Usage Guidelines

Angular JS
Day1

Education, Training and Assessment

We enable you to leverage knowledge anytime, anywhere!

Infosys® | Building Tomorrow's Enterprise

Copyright Guideline

Confidential Information

Course Information

Course Code	: LA1211
Course Name	: AngularJS
Version Number	: 1.2

Session Plan

- Introducing AngularJS
- Application Structure
- Bootstrapping Angular
- Angular Building Blocks-Model, View and Controller
- Scopes in Angular
- Angular Runtime
- Angular Expressions
- Directives
- Filters

Session Plan

- Dependency injection
- Services
- Modularizing Angular Applications
- Routing in Angular
- Custom components in Angular
- Form Handling in Angular
- Backbone v/s Angular

Day1 Objectives

- Introducing AngularJS
  - Introduction to AngularJS
  - Key Features
  - Philosophy
  - Multi-page application Architecture
  - MVC on server side
  - SPA
  - Emergence of MVC on client side
  - Getting Angular

Day 1 Objectives

- Application Structure
- Bootstrapping Angular
- Angular Building Blocks
  - Model
  - View
    - Understanding Views the Angular way
    - Two-way data binding
  - Controllers

References

- Brad Green, Shyam Seshadri, AngularJS , O'Reilly Media, 2013.
- http://docs.Angularjs.org/guide/
- https://github.com/Angular

Introducing AngularJS

Introduction To AngularJS

- Open-source JavaScript framework
- Intended for building Single Page Applications
- Funded by Google, was developed in 2009 by Misko Hevery and Adam Abrons
- Completely JavaScript and entirely client-side and hence can run anywhere where JavaScript can run.
- AngularJS works completely on the well-established technologies of web like HMTL, CSS and JavaScript.
- The technology lets the designer use HTML as template language and extend the HTML syntax to express the application components.

Introduction To AngularJS

- Include model-view-controller capability by providing a MVC framework.
- The MVC for a typical web application reduces development time, enforces a uniform structure and makes testing easier
- Angular automatically handles many of the tasks like DOM manipulation, setting event listeners and notifiers and validations on input.
- Documentation site: http://Angularjs.org

Key Features

- Minimal code
- Two-way data binding[1]
- MVC
- Dependency Injection[2]
  - The framework uses dependency injection techniques to connect the server and client side components.
  - This reduces the load on the backend
  - This eventually helps to create a lighter application
- REST ready[3]

Key Features

- Animations
- Templating
- Routing
- Testable
- No dependency on any external libraries.
- Clear separation between data, logic and presentation.
- Supports history, back button and forward buttons. It also supports bookmarking in singe-page apps.
- Works with many of the mobile browsers like Android, Chrome Mobile, iOS Safari, etc.

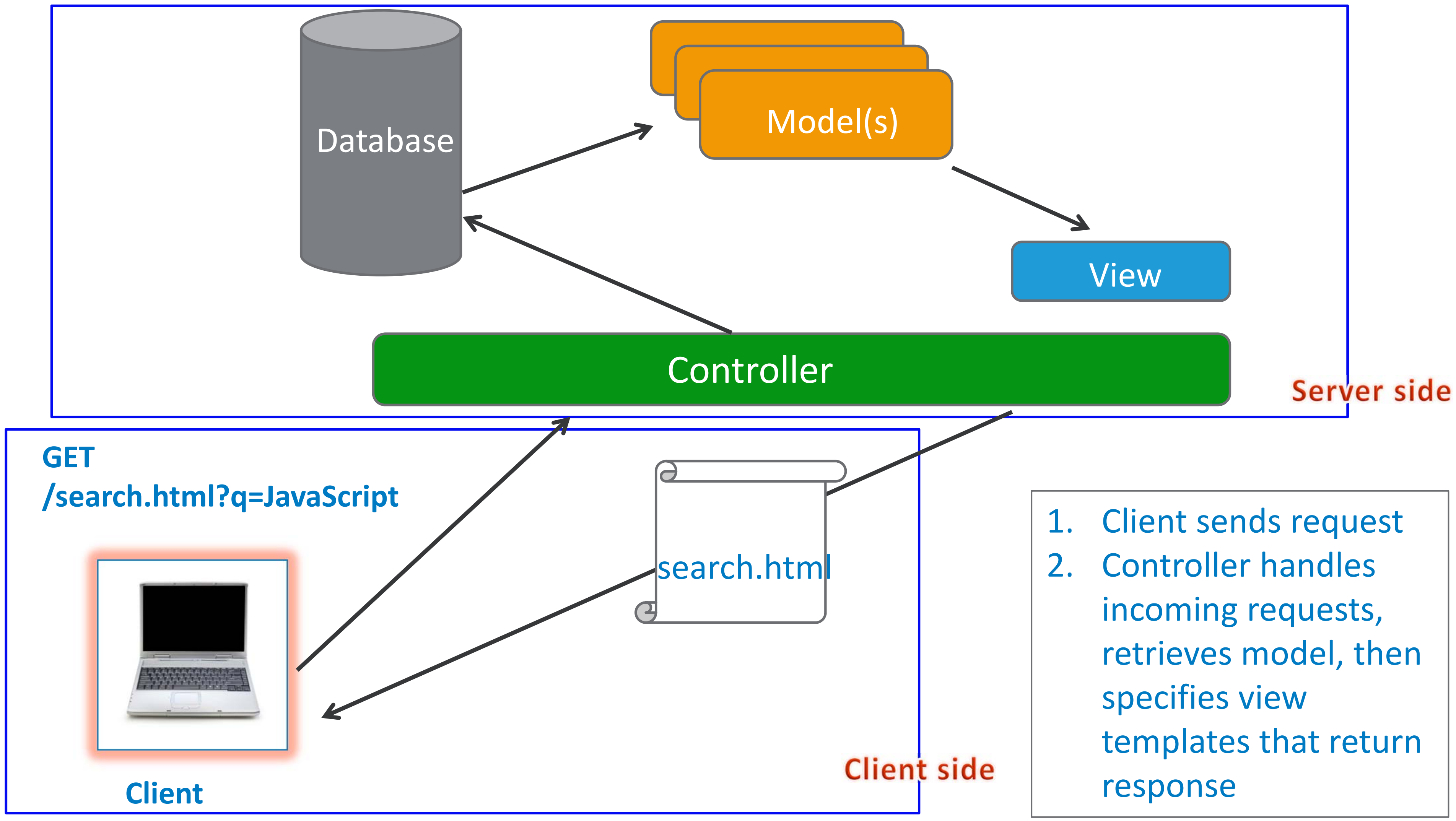
Philosophy

- Separate DOM manipulation & application logic
- Build UI declaratively

  &

  Write application logic imperatively[1]
- De-couple the client side of the application from the server side[2]
- Testability is very important

Pictorial Representation : (Multi-page Application Architecture)



1. Client sends request
2. Controller handles incoming requests, retrieves model, then specifies view templates that return response

MVC on server-side

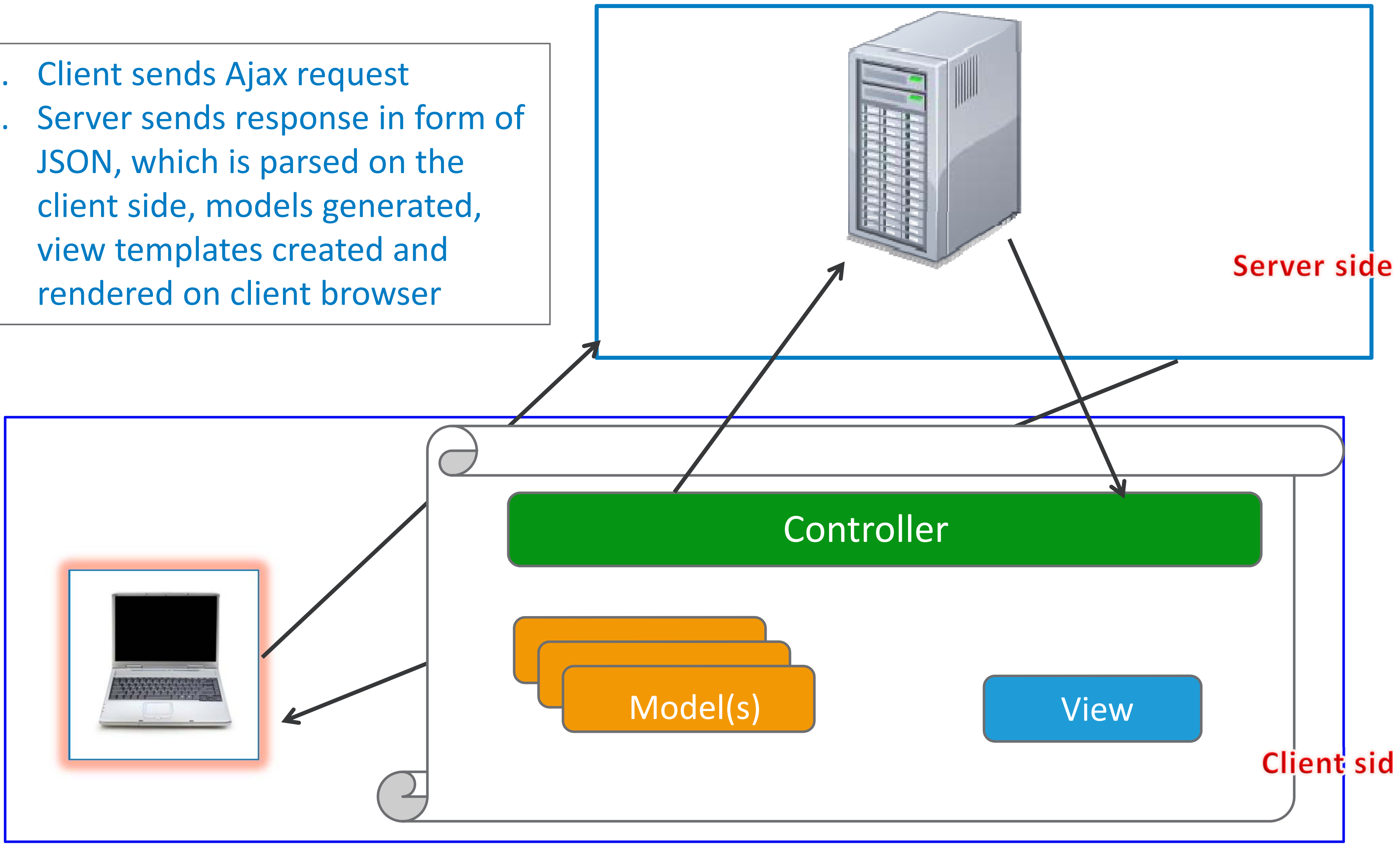
- MVC or Model-View-Controller design pattern was defined in 1979
- Several frameworks sporting this pattern could help in server side logic building and other operations, for ex : Struts,SpringMVC,Rails etc.
- Web-applications were rich on the server-side with all operations such as validations, event handling, generating response based on request from client etc. happening on the side of the server

Here MVC architecture helped by separating the responsibilities of :

- Routing to an appropriate handler which can read the request,
- Rendering template using view,
- Creating model to provide response based on action
- And coordination between the three when a request comes

Pictorial Representation : (Single Page Application Architecture)

1. Client sends Ajax request
2. Server sends response in form of JSON, which is parsed on the client side, models generated, view templates created and rendered on client browser



Emergence of MVC paradigm on client

- Rich-client side controlled applications started becoming famous due to the emergence of a new breed of JavaScript frameworks like Prototype, jQuery etc.
- This client-side rich approach had several advantages such as –
- Elegant, simple and cheaper to code
- State changes and handlers are local and extensive use of AJAX for making calls to the server
- Validation, event handling etc. happens on client side, thus reducing server load

Another reason for this change from rich server-side to rich client-side applications is because the browser now has become a **powerful** tool which can perform much of the logic and coding at the client side with ease thanks to the JavaScript frameworks and the server side has been reduced to performing data transactions primarily in the form of JSON

Getting Angular

- The library of AngularJS contains the AngularJS JavaScript file.
- We can downloaded the library from the official site.

http://angularjs.org

- Alternatively, the library can be accessed directly through the CDN.

https://ajax.googleapis.com/ajax/libs/angularjs/1.3.9/angular.min.js

Application Structure

Understanding the Structure of an Angular Page

```
<!doctype html>
<html ng-app>
<head>
<title>My First Angular Web
Page</title>
<style type="text/css">
        @import "styles/style.css";
    </style>
<script
src="lib/Angular/angular.js"></script>
</head>
<body>
```

Root of Angular App

Importing user-defined style sheet

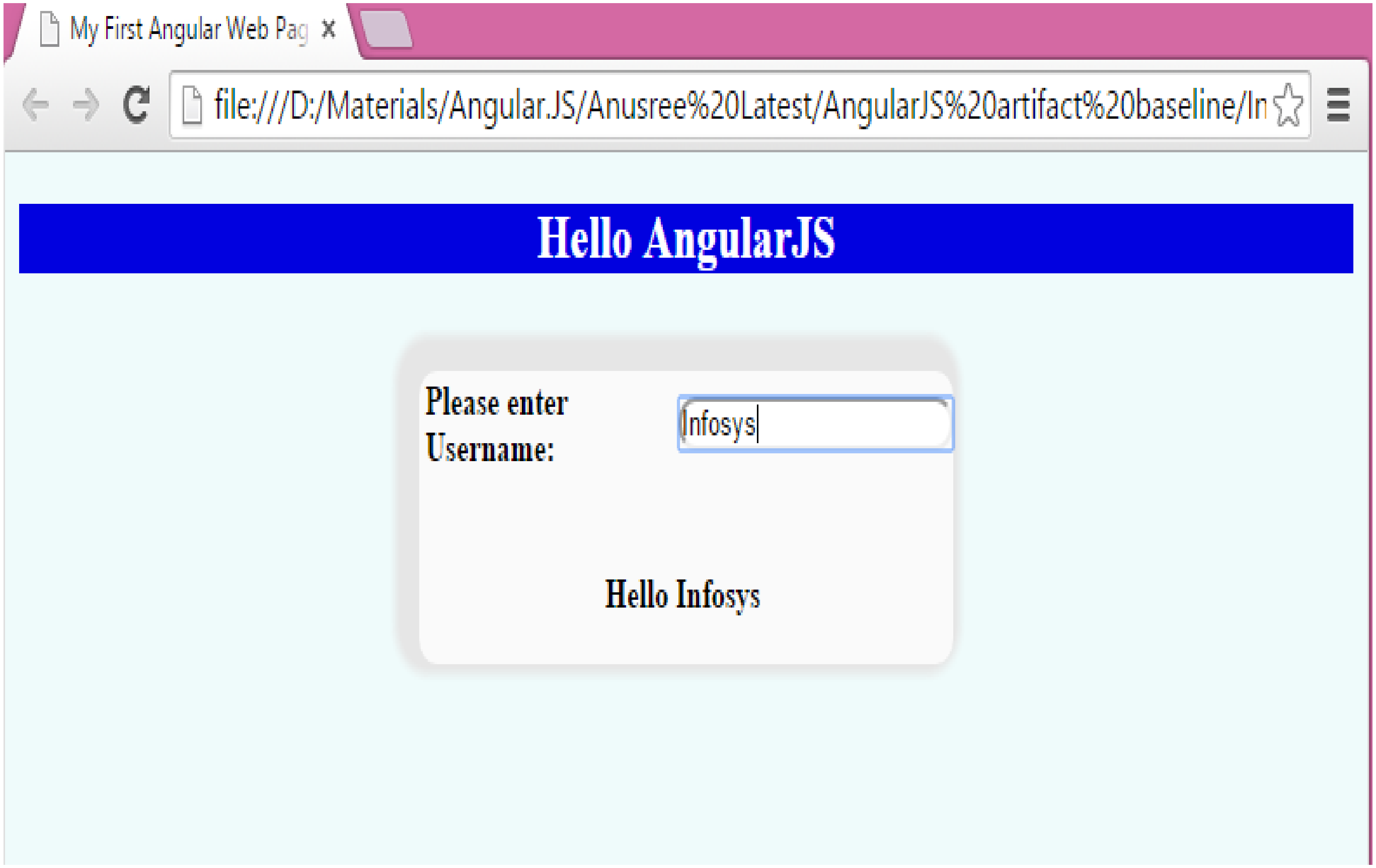
Including Angular Library

```
<center>
<header><h2>
{{ 'Hello '+' AngularJS' }} </h2></header><br>
 <div class="LoginFormDiv">
<table >
<tr>
<td><b>Please enter Username:</b></td>
<td><input  ng-model="username" ></td>
</tr>
</table>  <br><br>
<b> Hello {{username}} </b><br><br>
</div></center></body></html>
```

Angular Expression

Binding Angular Model

The First Angular Web page



My First Angular Web Pag
file:///D:/Materials/Angular.JS/Anusree%20Latest/AngularJS%20artifact%20baseline/In

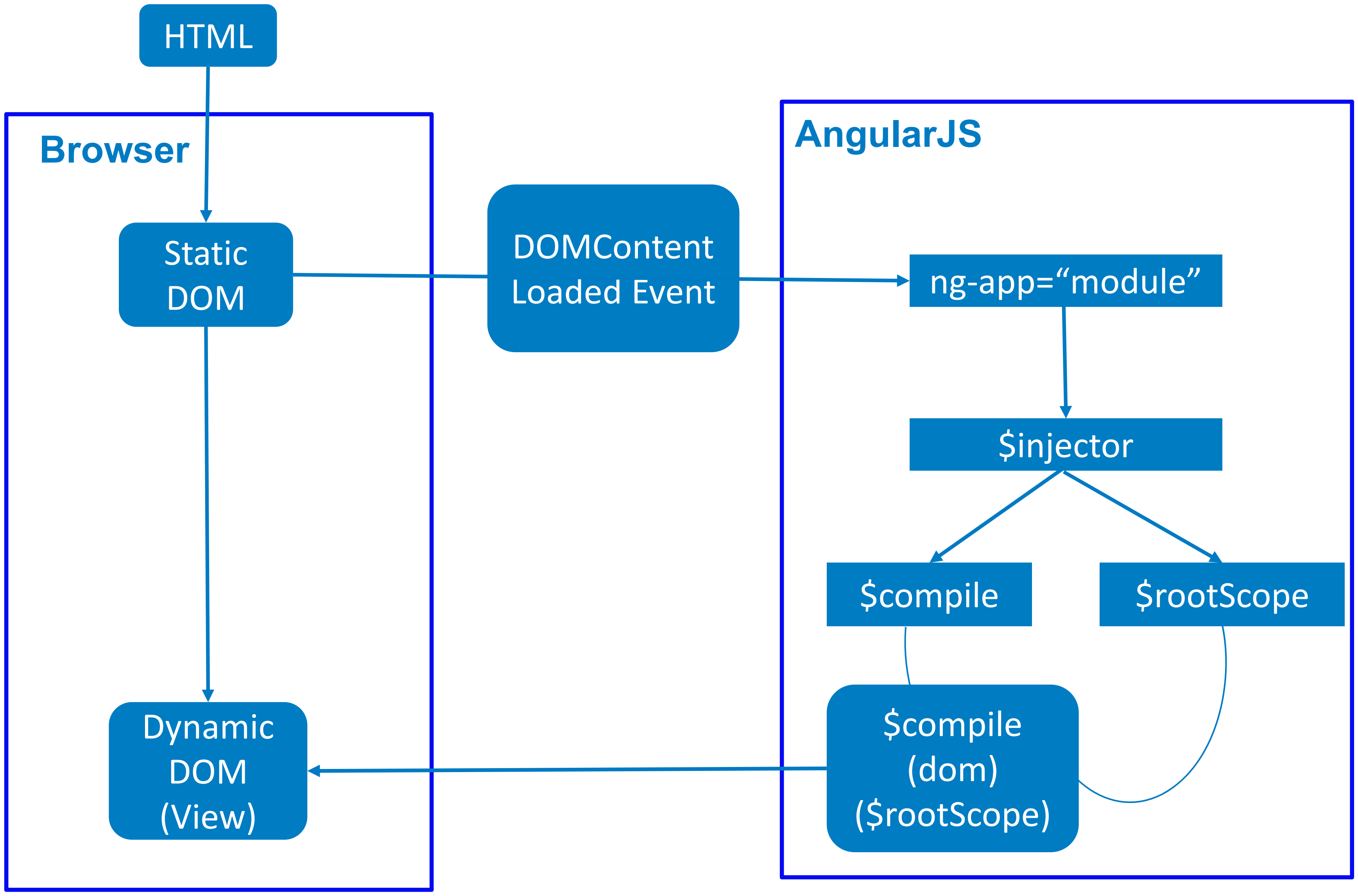
Hello AngularJS

Please enter Username: Infosys

Hello Infosys

Bootstrapping Angular

AngularJS: Bootstrap process

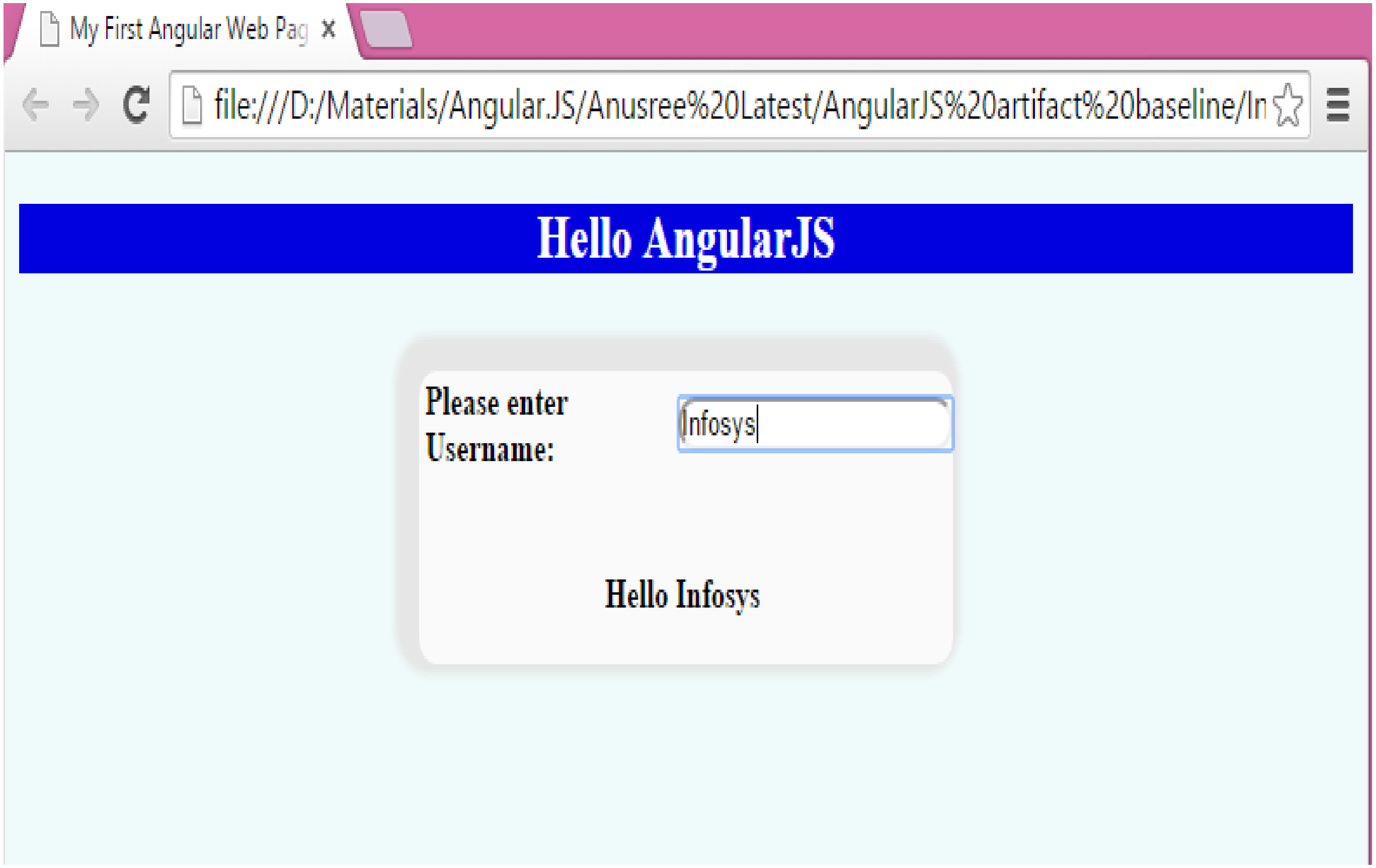


AngularJS: Automatic Initialization

Angular initializes automatically when DOMContentLoaded event is triggered.

1. Angular tries to find ng-app directive which designates the app root.
2. Once ng-app is found, the module associated with the directive is loaded
3. Application *injector* is created.
4. The HTML compiler walks the DOM to look out for attributes/directives.
5. The compiler compiles the DOM by treating ng-app directive as root of compilation. This results in a linking function.
6. The Link phase combines the directives with a scope and produces a live view. The link phase attaches all the directives to scope.

The First Angular Web page: Bootstrapping


My First Angular Web Pag

file:///D:/Materials/Angular.JS/Anusree%20Latest/AngularJS%20artifact%20baseline/In

Hello AngularJS

Please enter Username: Infosys

Hello Infosys

AngularJS: Automatic Initialization

```
<!doctype html>
<html ng-app>
<head>
<title>My First Angular Web
Page</title>
<style type="text/css">
          @import "styles/style.css";
      </style>
<script
src="lib/Angular/angular.js"></script>
</head>
<body>
```

1.Looks out for ng-app and loads associated module, if any.

```
<center>
<header><h2>
 Hello AngularJS  </h2></header><br>
 <div class="LoginFormDiv">
<table >
<tr>
<td><b>Please enter Username:</b></td>
<td><input  ng-model="username" ></td>
</tr>
</table>  <br><br>
<b> Hello Infosys </b><br><br>
</div></center></body></html>
```

3. Linking to produce live view

2. HTML compiler looks out for directives and compiles the same

3. Linking to produce live view

AngularJS: Manual Initialization

- AngularJS also provides the facility to manually control the initialization process.
- This option is generally used when some operations need to be performed before Angular compiles a page or when script loaders need to be used.
- Sequence to be followed is:
  - Once the page and its content are loaded, we have to find out the root of the application (root of the document).
  - We should then call api/Angular.bootstrap method to compile the template into an executable application.

AngularJS: Manual Initialization

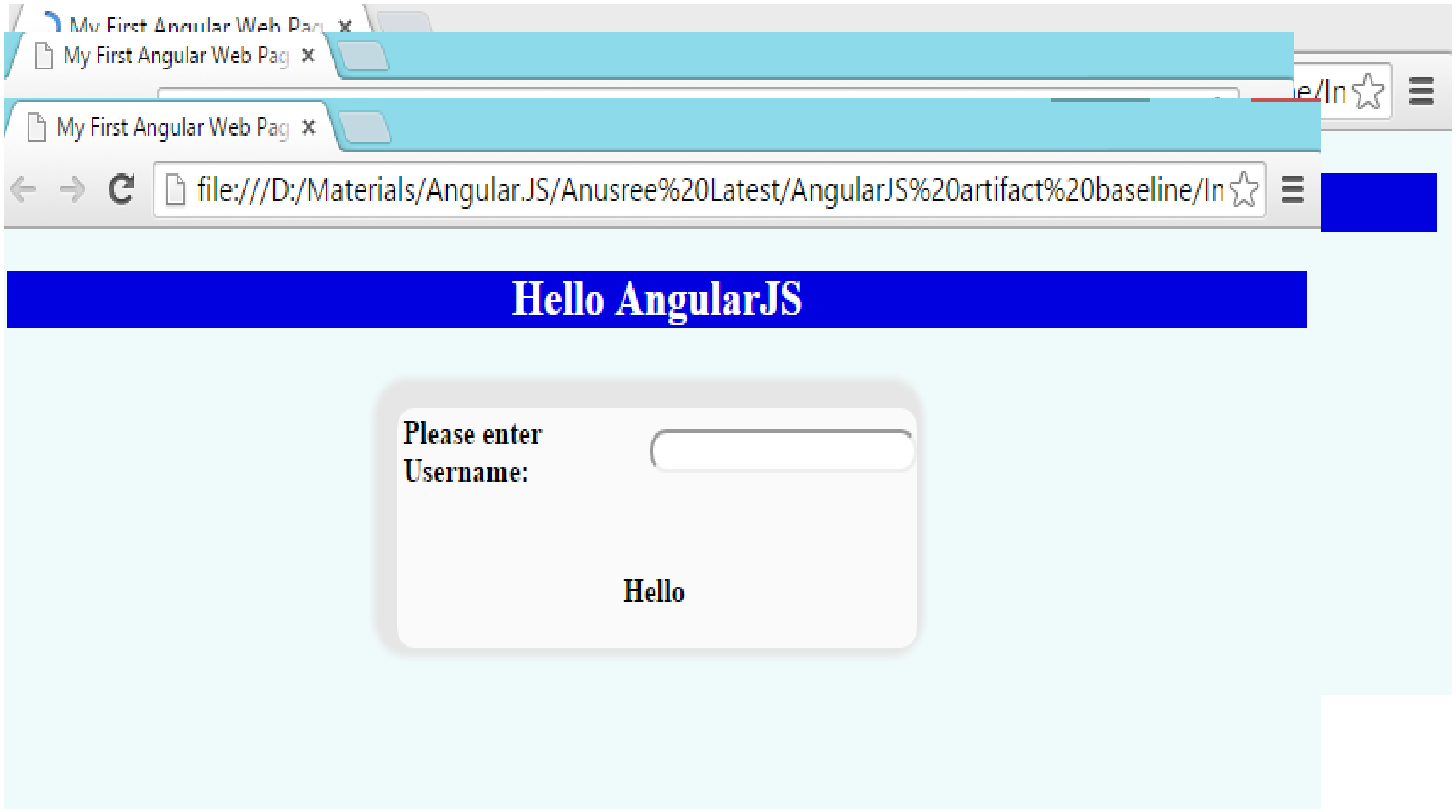
```
<!doctype html>

<html>

<head>

  <title>My First Angular Web Page</title>

   <style type="text/css">

            @import "styles/style.css";

       </style>

  <script src="lib/Angular/angular.js"></script>

<script>
angular.element(document).ready(function() {

var ans=confirm("Are you sure you want to load
angular");

if(ans) {

                    angular.bootstrap(document);

 }      });

    </script></head><body>
```

Use JS confirm dialog to get user confirmation

If user confirms, then Angular Bootstrapping happens

```
<center>

<header>

 <h2> {{ 'Hello '+' AngularJS' }} </h2>

</header>

<br>

 <div class="LoginFormDiv">

            <table >

             <tr>

                <td><b>Please enter Username:</b></td>

                <td><input  ng-model="username" ></td>

             </tr>

            </table>  <br><br>

                                <b> Hello {{username}} </b>

                              >  </div>

</center></body></html>
```

The First Angular Web page: Manual Bootstrapping



My First Angular Web Pag ×

My First Angular Web Pag ×

My First Angular Web Pag ×

file:///D:/Materials/Angular.JS/Anusree%20Latest/AngularJS%20artifact%20baseline/In

Hello AngularJS

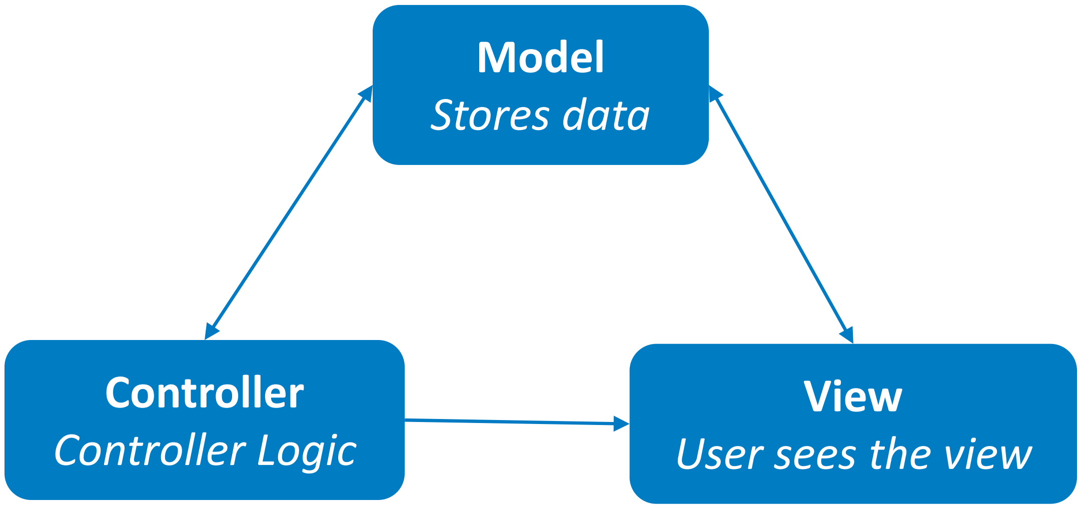
Please enter Username:

Hello

Angular Building Blocks

AngularJS MVC

- AngularJS is MVC which means we have dedicated model, view and controller defined.
- MVC structure of AngularJS enables creation of better maintainable code.



Model

Model

- Model represents data in the application.
- The identifier for a model can be the name of any valid property.
- No restrictions are associated with the Angular model, i.e. need not inherit from any class, does not have special getter or setter methods to get/set its values.
- Value that can be associated with a model can be any primitive value or any JavaScript objects like arrays.
- Model notifies the view and the controller on change of data.
- Model is merged with the template to produce the view.

Model- Creation

- Models can be created by using the “**ng-model**” attribute which makes **two-way data binding** possible.
  - Form input, select, textarea and form controls:
    - Eg:
      ```
      <input ng-model=“company" value=“infosys">
      ```
    - The above code creates a model called ‘company’ and sets the value ‘Infosys’ to the same.
  - We can use an Angular expression with assignment operator.
    - Eg:
      ```
      <button ng-click="{{prop='bar'}}">Click me</button>
      ```
  - We can also use ngInit directive to create models explicitly.
    - Eg:
      ```
      <body ng-init=" prop = 'bar' ">
      ```

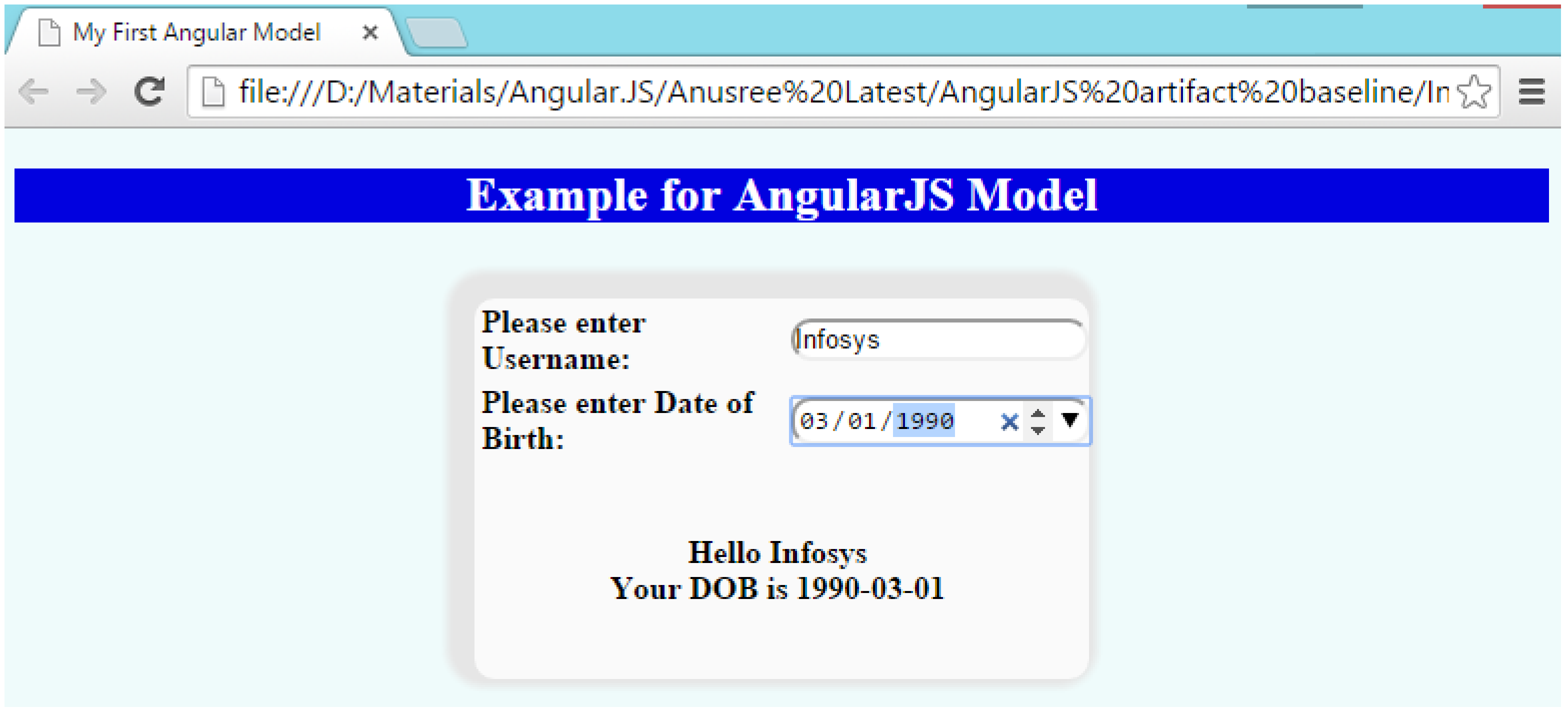
Model creation

```
<!doctype html>
<html ng-app>
<head>
  <title>My First Angular Web Page</title>
   <style type="text/css">
          @import "styles/style.css";
       </style>
<
<
<
<header>
 <h2>
 {{ 'Example for '+' AngularJS Model' }} </h2>
</header> <br>
```

Here Username and Date of Birth are set as models.
Any changes made to the input will be automatically reflected in the view as well

```
<div class="LoginFormDiv">
<table >
<tr>
<td><b>Please enter Username:</b></td>
<td><input type="text" ng-model="username" ></td>
</tr>
<tr>
<td><b>Please enter Date of Birth:</b></td>
<td><input  type="date" ng-model="dob" ></td>
</tr>       </table>  <br><br>
<b> Hello {{username}} </b><br>
<b> Your DOB is {{dob}} </b>  <br><br><br>
  </div></center>
</body></html>
```

The First Angular Model


My First Angular Model
file:///D:/Materials/Angular.JS/Anusree%20Latest/AngularJS%20artifact%20baseline/In

Example for AngularJS Model

Please enter Username: Infosys

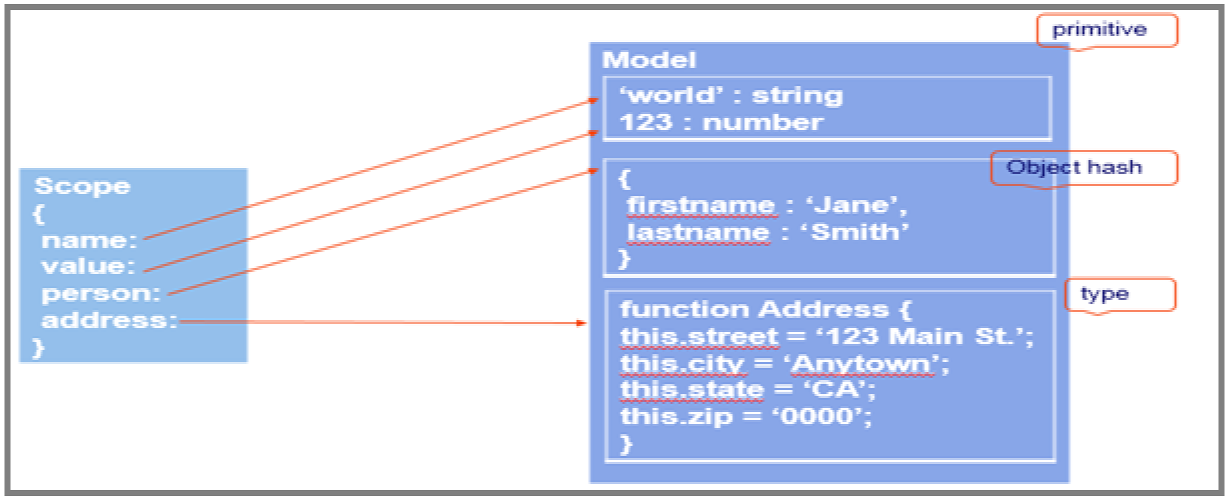
Please enter Date of Birth: 03/01/1990

Hello Infosys
Your DOB is 1990-03-01

Model

- To make a JavaScript object a model in Angular, we need to ensure that the object can be referenced as a scope property by the scope object of Angular. Property reference could be created explicitly or implicitly.

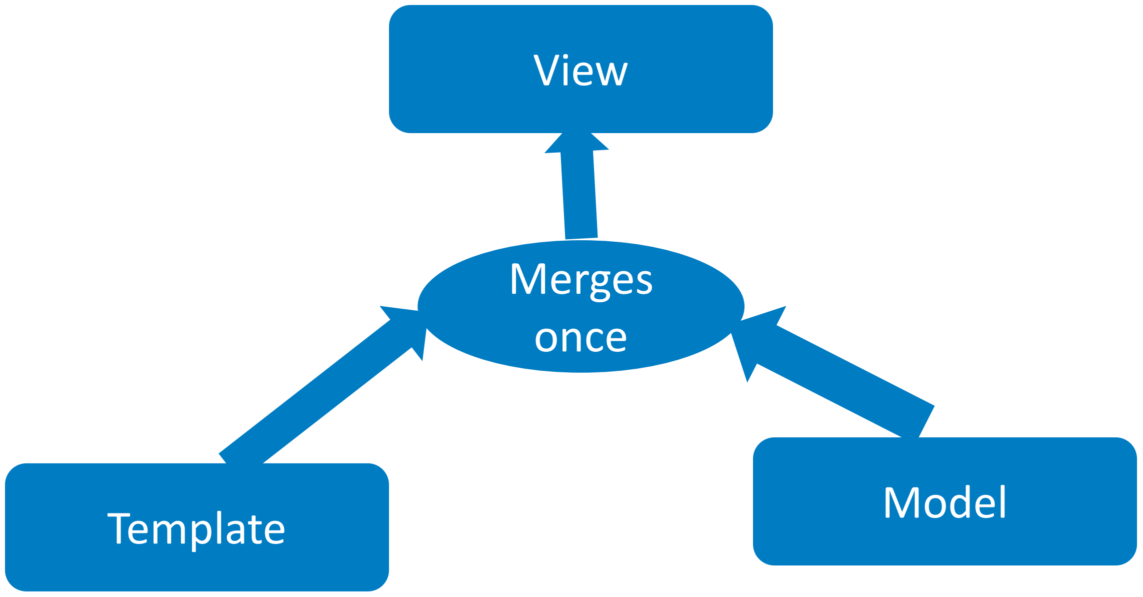
- We should be able to refer the model from the scope so that it can rendered in the view.



View

Views

- View:
  - Views constitute the presentation layer of the application.
  - View is what the user sees on the screen.
  - In traditional views, the data given by the user is combined with the static template text and the element's innerHTML is then updated with the result.



Views

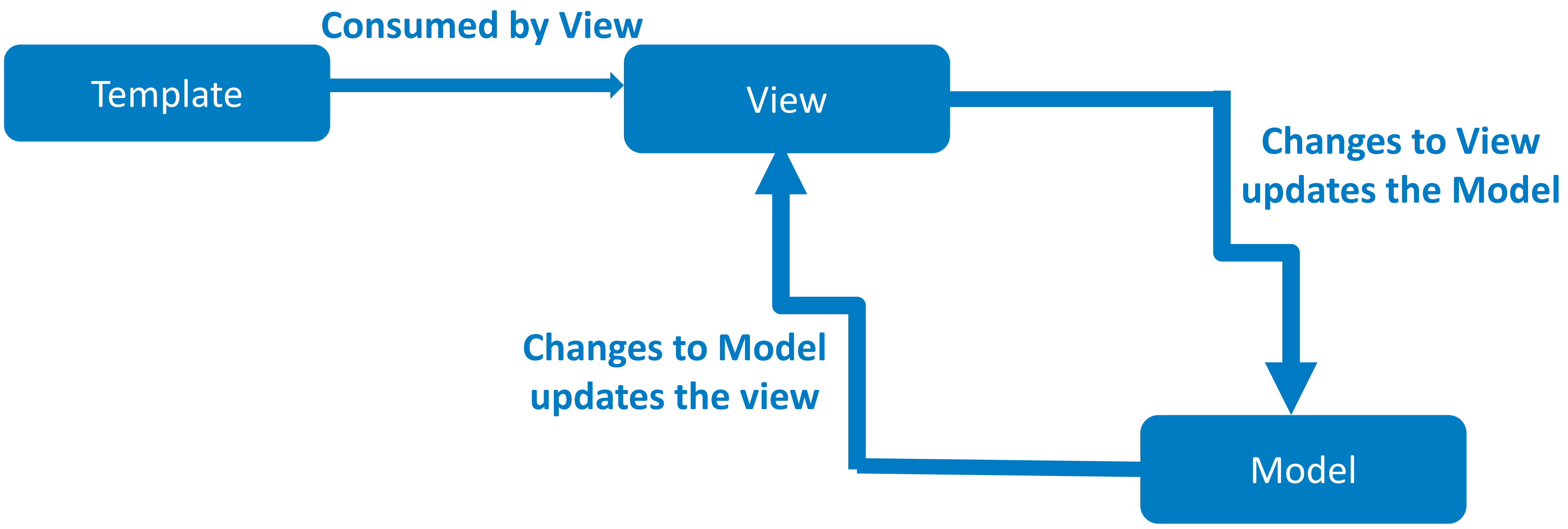
- Disadvantage of the traditional approach:
  - We need to read user input and merging with the template.
  - After merging, update the DOM.
  - User input can be clobbered because of over-writing.

Understanding Views- the Angular way

- Angular View is initially a template, then merged with the model (for data to be shown) and then rendered into the browser's DOM.
- The template is written in html sprinkled with directives like: ng-app, ng-init, ng-repeat, etc
- Angular compiler consumes the DOM with the directives and not the string templates.
- This results in a link function where the result is combined with the scope model and a live view is generated.
- On creating the view, we provide information regarding the model also.
- Views display the current state of the model because of two-way data binding
- No code needs to be written explicitly for updating the view.

Two way data binding

- Angular follows two-way data binding approach.

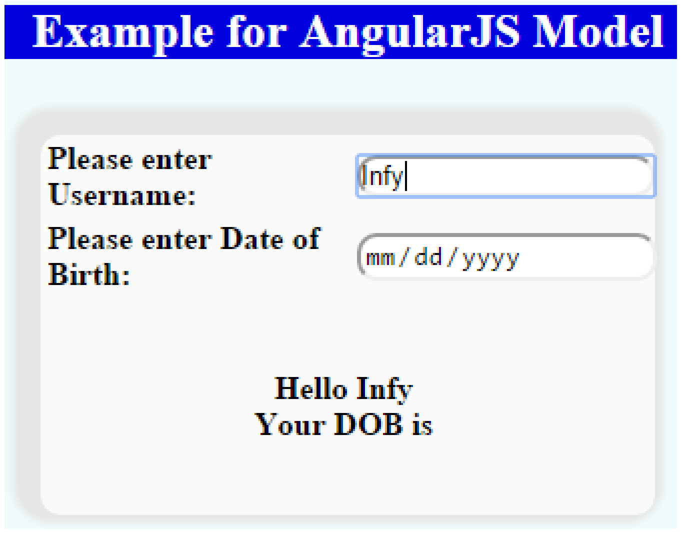

Consumed by View
Template
View
Changes to View updates the Model
Changes to Model updates the view
Model

Two way data binding

- View is updated automatically the moment the data in the model changes.
- Model is updated automatically when a value in the view is updated.
- This avoids active manipulation of the DOM and supports bootstrapping and rapid development of web applications.
- $scope service in Angular detects changes to the model and modifies HTML in the view accordingly.
- Any modifications in the view are updated in the model via controller in the same way.

Two way data binding

```
<!doctype html>
<html ng-app>
<!--- initial code goes here →
<table >
<tr>
<td><b>Please enter Username:</b></td>
<td><input type="text" ng-model="username" ></td>
</tr>
<tr>
<td><b>Please enter Date of Birth:</b></td>
<td><input  type="date" ng-model="dob" ></td>
</tr>        </table>  <br><br>
<b> Hello {{username}} </b><br>
<b> Your DOB is {{dob}} </b>
<!--- rest of the static code goes here →
```

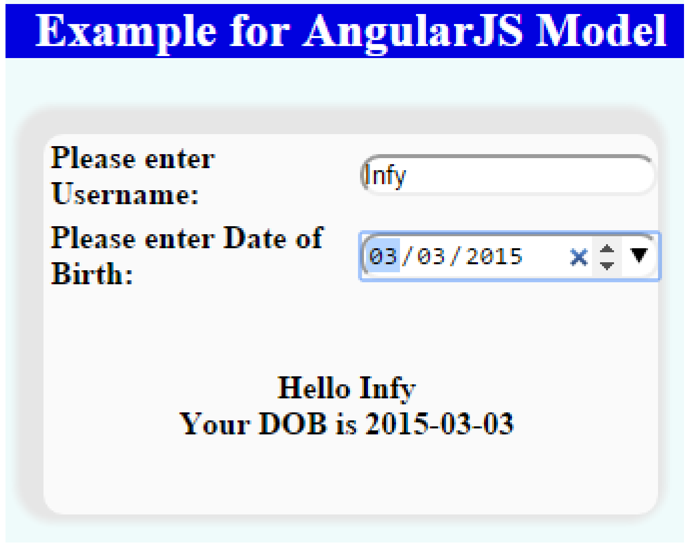


Example for AngularJS Model

Please enter Username: Infy

Please enter Date of Birth: mm/dd/yyyy

Hello Infy
Your DOB is



Example for AngularJS Model

Please enter Username: Infy

Please enter Date of Birth: 03/03/2015

Hello Infy
Your DOB is 2015-03-03

Controller

Controller

- The controller contains the code which sets the behavior of the app
- It constructs the model and publishes it to the view, along with some callback methods
- It contains the **$scope** as argument using which it accesses the model
- The **$scope** variable is specific to where the controller is included in the webpage.
- The **$scope** is responsible for binding the models between the view(Html) and the controller
- The controller is bound to the element you attach it to and it's children

Controller

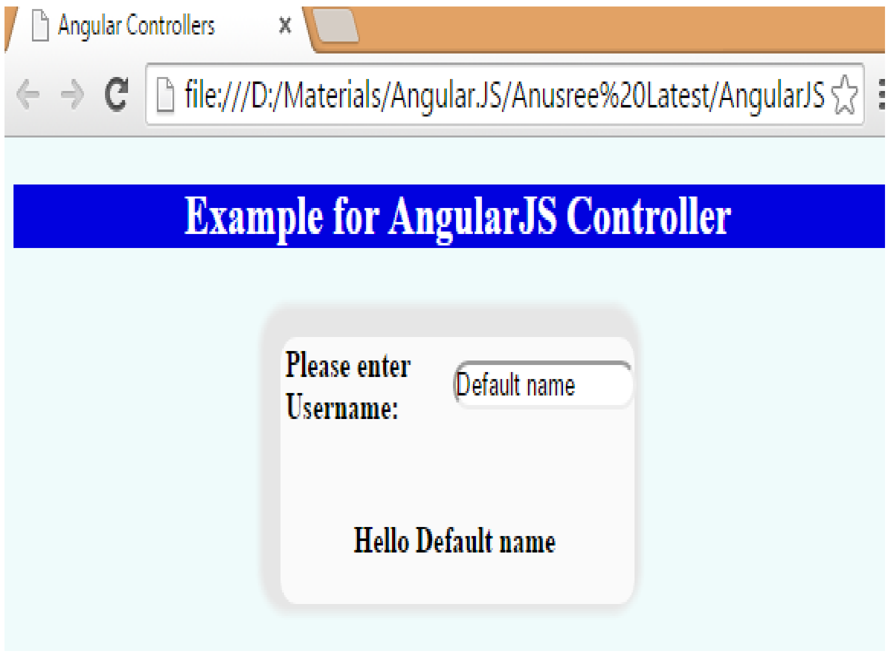
- Controllers can be added in three ways:
  1. Declaring a global function
  2. Declaring controller on entire app
  3. Declaring controller in a module

Declaring Controller as a global function

```
<!doctype html>
<html ng-app>
<head>
  <title>Angular Controller</title>
   <style type="text/css">
          @import "styles/style.css";
       </style>
  <script src="lib/Angular/angular.js"></script>
  <script type="text/javascript">
function MainController ($scope)
{   $scope.username="Default name";  }
  </script></head>
<body>
<center>
```

```
<header>
 <h2> {{ 'Example for '+' AngularJS Controller' }} </h2>
</header>
<br>
 <div class="LoginFormDiv" ng-controller="MainController">
<table >
<tr>
<td><b>Please enter Username:</b></td>
<td><input type="text" ng-model="username" ></td>
</tr>
</table>  <br><br>
<b> Hello {{username}} </b><br><br>
   </div>
</center></body></html>
```

Declaring Controller as a global function

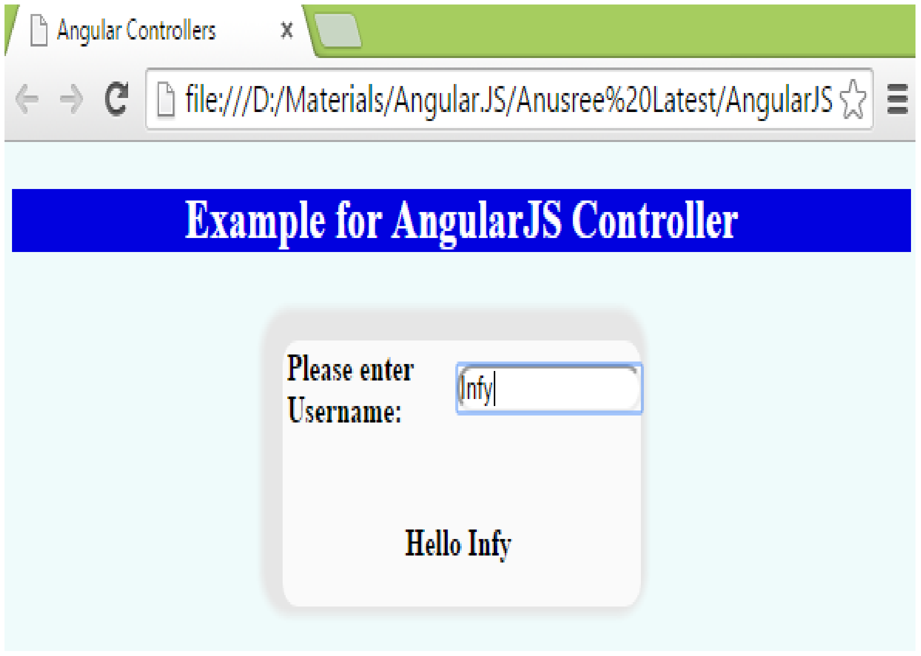

Angular Controllers

file:///D:/Materials/Angular.JS/Anusree%20Latest/AngularJS

Example for AngularJS Controller

Please enter Username: Default name

Hello Default name


Angular Controllers

file:///D:/Materials/Angular.JS/Anusree%20Latest/AngularJS

Example for AngularJS Controller

Please enter Username: Infy

Hello Infy

Declaring Controller on entire app

- Angular code can be written more efficiently by creating modules.

```
var myModule = angular.module(‘myModule’,[])
```

- The module can now be set as the app module:

```
<html ng-app = 'myModule' >
```

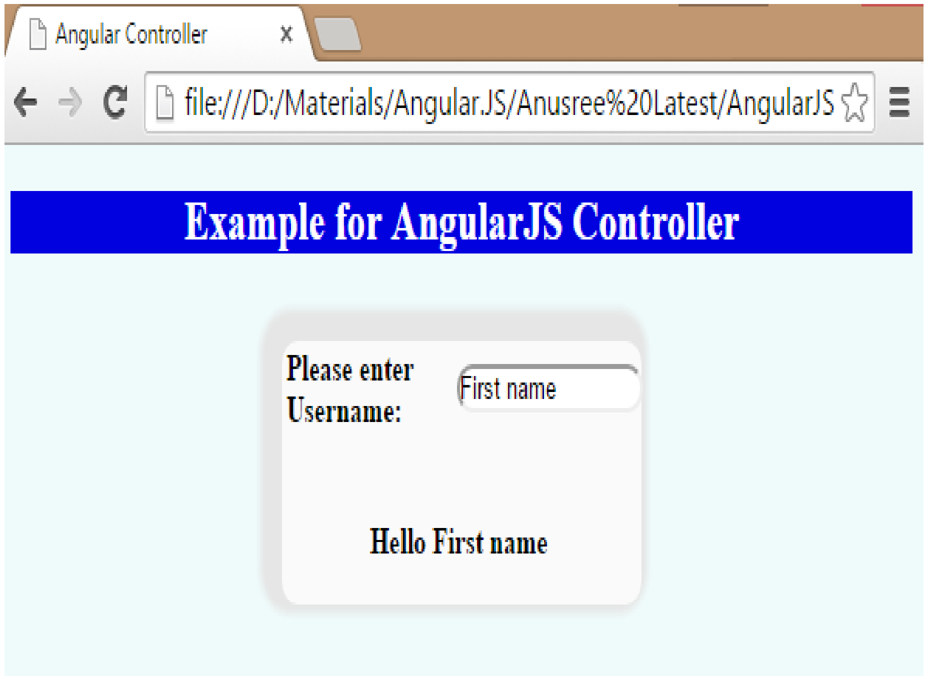
- This approach prevents pollution of global namespace.
- The module ‘myModule’ will be loaded when Angular app loads and hence the components associated with the main module will also be loaded.
- Controllers can also be added to the main module.

Declaring Controller on entire app

```
<!doctype html>

<html ng-app="myModule">

<head>

  <title>Angular Controller</title>

   <style type="text/css">

            @import "styles/style.css";

        </style>

  <script src="lib/Angular/angular.js"></script>

  <script type="text/javascript">

var myModule=angular.module('myModule',[])

myModule.controller('MainController',
function($scope) {
                  $scope.username="First name";

                  });

  </script></head><body><center>
```

```
<header>

 <h2> {{ 'Example for '+' AngularJS Controller' }} </h2>

</header>

<br>

 <div class="LoginFormDiv" ng-controller="MainController">

<table >

<tr>

<td><b>Please enter Username:</b></td>

<td><input type="text" ng-model="username" ></td>

</tr>

</table>  <br><br>

<b> Hello {{username}} </b><br><br>

   </div>

</center>

</body></html>
```

Declaring Controller on entire app

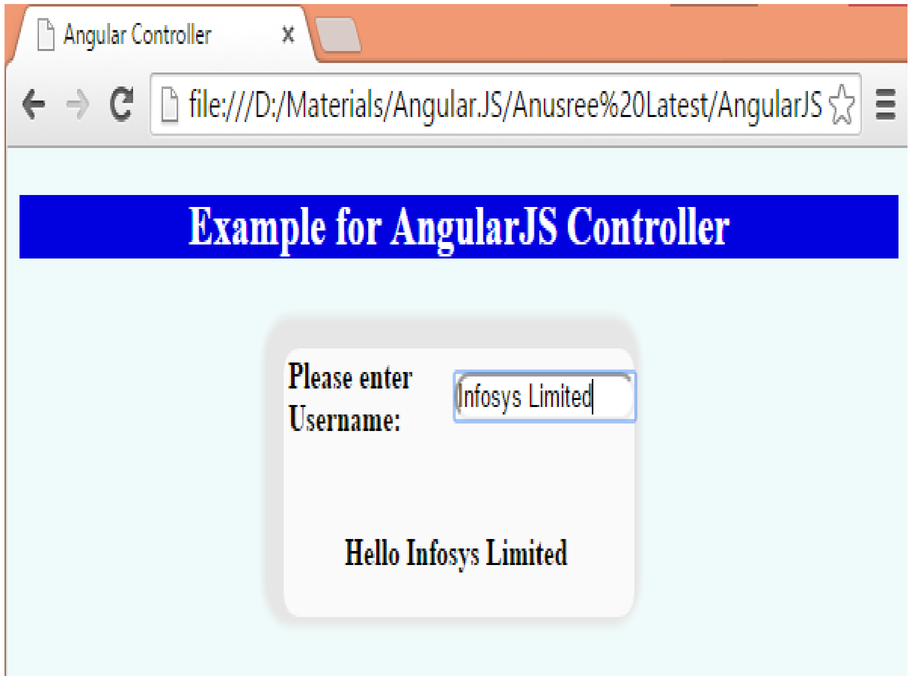

Angular Controller

file:///D:/Materials/Angular.JS/Anusree%20Latest/AngularJS

Example for AngularJS Controller

Please enter Username: First name

Hello First name


Angular Controller

file:///D:/Materials/Angular.JS/Anusree%20Latest/AngularJS

Example for AngularJS Controller

Please enter Username: Infosys Limited

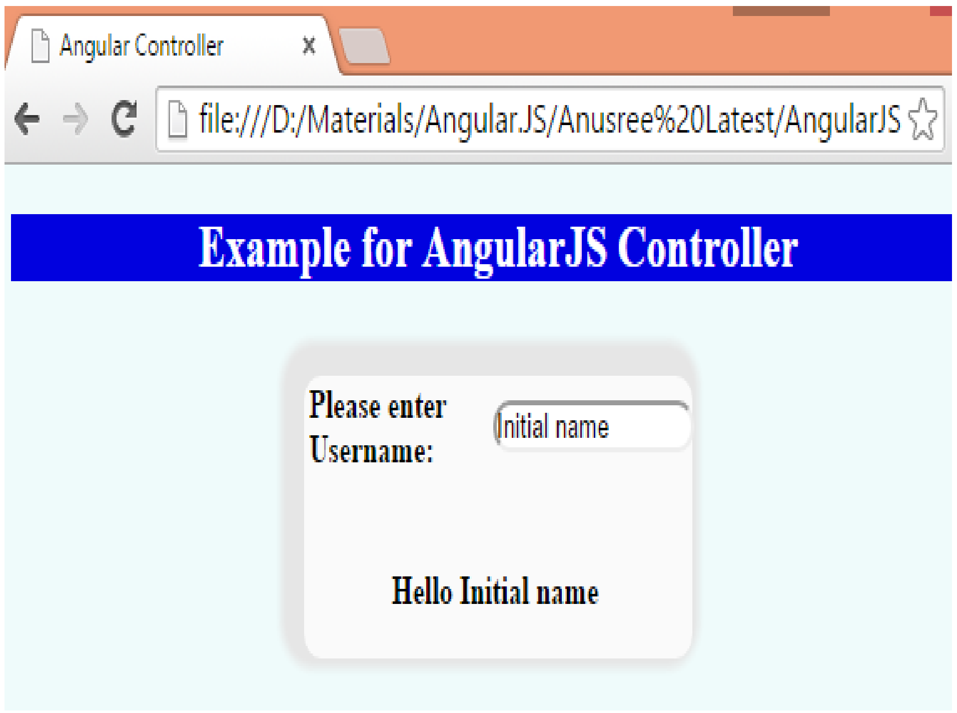
Hello Infosys Limited

Declaring Controller in a module

```
<!doctype html>

<html ng-app="myModule">

<head>

  <title>Angular Controller</title>

   <style type="text/css">

            @import "styles/style.css";

        </style>

  <script src="lib/Angular/angular.js"></script>

  <script type="text/javascript">

var myModule=angular.module('myModule',['controllers']);

var controllers=angular.module('controllers',[]);

controllers.controller('MainController', function($scope) {

                  $scope.username="Initial name";

                  });

  </script></head><body><center>
```

```
<header>

 <h2> {{ 'Example for '+' AngularJS Controller' }}
</h2>

</header>

<br>

 <div class="LoginFormDiv“

 ng-controller="MainController">

<table >

<tr>

<td><b>Please enter Username:</b></td>

<td><input type="text" ng-model="username" >
</td></tr>

</table>  <br><br>

<b> Hello {{username}} </b><br><br>

   </div></center></body></html>
```

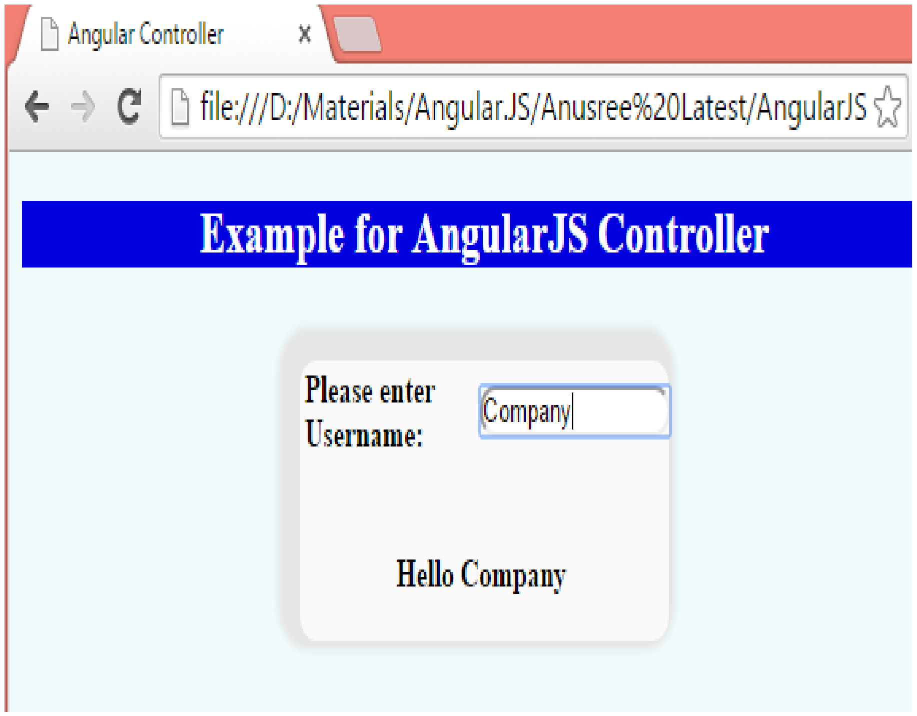
Declaring Controller in a module



Example for AngularJS Controller

Please enter Username: Initial name

Hello Initial name



Example for AngularJS Controller

Please enter Username: Company

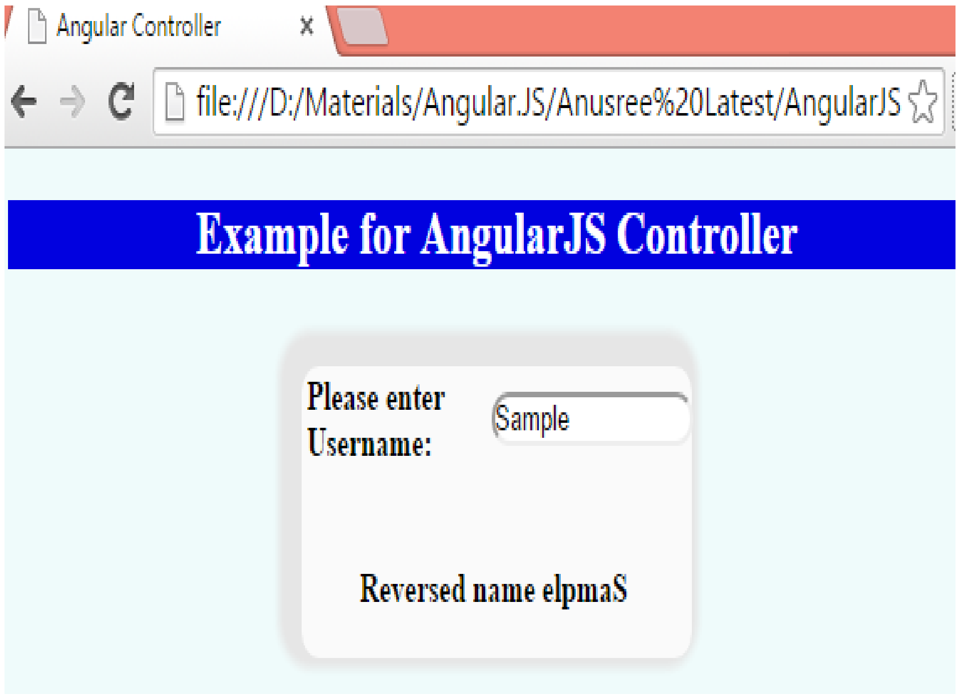
Hello Company

Adding Custom Functions inside controllers

```
<!doctype html>

<html ng-app="myModule">

<head>

  <title>Angular Controller</title>

   <style type="text/css">  @import "styles/style.css"; </style>

  <script src="lib/Angular/angular.js"></script>

  <script type="text/javascript">

var myModule=angular.module('myModule',['controllers']);

var controllers=angular.module('controllers',[]);

controllers.controller('MainController', function($scope) {

$scope.username="Sample";

 $scope.reverseUsername=function (user) {

                  return user.split("").reverse().join("");}

                 });

</script></head><body>
```

```
<center><header>

 <h2> {{ 'Example for '+' AngularJS Controller' }}

</h2></header><br>

 <div class="LoginFormDiv" ng-
controller="MainController">

<table >

<tr>

<td><b>Please enter Username:</b></td>

<td><input type="text" ng-model="username" >

</td></tr> </table>  <br><br> <b>

Reversed name {{ reverseUsername(username)}}
</b>       <br><br>

   </div>

</center>

</body></html>
```

Adding Custom Functions inside controllers



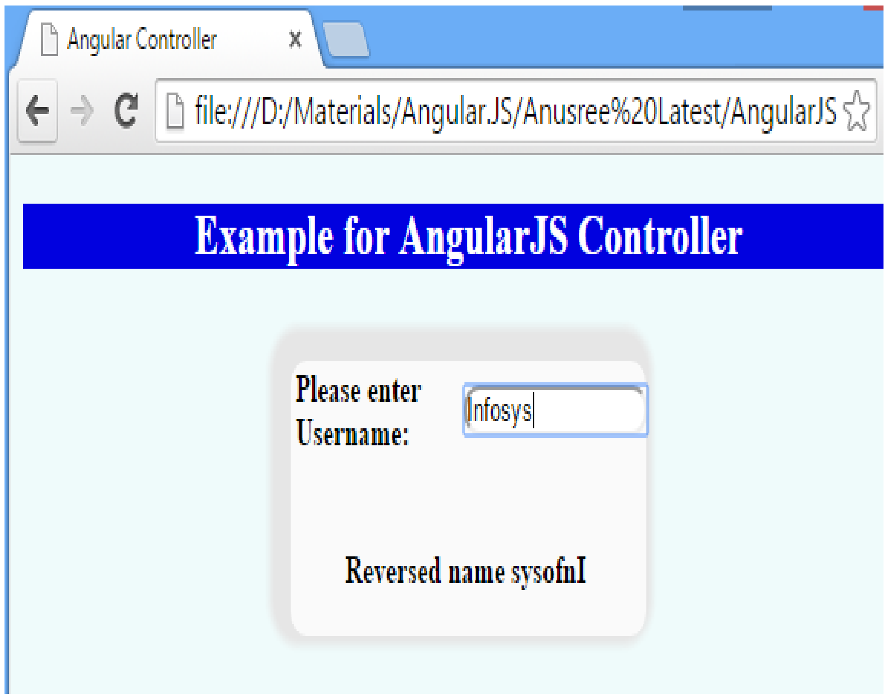
Angular Controller

file:///D:/Materials/Angular.JS/Anusree%20Latest/AngularJS

Example for AngularJS Controller

Please enter Username: Sample

Reversed name elpmaS



Angular Controller

file:///D:/Materials/Angular.JS/Anusree%20Latest/AngularJS

Example for AngularJS Controller

Please enter Username: Infosys

Reversed name sysofnI

Case Study – Shopping List

Shopping List-AngularMVC.html

- Lets see a sample app which makes use of all concepts learnt till now :

```
<!doctype html>

<html ng-app>

<head>

<!—code for including   gular library,
controllers_casestudy  ile and
style_casestu

</he

<bo

<center><h
Shopping</h2></

<div ng-controll     mCtrl">

<h4>Please c   k the items as and when you
finish buyi

<span
left</spa
```

```
<table>

<th>Status</th><th>Item</th>

<tr ng-repeat="item in items">

<td> <input type="checkbox"  ng-model="item.done"> </td>
<td> {{item.text}} </td>

</tr>

able> <br

<div class="ad

<form ng-submit="addItem()

<input type="text" ng-m   l="itemText" class="txt"
placeholder="Add new it        checklist"><br><br>

 button  type="sub                                    to
checklist</button

</form></div></div
```

Controller for the div

remaining() is a function defined in the controller and items is a model

The text input is set as a model with name itemText

A new row is created for every item in items array

The input is set as a model with name item.done. Hence whenever checkbox input changes, item.done value changes

ng-submit stops the default action of the form and executes the addItem() function

Shopping List : controllers_casestudy.js

```
function ItemCtrl($scope) {
 $scope.items = [
   {text:'Wheat Flour', done:true},
   {text:'Toothpaste', done:false}
];
 $scope.remaining = function() {
   var count = 0;
                angular.forEach($scope.items, function(item) {
               if(item.done==false)
                                  count=count+1;
               });   return count;
  };
$scope.addItem = function() {
   $scope.items.push({text:$scope.itemText, done:false});
   $scope.itemText = '';    };
```

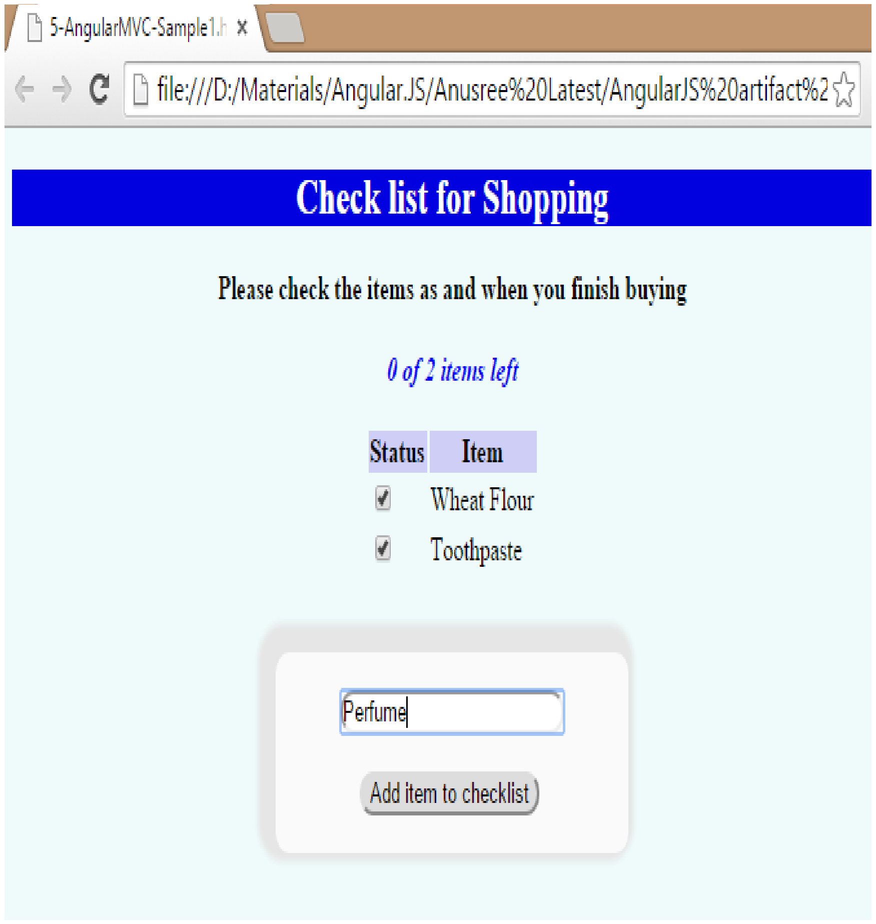
Shopping List : styles_casestudy.css

```
body
{
background-color:#EFFBFB;
}
header
{
background-color:#0101DF;
 color: white;
}
span
{
font-style:italic;
color:blue;
font-weight:bolder;
}
```

```
.button
{
border-radius:10px;
}
.txt
{
border-radius:10px;
}
th
{
    background-color: #CECEF6;
}
```

```
.add
{
    Border-radius: 10px;
    background-color: #FAFAFA;
    width:40%;
    box-shadow: -1px -1px 2px 15px
#FBFBEF;
    -webkit-box-shadow:-5px -5px 5px
9px #E6E6E6;
    -moz-box-shadow:0px 0px 3px 3px
#FBFBEF;
}
```

Shopping List


5-AngularMVC-Sample1.h
file:///D:/Materials/AngularJS/Anusree%20Latest/AngularJS%20artifact%2

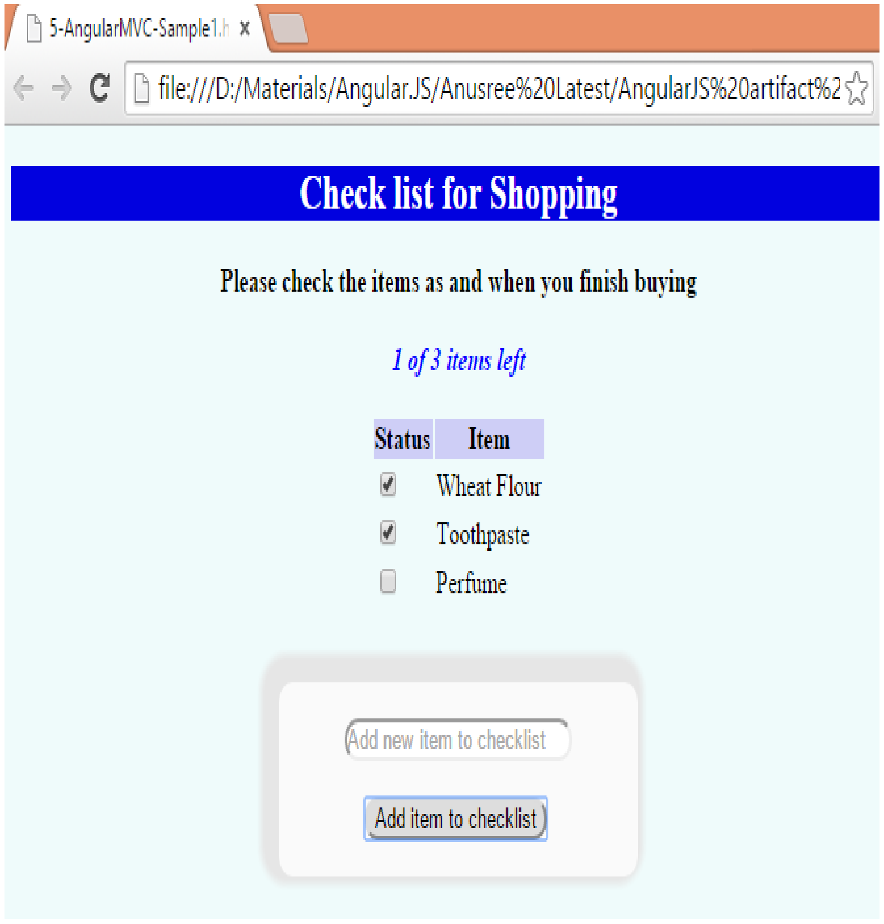
Check list for Shopping

Please check the items as and when you finish buying

0 of 2 items left

Status	Item
✓	Wheat Flour
✓	Toothpaste

Perfume
Add item to checklist


5-AngularMVC-Sample1.h
file:///D:/Materials/AngularJS/Anusree%20Latest/AngularJS%20artifact%2

Check list for Shopping

Please check the items as and when you finish buying

1 of 3 items left

Status	Item
✓	Wheat Flour
✓	Toothpaste
☐	Perfume

Add new item to checklist
Add item to checklist

Assignment

Assignment: Library Management System

We will progressively build an Angular application on ‘Library Management System’ as part of the assignment from Day1 to Day5.

Problem Statement:

We have two roles, Student and Librarian.

Student Login:

Student can selectively view all books or view only available books. Student can search for a book in the book list displayed. Student can also sort the book cost-wise or topic-wise.

Librarian Login:

Librarian can view all the books all times. Librarian gets three extra options:

He can issue an available book.

He can return an issued book.

He can also add a new book to the book list.

Assignment: Library Management System

Half Cooked code is shared with participants.

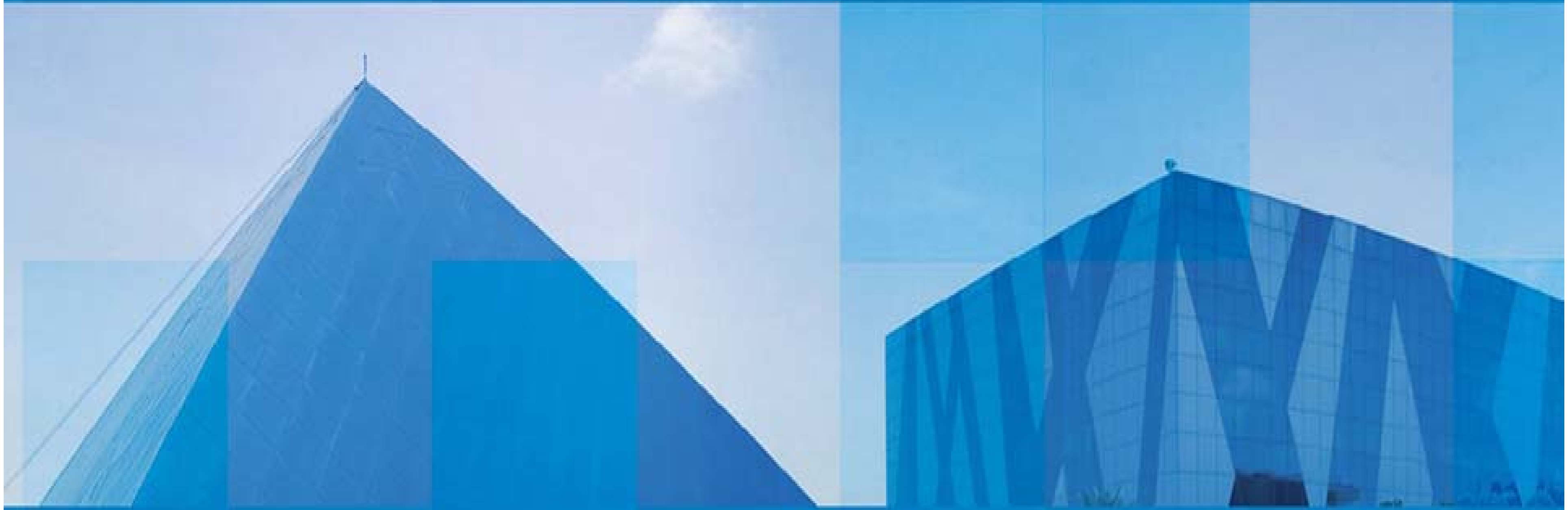
Participants need to modify the files according to everyday requirement and progressively build the application.

Screenshots are attached in the Assignment document.

Summary

- Introduction to AngularJS
- Application Structure
- Bootstrapping Angular
- Angular Building Blocks
  - Model
  - View
    - Understanding Views the Angular way
    - Two-way data binding
  - Controllers

Thank You



Infosys® | Building Tomorrow's Enterprise