ANGULARJS 1.0
&
FACEBOOK LOGIN USING FIREBASE API

- BY PULKIT KAKKAR

WHAT WE ARE GOING TO MAKE ?

We are going to lay the fundamentals to build a **Group Chat Box** with following features :-

1) Create Account Using Facebook Login
2) Sign-In Page Using Firebase Auth API
3) Option to create new Chat Box(Group) for Each Topic
4) Group Chat for each Chat Box
5) Change ChatBox(Group) Image

Firebase

Bootstrap

facebook developers

INTRO TO ANGULAR JS

1. A Brief History About AngularJS
2. Why Angular JS
3. Intro To MVC Programming Paradigm
4. HelloWorld
5. Controllers
6. Views
7. Routing
8. Directives(built-in and custom)
9. Filters
10. Service, Providers, Factory

HISTORY OF ANGULAR JS

WHY ANGULAR JS ?

1 - It Was Developed by Google

This one may seem obvious, but it's important to remember that many (not all) frameworks are made by hobbyists in the open source community. While passion and drive have forged frameworks, like Cappucino and Knockout, Angular is built and maintained by dedicated (and highly talented) Google engineers. This means you not only have a large open community to learn from, but you also have skilled, highly-available engineers tasked to help you get your Angular questions answered.

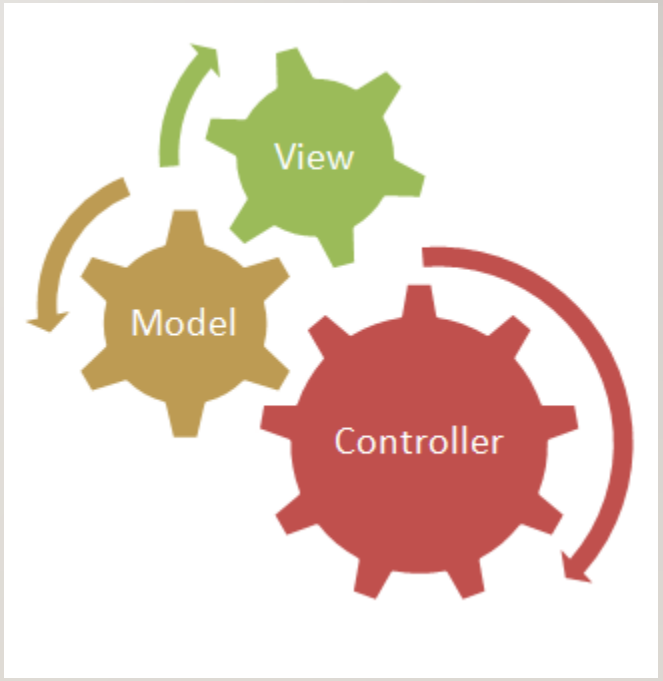
2- IT'S DECLARATIVE AND NOT IMPERATIVE

In contrast to the imperative programming, declarative programming is about describing what you're trying to achieve, without instructing how to do it.

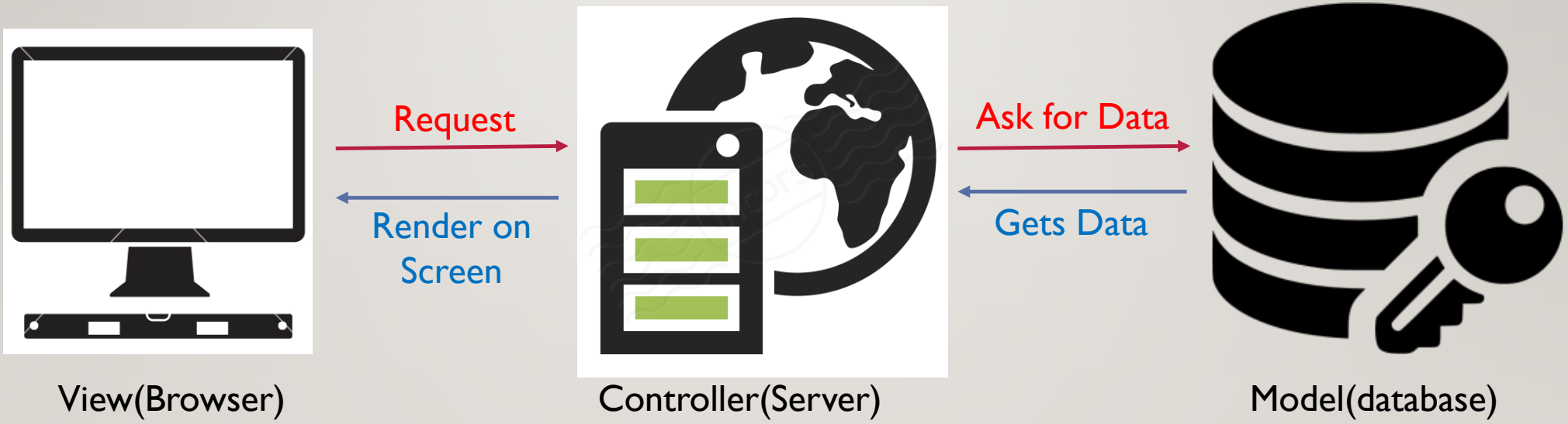
3- IT'S MVC

What is 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**. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.



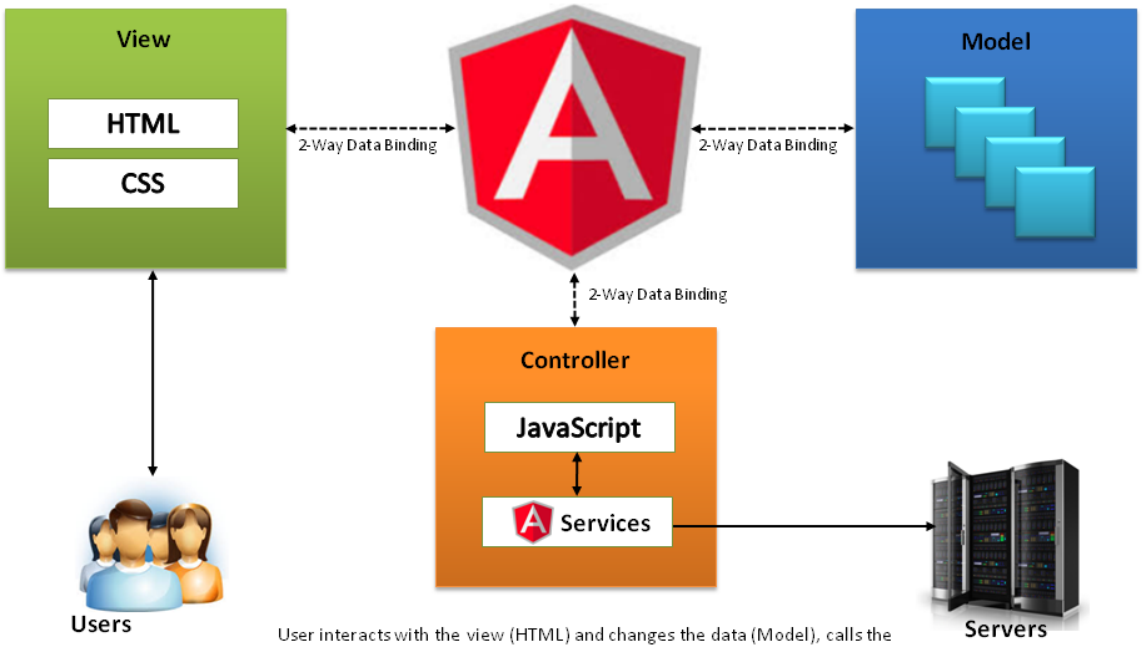
MVC ANALOGY


Request
Render on Screen
Ask for Data
Gets Data
View(Browser)
Controller(Server)
Model(database)

MVC COMPONENTS

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

MVC RELATIONSHIP IN ANGULAR



LET'S START WITH REAL-TIME CODING

"Most good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program." – Linus Torvalds

INSTALLING ANGULAR

1. Open Your Base Template :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Base Template</title>
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">
    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="js/bootstrap.min.js"></script>
  </head>
  <body>
    <h1>Hello, Peeps!!!</h1>
  </body>
</html>
```

2. Copy The Following Line into Your Code

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.6/angular.min.js"></script>
```

There are two types of Angular script URLs you can point to, one for development and one for production:
angular.js — This is the human-readable, non-minified version, suitable for web development.
angular.min.js — This is the minified version, which we strongly suggest you use in production.

3. Congrats you've Finally installed angular. You may use the other installation options available at https://docs.angularjs.org/misc/downloading

This is How our Final Code Looks Like:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Base Template</title>
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">
    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="js/bootstrap.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.6/angular.min.js"></script>
  </head>
  <body>
    <h1>Hello, Peeps!!!</h1>
  </body>
</html>
```

HELLOWORLD

In Order To illustrate the basic concepts of Angular JS, We'll Code a basic HelloWorld which Simply binds an input from text field , to the heading tag

Code:

```
<body ng-app>
  <br/>
  <center>
    <label for="Greeting">Greeting Text:</label>
    <input type="text" ng-model="GreetText"/>
    <br/>
    <h1>Hey, {{GreetText}}</h1>
  </center>
</body>
```

** Code Available in file '1 Helloworld.html'

What's Different ??

i.) ng-App

```
<body ng-app>
```

The green colored attribute above is a Directives in AngularJS, Directive often starts with 'ng' or 'data-ng'
Read more about ng VS data-ng at: http://stackoverflow.com/questions/16589853/ng-app-vs-data-ng-app-what-is-the-difference

ng-app directive is used to **auto-bootstrap** an AngularJS application. The ngApp directive designates the **root element** of the application and is typically placed near the root element of the page like <body> or <html>

Remember Few things :

1. Only one AngularJS application can be auto-bootstrapped per HTML document. To run multiple applications in an HTML document you must manually bootstrap them using angular.bootstrap function instead
2. AngularJS applications cannot be nested within each other.

What's Different ??

ii) ng-model

```
<input type="text" ng-model="GreetText"/>
```

The ngModel directive binds an input,select, textarea (or custom form control) to a property on the scope

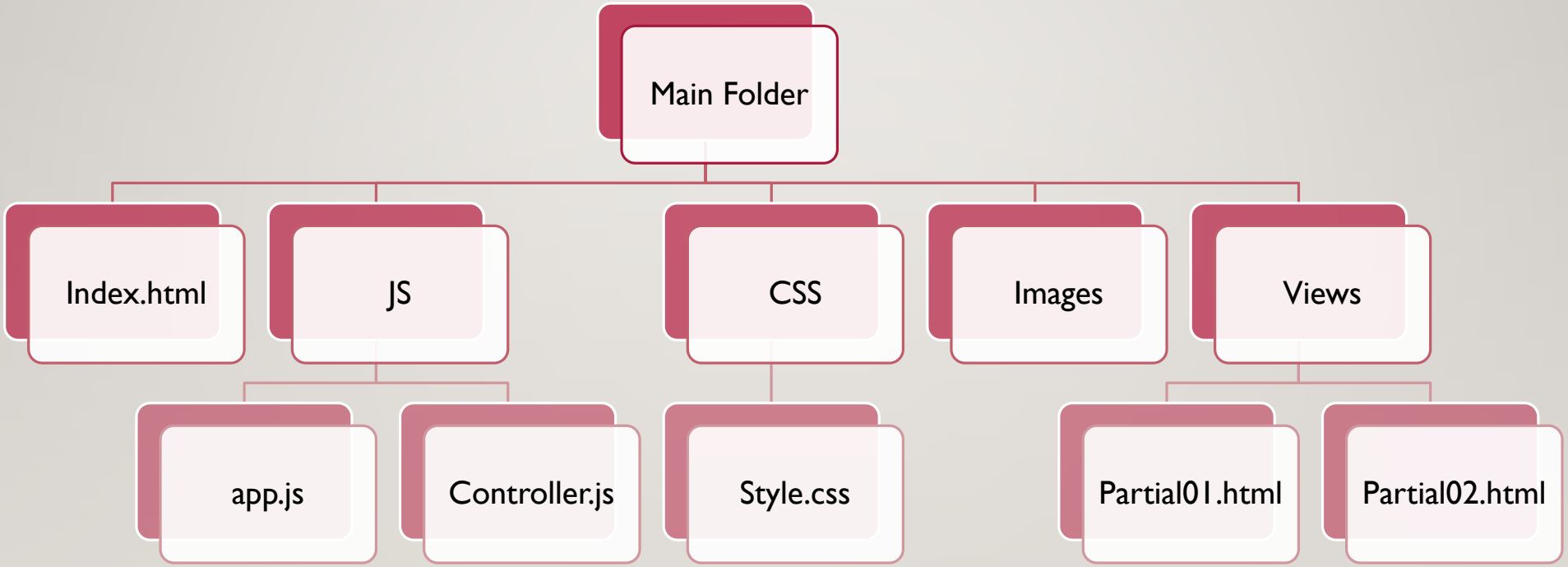
Note: ngModel will try to bind to the property given by evaluating the expression on the current scope. If the property doesn't already exist on this scope, it will be created implicitly and added to the scope.
Read more at : https://docs.angularjs.org/api/ng/directive/ngModel

iii) Double Curly Brackets {{Model_Name}}

```
<h1>Hey, {{GreetText}}</h1>
```

The double curly brace notation {{ }} binds expressions to elements, it is built-in in Angular markup.
It Can also be used to perform basic mathematical operations

DIRECTORY STRUCTURE FOR ANGULAR APPS



GETTING STARTED WITH THE CHAT APP

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Base Template</title>
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="CSS/style.css">
    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.6/angular.min.js"></script>
  </head>
  <body ng-app>
  <div class="container">
      <form class="form-signin">
          <h2 class="form-signin-heading">Please sign in</h2>
          <label for="inputEmail" class="sr-only">Email address</label>
          <input type="email" id="inputEmail" class="form-control" placeholder="Email address" required autofocus>
          <label for="inputPassword" class="sr-only">Password</label>
          <input type="password" id="inputPassword" class="form-control" placeholder="Password" required>
          <button class="btn btn-lg btn-primary btn-block col-md-6">Sign In</button>
          <button class="btn btn-lg btn-primary btn-block col-md-6">Sign Up</button>
      </form>

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

CREATING A MODULE

An AngularJS module defines an application. It defines the scope of an application. It is a container for the different parts of an application. It contains services, directives, controllers, filters, and configuration information

An App may have multiple modules but right now we just have one module named "PureChat"

```
<body ng-app="PureChat">
```

To Define a module Open JS Folder and add following line to your app.js:

```
var myApp = angular.module('PureChat',[]);
```

CONTROLLERS

- Controllers are used to create **Presentational Logic** for the applications
- They are also used for Scope initializations
- They shouldn't be used for writing Business Logics, Factory and services must be used instead
- It shouldn't also be used for DOM manipulation and Filtering
- Syntax For Defining Controller :

```
myApp.controller('NameCtrl', ['$scope', function ($scope) {

}]);
```

SIGN IN CONTROLLER DETAIL

Let's Create Our very First Controller For Signing In , and name it as "SignInCtrl"

```
myApp.controller('SignInCtrl', ['$scope', function ($scope) {
    $scope.text = "Hello";
}]);
```

```
<form ng-controller="SignInCtrl" class="form-signin">
   <h2 class="form-signin-heading">{{text}} Please sign in</h2>
```

- Scope Properties can be accesed only within the scope of controller
- A module may have more than one controllers
- It Can Be used for initializing too, but ng-init is a better option
- Functions can also be declared to the scope properties, as illustrated.
- Double Braces can also be used to invoke a function and print the return parameter

UNDERSTANDING $SCOPE

It's an object that refer to application model

They are arranged hierarchically

Access using $parent in nested controllers

$scope

They are intermediate between the view and controllers

Can by accessed using ng-model and ng-bind(braces and ng-bind notation)

USING CONTROLLER AS SYNTAX

Index.html

```
<div ng-controller="Ctrl3 as parent">
  <h1>Hey,{{parent.prop}}</h1>
  <div ng-controller="Ctrl4 as son">
    <h1>Hey,{{son.prop}} son of {{parent.prop}}</h1>
  </div>
</div>
```

App.js

```
.controller('Ctrl3',  function ($scope) {
    this.prop = "Ctrl3";
})
.controller('Ctrl4',  function ($scope) {
    this.prop = "Ctrl4";
})
```

ANGULAR SERVICES

- As mentioned earlier, services should be used to write the business logic of the application
- Code is shared using the angular dependency injection
- Services are Lazy Loaded and hence are good for memory management
- Services are singleton (only one object)
- Angular Provides a bunch of built-in services like :
    1. $http (post, get)
    2. $log (To log to console)
    3. $filter
    4. $locale etc….
- You can Create your own custom service in angular

SERVICE, PROVIDERS AND FACTORY

Service, Provider and Factory does almost the same thing with little differences

Service: Whenever a Sevice is initiated, angular creates an object with 'new' and hence all methods and properties are accessed using 'this' keyword

Factory: It allows to only access the properties and methods returned by the factory

Provider: It's The most verbose version of services, it can be configured

```
myApp.service('basicService', [function () {
    this.serviceVar = "This is service variable";
    this.getServiceVar = function(){
        return this.serviceVar;
    };
}]);
myApp.factory('basicFactory', [function () {

    var fac ={};
    fac.facVar = "This is factory Variable";
    fac.getfacVar = function () {
        return fac.facVar;
    }
    return fac;
}]);
```

```
myApp.controller('serviceCtrl', ['$scope','basicService', function ($scope,basicService) {
    $scope.value = basicService.getServiceVar();

}]);
myApp.controller('factoryCtrl', ['$scope','basicFactory', function ($scope,basicFactory) {
    $scope.value = basicFactory.getfacVar();

}]);
```

BEST PRACTICES

1) Do not Declare your Module as a global variable
2) Use “controller as” syntax instead of $scope
3) Separate factories , modules and services in different files
4) Do not write business logic in Controllers
5) Avoid $parent.$parent….. Syntax in nested controllers

ROUTING IN ANGULAR SPA

If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.
The ngRoute module routes your application to different pages without reloading the entire application.

Routing Can not be tested on the local machine, so in order to use and test routing offline we'll use nodeJS , http-server module

SETTING UP HTTP-SERVER IN NODEJS

1) Download and install node from https://nodejs.org/en/download/
2) Open CMD and execute following command:

```
npm install http-server -g
```

3) Once Installed , Use the command http-server in the CLI in the root of project, whenever you need to create the local server
4) Read More at : https://www.npmjs.com/package/http-server

NG-ROUTE & NG-VIEW

- They both together provides, dynamic linking service for SPAs
- They are used for deep linking
- Ng-route couples with ng-view to provide linking
- We set up our routing during the **configuration phase** of module, so that we have routing ready before the application is bootstrapped
  We can do this because in '**config**' block of the module we can only inject constants or providers and the routing component is a **provider**

SETTING UP ROUTING

1. Include angular-route.min.js in index.html

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.5.6/angular-route.min.js"></script>
```

2. Add ngRoute in module dependency

```
angular.module('PureChat',['ngRoute']);
```

3. Declare a config block and inject $routeProvider

```
angular.module('PureChat',['ngRoute'])
.config(['$routeProvider',function($routeProvider) {

}])
```

4. Start Using It !!!

ADD SIGN UP VIEW TO CHAT APP USING ROUTING

1) Add Sign-up view in Views folder
2) Configure the routeProvider

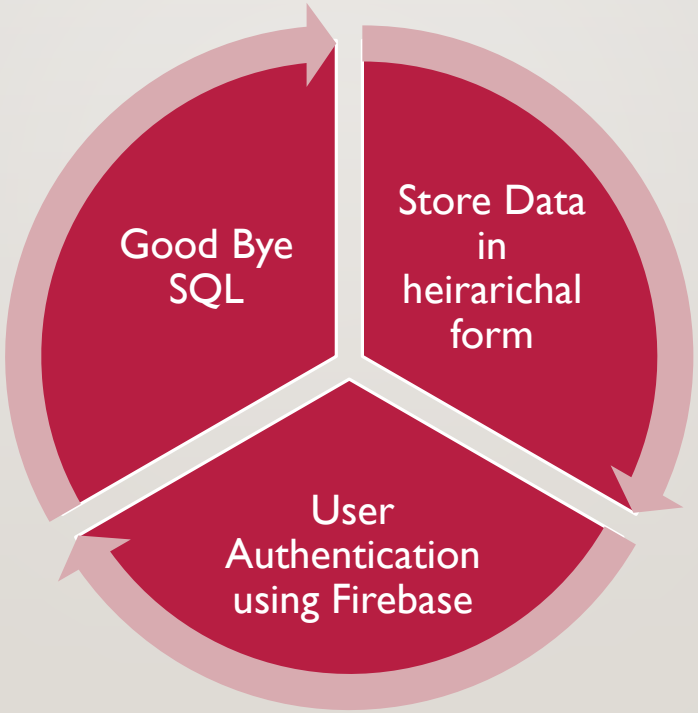
```
.config(['$routeProvider','$locationProvider',function($routeProvider, $locationProvider) {
    $routeProvider.when('/signup', {
        templateUrl: 'Views/signup.html',
        controller: 'SignupCtrl'
    }).otherwise({ redirectTo: '/' })
    $locationProvider.html5mode(true);
}])
```

3) Use ng-href directive

```
<a class="btn btn-lg btn-primary btn-block col-md-6" ng-href="#/signup">Sign Up</a>
```

INTRODUCTION TO FIREBASE

Firebase is a cloud services provider and backend as a service organization.



ANGULAR FIRE AND FIREBASE

AngularFire is the officially supported AngularJS binding for Firebase. The combination of Angular and Firebase provides a three-way data binding between your HTML, your JavaScript, and the Firebase database.

1) **Create an account:**

   The first thing we need to do is sign up for a free Firebase account. A brand new Firebase app will automatically be created with its own unique URL ending in firebaseio.com. We'll use this URL to authenticate users and store and sync data with AngularFire.

2) **Add Script Dependencies**

```
<script src="https://cdn.firebase.com/js/client/2.2.4/firebase.js"></script>
<script src="https://cdn.firebase.com/libs/angularfire/1.2.0/angularfire.min.js"></script>
```

ANGULAR FIRE AND FIREBASE

3.) Inject the AngularFire Services

```
angular.module('PureChat',['ngRoute','firebase'])
```

4) Now the $firebaseObject, $firebaseArray, and $firebaseAuth services are available to be injected into any controller, service, or factory.

ADD THREE-WAY, OBJECT BINDINGS

Angular is known for its two-way data binding between JavaScript models and the DOM, and Firebase has a **lightning-fast, realtime database**. For synchronizing simple key / value pairs, AngularFire can be used to glue the two together, creating a "three-way data binding" which **automatically synchronizes** any changes to your DOM, your JavaScript, and the Firebase database.

```
.controller('dashCtrl', ['$scope','$firebaseObject', function ($scope, $firebaseObject) {
    var ref = new Firebase("https://purechatapp.firebaseio.com/data");
  // download the data into a local object
  var syncObject = $firebaseObject(ref);
  // synchronize the object with a three-way data binding
  // click on `index.html` above to see it used in the DOM!
  syncObject.$bindTo($scope, "data");

}])
```

SYNCHRONIZE COLLECTIONS AS ARRAYS

Three-way data bindings are amazing for simple key / value data. However, there are many times when an array would be more practical, such as when managing a collection of messages. This is done using the **$firebaseArray** service.

We synchronize a list of messages into a **read-only array** by using the **$firebaseArray** service and then assigning the array to **$scope**

```
.controller('dashCtrl', ['$scope','$firebaseArray', function ($scope, $firebaseArray) {
    var ref = new Firebase("https://purechatapp.firebaseio.com/data");
    $scope.messages = $firebaseArray(ref);


}])/*
```

NG REPEAT DIRECTIVE

The ng-repeat directive repeats a set of HTML, a given number of times. The set of HTML will be repeated once per item in a collection. The collection must be an array or an object.

```
<h1 ng-repeat="dataset in messages">{{ dataset.text }}</h1>
```

USER AUTHENTICATION

Firebase provides a hosted authentication service which offers a completely client-side solution to account management and authentication. It supports anonymous authentication, email / password login, and login via several OAuth providers, including Facebook, GitHub, Google, and Twitter.

AngularFire provides a service named $firebaseAuth which wraps the authentication methods provided by the Firebase client library. It can be injected into any controller, service, or factory.

USER CREATION IN FIREBASE

createUser() Creates a new email / password user with the provided credentials. This method does not authenticate the user. After the account is created, the user may be authenticated with authWithPassword().

```
      ref.createUser({
 email: "pulkitkkr@gmail.com",
 password: "abcd1234"
, function(error, userData) {
 if (error) {
   switch (error.code) {
     case "EMAIL_TAKEN":
       console.log("The new user account cannot be created because the email is already in use.");
       break;
     case "INVALID_EMAIL":
       console.log("The specified email is not a valid email.");
       break;
     default:
       console.log("Error creating user:", error);
   }
 } else {
   console.log("Successfully created user account with uid:", userData.uid);
 }
);

 }
```

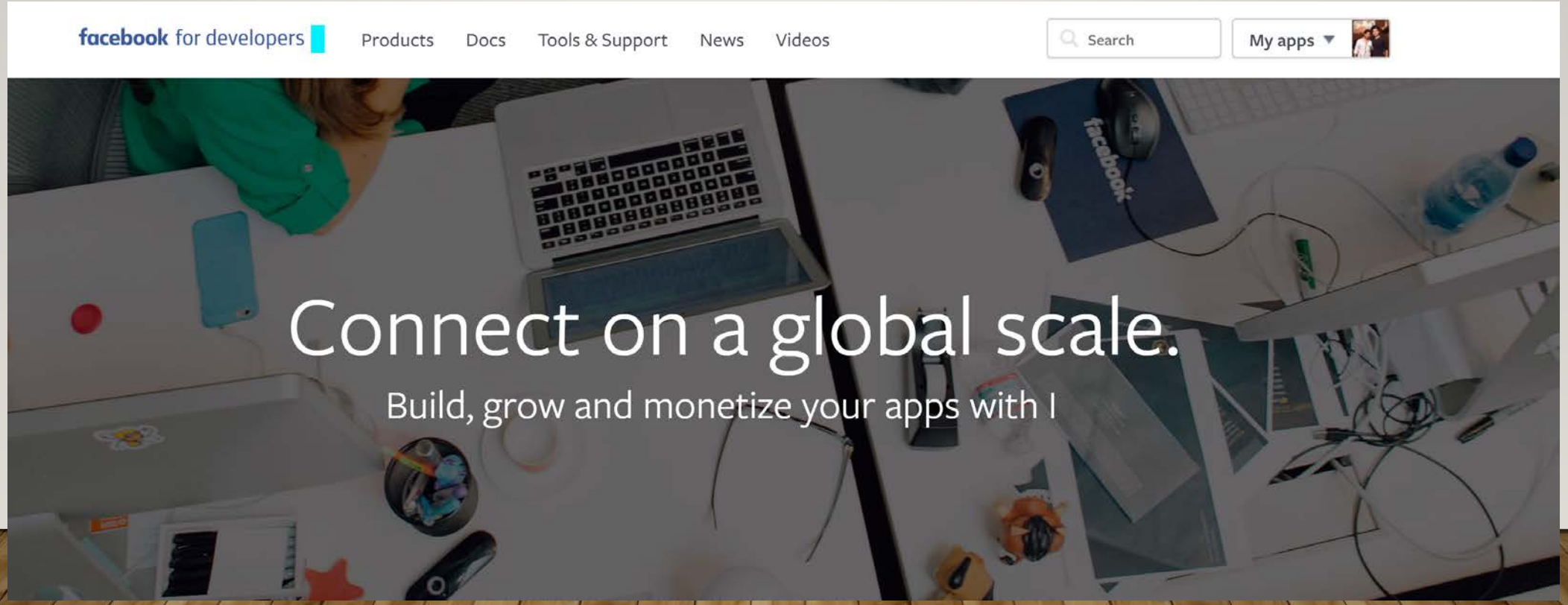
LOGGING IN USING ANGULARFIRE

authWithPassword() Authenticates a Firebase client using email / password credentials. The session will live until its configured expiration time in the Login & Auth tab of your App Dashboard, or when you explicitly end the session by calling unauth().

```
$scope.addMessage = function(){
    ref.authWithPassword({
        "email": "pulkitkkr@live.com",
        "password": "abcde12345"
    },function(error,authData){
            if(error){
                console.log("Login Failed!", error);
            }else{
                console.log("Authenticated successfully with payload:", authData);
            }
    });
};
```

LOGGING IN USING FACEBOOK

1) Create a facebook app at Facebook developer portal



FACEBOOK LOGIN

2) Add following lines of codes :-

```
$scope.addMessage = function () {

    auth.$authWithOAuthPopup("facebook").then(function(authData) {
  console.log("Logged in as:", authData);
}).catch(function(error) {
  console.log("Authentication failed:", error);
});
}
```

3) Configurations, domains settings, Oauth redirect URI

WRAPPING ALL UP

Pulkit Kakkar

(pulkitkkr@gmail.com)

https://www.facebook.com/pkakkar1

https://www.instagram.com/pulkitkkr001/