



DURHAM
2014



SharePoint
SATURDAY

#SPS EVENTS
@SPSDURHAM

Building SharePoint
2013 Apps with
AngularJS

Steve Pietrek
Cardinal Solutions
@spietrek





Steve Pietrek



Cardinal Solutions

SharePoint Application Architect
JavaScript Developer
Microsoft Practice Manager
Raleigh/Durham

Contact

@spietrek
spietrek@cardinalsolutions.com

References

<http://jsfiddle.net/spietrek/VKbpu/>
<https://github.com/spietrek/SPSDurham2014>

Agenda

- 1 Overview of App Model
- 2 Tools
- 3 AngularJS Overview
- 4 Best Practices

New app model

SharePoint 2007 Challenge

FULL TRUST

- Developers build custom solutions
- Administrators can only secure solutions with Code Access Security (CAS)
 - Hard to control what is in custom code (Elevated Privileges)
- Biggest cause of SharePoint support cases: custom code



Developer

- Design, build and test customizations



Administrator

- Install and monitor customizations



Site Collection Owner

- Activate and use customizations

SharePoint 2010 Challenge

SANDBOX SOLUTIONS

- Developers build custom solutions
- Administrators leverage resource monitors to check site collection usage
- Site collection owners deploy, activate and implement the customizations



Developer

- Design, build and test customizations



Administrator

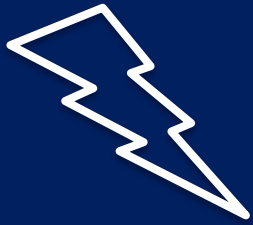
- Monitor customizations



Site Collection Owner

- Install customizations
- Activate and use customizations

SharePoint Development Challenges



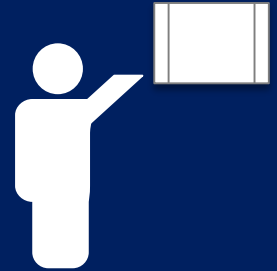
Performance
and Stability



Lifecycle
Management



Not Cloud
Ready



Resources

Benefits of new app model



Infrastructure

- Isolation & multitenant
- Reduces risk to farm
- Simplifies SharePoint upgrades
- Cloud ready
- Corporate App Catalog facilitates governance



Developers

- Larger developer reach
- Use industry standards
- Use non-SharePoint technologies
- Lifecycle management (development, deployment, versioning, upgrades)



Users

- Based on familiar app model
- Apps can be downloaded from SharePoint Store or Corporate App Catalog

Hosting Options

Cloud-hosted apps

- Use server code
- Receive SP events
- Use REST/CSOM/OAuth to access SP
- Use SP artifacts & out-of-box web parts
- May require own handling of mulitenancy & permissions management
- Access external data

Provider-hosted app

Provide your own hosting environment
(Dedicated server or hosting service)

SharePoint
Host Web

Your Hosted
Site

Autohosted app

Windows Azure + SQL Azure
provisioned automatically as apps are
installed

SharePoint
Host Web

Azure




SharePoint-hosted app

- Provisions an isolated sub web on a host web
- Use SP artifacts & out-of-box web parts
- Use HTML & JavaScript for UI & client-side logic
- Access external data
- **IMPORTANT** RequestExecutor when accessing data in Host Web

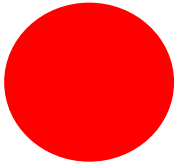
SharePoint
Host Web

SharePoint
App Web

App Shapes

Shape	Description
	<p>Full Page</p> <p>App that implements a new business scenario (LOB).</p>
	<p>App Part</p> <p>Provides new web parts you can add to your sites</p>
	<p>Extension App (Custom Action)</p> <p>Add new actions for documents and items to the ribbon or menu</p>

When NOT to Use App Model



- Server-side Object Model
- Deploying to the SharePoint server
- Branding (master pages, application pages)
- Timer jobs
- Custom field types
- Multiple widgets on single page (maybe?)

Project 1 – Line of Business App

Technical Requirements

- Full Page
- Deployed to Office 365
- Content stored in SharePoint lists/libraries
- SharePoint-hosted app
- 2 Developers (1 had no SharePoint experience)
- Limited JavaScript experience

Lessons Learned

- AngularJS framework learning curve
- Positive non-SharePoint developer productivity
- Lifecycle management (development, deployment, versioning, upgrades)
- Browser developer tools
- Slow performance vs. traditional JavaScript SPA development

Project 2 – Intranet Site

Technical Requirements

- Upgrade from SP 2003
- Work in Office 365. Deploy on-premise.
- SharePoint-hosted apps
- Many widgets (8-12) per page (4 pages)
- Content stored in SharePoint lists/libraries
- 3 Developers (2 had no SharePoint experience)

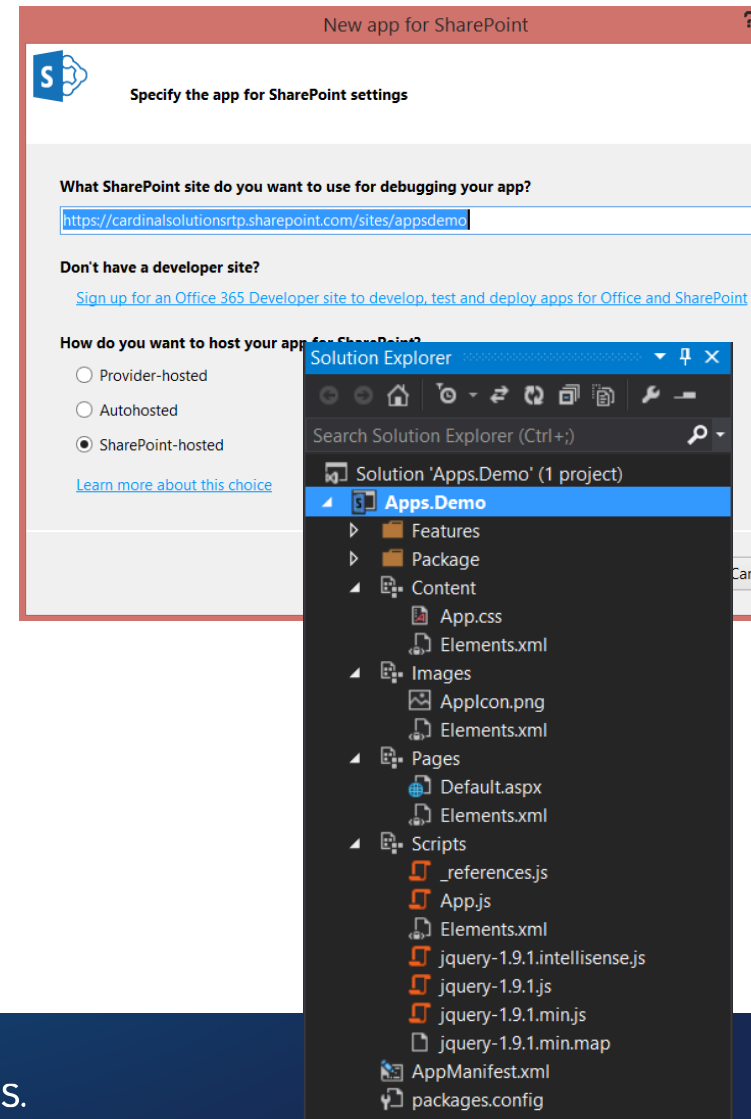
Lessons Learned

- Cross-domain performance issues
- Forced to mix client-side API's
- Build on-premise App Catalog BEFORE development starts (if needed)
- Test all devices early (i.e. iPad) >> scrolling issues
- Many app parts != Apps
- Rewrote all apps (except 2) to use straight JavaScript

Tools

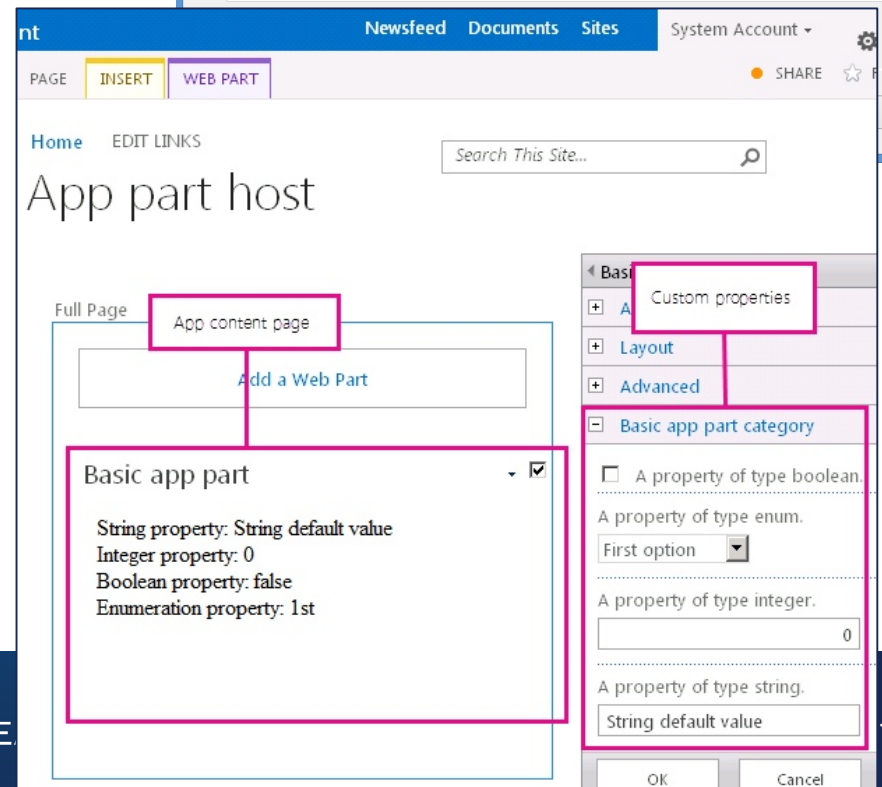
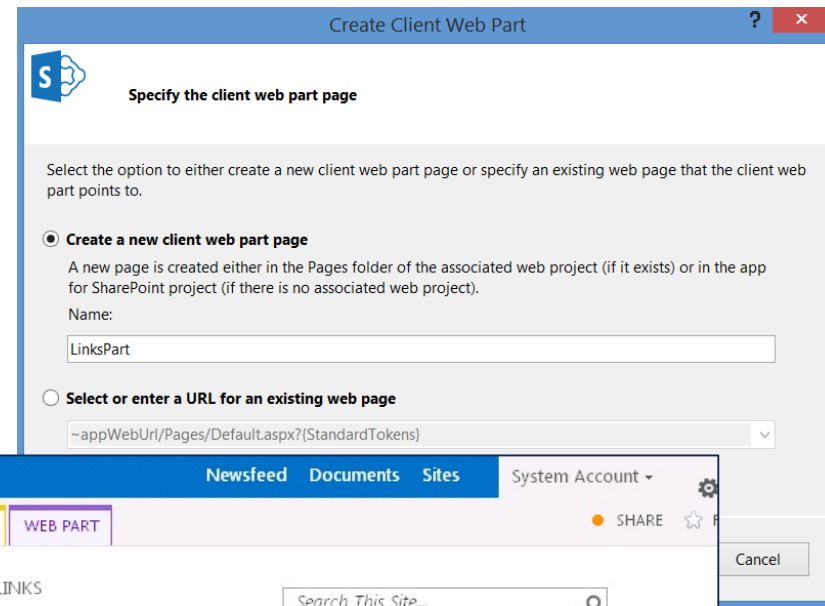
Visual Studio Tools

- Tools
 - Project templates, Intellisense, debugging, etc.
 - Microsoft Office Developer Tools for Visual Studio 2012
 - Included in Visual Studio 2013
- Development environments
 - Develop against a local SharePoint server
 - Develop remotely against Office 365 using Developer site
- Developer Site Template



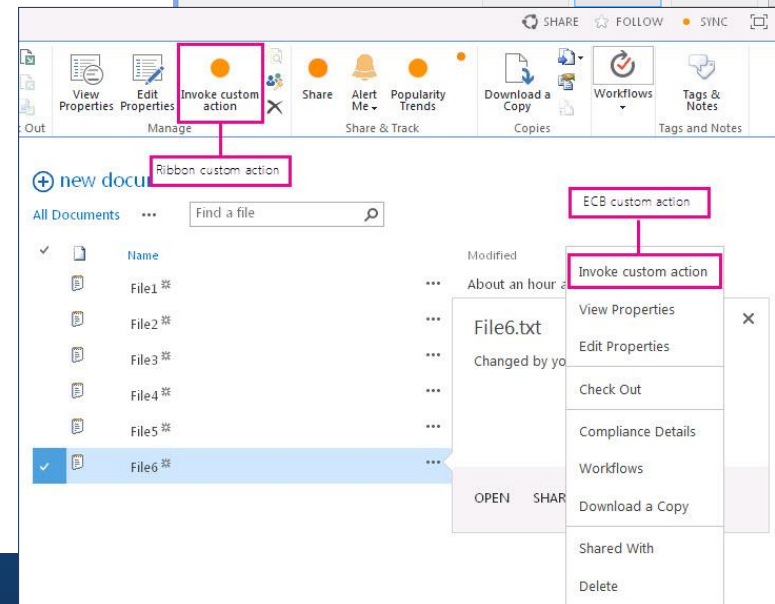
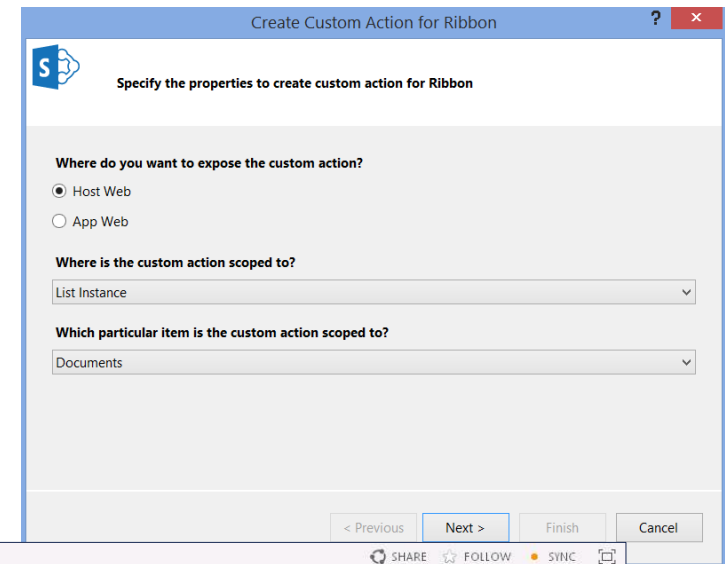
Client App Parts

- Wizard creates XML and page
- Web part properties can be added to XML file or through properties
- Cannot scroll iFrames in iOS



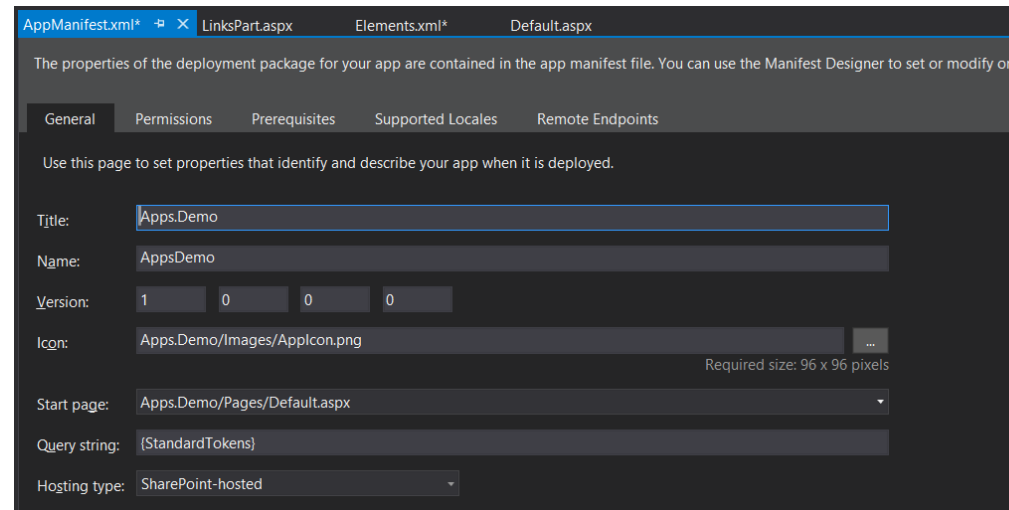
UI Custom Actions

- Wizard creates XML
- Creates UI extensions
- Show on the ribbon
- Add to item menu (ECB)



App Manifest

- General properties
- Permissions
- Prerequisites
- Supported Locales
- Remote Endpoints



The screenshot shows the 'AppManifest.xml*' window in a web browser. The 'General' tab is selected, showing fields for Title, Name, Version, Icon, Start page, Query string, and Hosting type. The Title is 'Apps.Demo', Name is 'AppsDemo', Version is '1.0.0', Icon is 'Apps.Demo/Images/AppIcon.png', Start page is 'Apps.Demo/Pages/Default.aspx', Query string is '{StandardTokens}', and Hosting type is 'SharePoint-hosted'.

AppManifest.xml* LinksPart.aspx Elements.xml* Default.aspx

The properties of the deployment package for your app are contained in the app manifest file. You can use the Manifest Designer to set or modify o

General Permissions Prerequisites Supported Locales Remote Endpoints

Use this page to set properties that identify and describe your app when it is deployed.

Title: Apps.Demo

Name: AppsDemo

Version: 1 0 0 0

Icon: Apps.Demo/Images/AppIcon.png Required size: 96 x 96 pixels

Start page: Apps.Demo/Pages/Default.aspx

Query string: {StandardTokens}

Hosting type: SharePoint-hosted

Permissions

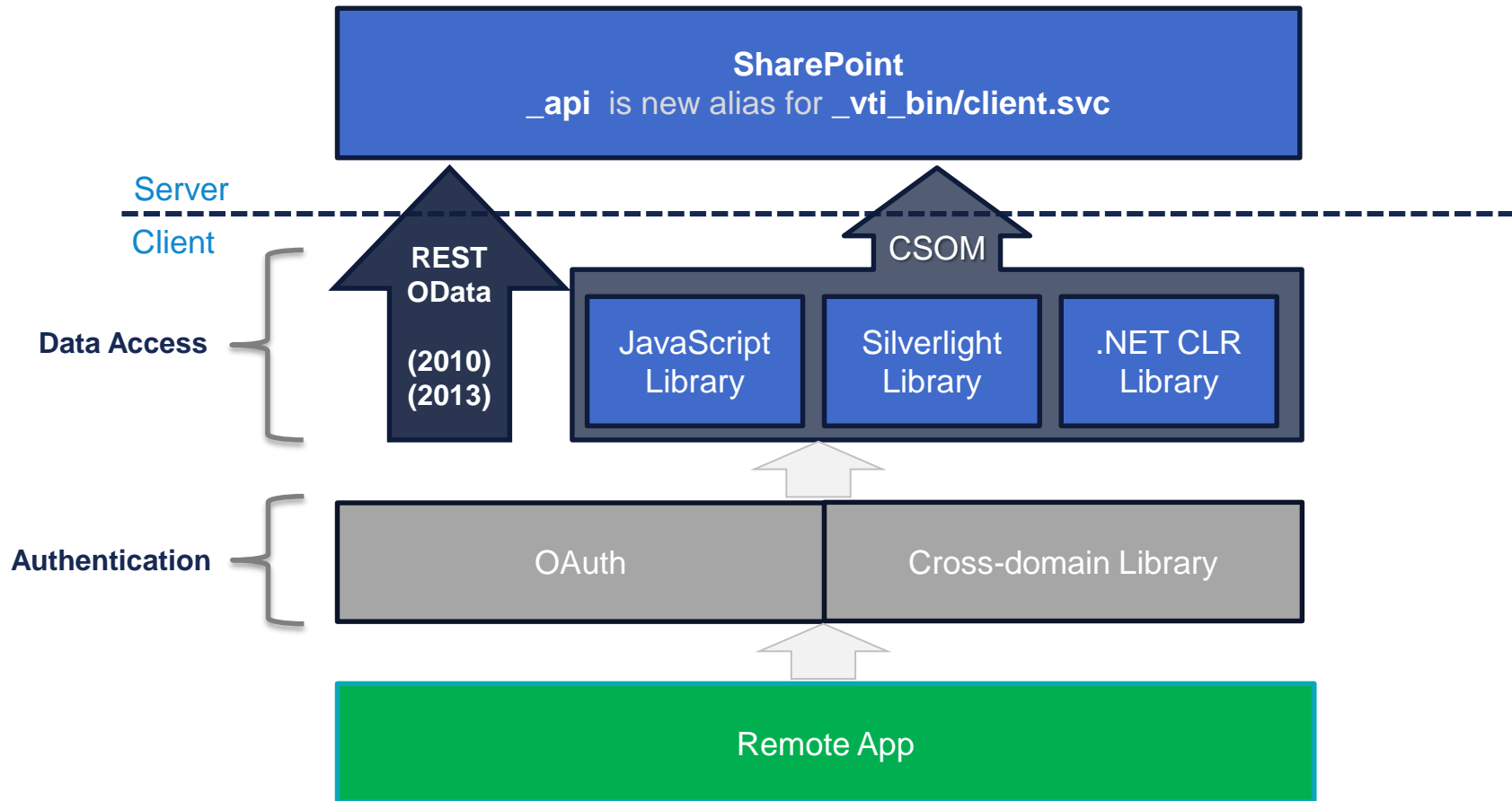
Developers

- App rights (Read, Write, Manage, Full Control)
- App scopes (Site, Web, List, Tenant, etc.)
- Authorization Policies (User Only, User + App Policy, App-only Policy)
- Cannot specify specific site artifacts (i.e. sites or lists)
- Can specify a specific list template

Users

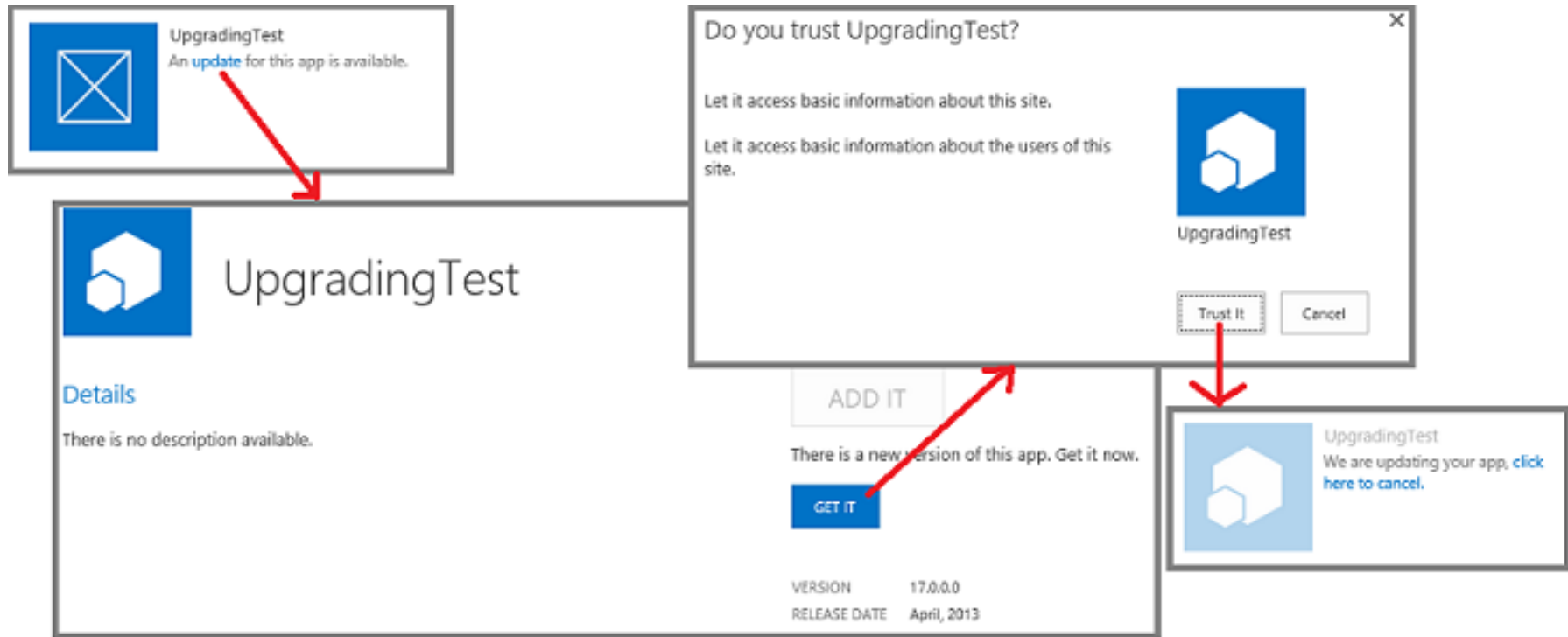
- Grant permissions when installing the app
- Cannot grant the app more permissions than the user herself has
- Permissions are all or nothing
- App identity passed around using OAuth tokens

SharePoint 2013 Remote API



Note: ASMX web services have been deprecated

Publish Apps for SharePoint



AngularJS Overview



ANGULARJS
by Google

AngularJS Main Features

Two-way Data Binding

MVC/MVVM

Dependency Injection

Testing

jqLite

Templates

Routing

History

Services/Factories/Providers

Modules

Services

Controllers

Expressions

Filters

Directives

Form Validation

Easy to Learn

Flexible

Similar to Silverlight

Backed by Google

ng-app

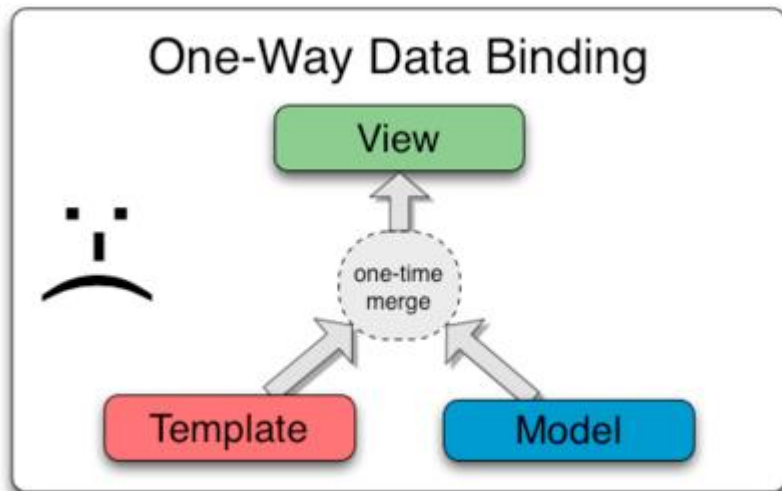
- Directive to bootstrap an Angular application

```
<html ng-app>
```

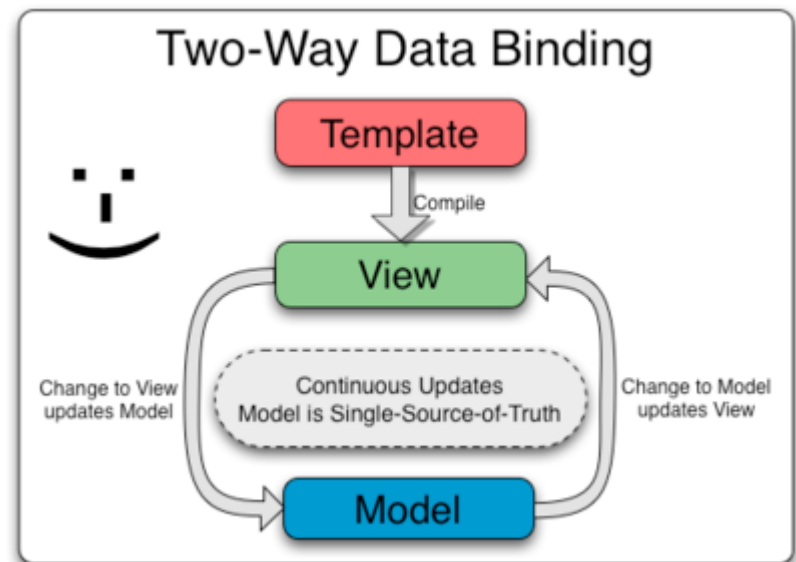
- Only one ng-app per page
- Need more ... manually bootstrap

```
angular.bootstrap(document.body, ['app1']);
```

Data Binding



Create complex objects.
Bind in one direction (merge).
Change to Model is not updated in View (and vice versa).
Additional code is required to keep Model and View in sync.



Use POJO's.
Template is compiled into a "live view".
Changes to View or Model are kept in sync.
The "glue" is \$scope.

Data Binding

```
<div ng-controller="DemoController">
  Search for President:
  <input ng-model="demo.name" />
  <h5>Search Term: {{demo.name}}</h5>

  <li ng-repeat="president in demo.presidents | filter:demo.name">
    {{president.name}}
  </li>

  <p>Number of Filtered Presidents: {{(demo.presidents | filter:demo.name).length}}</p>
  <p>Number Presidents: {{(demo.presidents).length}}</p>
</div>
```

Filters

- Formats the value of an expression displayed to users

`ng-repeat="president in demo.presidents | filter:demo.name"`

- OOTB examples include:
 - currency, date, filter, json, limitTo, lowercase, number, orderBy, uppercase
- Create custom ones as well
- Chain multiples using |

Controllers

- Use to:
 - Set the initial state of \$scope
 - Add behavior to the \$scope object (i.e. business logic).
- Do not use to:
 - Manipulate the DOM. Use data binding or directives.
 - Communicate between controllers. Use services.
 - Filter objects. Use filters.

Controllers

```
var controllerId = 'DemoController';
demoApp.controller(controllerId, ['$scope', DemoController]);

function DemoController($scope) {
    $scope.demo = {};
    $scope.demo.name = '';

    $scope.demo.presidents = [↔];
}
```

Basic AngularJS Demo Walkthrough

<http://jsfiddle.net/spietrek/VKbpu/>

Providers

- Key concepts:
 - Keep data around for lifetime of application
 - Separation of concerns
 - Singleton objects
 - Lazy loaded when needed

Constants

Values

Factories

Services

Providers

Directives

- Allow you to extend HTML vocabulary for your application.
 - Function that runs when the compiler encounters it in DOM.
 - Extend attributes, elements, classes, comments
- Commonly used options:
 - restrict, replace, template or templateUrl, link
 - link function has scope, element and attributes visibility
 - compile function performs any DOM transformation before link

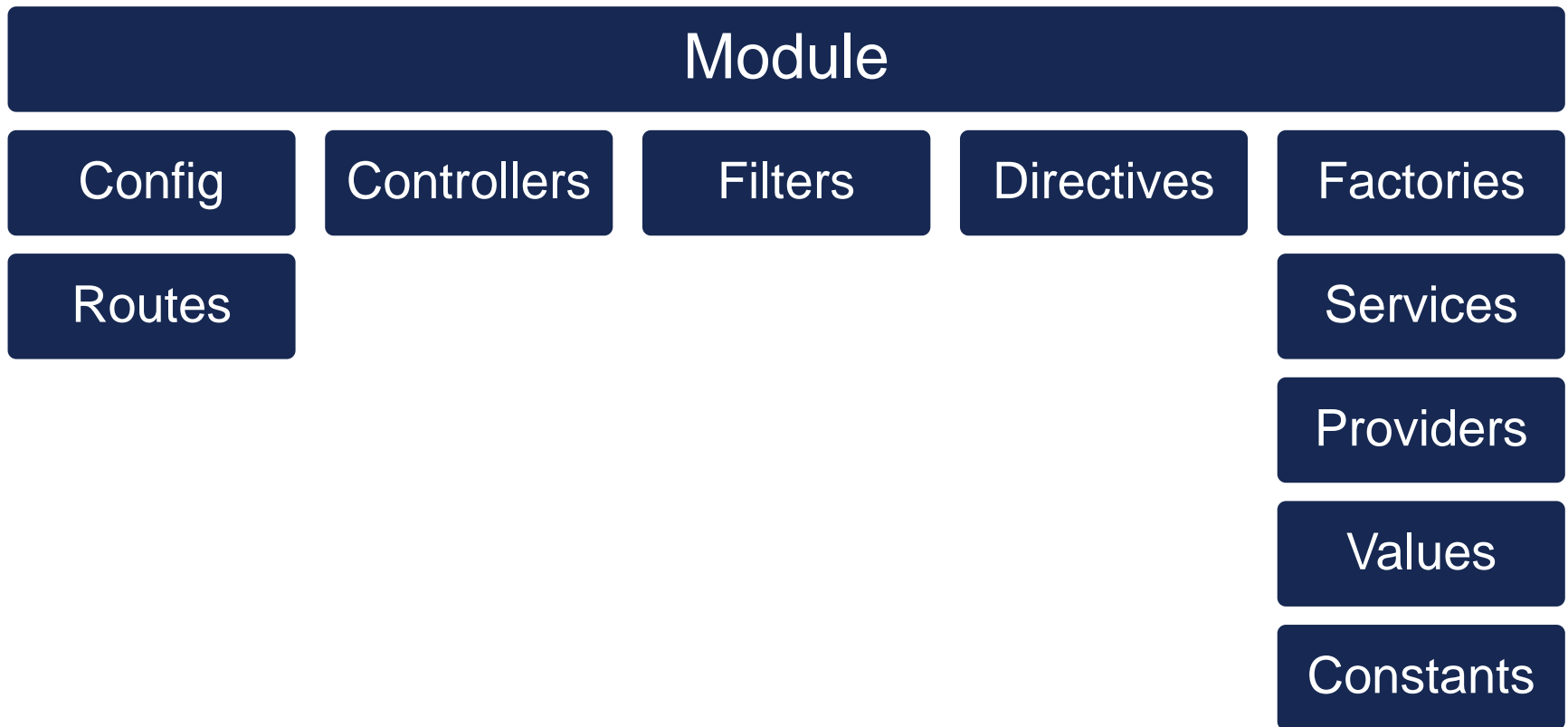
DISCLOSURE: Most complex area in AngularJS. Important to learn for code reuse and separation of concerns.

Directives

```
app.directive('csgLinks', function() {  
    return {  
        restrict: 'A',  
        replace: true,  
        scope: {  
            openList: '=openList',  
            refreshData: '=refreshData',  
            webPartTitle: '=webPartTitle',  
            links: '=links',  
            linksCount: '=linksCount',  
            errorMessage: '=errorMessage'  
        },  
        /*link: function (scope, element, attrs) {  
  
        },*/  
        templateUrl: function(element, attrs) {  
            return attrs.templateUrl;  
        }  
    };  
});
```

Modules

- Organize your code. Similar to .NET namespaces.



SharePoint 2013 App Model Walkthrough

<https://github.com/spietrek/SPSDurham2014>

Best Practices

Best Practices

Line of Business (LoB) Apps

Avoid many app parts on single page

Structure Code using JavaScript frameworks

Allow browser to cache files

Assign minimum app permissions

Filter data to reduce data loads

Familiarize on Fiddler and Browser Developer Tools

Test in multiple browsers

Version Control

References

- <https://docs.angularjs.org/guide>
- <http://www.youtube.com/watch?v=i9MHigUZKEM>
- <http://www.youtube.com/watch?v=8ILQOFAgaXE>
- [http://msdn.microsoft.com/en-us/library/office/fp179930\(v=office.15\).aspx](http://msdn.microsoft.com/en-us/library/office/fp179930(v=office.15).aspx)
- [http://msdn.microsoft.com/en-us/library/office/jj163816\(v=office.15\).aspx](http://msdn.microsoft.com/en-us/library/office/jj163816(v=office.15).aspx)
- <http://www.jeremythake.com/2014/01/sharepoint-rest-api-to-host-web-with-angularjs-services/>



**Thank you for
joining us Today!**

Don't Forget SharePint

Join us right after the event at **Tyler's Restaurant & Taproom!** Socialize and unwind after our day of learning.

324 Blackwell St, Durham, NC 27701

Steve Pietrek spietrek@cardinalsolutions.com [@spietrek](https://twitter.com/spietrek)