Pipes:

* We can transform data displayed without changed the actual data with the help of pipes
* Angular offers a couple of pipe functions we can use. For example, the uppercase and the date pipe.
* Pipes are added using the | symbol after the binding. We can also configured to display in different formats. For example, we want to change the date format. We can do this by adding a : after the pipe function and then entering the parameter that determines the format.
* Multiple pipes are applied in the order they are mentioned in the view

Custom Pipes:

* A pipe is essentially a class that is exported which contains functions that transform the data provided. We start with implementing the PipeTransform interface.
* We then implement in the transform function provided by it. It takes the value and arguments as parameters. It returns the transformed data.
* To make angular know that it is a Pipe, we use the @Pipe decorator, which specifies the name of the pipe. The pipe is made available through the app by adding it to the declarations array in the app module.
* The decorator also has a second property, the pure property. This determines whether the pipe needs to be reapplied on change of data in the case of arrays and objects. Changing its value to false will allow this.

Asynchronous Pipes:

* This is a super useful pipe that basically allows us to transform data based on its availability. Asynchronous tasks are common these days that can be handled efficiently in this manner. The HTTP module will take advantage of this.
* We can use this pipe by piping async to the relevant value.

HTTP Requests:

* We use HTTP requests to get data from a server. Most of the communication to the database is done from the server due to security concerns.
* To enable the use of HTTP requests in Angular, we have to import the HTTPClientModule from @angular/common/http.
* We will be using [Firebase](https://firebase.google.com) as our backend since its free and not very difficult to use. We will be using a real-time database. Since Firebase already has an API in place to send and get data from the backend, we just have to use them in our app.
* We should never communicate with our database directly. It is a HUGE security flaw.
* Since Firebase stores data in json format, creation of data also should follow the same format. For example, if we want to create a posts json object in our data base, we sent a post request in this fashion - <URL\_TO\_FIREBASE\_API>/post.json.
* Angular is quite smart when it comes to dealing with HTTP requests. With of rxjs, we know we can use Observables to subscribe to any data retrieval or change that occurs asynchronously.
* In our case, if no subscribes to get the response of our HTTP request, then it will not send the request at all.
* All requests made using the HTTPClientModule are using Observables which are then subscribed to. We also use pipe() to transform the data using the operators provided from rxjs/operators.
* To assist typescript and enable auto-completion of code, we need to specify the type of data we are going get or send. We can either do this at each parameter level or , more efficiently, at the HTTP method level.
* All the methods exposed by the HTTP module can have a generic type, which describes the kind of data it is sending or getting. It always good practice to mention this to assist our development process and avoid typescript errors.

Error Handling:

* When we subscribe to an HTTP request observable, we get 2 functions in response - The data response and the Error response.
* We can use this error response to dynamically show an error message based on type of error received. However, this is just one way of dealing with HTTP errors.