Building SharePoint 2013 Apps

Steve Pietrek
Cardinal Solutions
@spietrek

cardinal
PEOPLE.IDEAS.SOLUTIONS.

Durham Technical Community College

Lucidius

KnowledgeLake®

SPS EVENTS
Durham

K2®

LOAF
L

KWizCom
Knowledge Worker Components

Metalogix

Ninth Street Bakery

AvePoint®

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 being done in custom code (Elevated Privileges)
- Biggest cause of SharePoint support cases: custom code

Developer
- Design, build and test customizations

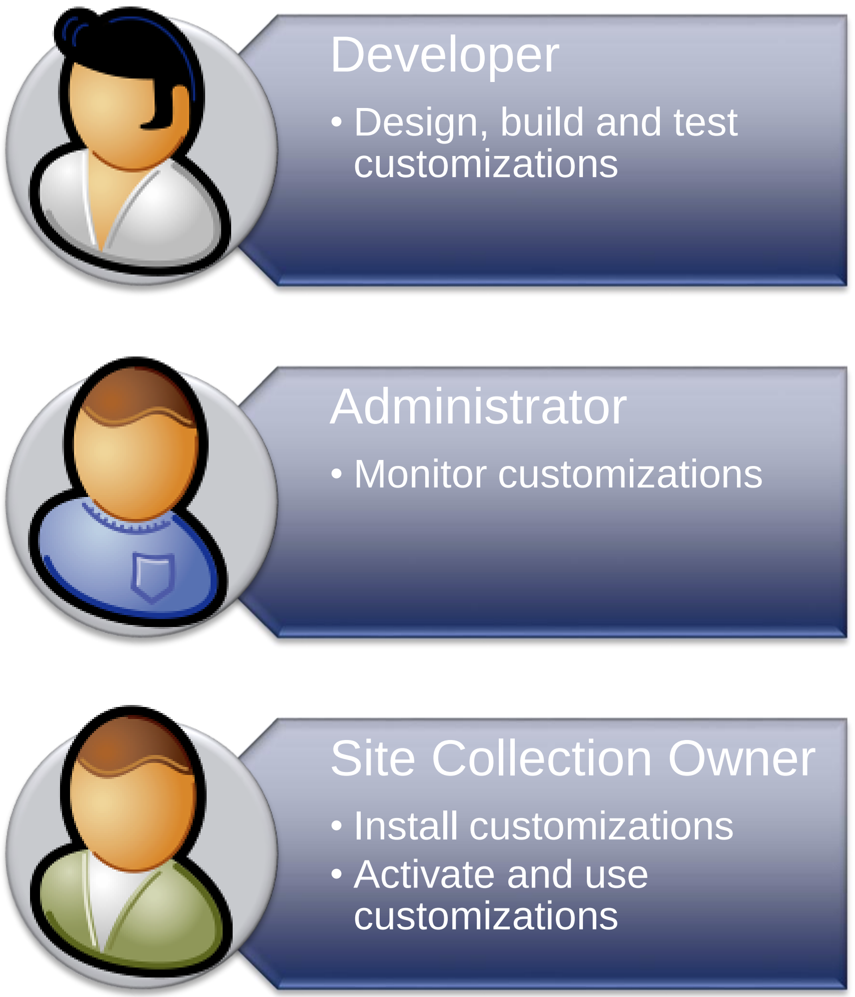
Administrator
- Install and monitor customizations

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
- Missing functionality in Sandbox Solutions.



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

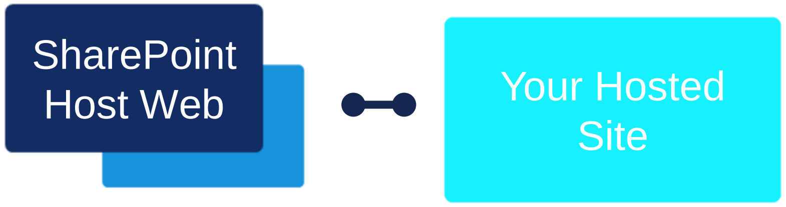
Hosting Options

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

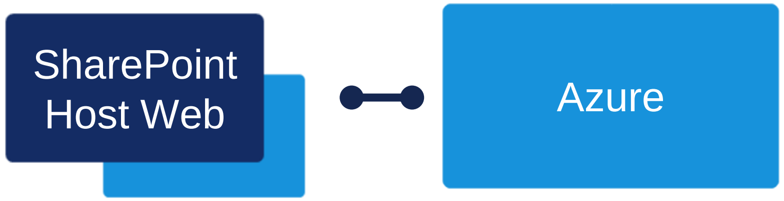
Provider-hosted app

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



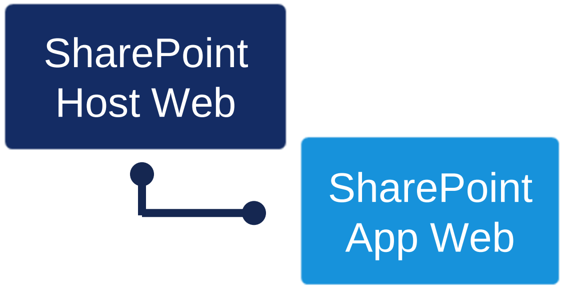
Autohosted app

Windows Azure + SQL Azure provisioned automatically as apps are installed



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 (cross-domain)



App Shapes

	Shape	Description
	Full Page	App that implements a new business scenario (LOB). Common term is Single Page Application (SPA).
	App Part	Provides new web parts you can add to your sites
	Extension App (Custom Action)	Add new actions for documents and items to the ribbon or menu

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

- Single Page Application
- 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
- Slower 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)

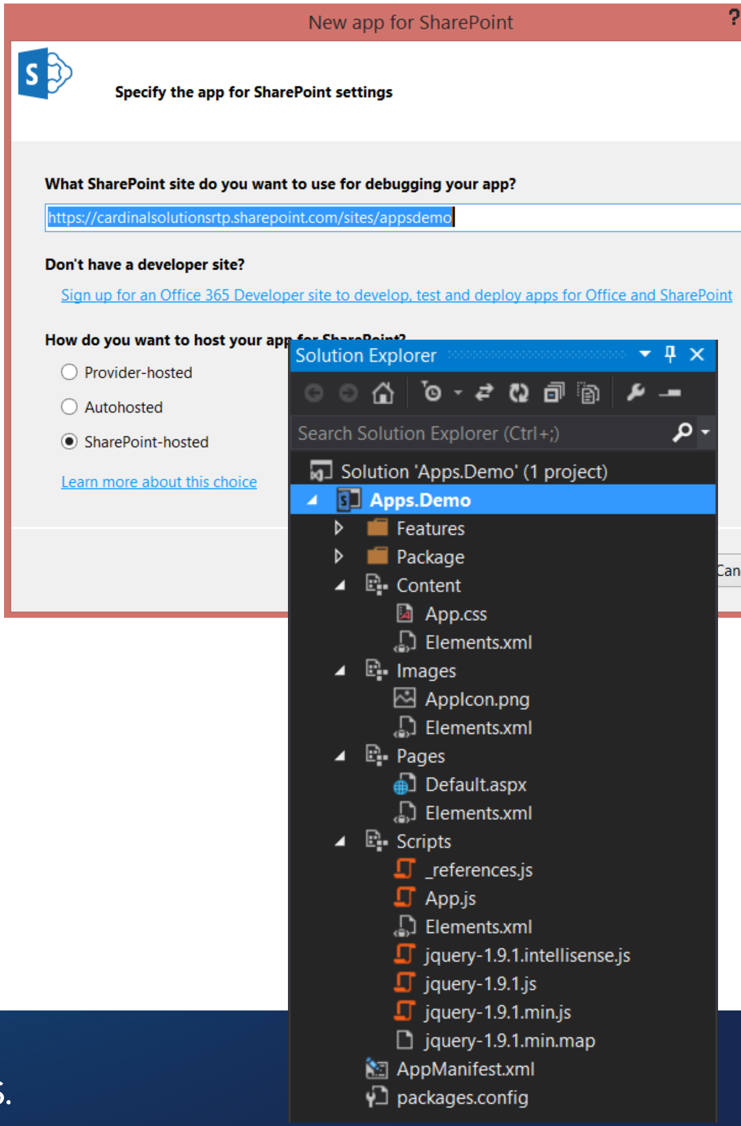
Lessons Learned

- 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


New app for SharePoint

Specify the app for SharePoint settings

What SharePoint site do you want to use for debugging your app?

https://cardinalsolutionsrtp.sharepoint.com/sites/appsdemo

Don't have a developer site?

Sign up for an Office 365 Developer site to develop, test and deploy apps for Office and SharePoint

How do you want to host your app for SharePoint?

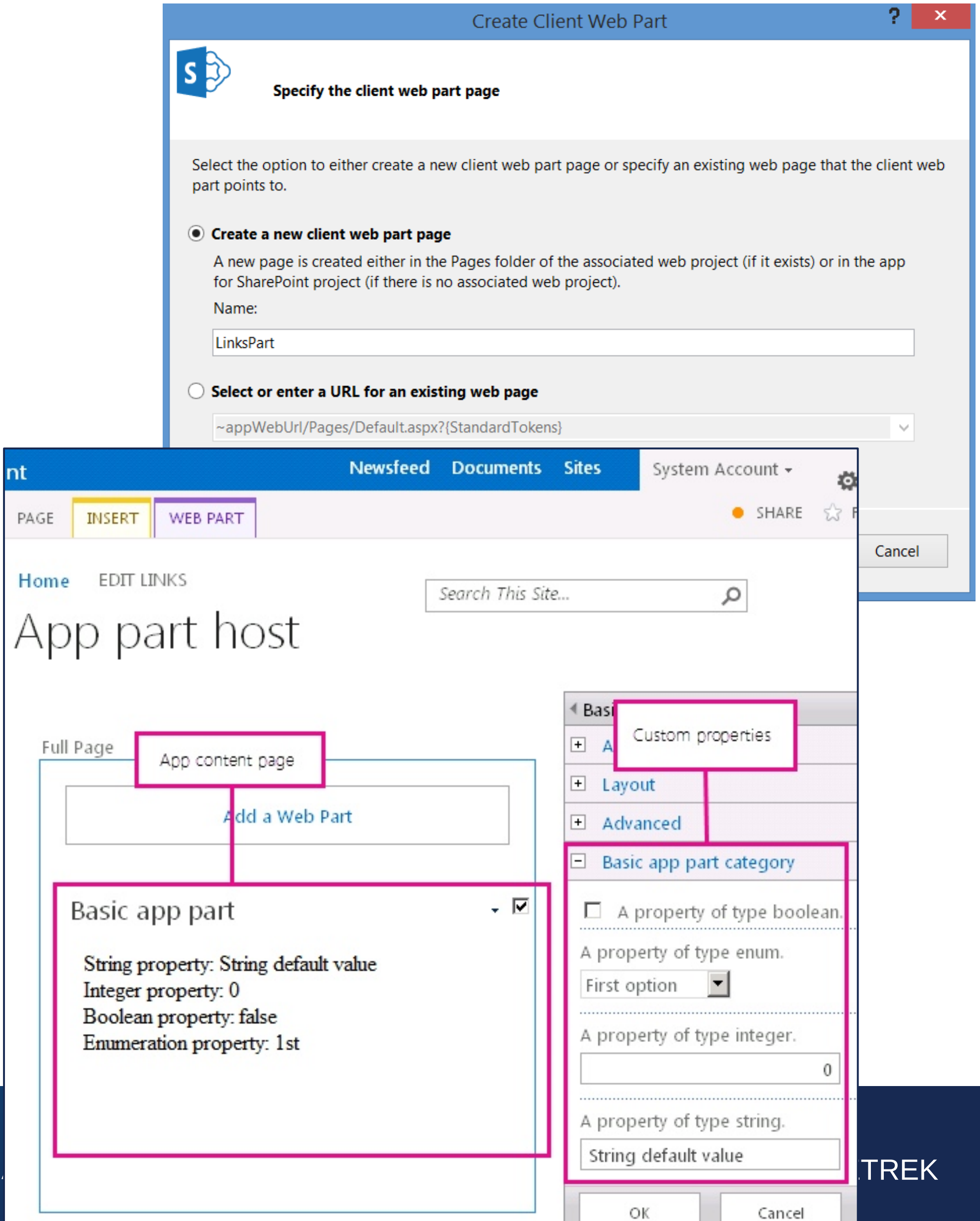
( ) Provider-hosted
( ) Autohosted
(●) SharePoint-hosted

Learn more about this choice

```
Solution 'Apps.Demo' (1 project)
  Apps.Demo
    Features
    Package
    Content
      App.css
      Elements.xml
    Images
      AppIcon.png
      Elements.xml
    Pages
      Default.aspx
      Elements.xml
    Scripts
      _references.js
      App.js
      Elements.xml
      jquery-1.9.1.intellisense.js
      jquery-1.9.1.js
      jquery-1.9.1.min.js
      jquery-1.9.1.min.map
    AppManifest.xml
    packages.config
```

Client Web Parts (App Parts)

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



Create Client Web Part

Specify the client web part page

Select the option to either create a new client web part page or specify an existing web page that the client web part points to.

[✓ **Create a new client web part page**]

A new page is created either in the Pages folder of the associated web project (if it exists) or in the app for SharePoint project (if there is no associated web project).

Name: LinksPart

Select or enter a URL for an existing web page

~appWebUrl/Pages/Default.aspx?{StandardTokens}

Basic app part

String property: String default value
Integer property: 0
Boolean property: false
Enumeration property: 1st

Basic app part category

A property of type boolean.

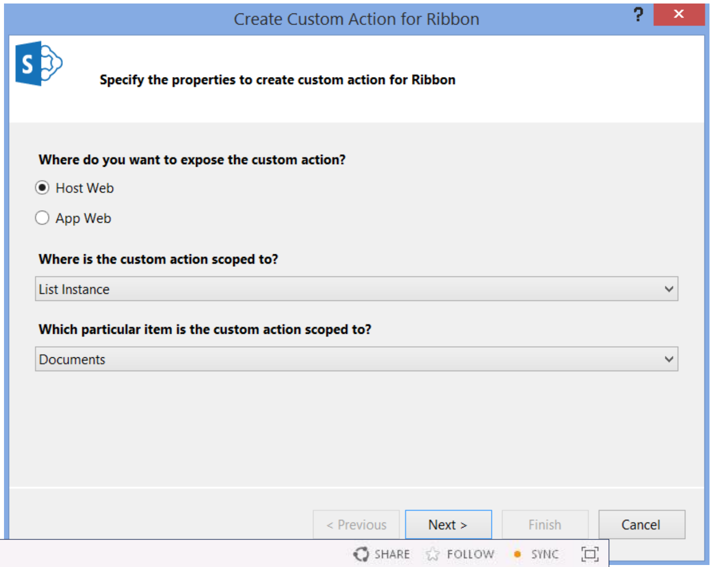
A property of type enum. First option

A property of type integer. 0

A property of type string. String default value

UI Custom Actions

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



Create Custom Action for Ribbon

Specify the properties to create custom action for Ribbon

Where do you want to expose the custom action?

(●) Host Web
( ) App Web

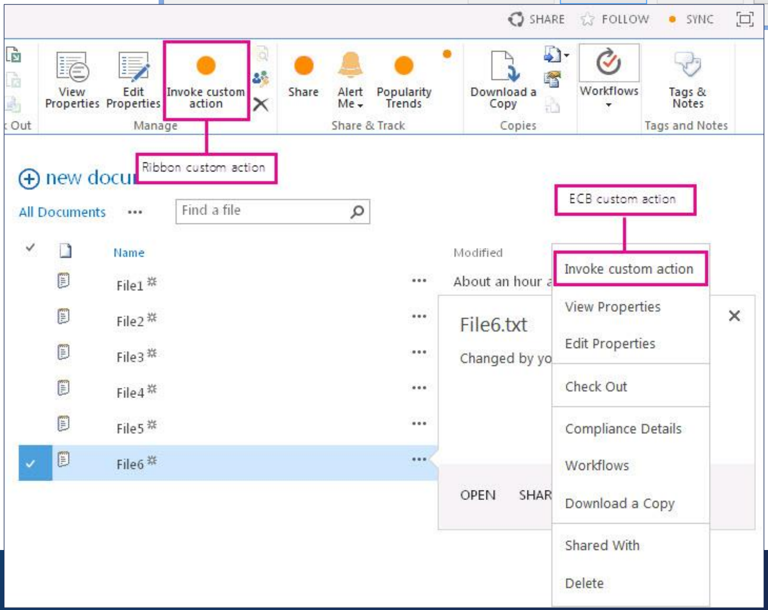
Where is the custom action scoped to?

List Instance

Which particular item is the custom action scoped to?

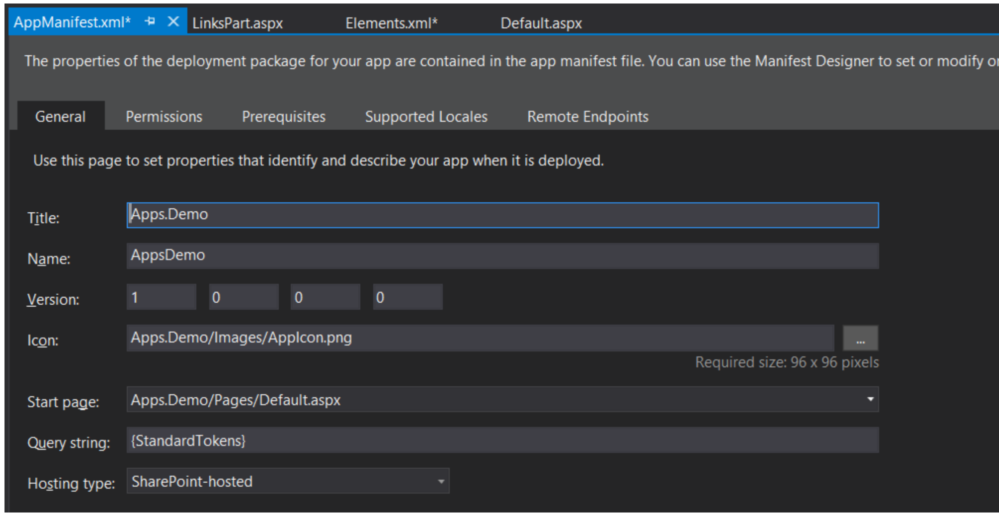
Documents

< Previous | Next > | Finish | Cancel



App Manifest

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



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

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

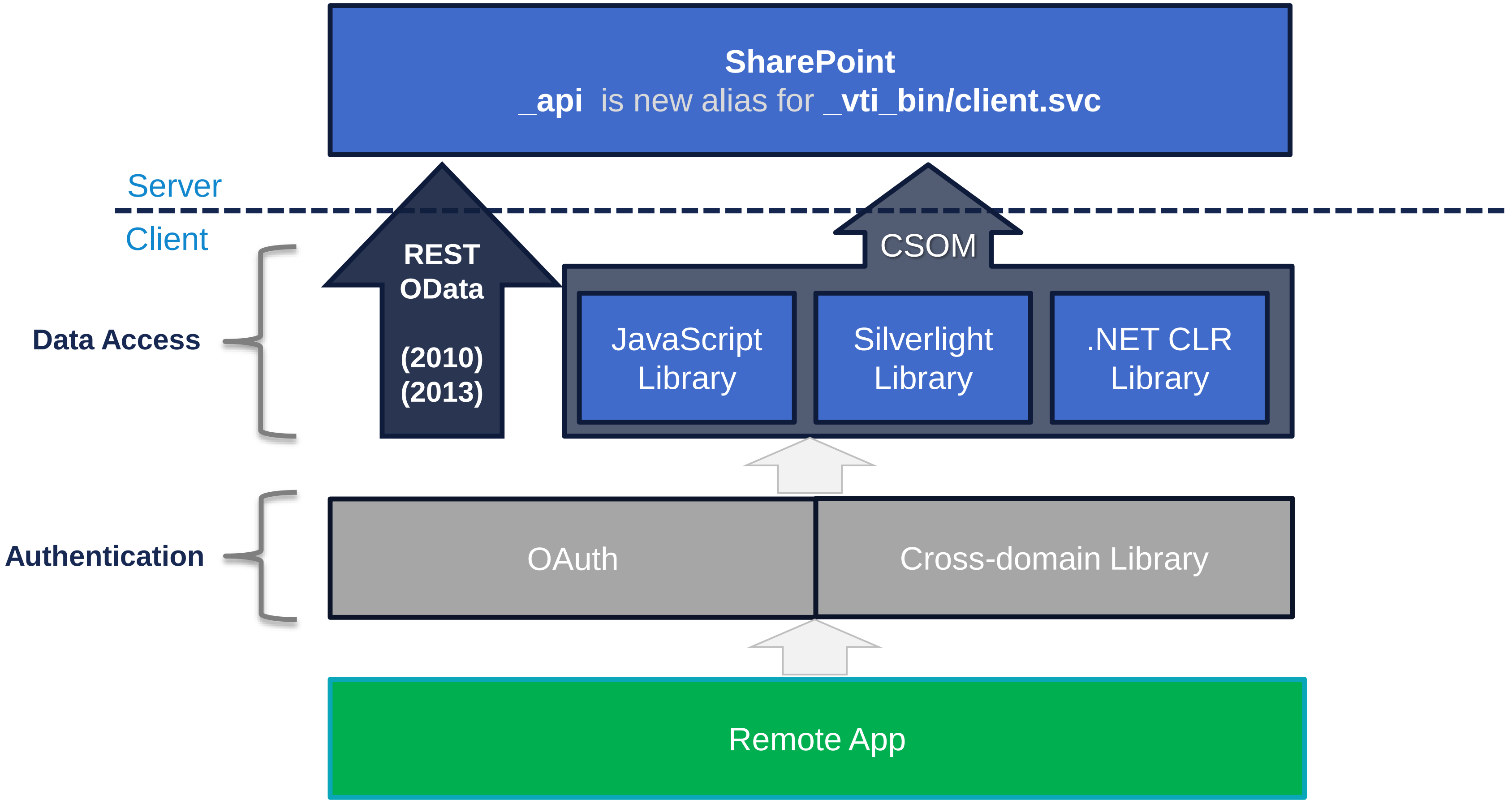
Developers

- 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

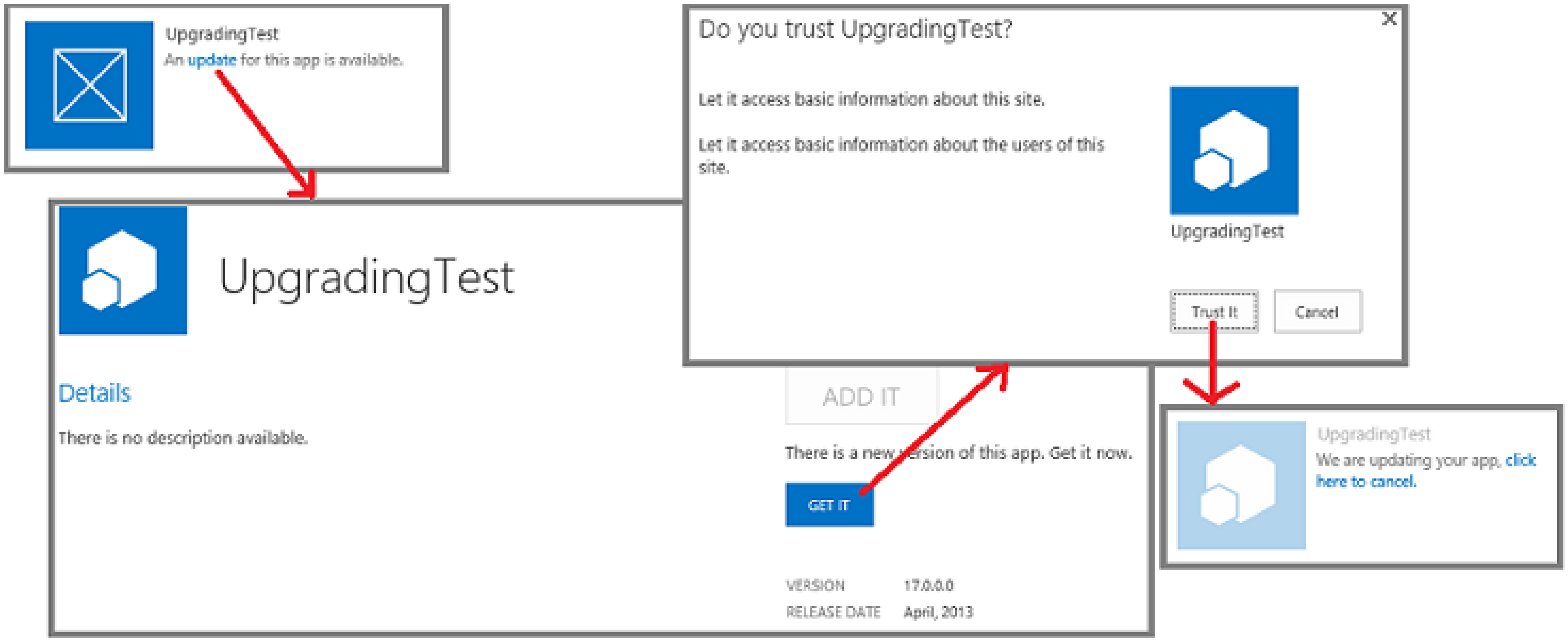
- Grant permissions when installing the app
- Cannot grant the app more permissions than the user herself has
- Permissions are all or nothing
- App identity passed around using OAuth tokens

SharePoint 2013 Remote API



Note: ASMX web services have been deprecated

Publish Apps for SharePoint


UpgradingTest
An update for this app is available.

UpgradingTest

Details

There is no description available.

ADD IT

There is a new version of this app. Get it now.

GET IT

VERSION	17.0.0.0
RELEASE DATE	April, 2013

Do you trust UpgradingTest?

Let it access basic information about this site.

Let it access basic information about the users of this site.

UpgradingTest

Trust It | Cancel

UpgradingTest
We are updating your app, click here to cancel.

Note: Apps do not automatic

AngularJS Overview

ANGULARJS

by Google

AngularJS Main Features

Backed by Google	Great for SPA's	Two-way Data Binding	MV*	Dependency Injection
Testing	jqLite	Templates	Modules	Controllers
Expressions	Filters	Providers	Directives	Routing
History	Form Validation	Easy to Learn	Flexible	Similar to Silverlight

ng-app

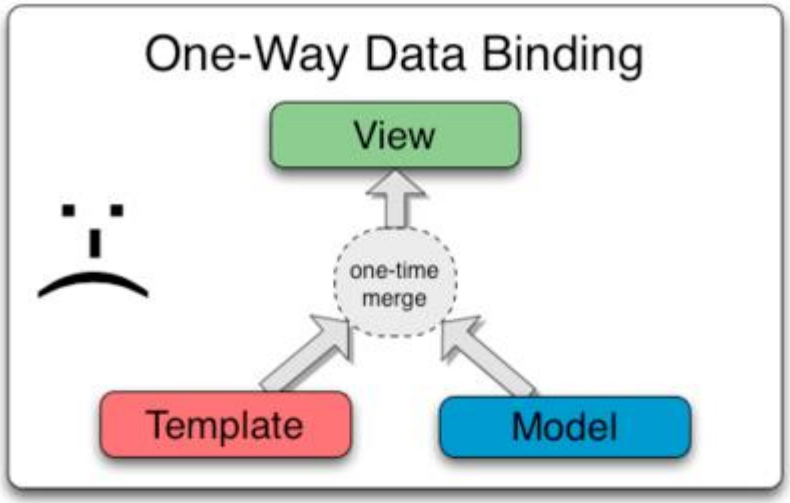
- Directive to bootstrap an Angular application

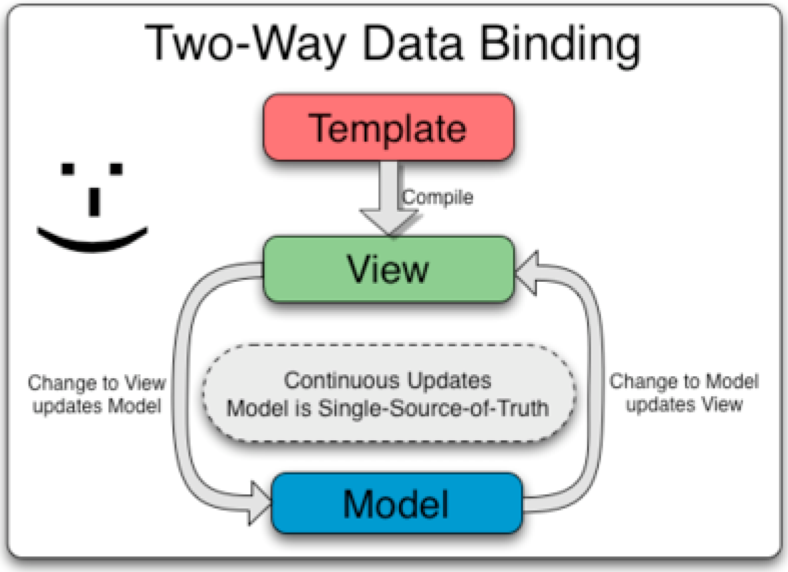
```
<html ng-app>
```

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

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

Data Binding




Two-Way Data Binding
Template
Compile
View
Change to View updates Model
Continuous Updates Model is Single-Source-of-Truth
Change to Model updates View
Model

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.

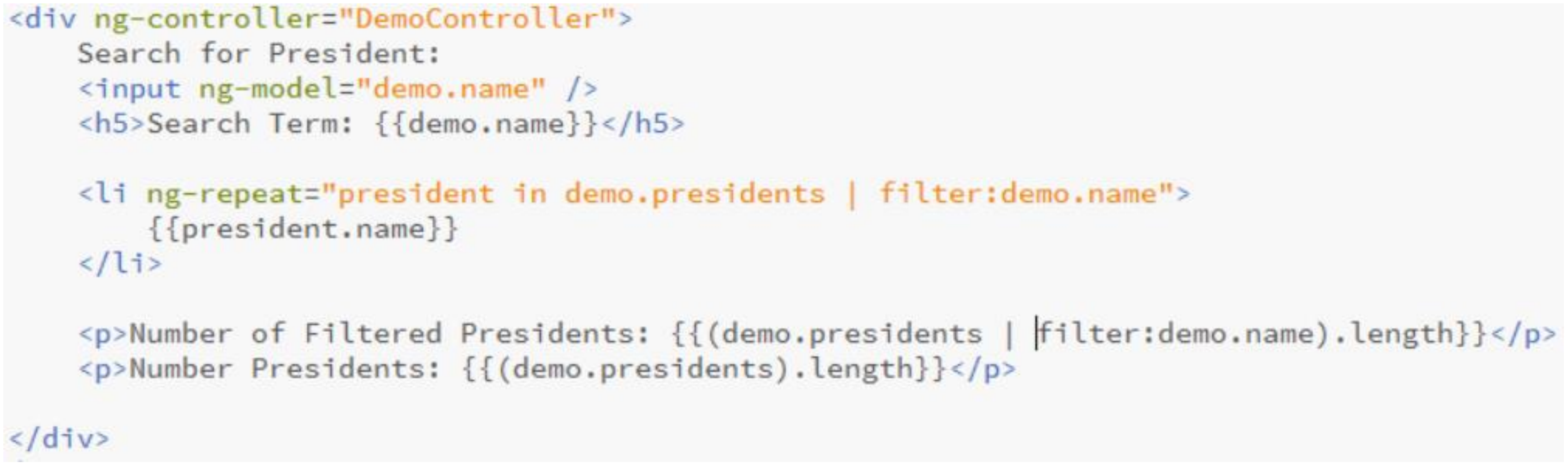
Use POJO’s.

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

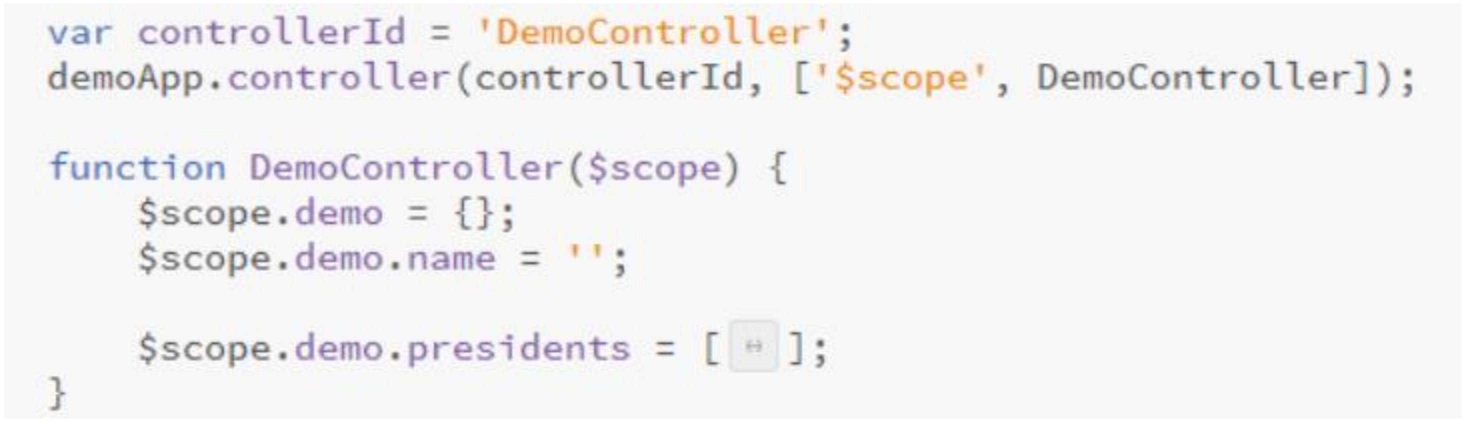
```
ng-repeat="president in demo.presidents | filter:demo.name"
```

- 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

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

Providers

- Key concepts:
  - Keep data around for lifetime of application
  - Separation of concerns
  - Singleton objects
  - Lazy loaded when needed
  - Logical place to access external data ($HTTP or $RESOURCE)

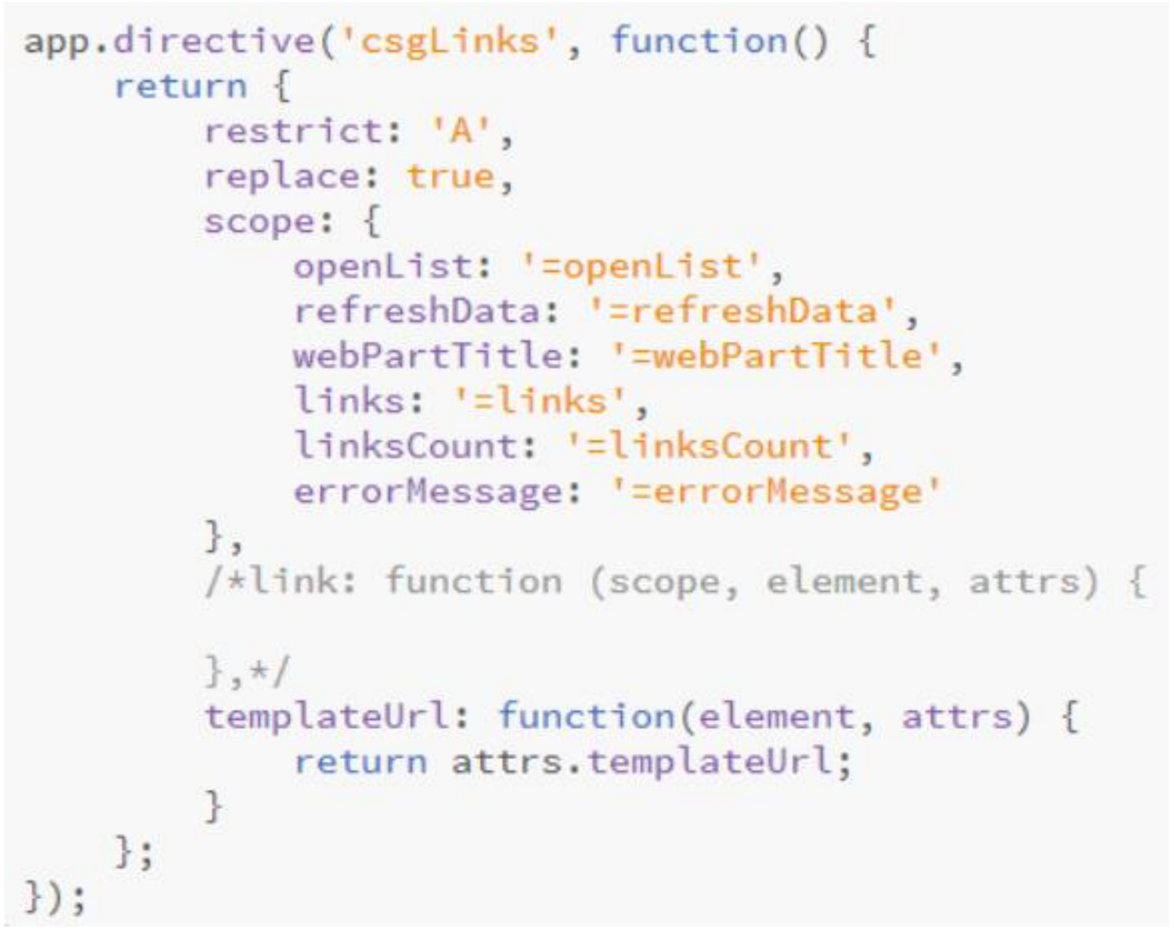
Constants	Values	Factories	Services	Providers

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

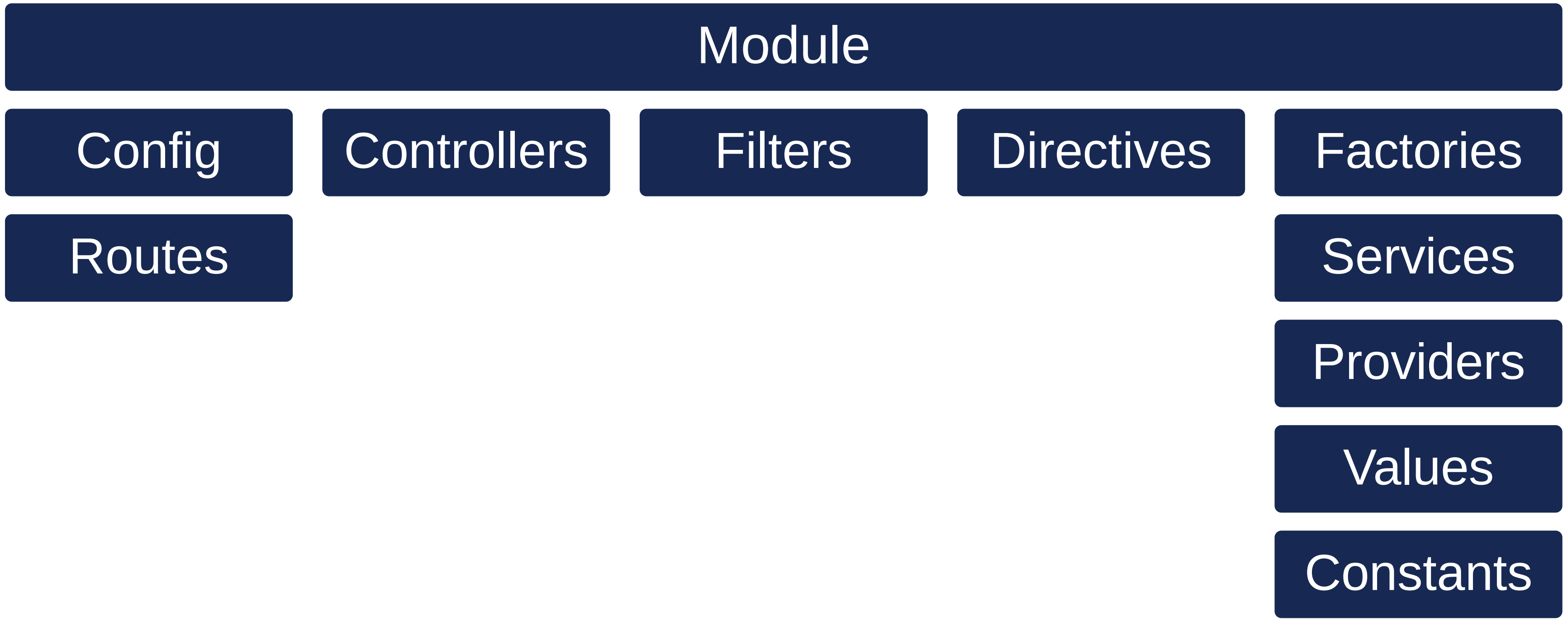


```
app.directive('csgLinks', function() {
    return {
        restrict: 'A',
        replace: true,
        scope: {
            openList: '=openList',
            refreshData: '=refreshData',
            webPartTitle: '=webPartTitle',
            links: '=links',
            linksCount: '=linksCount',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

        },*/
        templateUrl: function(element, attrs) {
            return attrs.templateUrl;
        }
    };
});
```

Modules

- Organize your code. Similar to .NET namespaces.
- Include multiple Controllers, Filters, Directives, and Providers.



SharePoint 2013 App Model Demo

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/jj163816(v=office.15).aspx
- http://www.jeremythake.com/2014/01/sharepoint-rest-api-to-host-web-with-angularjs-services/

SPS EVENTS
DURHAM 2014
SharePoint SATURDAY
#SPS EVENTS
@SPSDURHAM

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**