Web Technology and Standards

Dr. Alawami

Fall 2018

Agenda

- Caution
- Website design patterns
  - MVC
  - MVVM
- Front end framework
  - Angularjs Crash course

Running web apps

- Node js
  - npm install -g live-server
  - Navigate to your web app directory
    - Live-server

Install live server globally in your machine

Will run your web app in your browser

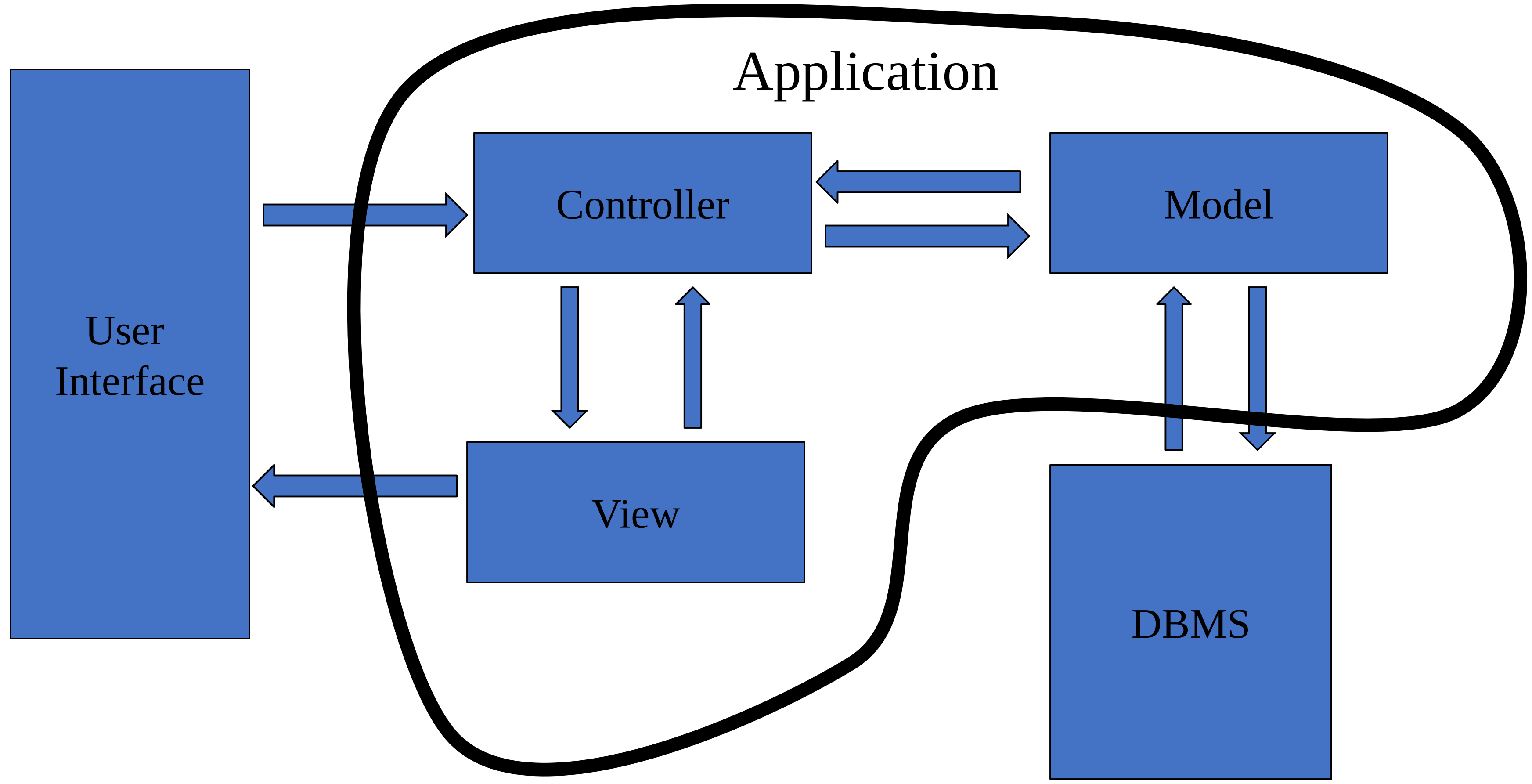
- Web Development environment (XAMPP, Apache, etc)
- Hosting provider in the web.

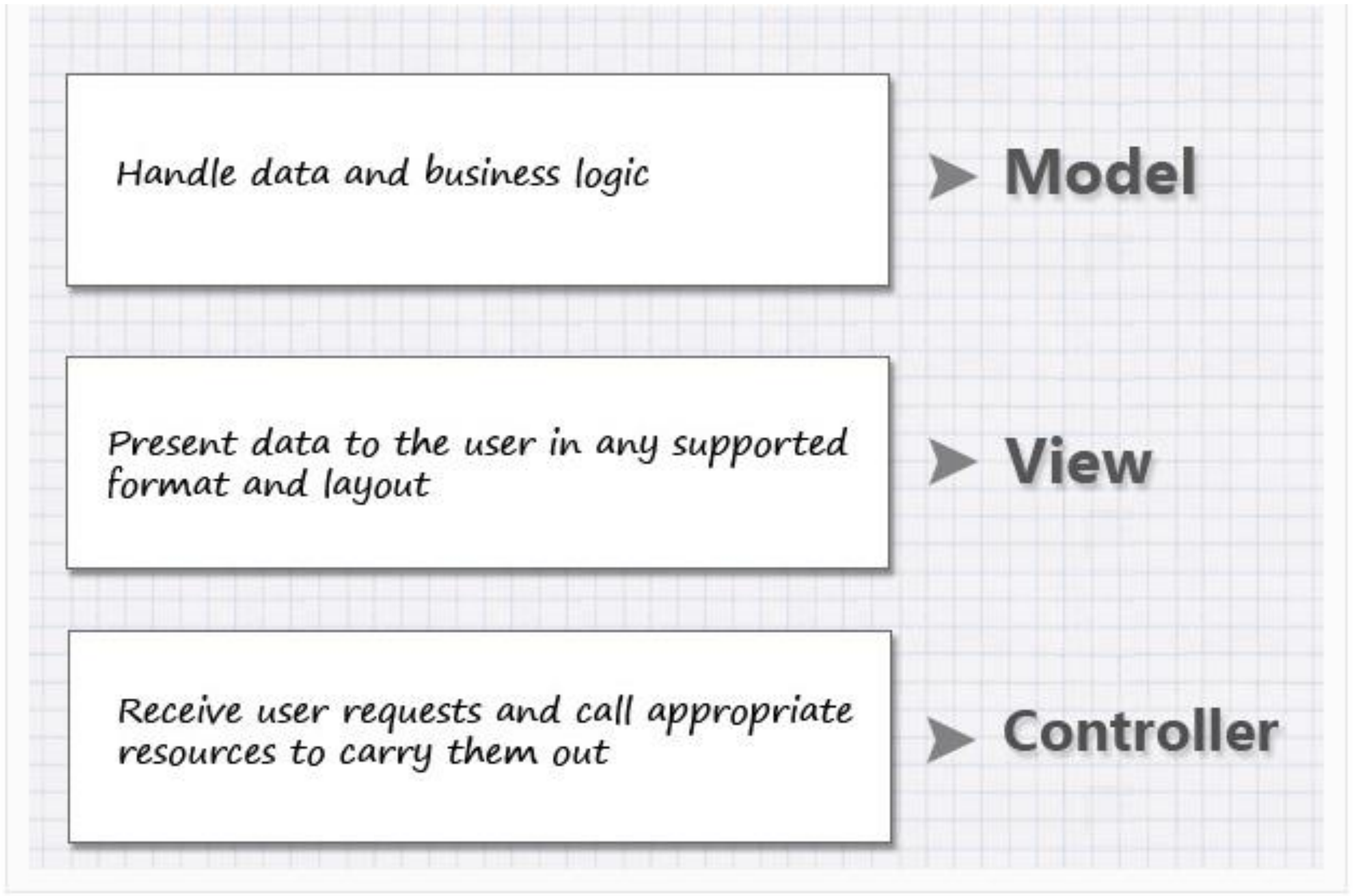
Websites Design Patterns

MVC Architecture

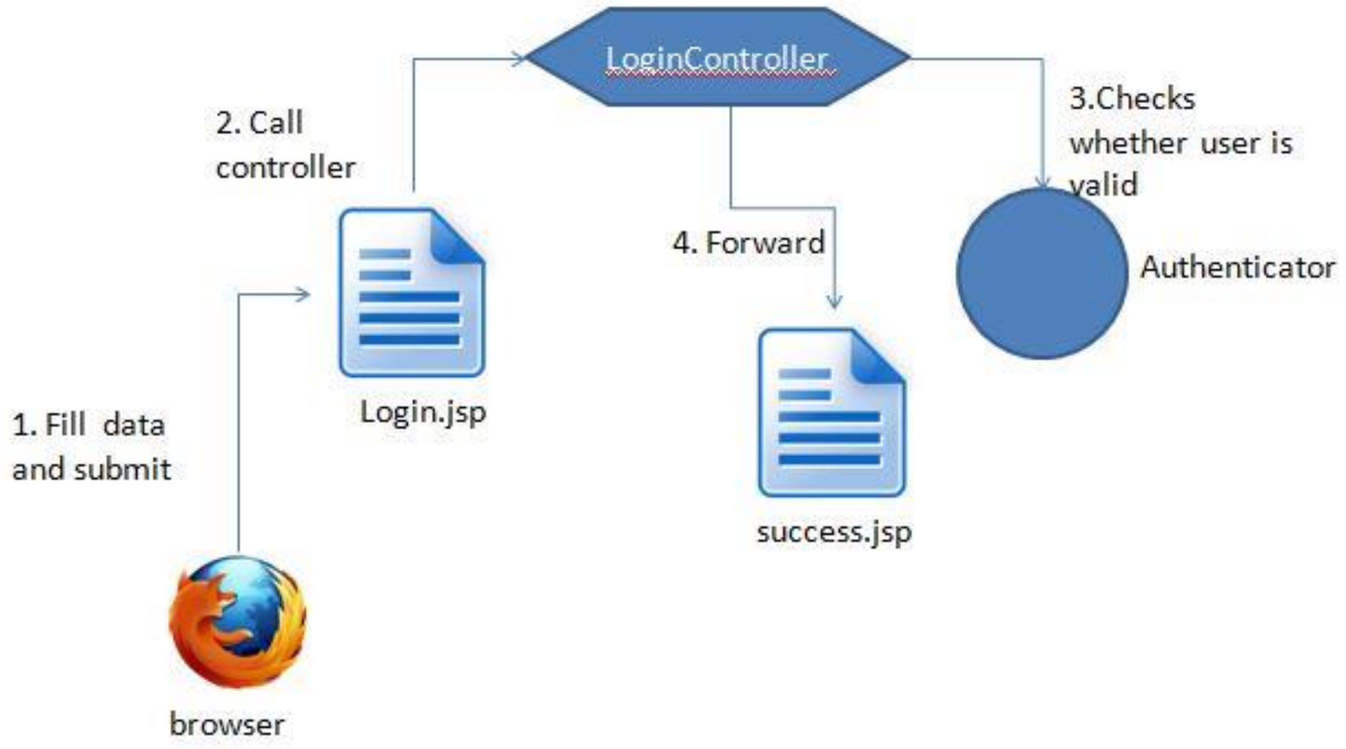
- The Model View Controller (MVC) Architecture is more than 30 years old.
  - It came from work at Xerox PARC on Smalltalk.
- It takes on various forms in various languages and involves three to five components:
  - Model – the abstract data model under control
  - View – one or more depictions of the model
  - Controller – business logic that determines the view based on model manipulation
  - DB – the ultimate repository for the model instantiation
  - Interface – the view from the client side which allows user input to the controller

MVC Graphically

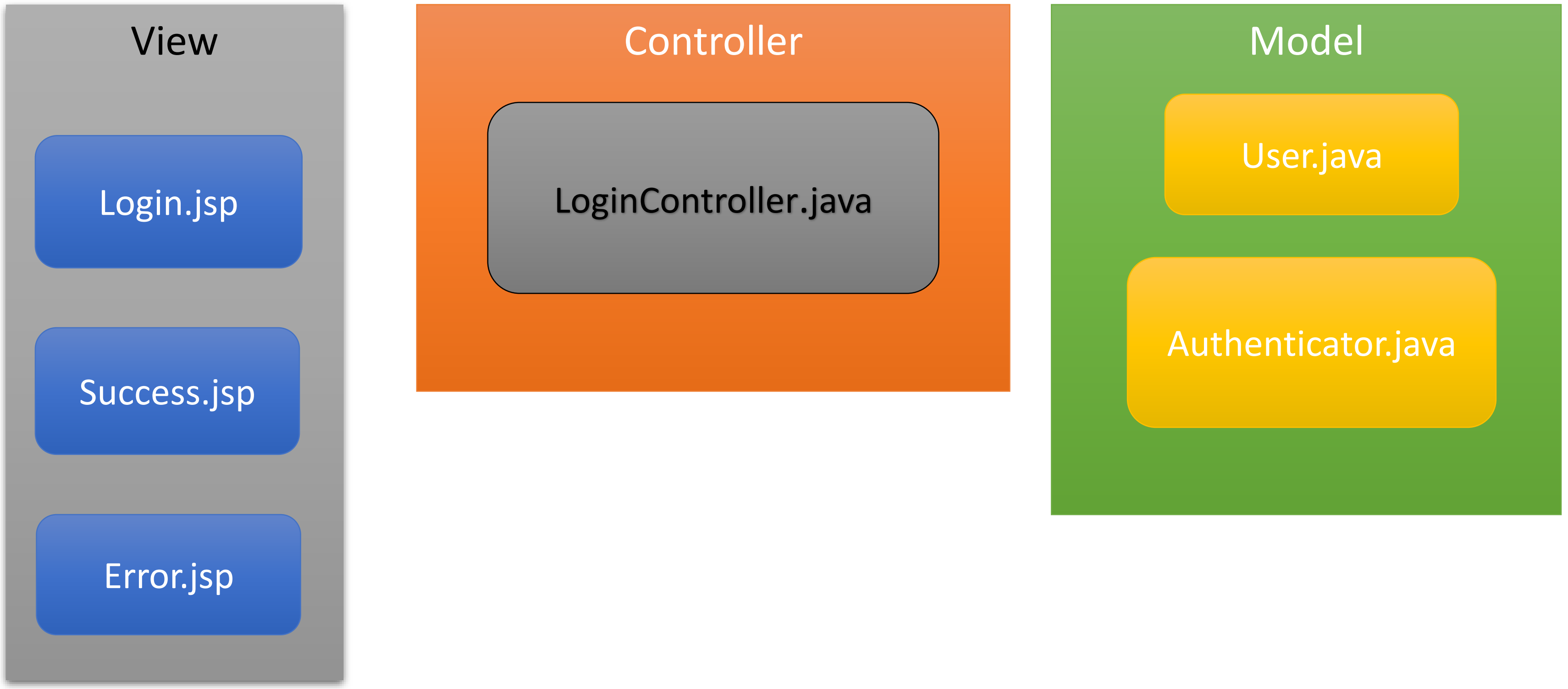



Handle data and business logic
Model
Present data to the user in any supported format and layout
View
Receive user requests and call appropriate resources to carry them out
Controller

MVC


LoginController
2. Call controller
3.Checks whether user is valid
4. Forward
Authenticator
Login.jsp
success.jsp
1. Fill data and submit
browser

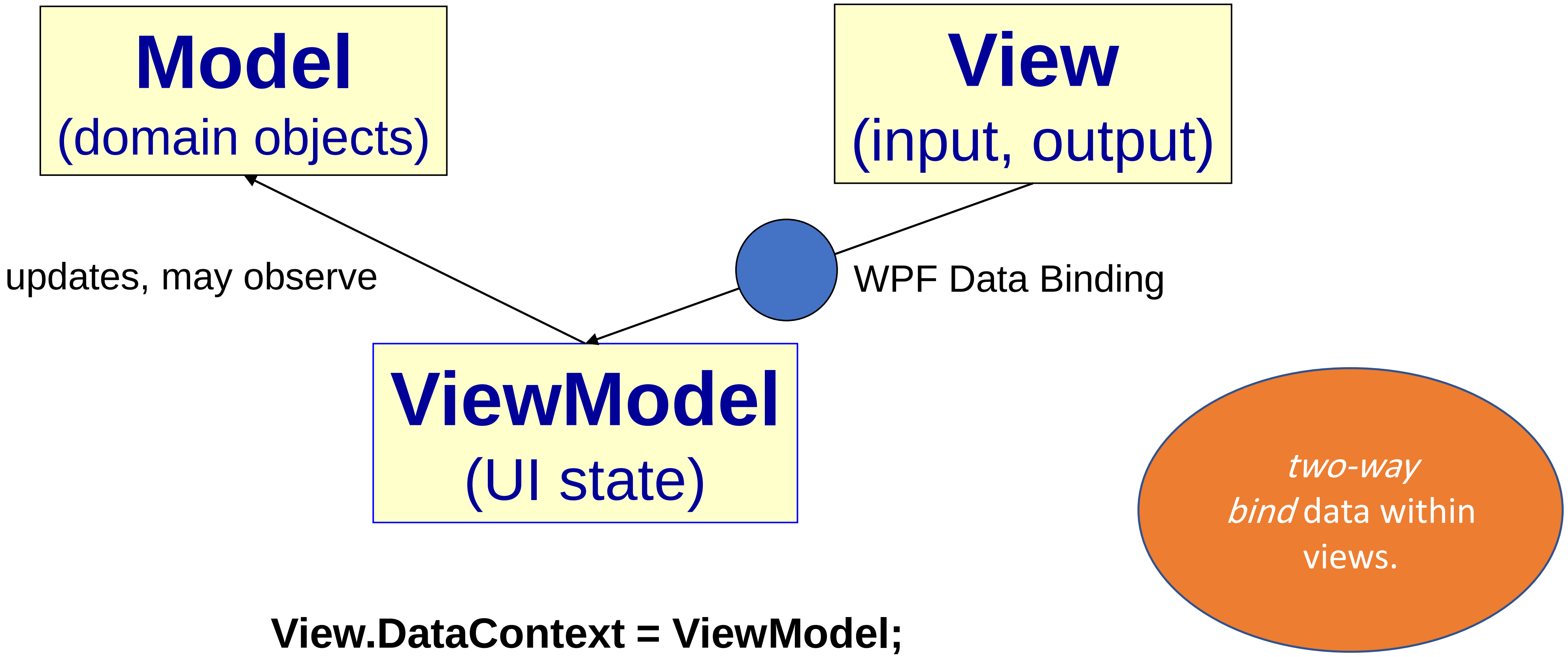
MVC example


View
Login.jsp
Success.jsp
Error.jsp
Controller
LoginController.java
Model
User.java
Authenticator.java

MVVM

Overview of MVVM

Model-View-ViewModel



View.DataContext = ViewModel;

http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx

Overview of MVVM

Separation of Concerns

Model	View	ViewModel
Read list of countries from the database	Position UI elements on screen	Validate input and show error indicators if necessary
Create shipment	Control visual appearance of the UI elements: colors, fonts, etc.	Call model to create shipment with data entered by the user
	Translate keystrokes to navigation and edit actions	Disable subdivision combo box if selected country has no subdivisions
	Translate mouse clicks to focus changes and button commands	

Overview of MVVM

Important MVVM Traits

- View is isolated from the model
- ViewModel does not manipulate controls directly
- **Most** of the View $\leftrightarrow$ ViewModel interaction is via data binding
- Codebehind is therefore kept to a minimum

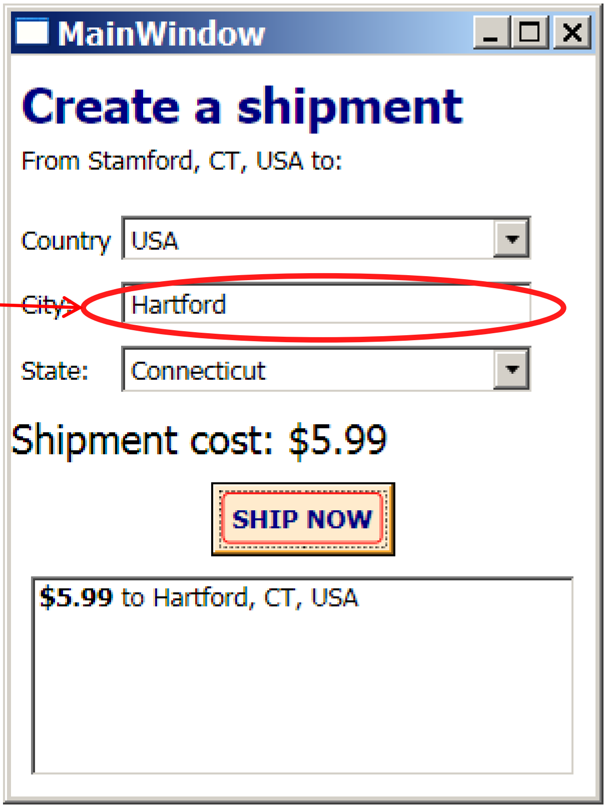
Overview of MVVM

WPF Data Binding

<TextBox Text="{Binding City}" />

```
class MainWindowViewModel
{
    public string City
    {
        get { … }
        set { … }
    }
    …
}
```

binding



What is the difference?

Does view model replace controller (MVVM vs MVC)

MVVM VS MVC

- MVVM
  - Client side
  - Two way binding data
- MVC
  - a way of separating concerns *on the server-side*.

Angular js

Slides modified from Stanford University CS142

Slides from CMU university Prof.**Michael J. McCarthy**

What is Angular js?

- MVVM Java Script Framework by Google for Rich Web Application Development
- Why?

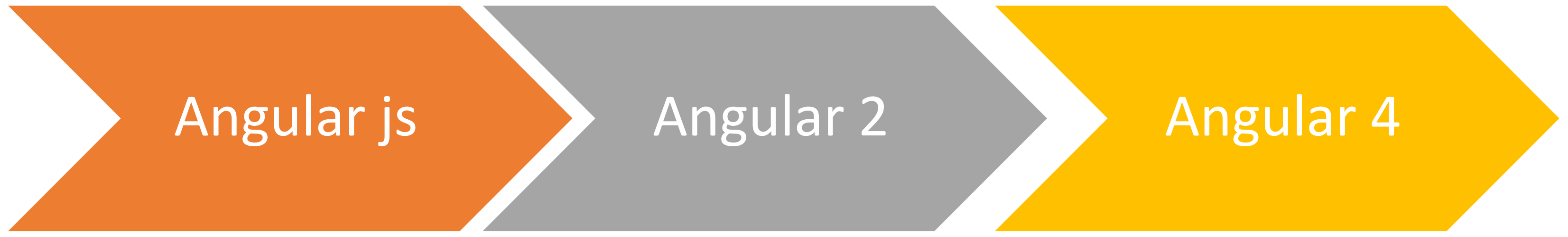
  Overcome HTML Shortcoming

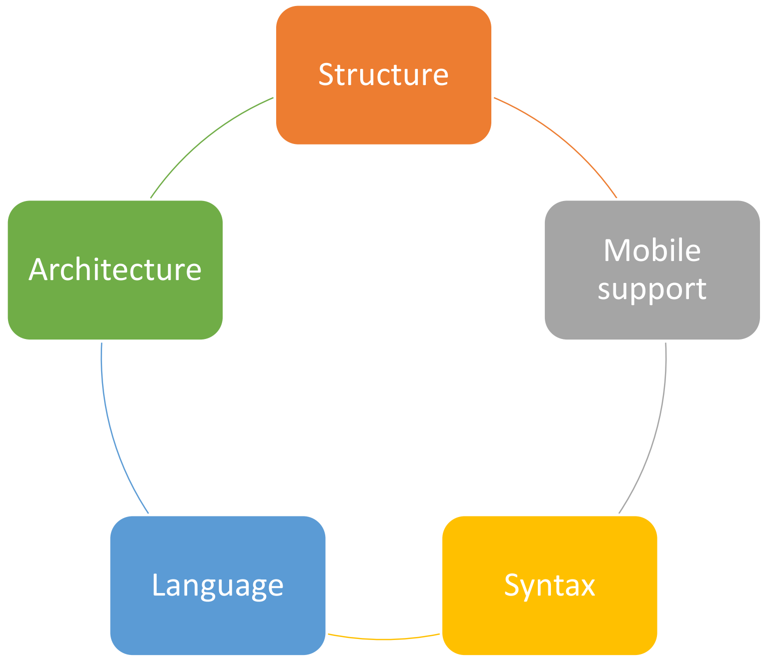
Why Angular?

“Other frameworks deal with HTML’s shortcomings by either abstracting away HTML, CSS, and/or JavaScript or by providing an imperative way for manipulating the DOM. Neither of these address the root problem that HTML was not designed for dynamic views”.

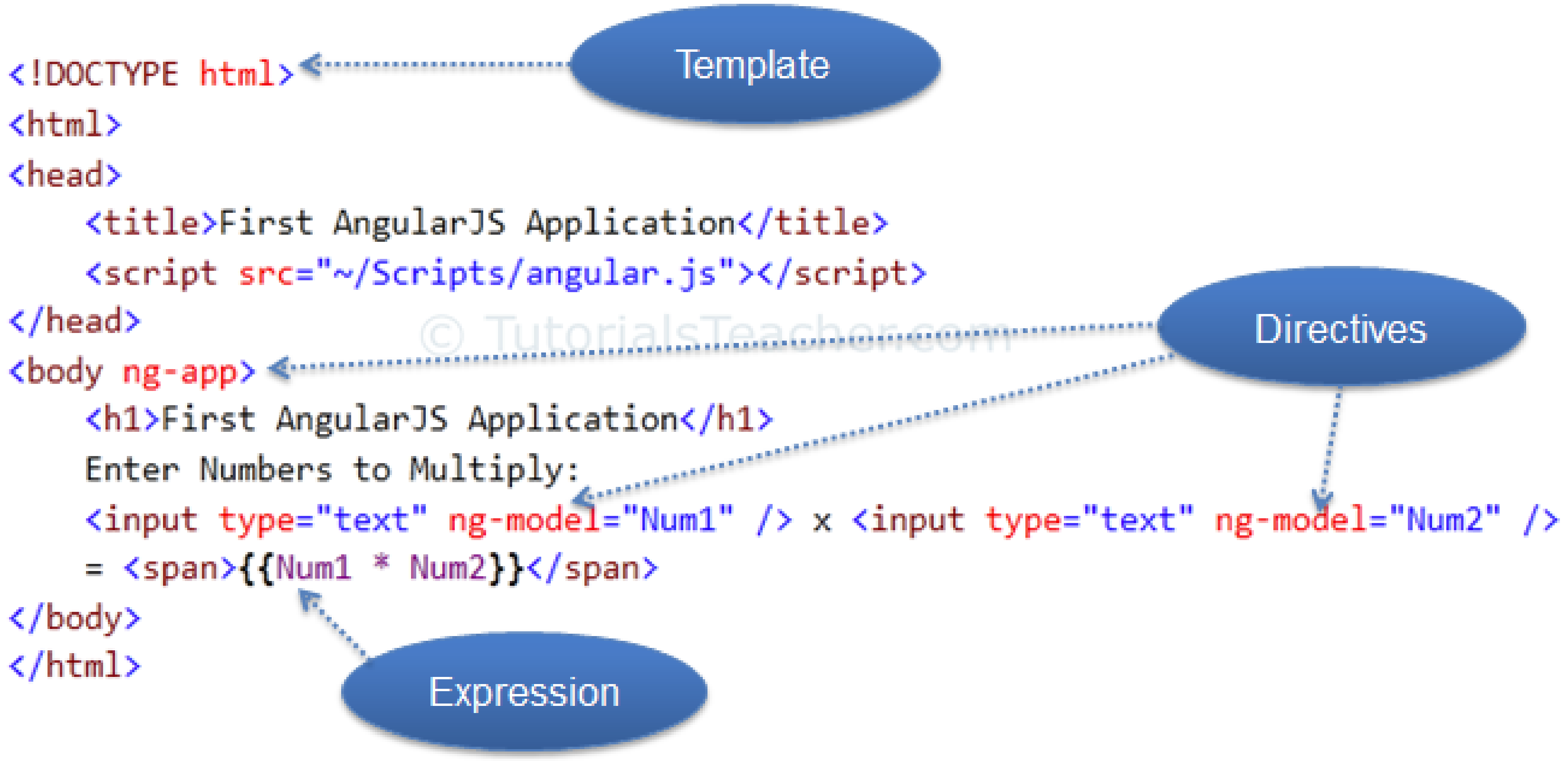
- Structure, Quality and Organization
- Lightweight (< 36KB compressed and minified)
- Free
- Separation of concern
- Modularity
- Extensibility & Maintainability
- Reusable Components

“ HTML? Build UI Declaratively! CSS? Animations! JavaScript? Use it the plain old way!”


Angular js
Angular 2
Angular 4


Structure
Mobile support
Syntax
Language
Architecture

Big picture



```
<!DOCTYPE html>
<html>
<head>
    <title>First AngularJS Application</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app>
    <h1>First AngularJS Application</h1>
    Enter Numbers to Multiply:
    <input type="text" ng-model="Num1" /> x <input type="text" ng-model="Num2" />
    = <span>{{Num1 * Num2}}</span>
</body>
</html>
```

First AngularJS Application

In AngularJS, a template is HTML with additional markups. AngularJS compiles templates and renders the resultant HTML.

JQuery

- Allows for DOM Manipulation
- Does not provide structure to your code
- Does not allow for two way binding

No installation

- It is recommended that you load the AngularJS library either in the <head> or at the start of the <body>.

Simple Angular js Page

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="">
<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="name"></p>
<p ng-bind="name"></p>
</div>
</body>
</html>
```

Features of AngularJS

- Two-way Data Binding – Model as single source of truth
- Directives – Extend HTML
- MVVM and MVC
- Dependency Injection
- Testing
- Deep Linking (Map URL to route Definition)
- Server-Side Communication

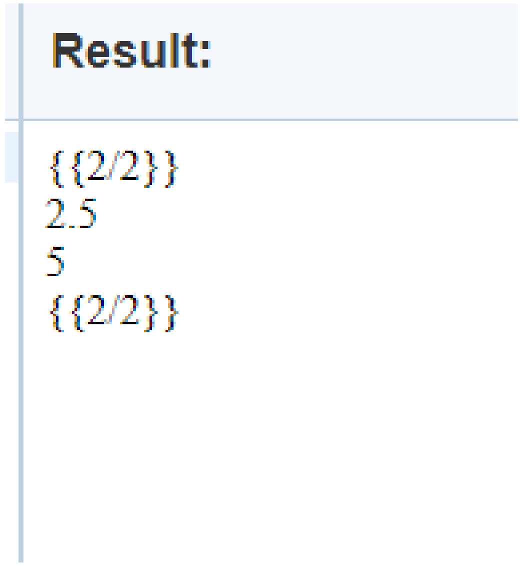
Data Binding

```
<html ng-app>
<head>
  <script src='angular.js'></script>
</head>
<body>
  <input ng-model='user.name'>
  <div ng-show='user.name'>Hi {{user.name}}</div>
</body>
</html>
```

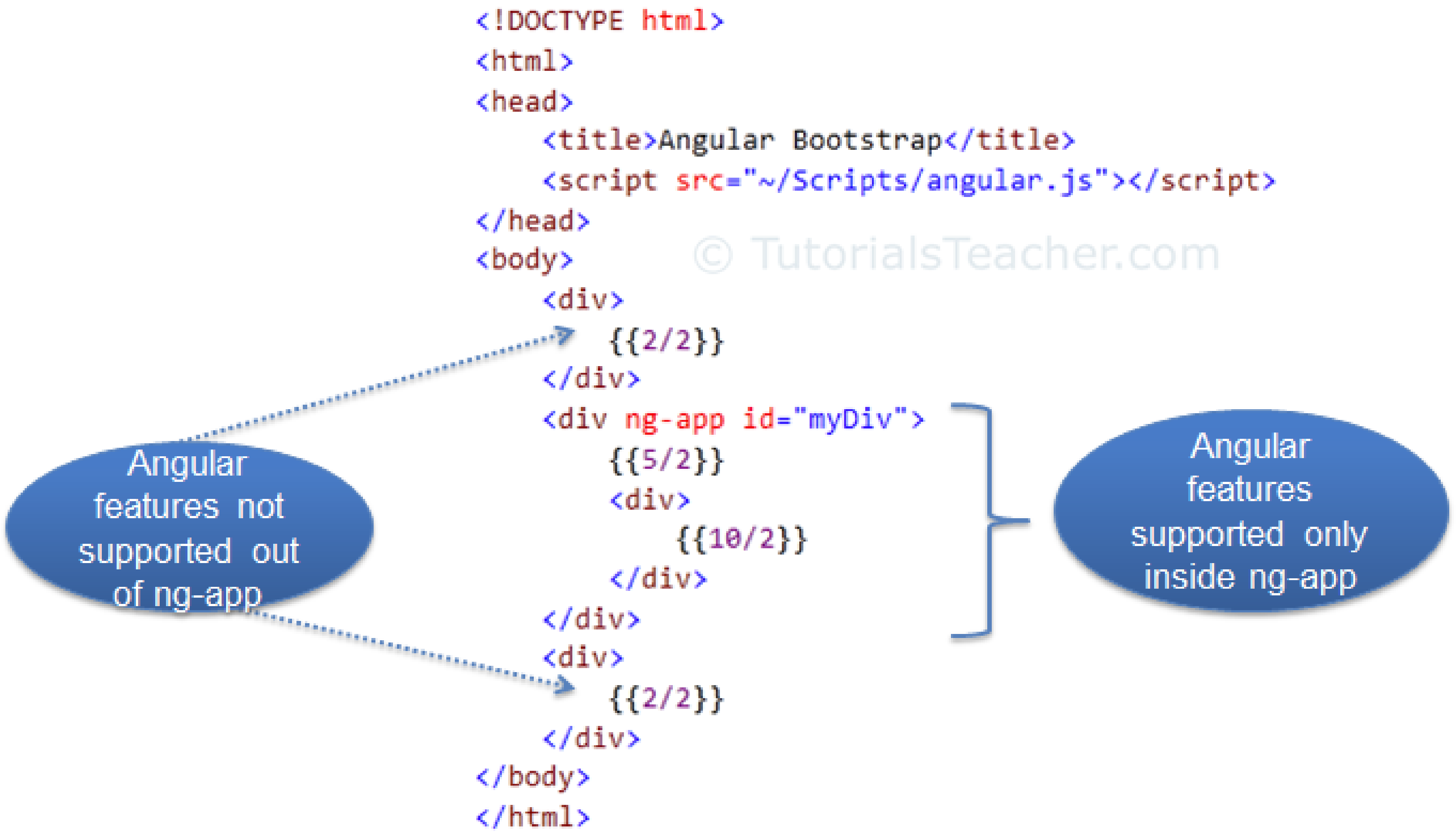
ng-app in <div>

```
<!DOCTYPE html>
<html>
<head>
<title>ng-app Directive</title>
<script src="../Scripts/angular.min.js"></script>
</head>
<body >
<div> {{2/2}} </div>
<div id="myDiv" ng-app>
{{5/2}}
       <div> {{10/2}} </div>
</div>
<div>{{2/2}}</div>
 </body>
 </html>
```

AngularJS framework will only process the DOM elements and its child elements where the ng-app directive is applied


Result:

```
{{2/2}}
2.5
5
{{2/2}}
```

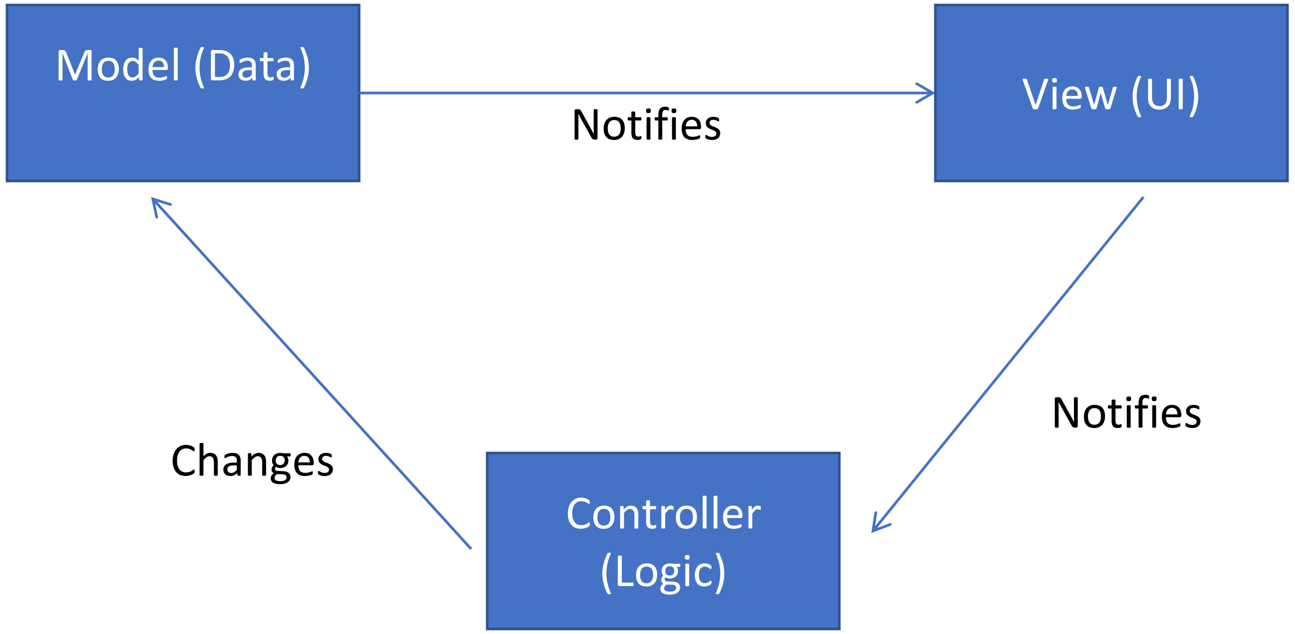


```
<!DOCTYPE html>
<html>
<head>
    <title>Angular Bootstrap</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body>
    <div>
        {{2/2}}
    </div>
    <div ng-app id="myDiv">
        {{5/2}}
        <div>
            {{10/2}}
        </div>
    </div>
    <div>
        {{2/2}}
    </div>
</body>
</html>
```

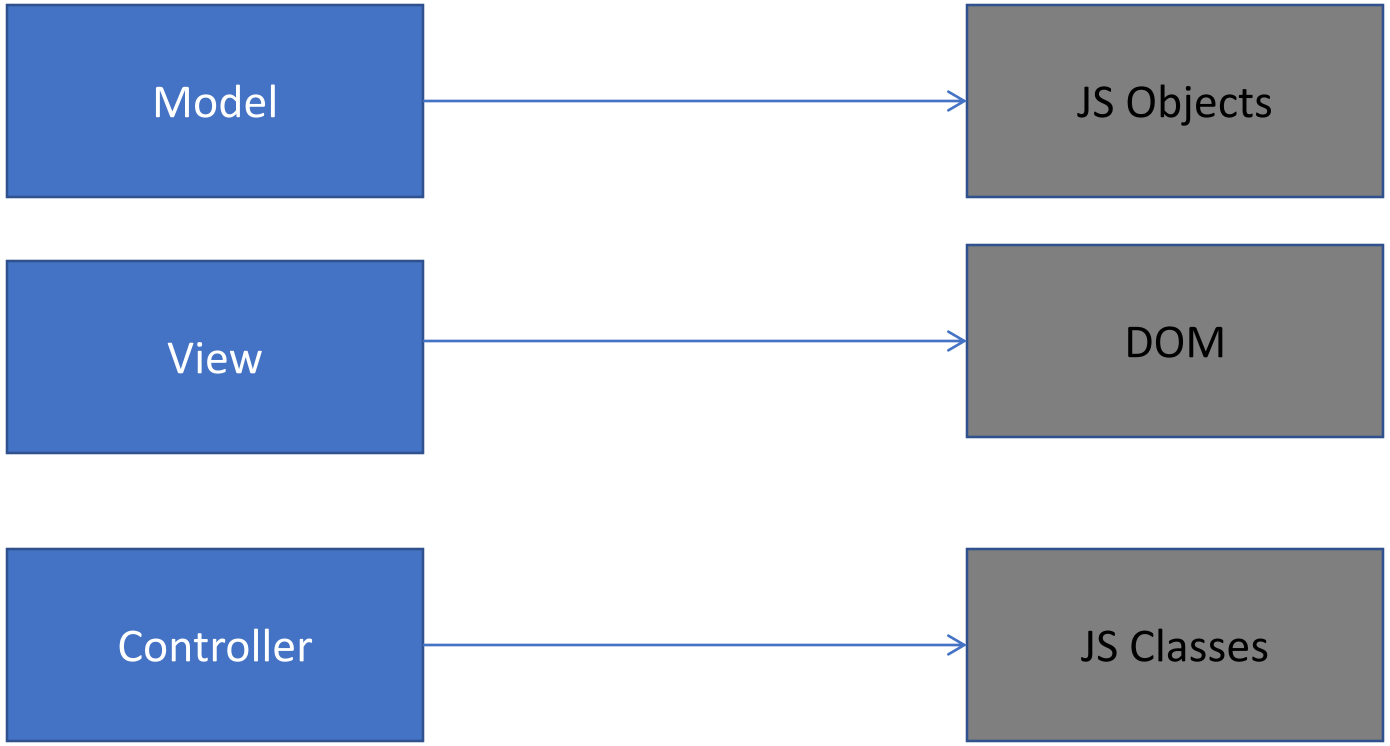
Angular features not supported out of ng-app
Angular features supported only inside ng-app

Bootstrap

MVC


Model (Data)
Notifies
View (UI)
Notifies
Changes
Controller (Logic)

MVC



MVC

```
<html ng-app>
<head>
  <script src='angular.js'></script>
  <script src='controllers.js'></script>
</head>
<body ng-controller='UserController'>
  <div>Hi {{user.name}}</div>
</body>
</html>

function  XXXX($scope) {
  $scope.user = { name:'Larry' };
}
```

Hello HTML

```
<p>Hello World!</p>
```

Hello Javascript

```
<p id="greeting1"></p>
<script>
var isIE = document.attachEvent;
var addListener = isIE ? function(e, t, fn) {
      e.attachEvent('on' + t, fn);}
  : function(e, t, fn) {
      e.addEventListener(t, fn, false);};
addListener(document, 'load', function(){
 var greeting = document.getElementById('greeting1');
  if (isIE) {
    greeting.innerText = 'Hello World!';
  } else {
    greeting.textContent = 'Hello World!'; }});
</script>
```

Hello JQuery

```
<p id="greeting2"></p>


<script>
$(function(){
  $('#greeting2').text('Hello World!');
});
</script>
```

Hello AngularJS

```
<p ng:init="greeting = 'Hello World!'">{{greeting}}</p>
```

Step by Step tutorial (very simple)

Feeder App

http://www.toptal.com/angular-js/a-step-by-step-guide-to-your-first-angularjs-app

Execution sequence for typical Angular js App

- When we access .html file from browser, the browser initially loads DOM.
- While loading the AngularJs, it creates the AngularJs global object
- This angular object will compile and executes the AngularJs related elements.
- AngularJS framework generates dynamic content, based on the directives(ng-app, ng-model) which we used in the html file.
    - ng-app directory is used to specify the region on which we can apply the AngularJs. It initializes the AngularJs application.
    - ng-init directory is used to initialize the AngularJs application data.
    - ng-model directive represents to bind the input value in html with AngularJs variables. It represents the two way binding in AngularJs.
    - {{x}} is a AngularJs ***expression***, it represents one way binding in AngularJs

Expressions

Expressions allow you to execute some computation in order to return a desired value.

AngularJS expressions are written inside double braces: **{{ expression }}**.

- {{ 1 + 1 }}
- {{ 946757880 | date }}
- {{ user.name }}

AngularJS will "output" data exactly where the expression is written:

you shouldn't use expressions to implement any higher-level logic.

Directives

- Directives are markers (such as attributes, tags, and class names) that tell AngularJS to attach a given behavior to a DOM element (or transform it, replace it, etc.)

- Directives are HTML attributes with an **ng** prefix.
- You can use **data-ng-**, instead of **ng-**, if you want to make your page HTML valid.

Some directives

- The **ng-app** - Bootstrapping your app and defining its scope.
- The **ng-controller** - defines which controller will be in charge of your view.
- The **ng-repeat** - Allows for looping through collections
- The **ng-app** directive tells AngularJS that the <div> element is the "owner" of an AngularJS **application**.
- The **ng-model** directive binds the value of the input field to the application variable **name**.
- The **ng-bind** directive binds the content of the <p> element to the application variable **name**.
- https://www.w3schools.com/angular/angular_ref_directives.asp

ng-init

- **ng-init** directive initializes AngularJS application variables.

```
<div ng-app="" ng-init="firstName='John'">

<p>The name is <span ng-bind="firstName"></span></p>

</div>
```

Angular Datatypes

- Exactly the same as JavaScript
- String
- Numbers
- Objects
- Array
- Same syntax
- Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.
- Unlike JavaScript expressions, *AngularJS expressions can be written inside HTML.*

Angular module

- The ng-app directive can also specify an application module name. This application module separates different parts of your application such as controllers, services, filters etc.

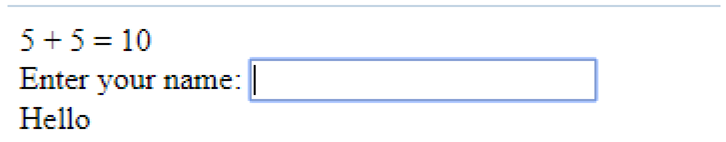
```
<!DOCTYPE html>
<html>
<head>
    <title>ng-app Directive</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app="myAngularApp">
    <div>
        {{2/2}}
    </div>
    <div>
        {{5/2}}
        <div>
            {{10/2}}
        </div>
    </div>
    <script>
        var app = angular.module('myAngularApp', []);
    </script>
</body>
</html>
```

ng-model and ng-bind

- The ng-model directive is used for two-way data binding in AngularJS. It binds <input>, <select> or <textarea> elements to a specified property on the $scope object. So, the value of the element will be the value of a property
- The ng-bind directive binds the model property declared via $scope or ng-model directive or the result of an expression to the HTML element.
- It also updates an element if the value of an expression changes

Example

```
<!DOCTYPE html>
<html >
<head>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app="">
    <div>
        5 + 5 = <span ng-bind="5 + 5"></span> <br />

        Enter your name: <input type="text" ng-model="name" /><br />
        Hello <span ng-bind="name"></span>
    </div>
</body>
</html>
```



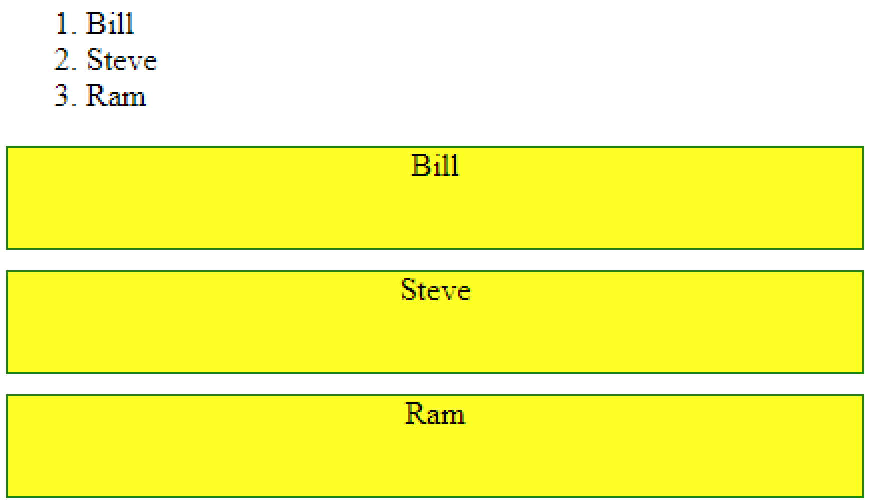
```
5 + 5 = 10
Enter your name: 
Hello
```

Try it here

ng-repeat

```
<!DOCTYPE html>
<html>
<head>
    <script src="~/Scripts/angular.js"></script>
    <style>
        div {
            border: 1px solid green;
            width: 100%;
            height: 50px;
            display: block;
            margin-bottom: 10px;
            text-align:center;
            background-color:yellow;
        }
    </style>
</head>
<body ng-app="" ng-init="students=['Bill','Steve','Ram']">
    <ol>
        <li ng-repeat="name in students">
            {{name}}
        </li>
    </ol>
    <div ng-repeat="name in students">
        {{name}}
    </div>
</body>
</html>
```

The ng-repeat directive repeats HTML once per each item in the specified array collection.

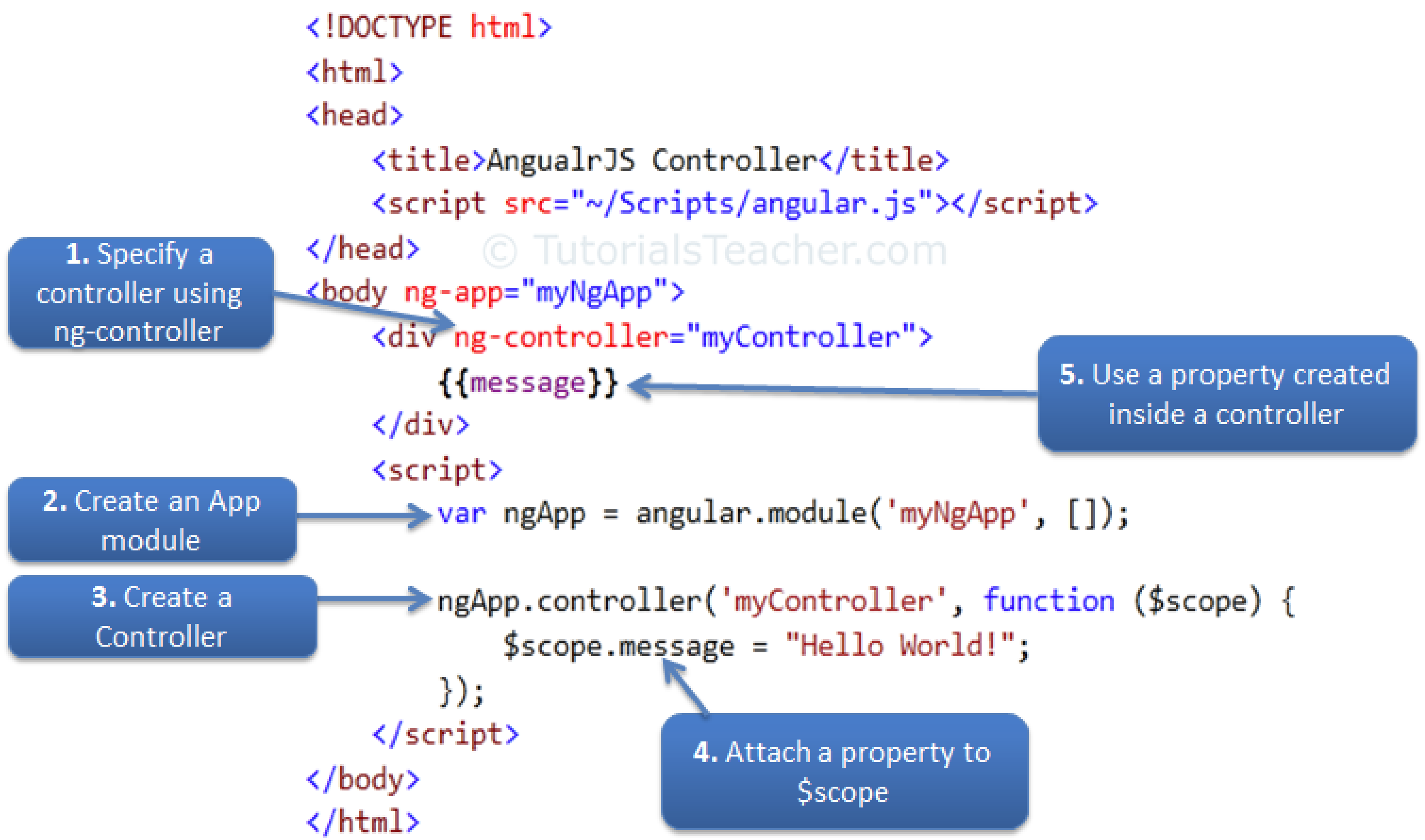


```
1. Bill
2. Steve
3. Ram

Bill

Steve

Ram
```

Angular Controller

- The controller in AngularJS is a JavaScript function that maintains the application data and behavior using $scope object.
- The $scope object is a glue between the controller and HTML

- Illustrated in the next slide.



```
<!DOCTYPE html>
<html>
<head>
    <title>AngualrJS Controller</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app="myNgApp">
    <div ng-controller="myController">
        {{message}}
    </div>
    <script>
        var ngApp = angular.module('myNgApp', []);

        ngApp.controller('myController', function ($scope) {
            $scope.message = "Hello World!";
        });
    </script>
</body>
</html>
```

1. Specify a controller using ng-controller
2. Create an App module
3. Create a Controller
4. Attach a property to $scope
5. Use a property created inside a controller

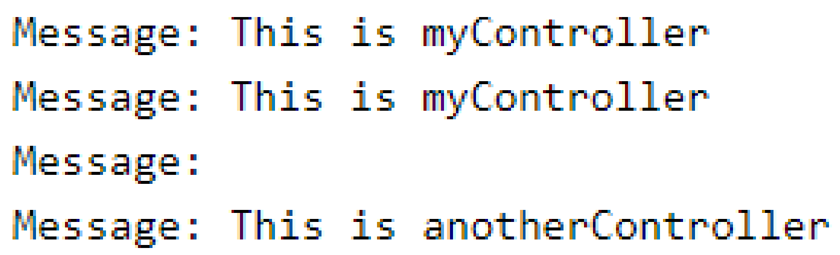
Steps to create an AngularJS Controller

- Try it here

Note that the properties and methods attached to the scope object inside a particular controller is only available to the HTML elements and its child elements where ng-controller directive is applied.

Two controllers

```html
<!DOCTYPE html>
<html>
<head>
    <title>AngualrJS Controller</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app="myNgApp">
    <div id="div1" ng-controller="myController">
        Message: {{message}} <br />
        <div id="div2">
            Message: {{message}}
        </div>
    </div>
    <div id="div3">
        Message: {{message}}
    </div>
    <div id="div4" ng-controller="anotherController">
        Message: {{message}}
    </div>
    <script>
        var ngApp = angular.module('myNgApp', []);

        ngApp.controller('myController', function ($scope) {
            $scope.message = "This is myController";
        });

        ngApp.controller('anotherController', function ($scope) {
            $scope.message = "This is anotherController";
        });
    </script>
</body>
</html>
```



```
Message: This is myController
Message: This is myController
Message:
Message: This is anotherController
```

Structure

- It is common in AngularJS applications to put the module and the controllers in JavaScript files.
- In this example, "myApp.js" contains an application module definition, while "myCtrl.js" contains the controller:

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script src="myApp.js"></script>
<script src="myCtrl.js"></script>

</body>
</html>
```

myApp.js

```
var app = angular.module("myApp", []);
```

myCtrl.js

```
app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName= "Doe";});
```

Angularjs

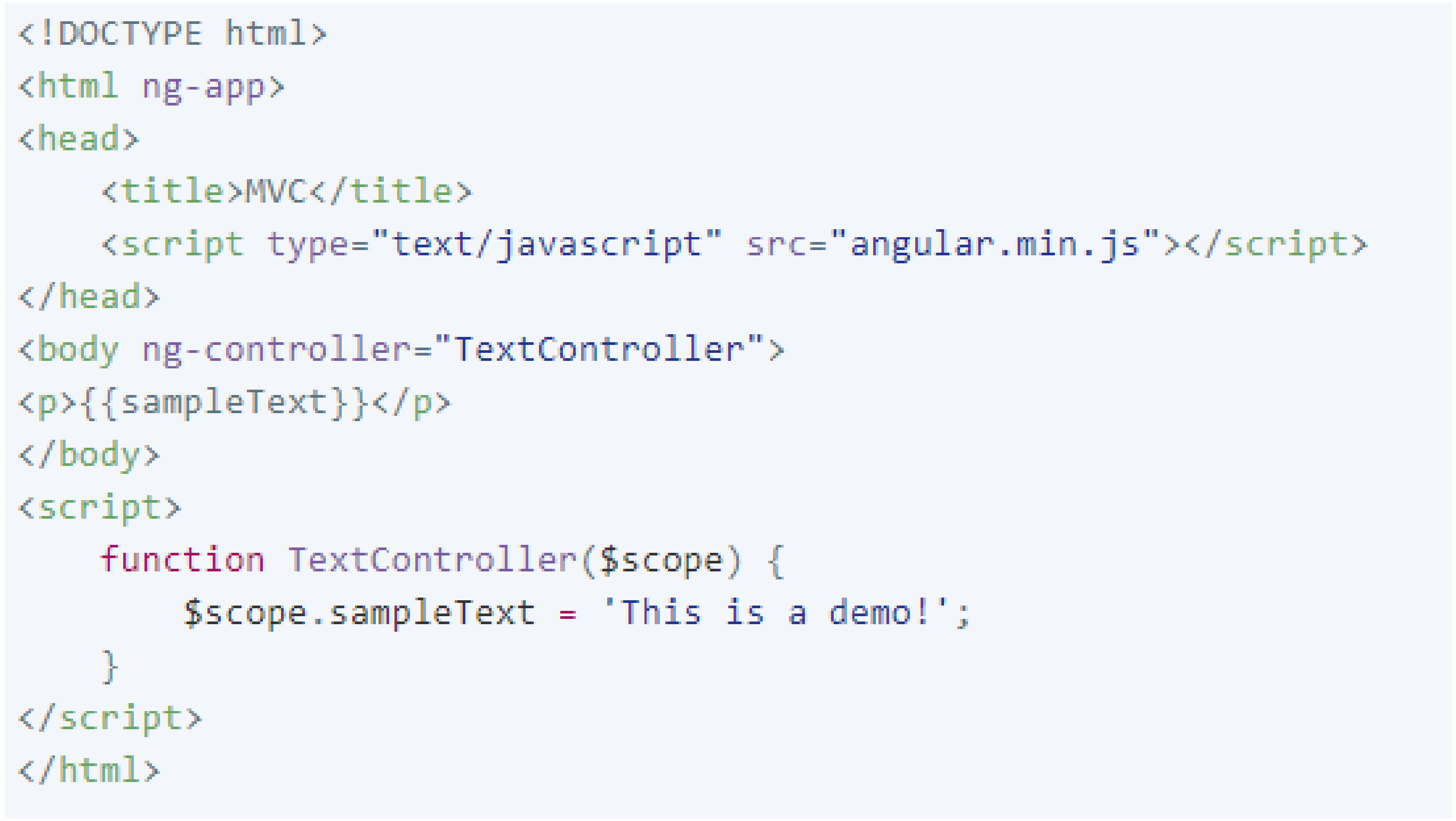
MVC

- Model: **Model** is nothing but data.
- View: **View** represents this data.
- Controller: **Controller** mediates between the two.

MVVM

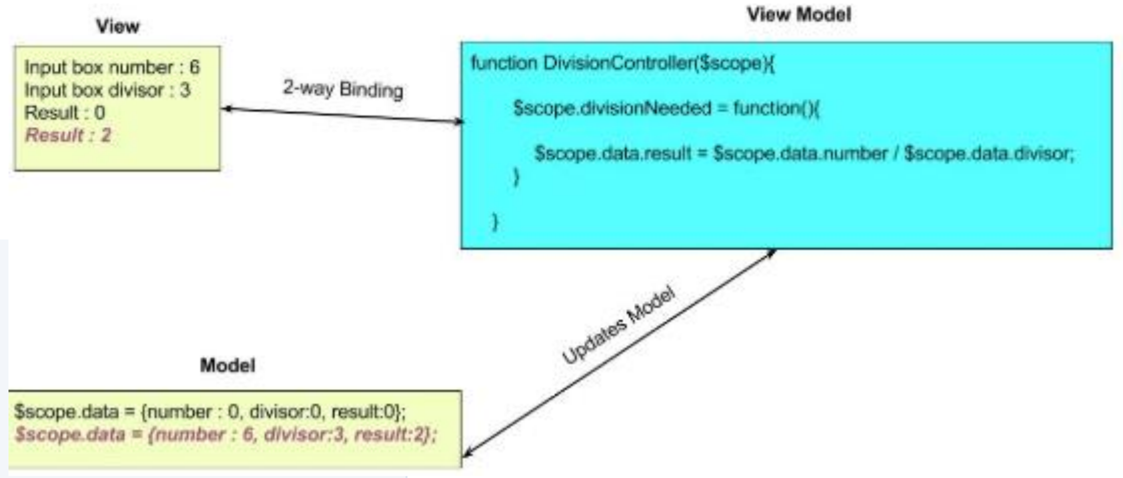
- Model
- View
- View model (JavaScript function)
- if we update anything in **view**, it gets updated in **model**, change anything in **model**, it shows up in **view**, which is what we call **2-way binding**

Angularjs MVC



```html
<!DOCTYPE html>
<html ng-app>
<head>
    <title>MVC</title>
    <script type="text/javascript" src="angular.min.js"></script>
</head>
<body ng-controller="TextController">
<p>{{sampleText}}</p>
</body>
<script>
    function TextController($scope) {
        $scope.sampleText = 'This is a demo!';
    }
</script>
</html>
```

Angularjs MVVM



```
Input box number : 6
Input box divisor : 3
Result : 0
Result : 2
```

```
function DivisionController($scope){

    $scope.divisionNeeded = function(){

        $scope.data.result = $scope.data.number / $scope.data.divisor;
    }

}
```

```
$scope.data = {number : 0, divisor:0, result:0};
$scope.data = {number : 6, divisor:3, result:2};
```

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Number Divisor</title>
    <script type="text/javascript" src="angular.min.js"></script>
</head>
<body>
<form ng-controller="DivisionController">
    <label>Number :</label> <input name="number" ng-change="divisionNeeded()" ng-model="data.number">
    <label>Number entered by User :</label> {{data.number}} <br>
    <label>Divisor :</label> <input name="divisor" ng-change="divisionNeeded()" ng-model="data.divisor"
    <label>Number entered by User :</label> {{data.divisor}} <br>
    <label>Result :</label> {{data.result}}
</form>
</body>
<script>
    function DivisionController($scope) {
        $scope.data = {number: 0, divisor: 0, result: 0};
        $scope.divisionNeeded = function () {
            $scope.data.result = $scope.data.number / $scope.data.divisor;
        }
    }
</script>
</html>
```

Disadvantages of AngularJS

- **Not Secure –** Its applications are not safe. Server side authentication and authorization is necessary to keep an application secure.
- **Not Degradable –** If user of your application disables the JavaScript then it displays nothing except basic page.
- **Complex at times –** At times AngularJS becomes complex to handles as there are multiple ways to do the same thing. This creates confusion and requires considerable efforts.

Useful Links

- https://angularjs.org/
- http://www.toptal.com/angular-js/a-step-by-step-guide-to-your-first-angularjs-app
- https://github.com/raonibr/f1feeder-part1
- Don’t forget *Your Best Friend W3Schools*

Tutorial

No Copying
No tutorials from the web
Your code will be checked
for Plagiarism

- Dynamic form using angular js.
- Create a small form that takes multiple input (user sign up, order, student registration, etc)
- Utilize Angularjs (not angular 2 or angular 4) by creating a controller for this app
- Utilize Angularjs client-side form validation capabilities.
- <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
- https://www.w3schools.com/angular/angular_validation.asp
- https://www.w3schools.com/angular/angular_http.asp

For Next week

- We will Cover Web Storage
- Reading
  - Node.js in Action chapter 8 (covers databases)
  - Review MYSQL
  - https://www.w3schools.com/html/html5_webstorage.asp
  - https://www.w3schools.com/nodejs/nodejs_mysql.asp
  - https://www.w3schools.com/nodejs/nodejs_mongodb_create_db.asp
- Install
  - Node.js
  - MongoDB
  - MYSQL(if you have XAMPP installed you probably have MySQL installed with it)