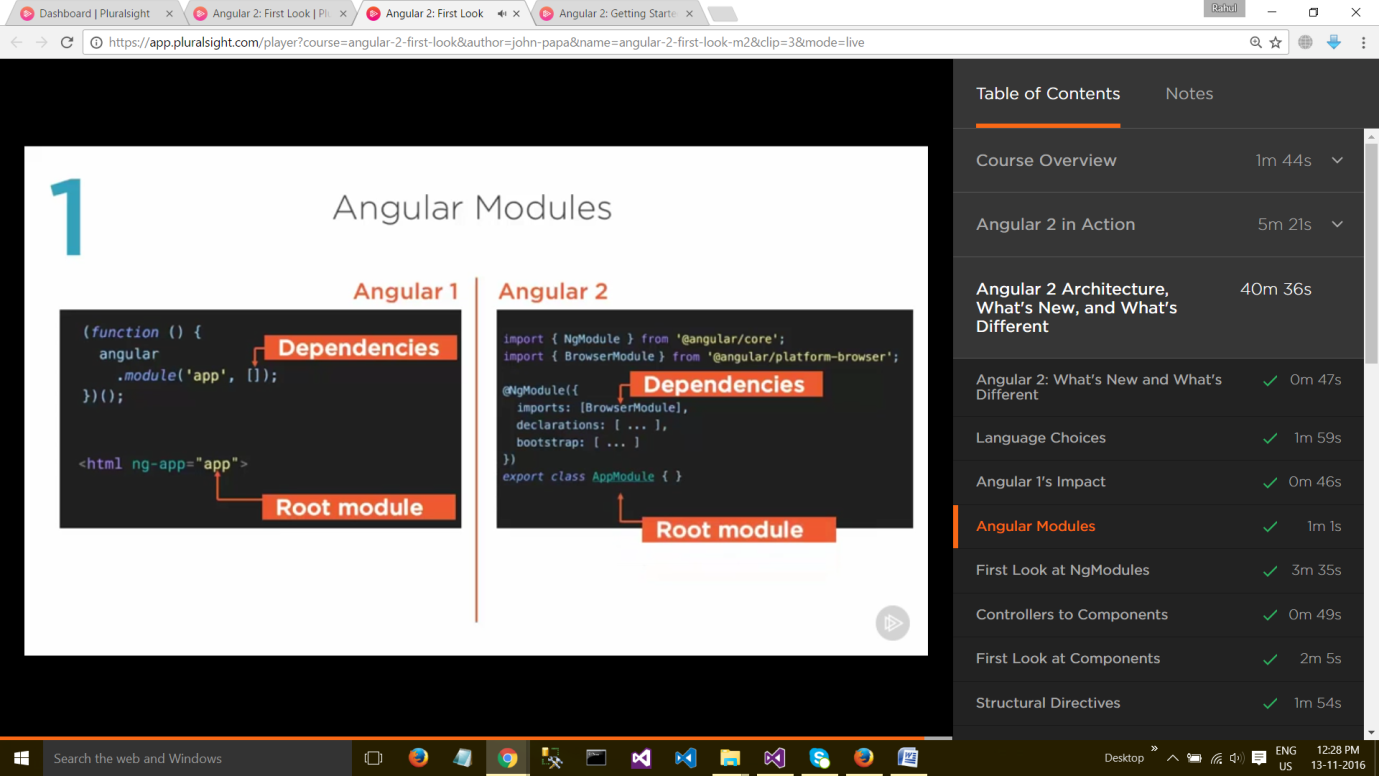
* In Angular2, Module concepts are different;
* We call then **Angular Modules**;
* Angular Modules are NgModules because we import them from Angular;
* We use them as decorator of a class;
* We define a class called AppModule and then we use a decorator to tell this module(class) where it's going to get its imports from;
* Effectively we are going to run this in a browser, so we import the Browser Module which Angular provides to us;



* **Angular Modules** help organize an application into cohesive blocks of functionality.
* An **Angular Module** is a **class** adorned with the **@NgModule** decorator function. @**NgModule** takes a **metadata** **object** that tells Angular how to **compile** and **run** **module** code. It identifies the module's own components, directives and pipes, making some of them public so external components can use them. It may add service providers to the application dependency injectors. And there are more options covered here.
* Many Angular libraries are modules (e.g, FormsModule, HttpModule, RouterModule). Many third party libraries are available as Angular modules (e.g.,[Material Design](https://material.angular.io/), [Ionic](http://ionicframework.com/), [AngularFire2](https://github.com/angular/angularfire2)).
* Angular modules consolidate components, directives and pipes into cohesive blocks of functionality, each focused on a feature area, application business domain, workflow, or common collection of utilities.
* Modules can also add services to the application. Such services might be internally-developed such as the application logger. They can come from outside sources such as the Angular router and Http client.
* An Angular module is a class decorated with @NgModule metadata. The metadata:
  + declare which components, directives and pipes *belong* to the module.
  + make some of those classes public so that other component templates can use them.
  + import other modules with the components, directives and pipes needed by the components in *this* module.
  + provide services at the application level that any application component can use.

Every Angular app has at least one module class, the root module. We bootstrap that module to launch the application.

The root module is all we need in a simple application with a few components. As the app grows, we refactor the root module into **feature modules** that represent collections of related functionality. We then import these modules into the root module.

=================================================================================

### Angular modules vs. JavaScript modules

The Angular module — a class decorated with @NgModule — is a fundamental feature of Angular.

JavaScript also has its own module system for managing collections of JavaScript objects. It's completely different and unrelated to the Angular module system.

In JavaScript each *file* is a module and all objects defined in the file belong to that module. The module declares some objects to be public by marking them with the export key word. Other JavaScript modules use *import statements* to access public objects from other modules.

=================================================================================

Reference:

https://angular.io/docs/ts/latest/guide/ngmodule.html