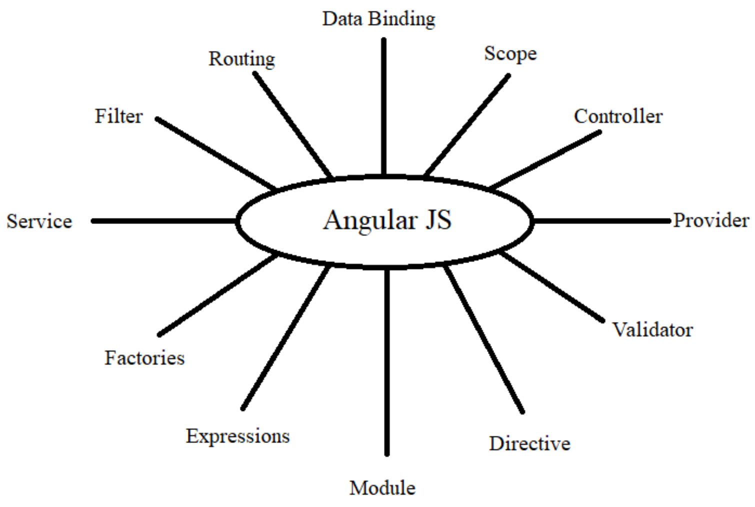
Q1

What is AngularJS is a structural framework for dynamic web applications. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application components clearly and succinctly. Its data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology. It was originally developed by Misko Hevery and Adam Abrons.

HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web-applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.



Architectural view of Angular JS

- **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 XML Http Requests. 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.

- **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 inthe 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 you 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.

How Angular JS work

Following is the simple Example of Angular JS with source code

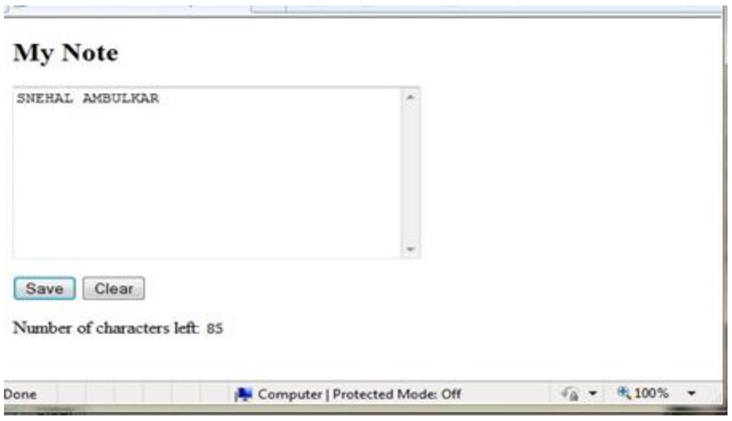
```
    <html ng-app="myNoteApp">
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/ 1.4.8/angular.min.js">

    </script>
<body>

    <div ng-controller="myNoteCtrl">
<h2>MyNote</h2>
<p><textarea ng-

model="message" cols="40" rows="10"></textarea></p>
<p>
<button ng-click="save()">Save</button>
<button ng-click="clear()">Clear</button>
</p>
<p>Number of characters left: <span ng-bind="left()">
    </span>
    </p>
</div>
```

Output:



My Note

```
SNEHAL AMBULKAR
```

Number of characters left: 85

Q2

What is MEAN product stack?

A relatively new stack, **MEAN** stands for **M**ongoDB, **E**xpress.js, **A**ngularJS, and **N**ode.js. MEAN is an end-to-end JavaScript stack largely used for cloud-ready applications. Understanding why you might use it, identifying examples of when to employ it, and diving deeper into the individual components can help you maximize the value of MEAN for software development. The MEAN stack is highly favorable in future programming with full-stack JavaScript development where technology drives easier and simpler way to build the feature-rich applications. It is simpler, reliable, and flexible for the dynamic applications.

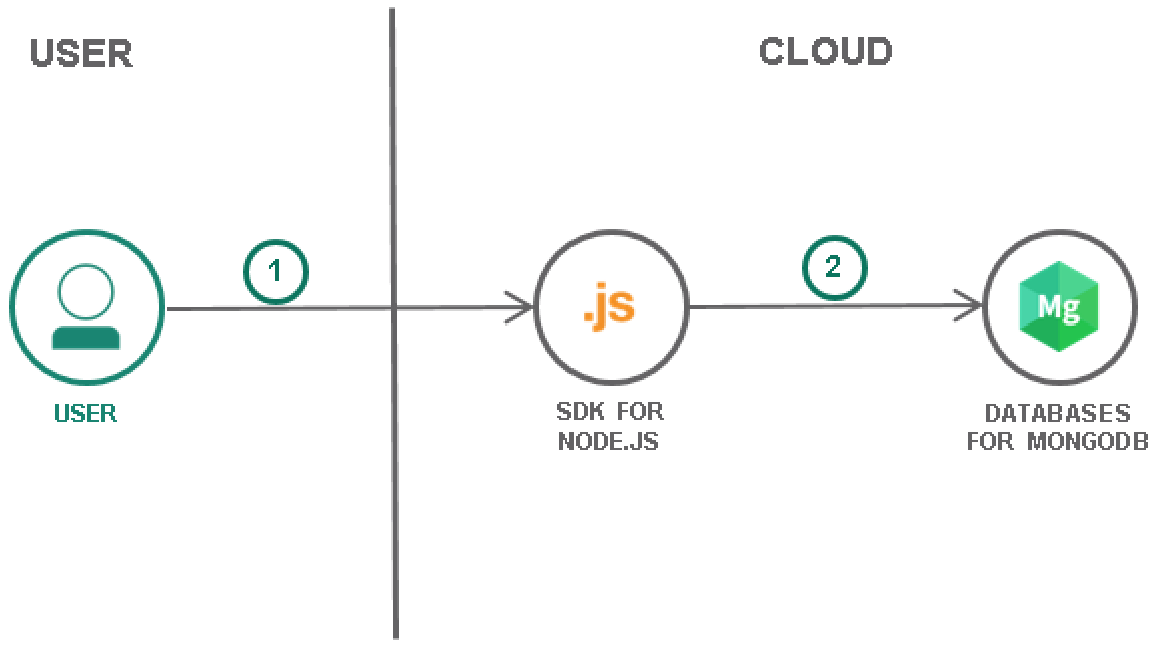
Few benefits of MEAN stack:

- Extensive library and great support for developer community.
- Free, open-source, and reliable framework for web and app development.
- Highly efficient and robust performance with 4 power-packed technologies.
- Code-reusability and flexible environment.
- A perfect tool for full-stack web development with front-end and back-end data processing.
- End-to-end development processes based on JavaScript language.

Use cases

While the MEAN stack isn't perfect for every application, there are many uses where it excels. It's a strong choice for developing cloud-native applications because of its scalability and its ability to manage concurrent users. The AngularJS frontend framework also makes it ideal for developing single-page applications (SPAs) that serve all information and functionality on a single page. Here are a few examples for using MEAN:

- Calendars
- Expense tracking
- News aggregation sites
- Mapping and location finding



An example architecture for a Node.js runtime with MongoDB on a MEAN stack.

Components

MongoDB

MongoDB is an open source, NoSQL database designed for cloud applications. It uses object-oriented organization instead of a relational model.

In the MEAN stack, MongoDB stores the application's data. Because both the application and the database use JavaScript, there's no need to translate the object as it journeys from the application to the database and back. The application can push and pull objects between the back end and the database without missing a beat.

MongoDB is touted for its scalability in both storage and performance. You can add fields to the database without reloading the entire table, and MongoDB is well known for its ability to manage large amounts of data without compromising on data access. With just a few clicks, you can expand the resources available to your database, making it perfect for applications with occasional periods of increased activity.

Express

Express is a web application framework for Node.js. It balances ease of use and a full feature set.

Forming the backend of the MEAN stack, Express handles all the interactions between the frontend and the database, ensuring a smooth transfer of data to the end user. It's designed to be used with Node.js and so continues the consistent use of JavaScript throughout the stack.

Express is minimalist—it's designed to efficiently handle processes without cluttering your application. But don't confuse minimalist with featureless. Express offers excellent error handling and templating functionality to aid your development.

Express can also protect you from yourself because it uses the CommonJS module standard to prevent inadvertent overwriting of variables within the shared namespace. You can't accidentally redefine a variable that you previously created. This enforcement of JavaScript closures can help prevent a time-consuming and costly error.

AngularJS

AngularJS—Google's JavaScript frontend framework—isn't the only frontend framework in use, but it's exceedingly popular. It is effectively the default for frontend JavaScript development. If you're developing a web application in JavaScript, you're using AngularJS.

The MEAN stack includes AngularJS to help developers build the user-facing side of the application. Because the backend, frontend and database are all built on JavaScript, there's a smooth flow of information between all parts of your application.

AngularJS didn't become the most popular JavaScript frontend framework by mistake. Its ability to simultaneously develop for desktop and mobile use, its well-tuned performance and its easy-to-use templates make it the ideal front end to build cloud-native applications.

Node.js

Node.js is an open source JavaScript framework that uses asynchronous events to process multiple connections simultaneously. It is an ideal framework for a cloud-based application, as it can effortlessly scale requests on demand. You're likely to find Node.js behind most well-known web presences.

Node.js is the backbone of the MEAN stack. Express is purpose-built to work on top of Node.js, and AngularJS connects seamlessly to Node.js for fast data serving. Node.js comes complete with an integrated web server, making it easy to deploy your MongoDB database and application to the cloud.

The greatest strength of Node.js is its scalability. Cloud applications are best when they can respond quickly to usage spikes. What good is virtually unlimited processing power if it's only available after your users time out? By expanding your resources as they're needed, you're able to serve more users while the framework's single-thread architecture allows the application to effectively provide a smooth user experience across numerous connections. Node.js can support as many as a million simultaneous connections.

Remember, Node.js works best with many low-resource requests as opposed to resource-intensive requests. While a single thread protects against process deadlocks, it's not immune to a large process freezing the system for all clients.