CitiusTech

Introduction to AngularJS

May 2015

Agenda

- **History**
- Single Page Application
- Key Challenges Before AngularJS
- What is AngularJS?
- AngularJS Complete Client Side Solution
- AngularJS Core Design Principles
- AngularJS Core Building Blocks
- Demo On Basic Example

History

Overall journey of web applications, from server-side frameworks to client-side frameworks is as follows:

- Traditional web application architecture using ASP.net , JSP
- Wide use of server side frameworks
- Focus shifted from server to client - JavaScript
- Need for rich Application UI
- Use of AJAX
- Jquery - Scripts which can run on all browsers uniformly
- Introduction to SPA
- Client side frameworks. for e.g.: AngularJs

Agenda

- History
- **Single Page Application**
- Key Challenges Before AngularJS
- What is AngularJS?
- AngularJS Complete Client Side Solution
- AngularJS Core Design Principles
- AngularJS Core Building Blocks
- Demo On Basic Example

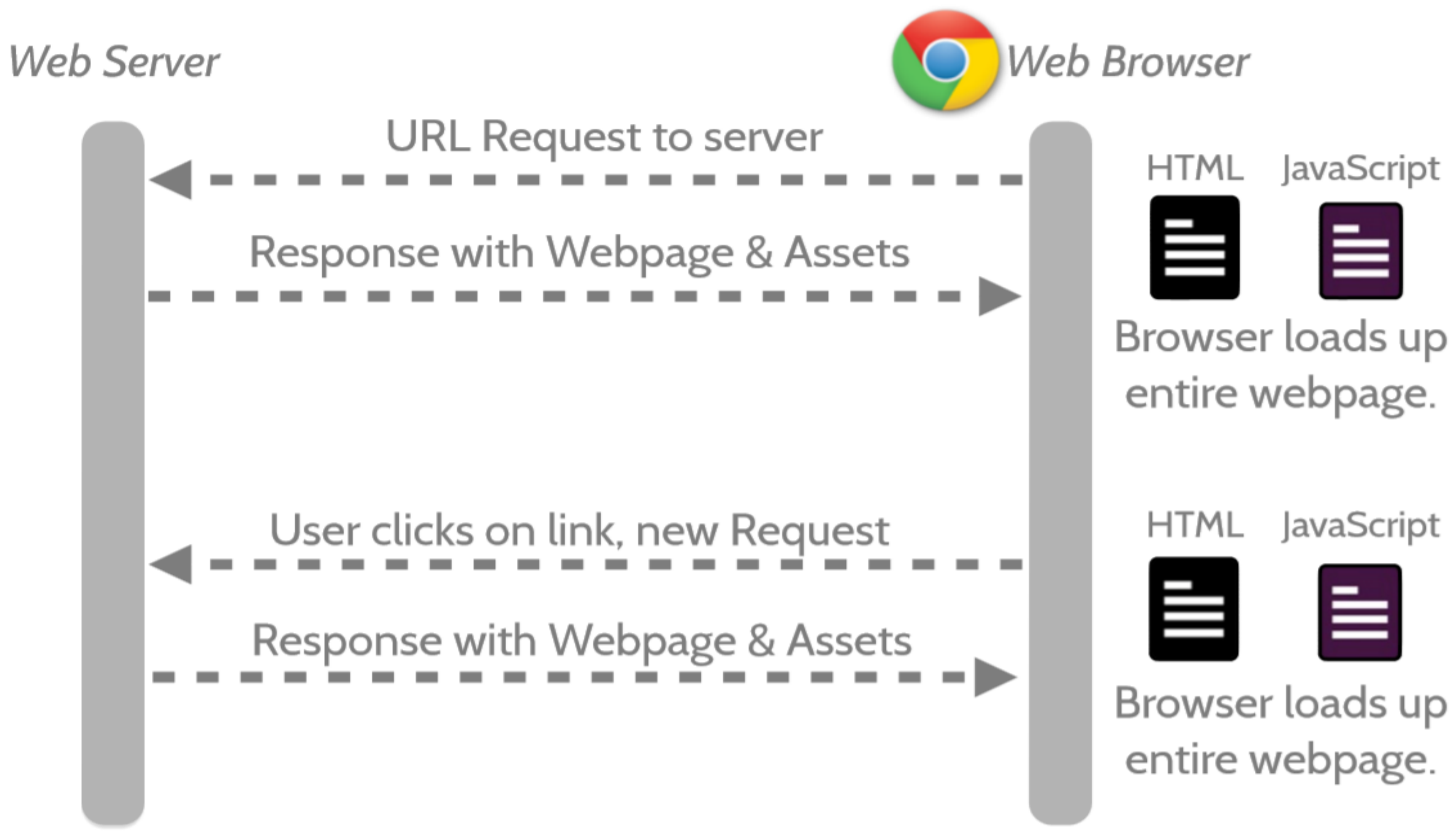
Single Page Application (SPA) (1/3)

Before SPA – Features Of Traditional Web Applications

- It had many web pages, each hyperlink on home page loaded new web page in the browser
- Required full page refresh cycle with GET/POST request to server
- Used server side routing with navigation between pages via server (servlet controller)
- All resultant html snippets/templates generated from server side
- Required redundant loading of libraries on every new page load
- Resulted in high amount of network traffic (data flow between UI & server)

Single Page Application (SPA) (2/3)

Before SPA – Traditional Web Page Refresh Cycle


Web Server
Web Browser
URL Request to server
Response with Webpage & Assets
HTML
JavaScript
Browser loads up entire webpage.
User clicks on link, new Request
Response with Webpage & Assets
HTML
JavaScript
Browser loads up entire webpage.

Single Page Application (SPA) (3/3)

What is SPA?

- It is a web application that fits on a **single web page**
- It provides a more fluid user experience similar to a desktop application
- Uses client side routing with navigation logic coded on client side
- Avoids round trip to the server for every request, as all view related logic and html templates are present at client side only
- Every hyperlink on home/main page, loads new html template onto the main page
- All libraries on main page are loaded only once at the start
- Most of the data-flow is to & from Client side UI-Components. Hence, results in less amount of network traffic

Agenda

- History
- Single Page Application
- **Key Challenges Before AngularJS**
- What is AngularJS?
- AngularJS Complete Client Side Solution
- AngularJS Core Design Principles
- AngularJS Core Building Blocks
- Demo On Basic Example

Key Challenges Before AngularJS

Following were the key challenges faced in developing web application before AngularJS:

- Manipulating Html DOM programmatically – Had to write lot of code to fetch data from DOM elements and to update the DOM
- Lot of boilerplate code has to be written just to get started (e.g. using Ajax in JavaScript)
- Messy Code resulting from nested call-backs and registering call-backs
- No clear cut division of responsibility resulting in tight coupling between:
  - View and Model
- Hard coding of component dependencies resulting in tight coupling between:
  - Presentation logic and Service layer components

Agenda

- History
- Single Page Application
- Key Challenges Before AngularJS
- **What is AngularJS?**
- AngularJS Complete Client Side Solution
- AngularJS Core Design Principles
- AngularJS Core Building Blocks
- Demo On Basic Example

What is AngularJS?

- Open source JS framework by Google & community
- AngularJS is one core library
- Supports creation of Single Page Application (SPA)
- Gives MV* capability to Single page web application (SPA)
- Framework based on good design principles.

Agenda

- History
- Single Page Application
- Key Challenges Before AngularJS
- What is AngularJS?
- **AngularJS Complete Client Side Solution**
- AngularJS Core Design Principles
- AngularJS Core Building Blocks
- Demo On Basic Example

AngularJS – A Complete Client Side Solution

- Before AngularJS, developers used following libraries/frameworks:

Feature	Library
Client-based routing	BackboneJS, Sammy JS
DOM Manipulation	jQuery
Two way binding	KnockoutJS

- With AngularJS, none of these libraries are needed as Angular provides support for all the above features
- Angular comes with a lightweight version of jQuery called jQLite
- Results in clean and concise code
- Can do mind-boggling things in just few lines of code
- Easy to develop web applications that are extensible and maintainable

Agenda

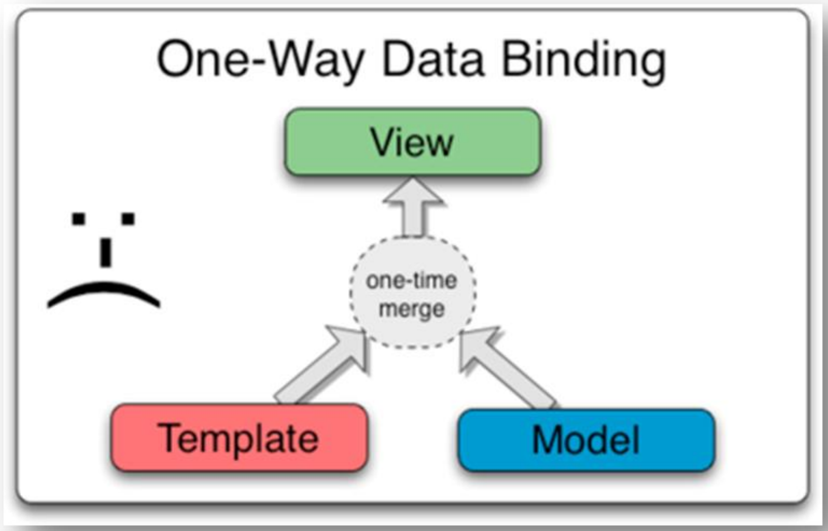
- History
- Single Page Application
- Key Challenges Before AngularJS
- What is AngularJS?
- AngularJS Complete Client Side Solution
- **AngularJS Core Design Principles**
- AngularJS Core Building Blocks
- Demo On Basic Example

AngularJS Core Design Principles (1/6)

- AngularJs works on following core design principles:
  - Two way data-binding
  - Separation Of Concern
  - Declarative approach – extends Html vocabulary by using directives
  - Dependency Injection (DI)

AngularJS Core Design Principles (2/6)

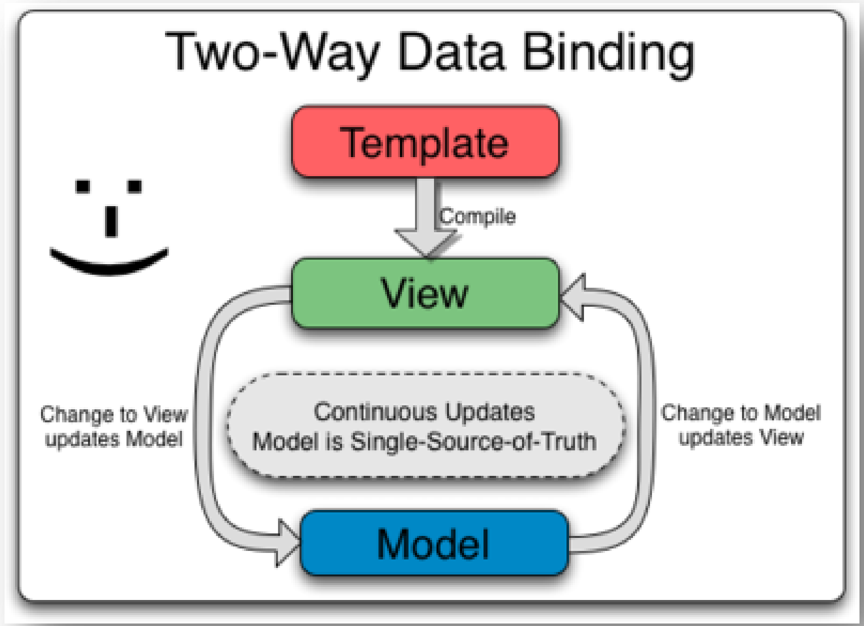
One-way or Classical data binding before AngularJS:



- One way data binding merges the template and model components together into a view
- View or Model are not updated automatically when the View or Model is changed
- Developers have to write extra code to sync the View and Model

AngularJS Core Design Principles (3/6)

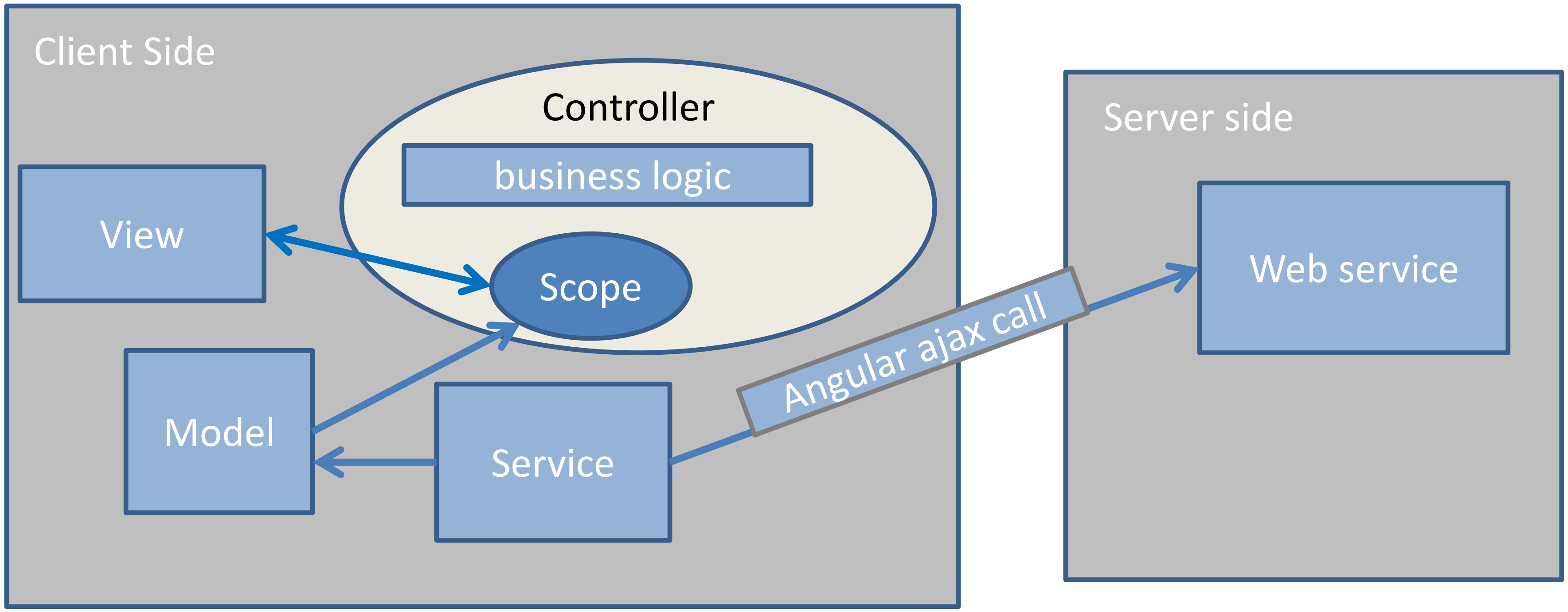
Two-way data binding of AngularJS:


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

- Two way data binding is the core of Angular’s magical spell
- View is updated automatically when the Model is changed
- Model is updated automatically when the value in the view has changed

AngularJS Core Design Principles (4/6)

Separation Of Concerns:



- Structured and Modularized code
- Well defined components with clear cut division of responsibility

AngularJS Core Design Principles (5/6)

Declarative UI:

- Declarative UI is a UI that's designed in a declarative way (you describe what it should be like) rather than an imperative way (you code the steps to create it.)
- Angular uses Html to define UI which is declarative language
- It adds ‘directives’ to Html
- Directives inform angular what should be done instead of we explicitly coding for it
- Directives are used as attributes of html element inside html page as shown below:
  - <div id="myid" ng-controller="DemoController"></div>

    ng-controller – is angular directive that informs angular to instantiate the Controller function and inject dependencies to it.

  - <body ng-app>

    ng-app – This directive auto-bootstraps angularjs application at body element

AngularJS Core Design Principles (6/6)

Dependency Injection (DI):

- Object is given it’s dependencies, rather than object creating them itself.
- Dependency is instantiated and injected by Angular when dependency name is mentioned as function argument inside controller or service function

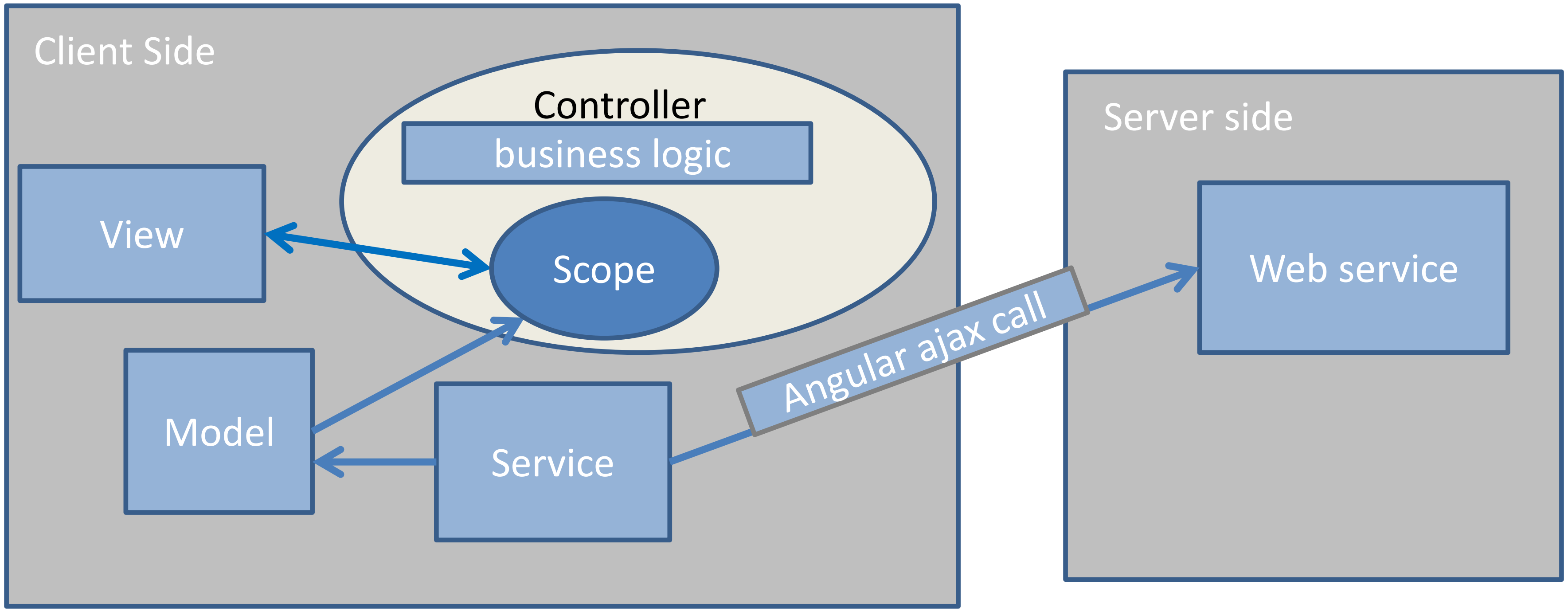
```
 //  ’calcService’ is dependency
        module.controllerFunction(calcService) {
        // once injected we can use it in normal way invoking
functions on it
                 calcService.add(3,5);
                     … }
```

- Promotes loose coupling by keeping dependency creation logic outside application.
- It is possible to change dependency without breaking the code that uses that dependency
- Allows injecting mock objects as dependencies in absence of actual dependency - making unit testing easy
- Angular has two important services which makes DI possible - **$injector** and **$provide** .

Agenda

- History
- Single Page Application
- Key Challenges Before AngularJS
- What is AngularJS?
- AngularJS Complete Client Side Solution
- AngularJS Core Design Principles
- **AngularJS Core Building Blocks**
- Demo On Basic Example

AngularJS Core Building Blocks



- Model
- View
- Controller
- Services
- Modules
- Filters

Agenda

- History
- Single Page Application
- Key Challenges Before AngularJS
- What is AngularJS?
- AngularJS Complete Client Side Solution
- AngularJS Core Design Principles
- AngularJS Core Building Blocks
- **Demo On Basic Example**

Demo On Basic Example

```
// Basic example that depicts two-way binding  in AngularJS
<!DOCTYPE html>
<html ng-app>
<head>    <title>Hello World, AngularJS </title>
<script type="text/javascript"
            src="http://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js">
</script>
</head>
<body>
<div>    Name:  <input type="text"  ng-model=“greet"/>
                 <h2> {{ greet }} </h2>
</div>
</body>
</html>
```

- Demo & Explain above example.
- Explain two-way binding, directives-ng-app, ng-model.

* Refer ‘Directives ppt’ for ng-app, ng-model

Demo Links

- Basic Demo that depicts ‘two-way binding’ in angularjs.

  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day1/day1_Demos/1_TwoWayBinding.html

Introduction To AngularJS - Finding your way (1/2)

Technical Questions:

1. What are the other UI frameworks popular in market
2. What is RequireJS? How can it be used with AngularJS?

As you start to work with AngularJS, you will frequently encounter technical issues which are not covered by this training.

How will you resolve these technical issues on AngularJS?

Introduction To AngularJS - Finding your way (2/2)

Resources	Remarks
http://viralpatel.net/blogs/angularjs-introduction-hello-world-tutorial/	Good blog post on Angularjs Introduction
https://docs.angularjs.org/guide/introduction	AngularJS official doc site
CurioCT - https://interct/SitePages/CurioCT.aspx	
CTCourses – Introduction to AngularJS	In addition to updated course material, CTCourse contains reference sites (Library) and list of project teams with expertise on ‘Introduction To AngularJS’

CurioCT - CitiusTech's Technology Q & A Forum

- In case of any questions please log on to https://interct/SitePages/CurioCT.aspx

Thank You