



Angular Course

In this course you will get a hands-on of the major features of Angular

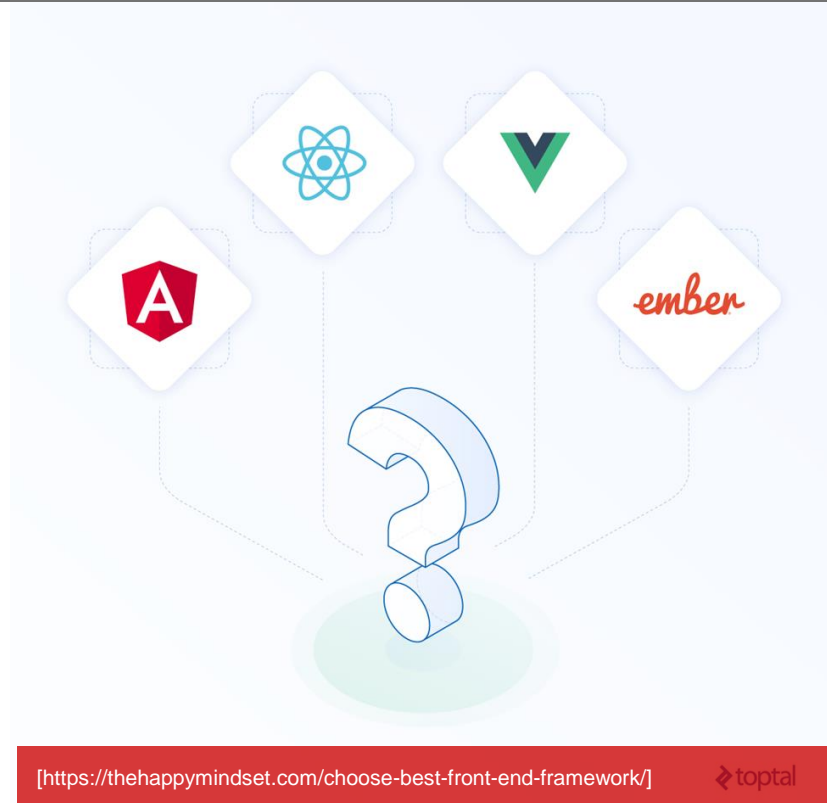
David Oliveira

Senior Consultant

23th January, 2018

Why Angular?

- Availability of Learning Resources
- Popularity
- Core Features
- Usability
- Ease of Integration (with Other Libraries)
- Outputs: Web, Mobile, Desktop



- Components
- Forms
- Directives
- Pipes
- Services
- Routing
- Observables



- App_Initializer
- Redux in Angular
- Unit Testing
- The mechanics of DOM updates in Angular (View encapsulation)
- Analyze an Angular App's Bundle Size
- Etc.



```
function doSomething(someData)
{
    // do something
}
```

JS
ES6

TypeScript

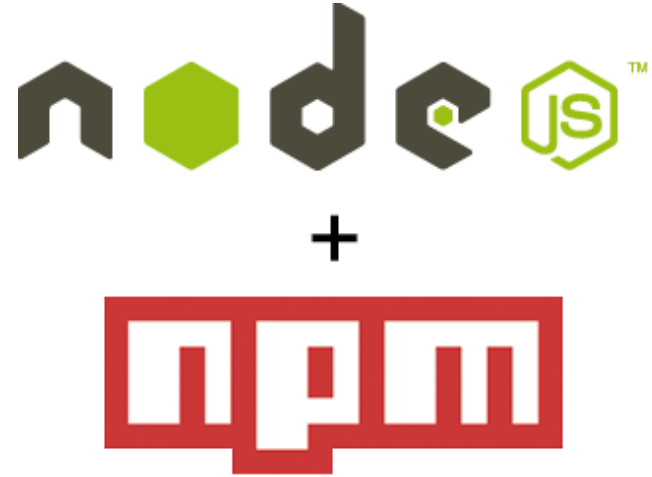
Type and Properties of someData?

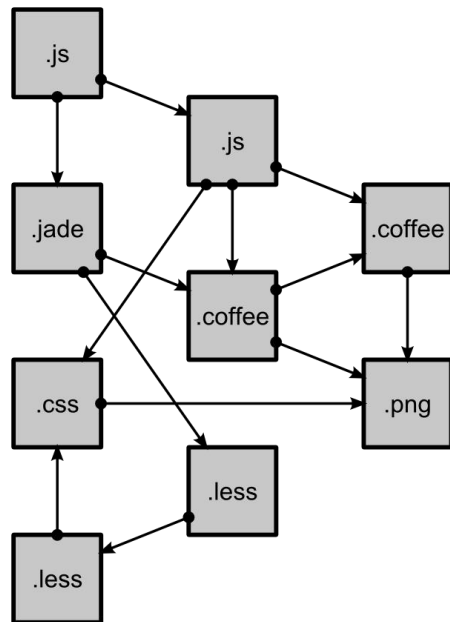
Node JS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

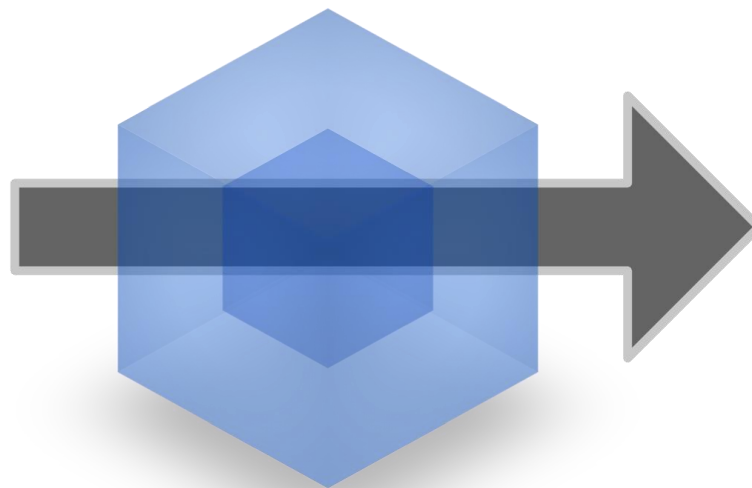
NPM

Node.js' package ecosystem, [npm](https://www.npmjs.com/), is the largest ecosystem of open source libraries in the world.

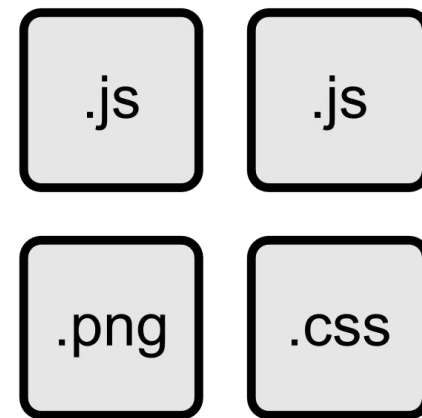




modules
with dependencies



webpack
MODULE BUNDLER



static
assets

The Angular CLI is a tool to initialize, develop, scaffold and maintain Angular applications

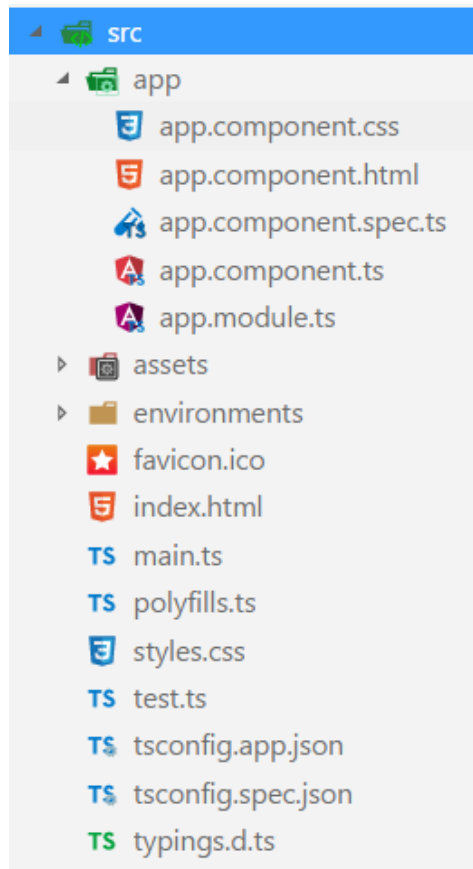
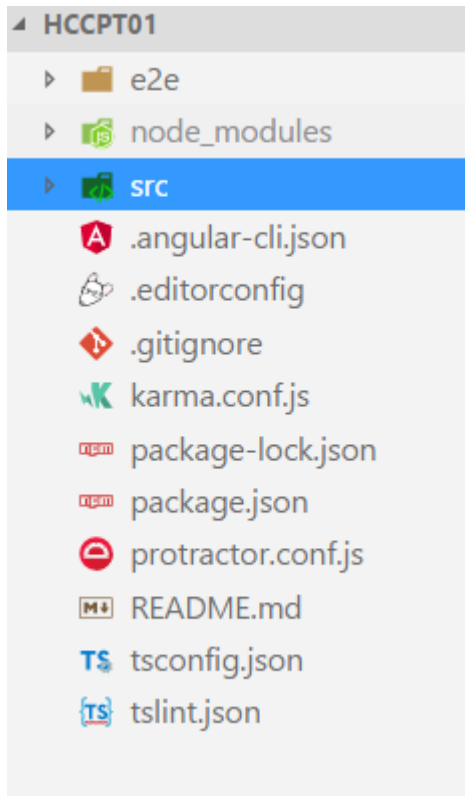
```
npm install -g @angular/cli
```

```
ng new hccpt  
cd hccpt  
ng serve
```

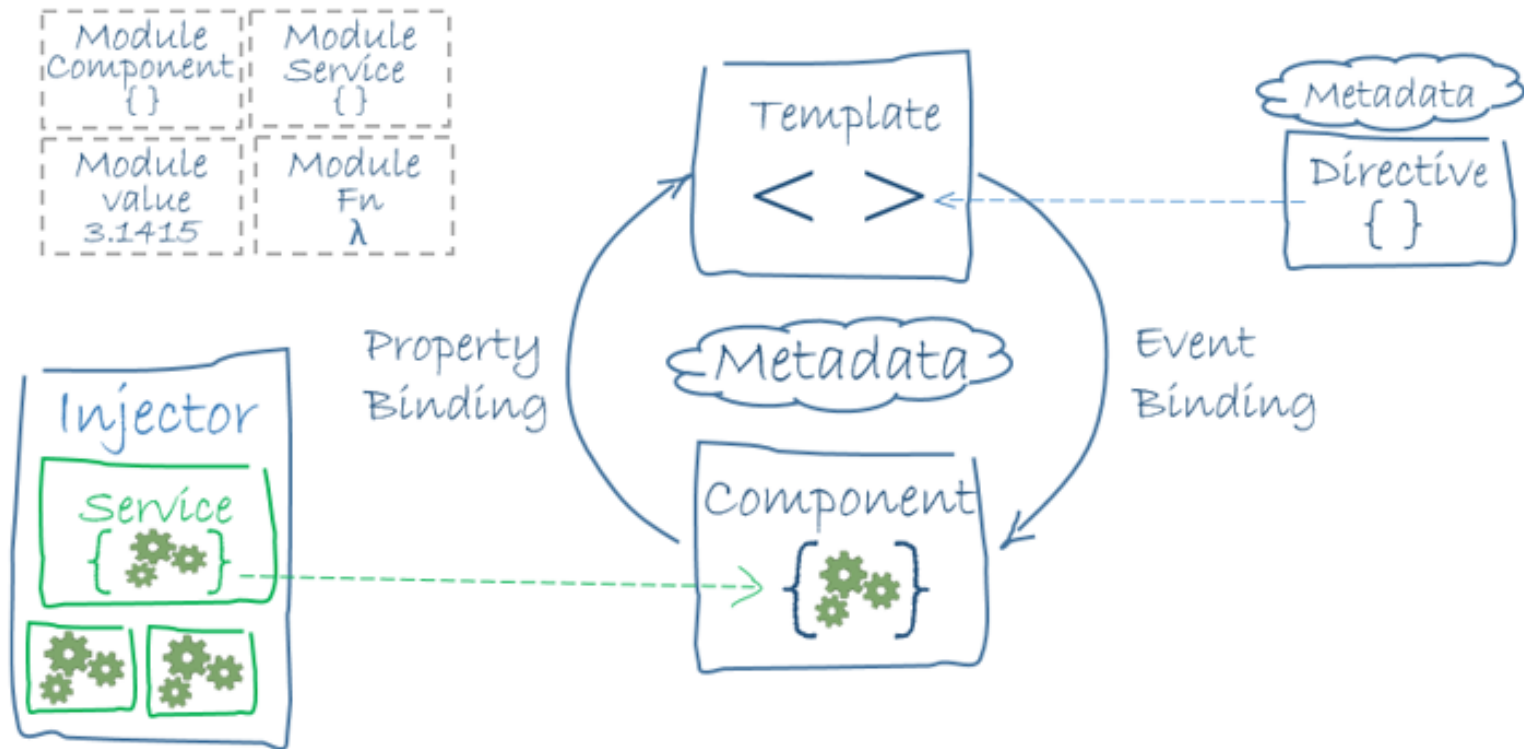
Navigate to <http://localhost:4200/>.

The app will automatically reload if you change any of the source files.

Angular Solution File Structure



Angular Architecture Overview





Angular Course

In this course you will get a hands-on of the major features of Angular

David Oliveira

Senior Consultant

30th January, 2018

Our App to build with Angular

We will build an application to manage a Condominium Administration with the following features:

- Add/Edit Contacts (Owners and Suppliers)
- Add/Edit Payments
- Dashboard with Charts about Payments



<https://github.com/davidoliveira/AngularCourse>

```
ng new KondominioApp --routing --style=scss
```

```
cd KondominioApp
```

```
npm install bootstrap@next --save
```

```
npm install --save @ng-bootstrap/ng-bootstrap
```

```
ng serve
```

<https://github.com/angular/angular-cli/wiki/generate>

<https://github.com/angular/angular-cli/wiki/stories>

<https://angular.io/guide/ngmodules>

<https://angular.io/guide/ngmodule-vs-jsmodule>

```
@NgModule({  
  declarations: [  
    AppComponent  
  ],  
  imports: [  
    BrowserModule,  
    AppRoutingModule,  
    NgbModule.forRoot()  
  ],  
  providers: [],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```

declarations—this application's lone component.

imports—import [BrowserModule](#) to have browser specific services such as DOM rendering, sanitization, and location.

providers—the service providers.

bootstrap—the *root* component that Angular creates and inserts into the index.html host web page.

A declarable can only belong to one module, so only declare it in one `@NgModule`. When you need it elsewhere, import the module that has the declarable you need in it.

ng g component Login

ng g module Contacts --routing

- ng g component Index
- ng g component Detail

ng g module Payments --routing

- ng g component Index
- ng g component Detail

<https://angular.io/guide/router>

<https://angular.io/guide/lazy-loading-ngmodules>

```
const routes: Routes = [  
  { path: 'login', component: LoginComponent },  
  {  
    path: 'contacts',  
    loadChildren: 'app/contacts/contacts.module#ContactsModule'  
  },  
  {  
    path: 'payments',  
    loadChildren: 'app/payments/payments.module#PaymentsModule'  
  }  
];  
  
@NgModule({  
  imports: [RouterModule.forRoot(routes)],  
  exports: [RouterModule]  
})  
  
export class AppRoutingModule { }
```




Angular Course

In this course you will get a hands-on of the major features of Angular

David Oliveira

Senior Consultant

5th February, 2018

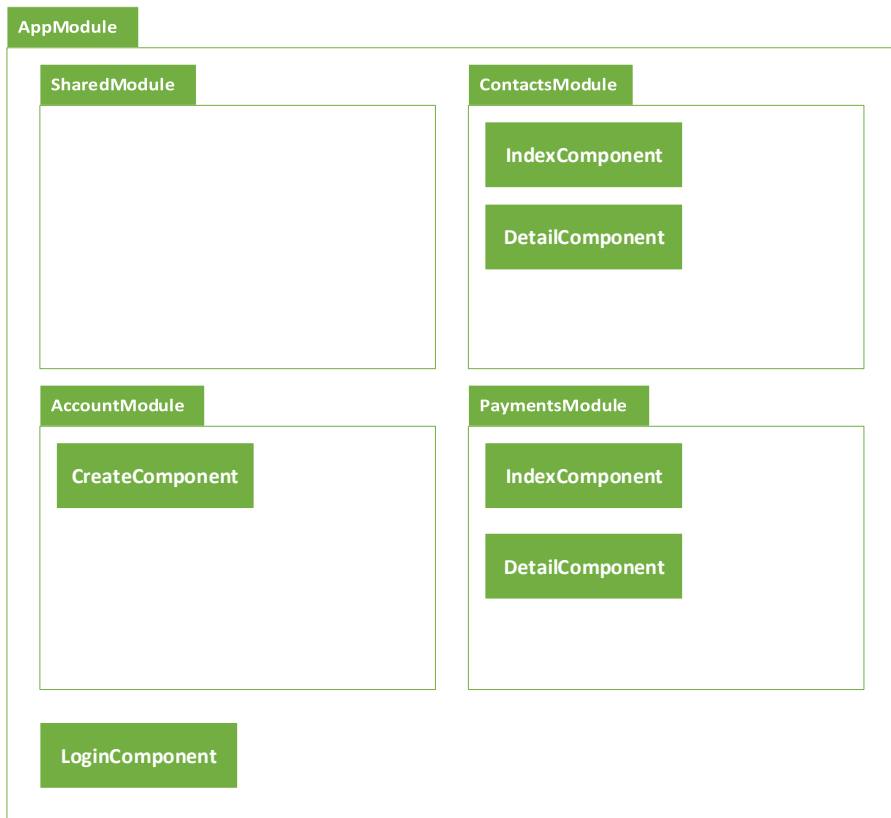
Last Week

- Setup our App to manage a Condominium
- Add Modules and Components for each Module
- Setup Routing

Angular CLI

Scaffold	Usage
Component	<code>ng g component my-new-component</code>
Directive	<code>ng g directive my-new-directive</code>
Pipe	<code>ng g pipe my-new-pipe</code>
Service	<code>ng g service my-new-service</code>
Class	<code>ng g class my-new-class</code>
Guard	<code>ng g guard my-new-guard</code>
Interface	<code>ng g interface my-new-interface</code>
Enum	<code>ng g enum my-new-enum</code>
Module	<code>ng g module my-module</code>

Our App Structure



```
const routes: Routes = [
  { path: 'login', component: LoginComponent },
  {
    path: 'account',
    loadChildren: 'app/account/account.module#AccountModule'
  },
  {
    path: 'contacts',
    loadChildren: 'app/contacts/contacts.module#ContactsModule'
  },
  {
    path: 'payments',
    loadChildren: 'app/payments/payments.module#PaymentsModule'
  }
];
```

Routing configuration to navigate through our application.
Check Angular documentation about it:

<https://angular.io/guide/feature-modules>
<https://angular.io/guide/lazy-loading-ngmodules>



Angular Course

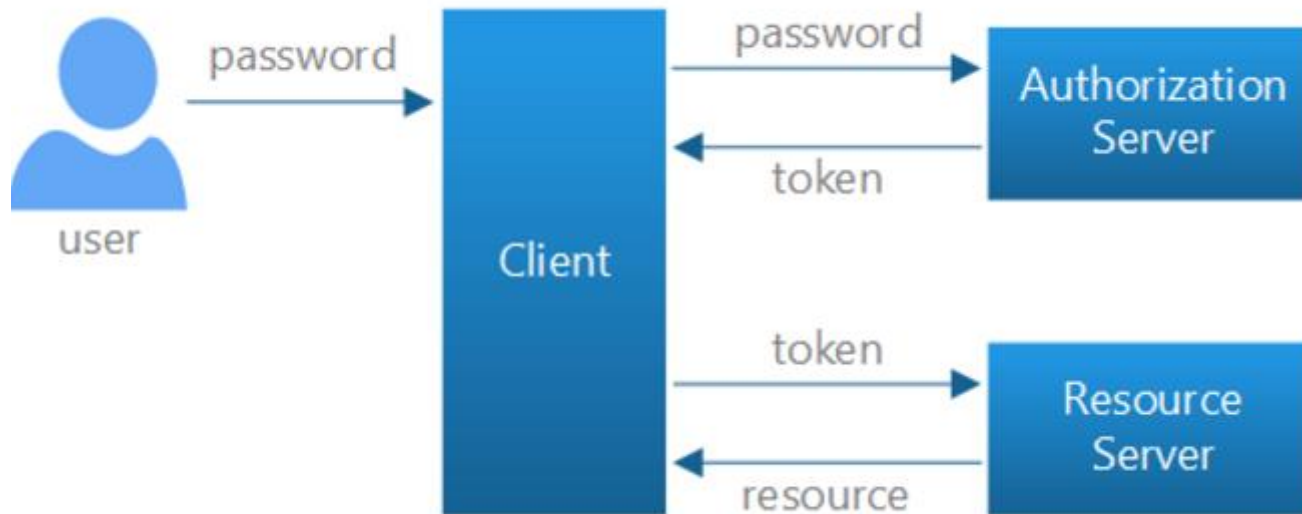
In this course you will get a hands-on of the major features of Angular

David Oliveira

Senior Consultant

27th February, 2018

Secure our App



Angular's route guards are interfaces which can tell the router whether or not it should allow navigation to a requested route. They make this decision by looking for a true or false return value from a class which implements the given guard interface. There are five different types of guards and each of them is called in a particular sequence.

- **CanActivate** to mediate navigation *to* a route.
- CanActivateChild to mediate navigation *to* a child route
- CanDeactivate to mediate navigation *away* from the current route.
- CanLoad to perform route data retrieval *before* route activation.
- Resolve to mediate navigation *to* a feature module loaded *asynchronously*.

<https://angular.io/guide/router#milestone-5-route-guards>

Intercepts HttpRequest and handles them.

Any interceptor that we want to create needs to implement the `HttpInterceptor` interface. This means that our new class must have a method called `intercept` with `HttpRequest` and `HttpHandler` parameters. Using interceptors is all about changing outgoing requests and incoming responses, but we can't tamper with the original request—it needs to be immutable. To make changes we need to clone the original request.

The interceptor needs to be added to the `HTTP_INTERCEPTORS` array. This is done by making the existing `HTTP_INTERCEPTORS` array use the new class we've created. Add this in the providers array for our application's module.

<https://angular.io/api/common/http/HttpInterceptor>



Questions and Discussion

HITACHI
Inspire the Next 