

# Developing React Web Parts



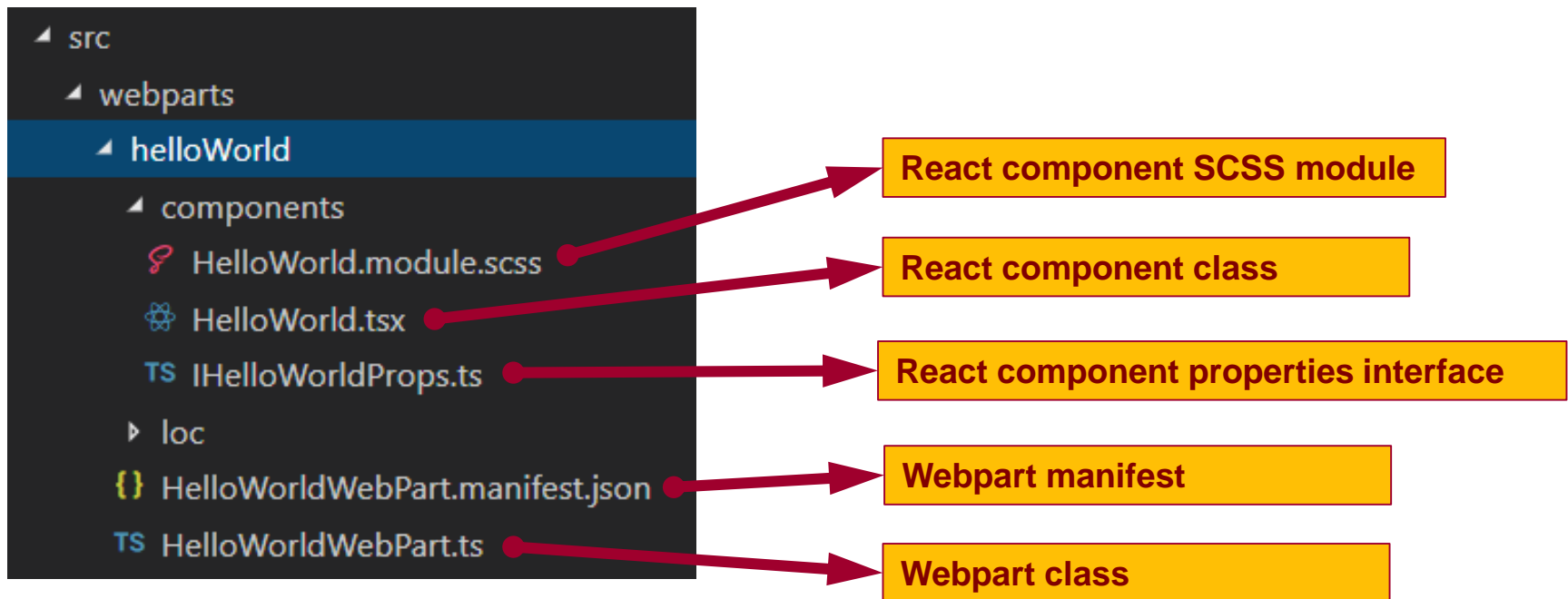
# Agenda

- Designing and Developing React Web Parts
- Web Part Properties versus React Component State
- Leveraging the Office UI Fabric React Library
- Developing Web Parts using the SharePoint REST API



# 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 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 class  
instance

React.CreateElement

description

React component  
instance



# React Webpart Styling

```
HelloWorld.module.scss •  
  
.helloWorld {  
  background-color: lightsalmon;  
  border: 4px solid purple;  
  border-radius: 12px;  
  
  .title {  
    padding: 8px;  
    font-size: 48px;  
  }  
}
```



```
HelloWorld.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>  
    );  
  }  
}
```



# Agenda

- ✓ Designing and Developing React Web Parts
- Web Part Properties versus React Component State
  - Leveraging the Office UI Fabric React Library
  - Developing Web Parts using the SharePoint REST API



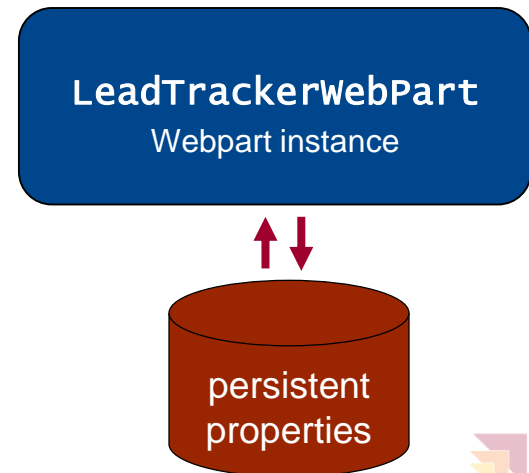
# Webpart Persistent Properties

- 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

```
{  
  "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"  
      }  
    }  
  ]  
}
```





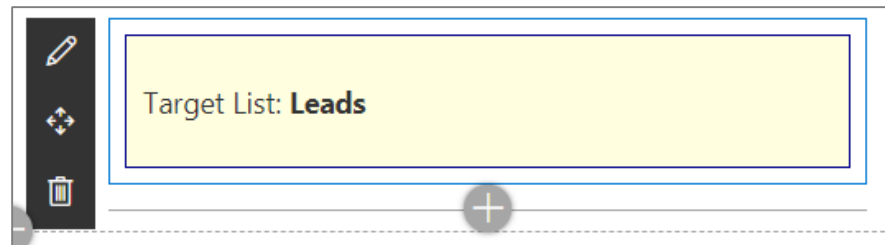
# Designing the React Component

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

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

```
import { ILeadTrackerProps } from './ILeaderTrackerProps';  
import { ILeadTrackerState } from './ILeaderTrackerState';  
  
export default class LeadTracker extends React.Component<ILeaderTrackerProps, ILeadTrackerState> {  
  
  public state: ILeadTrackerState = {  
    targetList: this.props.targetListDefault,  
    loading: false  
  };  
  
  public render(): React.ReactElement<ILeaderTrackerProps> {  
    return (  
      <div className={styles.leadTracker}>  
        <p>Target List: <strong>{ this.state.targetList }</strong></p>  
      </div>  
    );  
  }  
}
```

LeadTracker  
React component





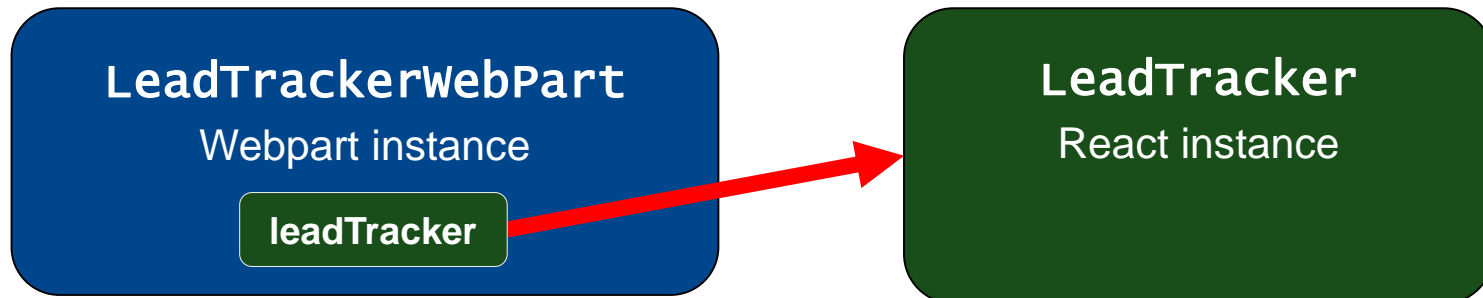
# Referencing the React Component Instance

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

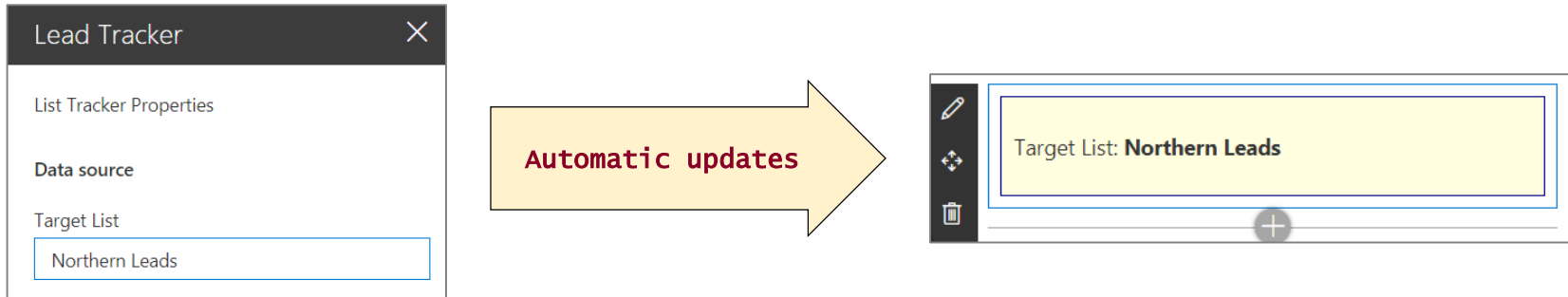
export default class LeadTrackerWebPart extends BaseClientSideWebPart<ILeadTrackerWebPartProps> {

  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



```
protected onPropertyPaneFieldChanged(propertyPath: string, oldValue: any, newValue: any): void {  
    super.onPropertyPaneFieldChanged(propertyPath, oldValue, newValue);  
  
    if (propertyPath === 'targetList' && newValue) {  
        this.leadTracker.setState({ targetList: newValue });  
    }  
}
```



# Agenda

- ✓ Designing and Developing React Web Parts
- ✓ Web Part Properties versus React Component State
- Leveraging the Office UI Fabric React Library
  - Developing Web Parts using the SharePoint REST API



# What is the Office UI Fabric?

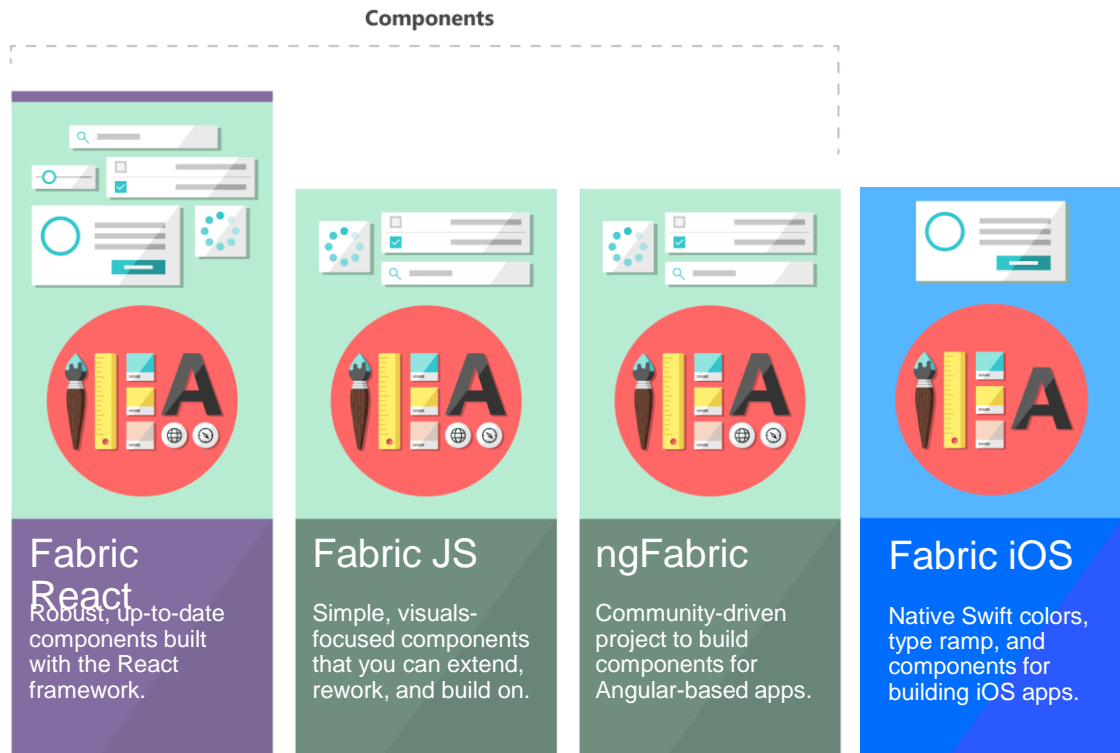
- Office UI Fabric is responsive, mobile-first, front-end style framework
  - Built by Microsoft to style Office 365, OneDrive and SharePoint sites
  - All about styling instead of JavaScript
  - Can be used by 3rd party developers



## Fabric Core

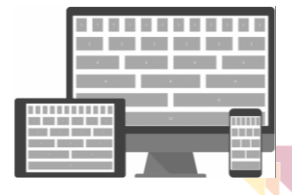
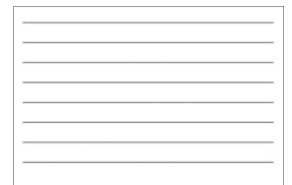
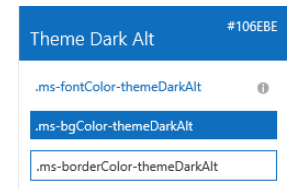
Core elements of the design language including icons, colors, type, and the grid

or



# 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



# Styles

- The Office UI Fabric provides styles for..
  - Typography
  - Color
  - Icons
  - Animations
  - Responsive Grid
  - Localization



# 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
<code>.ms-fontWeight-light</code>	Light
<code>.ms-fontWeight-semilight</code>	Semilight
<code>.ms-fontWeight-regular</code>	Regular
<code>.ms-fontWeight-semibold</code>	Semi Bold



# Color

- Includes 9 theme colors and 11 neutral colors.
  - Helper classes for text, border, background, and hover states.
  - Color classes act as hooks into the Office 365 theming system

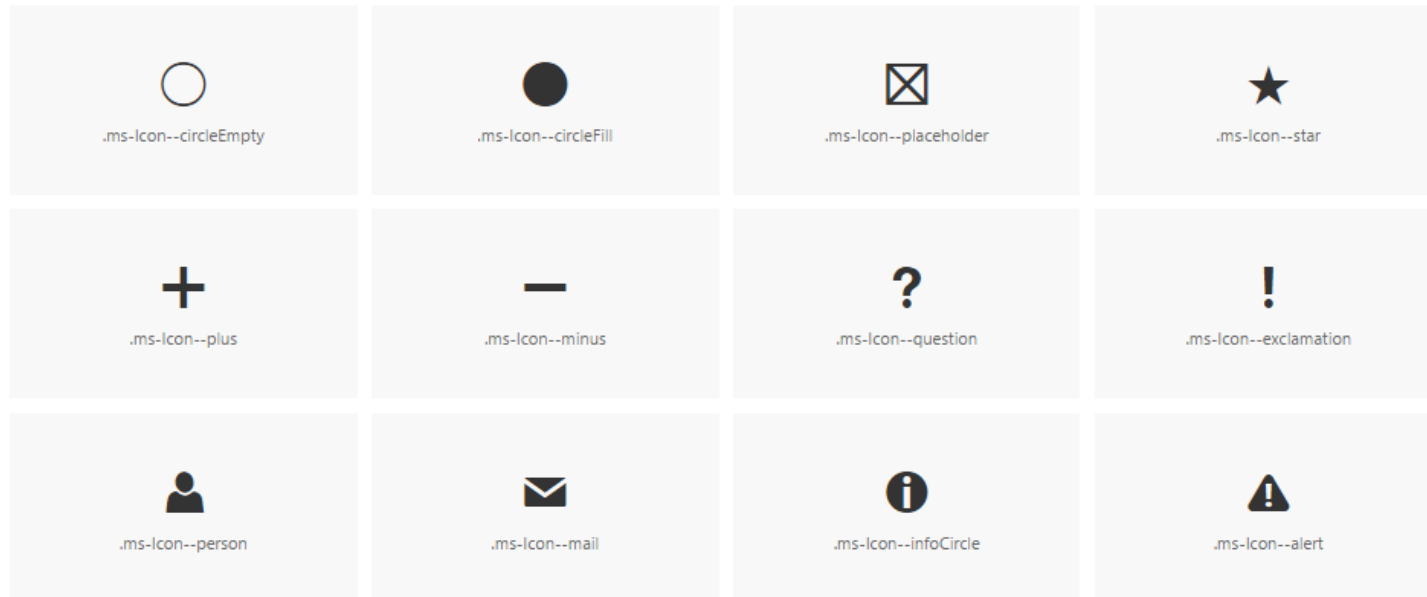
Theme colors		Neutral colors	
themeDarker	#00457B	black	#000000
themeDark	#005A9E	neutralDark	#212121
themeDarkAlt	#106EBC	neutralPrimary	#333333
themePrimary	#0078D7	neutralPrimaryAlt	#3C3C3C
themeSecondary	#2B88D8	neutralSecondary	#666666
themeTertiary	#71A6E5	neutralTertiary	#A6A6A6
themeLight	#C7E0F4	neutralTertiaryAlt	#C8C8C8
themeLighter	#DEEDF9	neutralLight	#EAEAEA
themeLighterAlt	#F9F9FC	neutralLighter	#F4F4F4
		neutralLighterAlt	#F9F9F9
		white	#FFFFFF

Accent colors			
<b>Yellows</b>	<b>Oranges</b>	<b>Reds</b>	<b>Magentas</b>
yellow #FFD700	orange #FF8C00	redDark #800000	magentaDark #800080
yellowLight #FFFF00	orangeLight #FFA500	red #FF0000	magenta #FF00FF
	orangeLighter #FFDAB9		magentaLight #FFC0CB
<b>Purples</b>	<b>Blues</b>	<b>Teals</b>	<b>Greens</b>
purpleDark #483D8B	blueDark #000080	tealDark #008080	greenDark #008000
purple #800080	blueMid #0000CD	teal #00CED1	green #00FF00
purpleLight #DDA0DD	blue #1E90FF	tealLight #7FFFD4	greenLight #90EE90
	blueLight #6495ED		



# Icons

- Fabric uses a custom font for its iconography.
  - Font contains glyphs you can scale, color, and style



# Icons

- To use the icons, combine the base **ms-Icon** class with a modifier class for the specific icon.

```
<i class="ms-Icon ms-Icon--mail" aria-hidden="true"></i>
```



.ms-Icon--xCircle



.ms-Icon--mailOpen



.ms-Icon--people



.ms-Icon--bell



.ms-Icon--calendar



.ms-Icon--scheduling



.ms-Icon--event



.ms-Icon--folder





# Office UI Fabric React Components

- Fabric's React components are building blocks for UI
  - Components can be used in general React development
  - Components can be used in SharePoint Framework development

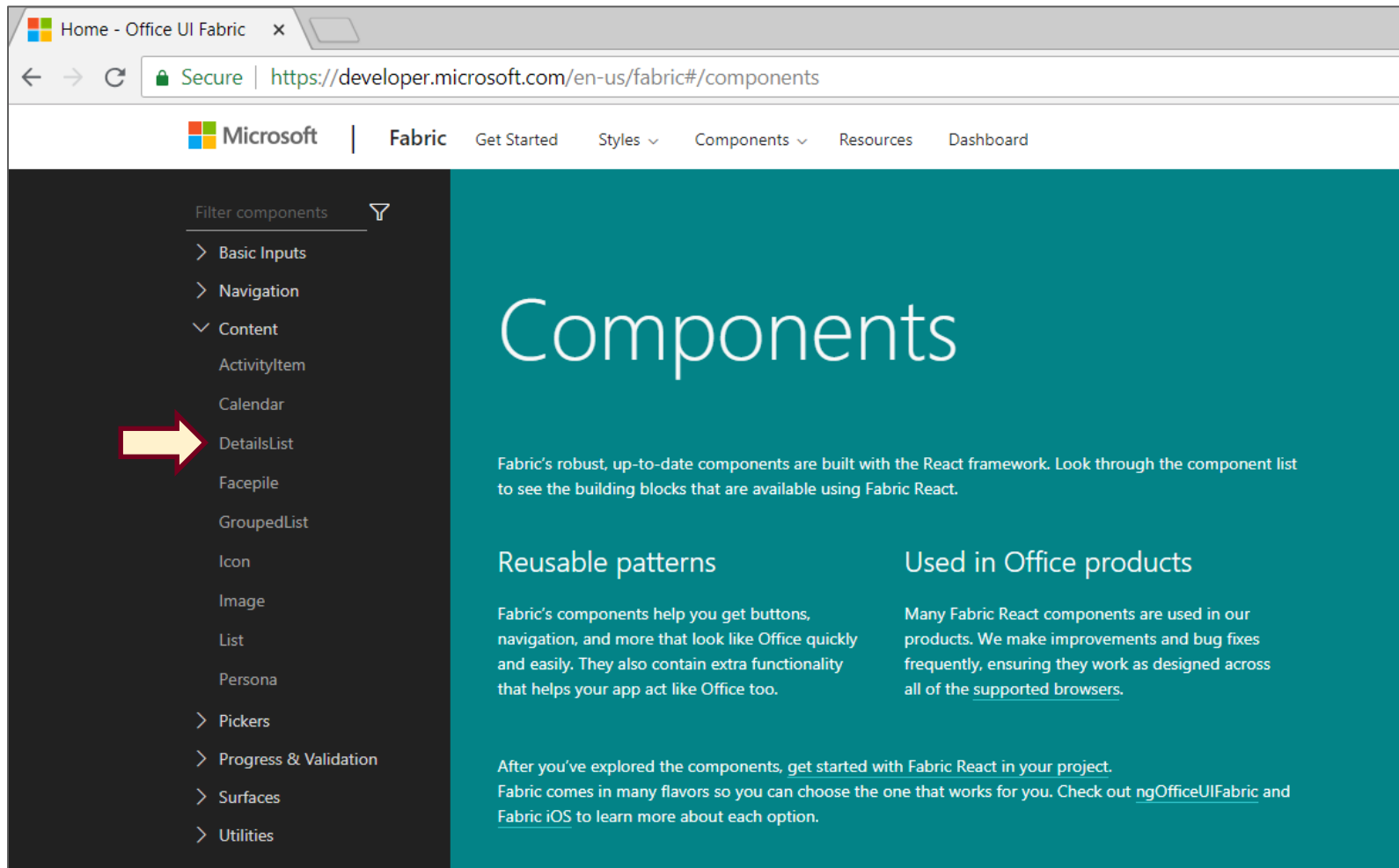
```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { PrimaryButton } from 'office-ui-fabric-react/lib/Button';

ReactDOM.render(
  <PrimaryButton>
    I am a button.
  </PrimaryButton>,
  document.body.firstChild
);
```



# Office UI Fabric React Component Library

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





# 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>
  );
}
```



# Agenda

- ✓ Designing and Developing React Web Parts
- ✓ Web Part Properties versus React Component State
- ✓ Leveraging the Office UI Fabric React Library
- Developing Web Parts using the SharePoint REST API
  - Designing Web Parts to Manage SharePoint Lists



# RESTful Web Services

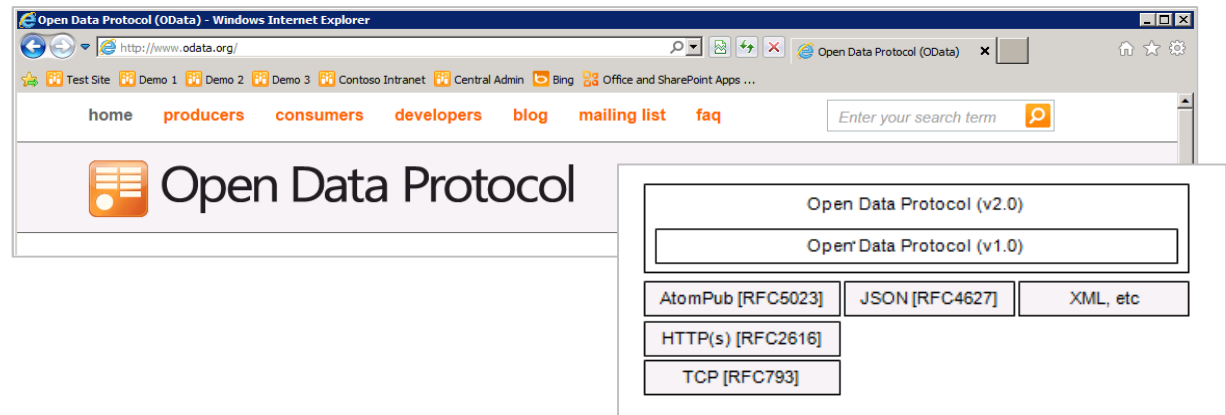
- RESTful Web Service
  - implemented using the principles of REST
  - REST URI = [base URI] + [resource path] + [query options]
  - Calls based on standard HTTP verbs (**GET**, **POST**, **PUT**, **DELETE**)
  - Passes data to and from client using representations
  - Can be designed to implement custom APIs and/or standard APIs
- Data passed across network using representations
  - Representations model resources – but they're different
  - Based on common formats: HTML, XML, ATOM and JSON
  - Based on specific Internet media types



# OData Primer

- What is OData?
  - A standardized REST API interface for common CRUD operations
  - Defined by Open Data Protocol specification
  - OData services becoming more popular on Internet (e.g. Netflix)
  - SharePoint 2010 introduced a REST API for dealing with list items
  - SharePoint 2013 introduces new and expanded REST API

for an excellent resource go to  
<http://www.odata.org>



# OData Query Option Parameters

## ▪ \$select

- [http://services.odata.org/OData/OData.svc/Products?\\$select=Price,Name](http://services.odata.org/OData/OData.svc/Products?$select=Price,Name)

## ▪ \$filter

- [http://services.odata.org/OData/OData.svc/Products?\\$filter=startswith\(CompanyName, 'Alfr'\)](http://services.odata.org/OData/OData.svc/Products?$filter=startswith(CompanyName, 'Alfr'))

## ▪ \$orderby

- [http://services.odata.org/OData/OData.svc/Products?\\$orderby=Rating](http://services.odata.org/OData/OData.svc/Products?$orderby=Rating)

## ▪ \$top

- [http://services.odata.org/OData/OData.svc/Products?\\$top=5](http://services.odata.org/OData/OData.svc/Products?$top=5)

## ▪ \$skip

- [http://services.odata.org/OData/OData.svc/Products?\\$skip=5](http://services.odata.org/OData/OData.svc/Products?$skip=5)
- [http://services.odata.org/OData/OData.svc/Products?\\$skip=5&\\$top=5](http://services.odata.org/OData/OData.svc/Products?$skip=5&$top=5)

## ▪ \$expand

- [http://services.odata.org/OData/OData.svc/Categories?\\$expand=Products](http://services.odata.org/OData/OData.svc/Categories?$expand=Products)



# Using the \$filter Parameter

Logical Operators		
Eq	Equal	/Suppliers?\$filter=Address/City eq 'Las Vegas'
Ne	Not equal	/Suppliers?\$filter=Address/City ne 'London'
Gt	Greater than	/Products?\$filter=Price gt 20
Ge	Greater than or equal	/Products?\$filter=Price ge 10
Lt	Less than	/Products?\$filter=Price lt 20
Le	Less than or equal	/Products?\$filter=Price le 100
And	Logical and	/Products?\$filter=Price le 200 and Price gt 3.5
Or	Logical or	/Products?\$filter=Price le 3.5 or Price gt 200
Not	Logical negation	/Products?\$filter=not endswith(Description,'milk')
Arithmetic Operators		
Add	Addition	/Products?\$filter=Price add 5 gt 10
Sub	Subtraction	/Products?\$filter=Price sub 5 gt 10
Mul	Multiplication	/Products?\$filter=Price mul 2 gt 2000
Div	Division	/Products?\$filter=Price div 2 gt 4
Mod	Modulo	/Products?\$filter=Price mod 2 eq 0
Grouping Operators		
( )	Precedence grouping	/Products?\$filter=(Price sub 5) gt 10



# \$filter Parameter String Functions

## String Functions

bool substringof(string p0, string p1)	Customers?\$filter=substringof('Alfreds', CompanyName) eq true
bool endswith(string p0, string p1)	Customers?\$filter=endswith(CompanyName, 'Futterkiste') eq true
bool startswith(string p0, string p1)	Customers?\$filter=startswith(CompanyName, 'Alfr') eq true
int length(string p0)	Customers?\$filter=length(CompanyName) eq 19
int indexof(string p0, string p1)	Customers?\$filter=indexof(CompanyName, 'lfreds') eq 1
string replace(string p0, string find, string replace)	Customers?\$filter=replace(CompanyName, ' ', '') eq 'AlfredsFutterkiste'
string substring(string p0, int pos)	Customers?\$filter=substring(CompanyName, 1) eq 'lfreds Futterkiste'
string substring(string p0, int pos, int length)	Customers?\$filter=substring(CompanyName, 1, 2) eq 'lf'
string tolower(string p0)	Customers?\$filter=tolower(CompanyName) eq 'alfreds futterkiste'
string toupper(string p0)	Customers?\$filter=toupper(CompanyName) eq 'ALFREDS FUTTERKISTE'
string trim(string p0)	Customers?\$filter=trim(CompanyName) eq 'Alfreds Futterkiste'
string concat(string p0, string p1)	Customers?\$filter=concat(concat(City, ' '), Country) eq 'Berlin, Germany'





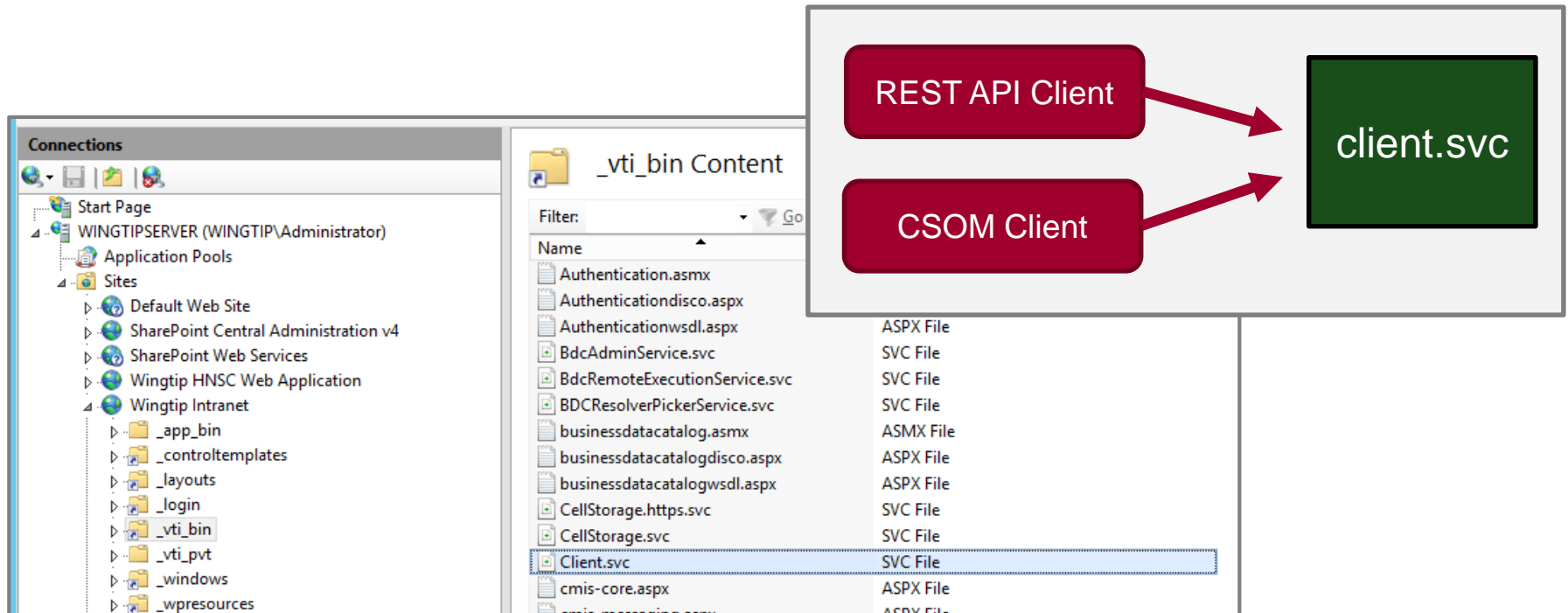
# Remote Communications with SharePoint

- In SharePoint 2003 and SharePoint 2007
  - SOAP-based web services (e.g. Lists.asmx)
- In SharePoint 2010
  - Client-side Object Model (CSOM)
  - REST API for list items accessible through **ListData.svc**
- In SharePoint 2013
  - Expanded CSOM Support
  - New SharePoint REST API replaces **ListData.svc**
- In SharePoint 2016 and SharePoint Online
  - REST API improved with greater support for ODATA 4.0



# SharePoint REST API Architecture

- REST API entry point is client.svc
  - In SharePoint 2010, client.svc only used by CSOM
  - Since SharePoint 2013, client.svc used by CSOM and REST API



# SharePoint REST URLs and the **\_api** Alias

- SharePoint REST API provides **\_api** alias
  - The **\_api** alias maps to **\_vti\_bin/client.svc**
  - Alias used to make SharePoint REST API URLs cleaner
  - Alias serves to decouple URLs from underlying architecture
- This URL works but it is not recommended
  - [http://intranet.wingtip.com/\\_vti\\_bin/client.svc/web](http://intranet.wingtip.com/_vti_bin/client.svc/web)
- SharePoint REST API URLs should be created with **\_api**
  - [http://intranet.wingtip.com/\\_api/web](http://intranet.wingtip.com/_api/web)



# Anatomy of a SharePoint REST URL

- SharePoint REST made up of three parts
  - Base URI  
`http://intranet.wingtip.com/_api`
  - Target SharePoint Object  
`web`
  - Query String Parameter options  
`?$select=Id,Title,MasterUrl`

```
http://intranet.wingtip.com/_api/web/?$select=Id,Title,MasterUrl
```



# Mapping SharePoint Objects to URLs

SharePoint Object	Object mapping
Site Collection	site
Site	web
Lists collection	web/lists
List by ID	web/lists(guid'402cd788-9c5c-4931-92d6-09f18efb368c')
List by Title	web/lists/getByTitle('Customers')
List property	web/lists/getByTitle('Customers')/Title
List items collection	web/lists/getByTitle('Customers')/items
List item	web/lists/getByTitle('Customers')/items(1)
List item property	web/lists/getByTitle('Customers')/items(1)/FirstName



# ODATA Formats and the Accept Header

- Verbose (aka Full Metadata)

`accept: application/json; odata=verbose`

- Minimal Metadata

`accept: application/json; odata=minimalmetadata`

`accept: application/json`

- No Metadata

`accept: application/json; odata=nometadata`



# Passing SPHttpClient to the React Component

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

export interface ILeadTrackerProps {
  targetListDefault: string;
  siteUrl: string;
  spHttpClient: SPHttpClient | undefined;
}
```

```
public render(): void {

  const element: React.ReactElement<ILeadTrackerProps> = React.createElement(
    LeadTracker, {
      targetListDefault: this.properties.targetList,
      siteUrl: this.context.pageContext.web.absoluteUrl,
      spHttpClient: <SPHttpClient>this.context.spHttpClient
    }
  );

  this.leadTracker = <LeadTracker>ReactDOM.render(element, this.domElement);
}
```





# Service Class using SPHttpClient

```
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
        }));
      });
  }
}
```



# Calling SPHttpClient.get

```
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
            }));
        });
}
```

```
public getLeadsLists(): Promise<IList[]> {

    let restUrl = this.siteUrl + "/_api/web/lists/" +
        "?$select=Id,Title&$filter=BaseTemplate+eq+105";

    return this.spHttpClient.get(restUrl, SPHttpClient.configurations.v1)
        .then(response => response.json())
        .then(response => {
            return response.value.map(list => <IList>({
                id: list.Id,
                title: list.Title
            }));
        });
}
```



# Agenda

- ✓ Designing and Developing React Web Parts
- ✓ Web Part Properties versus React Component State
- ✓ Leveraging the Office UI Fabric React Library
- ✓ Developing Web Parts using the SharePoint REST API
- Designing Web Parts to Manage SharePoint Lists





**DEMO**

## Reading Items from a SharePoint List



# Summary

- ✓ Designing and Developing React Web Parts
- ✓ Web Part Properties versus React Component State
- ✓ Leveraging the Office UI Fabric React Library
- ✓ Developing Web Parts using the SharePoint REST API
- ✓ Designing Web Parts to Manage SharePoint Lists

