WEB DEVELOPMENT USING MEAN STACK

Unit 4 - Building Single Page App with Angular js

What is AngularJS?

AngularJS is a platform for making dynamic web pages. It is a framework of javascript that is structural too. AngularJS is one of the parts of MEAN stack technology, nowadays very popular consist of four parts.

- **MongoDB** which is a NoSQL database
- Express.js which is the server of a web application
- Angularjs for the front end part
- Node.js used for back-end purpose

Misko Hevery and **Adam Abrons** developed AngularJS initially in 2009. Later on, supported by Google 2010.

AngularJS is an open source framework. It means its original source code is freely available to all. Anyone can access its code as well as can make changes in it too. It works on a client side.

Features of AngularJS?

i. Not browser specific
It can run on all major browsers except internet explorer 8.0 and smartphones including Android and ios based phones/tablets.

ii. Code Less
A programmer has to write less and can perform more functionality with the same code.

iii. Productivity
A user interface is created in a simple way and that too fast also because of strong template syntax.

iv. Speed and performance
Speed is fast as angular apps code loads quickly due to code splitting.

v. Dependency Injection
This built-in injection helps in developing the application easily as well as it is easy to understand.

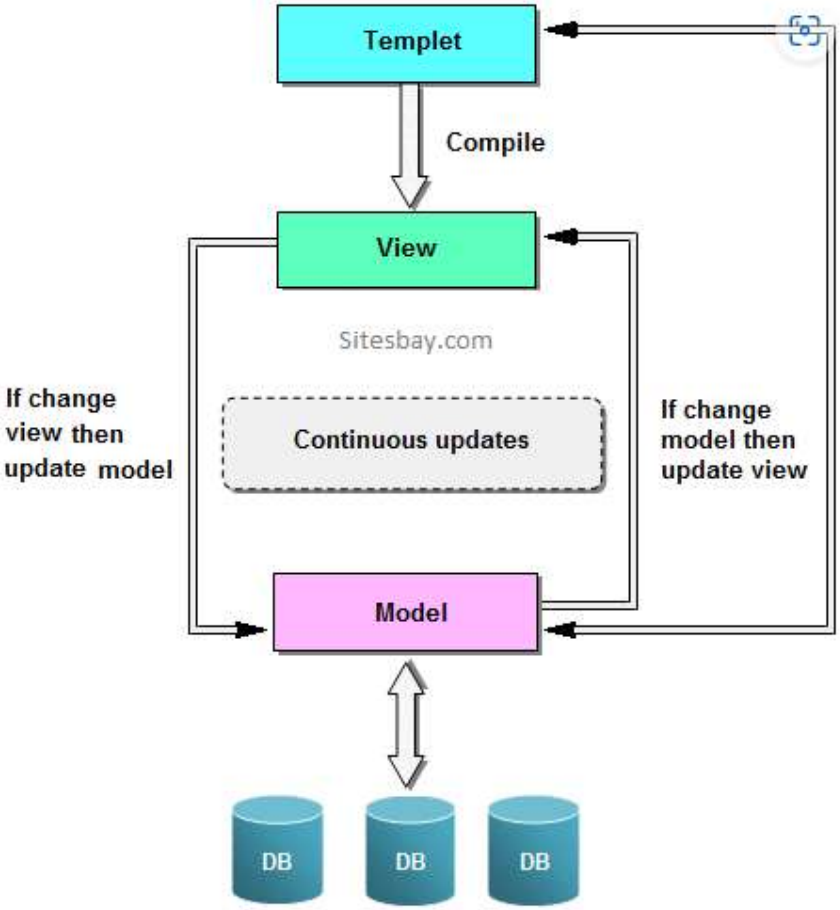
v. Deep Linking
It allows bookmarking the web page. The page gets saved by its URL without getting its state change.

vi. Routing
Routing allows the switching between views.

The following List of companies uses the AngularJS:

- Paypal
- Jetblue
- Istock photo
- Weather
- lego
- Netflix

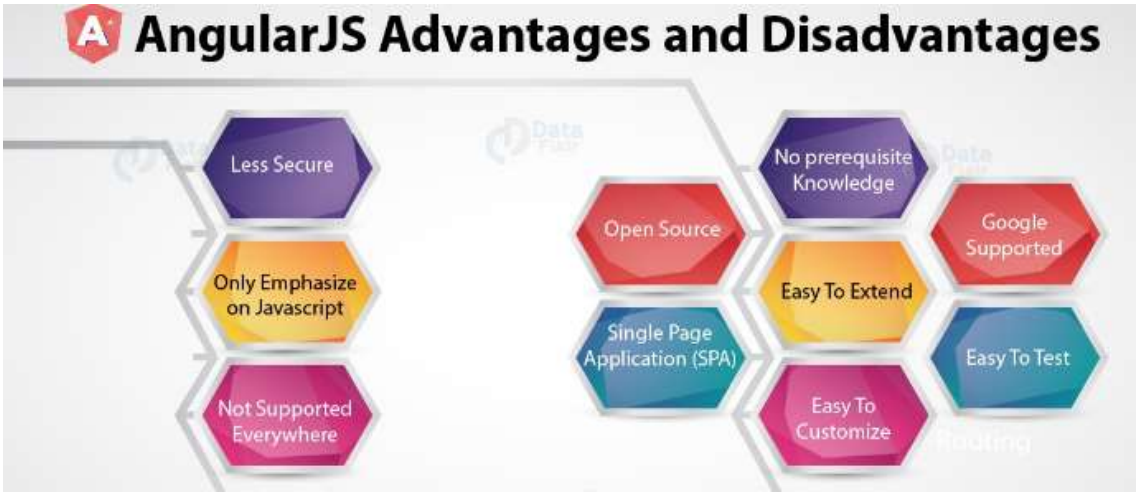
Architecture of AngularJS?



The angular framework is based on MVC architecture abbreviated as *Model View Controller*

- **Model:** It consists of the data of the application. Request made from the view part is handled by it.
- **View:** Presentation layer represents by a view part. The user interaction is done with a view. Display task is done by view part.
- **Controller:** The linking between the model part and view part handled by the controller. It receives input and then verifies it after that perform the operation to modify the state of the data model.

AngularJS Advantages and Disadvantages?



Advantages of AngularJS

i. Open Source

Since it is an open source JavaScript MVC framework. Therefore, custom apps can be made available to anyone at the affordable cost.

ii. Single Page Application (SPA)

AngularJS create single page application, which can work fast and it is user-friendly.

iii. No prerequisite Knowledge

JavaScipt and HTML both are the only prerequisites to learn AngularJS.

iv. Easy To Extend

IT is easy to extend as it contains built-in attributes that make it possible to extend the functionality of HTML by attaching a specific behaviour with it.

v. Easy to Customize

It is easy to customize as one can create its own directives too in it.

vi. Google Supported

AngularJS has the support of Google, a large community. One can fully relax about the regular updates as well as its performance should also get better.

vii. Easy to Test

An Angular code is easy to unit test as there is an inbuilt dependency injection comes in it which makes it easier to test individual components of the code. As well as, it supports both unit testing and integration testing.

Limitations of AngularJS

Less Secure

It is not secure because there is no server authorization and authentication in angular so one cannot verify the identity in it. Security is one of the major concerns in today's world technology is expanding rapidly so as the security concerns as it is necessary that your data should remain secure.

Only Emphasize on Javascript

It is just a single static page if its javascript side gets hidden.

Memory Leakage

Also, it is a framework of javascript and there is an issue of memory leak in javascript which makes its performance low.

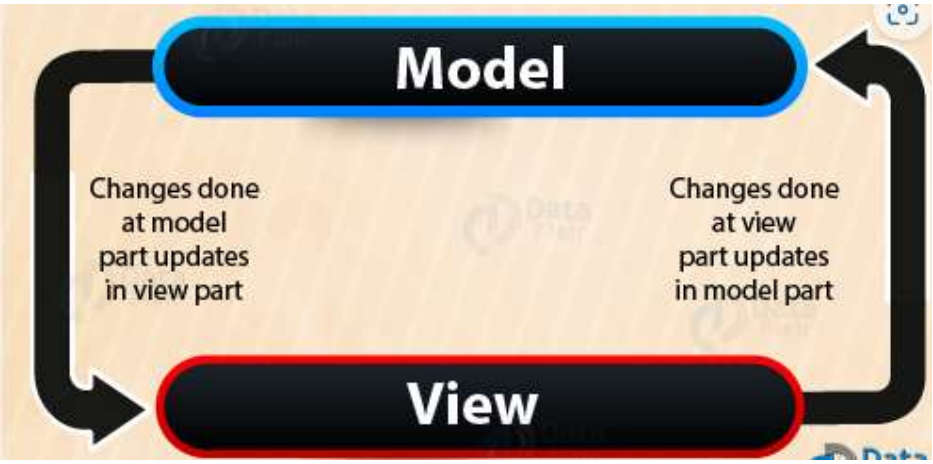
Not Supported Everywhere

AngularJS does not support internet explorer 8.0

What is Data Binding?

Data binding is the synchronization of data between business logic and view of the application. It serves as a bridge between two components of angular that is model part and view part. Data Binding is automatic and provides a way to wire the two important part of an application that is the UI and application data.

Whenever some changes are done at the model side it is reflected at view side too and vice versa is also possible. This happens so rapidly to make sure that view and the model part will get update all the time.



Types of Data Binding in AngularJS

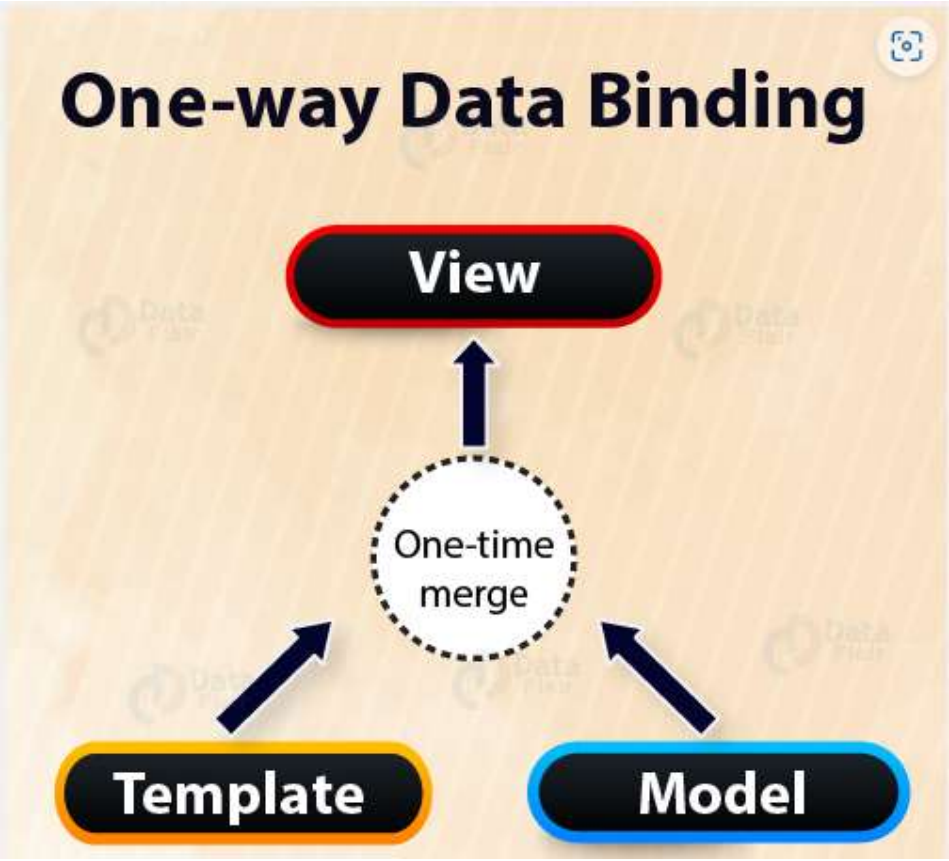
The various types of binding in angular js are as follow:

- One Way Data Binding
- Two Way Data Binding

One Way Data Binding

In one-way data binding, the flow of data restricts to one side only and that is from model to view. It follows a unidirectional approach. In the case of updating in the model part, same will sync in the view

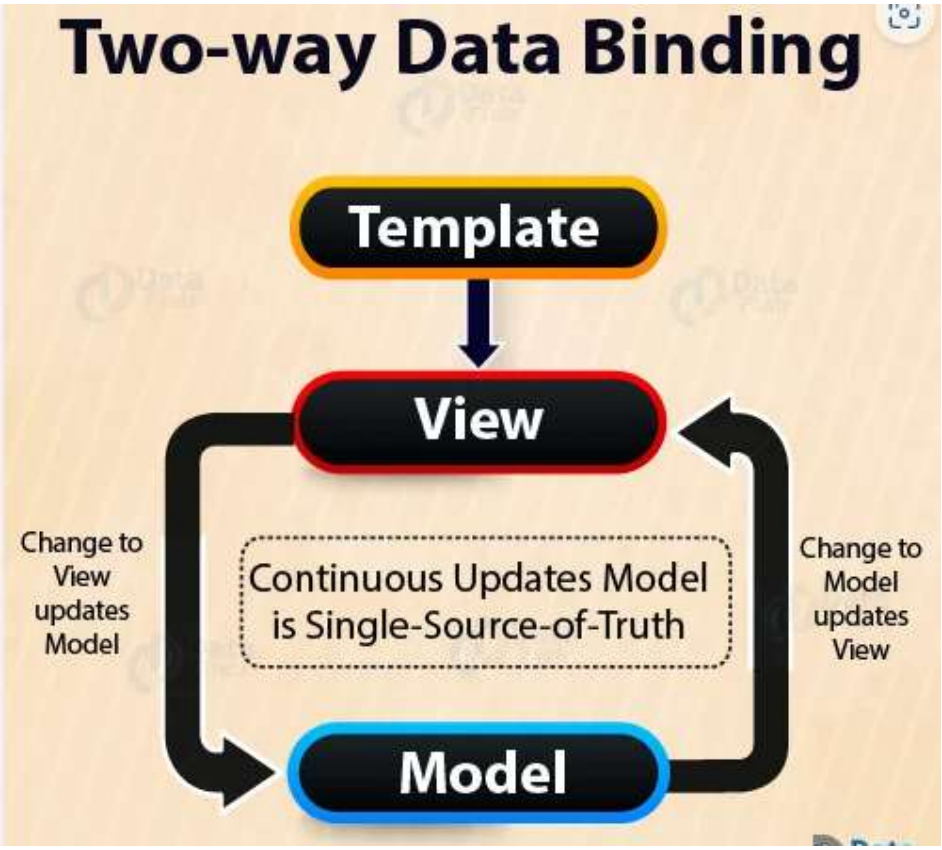
part also but the vice versa is not possible as in one way binding data can't flow from view to model. Interpolation binding (an expression is taken as an input and it is changed as a text using HTML elements), property binding (Value of a property is set on an HTML element) comes in the category of one-way data binding in AngularJS.



Two Way Data Binding in AngularJS

In the case of two data binding in AngularJS, the flow of data does not restricts to one side only. A flow of data is from model to view as well as the view to model is also possible. Two-way data binding follows a bidirectional approach as a flow of data is possible in both the side. Any changes made in the model part will sync in view part as well as any changes made in view part is synced in model part also.

Event binding (an event triggers at view side by the user event binding allows components to listen to that event.) comes in the category of two-way data binding in AngularJS.



AngularJS Expressions?

In AngularJS, expressions are used to bind application data to HTML. AngularJS resolves the expression, and return the result exactly where the expression is written.

Expressions are written inside double braces {{expression}}. They can also be written inside a directive:

```
<p>A simple expression example: {{ 5 + 5 }}</p>
</div>
</body>
```

AngularJS Controllers?

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

You can attach properties and methods to the $scope object inside a controller function, which in turn will add/update the data and attach behaviours to HTML elements. The $scope object is a glue between the controller and HTML.

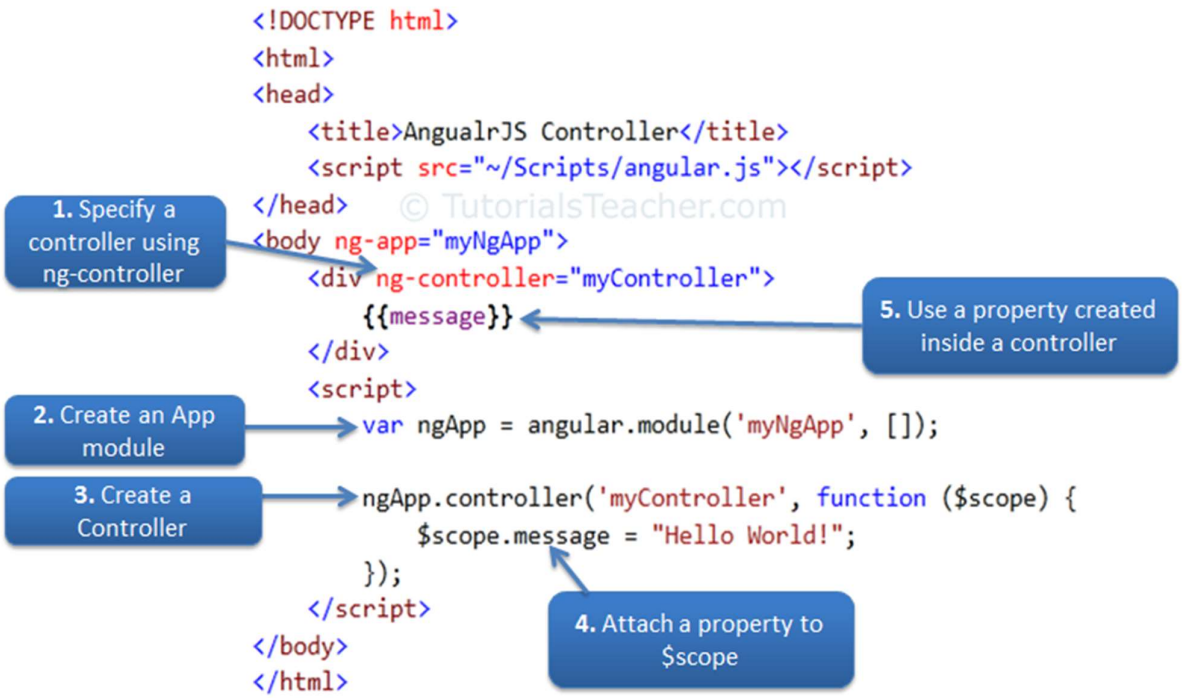
The `ng-controller` directive is used to specify a controller in HTML element, which will add behavior or maintain the data in that HTML element and its child elements.

The following example demonstrates attaching properties to the $scope object inside a controller and then displaying property value in HTML.

Example: AngularJS Controller

```
<!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>
```



AngularJS Modules?

An **angular** module is a container which we use to create an application. By containing angularjs specific functions in a module, angular JS stops polluting global scope. In AngularJS Modules, we can apply functionality of an application to the entire HTML page with the help of ng-directive.

AngularJS modules is used to define functionalities or to add controllers, filters, **directives** etc to the angular application. Also, make it easy to reuse in different applications. Separation of concern using angular support by using it.

An AngularJS module defines an application. The module is a container for the different parts of an application. The module is a container for the application controllers. Controllers always belong to a module.

Creating a Module

A module is created by using the AngularJS function `angular.module`

```
<div ng-app="myApp">...</div>

<script>

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

</script>
```

The "myApp" parameter refers to an HTML element in which the application will run.

Now you can add controllers, directives, filters, and more, to your AngularJS application.

Types of AngularJS Modules?

There are two types of AngularJS modules:

- Application Modules
- Controller Modules

Application Modules :- Using the function "**angular.module**" we can create an application module in angular. To create Angular Modules, application modules consider as global space, retrieving angular **j**s modules and registering angular modules.

Example: Create Application Module

```
<!DOCTYPE html>
<html >
<head>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app="myApp">
    @* HTML content *@
    <script>
        var myApp = angular.module('myApp', []);

    </script>
</body>
</html>
```

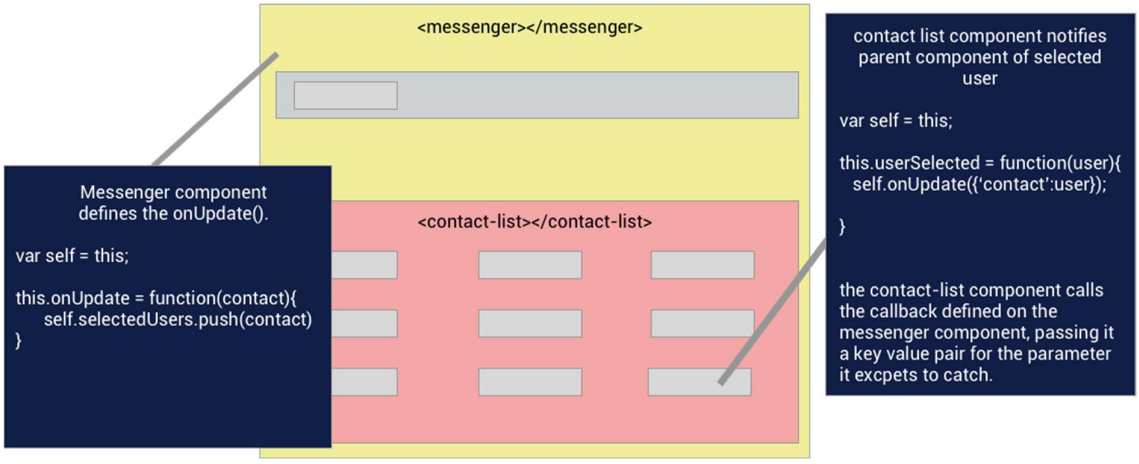
Controller Modules:- In the **controller module**, you have to add a controller in our application and that controller refers to the ng-controller directive.

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

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

Components in AngularJS?

- A Component is nothing but a simple TypeScript class where you can create your own methods and properties as per your requirement which is used to bind with a UI (html or cs html page) of our application.
- @component decorator provides an additional metadata that determines how the component should be processed, instantiated, and used at runtime.
- A component in Angular is an isolated entity that enables reuse and maintainability of the code. In simple terms, a component can be considered as a method/function that not only contains the controller logic required for a UI element to function but also the corresponding HTML tags to generate the element.
- Components also offer great help in making sure the design across the application is consistent.



Dependency Injection?

AngularJS Dependency injection defines, how the various components of the software are dependent on each other. Instead of hard-coding a component within another component, a component is given their own dependencies using dependency injection.

Why Dependency Injection?

- AngularJS Dependency injections make an application modularize.
- AngularJS DI makes easier to reuse components of an application.
- It makes easier to test components of an application.
- It makes easier to configure components of an application.
- Separate the process of creation and consumption of dependencies.
- Let the consumer worry only about how to use the dependency, and leave the process of creation of the dependency to somebody else.

AngularJS Filter?

AngularJS filter is a tool, which we can use to format the data. With this filter, the user can see and modify according to the requirement. It is added in angular to format the data that is being displayed on the view part.

Syntax of Filter
With pipe character (|), we can add filters in angularJS. Filters can club with the expression as well as a directive.

Syntax:

```
1.  {{expression| name_of_filter}}
```

Example: {{name | uppercase}}

Here, a name is the expression and uppercase is the built-in filter provided by angular. It converts the name in uppercase in view part.

Example: {{name | uppercase}}
Here, a name is the expression and uppercase is the built-in filter provided by angular. It converts the name in uppercase in view part.

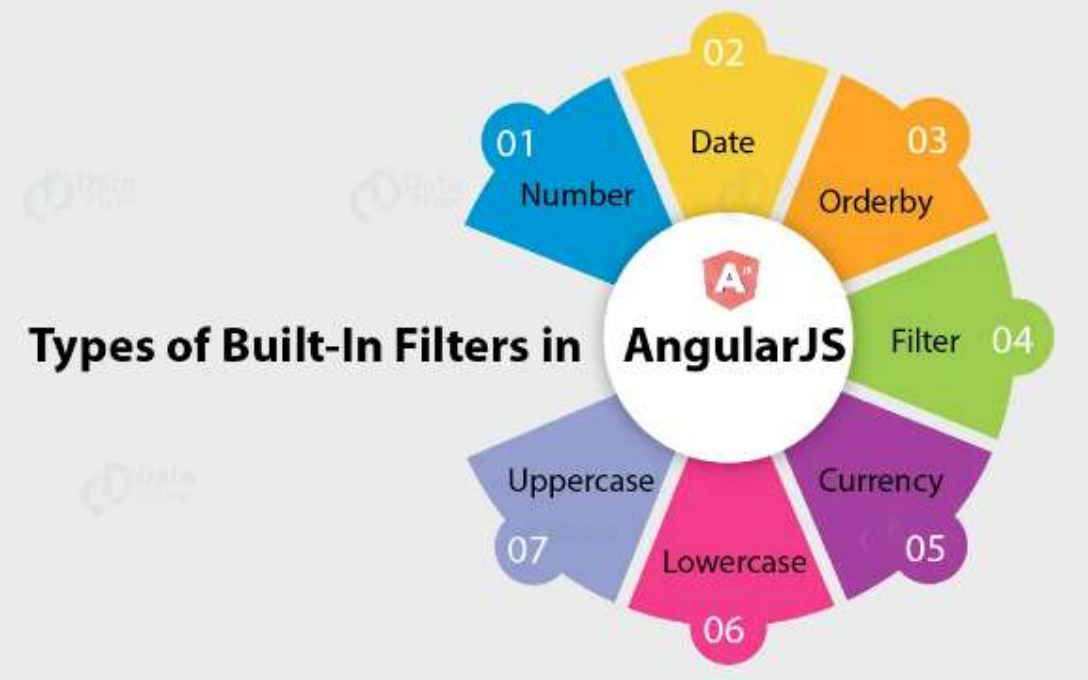
When to use a filter in AngularJS?

When we want to display the data in view part in a particular format in that scenario we can use AngularJS filter. We can display the data in the uppercase format, lowercase format etc. In whatever format, we entered the text but it can easily get displayed in any format by angular as per the type of filter used. AngularJS Filters can change the way of displaying the output.

Types of Filters in AngularJS?

- Built-in Filters
- Custom Filters
- Stateful Filters

Types of Built-In Filters in AngularJS



Custom Filters in AngularJS

Angular JS provides a way to create our own filter that is customized filter. You can create one by registering a filter factory function in a module. AngularJS Filter function should be a pure function. Pure function means the same result is produced as per the given input. Also, it should not affect an external state.

```
angular.filter("filter_name", function(){return function(){}}
```

Stateful Filter in AngularJS

It is discouraged to write stateful filters because angularJS cannot optimize its execution. Rather than writing a stateful filter, you can mark a filter as stateful. By marking a filter as stateful, during each digest cycle filter can execute one or more time.

AngularJS Tables?

The ng-repeat directive is perfect for displaying tables.

AngularJS Example

```
<div ng-app="myApp" ng-controller="customersCtrl">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
  $http.get("customers.php")
  .then(function (response) {$scope.names = response.data.records;});
});
</script>
```

Display with orderBy Filter

To sort the table, add an **orderBy** filter:

AngularJS Example

```
<table>
  <tr ng-repeat="x in names | orderBy : 'Country'">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>
```

AngularJS Forms & Forms validation?

Forms are an integral part of a web application. Practically every application comes with forms to be filled in by the users. Angular forms are used to log in, update a profile, enter sensitive information, and perform many other data-entry tasks. In this article, you will learn about how to create a form and validate the information filled.

Types of Angular Forms

There are two types of form building supported by Angular Forms.

- Template-Driven Approach
- Reactive Approach

Template-Driven Approach

- In this method, the conventional form tag is used to create forms. Angular automatically interprets and creates a form object representation for the tag.
- Controls can be added to the form using the NGModel tag. Multiple controls can be grouped using the NGControlGroup module.
- A form value can be generated using the “form.value” object. Form data is exported as JSON values when the submit method is called.
- Basic HTML validations can be used to validate the form fields. In the case of custom validations, directives can be used.
- Arguably, this method is the simplest way to create an Angular App.

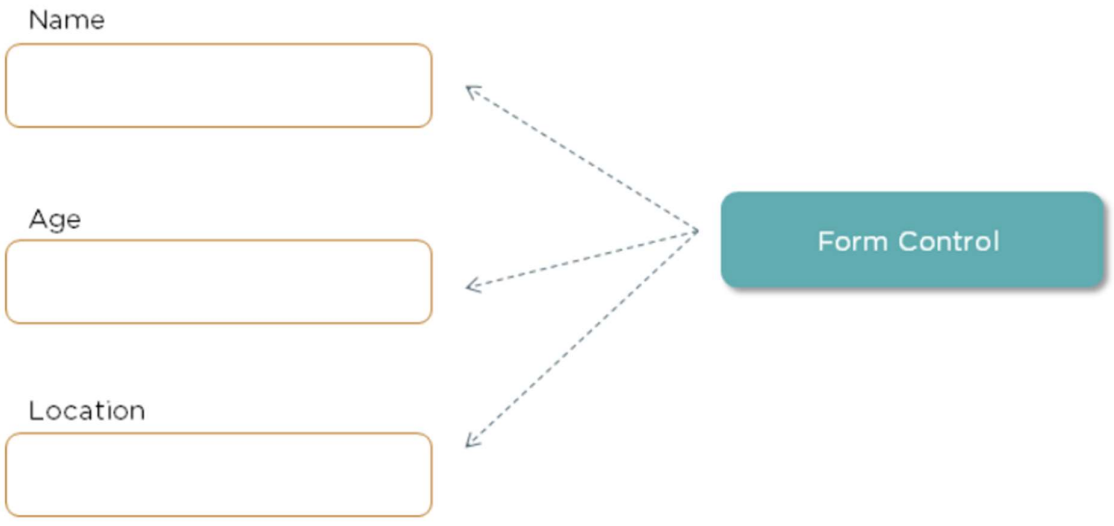
Reactive Form Approach

- This approach is the programming paradigm oriented around data flows and propagation of change.
- With Reactive forms, the component directly manages the data flows between the form controls and the data models.

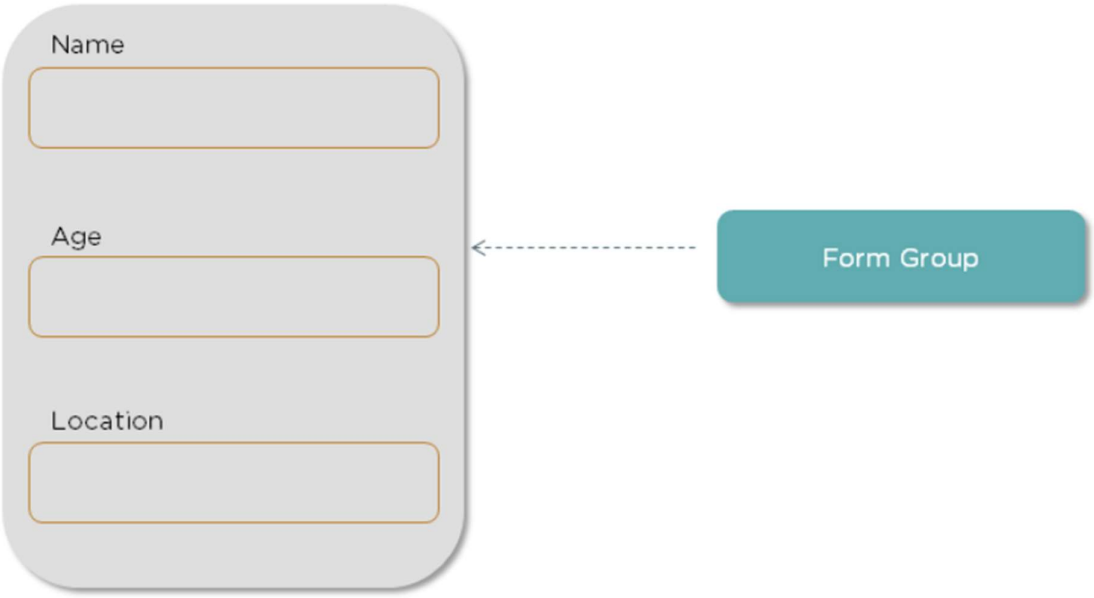
- Reactive forms are code-driven, unlike the template-driven approach.
- Reactive forms break from the traditional declarative approach.
- Reactive forms eliminate the anti-pattern of updating the data model via two-way data binding.
- Typically, Reactive form control creation is synchronous and can be unit tested with synchronous programming techniques.

Form Control

Form Control is a class that enables validation. For each input field, an instance of this class is created. These instances help check the values of the field and see if they are touched, untouched, dirty, pristine, valid, invalid, and so on.



Form Group



Form Group class represents a group of controls. A form can have multiple control groups. The Form Group class returns true if all the controls are valid and also provides validation errors, if any.

AngularJS Form Validation?

AngularJS performs **form validation** on the client side. AngularJS monitors the state of the form and input fields (input, text-area, select), and notify the user about the current state. AngularJS also holds information about whether the input fields have been touched, modified, or not. Form input fields have the following states:

- **$untouched:** It shows that field has not been touched yet.
- **$touched:** It shows that field has been touched.
- **$pristine:** It represents that the field has not been modified yet.
- **$dirty:** It illustrates that the field has been modified.
- **$invalid:** It specifies that the field content is not valid.
- **$valid:** It specifies that the field content is valid

Introduction to AngularJs ng-options?

AngularJS ng-options is an In-Built AngularJS directive widely used in the HTML view to loop through the List of contents containing Objects on UI and utilizes or display any particular object attribute objects of each element of the list in HTML page along with styling in an ANGULARJS application. For example, the ng-change attribute is used in UI along with dropdown and provides the ability to add additional options or default selected option values.

```
<div title="Country List">Country</div>
<select class="any-style-for-dropdown"
ng-options="country.capital for country in vm.Countries" ng-
model="vm.selectedCountry">
</select>
```

The ng-options attribute is always used along with the HTML dropdown using the select HTML element. The select HTML element accepts the list of options which needs to be displayed on the UI, so now it’s not always that the value which needs to be displayed on UI dropdown as constant values; some values come from some external service or after manipulating it in the controller we want that list of objects to be used on the UI, in such case ng-options comes into the picture. Ng-options allows the Developer to create a local object

variable which can be used in ng-options value to refer to a particular attribute of each object in the list. So if you need a particular value to be displayed as the default value for at the load of the page, then that can do by using the HTML option element and using the attribute selected. This will keep the default value selected.

ng-init can also be used along with the ng-options directive, which can help you to add an additional parameter for each object in the list ONLY for that scope of the template instance. Something to keep a note of while using ng-options is that any object starting with the $ symbol will not be read by the ng-options directive and get ignored as its AngularJS and $ is a prefix used by AngularJS for public or private properties. Also, another point to keep a note of is that the ng-model is required for ng-options to work seamlessly as this model value gets set at the selected value in the HTML view.

AngularJS AJAX?

AJAX is the short form of Asynchronous JavaScript and XML. AJAX was primarily designed for updating parts of a web page, without reloading the whole page. The reason for designing this technology was to reduce the number of round trips which were made between the client and the server and the number of entire page refresh which used to take place whenever a user required certain information.

Many modern-day web applications actually follow this technique for refreshing the page or getting data from the server.

Angular provides two different APIs to handle Ajax requests. They are

- $resource
- $http

AngularJS provides a $http service for reading data from remote servers. It is used to retrieve the desired records from a server.

AngularJS requires data in JSON format. Once the data is ready, $http gets the data form server in the following manner:

```
function employeeController($scope,$http) {
r url = "data.txt";

 $http.get(url).success( function(response) {
   $scope.employees = response;
});
```

Here the file "data.txt" is employee's records. $http service makes an AJAX call and sets response to its property employees. This model is used to draw tables in HTML.

How to register the controller with the module

```
//Create the module
var myApp = angular.module("myModule", []);

//Create the controller
var myController = function ($scope) {
    $scope.message = "AngularJS Tutorial";
}

// Register the controller with the module
myApp.controller("myController", myController);
```

OR

```
//Create the module
var myApp = angular.module("myModule", []);

// Creating the controller and registering with the module all done in one line
myApp.controller("myController", function ($scope) {
    $scope.message = "AngularJS Tutorial";
});
```

Attaching a complex object to the scope

```
myApp.controller("myController", function ($scope) {
    var employee = {
        firstName: 'Mark',
        lastName: 'Hastings',
        gender: 'Male'
    };
    $scope.employee = employee;
});
```

With in the view, we can then retrieve the employee properties and display

```
<div ng-controller="myController">
    <div>First Name : {{ employee.firstName }}</div>
    <div>Last Name : {{ employee.lastName }}</div>
    <div>Gender : {{ employee.gender}}</div>
</div>
```

```
var app = angular
          .module("myModule", [])
          .controller("myController", function ($scope) {
              var employees = [
                  { firstName: "Ben", lastName: "Hastings", gender: "Male", salary: 55000 },
                  { firstName: "Sara", lastName: "Paul", gender: "Female", salary: 68000 },
                  { firstName: "Mark", lastName: "Holland", gender: "Male", salary: 57000 },
                  { firstName: "Pam", lastName: "Macintosh", gender: "Female", salary: 53000 },
                  { firstName: "Todd", lastName: "Barber", gender: "Male", salary: 60000 }
              ];

              $scope.employees = employees;
          });
```

```
<div ng-controller="myController">
    <table>
        <thead>
            <tr>
                <th>Firstname</th>
                <th>Lastname</th>
                <th>Gender</th>
                <th>Salary</th>
            </tr>
        </thead>
        <tbody>
            <tr ng-repeat="employee in employees">
                <td>{{ employee.firstName }}</td>
                <td>{{ employee.lastName }}</td>
                <td>{{ employee.gender }}</td>
                <td>{{ employee.firstName }}</td>
            </tr>
        </tbody>
    </table>
</div>
```