Software Stacks

An introduction to software stacks and their purpose in web development.

We'll cover the following

- Definition
- History and LAMP stack
- MEANStack
- MERNStack
- Isomorphic frameworks vs. software stacks
- Test your understanding!

Definition

A **software stack** refers to a combination of tools that are used together to create software. We have already seen examples of this in the previous lesson, where many of the isomorphic frameworks we discussed used a combination of other frameworks or tools to provide the specific features it offers. These tools are inbuilt in these frameworks and are part of a nuance that users do not need to worry about when using the framework to develop their applications.

However, it is often the case that multiple frameworks need to be used together along with other tools as a stack to create a web application, and in this case, each one of them needs to be explicitly included and linked by users themselves. We have already studied both front end and back end frameworks in the previous lessons; now we will look into using them together as a software stack to develop whole web applications.

History and LAMP stack

LAMP Stack entails using Linux, Apache, MySQL, and PHP together to build an application. Linux is the operating system, Apache acts as the HTTP server,

MySQL provides the relational database to persist the application's information, and PHP is the programming language in which the application is built. LAMP was amongst the earliest software stacks that web applications were typically built using.

In recent times, however, Single Page Applications have grown in popularity in that they provide a more seamless user experience where lightweight server calls change what is rendered on the screen without having to refresh the entire page. Consequently, front-end frameworks are increasingly being used to achieve the benefits that SPAs pose. Similarly, NoSQL databases have also gained popularity due to the simplicity and efficiency they provide. As a result, modern software stacks have evolved to include front-end frameworks such as Angular and NoSQL databases such as MongoDB. The most common types include MEANStack and MERNStack, both of which we will be discussing in the following sections.

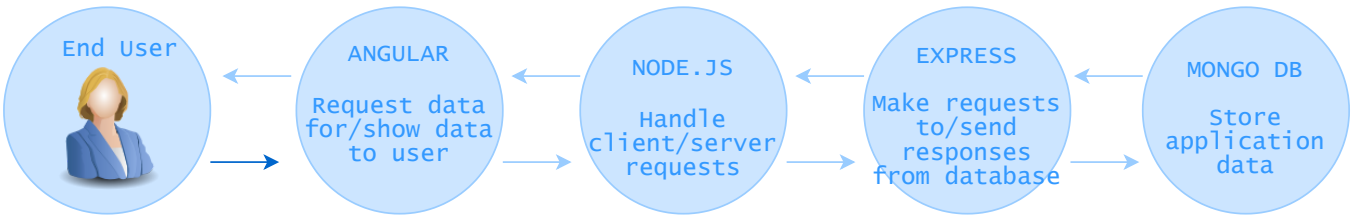
MEANStack

MEANStack refers to integrating the following frameworks and tools to create a web application:

1. MongoDB
2. Express JS
3. Angular JS
4. Node JS

We have discussed each of these at length in our earlier lessons, so we won't be going into too much detail here, but will briefly outline how they work together to form a web application and what role each component plays. MongoDB, as you might already have guessed, serves as the database and, thus, forms the storage component of the website. To put it in MVC terms, it acts as the Model component of the application. Express JS is used to develop the back-end of the web application and it is used to do everything from creating the server at the back-end to handling all kinds of requests coming in from the client end and processing data from the database accordingly. The express component, therefore, forms the controller of the MVC application. Angular JS, in this case, is used to develop the front-end of the web application, and it is basically used to program everything that happens

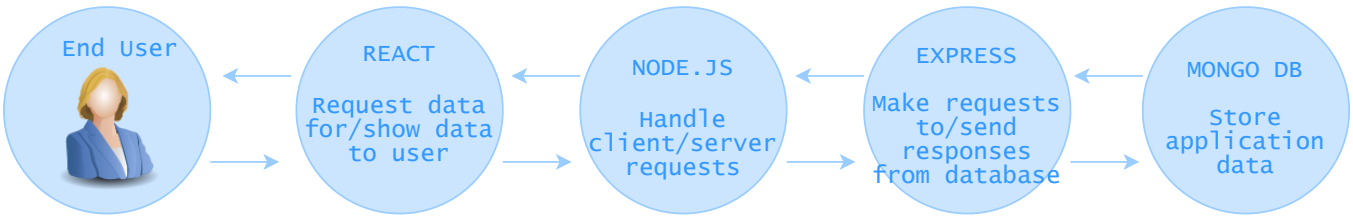
within the browser on the client-side. The AngularJS component is, thus, synonymous to the View component of your MVC application. Node js, in this case, refers to the back end runtime environment of the web application. Essentially, express is built on Node js, and its purpose is to simplify operations that may be complicated in the basic Node js code. The underlying runtime environment for the event-driven code that you will be writing for your application is, therefore, Node JS.



Structure of a MEANStack application

MERNStack

MERNStack applications have the same structure as MEANStack; a database, a server-side framework, and a client-side framework. The only difference, in this case, is that the front-end framework being used is different. The front-end of MERNStack applications is developed using React JS instead of Angular JS. There are no striking benefits of React compared to Angular or vice versa, so the software stack you select depends largely on the specific requirements of your application. Both React and Angular have been discussed in the lesson on front-end frameworks, and it is primarily a matter of preference when choosing between the two.



Structure of a MERNStack application

Isomorphic frameworks vs. software stacks

So far, we have discussed both software stacks that entail using multiple frameworks together to create a web application and isomorphic frameworks that already have multiple frameworks integrated within them for users to leverage the benefits of each without having to worry about putting the pieces

together. The question that then follows is obvious; *Which of the two is a better approach to web development?* The answer requires an analysis of the

differences between the two. Let's enumerate the key differences between two of the most popular frameworks of both types, Meteor JS and MEANStack, one by one:

- One key difference is simplicity. Meteor has a strong focus on ease of use and is relatively easy to become equipped with since it masks implementation level details of integrating the multiple frameworks it uses and allows users to focus on development. MEAN, on the other hand, requires multiple components to be integrated manually, each of which needs to be learned in detail before the development can begin.
- Extending on the first point, Meteor is a *full stack framework*, which means that it has an inbuilt database, as well as the capabilities of both a server-side and a client-side framework. MEANStack, on the other hand, requires each component of the stack to be installed, learned, and integrated manually. This makes it clear that Meteor is a much more convenient option.
- MEANStack is more flexible since the developer has to put all the components together, and it is, therefore, possible to swap out some of the components with alternate ones. On the other hand, Meteor has a stringent limitation on what technologies are being used since the underlying technologies are not of concern to the user. This means that for applications that have highly customized specifications, MEANStack is the more appropriate development model.
- There is a concept of reactivity in Meteor, which ensures that whenever there is a change in data, clients connected to the Meteor server get updated automatically without having to refresh. This is not the case in MEANStack, which requires additional technologies to be integrated to achieve this.

Test your understanding!

Quiz on Software Stacks!

1 What is the purpose of Node JS in MEANStack?

A) It serves as the server-side run time environment

B) It is used on top of a separate framework to make requests for data

C) It is used to develop the front-end of the application

COMPLETED 0%

1 of 4

That concludes the discussion on frameworks, how they can be combined to form software stacks, and how these stacks compare to isomorphic frameworks. As is apparent from our discussions so far, the choice of framework depends largely on the requirements of the application being developed. We have articulated the pros and cons of each one so you can make the most informed decision. Now that we have covered all of our frameworks, in the next lesson, we will build a MERNStack application to depict how frameworks are used together to develop websites.