55249A: Developing with the SharePoint Framework from Microsoft, you will find that this course covers significantly more content by including coverage of Power BI Embedding, Microsoft Teams, Azure Functions and SharePoint Webhooks. Also keep in mind that this is a 4-day course while Microsoft's 55249 course lasts 5 days.

Student Prerequisites

Each attendee requires their own Windows PC to complete lab exercises. Attendees should be in good health and should have professional development experience with Visual Studio, JavaScript, C#, the .NET Framework and ASP.NET

Upcoming Offerings

Date	Location	Instructor	Action
Nov-12	Tampa, FL	Ted Pattison	Register
Jan-7	Tampa, FL	Ted Pattison	Register

Course Details

Course Code	MSD365
Course Version	1.0
Course Length	4 Days
List Price	$2495 *

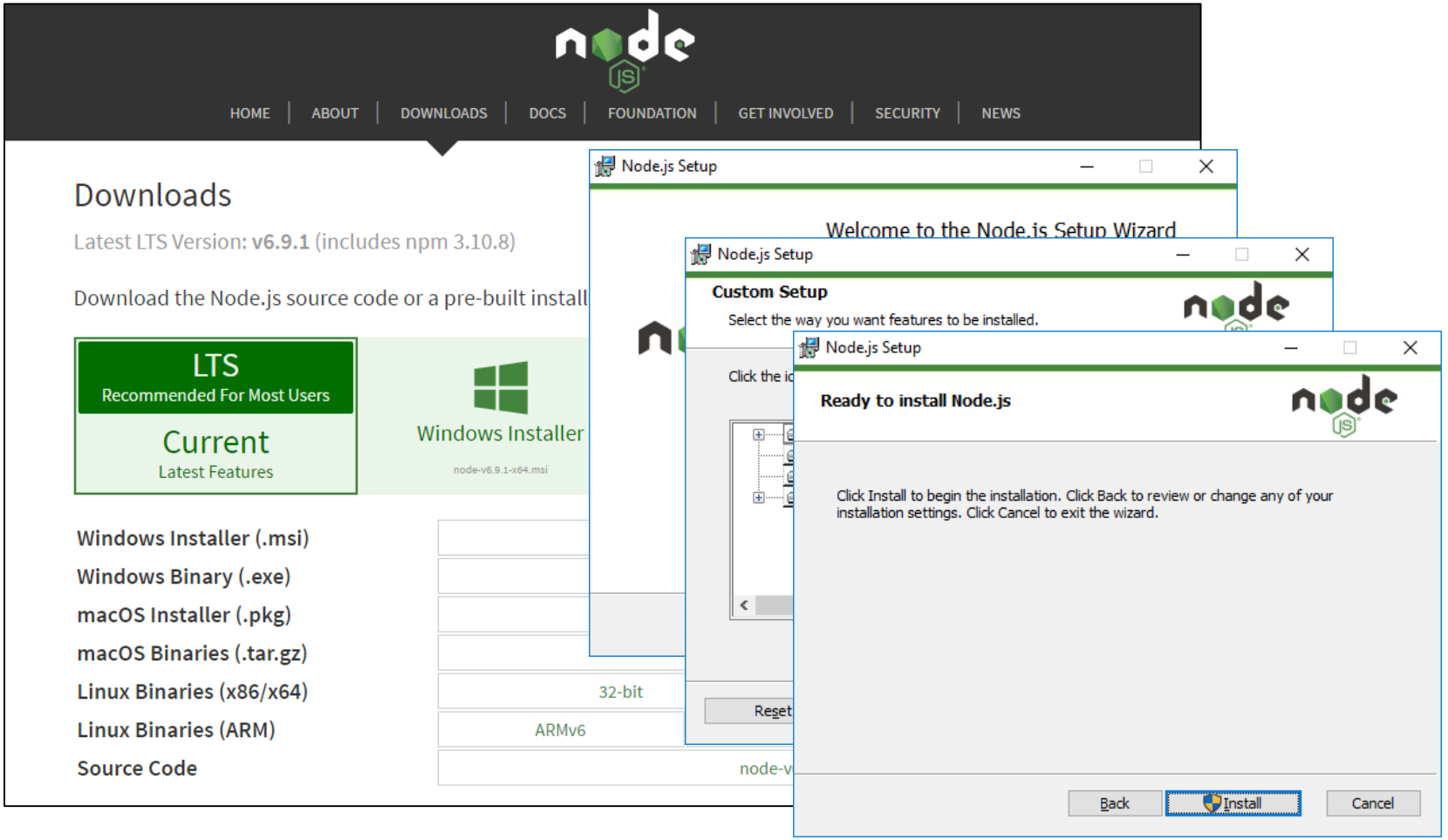
* Early bird registration reduces list price by up to 30%.

Agenda

- Developing with Node.js, TypeScript and Webpack
- Learning React.js Fundamentals
- Using the Office UI Fabric React Component Library
- Developing React Webparts with SharePoint Framework
- Calling the Microsoft Graph API from React Webparts

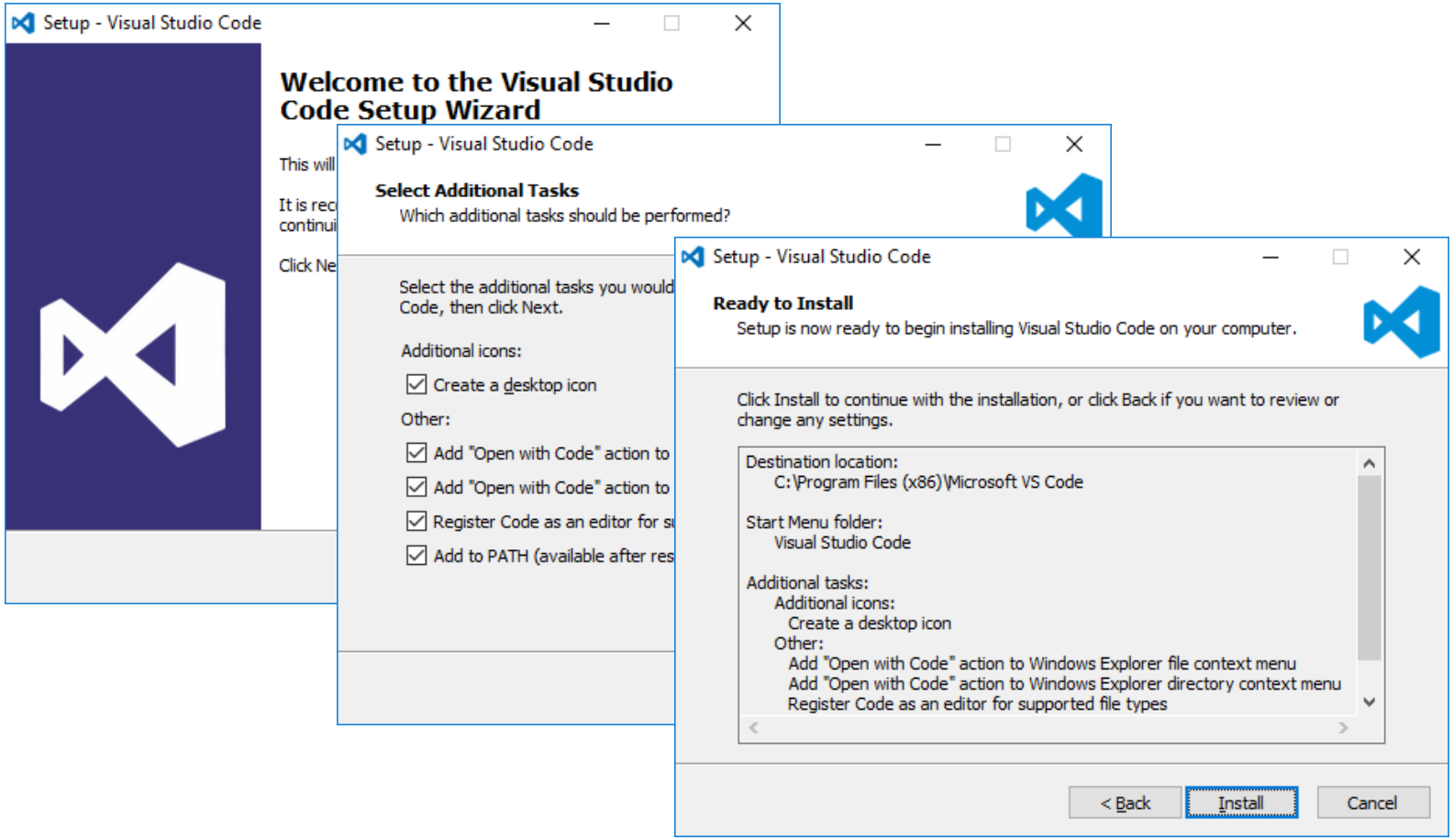
Installing node.js

- https://nodejs.org/en/download/



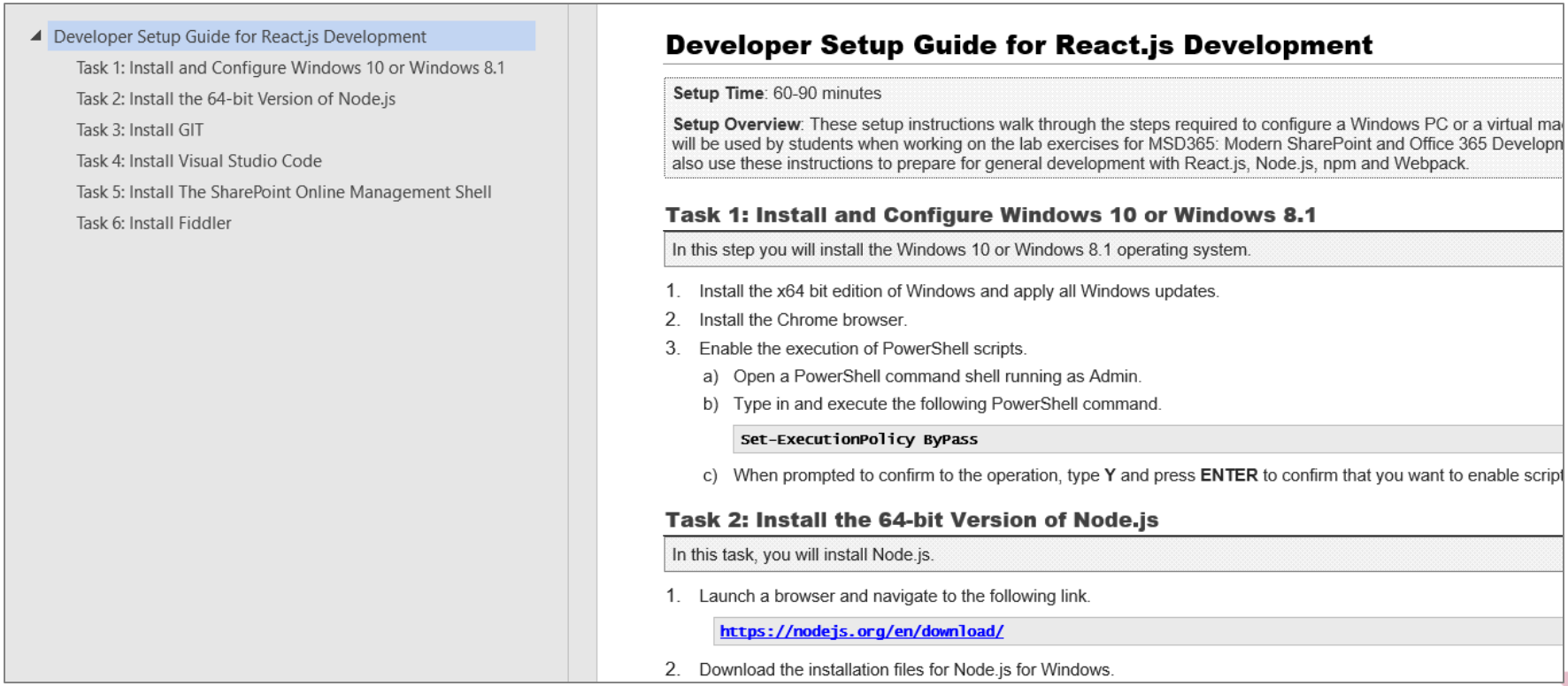
Install Visual Studio Code

- http://code.visualstudio.com/



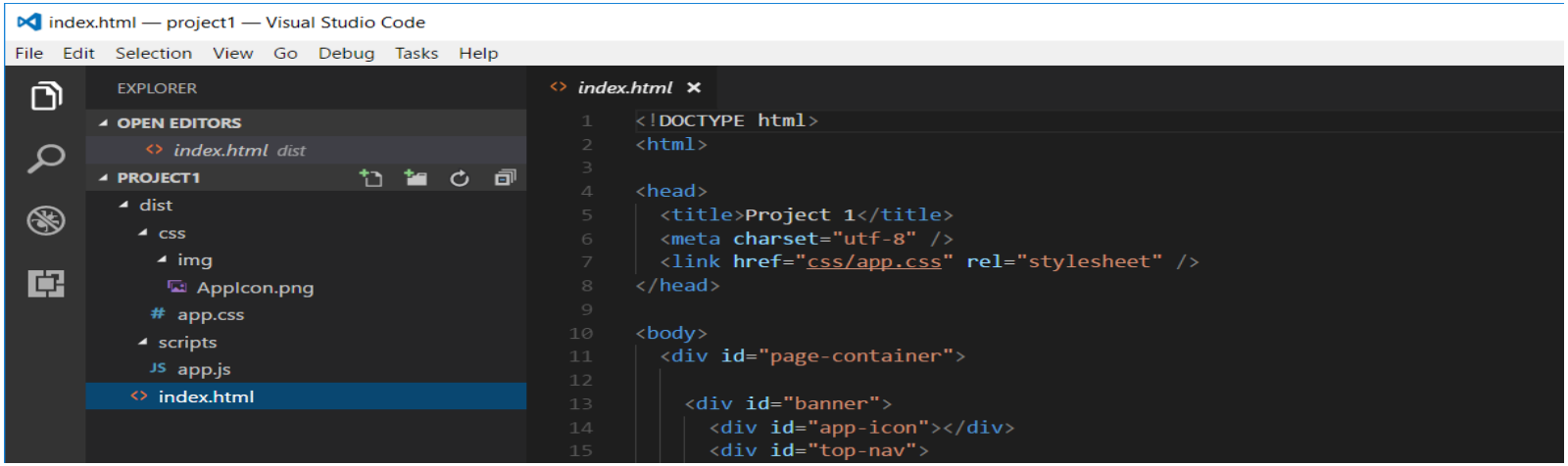
Modern React Developer Setup Guide

- Use the Critical Path setup guide to get started
  - Used to configure Windows PC for react development
  - https://github.com/CriticalPathTraining/React4SharePoint/raw/master/Setup.pdf



Developing with Visual Studio Code

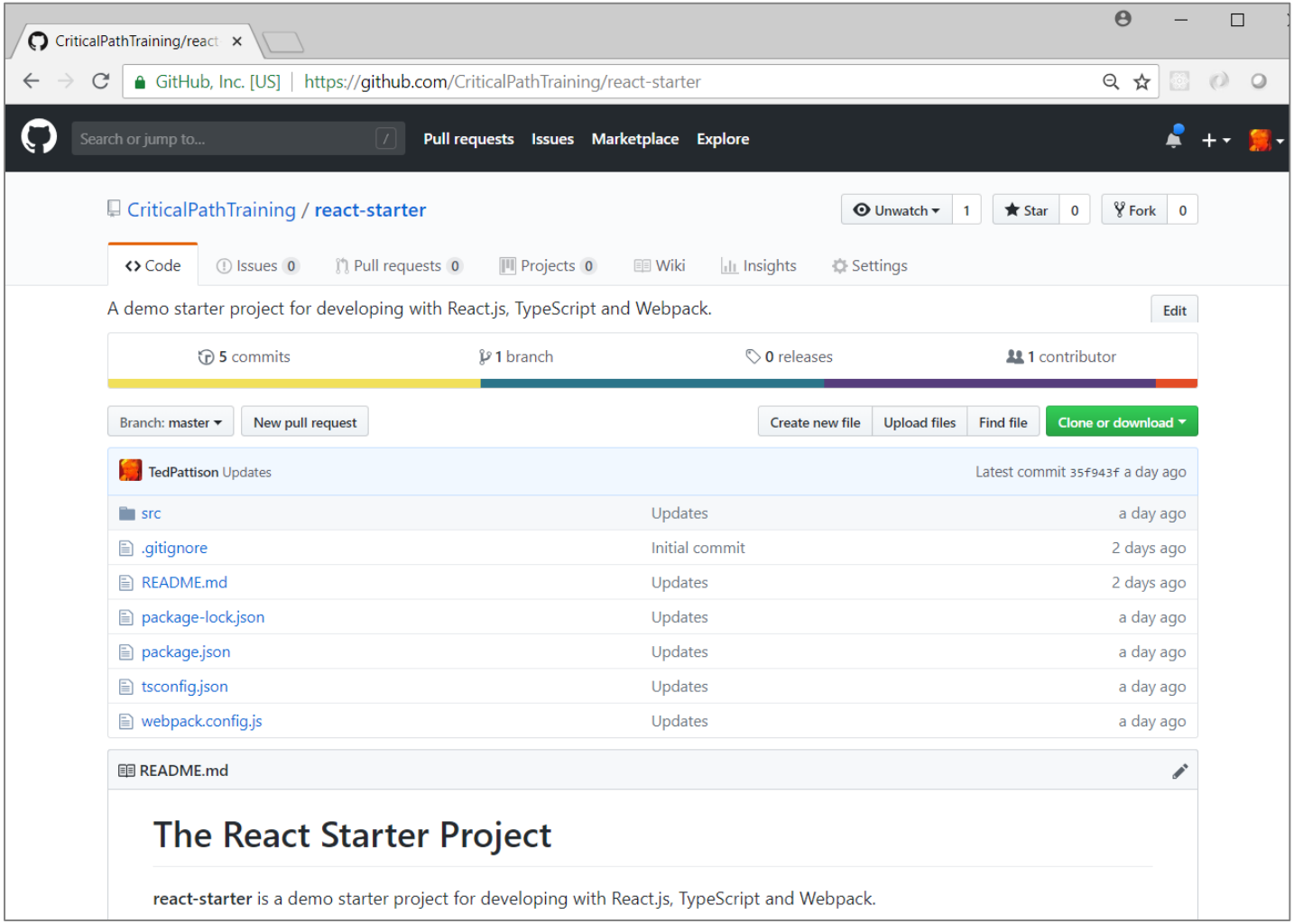
- Node.js is agnostic when it comes to developer IDE
  - There are many different IDEs that people use with Node.js
  - Visual Studio Code designed for Node.js-style development



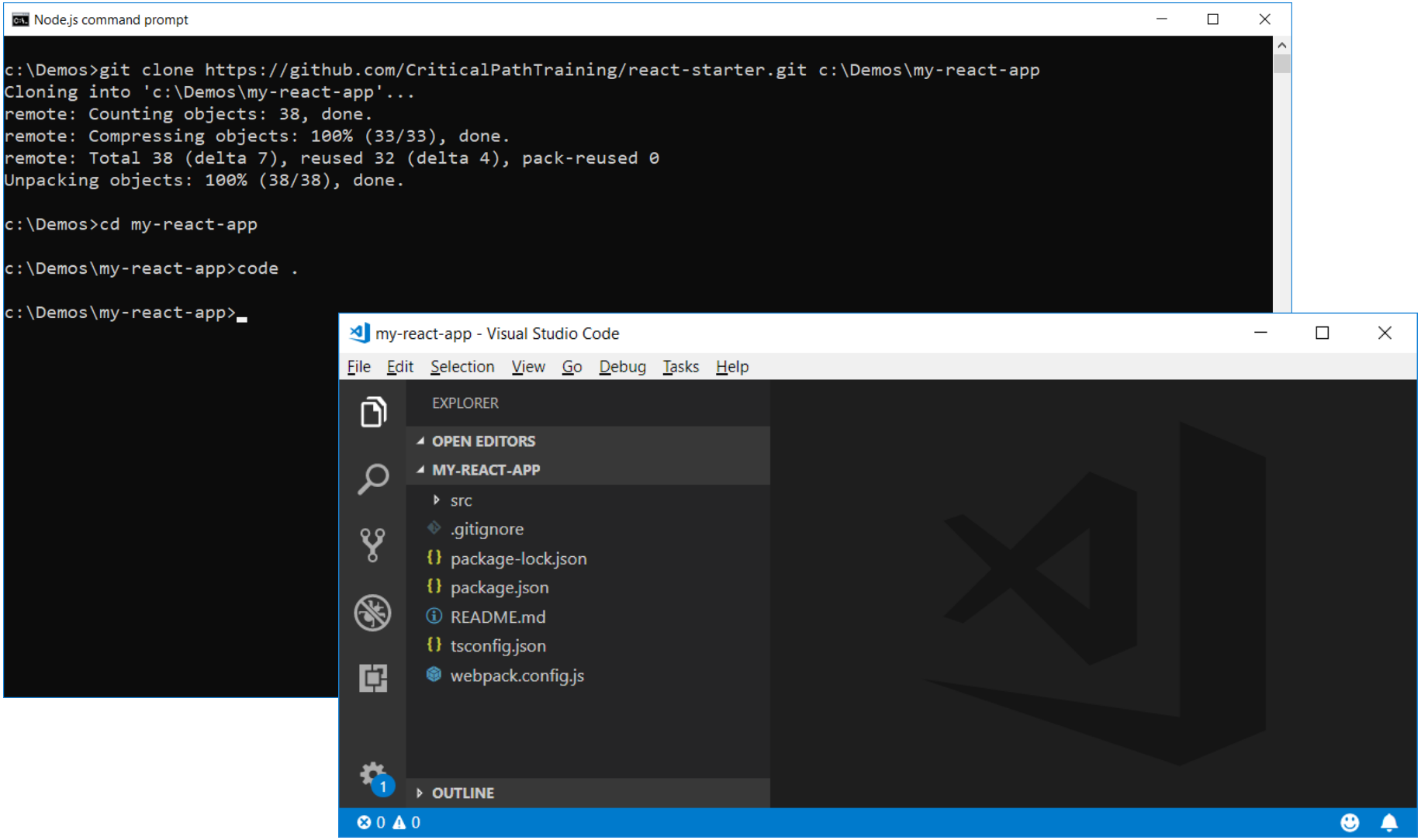
- Visual Studio is not a good fit for Node.js development
  - Visual Studio solution & project files incompatible with Node.js

The React Starter Project

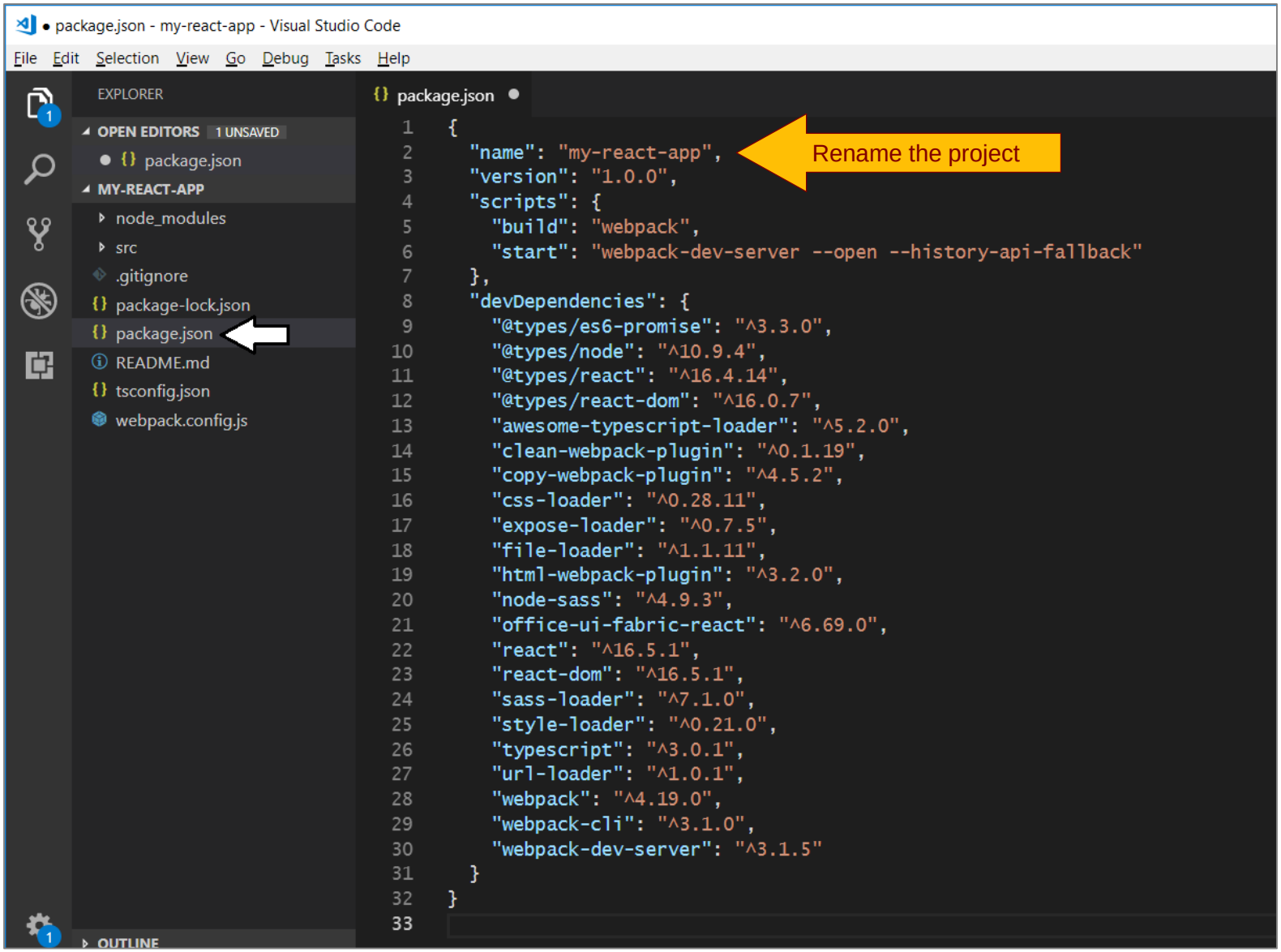
https://github.com/CriticalPathTraining/react-starter



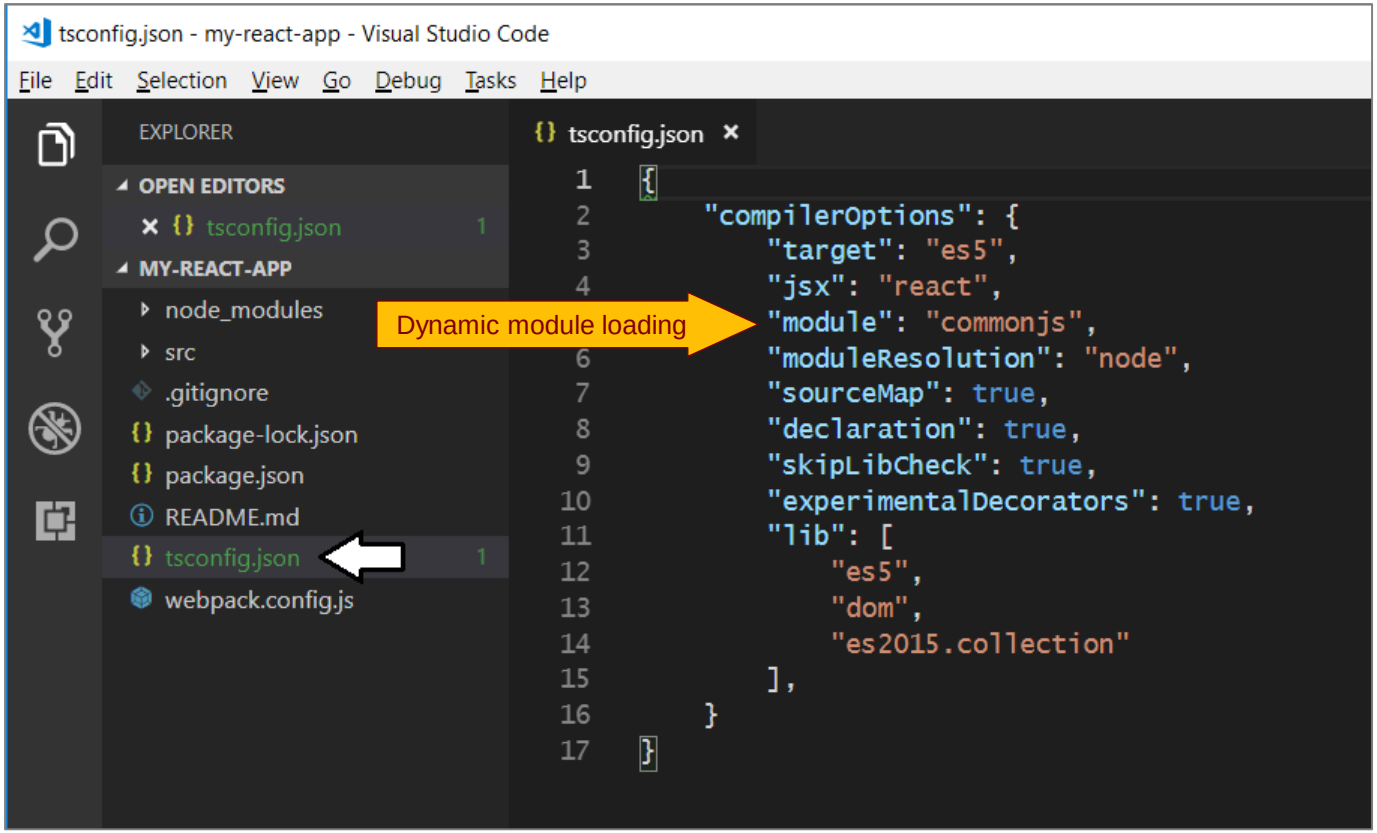
Cloning the Starter Project



Starter Project - package.json



Starter Project tsconfig.json



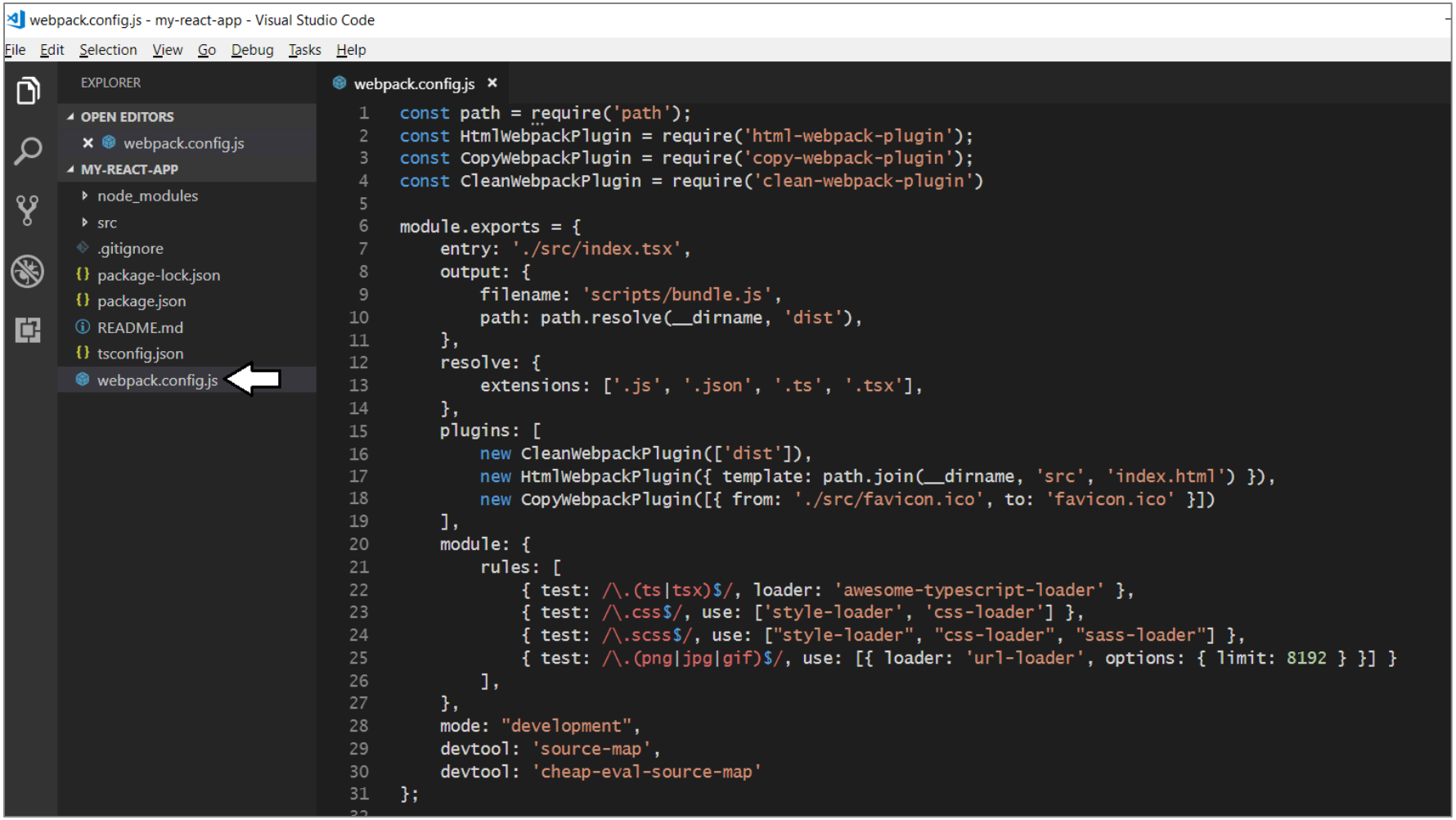
WebPack

- WebPack serves as a bundling utility
  - Bundles many js/ts files into a single file
  - Can handle dynamic module loading
  - Provides a dev server for testing and debugging

- When using Webpack 4
  - Install packages for webpack and webpack-cli

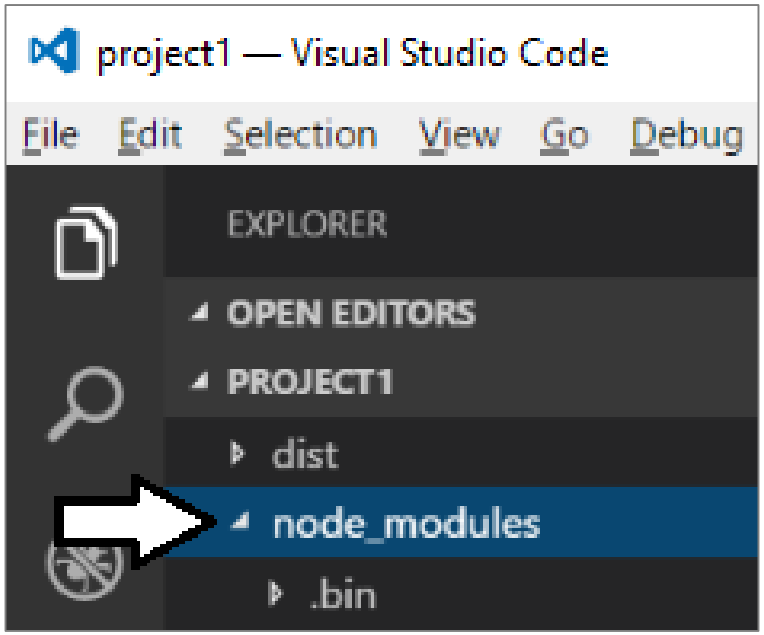
```
npm install webpack webpack-cli --save-dev
```

Starter Project - webpack.config.js

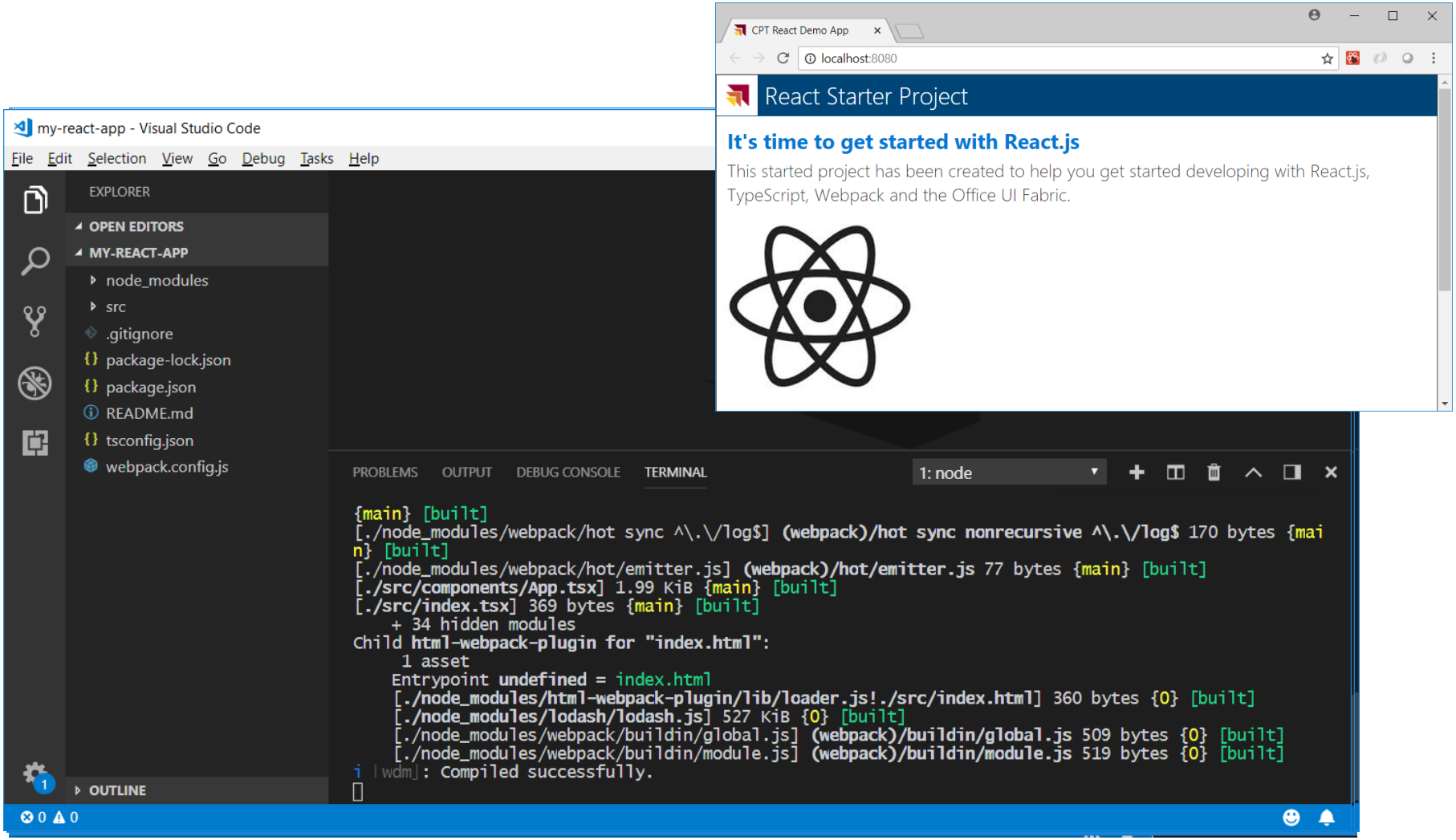


node_modules folder

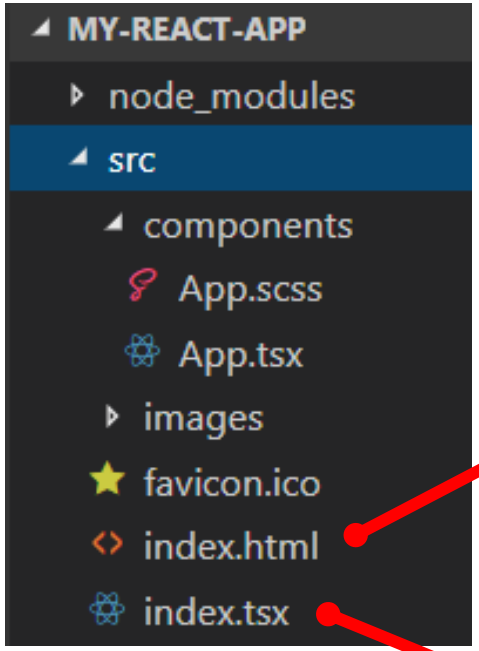
- Package files copied into **node_modules** folder
  - This folder often contain 100s of packages for a project
  - Contents of folder not saved into source control
  - Contents can be restored with **npm install** command



Executing npm Commands in Visual Studio Code



Starter Project Structure



index.html

```
<!DOCTYPE html>
<html>

<head>
  <title>CPT React Demo App</title>
  <meta charset="utf-8" />
</head>

<body>
  <div id="react-target" />
</body>

</html>
```

index.tsx

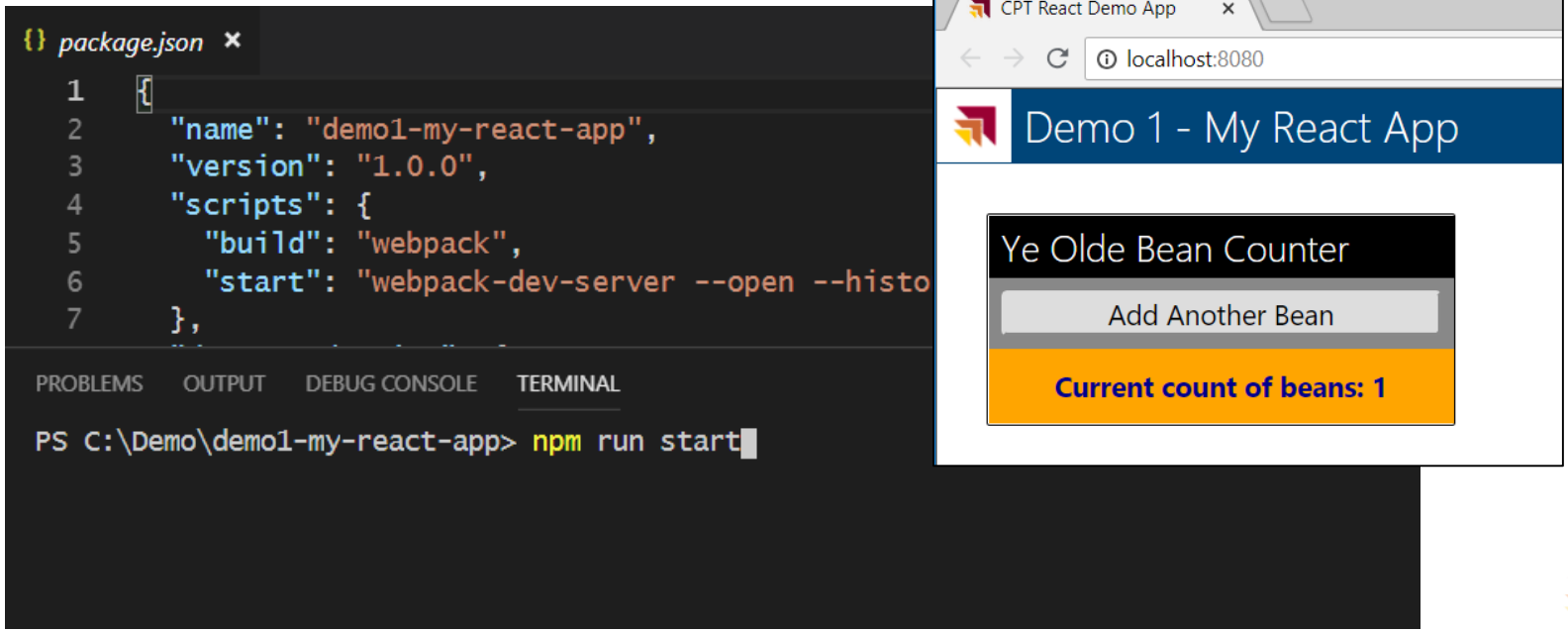
```
import * as React from 'react';
import { render } from 'react-dom';
import App from './components/App';

var topLevelAppComponent = <App />;
var target = document.getElementById('react-target');

render(topLevelAppComponent, target);
```

Webpack Dev Server

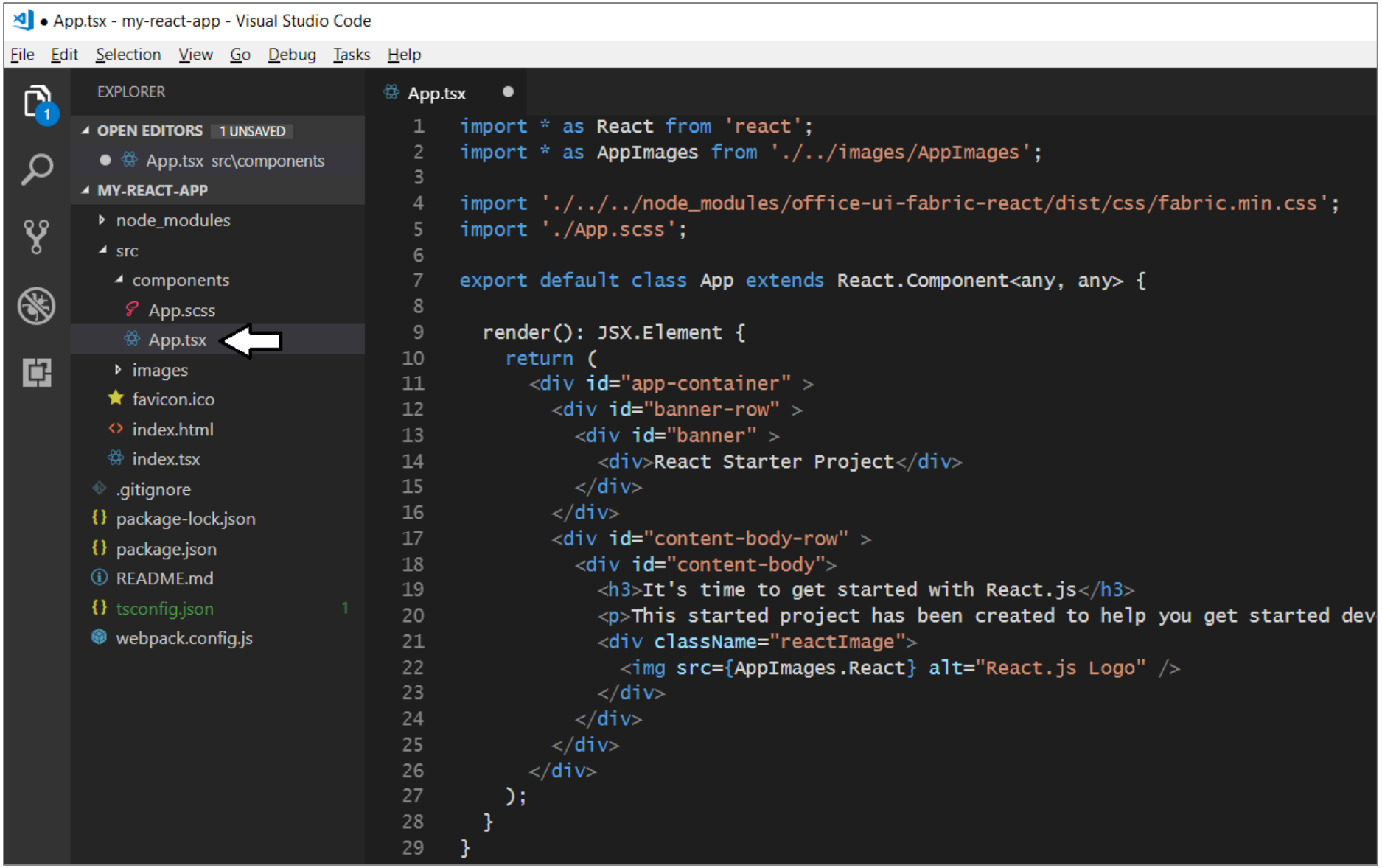
- Webpack provides its own development server
  - Install the webpack dev server package

    ```
    npm install webpack-dev-server --save-dev
    ```
  - Run your project using the webpack dev server CLI

    ```
    webpack-dev-server --open
    ```



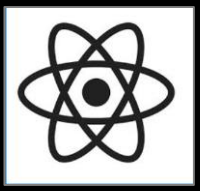
The Top-level App Component



Agenda

- ✓ Developing with Node.js, TypeScript and Webpack
- ➢ Learning React.js Fundamentals
- Using the Office UI Fabric React Component Library
- Developing React Webparts with SharePoint Framework
- Calling the Microsoft Graph API from React Webparts

Introducing React



- React is a library for building UI experiences
  - Not as all-encompassing as a framework like Angular
  - Focused on building HTML-based user experiences
  - Based on reusable component-based architecture
  - Components *react* to state changes by updating UI
  - React uses shadow DOM for efficient event handling
- React was originally designed for Facebook
  - Also a good fit for building SPFx web parts

Hello World with React.js and JavaScript

- Obtain the React library with npm or from a CDN
  - **npm install react --save**
  - **npm install react-dom --save**

SimpleReactApp.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Simple React App</title>
</head>
<body>

  <div id="app">
    <!-- this is where your app renders -->
  </div>

  <!-- React Libraries -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.5.4/react.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.5.4/react-dom.min.js"></script>

  <script>
    var reactComponenent = React.DOM.h1(null, "Hello, React!");
    var target = document.getElementById("app");
    ReactDOM.render(reactComponenent, target);
  </script>

</body>
</html>
```

React versus ReactDOM

- **React** and **ReactDOM** are separate libraries
  - **React (react.js)** is the primary library used to build out user experiences
  - **ReactDOM** (**react-dom.js**) is used to render **React** user experience in the browser
- **React** library exposes global **React** object
  - **React** object is the main entry point into React API
  - **React.DOM** wraps standard HTML elements
- **ReactDOM** library exposes global **ReactDOM** object
  - **ReactDOM** object used to render React components on web page

```
var reactComponenent = React.DOM.h1(null, "Hello, React!");

var target = document.getElementById("app");

ReactDOM.render(reactComponenent, target);
```

React Component Created Using ES5

- React component can be created using EcmaScript 5
  - React component definition created using **React.createClass**
  - React component must be defined with **render** method
  - React component can be instantiated with **React.createElement**

```
var myComponent = React.createClass({
  render: () => {
    return React.DOM.h1(null, "Hello React!")
  }
});

ReactDOM.render(
  React.createElement(myComponent),
  document.getElementById("app")
);
```

Defining React Components using TypeScript

- Component is class extending `React.Component`
  - Component usually defined in its own **tsx** file
  - Component class must define **render** method

my-component.tsx

```tsx
import * as React from 'react';

export class MyComponent extends React.Component<any, any> {
    render() {
        return <h2>Hello from my component</h2>;
    }
}
```

  - Component can be instantiated with JSX/TSX syntax

app.tsx

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

import { MyComponent } from "./components/my-component"

window.onload = () => {
  // Create and render component
  ReactDOM.render( <MyComponent/>, document.getElementById("app") );
}
```

Understanding JSX (and TSX)

- JSX provides better syntax for HTML composition
  - JSX allows extends JavaScript with XML-like syntax
  - JSX syntax must be transpiled into JavaScript code

```
var myHtml = <div id="myAppContainer" style={{ backgroundColor:"yellow", padding:8 }}>
               <h2>Hello JSX</h2>
               <p>I'm composing HTML elements using JSX syntax.</p>
             </div>;


ReactDOM.render( myHtml , document.getElementById("app") );
```

- JSX/TSX is separate from React library
  - JSX/TSX commonly used in React development
  - Babel compiler used to transpile JSX to JavaScript
  - TypeScript compiler used to transpile TSX to JavaScript

Component Properties and State

- Component can contain properties and state
  - Properties are initialized by external components
  - Properties are read-only to hosting component
  - State is set internally by hosting component
  - Changing state triggers UI refresh by calling render
  - UI experience created by ***reacting*** to changes in state

Designing with Properties and State

```tsx
export interface IBeanCounterProps {
  startingCount: number;
}
```

```tsx
export interface IBeanCounterState {
  count: number;
}
```

```tsx
export default class BeanCounter extends React.Component<IBeanCounterProps, IBeanCounterState> {

  public state = {
    count: this.props.startingCount
  }

  private addNewBean() {···
  }

  public render(): JSX.Element {···
  }

}
```

Component Rendering using State

- **render** method implemented using state
  - **render** method executes whenever state is updated

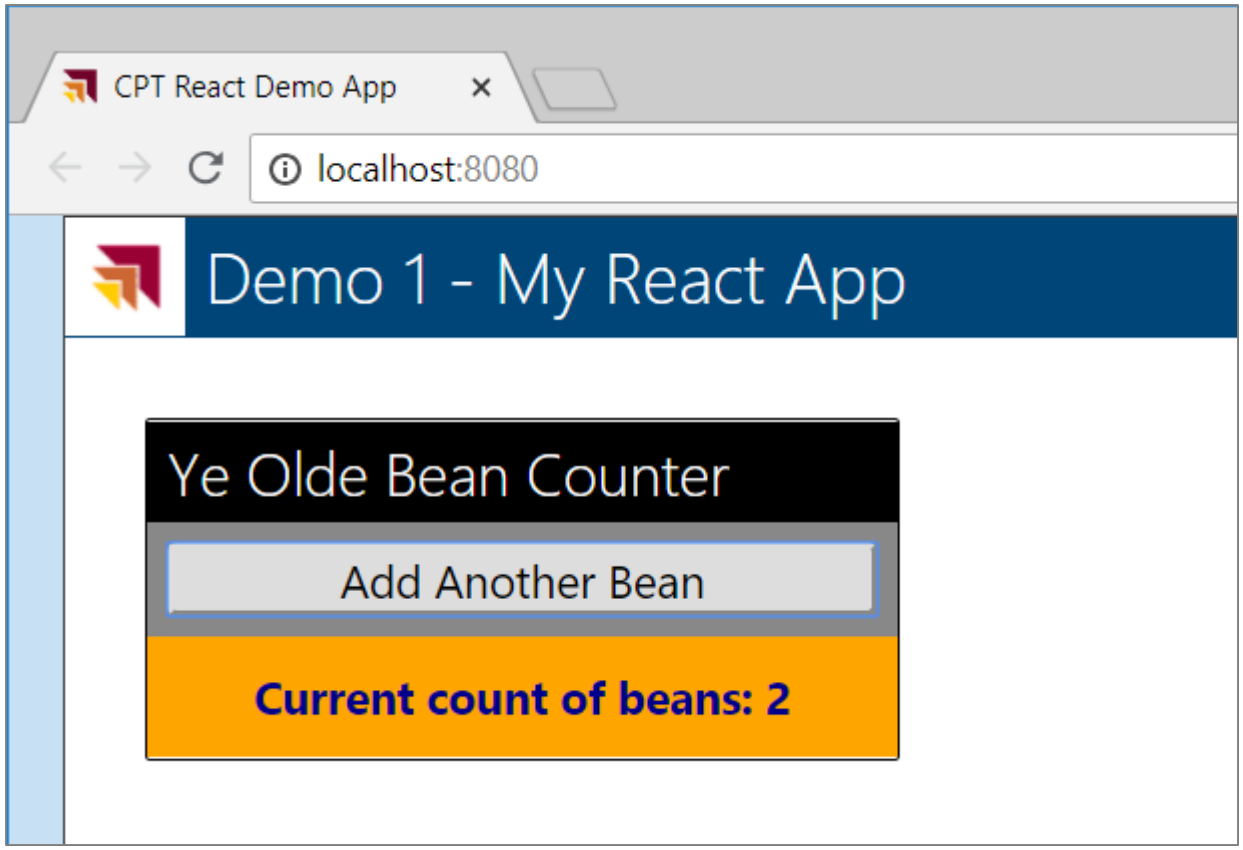
```
public render(): JSX.Element {
  return (
    <div id="bean-counter" >
      <div className="title">
        Ye Olde Bean Counter
    </div>
      <div className="toolbar">···
      <div className="display">
        Current count of beans: {this.state.count}
      </div>
    </div>
  );
}
```

React Provides Synthetic Events

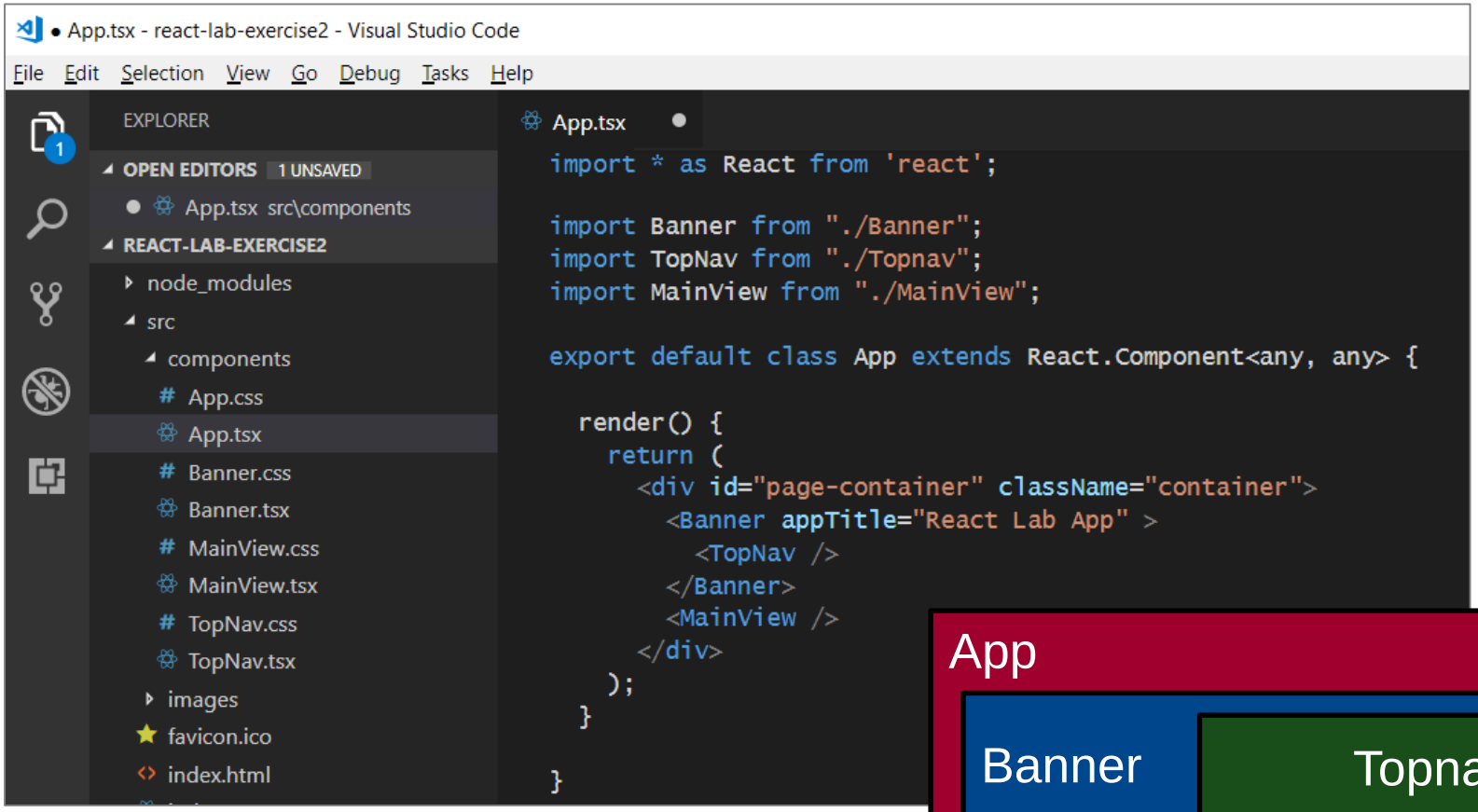
- Replaces standard DOM-based event handling
  - React creates virtual DOM for elements created by component
  - Provides high-performant code in busy web pages

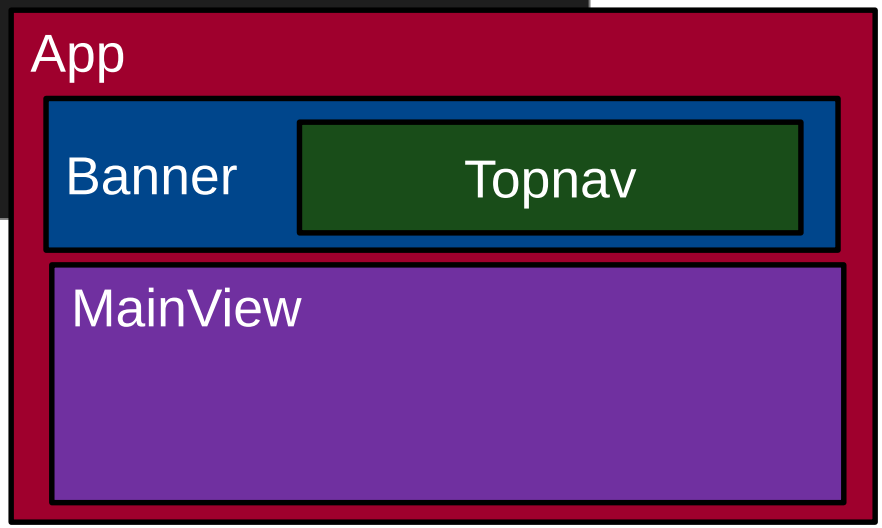
```
private addNewBean() {
  let newCount = (this.state.count + 1);
  this.setState({ count: newCount })
}

public render(): JSX.Element {
  return (
    <div id="bean-counter" >
      <div className="title">
        Ye Olde Bean Counter
    </div>
      <div className="toolbar">
        <button onClick={ (event): void => { this.addNewBean(); }} >Add Another Bean</button>
      </div>
      <div className="display">
        Current count of beans: {this.state.count}
      </div>
    </div>
  );
}
```

Demo 1 - Creating a Simple React Component

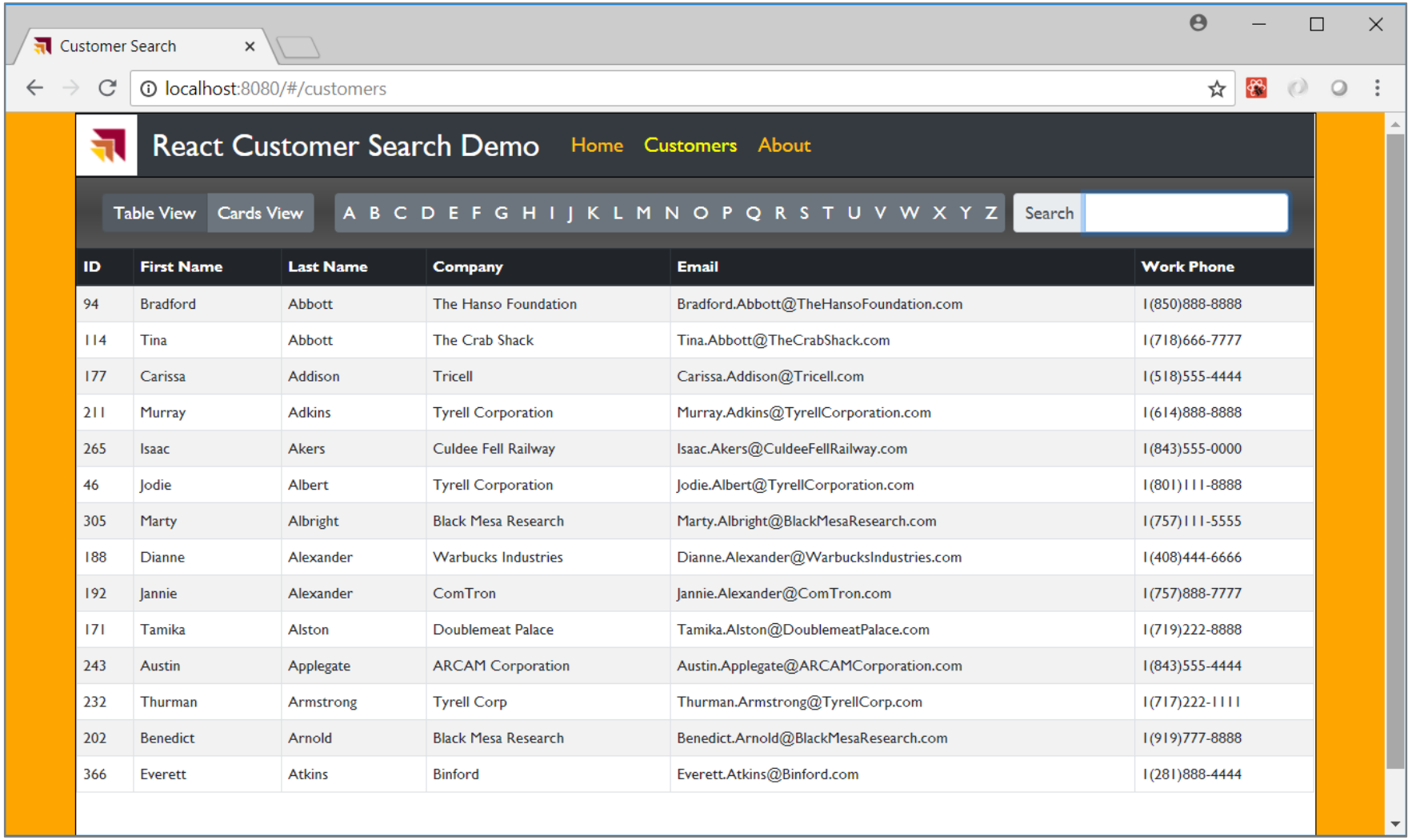


React Component Hierarchies





Demo 2 - Customer Search



ID	First Name	Last Name	Company	Email	Work Phone
94	Bradford	Abbott	The Hanso Foundation	Bradford.Abbott@TheHansoFoundation.com	1(850)888-8888
114	Tina	Abbott	The Crab Shack	Tina.Abbott@TheCrabShack.com	1(718)666-7777
177	Carissa	Addison	Tricell	Carissa.Addison@Tricell.com	1(518)555-4444
211	Murray	Adkins	Tyrell Corporation	Murray.Adkins@TyrellCorporation.com	1(614)888-8888
265	Isaac	Akers	Culdee Fell Railway	Isaac.Akers@CuldeeFellRailway.com	1(843)555-0000
46	Jodie	Albert	Tyrell Corporation	Jodie.Albert@TyrellCorporation.com	1(801)111-8888
305	Marty	Albright	Black Mesa Research	Marty.Albright@BlackMesaResearch.com	1(757)111-5555
188	Dianne	Alexander	Warbucks Industries	Dianne.Alexander@WarbucksIndustries.com	1(408)444-6666
192	Jannie	Alexander	ComTron	Jannie.Alexander@ComTron.com	1(757)888-7777
171	Tamika	Alston	Doublemeat Palace	Tamika.Alston@DoublemeatPalace.com	1(719)222-8888
243	Austin	Applegate	ARCAM Corporation	Austin.Applegate@ARCAMCorporation.com	1(843)555-4444
232	Thurman	Armstrong	Tyrell Corp	Thurman.Armstrong@TyrellCorp.com	1(717)222-1111
202	Benedict	Arnold	Black Mesa Research	Benedict.Arnold@BlackMesaResearch.com	1(919)777-8888
366	Everett	Atkins	Binford	Everett.Atkins@Binford.com	1(281)888-4444

React Router

- Used to create route map in single page application (SPA)
  - Installed as a pair of npm packages

```
npm install react-router @types/react-router --save-dev
npm install react-router-dom @types/react-router-dom --save-dev
```

- Router must be added in as top-level component above App

index.tsx

```tsx
import * as React from 'react';
import { render } from 'react-dom';
import App from './components/App';
import { HashRouter } from 'react-router-dom';

var topLevelAppComponent =
  <HashRouter>
    <App />
  </HashRouter>;

var target = document.getElementById('react-target');

render(topLevelAppComponent, target);
```

Using React Router

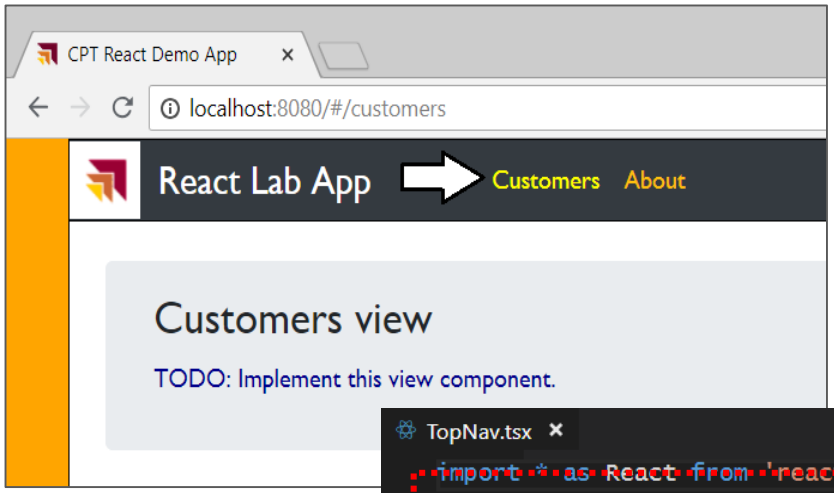
- Import Route and Switch components

```
import * as React from 'react';
import { Route, Switch } from 'react-router-dom';
```

- Create route map in HTML output

```
export default class App extends React.Component<any, any> {

  render() {

    return (
      <div id="page-container" className="container">
        <Banner appTitle="React Lab App" >
          <TopNav />
        </Banner>
        <Switch>
          <Route path="/" exact component={ViewHome} />
          <Route path="/customers" component={ViewCustomers} />
          <Route path="/about" component={ViewAbout} />
        </Switch>
      </div>
    );
```

Creating Route Links



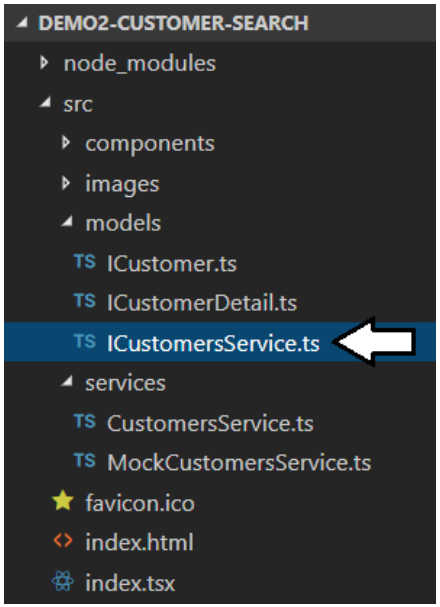
TopNav.tsx

```tsx
import * as React from 'react';
import { Link, NavLink } from 'react-router-dom';

import "./TopNav.css";

export default class TopNav extends React.Component<any, any> {

  render() {
    return (
      <div id="top-nav" className="navbar-collapse collapse" >
        <nav>
          <ul className="nav navbar-nav" >
            <li className="nav-item" >
              <NavLink exact to="/" className="navbar-link" activeClassName="active-nav-link" >
                Home
              </NavLink>
            </li>
            <li className="nav-item" >
              <NavLink to="/customers" className="navbar-link" activeClassName="active-nav-link" >
                Customers
              </NavLink>
            </li>
```

React Component Lifecycle Methods

- **componentWillUpdate**
  - executed before component is rendered
- **componentDidUpdate**
  - executed after component is rendered
- **componentWillMount**
  - executed before node is added to the DOM
- **componentDidMount**
  - executed after node is added to the DOM
- **componentWillUnmount**
  - executed before node is removed from the DOM
- **shouldComponentUpdate(newProps, newState)**
  - executed before component is updated

Defining Interfaces for Data Access Code



ICustomer.ts

```typescript
export default interface ICustomer {
  CustomerId: string;
  FirstName: string;
  LastName: string;
  Company: string;
  EmailAddress: string;
  WorkPhone: string;
  HomePhone: string;
}
```

ICustomersService.ts

```typescript
import ICustomer from "./ICustomer"
import ICustomerDetail from "./ICustomerDetail";

export default interface ICustomersService {
  getCustomers(): Promise<ICustomer[]>;
  getCustomersByLastName(lastNameSearch: string): Promise<ICustomer[]>;
  getCustomer(customerId: string): Promise<ICustomerDetail>;
}
```

Calling a Web Service using the Fetch API

```
getCustomers(): Promise<ICustomer[]> {
  const restUrl =
    "http://subliminalsystems.com/api/Customers/?" +
    "$select=CustomerId,LastName,FirstName,EmailAddress,WorkPhone,HomePhone,Company" +
    "&$filter=(CustomerId+le+12)&$top=200";
  return fetch(restUrl)
    .then(response => response.json())
    .then(response => {
      console.log(response.value);
      return response.value;
    });
}
```

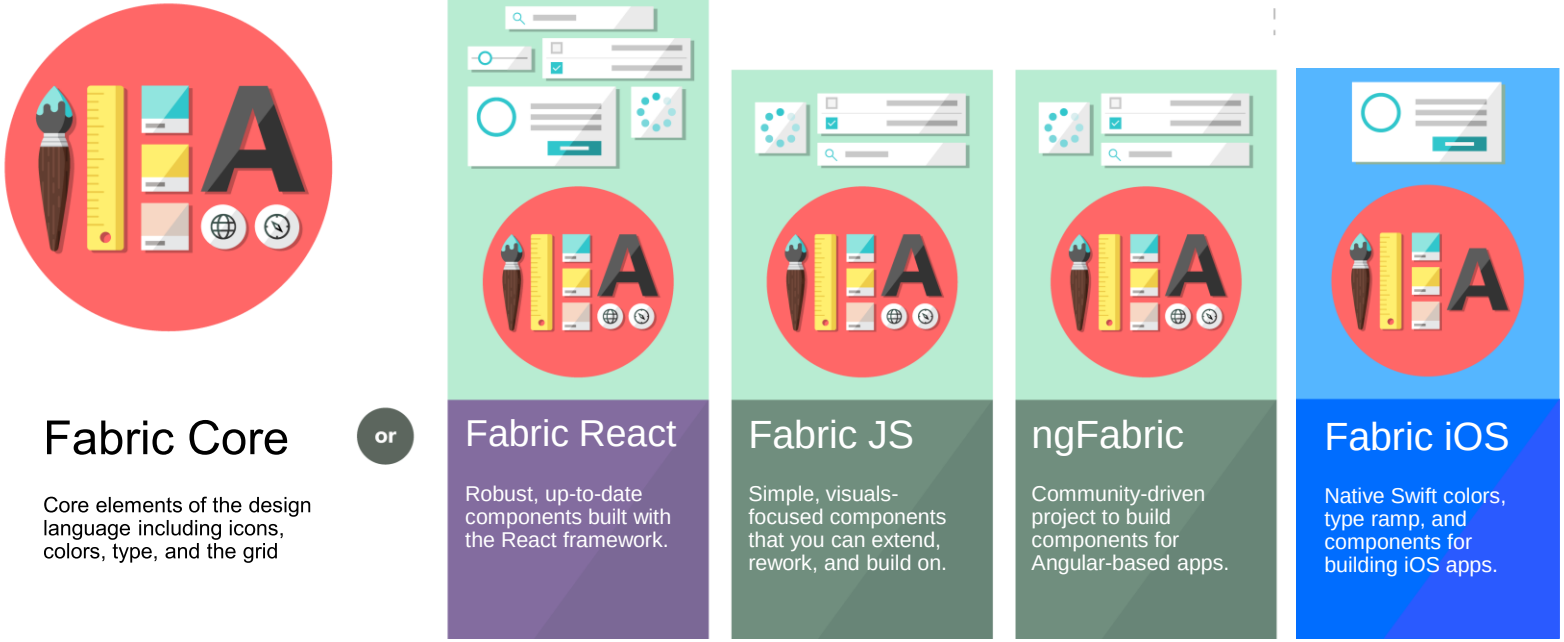
```
getCustomer(customerId: string): Promise<ICustomerDetail> {
  const restUrl = "http://subliminalsystems.com/api/Customers(" + customerId + ")";
  return fetch(restUrl)
    .then(response => response.json())
    .then(response => {
      console.log(response);
      return response;
    });
}
```

Agenda

- ✓ Developing with Node.js, TypeScript and Webpack
- ✓ Learning React.js Fundamentals
- ➢ Using the Office UI Fabric React Component Library
- • Developing React Webparts with SharePoint Framework
- • Calling the Microsoft Graph API from React Webparts

What is the Office UI Fabric?

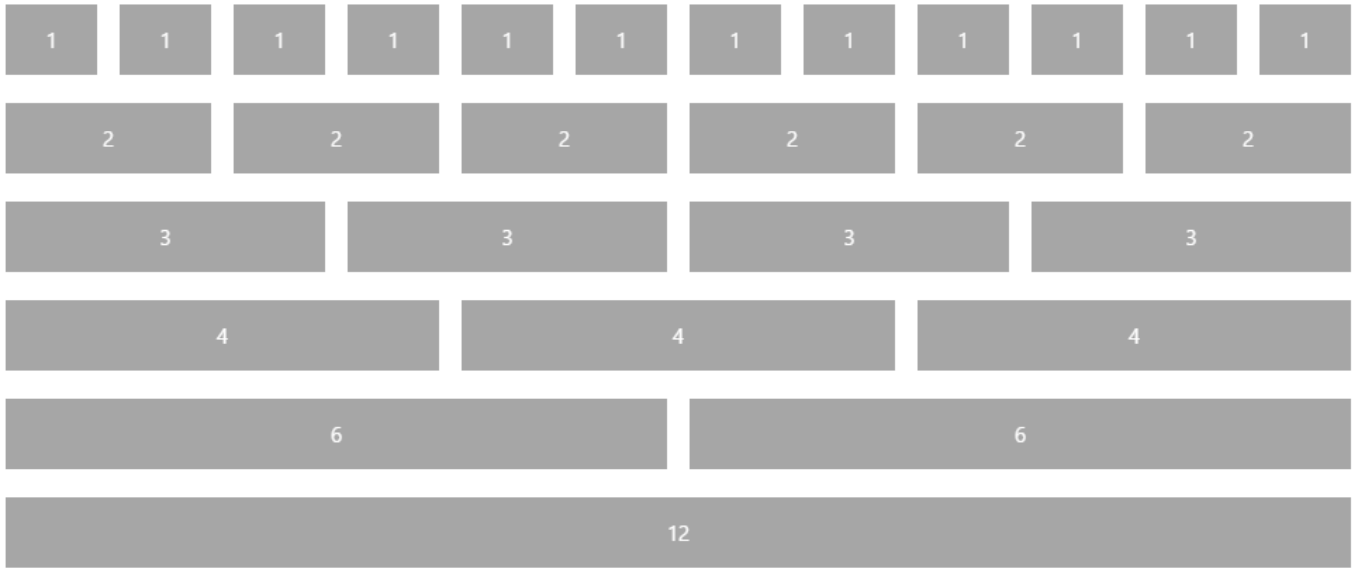
- Office UI Fabric is responsive, mobile-first, front-end style framework
  - Built by Microsoft to style Office 365, OneDrive and SharePoint sites
  - All about styling instead of JavaScript
  - Can be used by 3rd party developers



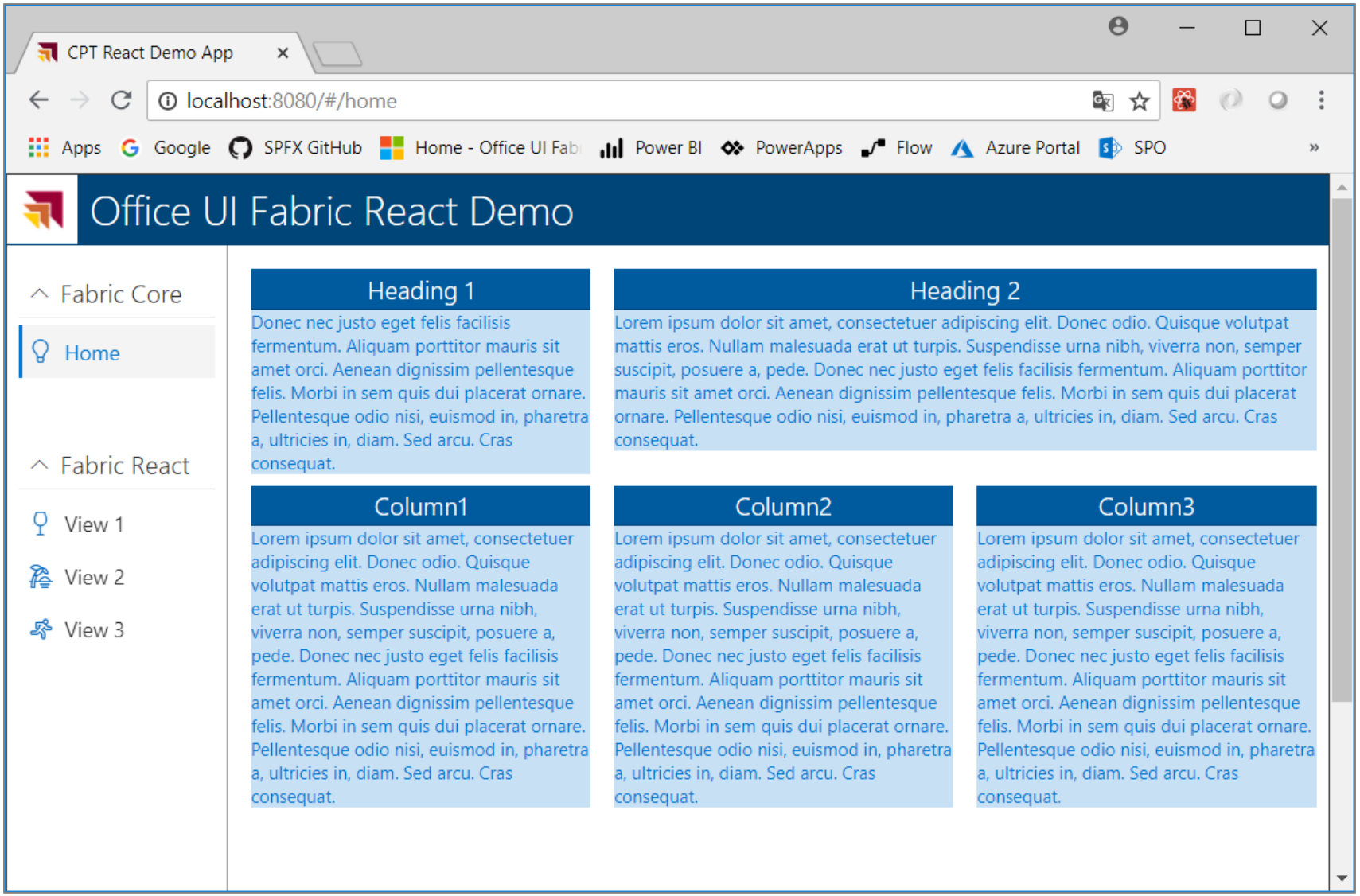
Responsive Grid

- Fabric comes with a mobile-first, responsive grid
  - Based on 12 column grid
  - Used to create flexible layouts

```
<div class="ms-Grid">
    <div class="ms-Grid-row">
        <div class="ms-Grid-col ms-u-sm6 ms-u-md4 ms-u-lg2">First</div>
        <div class="ms-Grid-col ms-u-sm6 ms-u-md8 ms-u-lg10">Second</div>
    </div>
</div>
```

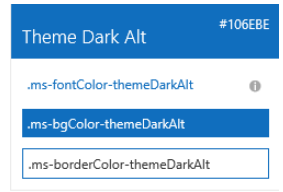


Demo 3 - Office UI Fabric



Fabric Core styling

- Fonts and typography
  - Segoe font family + type ramp
  - Official Office 365 iconography
- Color
  - Official Office 365 color palette
- Branded assets
  - Product symbols + product filetype symbols
- Animations
  - Official Office 365 selection of easings and animations
- Responsive grid
  - Tailored to Office 365 silhouettes



Typography

- Base font classes
  - Fabric includes 10 base font classes
  - Each base class sets a default size, weight, and color.

Class	Size	Weight	Color
.ms-font-su	42px	Segoe UI Light	ms-color-neutralPrimary
.ms-font-xxl	28px	Segoe UI Light	ms-color-neutralPrimary
.ms-font-xl	21px	Segoe UI Light	ms-color-neutralPrimary
.ms-font-l	17px	Segoe UI Semilight	ms-color-neutralPrimary

Typography

- Helper font classes
  - There are helper font classes to change the text weight.

Class	Weight
.ms-fontWeight-light	Light
.ms-fontWeight-semilight	Semilight
.ms-fontWeight-regular	Regular
.ms-fontWeight-semibold	Semi Bold

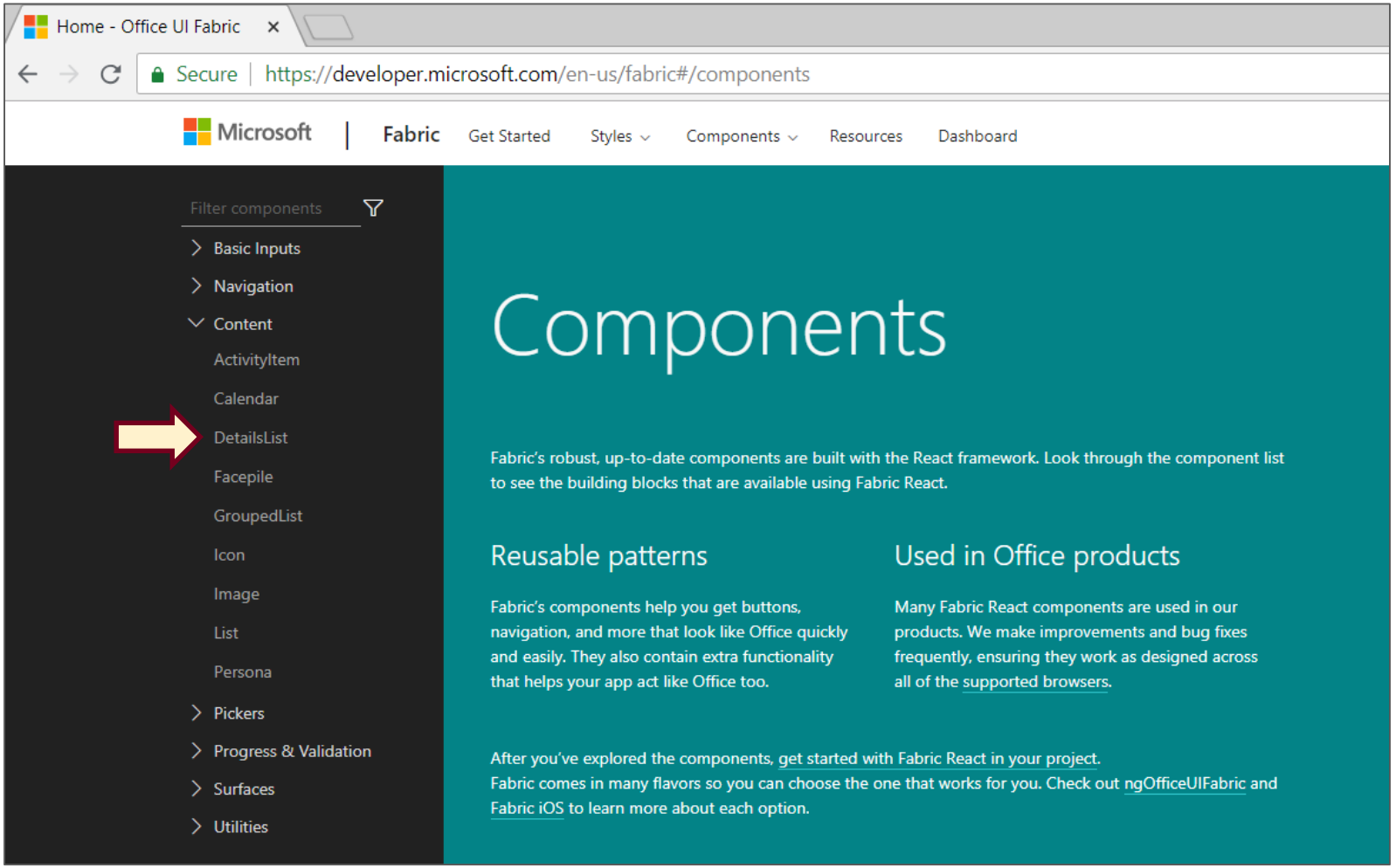
Putting Office UI Fabric Styles to Use

App.scss

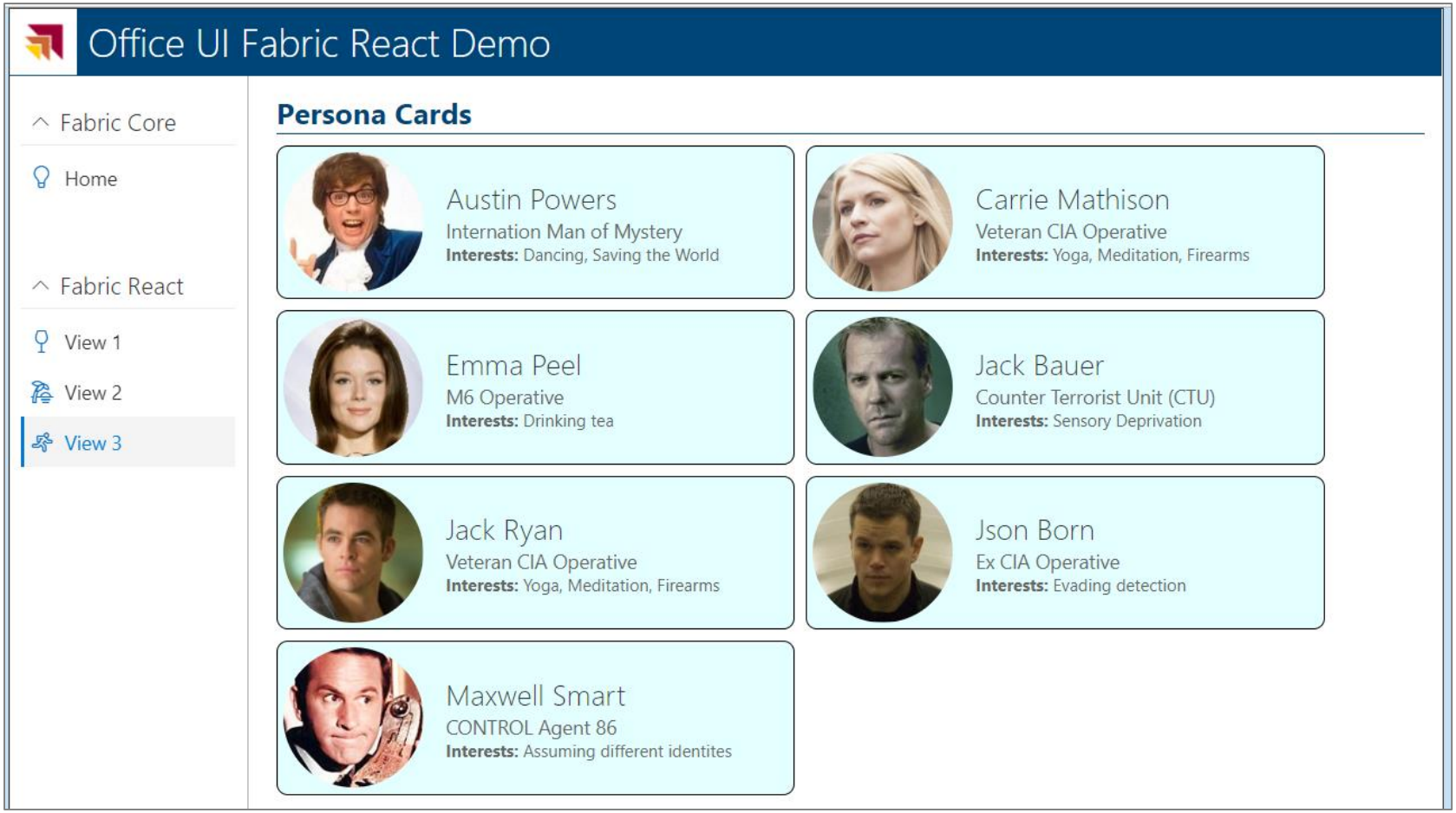
```scss
@import '~office-ui-fabric-react/dist/sass/_References.scss';

body {
  margin: 0px;
  padding: 0px;
  background-color: $ms-color-themeLight;
}

#app-container{
  @include ms-Fabric;
  @include ms-Grid;
  @include ms-borderColor-neutralPrimaryAlt;
  background-color: white;
  min-height: 600px;
  border-bottom-left-radius: 8px;
  border-bottom-right-radius: 8px;
  border: 1px solid ;
  max-width: 1024px;
  margin: auto;
}
```

Office UI Fabric React Component Library

- https://developer.microsoft.com/en-us/fabric#/components



Using Persona Cards



Adding a Persona Component

```
import {
  Persona,
  PersonaSize
} from 'office-ui-fabric-react/lib/components/Persona';
```

```
const teamMembers = [
  { key: 'AustinP', name: 'Austin Powers', title: 'Internation Man of Mystery', interests: 'Dancing, Saving
  { key: 'CarrieP', name: 'Carrie Mathison', title: 'Veteran CIA Operative', interests: 'Yoga, Meditation,
  { key: 'EmmaP', name: 'Emma Peel', title: 'M6 Operative', interests: 'Drinking tea', photo: AppImages.Emm
  { key: 'JackB', name: 'Jack Bauer', title: 'Counter Terrorist Unit (CTU)',interests: 'Sensory Deprivation
  { key: 'JackR', name: 'Jack Ryan', title: 'Veteran CIA Operative', interests: 'Yoga, Meditation, Firearms
  { key: 'JasonB', name: 'Json Born', title: 'Ex CIA Operative', interests: 'Evading detection', photo: App
  { key: 'MaxwellS', name: 'Maxwell Smart', title: 'CONTROL Agent 86', interests: 'Assuming different ident
];
```

```
render() {
  return (
    <div id="view3">
      <h4>Persona Cards</h4>
      <div>{teamMembers.map( (teamMember: any) => (
        <div className='persona-card'>
          <Persona
            key={teamMember.key}
            primaryText={teamMember.name}
            imageUrl={teamMember.photo}
            size={PersonaSize.size100}
            secondaryText={teamMember.title}
            onRenderTertiaryText={() => (
              <div><strong>Interests: </strong>{teamMember.interests}</div>
            )}
          />
        </div>
      ))}
      </div>
    </div>
  );
}
```

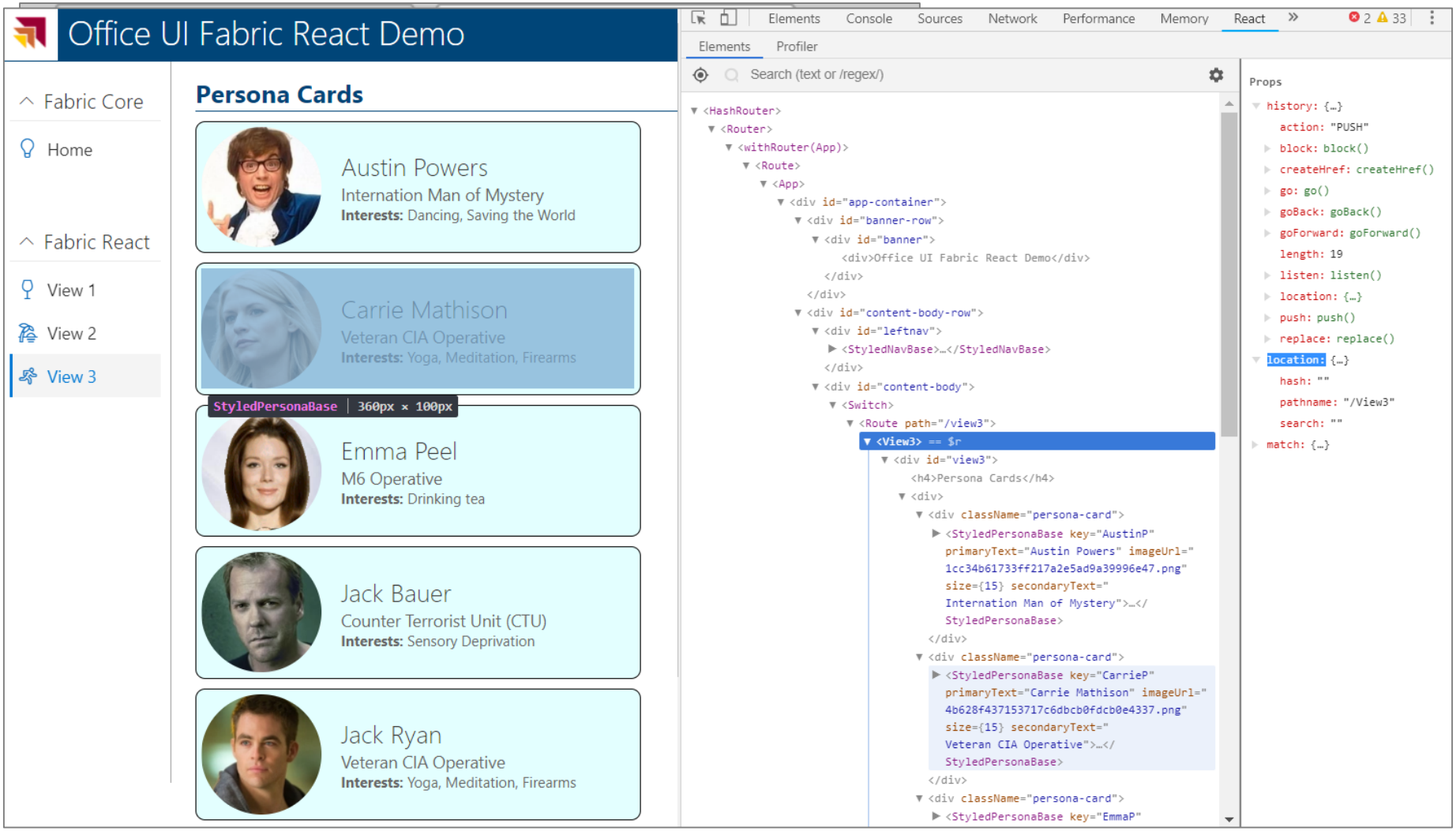
Using the DetailsList Component

```
import {
  DetailsList,
  IColumn,
  DetailsListLayoutMode
} from 'office-ui-fabric-react';
```

```
const leadColumns: IColumn[] = [
  { key: 'id', fieldName: 'id', name: 'ID', minWidth: 12, maxWidth: 24 },
  { key: 'firstName', fieldName: 'firstName', name: 'First Name', minWidth: 24, maxWidth: 64 },
  { key: 'lastName', fieldName: 'lastName', name: 'Last Name', minWidth: 24, maxWidth: 64 },
  { key: 'company', fieldName: 'company', name: 'Company', minWidth: 64, maxWidth: 120 },
  { key: 'emailAddress', fieldName: 'emailAddress', name: 'Email', minWidth: 100, maxWidth: 240 }
];
```

```
public render(): React.ReactElement<ILeadTrackerProps> {
  return (

    <div className={styles.leadTracker}>

      <DetailsList
        items={this.state.leads}
        columns={leadColumns}
        setKey='set'
        layoutMode={DetailsListLayoutMode.fixedColumns}
      />

    </div>
  );
}
```

React Developer Tools - Chrome Extension

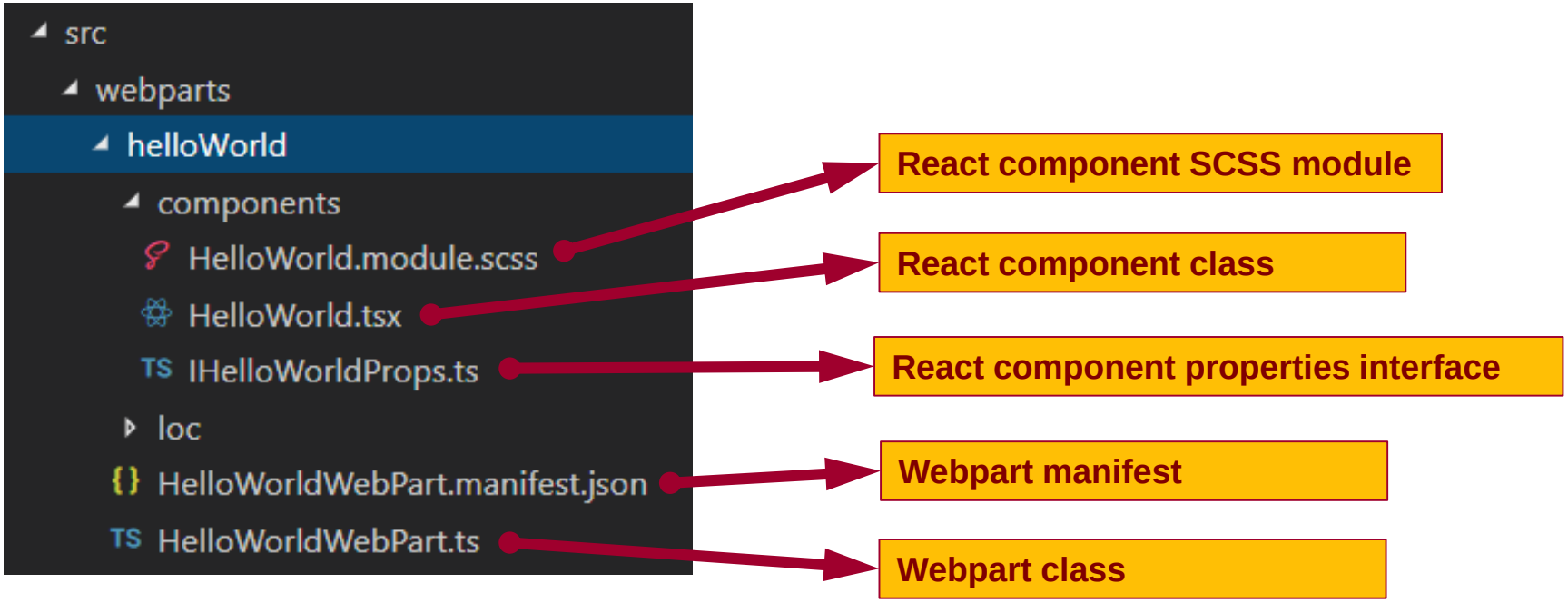


Agenda

- ✓ Developing with Node.js, TypeScript and Webpack
- ✓ Learning React.js Fundamentals
- ✓ Using the Office UI Fabric React Component Library
- ➢ Developing React Webparts with SharePoint Framework
- • Calling the Microsoft Graph API from React Webparts

Creating a React Webpart

- You can select React as framework for your webpart
  - You can create a React webpart when creating new project
  - You can add React webpart to existing project
  - React webpart made up of several different source files



React Webpart Styling

HelloWorld.module.scss

```scss
.helloWorld {
  background-color: lightsalmon;
  border: 4px solid purple;
  border-radius: 12px;

  .title {
    padding: 8px;
    font-size: 48px;
  }

}
```

HelloWorld.tsx

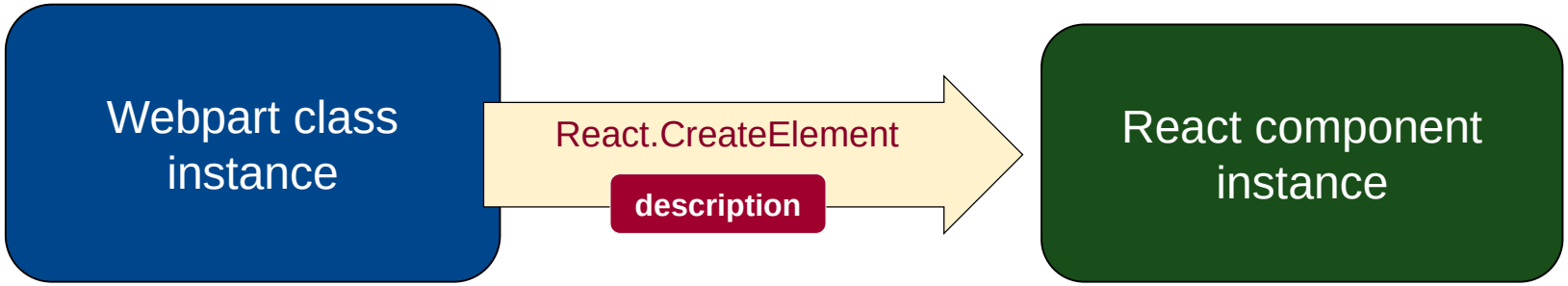
```tsx
import * as React from 'react';

import { IHelloWorldProps } from './IHelloWorldProps';

import styles from './HelloWorld.module.scss';

export default class HelloWorld extends React.Component<IHelloWorldProps, {}> {

  public render(): React.ReactElement<IHelloWorldProps> {
    return (
      <div className={styles.helloWorld}>
        <div className={styles.title}>
          {this.props.description}
        </div>
      </div>
    );
  }

}
```

React Webpart Architecture

```
export default class HelloWorldWebPart extends BaseClientSideWebPart<IHelloWorldWebPartProps> {

  public render(): void {
    const element: React.ReactElement<IHelloWorldProps > = React.createElement(
      HelloWorld, { description: this.properties.description }
    );
    ReactDom.render(element, this.domElement);
  }
```

```
export interface IHelloWorldProps {
  description: string;
}
```

```
import * as React from 'react';

import { IHelloWorldProps } from './IHelloWorldProps';

export default class HelloWorld extends React.Component<IHelloWorldProps, {}> {

  public render(): React.ReactElement<IHelloWorldProps> {
    return <div>{this.props.description}</div>;
  }

}
```



Webpart Persistent Properties

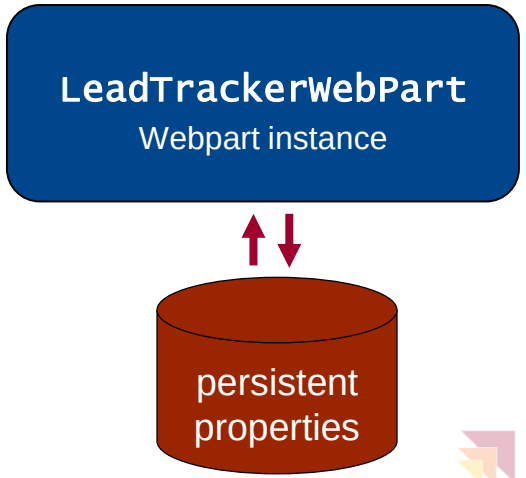
- Persistent properties defined in webpart using interface

```
export interface ILeadTrackerWebPartProps {
  targetList: string;
}

export default class LeadTrackerWebPart extends BaseClientSideWebPart<ILeadTrackerWebPartProps> {

  private MyMethod() {
    let list: string  = this.properties.targetList;
  }
```

- Property default values add to webpart manifest

LeadTrackerWebPart.manifest.json

```
  "preconfiguredEntries": [
    {
      "groupId": "5c03119e-3074-46fd-976b-c60198311f70",
      "group": { "default": "Other" },
      "title": { "default": "Lead Tracker" },
      "description": { "default": "a React webpart for tracking leads in SharePoint"},
      "officeFabricIconFontName": "ContactCard",
      "properties": {
        "targetList": "Leads"
      }
    }
  ]
}
```



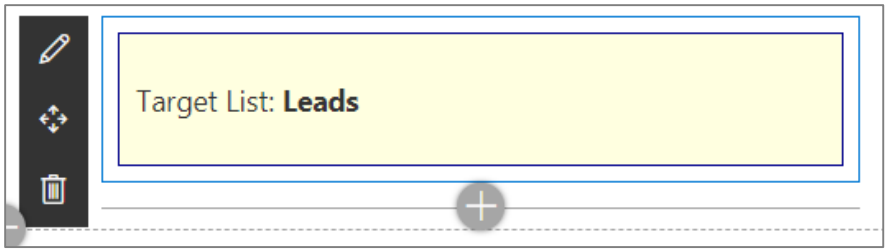
Designing the React Component

```tsx
export interface ILeadTrackerProps {
  targetListDefault: string;
}
```

```tsx
export interface ILeadTrackerState {
  targetList: string;
  loading: boolean;
}
```

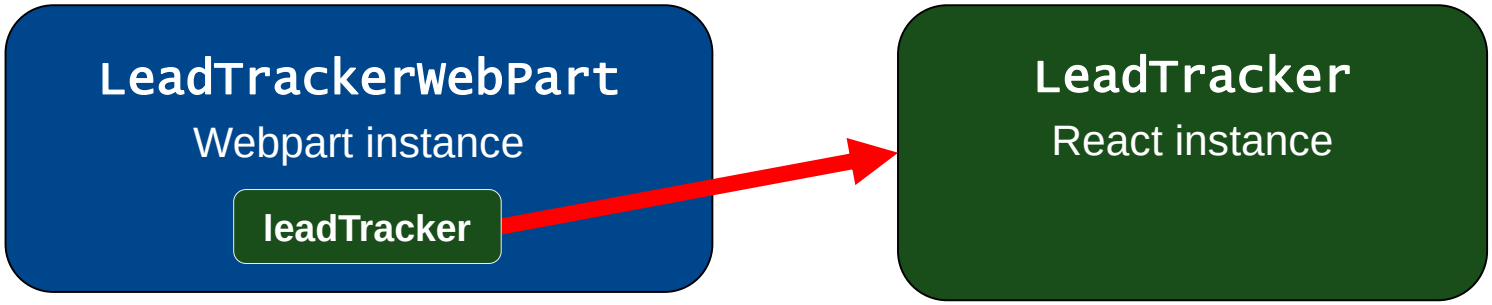
```tsx
import { ILeadTrackerProps } from './ILeadTrackerProps';
import { ILeadTrackerState } from './ILeadTrackerState';

export default class LeadTracker extends React.Component<ILeadTrackerProps, ILeadTrackerState> {

  public state: ILeadTrackerState = {
    targetList: this.props.targetListDefault,
    loading: false
  };

  public render(): React.ReactElement<ILeadTrackerProps> {
    return (
      <div className={styles.leadTracker}>
        <p>Target List: <strong>{ this.state.targetList }</strong></p>
      </div>
    );
  }
}
```

LeadTracker
React component

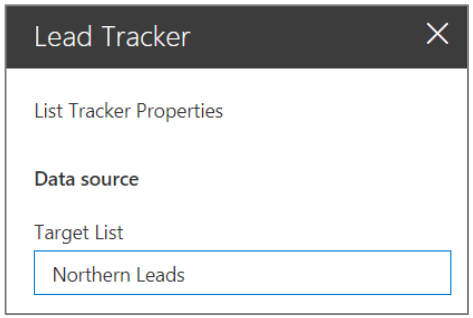


Referencing the React Component Instance

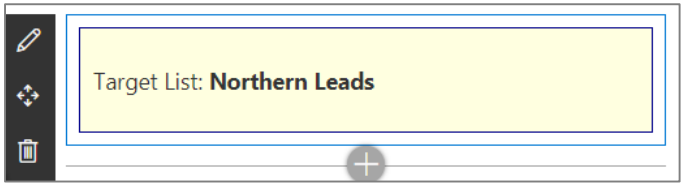
```
import LeadTracker from './components/LeadTracker';
import { ILeadTrackerProps } from './components/ILeadTrackerProps';

export default class LeadTrackerWebPart extends BaseClientSideWebPart<ILeadTrackerWebPartProps> {

  private leadTracker: LeadTracker;

  public render(): void {
    const element: React.ReactElement<ILeadTrackerProps> = React.createElement(
      LeadTracker, { targetListDefault: this.properties.targetList }
    );
    this.leadTracker = <LeadTracker>ReactDom.render(element, this.domElement);
  }
```



Synchronizing React State with Webpart Properties



Automatic updates




```
protected onPropertyPaneFieldChanged(propertyPath: string, oldValue: any, newValue: any): void {
  super.onPropertyPaneFieldChanged(propertyPath, oldValue, newValue);

  if (propertyPath === 'targetList' && newValue) {
    this.leadTracker.setState({ targetList: newValue });
  }

}
```

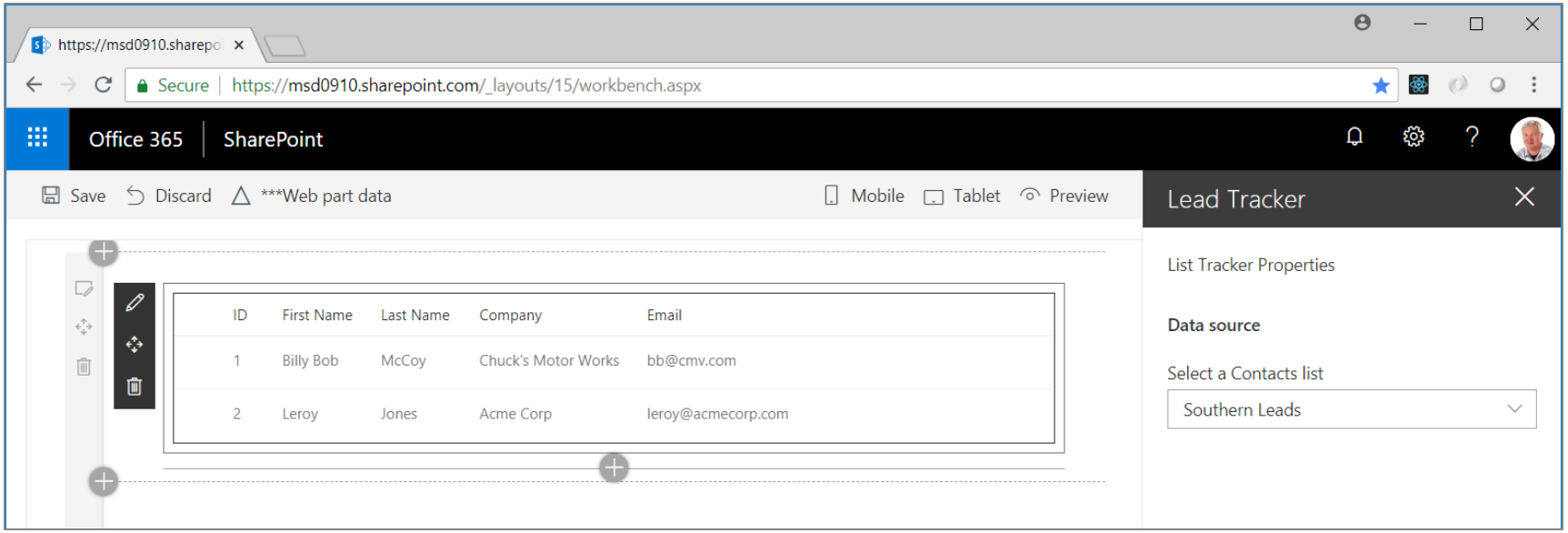
LeadTrackerWebPart
Webpart instance

React.Component.setState

targetList

LeadTracker
React instance

Demo 4 - The Lead Tracker React Webpart



Calling to the SharePoint Rest API

SharePointLeadsService.ts

```typescript
import {
  SPHttpClient,
  SPHttpClientResponse
} from '@microsoft/sp-http';

export default class SharePointLeadsService implements ILeadsService {

  constructor(private spHttpClient: SPHttpClient, private siteUrl: string) {
  }

  public getLeads(targetList: string): Promise<ILead[]> {

    let restUrl = this.siteUrl +
      `/_api/web/lists/getByTitle('${targetList}')/items/` +
      "?$select=Id,FirstName,Title,Company,Email";

    return this.spHttpClient.get(restUrl, SPHttpClient.configurations.v1)
      .then(response => response.json())
      .then(response => {
        return response.value.map(lead => <ILead>({
          id: lead.Id,
          firstName: lead.FirstName,
          lastName: lead.Title,
          company: lead.Company,
          emailAddress: lead.Email
        }));
      });
  }
```

Agenda

- ✓ Developing with Node.js, TypeScript and Webpack
- ✓ Learning React.js Fundamentals
- ✓ Using the Office UI Fabric React Component Library
- ✓ Developing React Webparts with SharePoint Framework
- ➢ Calling the Microsoft Graph API from React Webparts

SPFx Includes Microsoft Graph Client

- SharePoint Online already has an Azure AD application
  - SPFX solutions can call SharePoint REST API in the same domain
  - No extra authentication required
- What about calling Microsoft Graph API across domains?
  - SPFX provides proxy to call Microsoft Graph API
  - MSGraphClient is the new Microsoft Graph Client for SPFx
  - Moved out of preview to release with API v1.6
  - Abstracts the token acquisition from the SPFx development

SPFx Solutions Declare Permission Requests

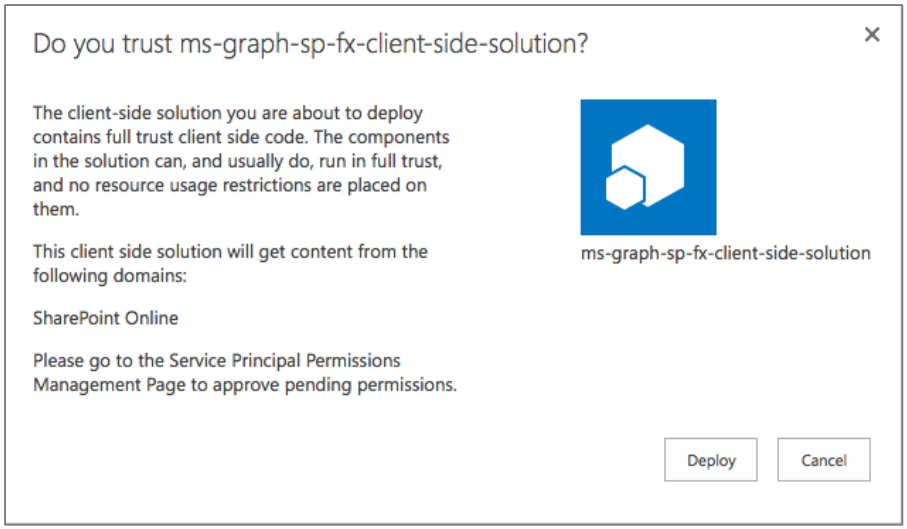
- Used to request tenant-wide permissions
- Permissions must be granted by tenant adminsitrator

package-solution.json

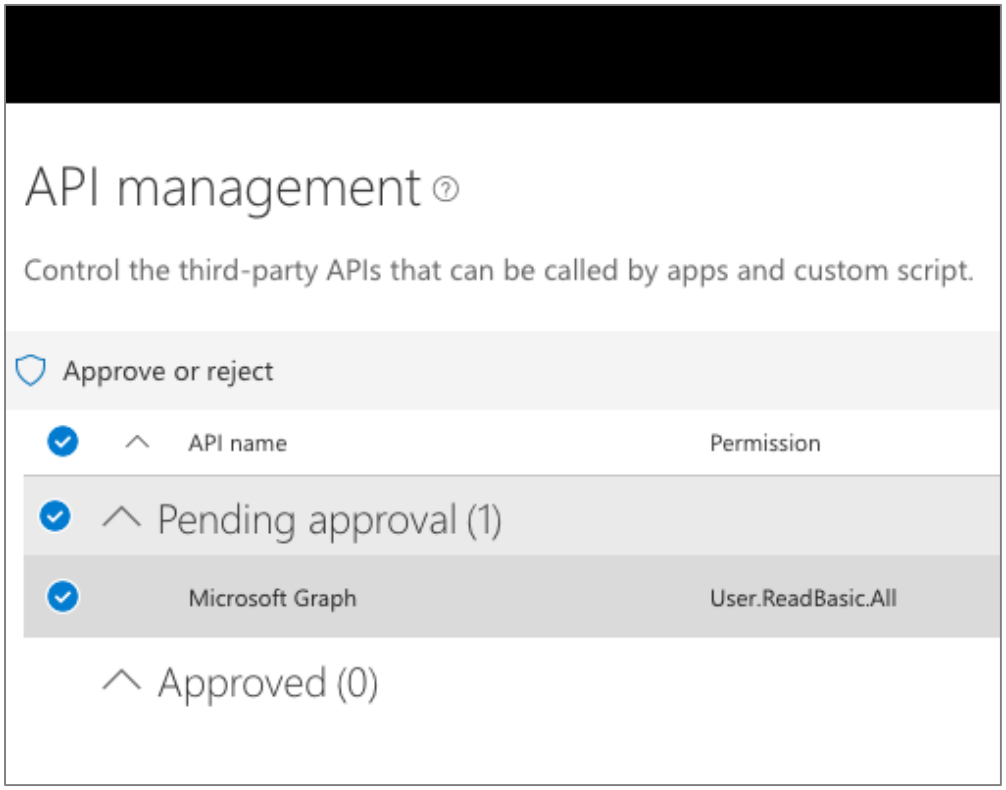
```
{
  "$schema": "https://developer.microsoft.com/json-schemas/spfx-build/package-solution.schema.json",
  "solution": {
    "name": "demo5-microsoft-graph",
    "id": "e708732a-1fab-4204-8c47-ceda6eb6a731",
    "version": "1.0.0.0",
    "includeClientSideAssets": true,
    "skipFeatureDeployment": true,
    "webApiPermissionRequests": [
      {
        "resource": "Windows Azure Active Directory",
        "scope": "User.Read"
      },
      {
        "resource": "Microsoft Graph",
        "scope": "User.ReadBasic.All"
      }
    ]
  },
  "paths": {
    "zippedPackage": "solution/demo5-microsoft-graph.sppkg"
  }
}
```

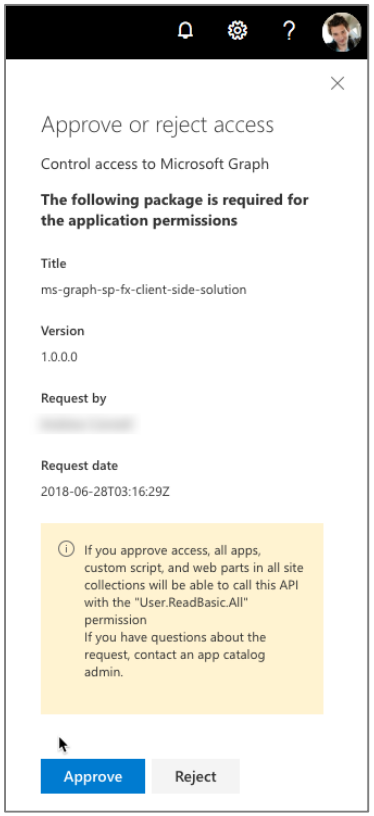
Add Package to SharePoint App Catalog

- Extra note in dialog notifies of additional step required
- While application can be installed in SharePoint sites, it does not have the permissions granted that it needs to access Azure AD protected resources

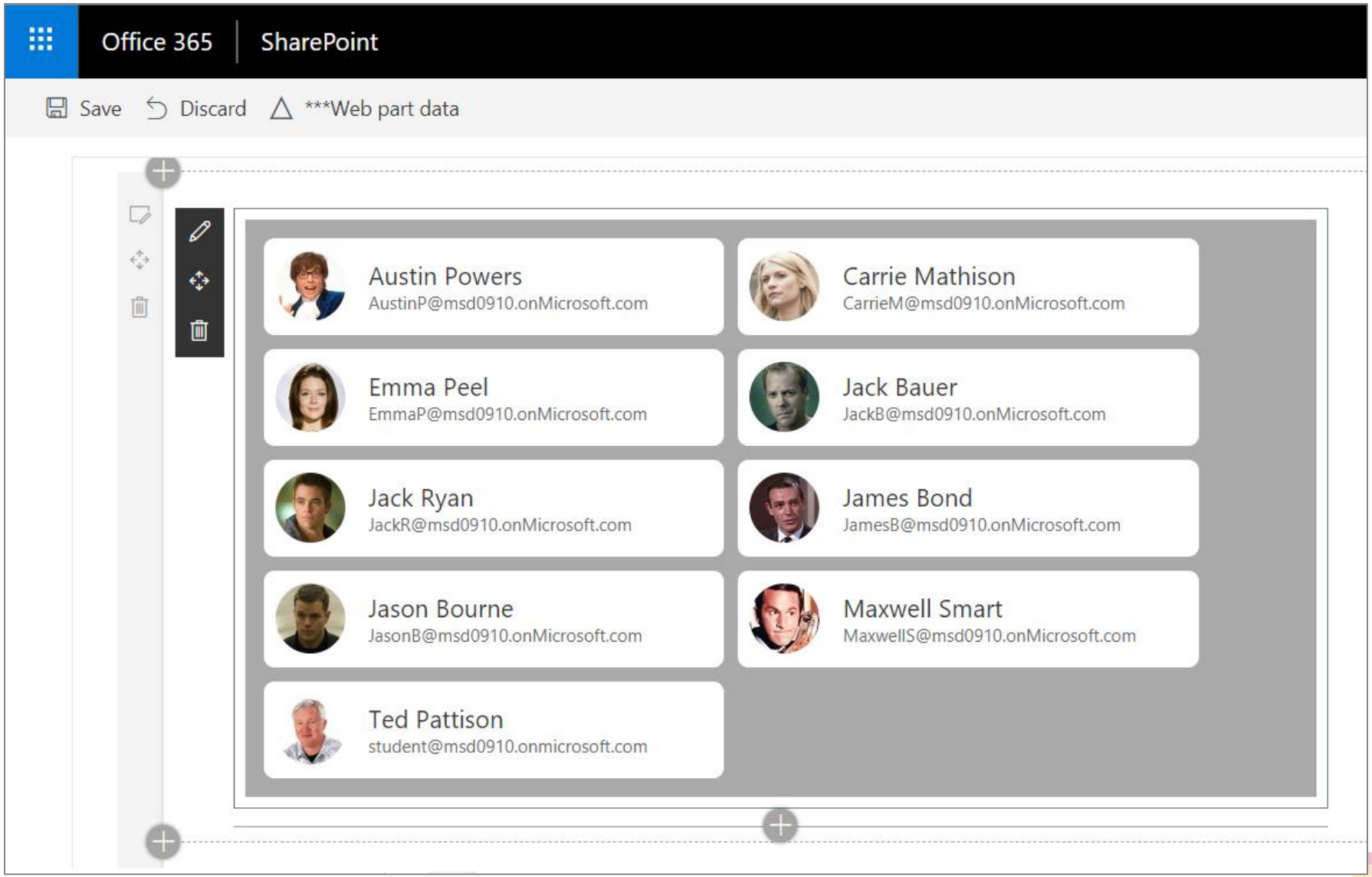


Approve / Reject with SharePoint Online API Management Page





Demo 5 - Calling Microsoft Graph API



Acquiring the MSGraphClient from WebPart

```
import { MSGraphClient } from "@microsoft/sp-http";

export default class UserViewerWebPart extends BaseClientSideWebPart<any> {

  public render(): void {

    this.context.msGraphClientFactory
      .getClient()
      .then((client: MSGraphClient): void => {
        // create React component by passing MSGraphClient
        const element: React.ReactElement<IUserViewerProps> = React.createElement(
          UserViewer, { msGraphClient: client }
        );
        ReactDom.render(element, this.domElement);
      });
  }

  protected onDispose(): void {
    ReactDom.unmountComponentAtNode(this.domElement);
  }

}
```

Microsoft Graph TypeScript Type Declarations

- Use the Microsoft Graph TypeScript Type Declarations
  - TypeScript type declarations used to introduce strong types
  - https://github.com/microsoftgraph/msgraph-typescript-typings
  - Installed using npm install

```
npm install @microsoft/microsoft-graph-types --save
```

  - Provides easy-to-use programming model

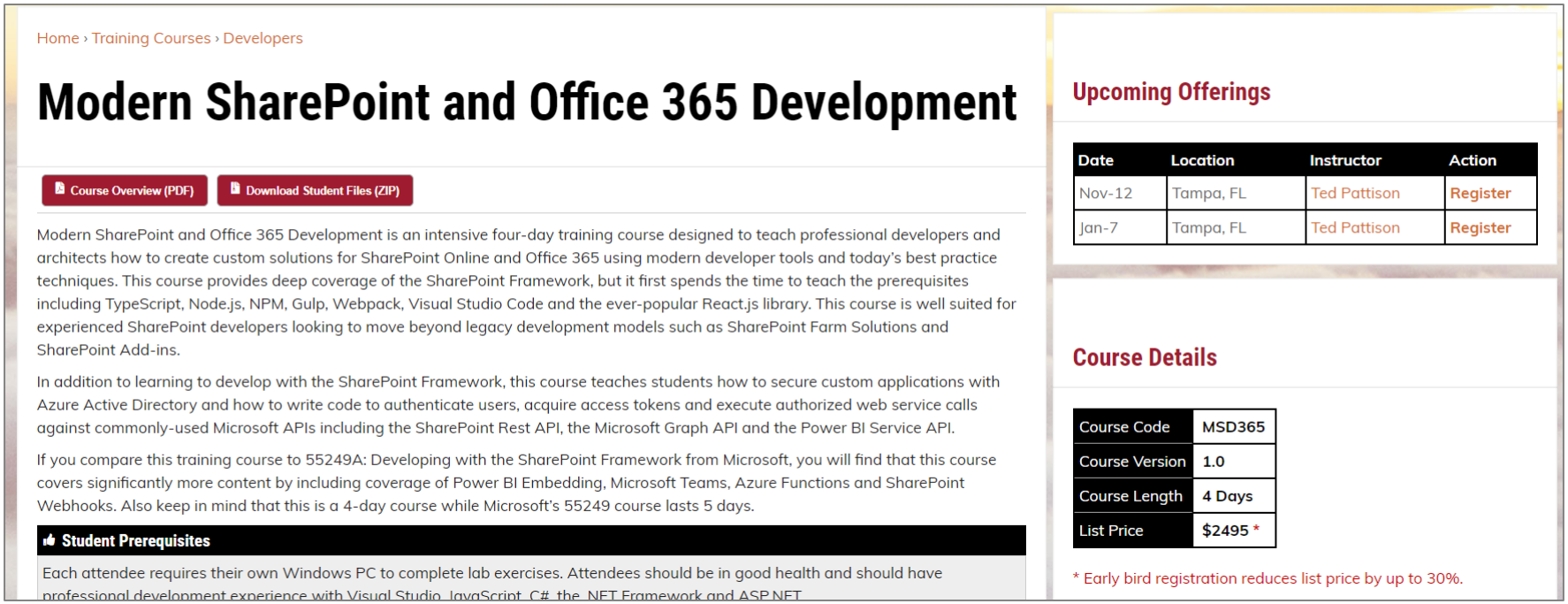
```
this.msGraphClient
  .api("me")
  .get((error: any, user: MicrosoftGraph.User, rawResponse?: any) => {
    // map response to IUser object
    return resolve(<IUser>({
      id: user.id,
      displayName: user.givenName + " " + user.surname,
      email: user.mail,
      phone: user.businessPhones[0]
    }));
  });
```

Service Class using Microsoft Graph API

```
import { MSGraphClient } from '@microsoft/sp-http';
import * as MicrosoftGraph from '@microsoft/microsoft-graph-types';

export default class MSGraphUsersService implements IUsersService {

  constructor(private msGraphClient: MSGraphClient) { }

  public getCurentUser(): Promise<IUser> {
    return new Promise<IUser>((resolve, reject) => {
      this.msGraphClient
        .api("me")
        .get((error: any, user: MicrosoftGraph.User, rawResponse?: any) => {
          // map response to IUser object
          return resolve(<IUser>({
            id: user.id,
            displayName: user.givenName + " " + user.surname,
            email: user.mail,
            phone: user.businessPhones[0]
          }));
        });
    });
  }
```

Effective SharePoint Framework Training

- MSD365: Modern SharePoint and Office 365 Development
  - 4-day of training with lots of hands-on labs
  - Learn how to develop with SPFx the right way right from the start
  - https://www.criticalpathtraining.com/courses/sharepoint/modern-sharepoint-office-365-development/



Home › Training Courses › Developers

Modern SharePoint and Office 365 Development

Course Overview (PDF) | Download Student Files (ZIP)

Modern SharePoint and Office 365 Development is an intensive four-day training course designed to teach professional developers and architects how to create custom solutions for SharePoint Online and Office 365 using modern developer tools and today's best practice techniques. This course provides deep coverage of the SharePoint Framework, but it first spends the time to teach the prerequisites including TypeScript, Node.js, NPM, Gulp, Webpack, Visual Studio Code and the ever-popular React.js library. This course is well suited for experienced SharePoint developers looking to move beyond legacy development models such as SharePoint Farm Solutions and SharePoint Add-ins.

In addition to learning to develop with the SharePoint Framework, this course teaches students how to secure custom applications with Azure Active Directory and how to write code to authenticate users, acquire access tokens and execute authorized web service calls against commonly-used Microsoft APIs including the SharePoint Rest API, the Microsoft Graph API and the Power BI Service API.

If you compare this training course to 55249A: Developing with the SharePoint Framework from Microsoft, you will find that this course covers significantly more content by including coverage of Power BI Embedding, Microsoft Teams, Azure Functions and SharePoint Webhooks. Also keep in mind that this is a 4-day course while Microsoft's 55249 course lasts 5 days.

Student Prerequisites

Each attendee requires their own Windows PC to complete lab exercises. Attendees should be in good health and should have professional development experience with Visual Studio, JavaScript, C#, the .NET Framework and ASP.NET

Upcoming Offerings

Date	Location	Instructor	Action
Nov-12	Tampa, FL	Ted Pattison	Register
Jan-7	Tampa, FL	Ted Pattison	Register

Course Details

Course Code	MSD365
Course Version	1.0
Course Length	4 Days
List Price	$2495 *

* Early bird registration reduces list price by up to 30%.

Summary

- ✓ Developing with Node.js, TypeScript and Webpack
- ✓ Learning React.js Fundamentals
- ✓ Using the Office UI Fabric React Component Library
- ✓ Developing React Webparts with SharePoint Framework
- ✓ Calling the Microsoft Graph API from React Webparts