AngularJS (Superheroic JavaScript Framework)

Learning Objective

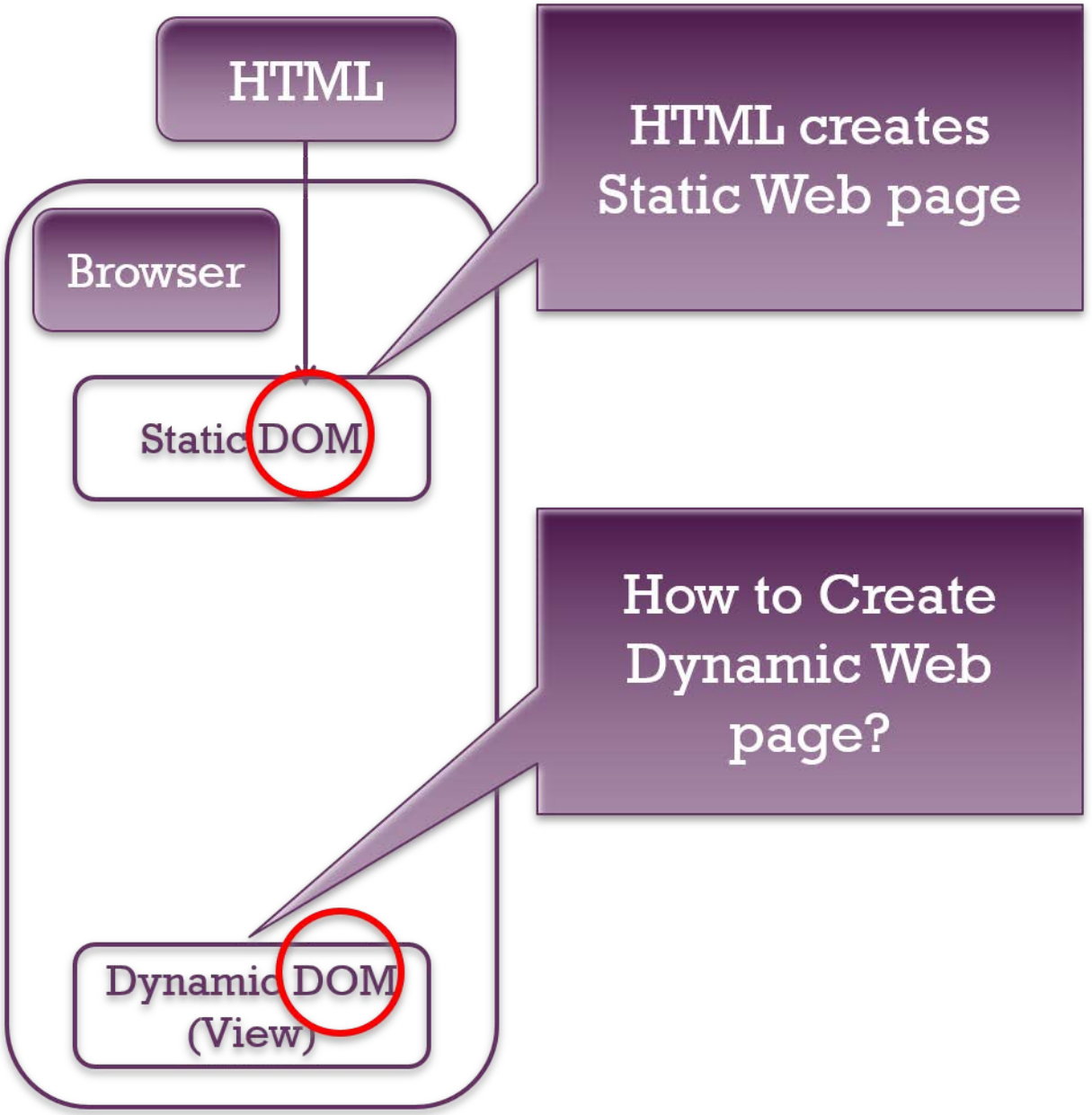
- Learning Objective 3
  - Examine current practices used in industry and apply this to Web Application Development

What's in this lecture?

- Introduction to AngularJS (Angular)
- The Basics
- Angular CLI
- Typescripting
- Build a simple AngularJS app using CLI

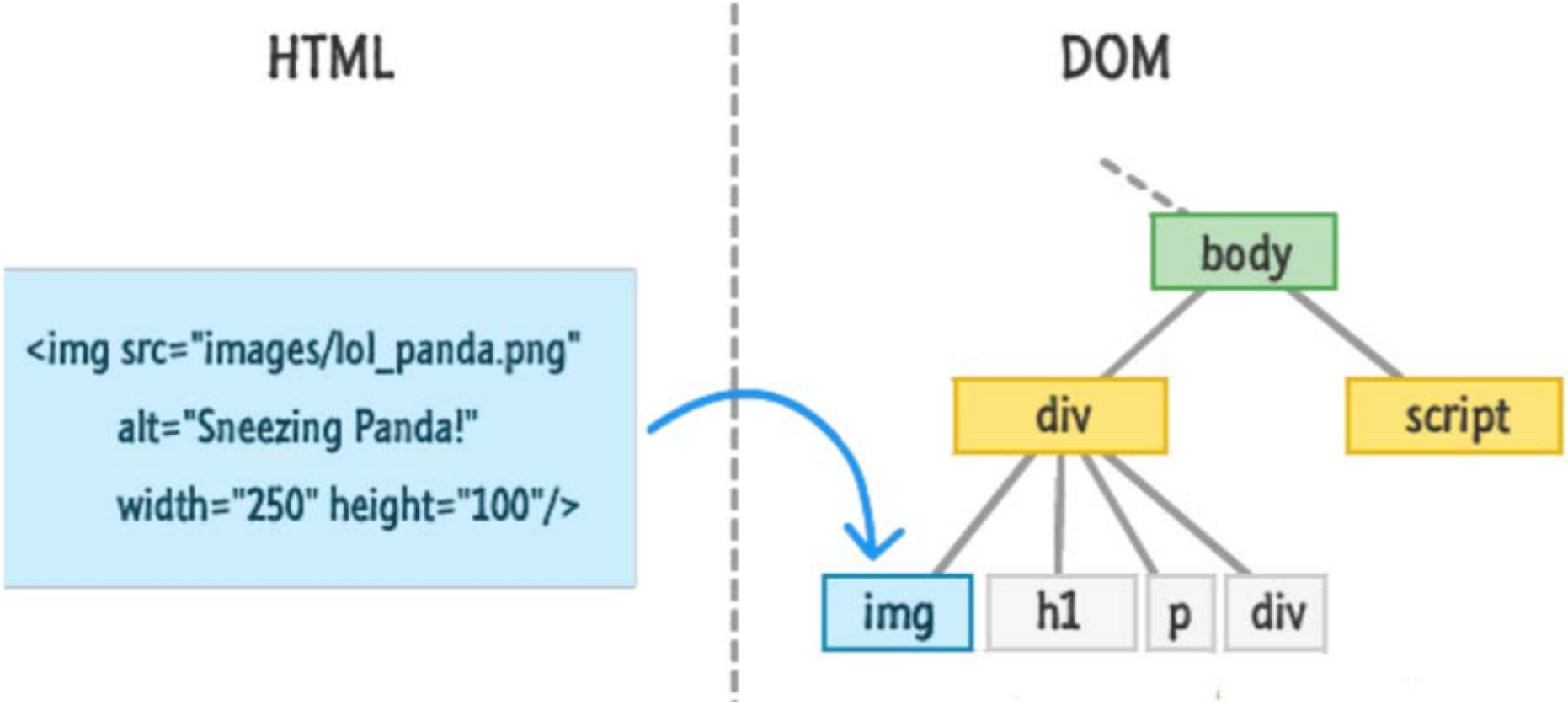
https://angularjs.org/
https://docs.angularjs.org/guide/concepts
https://angular.io/docs/ts/latest/
http://devdocs.io/angularjs~1.5/

Introduction to AngularJS

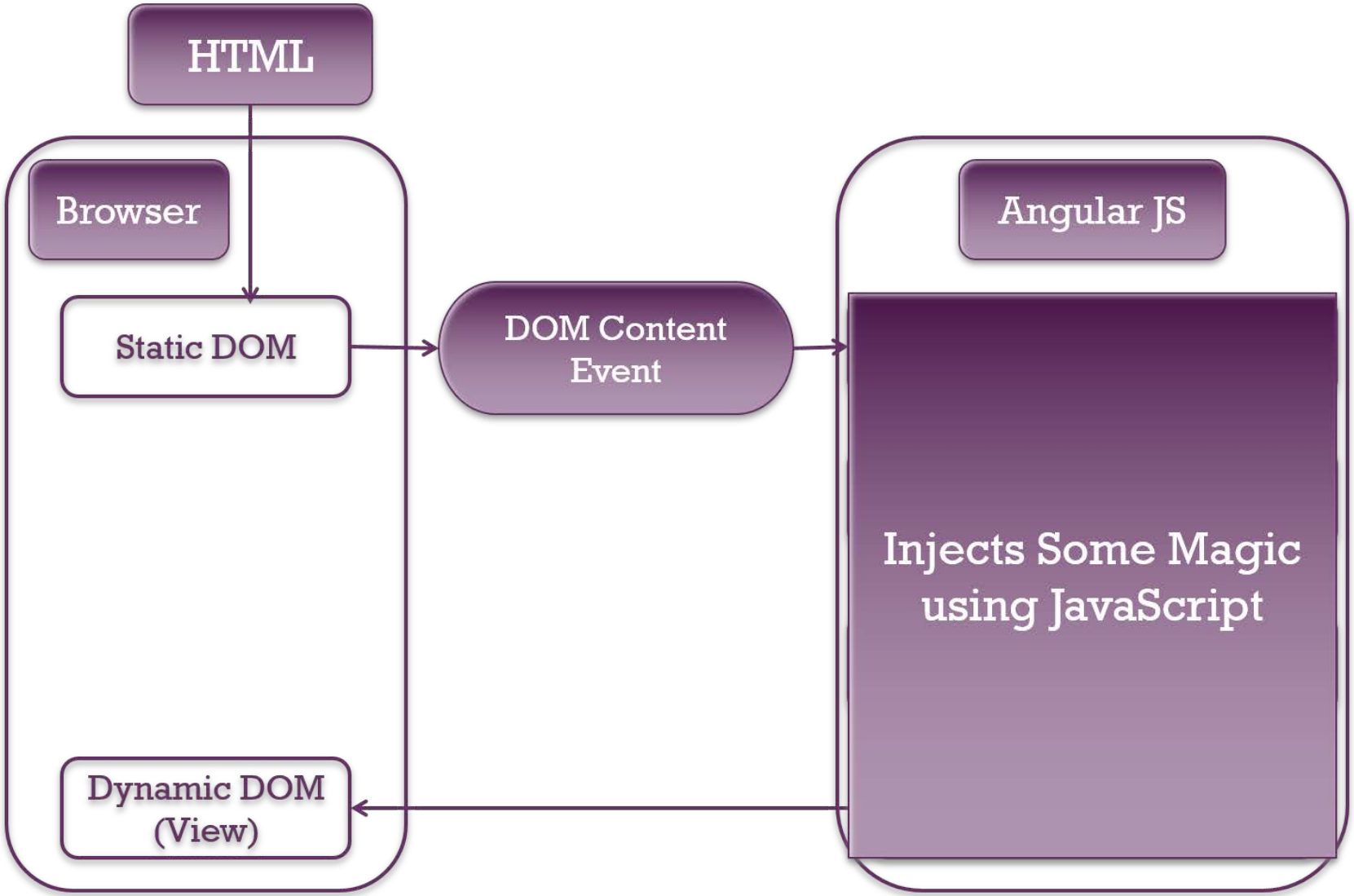


What is DOM?

- DOM – Document Object Model is a tree representing HTML page elements
- It is a structured representation of a document and it provides a programming interface which allows you to change the style and content of the web page.



How Angular JS converts Static DOM to Dynamic



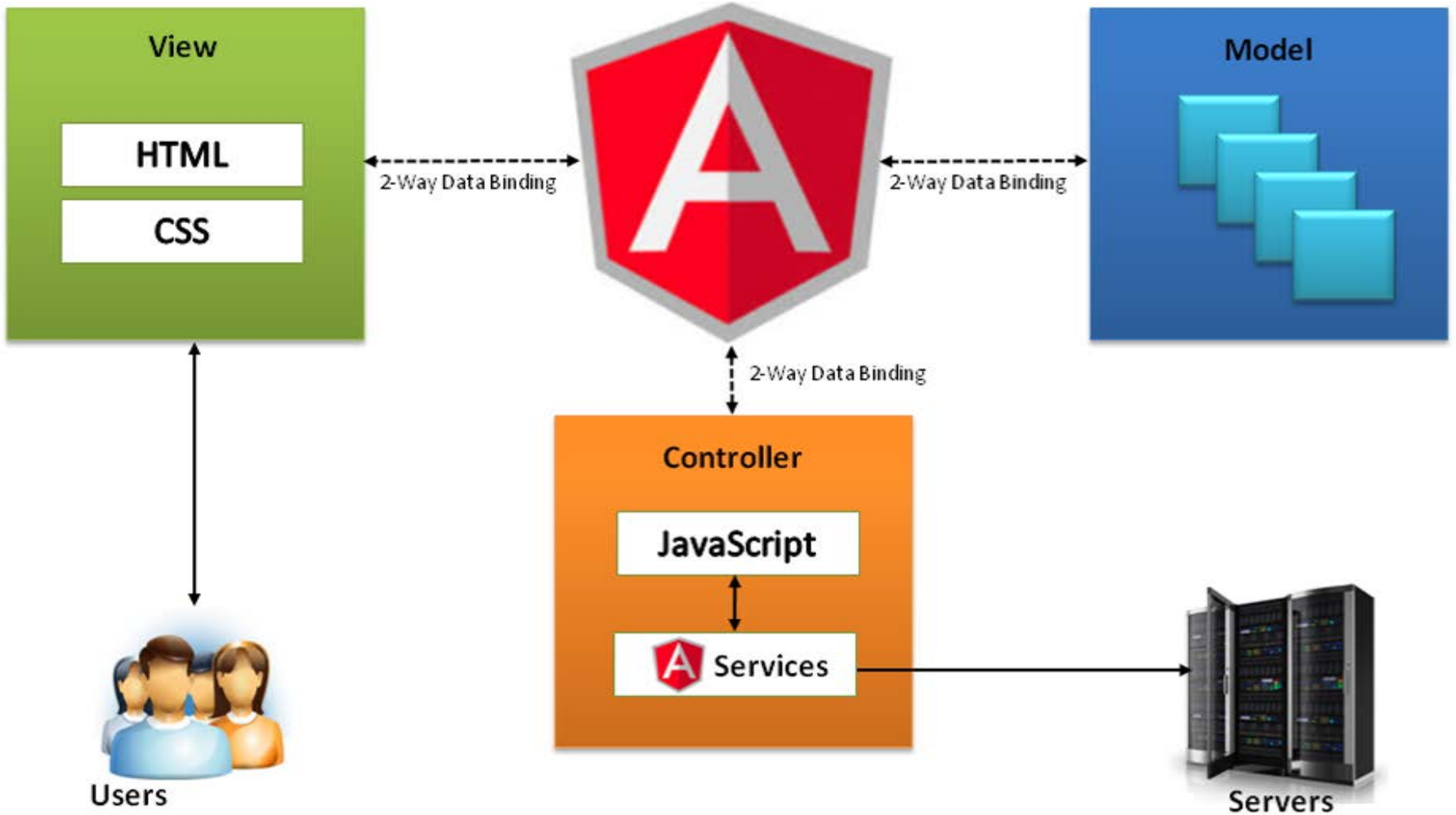
A complete client-side solution

- AngularJS is not a single piece in the overall puzzle of building the client-side of a web application.
- It handles all of the DOM and AJAX glue code you once wrote by hand and puts it in a well-defined structure.
- This makes AngularJS suitable for creating a complete CRUD (Create, Read, Update, Delete) application, taking care of:
  - Data-binding
  - Basic templating directives
  - Form validation
  - Routing
  - Deep-linking
  - Reusable components
  - Dependency injection.

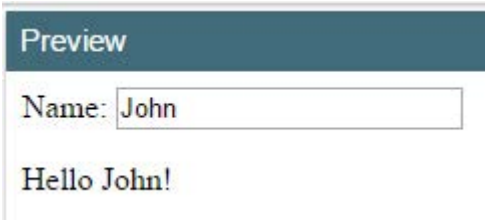
Angular Concepts and Terminology

- View/Template: HTML with additional markup used to describe what should be displayed
- Directive: Allows developer to extend HTML with own elements and attributes (reusable pieces)
- Scope: Context where the model data is stored so that templates and controllers can access
- Compliler: Processes the template to generate HTML for the browser
- Data Binding: Syncing of the data between the Scope and the HTML (two ways)
- Dependency Injection: Fetching and setting up all the functionality needed by a component
- Module: A container for all the parts of an application
- Service: A way of packaging functionality to make it available to any view

MVC in Angular JS

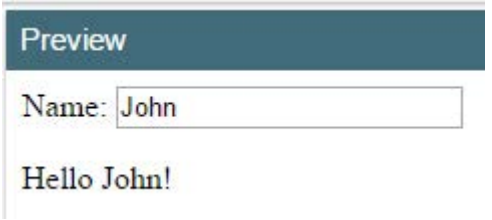


Angular Basic Code (index.html with embedded JavaScript)




```
<!DOCTYPE html>
<html ng-app="angular_mvc">
  <head>
    <meta charset="utf-8" />
    <title>AngularJS MVC Demo</title>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      angular.module('angular_mvc', [])
      .controller('MainCtrl', function($scope) {
      $scope.message = 'Hello';
    });
    </script>
  </head>
  <body ng-controller="MainCtrl" >
    <label>Name:</label>
     <input type="text" ng-model="yourName" placeholder="Enter a name here">
    <p>{{message}} {{yourName}}!</p>
  </body>
</html>
```

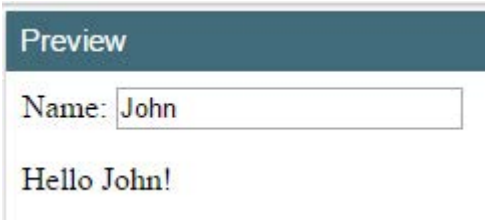
Angular Basic Code (index.html with embedded JavaScript)




```
<!DOCTYPE html>
<html ng-app="angular_mvc">
  <head>
    <meta charset="utf-8" />
    <title>AngularJS MVC Demo</title>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      angular.module('angular_mvc', [])
      .controller('MainCtrl', function($scope) {
      $scope.message = 'Hello';
    });
    </script>
  </head>
  <body ng-controller="MainCtrl" >
    <label>Name:</label>
     <input type="text" ng-model="yourName" placeholder="Enter a name here">
    <p>{{message}} {{yourName}}!</p>
  </body>
</html>
```

Include Angular JS

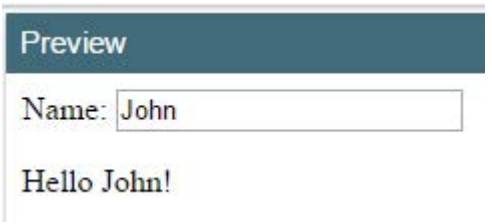
Angular Basic Code (index.html with embedded JavaScript)



Angular scans the html for ng-app directive

```
<!DOCTYPE html>
<html ng-app="angular_mvc">
  <head>
    <meta charset="utf-8" />
    <title>AngularJS MVC Demo</title>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      angular.module('angular_mvc', [])
      .controller('MainCtrl', function($scope) {
      $scope.message = 'Hello';
    });
    </script>
  </head>
  <body ng-controller="MainCtrl" >
    <label>Name:</label>
     <input type="text" ng-model="yourName" placeholder="Enter a name here">
    <p>{{message}} {{yourName}}!</p>
  </body>
</html>
```

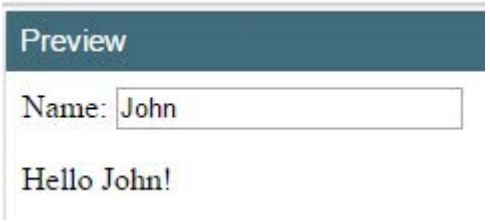
Angular Basic Code (index.html with embedded JavaScript)




```
<!DOCTYPE html>
<html ng-app="angular_mvc">
  <head>
    <meta charset="utf-8" />
    <title>AngularJS MVC Demo</title>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      angular.module('angular_mvc', [])
      .controller('MainCtrl', function($scope) {
      $scope.message = 'Hello';
    });
    </script>
  </head>
  <body ng-controller="MainCtrl" >
    <label>Name:</label>
     <input type="text" ng-model="yourName" placeholder="Enter a name here">
    <p>{{message}} {{yourName}}!</p>
  </body>
</html>
```

Defining Controller

Angular Basic Code (index.html with embedded JavaScript)




```
<!DOCTYPE html>
<html ng-app="angular_mvc">
  <head>
    <meta charset="utf-8" />
    <title>AngularJS MVC Demo</title>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      angular.module('angular_mvc', [])
      .controller('MainCtrl', function($scope) {
      $scope.message = 'Hello';
    });
    </script>
  </head>
  <body ng-controller="MainCtrl" >
    <label>Name:</label>
     <input type="text" ng-model="yourName" placeholder="Enter a name here">
    <p>{{message}} {{yourName}}!</p>
  </body>
</html>
```

Defining Scope in Controller.
Compiler - Scans DOM covered by the ng-app looking for templating markup - Fills in with information from scope.
This script may be shifted to a separate javaScript file

The Controller Directive attaches the controller class to the View. This is how Angular JS supports the principles of MVC design pattern.

Angular Basic Application - Demo

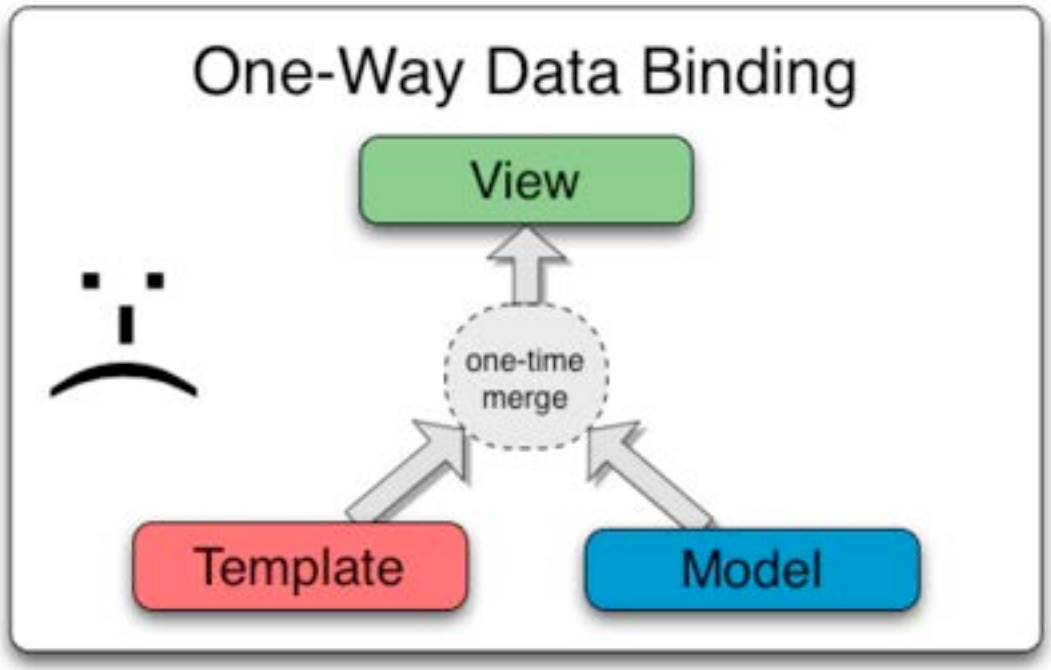
https://plnkr.co/edit/FXsyDXGpEpQqBgU6EZSe?p=preview

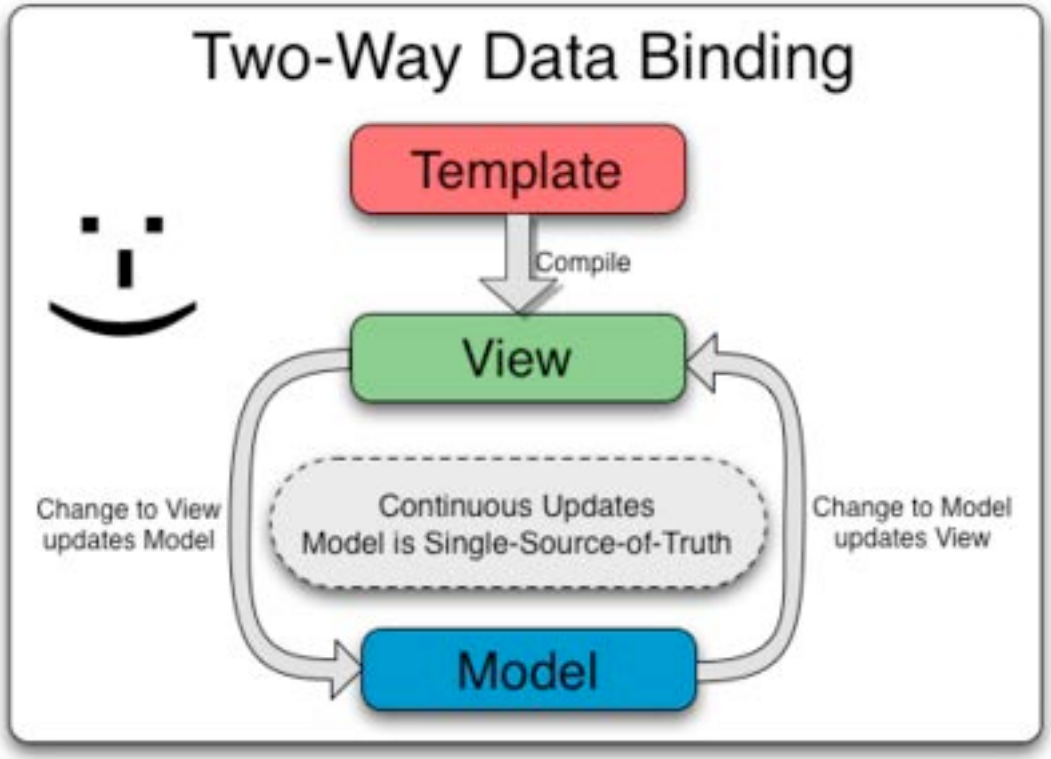
- You can play around with Angular JS online in Plunker
- You may create your own private plunks by signing with your GitHub account.

Tow-way Data Binding in AngularJs Templates

- AngularJS templates work differently.
- First the template (which is the uncompiled HTML along with any additional markup or directives) is compiled on the browser.
- The compilation step produces a live view.
- Any changes to the view are immediately reflected in the model, and any changes in the model are propagated to the view.
- The model is the single-source-of-truth for the application state, greatly simplifying the programming model for the developer.
- You can think of the view as simply an instant projection of your model.
- Controller acts as bridge between model and view applying MVC architecture.

Difference between one-way and two-way data binding





Module - ng-app directive

- You can think of a module as a container for the different parts of your app - controllers, services, filters, directives, etc.
- Most applications have a **main** method that instantiates and wires together the different parts of the application.
- AngularJS apps don't have a main method. Instead modules declaratively specify how an application should be bootstrapped.
- There are several advantages to this approach:
  - The declarative process is easier to understand.
  - You can package code as reusable modules.
  - The modules can be loaded in any order (or even in parallel) because modules delay execution.
  - Unit tests only have to load relevant modules, which keeps them fast.
  - End-to-end tests can use modules to override configuration.

Controllers - ng-controller directive

- In AngularJS, a Controller is defined by a JavaScript constructor function that is used to augment the AngularJS **Scope**.
- When a Controller is attached to the DOM via the ng-controller directive, AngularJS will instantiate a new Controller object, using the specified Controller's constructor function.
- A new child scope will be created and made available as an **injectable** parameter to the Controller's constructor function as $scope.
- Use controllers to:
  - Set up the initial state of the $scope object.
  - Add behavior to the $scope object.
- Example:

```
var myApp = angular.module('myApp',[]);
myApp.controller('GreetingController', ['$scope', function($scope) {

$scope.greeting = 'Hola!';
}]);
```

Scope

- Scope is an object that refers to the application model.
- It is an execution context for expressions.
- Scopes are arranged in hierarchical structure which mimic the DOM structure of the application. Scopes can **watch** expressions and propagate events.

See Example in Plunker

- https://plnkr.co/edit/xHxRAyc8Hee7iRK8XCLF?p=preview

Scope as Data Model

- Scope is the glue between application controller and the view. During the template linking phase the directives set up $watch expressions on the scope.
- The $watch allows the directives to be notified of property changes, which allows the directive to render the updated value to the DOM.
- Both controllers and directives have reference to the scope, but not to each other.
- This arrangement isolates the controller from the directive as well as from the DOM.
- This is an important point since it makes the controllers view agnostic, which greatly improves the testing story of the applications.

Dependency Injection

- Dependency Injection (DI) is a software design pattern that deals with how components get hold of their dependencies.
- The simplest way to get hold of the dependencies is to assume that the function parameter names are the names of the dependencies.
- Note: AngularJS uses **constructor injection.**

An Example:

```
<div ng-controller="MyController">
  <button ng-click="sayHello()">Hello</button> </div>

function MyController($scope, greeting) {
  $scope.sayHello = function() {
    greeting.greet('Hello World');   };
}
```

Node.js and npm

- **Node.js** is an open source **server** framework. (Angular is an open source **client** framework)
- Node.js allows you to run JavaScript on the server.
- Node.js has a set of built-in modules.
- Check version of node js installed on your computer `node -v`

- **NPM** is a **package manager** for Node.js packages, or modules.
- www.npmjs.com hosts thousands of free packages to download and use.
- The NPM program is installed on your computer when you install Node.js
- Check version of npm installed on your computer `npm -v`

- A **package** in Node.js contains all the files you need for a module.
- Modules are JavaScript libraries you can include in your project.
- Open the command line interface and tell NPM to download the package you want.
  - `npm install [package-name]`

Angular Command Line Interface (CLI)

- https://cli.angular.io/
- The Angular CLI is a tool to initialize, develop, scaffold and maintain Angular applications
- In cloud9, create a 'Blank Workspace'
- **To install the Angular CLI, type:**
  - **npm install -g angular-cli**
- **Generating and serving an Angular project via a development server Create and run a new project:**
  - **ng new my-project**
  - **cd my-project**
  - **ng serve --host 0.0.0.0 --port 8080 --live-reload-port 8081**
- **Navigate to http://[your-workspace-name]-[your-user-name].c9users.io:8080/.**
- **If everything is fine, you should see "app works!" message**
- **The app will automatically reload if you change any of the source files.**

Angular Command Line Interface (CLI)

- **Useful commands:**
  - **ng generate class [classname]**
  - **ng generate component [name]**
  - **ng generate service [name]**
  - **ng g roue [route/to/route-component] Alias: 'g'**
  - **ng g module [app-routing]**
    --flat Do not create the code in its own directory

Project file review

- **The src folder:**
  - **Your app lives in the src folder. All Angular components, templates, styles, images, and anything else your app needs go here. Any files outside of this folder are meant to support building your app.**
  - **app/app.component.{ts,html,css,spec.ts}**
    - **Defines the AppComponent along with an HTML template, CSS stylesheet, and a unit test. It is the root component of what will become a tree of nested components as the application evolves.**
  - **app/app.module.ts**
    - **Defines AppModule, the root module that tells Angular how to assemble the application. Right now it declares only the AppComponent. Soon there will be more components to declare.**
  - **assets/***
    - **A folder where you can put images and anything else to be copied wholesale when you build your application.**

The src folder

- **environments/***
  - **This folder contains one file for each of your destination environments, each exporting simple configuration variables to use in your application. The files are replaced on-the-fly when you build your app. You might use a different API endpoint for development than you do for production or maybe different analytics tokens. You might even use some mock services. Either way, the CLI has you covered.**
- **favicon.ico**
  - **Every site wants to look good on the bookmark bar. Get started with your very own Angular icon.**
- **index.html**
  - **The main HTML page that is served when someone visits your site. Most of the time you'll never need to edit it. The CLI automatically adds all js and css files when building your app so you never need to add any <script> or <link> tags here manually.**
- **main.ts**
  - **The main entry point for your app.**

The src folder

- **polyfills.ts**
  - **Different browsers have different levels of support of the web standards. Polyfills help normalize those differences.**
- **styles.css**
  - **Your global styles go here. Most of the time you'll want to have local styles in your components for easier maintenance, but styles that affect all of your app need to be in a central place.**
- **test.ts**
  - **This is the main entry point for your unit tests. It has some custom configuration that might be unfamiliar, but it's not something you'll need to edit.**
- **tsconfig.{app|spec}.json**
  - **TypeScript compiler configuration for the Angular app (tsconfig.app.json) and for the unit tests (tsconfig.spec.json).**

The root folder

- **The src/ folder is just one of the items inside the project's root folder.**
- **Other files help you build, test, maintain, document, and deploy the app. These files go in the root folder next to src/**
- **e2e/**
  - **Inside e2e/ live the end-to-end tests.**
- **node_modules/**
  - **Node.js creates this folder and puts all third party modules listed in package.json inside of it.**
- **.angular-cli.json**
  - **Configuration for Angular CLI. In this file you can set several defaults and also configure what files are included when your project is built. Check out the official documentation if you want to know more.**
- **.editorconfig**
  - **Simple configuration for your editor to make sure everyone that uses your project has the same basic configuration. Most editors support an .editorconfig file. See http://editorconfig.org for more information.**

The root folder

- .gitignore
  - Git configuration to make sure autogenerated files are not commited to source control.
- karma.conf.js
  - Unit test configuration for the Karma test runner, used when running ng test.
- package.json
  - npm configuration listing the third party packages your project uses. You can also add your own custom scripts here.
- protractor.conf.js
  - End-to-end test configuration for Protractor, used when running ng e2e.
- README.md
  - Basic documentation for your project, pre-filled with CLI command information. Make sure to enhance it with project documentation so that anyone checking out the repo can build your app!
- tsconfig.json
  - TypeScript compiler configuration for your IDE to pick up and give you helpful tooling.
- tslint.json: Linting helps keep your code style consistent.

Typescripting (http://www.typescriptlang.org/)

- **Angular is a modern framework built entirely in TypeScript, and as a result, using TypeScript with Angular provides a seamless experience.**
- **Angular CLI build and run Angular application in TypeScript (https://angular.io/guide/quickstart)**
- **TypeScript is a superset of JavaScript that solves a very specific problem – getting JavaScript development to scale.**
- **It compiles to simple javascript.**
- **It is open source, can run on any browser, any host, any operating system**
- **It starts from the same syntax and semantics that millions of JavaScript developers know today.**
- **The biggest selling point of TypeScript is tooling. It provides advanced autocompletion, navigation, and refactoring. Having such tools is almost a requirement for large projects.**
- **Still confused? Watch this video: https://channel9.msdn.com/posts/Anders-Hejlsberg-Introducing-TypeScript**

Further Readings

Angular Full Documentation
http://devdocs.io/angularjs~1.5/
https://angular.io/docs

For Concepts
https://docs.angularjs.org/guide/concepts

Understanding Scope
https://docs.angularjs.org/api/ng/type/$rootScope.Scope

Understanding Directives
https://docs.angularjs.org/guide/directive

Getting Started with Angular JS
https://www.slideshare.net/EdurekaIN/getting-started-with-angularjs-54293335

Angular CLI
https://cli.angular.io/