Module 3

Web Applications

Web Applications

- Web applications
- Web Application Frameworks
- MVC framework
- Angular JS
- Single Page Applications
- Responsive Web Design

Web Applications

- A web application is a Client-Server computer program that utilizes web browsers and web technology to perform tasks over the Internet.

How web application works?

- User triggers a request to the web server over the Internet, either through a web browser or the application's user interface.
- Web server forwards this request to the appropriate web application server.
- Web application server performs the requested task – such as querying the database or processing the data – then generates the results of the requested data.
- Web application server sends results to the web server with the requested information or processed data.
- Web server responds back to the client with the requested information that then appears on the user's display

EXAMPLE OF A WEB APPLICATION

Web applications include online forms, online retail sales, online auctions, wikis, instant messaging services, shopping carts, word processors, spreadsheets, video and photo editing, file conversion, file scanning, and email programs such as Gmail, Yahoo and AOL. Popular applications include Google Apps and Microsoft 365.

Google Apps for Work has Gmail, Google Docs, Google Sheets, Google Slides, online storage and more. Other functionalities include online sharing of documents and calendars. This lets all team members access the same version of a document simultaneously.

Benefits of Web Application

- Web applications run on multiple platforms regardless of OS or device as long as the browser is compatible.
- All users access the same version, eliminating any compatibility issues.
- They are not installed on the hard drive, thus eliminating space limitations.
- They reduce software piracy in subscription-based web applications (i.e. SaaS).
- They reduce costs for both the business and end user as there is less support and maintenance required by the business and lower requirements for the end user's computer.

Web Application Frameworks

- A **web framework** (**WF**) or **web application framework** (**WAF**) is a software framework that is designed to support the development of web applications including web services, web resources, and web APIs.
- Web frameworks provide a standard way to build and deploy web applications. Web frameworks aim to automate the overhead associated with common activities performed in web development.
- For example, many web frameworks provide libraries for database access, templating frameworks,and session management, and they often promote code reuse. Although they often target development of dynamic web sites, they are also applicable to static websites.

Types of framework architectures

- **Model–view–controller (MVC)**

Many frameworks follow the MVC architectural pattern to separate the data model with business rules from the user interface. This is generally considered a good practice as it modularizes code, promotes code reuse, and allows multiple interfaces to be applied. In web applications, this permits different views to be presented, such as web pages for humans, and web service interfaces for remote applications.

- **Push-based vs. pull-based**

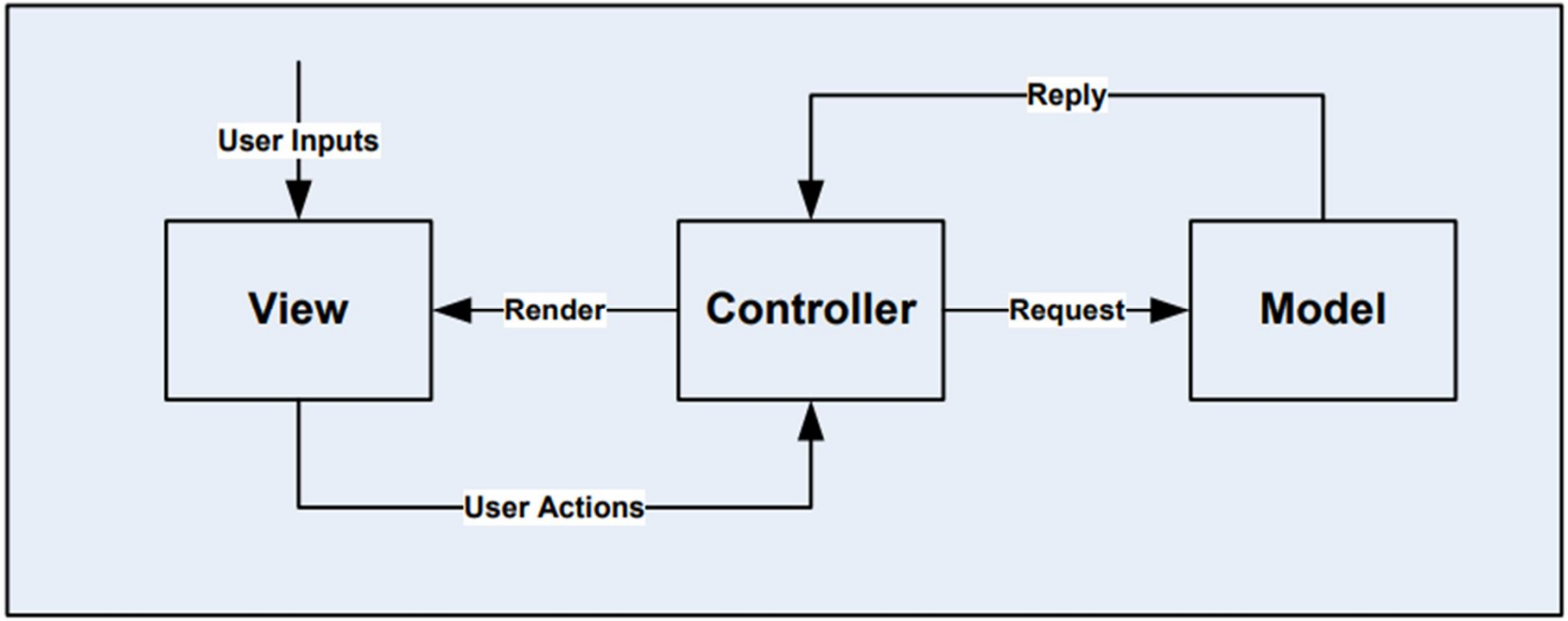
- Most MVC frameworks follow a push-based architecture also called "action-based". These frameworks use actions that do the required processing, and then "push" the data to the view layer to render the results. Django, Ruby on Rails, Symfony, Spring MVC, Stripes, Diamond, CodeIgniter are good examples of this architecture.
- An alternative to this is pull-based architecture, sometimes also called "component-based". These frameworks start with the view layer, which can then "pull" results from multiple controllers as needed. In this architecture, multiple controllers can be involved with a single view. Lift, Tapestry, JBoss Seam, JavaServer Faces, (µ)Micro, and Wicket are examples of pull-based architectures.
- Play, Struts, RIFE, and ZK have support for both push- and pull-based application controller calls.

- **Three-tier organization**

- In three-tier organization, applications are structured around three physical tiers: client, application, and database. The database is normally an RDBMS. The application contains the business logic, running on a server and communicates with the client using HTTP. The client on web applications is a web browser that runs HTML generated by the application layer. The term should not be confused with MVC, where, unlike in three-tier architecture, it is considered a good practice to keep business logic away from the controller, the "middle layer".

Model-View-Controller (MVC)

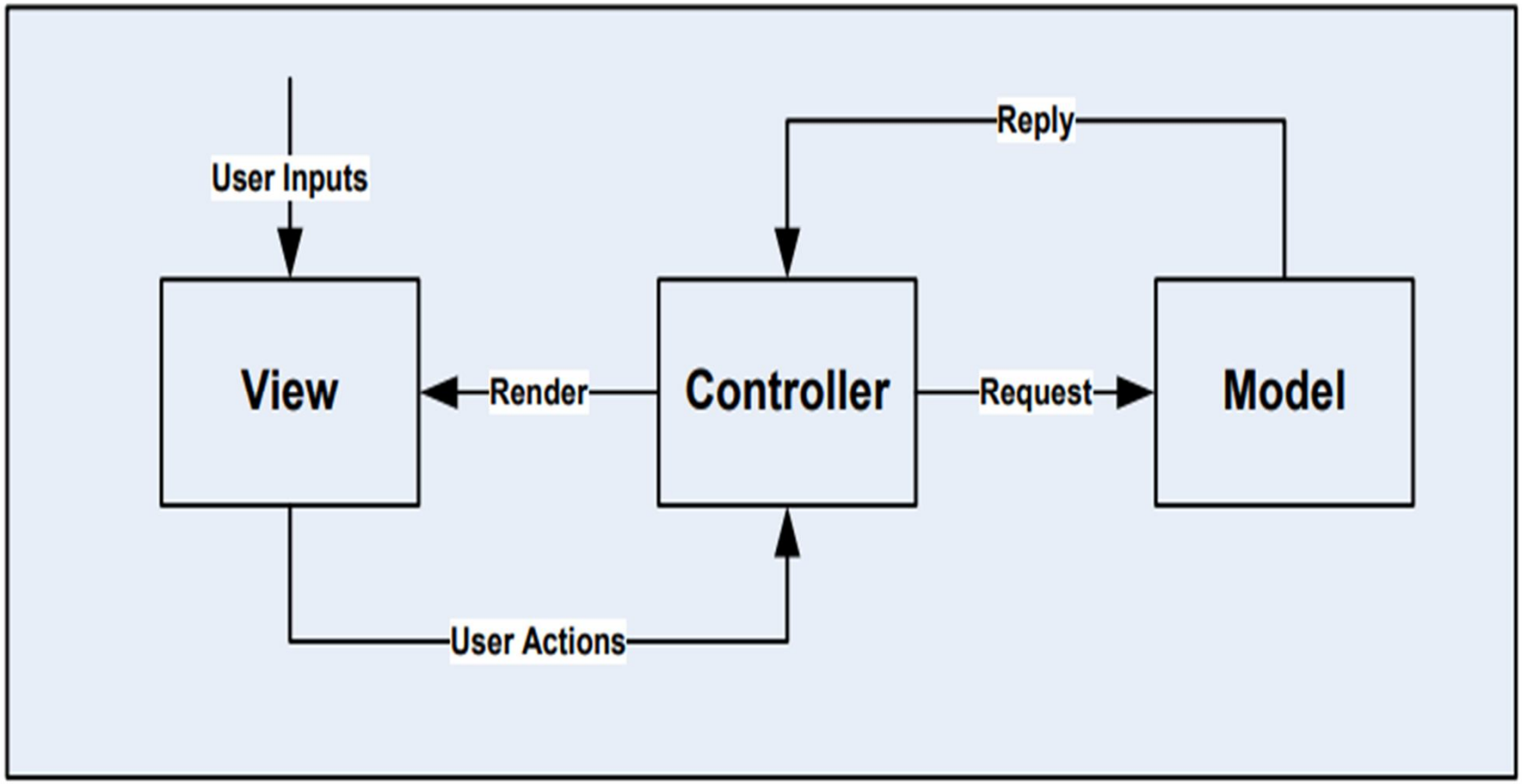
- The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller.
- MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

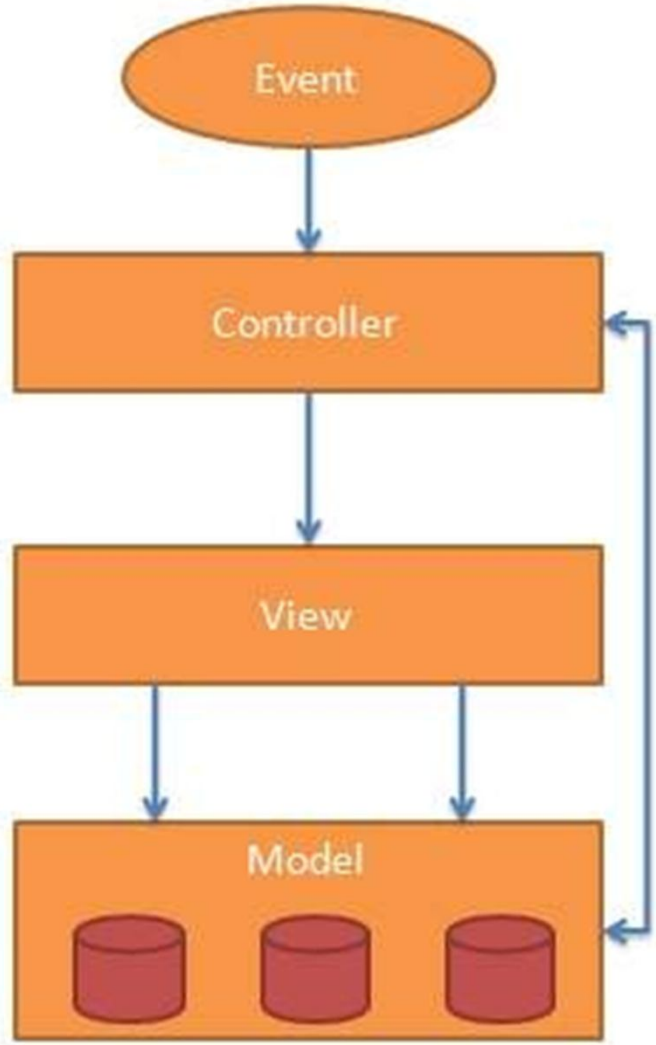


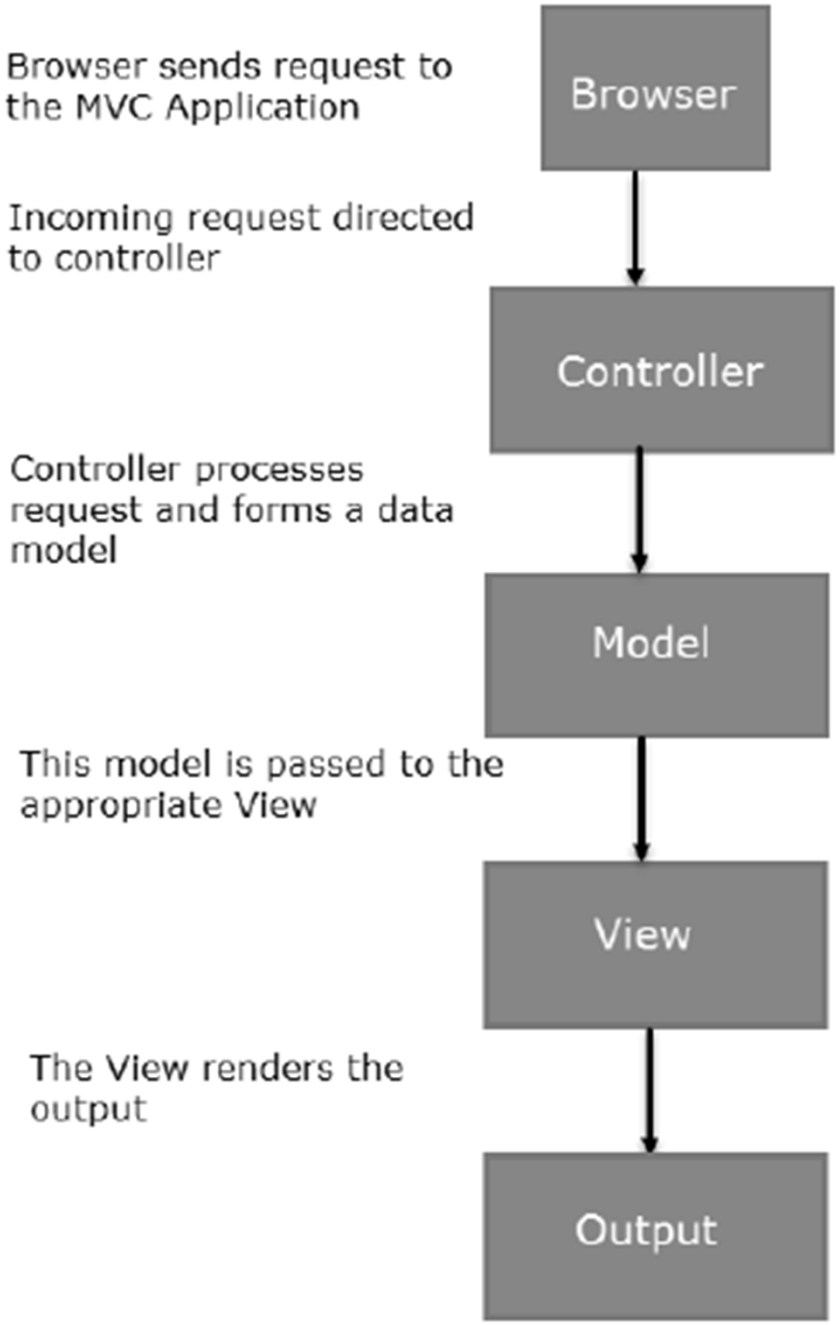
MVC

- MVC is popular because it isolates the application logic from the user interface layer and supports separation of concerns.
- The **controller** receives all requests for the application and then works with the model to prepare any data needed by the **view**.
- The **view** then uses the data prepared by the controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.

MVC


User Inputs
Reply
View
Render
Controller
Request
Model
User Actions


Event
Controller
View
Model


Browser sends request to the MVC Application
Browser
Incoming request directed to controller
Controller
Controller processes request and forms a data model
Model
This model is passed to the appropriate View
View
The View renders the output
Output

- **Model**

Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

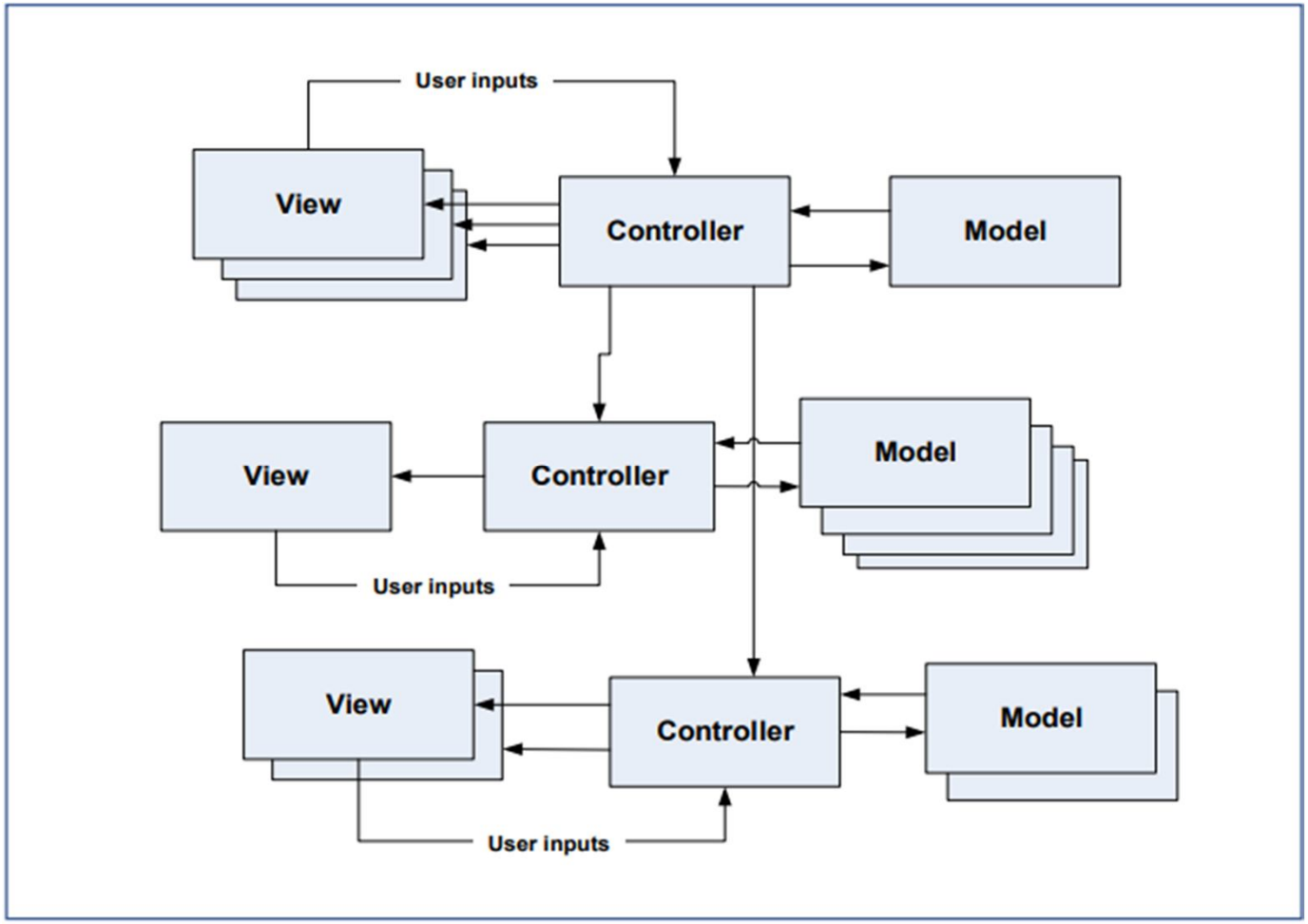
- **View**

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

- **Controller**

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

MVC -Multiple Controllers



Top Web Development Frameworks -2018

S.No	Name	Description
1	AngularJS	• JavaScript framework ,develop your front-end based apps • Google,Angular 4 , MVC architectural pattern • Netflix, Freelancer.com, GoodFIlms
2	Ruby on Rails	• server-side web development framework is built using Ruby • Basecamp, Ask.fm, GitHub, 500px
3	Yii	• Yii is an open-source web application development framework that is built in PHP5. Tass, Craftcms, HumHub
4	Meteor JS	• written in Node.js ,creating simple websites for personal use • developing for iOS, web, Android or desktop
5	Express JS	• developed in Node.js,develop web applications and APIs as fast and easy • Express is the best choice for those people whose goal it is to develop simple web services and APIs rather than web portals or high-loaded calculating backends.

6	Zend	• based on PHP • building more secure, modern, reliable, web services and applications.
7	Django	• written in Python • Model View Template (MVC) architecture
8	Laravel	• PHP development frameworks • develop the web and mobile applications for small websites and big businesses • MVC support
9	React.js	• JavaScript library maintained by Facebook • near future called, React Fibre
10	Node.js	• Node.js is not just a framework, it is a complete environment. • lightweight and efficient against real-time applications with a large amount of data running on distributed devices

11	Symfony	• PHP framework • largest open source platforms such as PHPBB, Piwik, and Drupal.
12	ASP.NET	• open-source and has close to 15% market share.
13	CakePHP	• written in PHP • model-controller-view and association data mapping
14	Vue.js	• open-source JavaScript library for developing user interfaces • released in 2013 and now it has 59600 stars on Github • Angular itself has 56300 stars on Github

Angular JS

- AngularJS is an open source Model-View-Controller framework which is similar to the JavaScript framework.
- AngularJS is a JavaScript framework written in JavaScript.
- It can be added to an HTML page with a <script> tag.
- AngularJS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.
- AngularJS is open source, completely free, and used by thousands of developers around the world.
- It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google.
- AngularJs 1.7.2 is the stable version in June 2018, but AngularJs 1.8 is the latest version 2020.

Why to Learn AngularJS?

- It is used in Single Page Application (SPA) projects.
- An efficient framework that can create Rich Internet Applications (RIA).
- Provides developers an options to write client side applications using JavaScript in a clean Model View Controller (MVC) way.
- Applications are cross-browser compliant.
- It automatically handles JavaScript code suitable for each browser.
- Its open source, completely free, and used by thousands of developers around the world.
- It is licensed under the Apache license version 2.0.
- Overall, AngularJS is a framework to build large scale, high-performance, and easyto-maintain web applications.

AngularJS Features

- **Data-binding** – It is the automatic synchronization of data between model and view components.
- **Scope** – These are objects that refer to the model. They act as a glue between controller and view.
- **Controller** – These are JavaScript functions bound to a particular scope.
- **Services** – AngularJS comes with several built-in services such as $http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
- **Filters** – These select a subset of items from an array and returns a new array.
- **Directives** – Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel, etc.

AngularJS Features

- **Templates** – These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using *partials*.
- **Routing** – It is concept of switching views.
- **Model View Whatever** – MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
- **Deep Linking** – Deep linking allows to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
- **Dependency Injection** – AngularJS has a built-in dependency injection subsystem that helps the developer to create, understand, and test the applications easily.

Disadvantages of AngularJS

Though AngularJS comes with lots of plus points but same time we should consider the following points –

- **Not Secure** – Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.
- **Not degradable** – If your application user disables JavaScript then user will just see the basic page and nothing more.

Download Angular JS Library

- Download it from https://angularjs.org/

- AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:

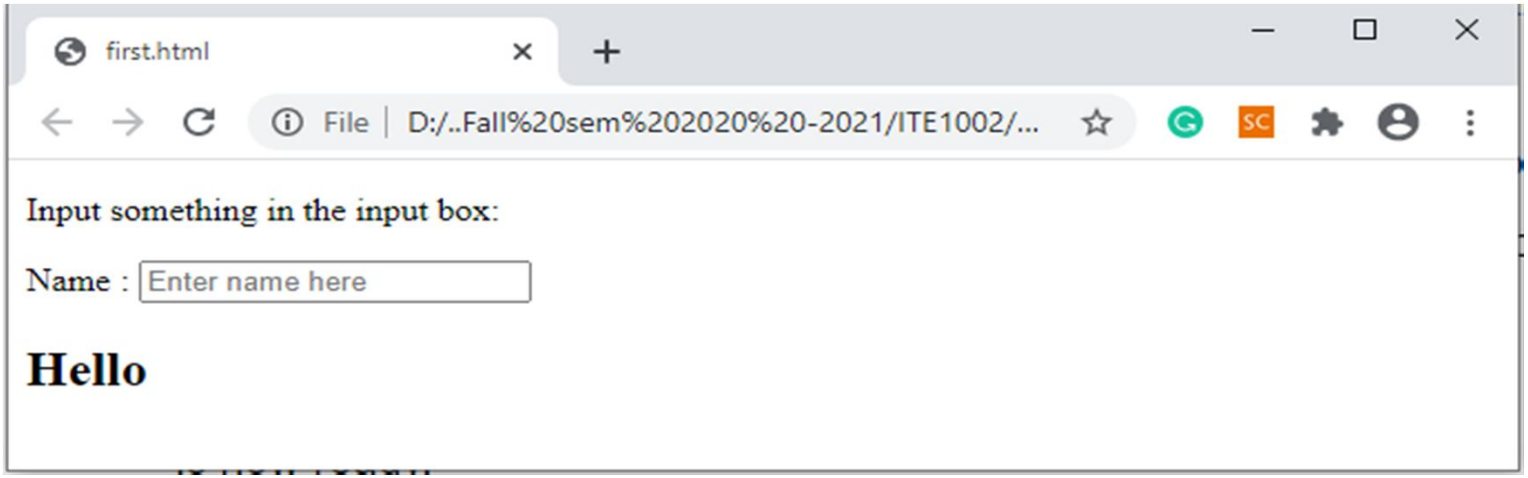
<script src="**https://ajax.googleapis.com/ajax/libs/angularjs/1.8.0/angular.min.js**"></script>

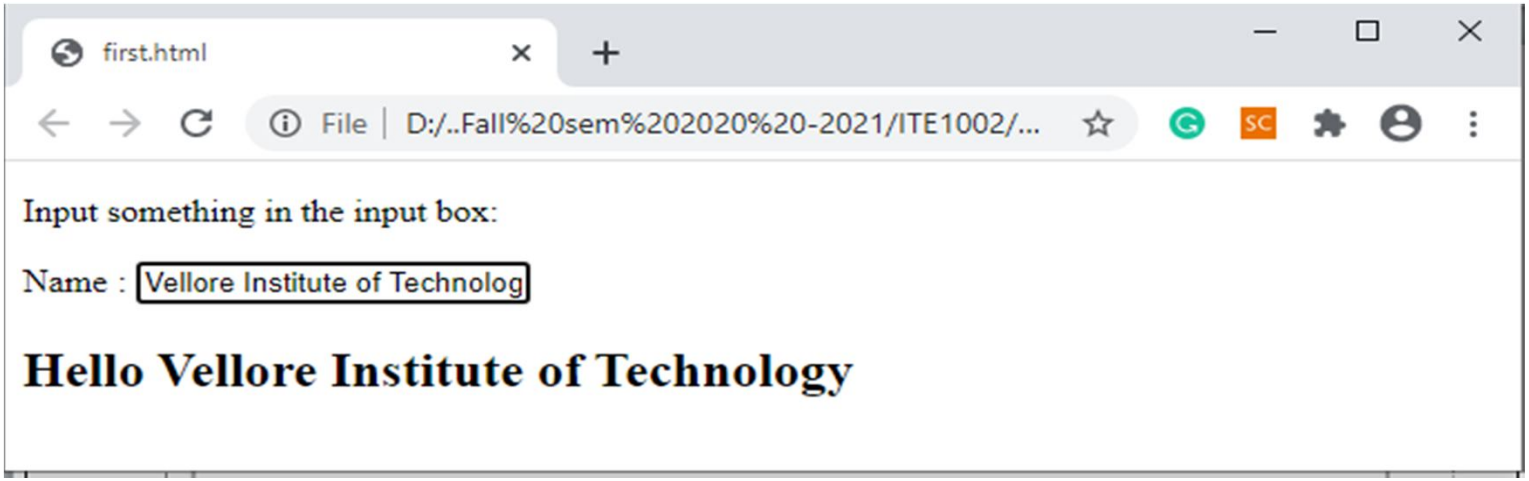
The AngularJS Components

- AngularJS extends HTML with **ng-directives**.
- The AngularJS framework can be divided into following three major parts –
  - **ng-app** – This directive defines and links an AngularJS application to HTML. Or AngularJS that the an HTML container element is the "owner" of an AngularJS **application**.
  - **ng-model** – This directive binds the values of AngularJS application data to HTML input controls. Or binds the value of the input field to the application variable.
  - **ng-bind** – This directive binds the AngularJS Application data to HTML tags. Or binds the content of the HTML element to the application variable.

```
<html><head>
<script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.8.0/angular.min.js"
></script>    </head>
<body><h1>Sample Application</h1><div ng-app = "">
<p>Enter your Name: <input type = "text" ng-model = "name"></p>
<p>Hello <span ng-bind = "name"></span>!</p></div>
</body> </html>
```

- ng-app directive indicates the start of AngularJS application.
- ng-model directive then creates a model variable named "name" which can be used with the html page and within the div having ng-app directive.
- ng-bind then uses the name model to be displayed in the html span tag whenever user input something in the text box.


first.html
File | D:/..Fall%20sem%202020%20-2021/ITE1002/...
Input something in the input box:
Name : Enter name here
Hello


first.html
File | D:/..Fall%20sem%202020%20-2021/ITE1002/...
Input something in the input box:
Name : Vellore Institute of Technolog
Hello Vellore Institute of Technology

AngularJS Expressions can be written inside double braces: {{ expression }} or ng-bind="expression"

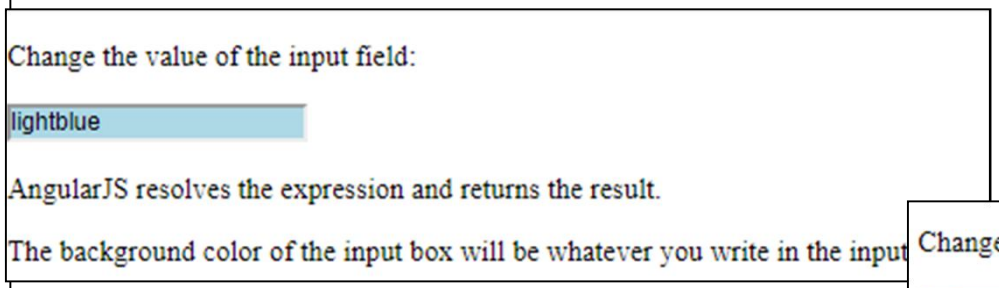

```
<html ng-app><head>
<script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.8.0/angular.min.js"></script>
</head><body><div><label>Name:</label>
<input type = "text" ng-model = "NameText" placeholder =
"Enter a name here">
<hr /><h1>Hello {{NameText}}!</h1>
</div></body></html>
```

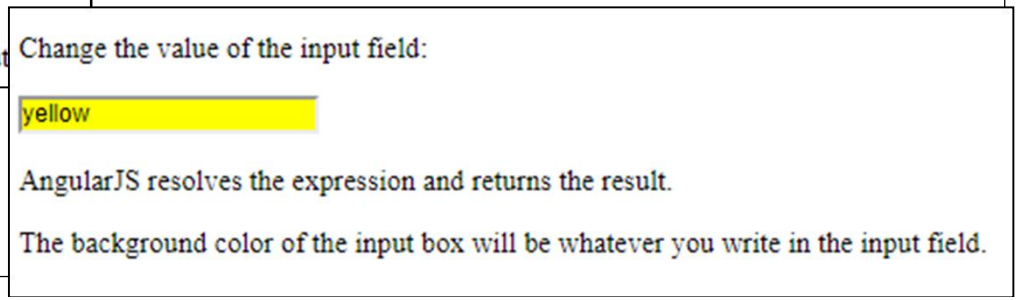
AngularJS Directives

- AngularJS directives are HTML attributes with an **ng** prefix.
- The **ng-init** directive initializes AngularJS application variables.

```
<!DOCTYPE html> <html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.0/angular.min.js"></script>
<body>
<div ng-app="" ng-init="firstName='John'">
<p>The name is <span ng-bind="firstName"></span></p>
</div></body></html>
```

```
<!DOCTYPE html> <html><script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.0/angular.min
.js"></script><body>
<p>Change the value of the input field:</p>
<div ng-app="" ng-init="myCol='lightblue'">
<input style="background-color:{{myCol}}" ng-model="myCol">
</div>
<p>AngularJS resolves the expression and returns the result.</p>
<p>The background color of the input box will be whatever you write in
the input field.</p></body></html>
```





AngularJS Numbers

```
<div ng-app="" ng-init="quantity=1;cost=5">
<p>Total in dollar: {{ quantity * cost }}</p>
<p>Total in dollar: <span ng-bind="quantity * cost">
</span></p></div>
```

Total in dollar: 5

Total in dollar: 5

AngularJS Arrays

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">
<p>The third result is {{ points[2] }}</p></div>
```

The third result is 19

Angular JS Strings

```
<div ng-app=""
ng-init="firstName='John';lastName='Doe'">
<p>The full name is {{ firstName + " " + lastName }}</p>
</div>
```

The full name is: John Doe

Angular JS Objects

```
<div ng-app=""
ng-init="person={firstName:'R',lastName:'Vijayan'}">
<p>The name is {{ person.lastName }}</p></div>
```

AngularJS Expressions vs. JavaScript Expressions

- Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.
- Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.
- AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.
- AngularJS expressions support filters, while JavaScript expressions do not.