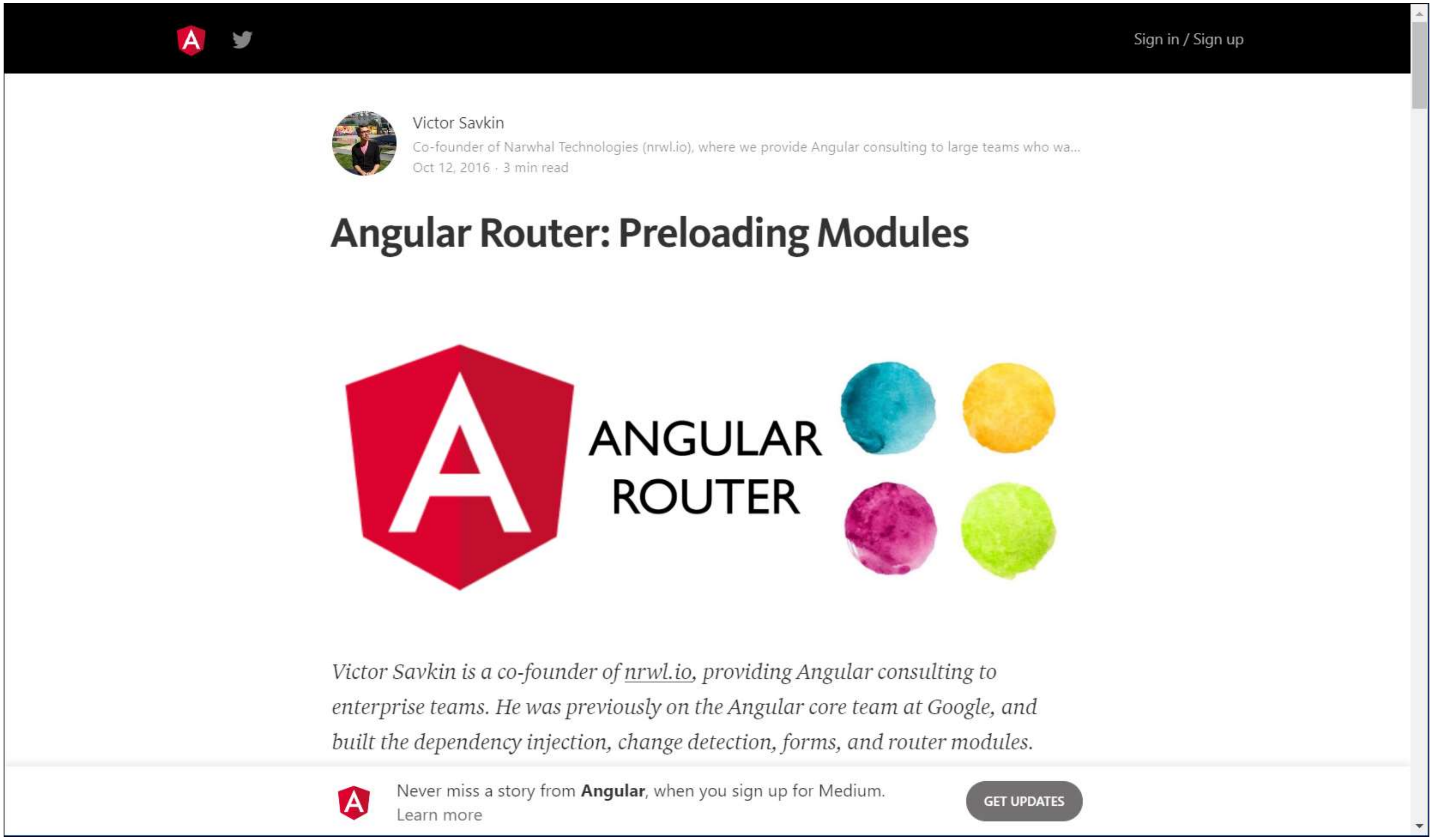
Reactive Angular met RxJS Routing & Lazy Loading

Peter Kassenaar –
info@kassenaar.com

What is lazy loading

- Deferred loading of modules, until the user needs them.
    - Modules are loaded once the user navigates to them.
- OR: for optimal user experience:
    - Load the minimum setup for the application to work, so the user can interact with the app.
    - Then asynchrounously load other modules.
    - They are instantly available if the user navigates to them
- Only *modules* can be loaded lazily, not *components*.
- Lazy loading works in conjunction with the router.
- It is considered best practice nowadays to use LL from the start

Victor Savkin – creator of the router


Sign in / Sign up

Victor Savkin
Co-founder of Narwhal Technologies (nrwl.io), where we provide Angular consulting to large teams who wa...
Oct 12, 2016 · 3 min read

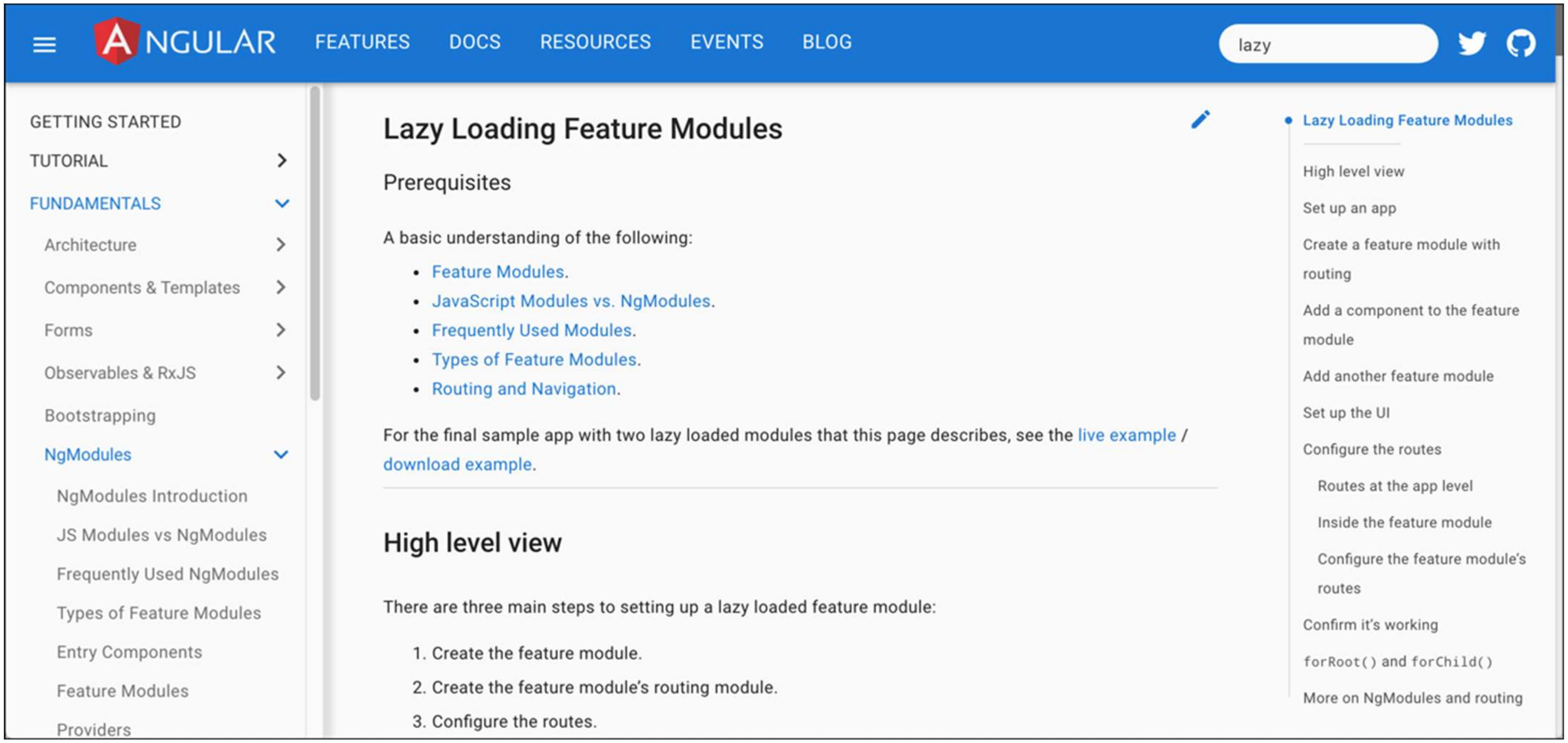
Angular Router: Preloading Modules

Victor Savkin is a co-founder of nrwl.io, providing Angular consulting to enterprise teams. He was previously on the Angular core team at Google, and built the dependency injection, change detection, forms, and router modules.

Never miss a story from **Angular**, when you sign up for Medium.
Learn more

https://vsavkin.com/angular-router-preloading-modules-ba3c75e424cb

Official documentation



Lazy Loading Feature Modules

Prerequisites

A basic understanding of the following:

- Feature Modules.
- JavaScript Modules vs. NgModules.
- Frequently Used Modules.
- Types of Feature Modules.
- Routing and Navigation.

For the final sample app with two lazy loaded modules that this page describes, see the live example / download example.

High level view

There are three main steps to setting up a lazy loaded feature module:

1. Create the feature module.
2. Create the feature module's routing module.
3. Configure the routes.

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

How to lazy load : Angular 4.x – 7.x

Add or edit `app-routing.module.ts`

- Don’t point directly to components
- Point to Modules instead. Use `loadChildren()`
- Note the (ugly) stringnotation

```
const routes: Routes = [
   {path: '', redirectTo: 'customers', pathMatch: 'full'},
   {path: 'customers', loadChildren: './customer/customer.module#CustomerModule'},
   {path: 'products', loadChildren: './products/products.module#ProductsModule'},
];

export const AppRoutingModule = RouterModule.forRoot(routes);
```

How to lazy load : Angular 8.x+

Add or edit `app-routing.module.ts`

- Use the WebPack `import()` statement
- More in line with other frameworks
- More Typesafe.

```
const routes: Routes = [
    {path: '', redirectTo: 'customers', pathMatch: 'full’},
    // New notation (Angular 8+):
    {
        path: 'customers',
        loadChildren: () => import('./customer/customer.module')
            .then(mod => mod.CustomerModule)
    },
    {
        path: 'products',
        loadChildren: () => import('./products/products.module')
            .then(mod => mod.ProductsModule)
    },
];
export const AppRoutingModule = RouterModule.forRoot(routes);
```

Edit `app.module.ts` (no more loading of modules)

```
// import routing module that defines the LL
import {AppRoutingModule} from './app.routing.module';

@NgModule({
    …
   imports     : [
      BrowserModule,
      AppRoutingModule
   ],
   bootstrap   : [AppComponent]
})
export class AppModule {
}
```

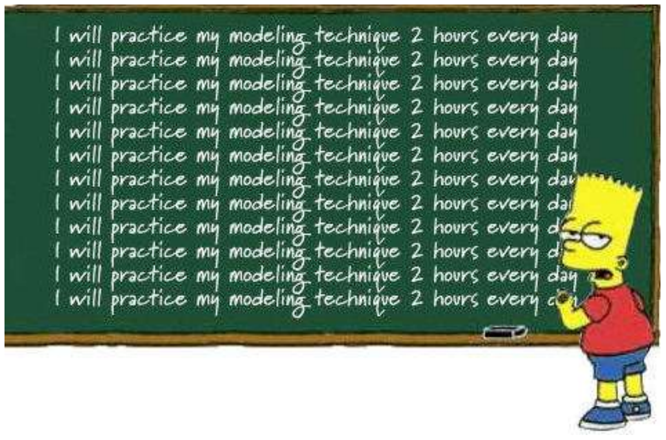
Edit separate modules, add `RouterModule.forChild()` with various components.

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

// Lazy loaded routes for this module
const customerRoutes: Routes = [
   {path: '', component: CustomerComponent}
];

@NgModule({
   imports     : [
      …
      RouterModule.forChild(customerRoutes)
   ],
    …
})
export class CustomerModule {
}
console.log('CustomerModule loaded lazily...');
```

Workshop

- Open `../110-lazy-loading`.
- Create a new module
- Create a new component inside this new module and give it some UI.
- Add a route to the new component
- Use the new module in the root module and lazy load it
- Add a link to navigate to the lazy loaded module.

- *OR:*
- Add LL from scratch to your own application, using the steps described in this module.
- Add a new (dynamic) child route to Module

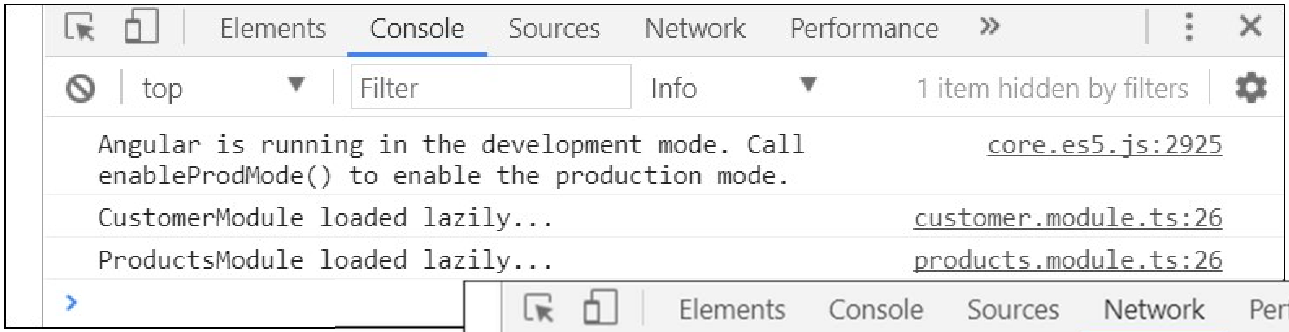


Preloading strategies

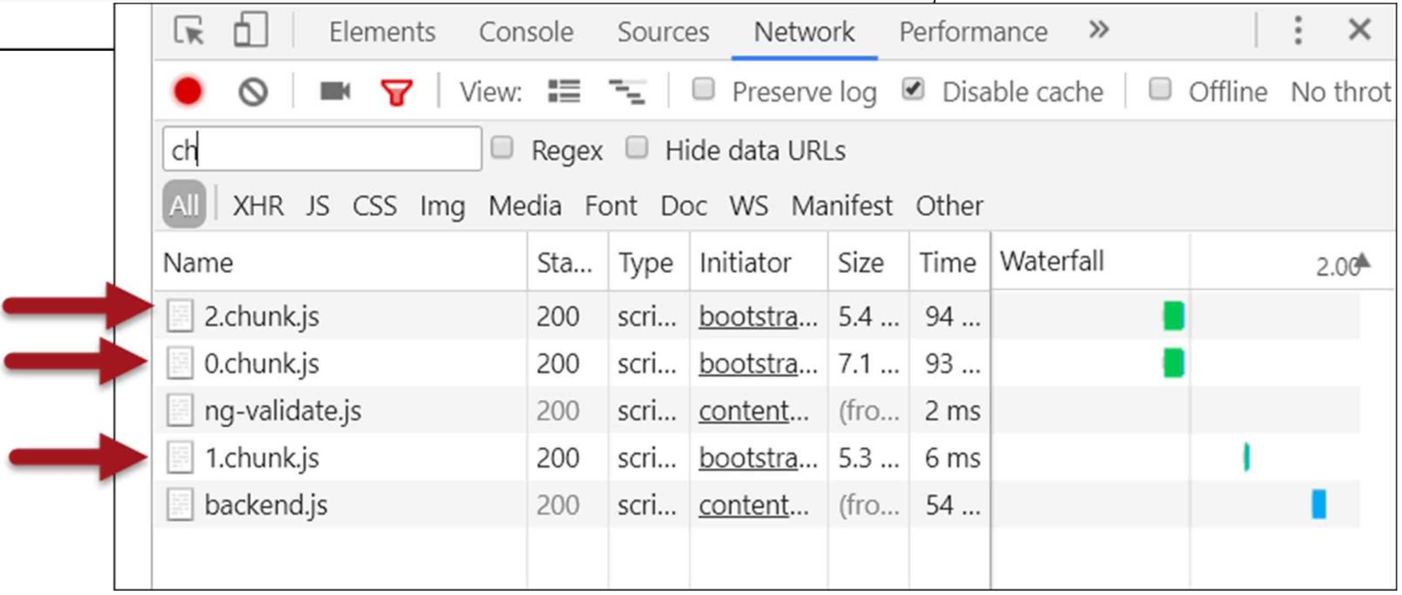
Preloading Strategies

- Optimize Lazy Loading even further: preloading strategies
  - Load all modules in background
  - Load only modules *you want to load* in the background
- Default preloading: `PreloadAllModules`

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


const config: ExtraOptions    = {
   preloadingStrategy: PreloadAllModules
};

export const AppRoutingModule = RouterModule.forRoot(routes, config);
```



```
Angular is running in the development mode. Call          core.es5.js:2925
enableProdMode() to enable the production mode.
CustomerModule loaded lazily...                       customer.module.ts:26
ProductsModule loaded lazily...                       products.module.ts:26
```

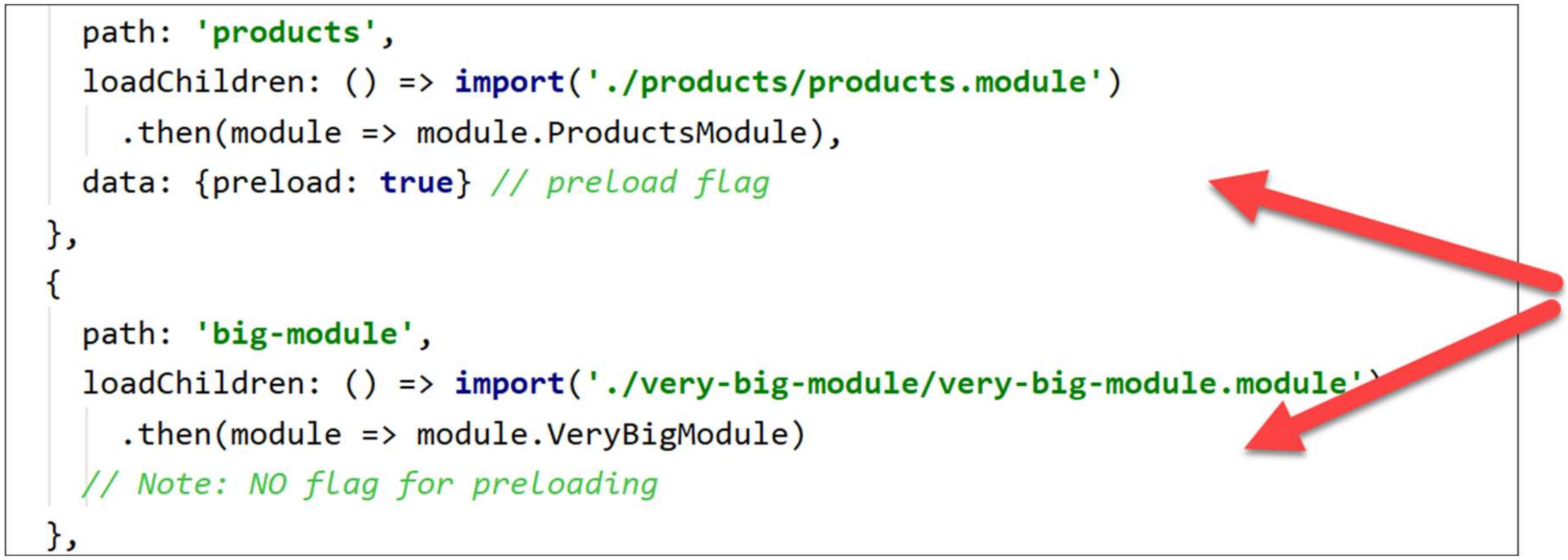


Name	Sta...	Type	Initiator	Size	Time
2.chunk.js	200	scri...	bootstra...	5.4 ...	94 ...
0.chunk.js	200	scri...	bootstra...	7.1 ...	93 ...
ng-validate.js	200	scri...	content...	(fro...	2 ms
1.chunk.js	200	scri...	bootstra...	5.3 ...	6 ms
backend.js	200	scri...	content...	(fro...	54 ...

https://angular.io/api/router/PreloadAllModules

Custom preloading strategy

- Define which module(s) are loaded lazily, while others are loaded on demand
- Solution: compose a strategy that *only* preloads routes when a custom `data.preload` flag is set to `true`



```
    path: 'products',
    loadChildren: () => import('./products/products.module')
      .then(module => module.ProductsModule),
    data: {preload: true} // preload flag
  },
  {
    path: 'big-module',
    loadChildren: () => import('./very-big-module/very-big-module.module')
      .then(module => module.VeryBigModule)
    // Note: NO flag for preloading
  },
```

Steps

1. Create new module, with a (potential) heavy load
2. Add `data` property and set `{ preload:true }` to every route you want to load lazily
3. Assign custom preloader to `preloadingStrategy`:

```
…
const config: ExtraOptions = {
   preloadingStrategy: MyCustomPreloader
};
@NgModule({
   imports   : [RouterModule.forRoot(routes, config)],
   exports   : [RouterModule],
   providers: [MyCustomPreloader]
})
export class AppRoutingModule {
}
```

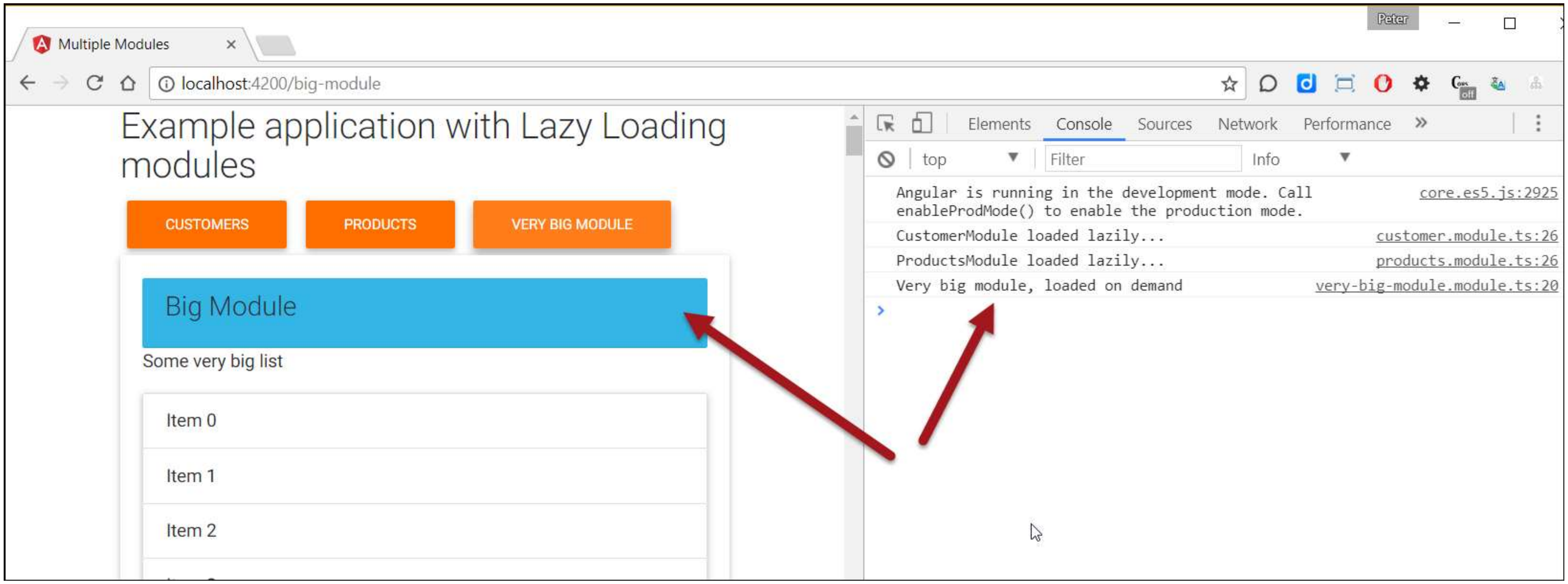
Define custom loader

```
// app.routing.loader.ts
import { PreloadingStrategy, Route } from '@angular/router';

import { Observable, of } from 'rxjs';

export class MyCustomPreloader implements PreloadingStrategy {
   preload(route: Route, load: Function): Observable<any> {
      // only preload the route if data attribute is set and preload===true
      return route.data && route.data.preload ? load() : of(null);
   }
}
```

Official documentation https://angular.io/guide/router#custom-preloading-strategy

Run the app

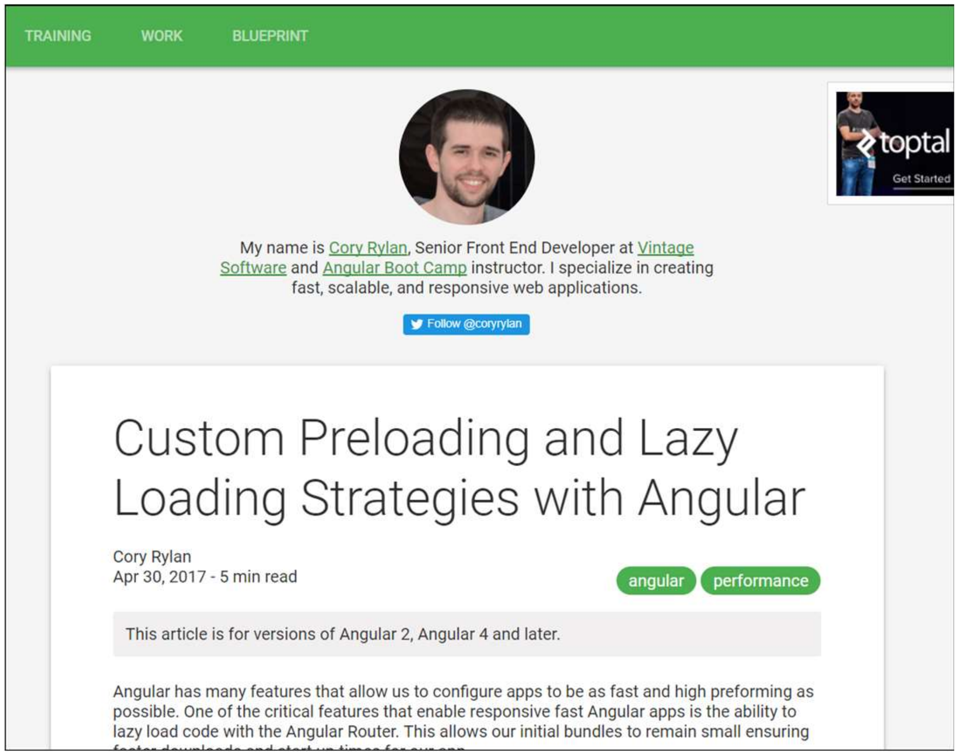
Run the app. The first 2 modules should be loaded lazily, the third module should be loaded on demand



```
Angular is running in the development mode. Call      core.es5.js:2925
enableProdMode() to enable the production mode.
CustomerModule loaded lazily...                  customer.module.ts:26
ProductsModule loaded lazily...                  products.module.ts:26
Very big module, loaded on demand      very-big-module.module.ts:20
```

Example: ../120-custom-preloading

More information



My name is Cory Rylan, Senior Front End Developer at Vintage Software and Angular Boot Camp instructor. I specialize in creating fast, scalable, and responsive web applications.

Custom Preloading and Lazy Loading Strategies with Angular

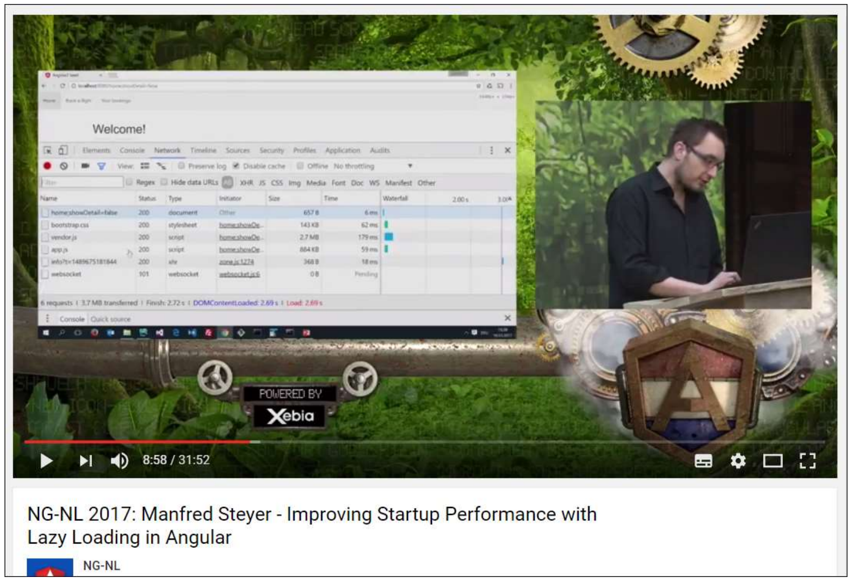
Cory Rylan
Apr 30, 2017 - 5 min read

This article is for versions of Angular 2, Angular 4 and later.

Angular has many features that allow us to configure apps to be as fast and high preforming as possible. One of the critical features that enable responsive fast Angular apps is the ability to lazy load code with the Angular Router. This allows our initial bundles to remain small ensuring

https://coryrylan.com/blog/custom-preloading-and-lazy-loading-strategies-with-angular

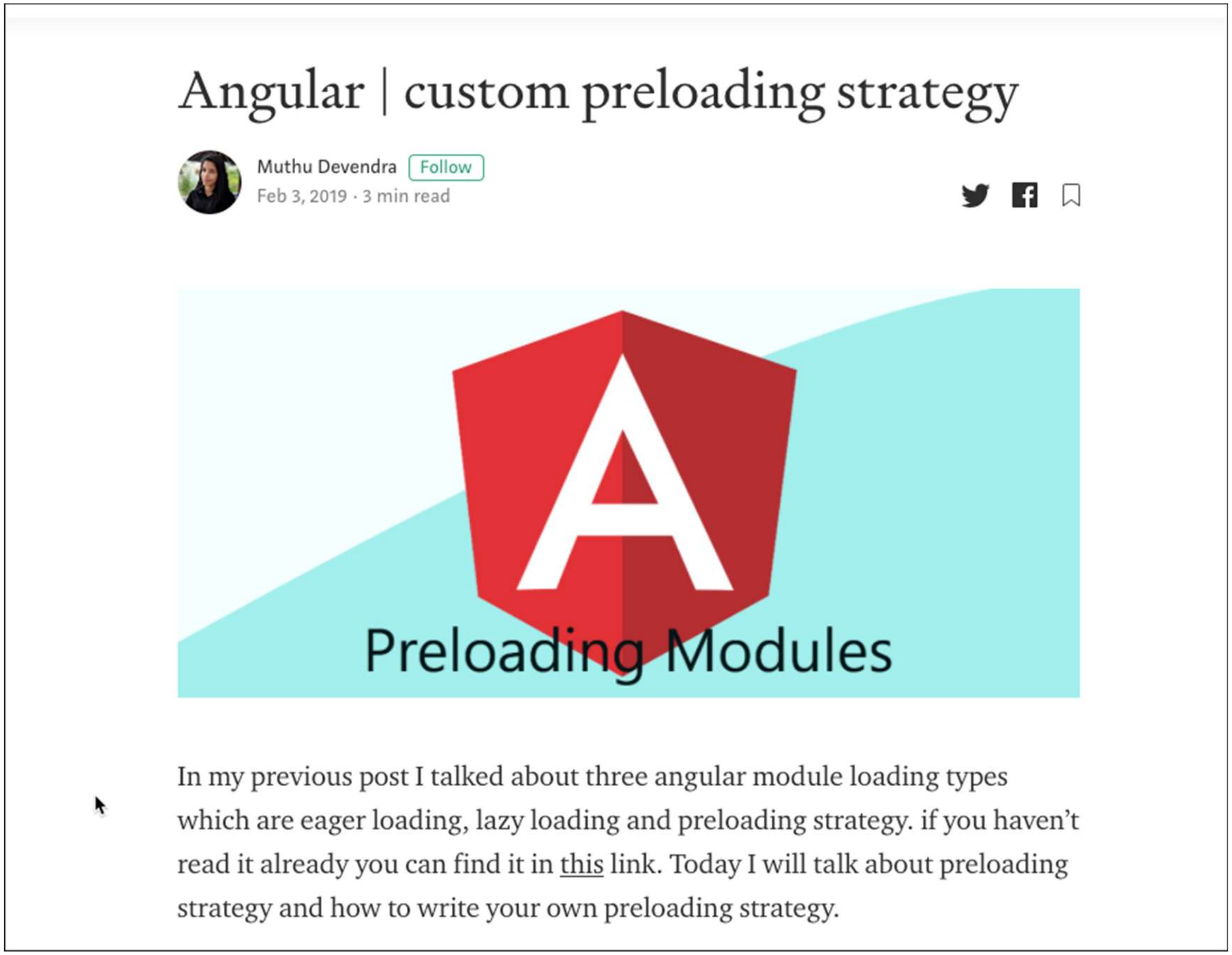
Manfred Steyer - Improving Startup Performance with Lazy Loading in Angular



NG-NL 2017: Manfred Steyer - Improving Startup Performance with Lazy Loading in Angular

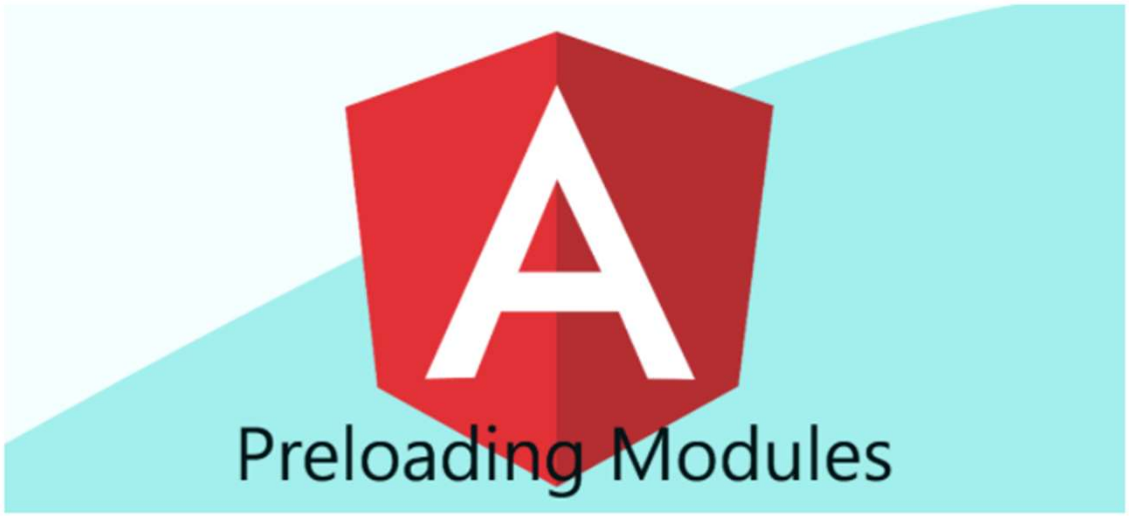
NG-NL

https://www.youtube.com/watch?v=n6EMOeCDfjc



Angular | custom preloading strategy

Muthu Devendra Follow
Feb 3, 2019 · 3 min read



In my previous post I talked about three angular module loading types which are eager loading, lazy loading and preloading strategy. if you haven't read it already you can find it in this link. Today I will talk about preloading strategy and how to write your own preloading strategy.

https://medium.com/@muthudevendra/angular-custom-preloading-strategy-32abe99944f8

Workshop

- Add a new module w/ component to your application.
- Add the module to the routing section of your application. Add a link to navigate to the route.
- Let *other* modules be loaded lazily by adding a `data` property
- Write a custom preloading service class, e.g. `preload.service.ts`
- Add the custom preloader to `app.routing.module.ts`.
    - Note: make sure this is (now) actually a Module, as it has to import and provide `app.preloader.ts`
- Example: `../120-custom-preloading`