Lazy Loading with Angular 7!

How Lazy Loading work?
Lazy Loading is load only what we need to use when first starting up the application. **If user navigate to a new page, then the component for that page will load immediately.** However, this does not mean that it load the component all the time we change page. In fact, it **load only for the first visit page, it will not load when we revisit that page again.**

Step 1 - Create a new App with Routing

Our app will load the AppComponent by default at the root URL, then when the user navigates to lazy/load-me, the lazy module will be loaded asynchronously.

ng new lazyDemo - -routing

Now add a link to the lazy url in the app component.

<button routerLink="/lazy/load-me">Click Here, I am Lazy!</button>
<router-outlet></router-outlet>

Step 2 - Generate the “Lazy” Module

Let’s create a module that will be lazy loaded, along with a couple components. The --flat flag prevents a directory from being created, then we can easily add components to the module via the Angular CLI.

ng g module lazy --flat
ng g component lazy-parent --module lazy
ng g component lazy-child --module lazy

Inside the lazy module we need to import the RouterModule. Two things to notice here:

The route path to the component is path: 'load-me', even though it will actually be activated on lazy/load-me. This is a child route, which you will see come together in step 3
The RouterModule calls forChild(routes).

```
import { NgModule, CUSTOM_ELEMENTS_SCHEMA} from '@angular/core';
import { CommonModule } from '@angular/common';
import { LazyParentComponent } from './lazy-parent/lazy-parent.component';
import { LazyChildComponent } from './lazy-child/lazy-child.component';

import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [
   { path: 'load-me', component: LazyParentComponent }
];

@NgModule({
  imports: [
    CommonModule,
    RouterModule.forChild(routes)
  ],
  declarations: [
    LazyParentComponent,
    LazyChildComponent
  ],
        schemas:[CUSTOM_ELEMENTS_SCHEMA]
})
export class LazyModule { }
```

And here's what the LazyParentComponent looks like. Just looping over the child component a few times.

```
<div *ngFor="let name of ['Foo', 'Bar', 'Baz']">
  <p>Hi, my name is {{name}}. I'm a lazy child component.</p>

  <lazy-child></lazy-child>

</div>
```

Your Angular CLI config might append an app prefix to the component selector name. If so, make sure remove the prefix and for the last section to work properly. For example…

```
@Component({ selector: 'lazy-child' })
```

Step 3 - Point the App Router to the Lazy Module
The final step is to point the lazy route to the lazy module from the app router. We can do this with the loadChildren property with the path to the module file, then reference the module itself with a hash #. This tells angular to only load LazyModule when the lazy url is activated.

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [
   { path: 'lazy', loadChildren: './lazy.module#LazyModule'}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Verify Lazy Loading is Working

Let's make sure Lazy Loading is working. In chrome, open developer tools and click the network tab. When you navigate to the lazy url, **after clicking on button you should see a lazy.module.chunk.js file rendered.** In this demo, you can see it took 2ms to load. And that has lazy-child and lazy-parent component, those are getting loaded with lazy loading.

Special router called loadChildren, this is use for lazy loading module and it accept value as a string. The path of the loadChildren is based on src directory, you need to provide the full path of your feature module and put the '#' sign then your export module class name.