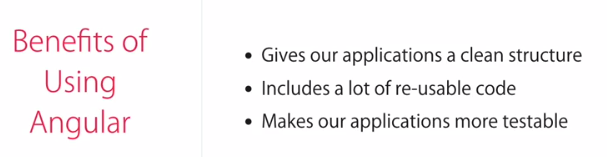
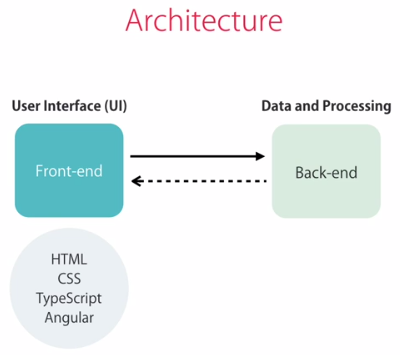
**What is Angular?**

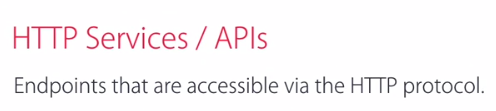








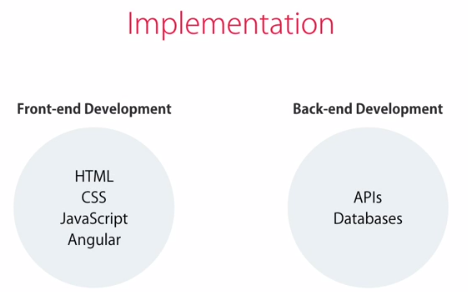
Well, you don't, in most cases you don't save the data on the client, because it can easily disappear as the user clears the browser data, or moves to a different computer. That's why we store the data on the server. So here we often have one or more databases, as well as a bunch of HTTP services or API's to make this data available to the clients. In case you don't know, API stands for **Application Programming Interface**.



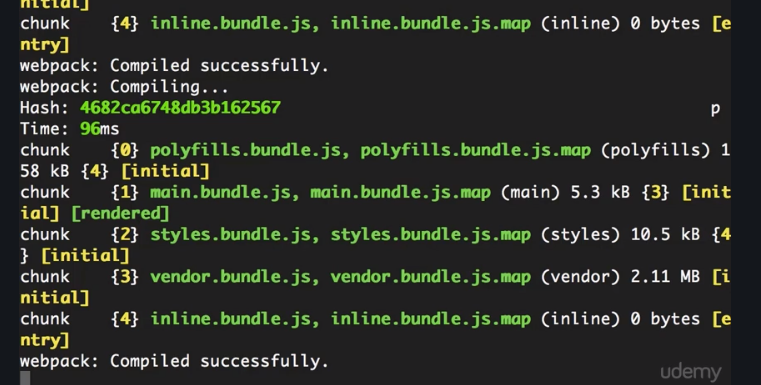


Now if you put this fancy name aside, these HTTP Services, or API's are essentially end points that are accessible via the HTTP protocol. So we can call them using simple HTTP requests, to get or to save the data.

Our web applications have a front-end and a back-end. On the back-end we have more than one databases, and API is what makes the data available to the clients. For a large application, around a complex domain, this is where we implement the business logic of the application, by calculating the tax and shipping costs, based on various parameters. The front-end or the client on the other hand, is all about the presentation. So here we have our HTML templates, As well as the presentation logic of our application. This lodging is all about displaying data and responding to user actions like what should happen when the user clicks on a button or navigates away from a page. These are examples of the presentation logic, or the logic that is all about the presentation aspect of an application.



webpack: Compliing...So Angular



**Webpack** - this is a build automation tool. It gets all of our scripts and style sheets combines them, puts them in a bundle, and then minifies that bundle, and this is for optimization. So here you can see we have a few bundles, like **poyfills**, which includes all of the scripts to fill the gap, between the version of the Javascript that Angular needs, and the version of Javascript supported by most browsers out there.

You have **main.bundle** which is all the source code of our application, you have **styles.bundle** which includes all of our style sheets. And note that here style sheets are actually stored in the Javascript bundle. You're going to see that in a second. We also have this **vendor.bundle**, which includes all the third party libraries.

**Whenever you change one of your files, where that file being a style sheet, a typescript file, or HTML file, web pack automatically recompiles your application and refreshes your bundles.**

Whenerver you modify any source file then Webpack automatically refreshes your browser, so this is a feature of web pack called Hot Module Replacement or Hot Module Reloading.

what's the difference between AngularJS, Angular 2, and

**Angular 4, and how did we jump from Angular 2, to Angular 4?**

Well, AngularJS was introduced in 2010, as a JavaScript framework for building client applications. Soon it gained popularity, and Angular team started adding new features to the core, but the framework was not designed with the needs of today's applications in mind.

Plus, it was overly complex, so Angular team decided to rewrite the original framework using TypeScript, and as a result, Angular 2 came out in mid 2016. This new version is entirely different from Angular 1, to the extent that you can think of it as a completely different framework. Now, this made a lot of developers unhappy. Because they had a lot of applications built with Angular 1, with each application over a few thousand lines of code that had to be rewritten.

But over all, I personally like the direction that Angular team took, and how they rebuilt Angular with TypeScript. This is a much better framework, it's a lot cleaner, and it's a lot easier to understand and work with. Now after a few minor upgrades to Angular 2, something strange happened. We were up to Angular 2.3, And then all of a sudden, Angular 4 came out, so a lot of developers including myself, wondered **what happened to Angular 3?**

We thought we missed something really big here. But unlike Angular 2, Angular 4 was not a new framework with a lot of breaking changes, in fact it wasn't even a major upgrade.

So let me explain what happened, Angular consists of a few different libraries, that are distributed as separate Node packages by an NPM, for example, we have the core library that we use in every Application, we have Angular compiler, HTTP for calling HTTP services, we have Angular router for adding navigation to applications, and a few other libraries.

Now all these libraries were versioned the same, except the router library. So in order to align these versions and avoid confusion in the future, Angular team decided to go straight to an Angular version 4. So in a sense, Angular 4 is not a major upgrade to Angular 2, and you can think of it as Angular 2.4, as simple as that. Now, after all this confusion in the community about Angular versions, the team decided to drop the version suffix and simply call the framework Angular. So now we have two kinds of Angular, we have Angular JS, which is the first generation of Angular written in JavaScript, and it's going to die sooner or later. And we also have Angular, which refers to Angular 2 or later. In our discussions, in our articles, and whenever we want to talk about Angular 2 or later, we should really use the word Angular, like I'm an Angular developer, instead of I'm an Angular 2 developer.

Or, I'm an Angular 4 developer. See, it's really wierd. We should use versions only when we need to talk about this specific release. For example, when we want to upgrade Angular 4, to Angular 5, then everybody knows what is the scope of this upgrade and what are the potential breaking changes that can impact an application. So this is all about Angular version history.