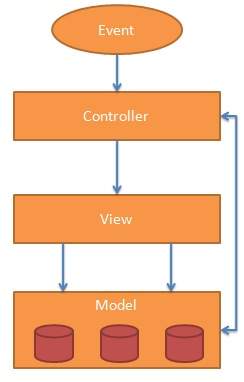
**AngularJS**

1)A single-page application (SPA) is a [web application](https://en.wikipedia.org/wiki/Web_application) or [web site](https://en.wikipedia.org/wiki/Web_site) that fits on a single [web page](https://en.wikipedia.org/wiki/Web_page) with the goal of providing a [user experience](https://en.wikipedia.org/wiki/User_experience) similar to that of a [desktop application](https://en.wikipedia.org/wiki/Desktop_application). In an SPA, either all necessary code – [HTML](https://en.wikipedia.org/wiki/HTML), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), and [CSS](https://en.wikipedia.org/wiki/CSS) – is retrieved with a single page load, or the appropriate resources are [dynamically loaded](https://en.wikipedia.org/wiki/Dynamic_loading) and added to the page as necessary, usually in response to user actions. The page does not reload at any point in the process, nor does control transfer to another page, although the [location hash](https://en.wikipedia.org/wiki/Fragment_identifier) can be used to provide the perception and navigability of separate logical pages in the application, as can the [HTML5](https://en.wikipedia.org/wiki/HTML5) [pushState() API](https://en.wikipedia.org/wiki/Comparison_of_layout_engines_(HTML5)" \l "APIs" \o "Comparison of layout engines (HTML5)). Interaction with the single page application often involves dynamic communication with the [web server](https://en.wikipedia.org/wiki/Web_server).

**2)M**odel **V**iew **C**ontroller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts −

* **Model** − It is the lowest level of the pattern responsible for maintaining data.
* **View** − It is responsible for displaying all or a portion of the data to the user.
* **Controller** − It is a software Code that controls the interactions between the Model and View.

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.



The Model

The model is responsible for managing application data. It responds to the request from view and to the instructions from controller to update itself.

The View

A presentation of data in a particular format, triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.

The Controller

The controller responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

3)An AngularJS application consists of following three important parts −

* **ng-app** − This directive defines and links an AngularJS application to HTML.
* **ng-model** − This directive binds the values of AngularJS application data to HTML input controls.
* **ng-bind** − This directive binds the AngularJS Application data to HTML tags.

4).At a high level, directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's **HTML compiler** ([$compile](https://docs.angularjs.org/api/ng/service/$compile)) to attach a specified behavior to that DOM element (e.g. via event listeners), or even to transform the DOM element and its children.

Angular comes with a set of these directives built-in, like ngBind, ngModel, and ngClass. Much like you create controllers and services, you can create your own directives for Angular to use. When Angular [bootstraps](https://docs.angularjs.org/guide/bootstrap) your application, the [HTML compiler](https://docs.angularjs.org/guide/compiler) traverses the DOM matching directives against the DOM elements.

The ngShow directive shows or hides the given HTML element based on the expression provided to the ngShow attribute. The element is shown or hidden by removing or adding the .ng-hide CSS class onto the element. The .ng-hide CSS class is predefined in AngularJS and sets the display style to none (using an !important flag). For CSP mode please add angular-csp.css to your html file (see [ngCsp](https://docs.angularjs.org/api/ng/directive/ngCsp)).

<!-- when $scope.myValue is truthy (element is visible) -->

<div ng-show="myValue"></div>

<!-- when $scope.myValue is falsy (element is hidden) -->

<div ng-show="myValue" class="ng-hide"></div>

When the ngShow expression evaluates to a falsy value then the .ng-hide CSS class is added to the class attribute on the element causing it to become hidden. When truthy, the .ng-hide CSS class is removed from the element causing the element not to appear hidden.

The ng-if directive removes the HTML element if the expression evaluates to false.If the if statement evaluates to true, a copy of the Element is added in the DOM.The ng-if directive is different from the ng-hide, which hides the display of the element, where the ng-if directive completely removes the element from the DOM.

The ngRepeat directive instantiates a template once per item from a collection. Each template instance gets its own scope, where the given loop variable is set to the current collection item, and $index is set to the item index or key.

Filters are used to change modify the data and can be clubbed in expression or directives using pipe character. Following is the list of commonly used filters.

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Name** | **Description** |
| 1 | uppercase | converts a text to upper case text. |
| 2 | lowercase | converts a text to lower case text. |
| 3 | currency | formats text in a currency format. |
| 4 | filter | filter the array to a subset of it based on provided criteria. |
| 5 | orderby | orders the array based on provided criteria. |

Add uppercase filter to an expression using pipe character. Here we've added uppercase filter to print student name in all capital letters.

Enter first name:<input type = "text" ng-model = "student.firstName">

Enter last name: <input type = "text" ng-model = "student.lastName">

Name in Upper Case: {{student.fullName() | uppercase}}

## lowercase filter

Add lowercase filter to an expression using pipe character. Here we've added lowercase filter to print student name in all lowercase letters.

Enter first name:<input type = "text" ng-model = "student.firstName">

Enter last name: <input type = "text" ng-model = "student.lastName">

Name in Lower Case: {{student.fullName() | lowercase}}

## currency filter

Add currency filter to an expression returning number using pipe character. Here we've added currency filter to print fees using currency format.

Enter fees: <input type = "text" ng-model = "student.fees">

fees: {{student.fees | currency}}

## filter filter

To display only required subjects, we've used subjectName as filter.

Enter subject: <input type = "text" ng-model = "subjectName">

Subject:

<ul>

<li ng-repeat = "subject in student.subjects | filter: subjectName">

{{ subject.name + ', marks:' + subject.marks }}

</li>

</ul>

## orderby filter

To order subjects by marks, we've used orderBy marks.

Subject:

<ul>

<li ng-repeat = "subject in student.subjects | orderBy:'marks'">

{{ subject.name + ', marks:' + subject.marks }}

</li>

</ul>

Custom Directives:

Custom directives are used in AngularJS to extend the functionality of HTML. Custom directives are defined using "directive" function. A custom directive simply replaces the element for which it is activated. AngularJS application during bootstrap finds the matching elements and do one time activity using its compile() method of the custom directive then process the element using link() method of the custom directive based on the scope of the directive. AngularJS provides support to create custom directives for following type of elements.

* **Element directives** − Directive activates when a matching element is encountered.
* **Attribute** − Directive activates when a matching attribute is encountered.
* **CSS** − Directive activates when a matching css style is encountered.
* **Comment** − Directive activates when a matching comment is encountered.