Ref: <https://tm.udemy.com/angular-2-and-nodejs-the-practical-guide/learn/lecture/13914132?start=0#overview>

**What is Mean?**

Mean comprises a set of 4 technologies, and this is what it stands for:

**M :** Mongo DB – this is database

**E :** Express JS – this is framework for Node JS – makes easier to work with Node JS

**A :** Angular - this client side technology – JS framework – to showcase interactive, r eactive UI

**N :** Node – its JS runtime server side language

**What is Angular?**

A Client-Side (Browser) Framework which allows you to build SPA, it handles the entire FRONT END LOGIC.

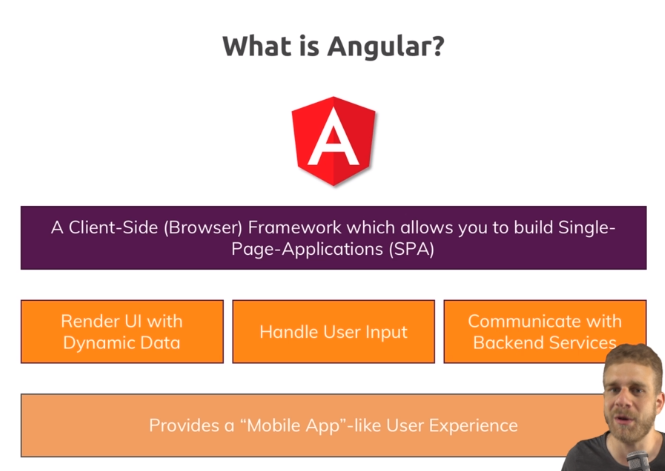
Angular's job is to render the user interface with dynamic data and that dynamic data part is important of course.

It's job is not just to render some static HTML and some CSS, we wouldn't need JavaScript for that, it's job is to update the UI whenever we have new information. Let's say, when you create a new post, we want to immediately update the UI to show that new post there.

It is also responsible for handling user input, so validating that user input and also sending it to the server because that's the third part it does.

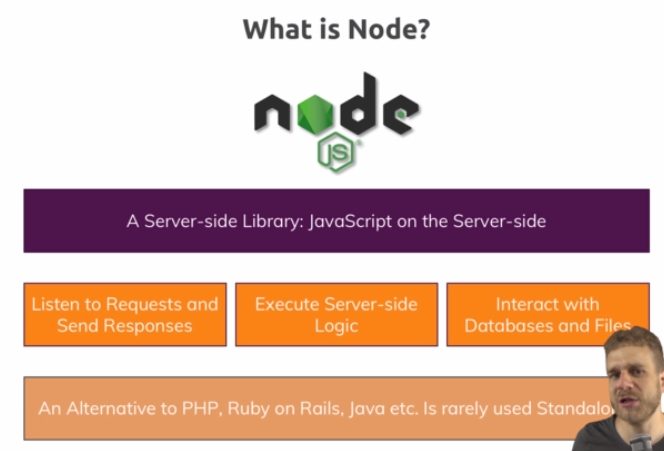
It communicates with our backend, so with that Node, Express and MongoDB combination that seems to be responsible for that.

Angular provides a mobile-app-like user experience because since we handle everything in the browser through JavaScript, we never need to reload the page, we just change parts of the page with JavaScript through Angular and therefore everything happens instantly in a very reactive way and that is the feeling we know from mobile apps.



Now that is still only the front-end though.

**What is Node?**



NodeJS is a server-side library, JavaScript runtime and you know that Javascript can run in the browser, well NodeJS simply takes it, adds some things that are useful on the server like working with files, working with HTTP requests and now we can use JavaScript on the server too, pretty amazing.

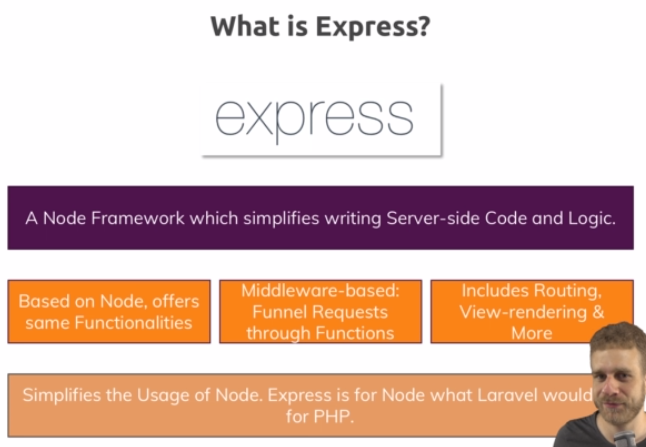
NodeJS listens to incoming requests and is able to send back responses, for example our Angular app could send a request to fetch a list of all the posts and Node could handle that request, do something, reach out to the database and send back a response with all these posts.

NodeJS executes server-side logic in general, so authentication, anything we don't want to run in the browser for security or performance reasons and NodeJS is also able to interact with databases and files. Angular can't do this, it doesnt have access to any file system and especially not to a file system on some remote machine on a server and whilst theoretically, you could connect to a database from Angular, since all of your client-side JavaScript code is visible to the user, you can have a look at it in your developer tools of your browser, it would be very insecure to connect to your database because you would expose all your credentials and everything, you don't want to do that.

Therefore, NodeJS is an alternative to PHP if you know that or Ruby on Rails, ASP.net, things like that and it's rarely used standalone, just like these languages, you typically use a framework along with it and that framework you typically use is Express.

**What is Express?**

**Express is a NodeJS framework, so it still uses NodeJS, the same language but it adds a lot of utility features. So it offers additional functionalities or in general, it makes things easier.**



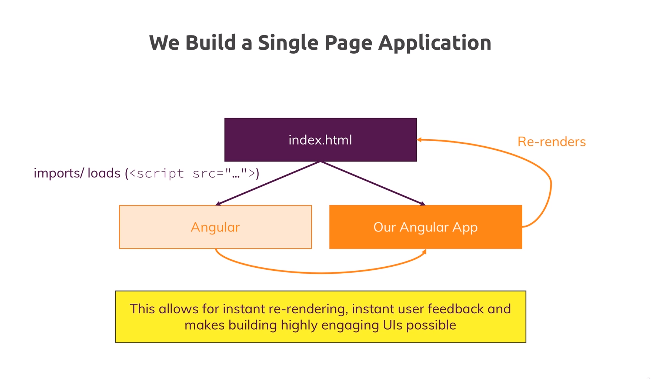
ExpressJS is middleware-based and we'll see what this means once we dive into the code. It basically funnels incoming requests through a chain of middlewares, of steps where we can do something with the request, read some data from it, manipulate it, check if the user is authenticated or basically send back a response immediately, this chain allows us to write very structured code and you will learn everything about it in this course of course.

And last but not least, it includes routing, which means that we could render views, so HTML pages with it, we're not going to do this in this course though because Angular should handle our entire front-end and again this is also something we'll see but more importantly, we can handle different requests to different endpoints which will be important for connecting Angular to the backend because if we want to fetch a list of posts, if we want to send some requests to /posts, so our-domain/posts, if we want to create a new post, we want to send a different request to our-domain/posts for example and ExpressJS allows us to implement this routing logic so that different requests to different URLs are handled correctly.

Again, this will all be implemented step-by-step throughout the course.

So to sum it up, ExpressJS simplifies the usage of NodeJS, it is a tool we definitely want to use.

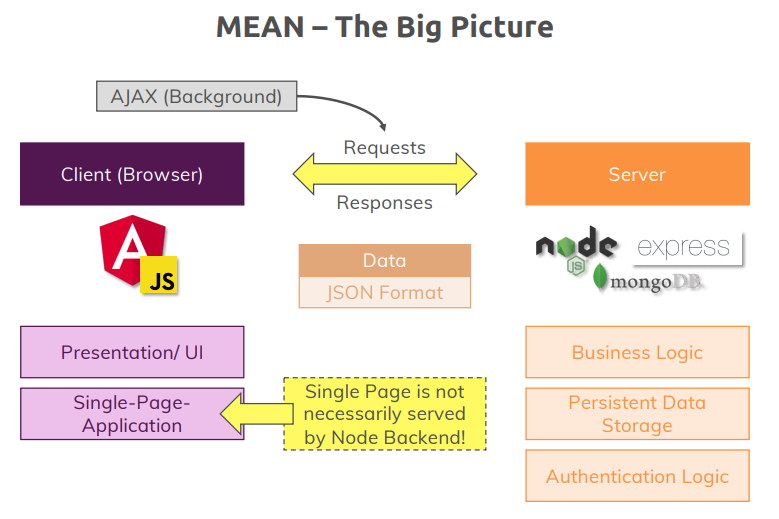
**What is Single Page Application?**



IN SPA we never need to reload the page just because the user maybe clicked on a post and want to see the details. We can instead navigate to that page directly because we don't really leave the page, we just remove some elements from the DOM and add new elements and all of that is handled for us by the Angular framework, it's really convenient to use and to work with the DOM with it.

And therefore we have a powerful way of immediately changing the page, maybe showing a spinner whilst we're fetching some data behind the scenes, so that list of posts which we probably still need to get but we will do that behind the scenes and this provides a highly-interactive, mobile-app-like feeling, a very responsive and fast web page where we never have to wait, where things always happen and that of course is a great user experience and this is why we'll use JavaScript and Angular therefore for the entire front-end, for the entire user interface and we will use Node, Express and MongoDB as a back-end to which we reach out behind the scenes to fetch and send data but the whole user interface is handled as one page only which is dynamically re-rendered all the time by Angular.

**MEAN – The Big Picture?**



With that, let's have a look at the big picture. How does the entire MEAN stack look like in this course or in general, not just in this course? We have the **client-side** and the **server-side**, that's important. The client-side is what the user sees, the webpage as it runs in the browser, server-side is somewhere on a server we deploy where we run our business logic and which the user only indirectly accesses, we'll see how that access works in a second.

So on the client, we use Angular which is a JavaScript framework so we also use JavaScript implicitly and we use it to build that user interface. On the server-side, we use Node, Express and MongoDB, Node/Express for the logic and MongoDB for the database, as I said, you don't want to connect directly to it from Angular. Now the client-side Angular is responsible for the user interface, so for the presentation.

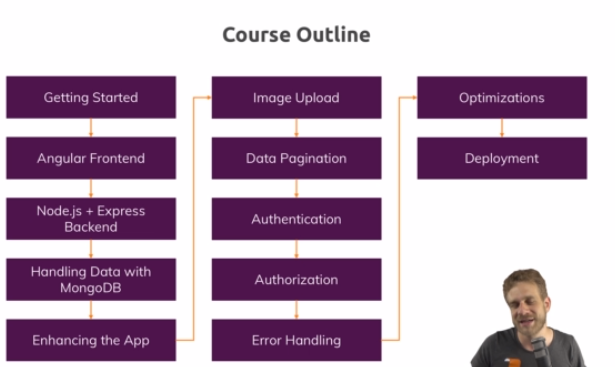
It's a Single Page Application as I explained in the last lectures and that Single Page Application can be rendered by our Node backend, so we could have one route which essentially returns that single HTML page but we can also be totally decoupled from that and be served from a totally different host, some static host like AWS S3 for example.

Now on the Node application, we have our core business logic, especially the logic that should not be exposed to the client due to security reasons or performance reasons. We have our persistent data storage, so that database and we also put our authentication logic there, for example, it's of course part of our business logic, I just want to really emphasize that here, the logic where we decide whether an email or password is valid happens on the server because it can easily be fiddled with on the client, not so much on the server.

How do we connect the two pieces then?

Well we exchange requests and responses and these requests and responses are sent behind the scenes, so-called Ajax requests, you might have heard of this before and we use exactly the same pattern in Angular. These are requests which can be sent without us needing to reload the page which is of course exactly what we want. Therefore the type of data we exchange is not HTML because we never want HTML code, we do all that presentation and re-rendering logic with Angular. Instead what we get is so-called JSON data, that's a data format that's really efficient for encoding data like a list of posts and you will see how it looks like in this course.

This is the big picture, this is how the MEAN stack works and this is exactly what we will implement in this course. So with that, I'd say enough of the words, let's get started and let's start setting up the base development environment we'll work with in this course and let's start building our MEAN stack application.



**Angular Frontend Basics**

**Module Introduction**

**Understanding the Folder Structure**

**Understanding the Angular Components**

Essentially you compose an entire page of these components, you build it with these components because the advantage of this is that you have small, easily to maintain and manage building blocks for your UI which you can even reuse because some components appear more than once on a page.

ngModel – 2 way data binding:

ngModel is an angular feature, it's a so-called directive. A directive is basically an instruction you place on an html element and angular or you can also create your own one so that the relative knows what to do on that element then. ngModel is a directive that actually will listen to user input and emit that data to us and also store new data in that text area or output it there.

Now important, ngModel by default won't work, it's a feature which is not included in the core angular package here, it's not included in the browser module which we already added, it's included in a different module which we need to add, it's included in the so-called forms module because it's related to forms inputs and this is part of @angular/forms, so that's still part of the core framework but of a different part of the framework.