Angular Bootcamp

Naveen Pete

naveen.shenoy.blr@gmail.com

Saturday, Oct 22, 2016 (10 am – 6 pm)

Agenda

- Front-end JavaScript Frameworks
- Introducing Angular
- Angular Vocabulary
- Angular Building Blocks
- Directives
- Expressions, Data Binding
- MVC / MVVM
- Module
- Controller
- Filter
- Scope
- Forms
- Dependency Injection
- Service
- Template
- SPA
- Routing
- Client Server Communication - $http, $resource
- Custom Directives
- Q & A

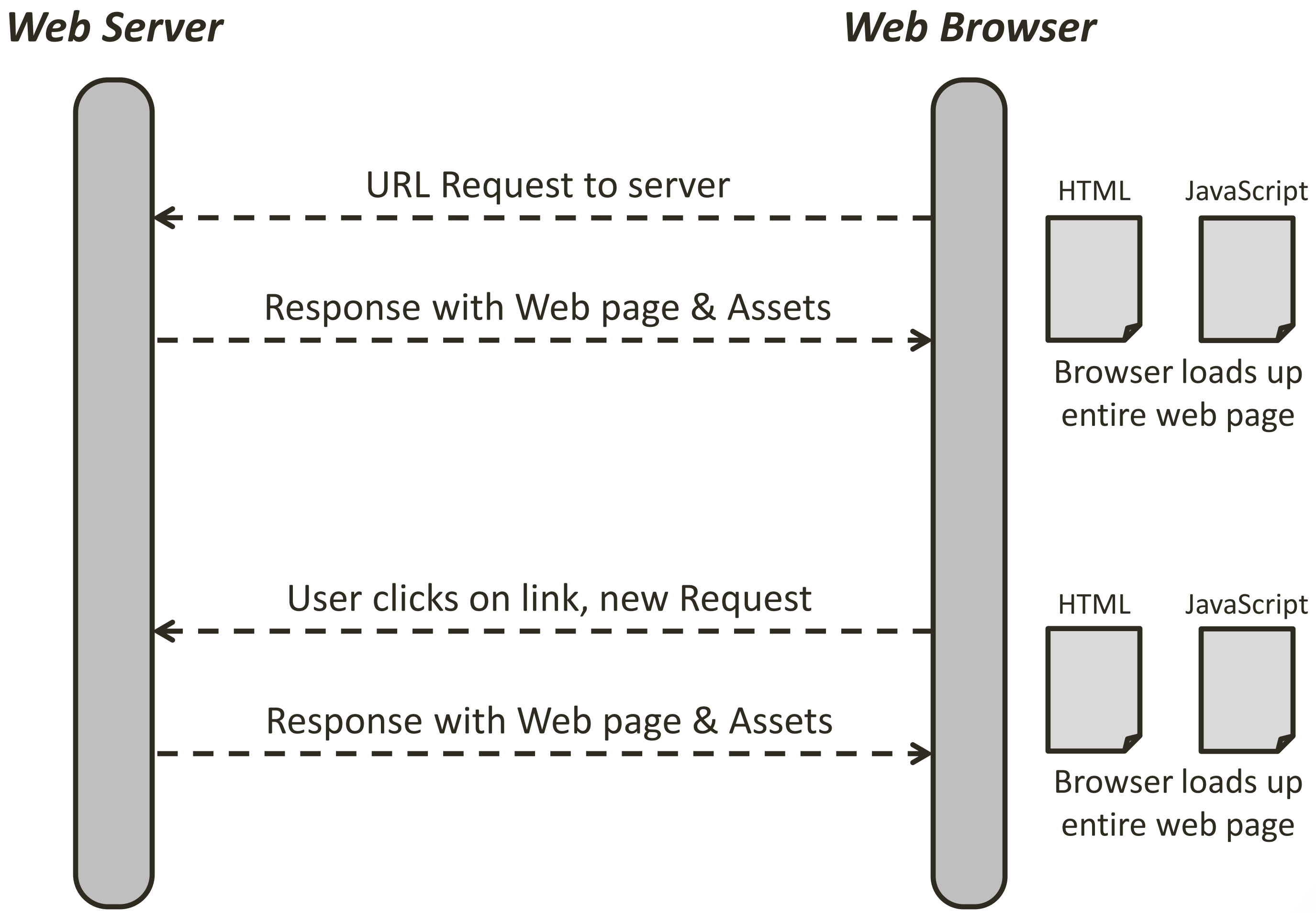
Need for JavaScript Frameworks

- HTML – good for static documents
- Dynamic Apps – DOM manipulation and data updates
- Impedance mismatch
- Library
  - A collection of reusable functions
  - Your code is in charge, calls into the library when it sees fit
  - E.g. jQuery
- Framework
  - A particular implementation of a web application
  - Provides generic functionality
  - Your code fills in details
  - Framework is in charge, it calls into your code, when it needs something
  - E.g. Backbone, Ember, Meteor

Need for JavaScript Frameworks

- Benefits:
  - Abstracts complexities of development
  - Increases developer productivity
  - Requires less in-depth expertise
  - Moving the application code forward in the stack
    - Reduces server load, thus reducing cost
    - Crowd-sourcing of computational power

Traditional Page Refresh


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

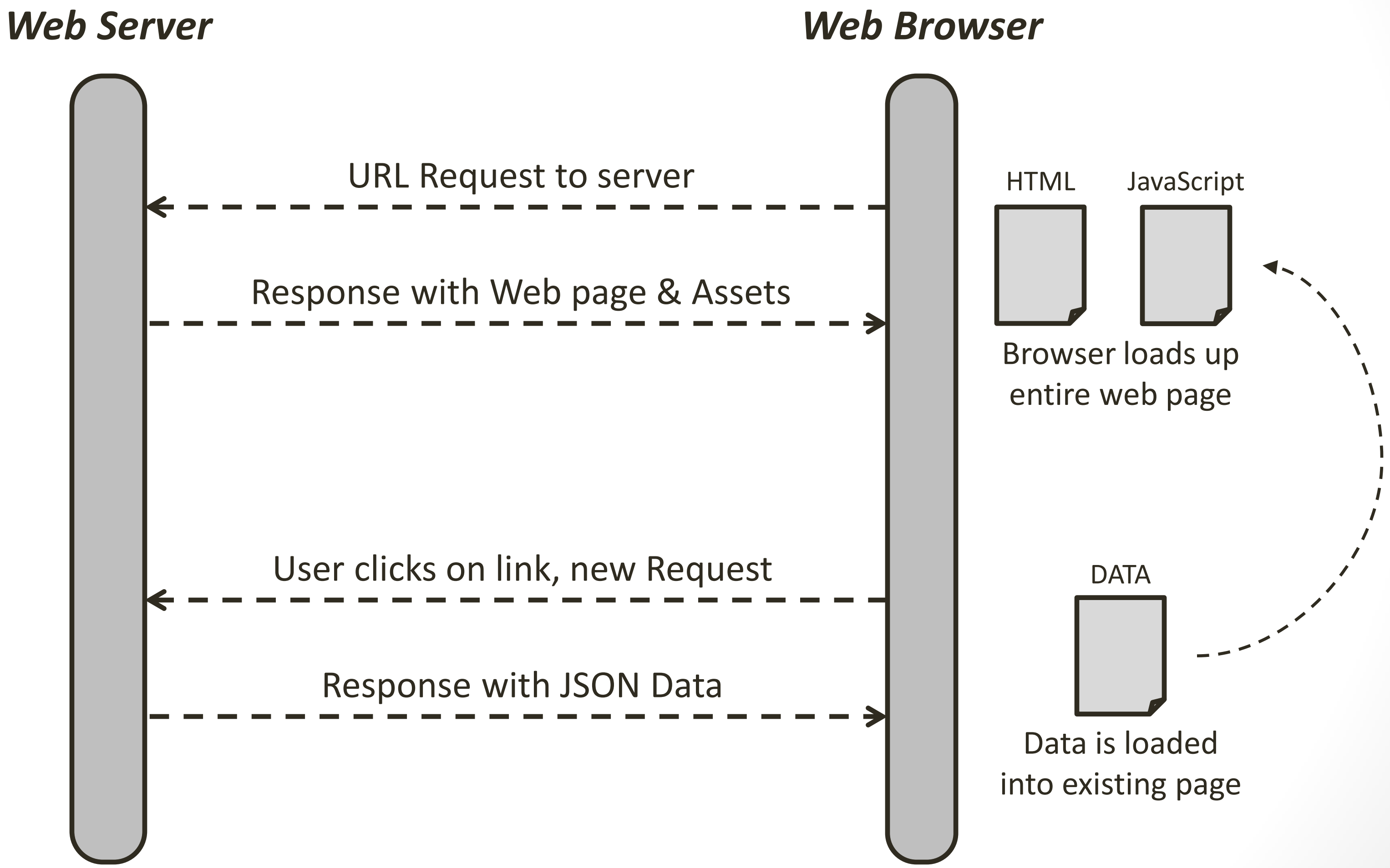
Introducing Angular

- Developed in 2009 by Misko Hevery
- Structural framework for dynamic web apps
- Front-end SPA, RIA framework
- Uses HTML as the template language
- Lets you extend HTML's syntax
- Declarative programming
- Not every app is a good fit for Angular
  - Best suited for building CRUD applications
  - Games and GUI editors – not a good fit

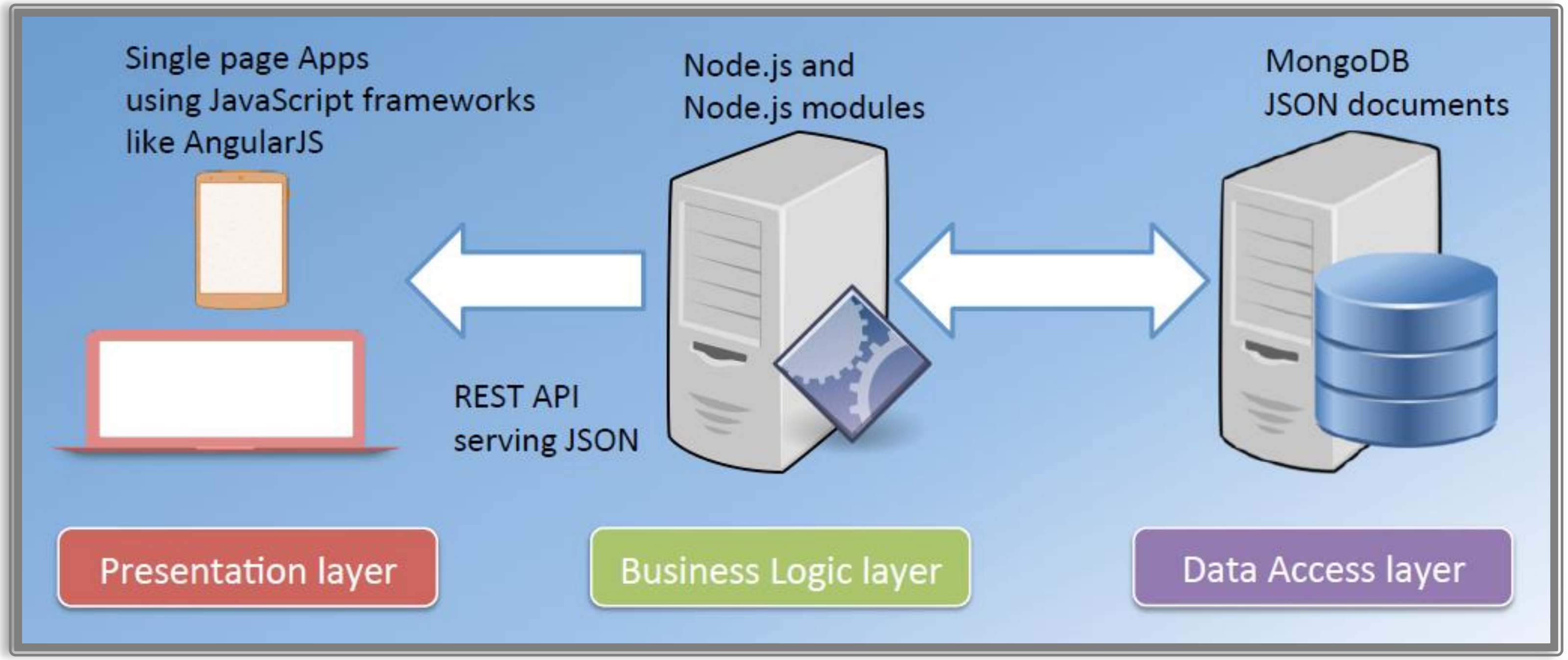
Angular - Advantages

- Helps you organize your JavaScript code
- Helps create responsive web apps
- Data Binding and Dependency Injection eliminates much of the manual code
- Decouple DOM manipulation from app logic
  - Improves testability
- Decouple client side of an app from the server side
  - Allows reuse
  - Allows parallel development

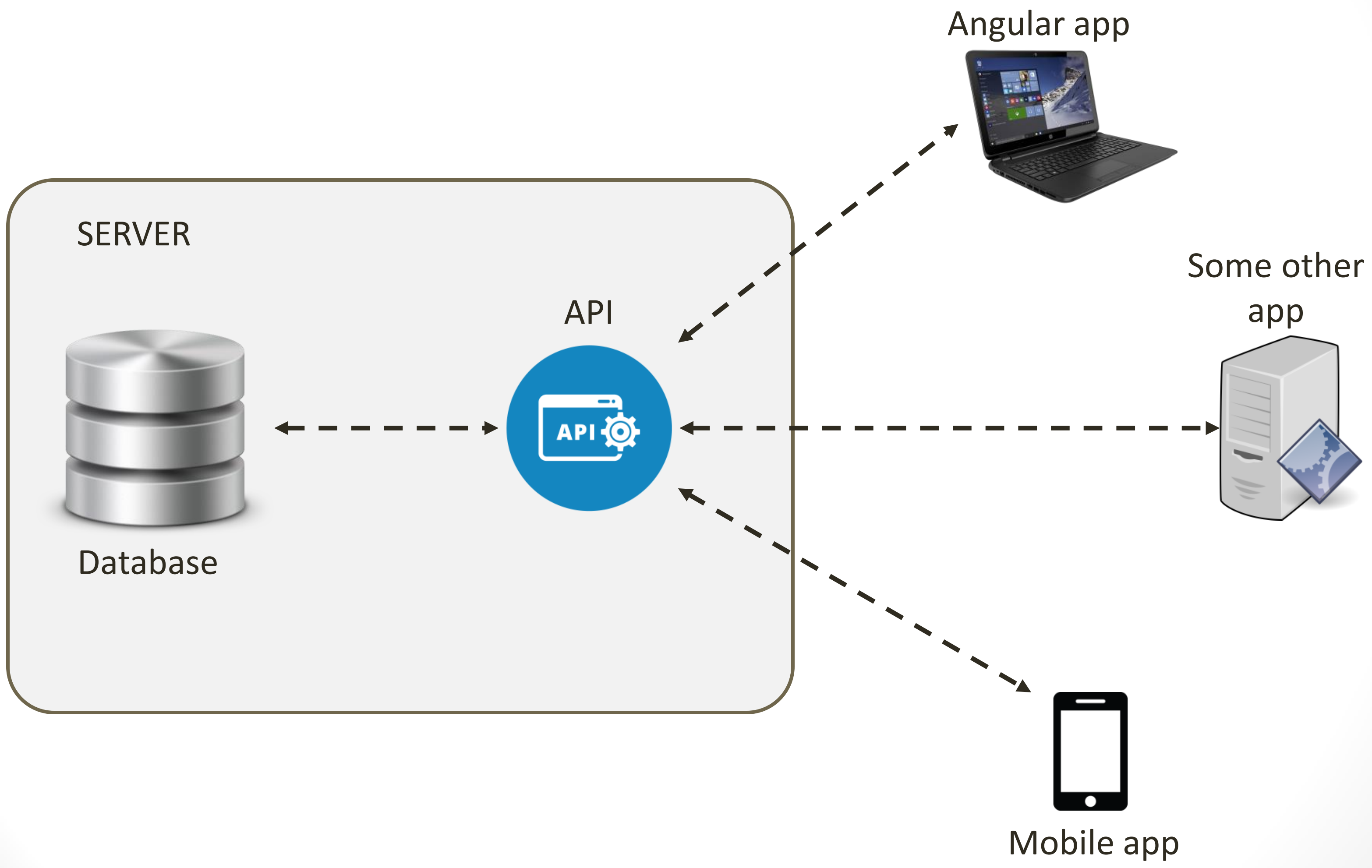
Responsive Page using Angular


Web Server
Web Browser
URL Request to server
Response with Web page & Assets
User clicks on link, new Request
Response with JSON Data
HTML
JavaScript
Browser loads up entire web page
DATA
Data is loaded into existing page

Where does Angular fit?



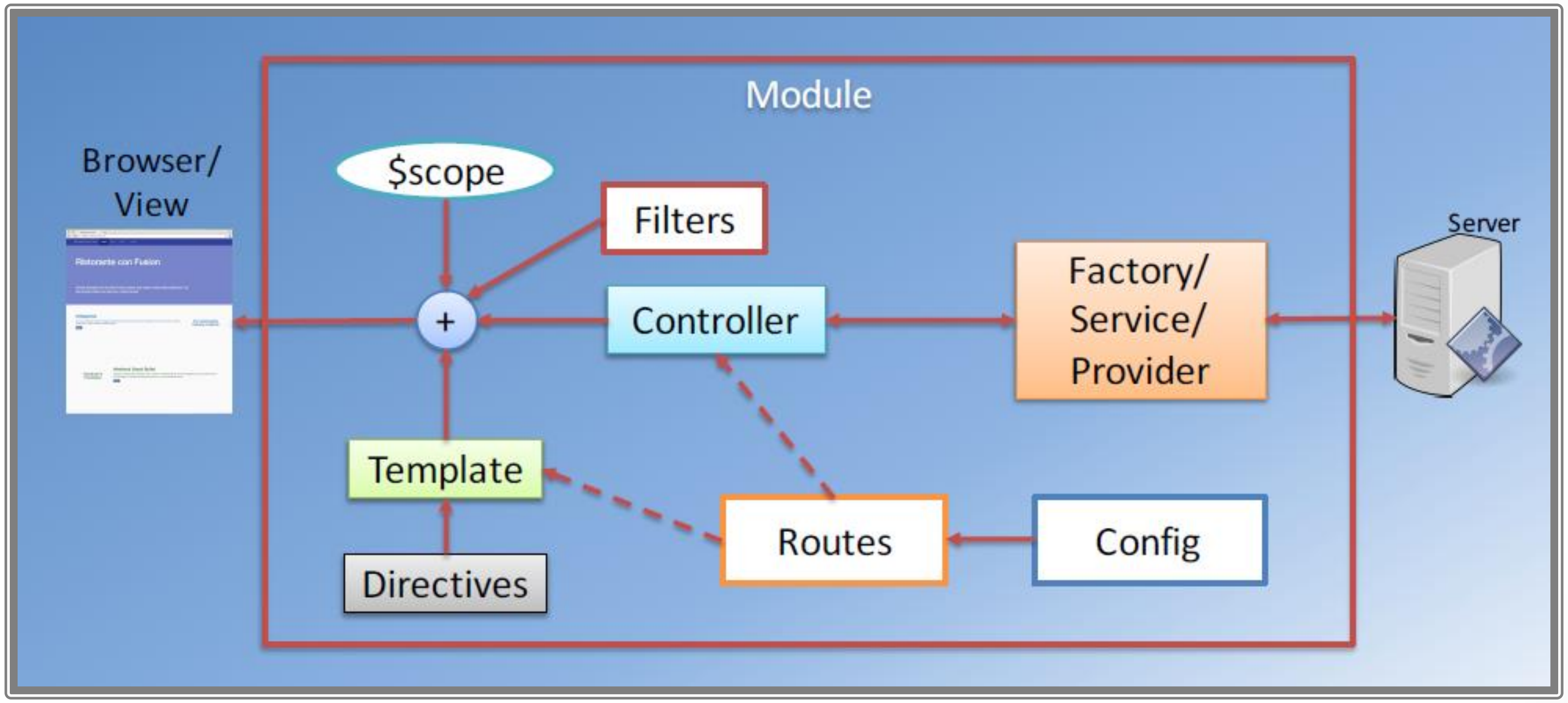
API Driven Approach



Angular Vocabulary

- Data binding
- Scope
- Directives
- Templates
- Routing
- Testing
- Modules
- Controllers
- Filters
- Services
- Views
- Form Validation
- MVC, ViewModel
- Dependency Injection

Angular Building Blocks



Getting Started

- Download Angular
  - http://angularjs.org/
  - angular.min.js
- Download Twitter Bootstrap
  - http://getbootstrap.com/
  - Bootstrap.min.css

Directives

- A marker on a HTML tag that tells Angular to run or reference some JavaScript code
- Binds behavior to HTML
- Custom attribute / element
- Helps you to extend HTML to support dynamic behavior
- Syntax: ng-<directive>
- Examples: ng-app, ng-bind, ng-model, ng-init, ng-repeat, etc.
- Declarative programming in action
  - Specifying what Angular needs to do

Directives

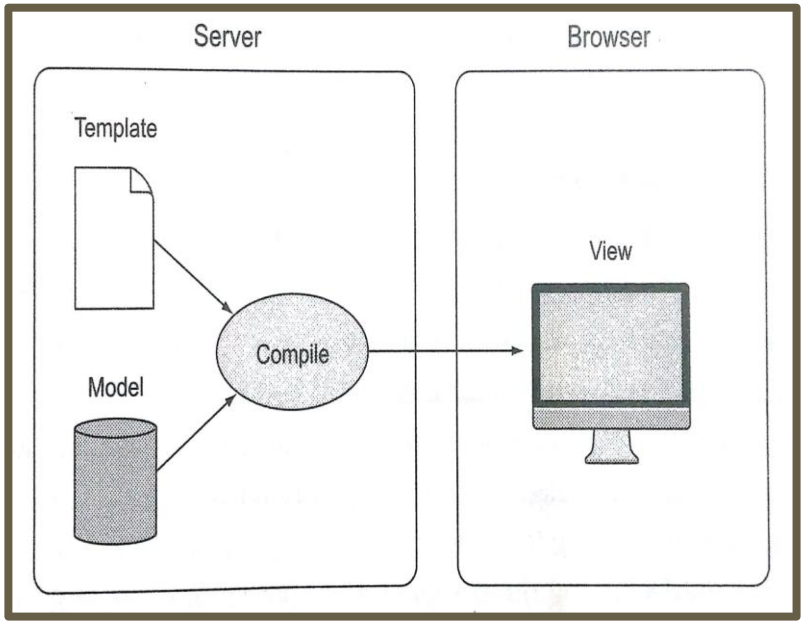
- ng-app
  - Runs the module when the document loads
  - Applied to specify the root of the application
  - Can be applied to any tag, typically applied to <html> tag
- ng-init
  - Used to evaluate an expression
  - Used to initialize a JavaScript variable
- ng-model
  - Binds an input value to a variable within the scope
- ng-repeat
  - Is a looping construct
  - Loops over items in a collection
  - Instantiates a template for each item

Expressions, Data Binding

- Expressions
  - Simple JavaScript expressions
  - Allow you to insert dynamic values into your HTML
  - No conditionals, loops or exceptions
  - Syntax: {{ <expression> }}

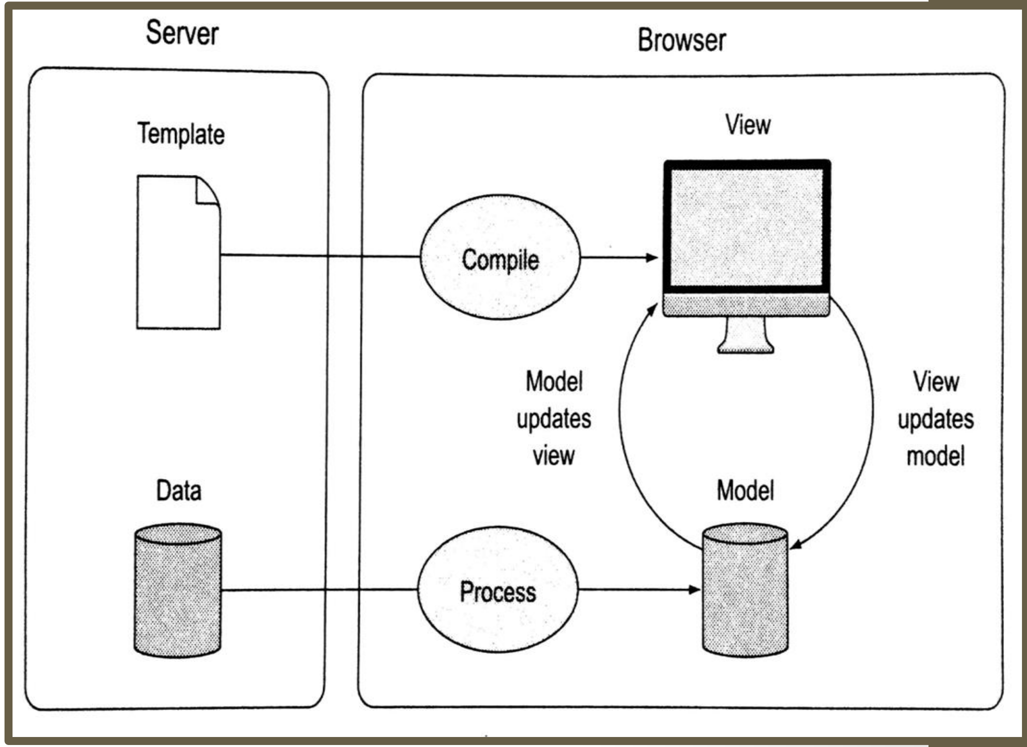
- Data Binding
  - Binding an HTML or CSS property to a JavaScript variable
  - Automatic synchronization of data between the model and view components

Data Binding



One-way data binding
The template and model are compiled on the server before being sent to the browser

Two-way data binding
The model and the view are processed in the browser and bound together, each instantly updating the other



MVC / MVVM

- Refer to my blog – MVC, MVVM and Angular
- URL:
  - https://naveenpete.wordpress.com/2016/09/07/mvc-mvvm-and-angular/

Module

- Where we write pieces of our Angular app
- Makes our code more maintainable, testable and readable
- Where we define dependencies for our app
- A collection of:
  - Controllers
  - Directives
  - Filters
  - Services
  - Other config information
- Usage
  - angular.module(‘store’, []);
  - <html ng-app=‘store’></html>

Controller

- JavaScript object containing properties and methods
- Defines app’s behavior by defining functions and values

- Usage:

```
var app = angular.module(‘store’);
app.controller(‘StoreController’, function() {
    …
})

<div ng-controller=“StoreController as storeCtrl”>
    …
</div>
```

Filter

- Format the value of an expression for display
- Does not modify underlying data
- Angular comes with many built-in filters
  - uppercase
  - lowercase
  - currency
  - date
  - filter
  - orderBy
  - limitTo
- Custom filters

Scope

- An object that refers to the application model
- Core of two-way data binding
- Glue between the view and the controller
  - Controller sets properties on the scope
  - View binds to the properties set by the controller
  - Angular keeps the two in sync
- $rootScope – topmost scope
  - Created by Angular when your app starts
- As Angular traverses the DOM, new scopes are created when it encounters some directives
  - ng-controller, ng-repeat
  - Scope tree similar to DOM tree
- A new scope is created as a child of a parent scope
  - Child has access to properties in the parent scope

Forms

- Two-way data binding
  - Define a JavaScript object on the $scope
  - Use ng-model directive on form fields
- Binding <SELECT>: ng-options directive
- Form validation
  - Turn off HTML5 validation: novalidate
  - Make sure to give the form a name
  - Form submit: ng-submit
  - Field Properties
    - $pristine – true if form / field has not been changed
    - $dirty – reverse of $pristine
    - $valid – true if form / field is valid
    - $invalid – reverse of $valid
- Bootstrap classes
  - has-error, has-warning, has-success
  - help-block

Dependency Injection (DI)

- Software design pattern that implements Inversion of Control (IoC) for resolving dependencies
- Coined by Martin Fowler in 2004
- Dependency: An object that can be used (a service)
- Injection: Passing of a dependency to a dependent object so that it can use it. The client does not need to build the object
- Three ways
  - Create dependency using new operator
  - Look up dependency using a global variable
  - Have dependency passed to it where needed
- Third approach is most flexible
- Four Roles
  - The Service
  - The Client
  - The Interfaces
  - The Injector

Angular and DI

- Separation of business logic and dependency construction
- Dependency is passed to consuming object where needed
- Angular injector subsystem
- DI is extensively used in Angular
  - Services, Factories
  - Filters
  - Controllers
- Dependency Annotations
  - Inline Array
  - $inject property
  - Implicit
- Refer to my blog – Dependency Injection in Angular apps
- URL:
  - https://naveenpete.wordpress.com/2016/10/24/dependency-injection-in-angular-apps/

Service

- Substitutable object wired together using DI
- Allows organizing and sharing code across an app
- Lazily instantiated
- Singleton
- Angular includes several built-in services
  - $http, $timeout, $window, $location, $rootScope, $log, $filter
- Five ways
  - service()
  - factory()
  - provider()
  - value()
  - constant()

- AJAX calls
- Business rules
- Calculations
- Share data between controllers

Angular Template

- Written in HTML
- Contains Angular specific elements and attributes
- Dynamic View = Template + Controller + Model
- Angular elements and attributes
  - Directives
  - Markup
  - Filters
  - Form Controls
- ng-include directive
  - Used to fetch, compile and include an Angular template
  - Creates a new scope

SPA

- Web application that fits in a single web page
- No need to reload entire page, redraw parts of the page when needed
- UX like a desktop / native application
- Most assets / resources are retrieved during the initial page load

- Challenges
  - Search engine optimization
  - Analytics
  - Initial page load
  - History

SPA

- Role of Server
  - Serves data using REST API
  - Supplies static HTML pages, Angular templates and other assets
- Role of Client
  - Rendering of view - Templating
  - Routing
- Deep linking
  - The use of a hyperlink that links to a specific, generally searchable or indexed, piece of web content on a website
  - E.g. http://example.com/path/page.html

SPA

- The $location service
  - Exposes the current URL in the browser address bar
    - Watch and observe the URL
    - Change the URL
  - Maintains synchronization between itself and the browser's URL when the user
    - Changes the address in the browser's address bar
    - Clicks the back or forward button in the browser
    - Clicks on a link in the page

Routing - ngRoute

- Associate a template and a controller with a specific URL
- In SPA, hash portion of the URL is used
- ngRoute module
  - Separate module: angular-route.js
  - Dependency Injection
    - angular.module(‘app-name', ['ngRoute']);
  - $routeProvider
    - Used for configuring routes
    - Wires together controller, view template and the current URL
  - $routeParams
    - Allows you to retrieve the current set of route parameters
  - ngView directive
    - Complements route service by including the rendered template of the current route into the main layout file (index.html)

Routing – UI Router

- Limitations
  - Only one view is allowed per page
  - Views are tied to URL
- Introducing UI Router
  - View are based on states instead of URL
    - Can change parts of the application even if the URL does not change
  - Multiple views
  - Nested views
  - Download: https://github.com/angular-ui/ui-router
  - Separate module: angular-ui-router.js
  - Dependency Injection
    - angular.module('app-name', ['ui.router']);

Routing – UI Router

- $stateProvider
  - Manages state definitions
- $stateParams
  - Object that will have one key per URL parameter
  - A perfect way to provide your controller with the individual parts of the navigated URL
- ui-sref directive
  - Equivalent to href or ng-href in <a /> elements except the target value is a state name
- ui-view directive
  - Renders views defined in the current state

Client Server Communication

- Web applications are not stand-alone
- Network operations cause unexpected delays
- Data is not instantly available
- HTTP – A client-server communication protocol
- HTTP response format
    - XML
    - JSON
- JSON
    - Lightweight data interchange format
    - Language independent
    - Self-describing, easy to understand
    - Data is structured as a collection of name-value pairs

$http

- Core Angular service to communicate with remote servers using HTTP protocol
- Uses browser's XmlHttpRequest object
- Asynchronous in nature
- Based on promise API exposed by $q service
- Returns a promise
-

```
$http( { method: 'GET', url: 'http://server/api' } )
    .then(
           function(response) {    // success    },
           function(error) {    // error    }
    );
```

$http

- Shortcut methods
  - $http.get()
  - $http.post()
  - $http.put()
  - $http.delete()
- Response
  - response.data – string / object containing the body of the message
  - response.status – HTTP status code
  - response.headers – HTTP response header information
  - response.statusText – HTTP status text of the response

$resource

- Provides a higher level abstraction than $http
- Useful for interacting with a RESTful API server
- Wrapper around a REST API to perform CRUD operations
- Not part of Angular core
- Separate module: angular-resource.js
- Dependency Injection
  - angular.module(‘app-name’, [‘ngResource’]);
- Usage

```
    $resource('http://localhost:3000/movies/:id',
        { id: '@id' },
        { update: { method: 'PUT' } }
    );
```

$resource

- $resource Methods
  - .query()
  - .get()
  - .save()
  - .remove()
  - .delete()

Custom Directives

- What is a Directive?
  - Markers on a DOM element that tell Angular's HTML compiler to attach a specified behavior to that DOM element
  - Teaches HTML new tricks
  - Makes HTML more expressive
- What can a directive do? Some examples
  - Manipulate the DOM
  - Iterate through data
  - Handle Events
  - Modify CSS
  - Validate data
  - Data binding

Custom Directives

- 3rd Party Directives
  - UI Bootstrap
  - AngularStrap
  - Angular UI Grid
  - Angular Translate
- Directive Types
  - Attribute directives
    - <div my-dir="value"></div>
  - Element directives
    - <my-dir></my-dir>
  - CSS class directives
    - <div class="my-dir: value;"><div>
  - Comment directives
    - <!-- directive: my-dir value -->

Custom Directives

- Directive Building Blocks
  - $compile
  - Directive Definition Object (DDO)
  - Template
  - Scope
- Features of DDO
  - Defines the template for the directive
  - Can include DOM manipulation code
  - Can define a controller for the directive
  - Controls the directive’s scope
  - Defines how the directive can be used

Custom Directives

Key DDO Properties	
restrict	scope
template	controller
templateUrl	link

```
angular.module('myApp')
    .directive('myDirective', function() {
        return {
            restrict: 'EA',
            scope: {},
            template: '<div>{{ myVal }}</div>',
            controller: MyController,
            link: function(scope, element, attrs) { }
        };
    });
```

Custom Directives

- Shared Scope
- Isolate Scope

Angular 2

- An integrated ecosystem that covers all of the concerns in building web apps
- Better performance, 5x faster than Angular 1.x
- Component based UI, controllers and directives are eliminated
- Uses Microsoft TypeScript as main programming language
- Meets ES6 specification
- Built for mobile, greatly simplifies app development on all mobile platforms
- Allows reuse of components across multiple platforms
- Controllers are replaced with Components
- $scope has been removed
- Directly uses valid HTML element properties and events. Built-in directives (ng-src, ng-show, ng-hide, etc.) of Angular 1.x are no more available

Angular 2

- Angular Architecture Overview
  - Check Angular 2 web site – Architecture Overview
  - URL
    - https://angular.io/docs/ts/latest/guide/architecture.html

Q & A

- Thank you