Angular Advanced
02 – Lazy loading

Peter Kassenaar –
info@kassenaar.com

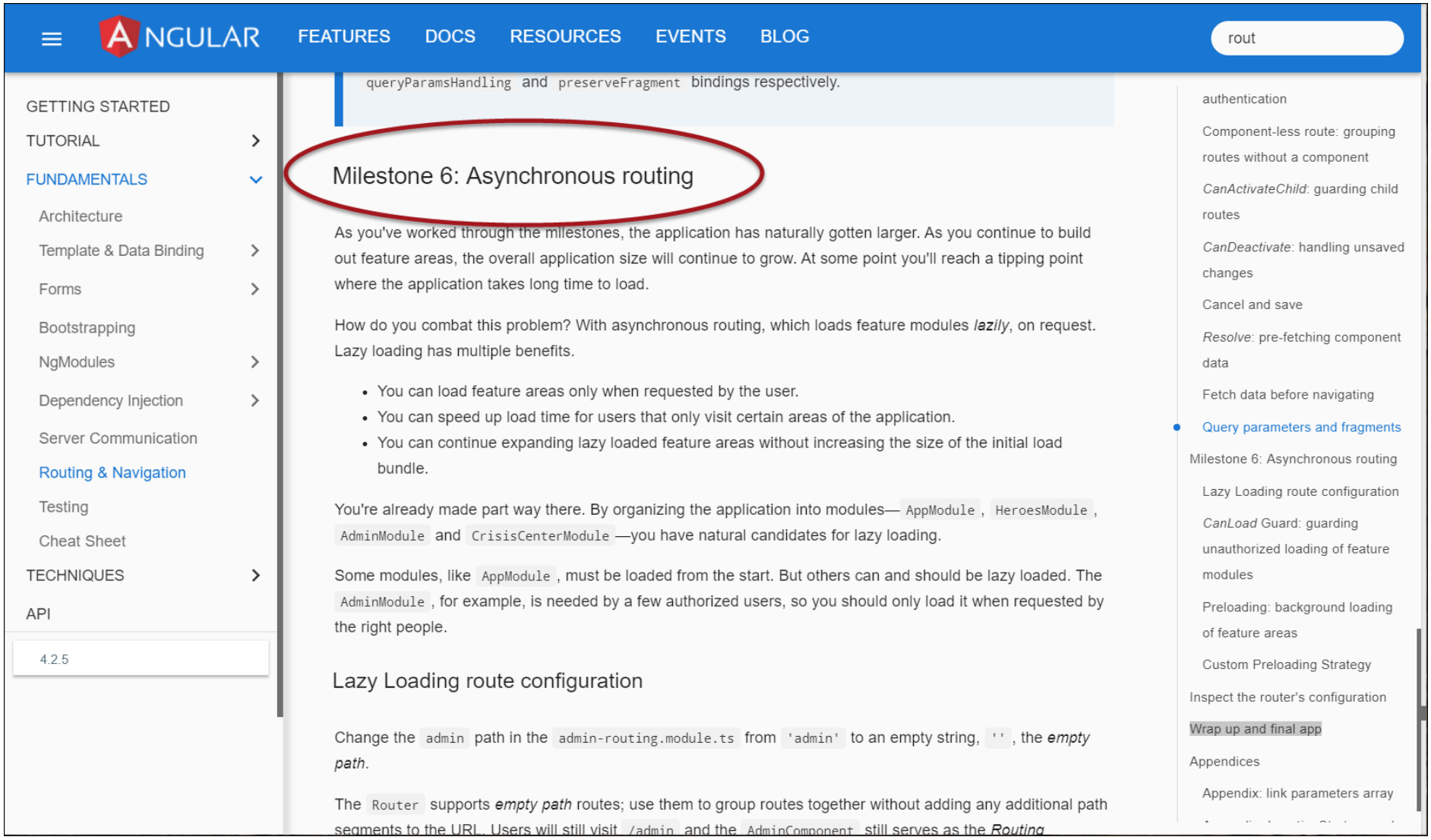
What is lazy loading

- Deferred loading of modules, until the user needs 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

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

Official documentation



https://angular.io/guide/router#asynchronous-routing

How to lazy load

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

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

```
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);
```

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 {
}
```

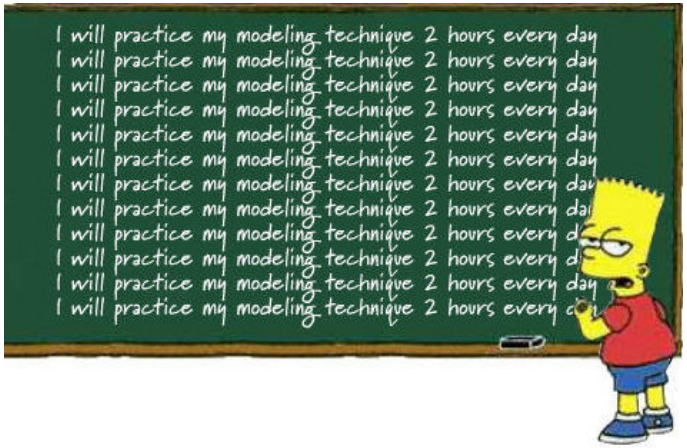
Edit separate modules,
add `RouterModule.forChild()` with various components.

```
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.

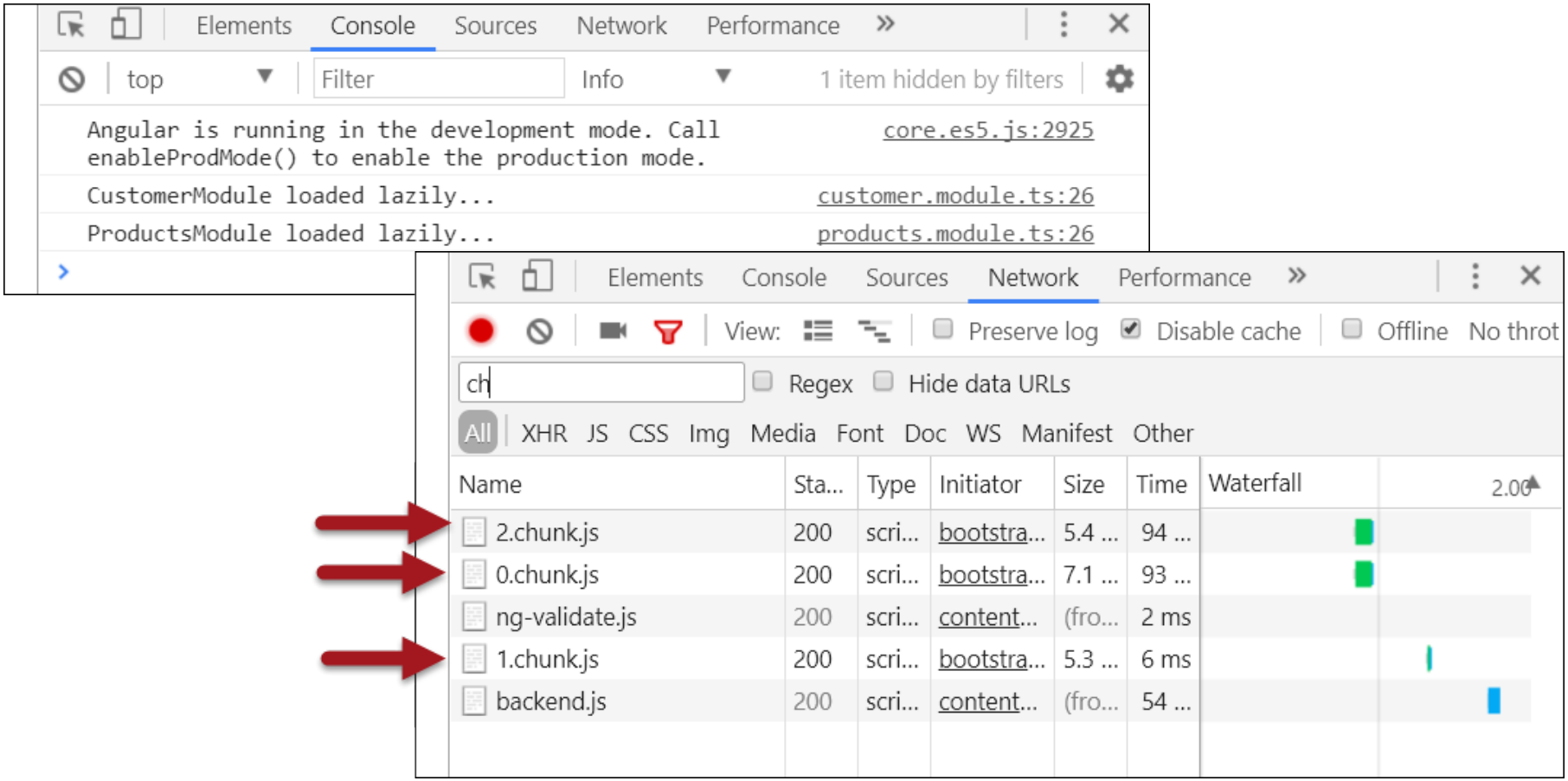


Preloading strategies

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);
```



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: './products/products.module#ProductsModule',
    data         : {preload: true}
},
{
    path         : 'big-module',
    loadChildren: './very-big-module/very-big-module.module#VeryBigModule'
},
```

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 {
}
```

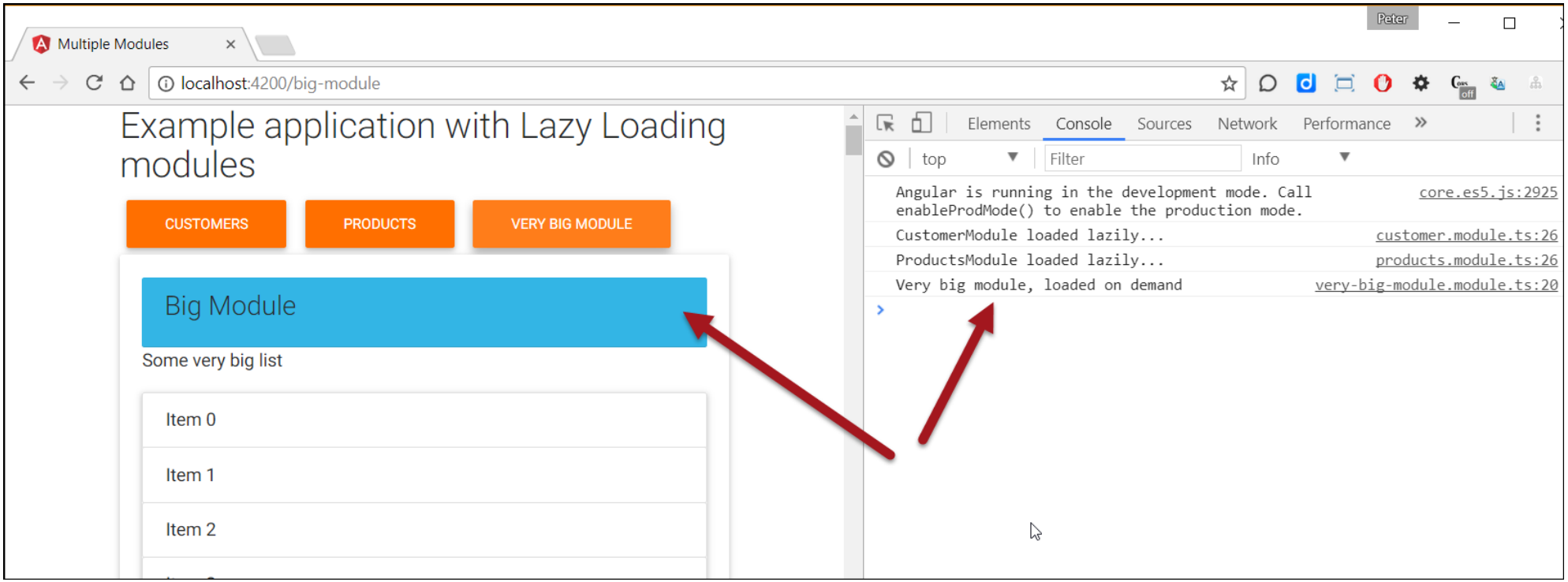
Define custom loader

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

import { Observable } from 'rxjs/Observable';
import 'rxjs/add/observable/of';

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() : Observable.of(null);
   }
}
```

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

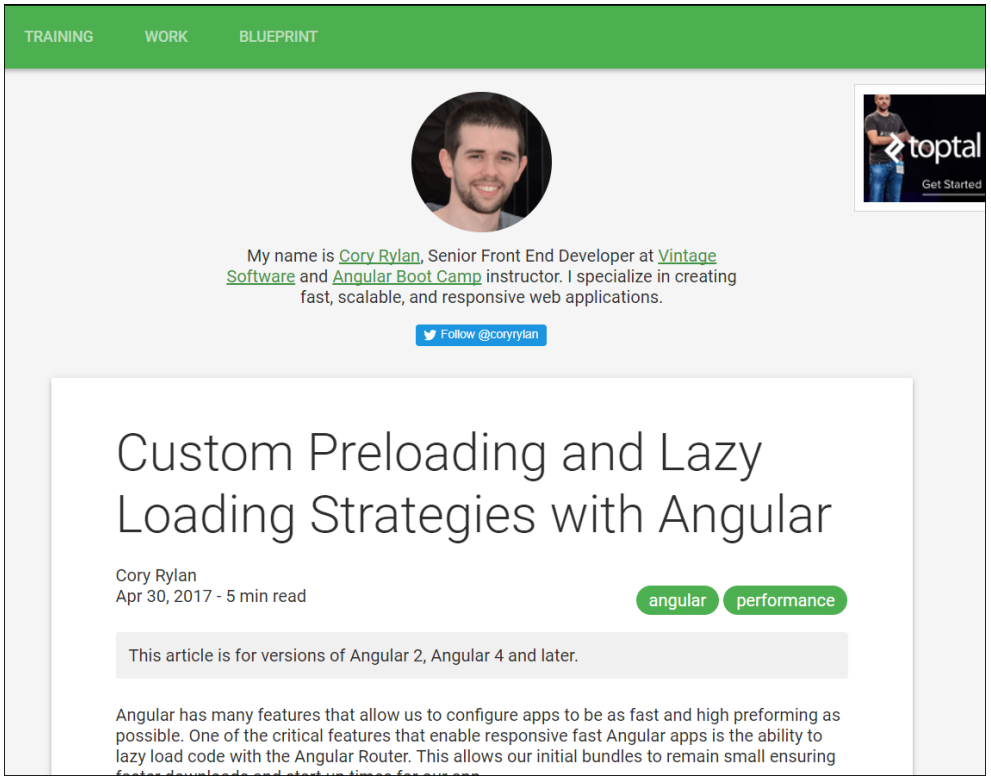
Run the app

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



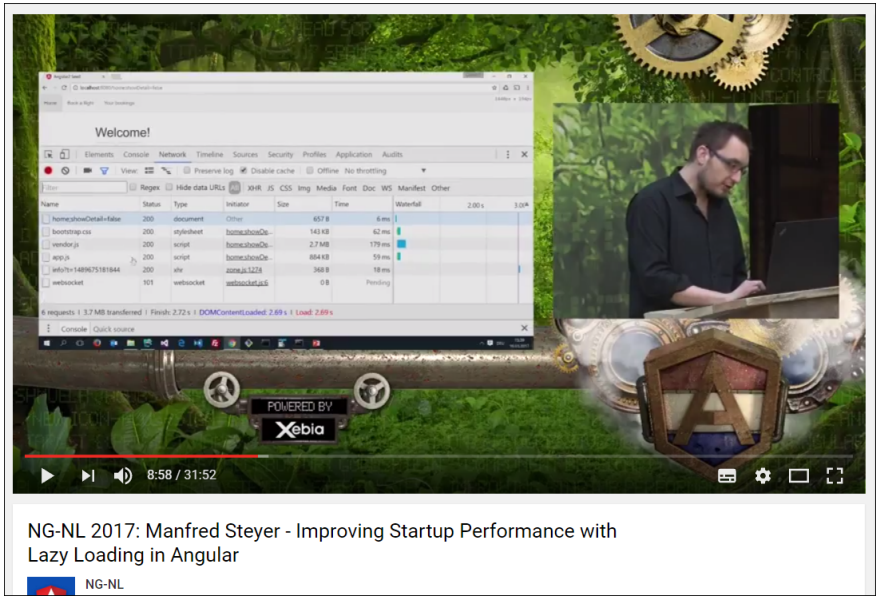
Example: ../120-custom-preloading

More information



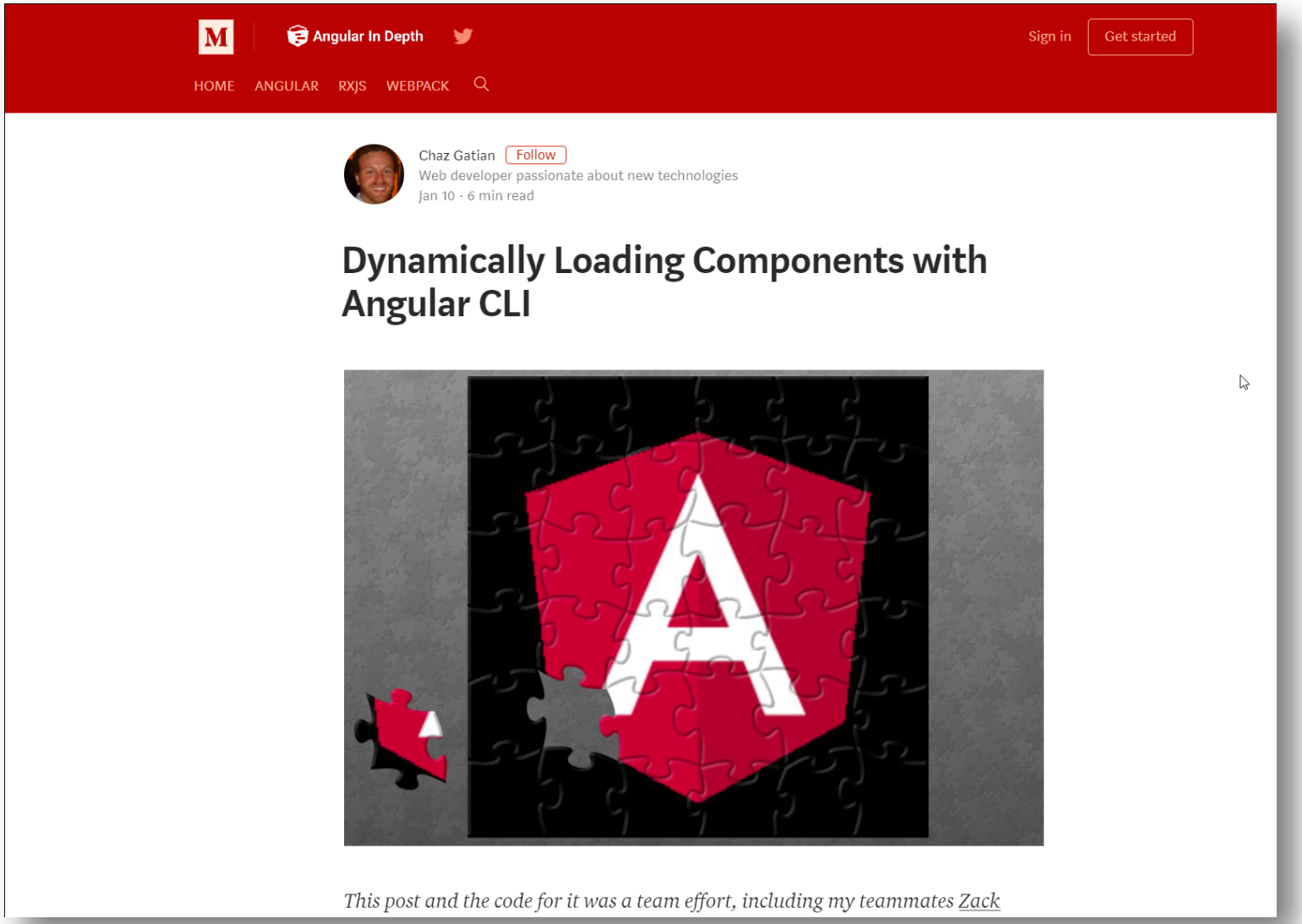
https://coryrylan.com/blog/custom-preloading-and-lazy-loading-strategies-with-angular

Manfred Steyer - Improving Startup Performance with Lazy Loading in Angular



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

Dynamically Loading *Components*



Beware!

Deep stuff.

https://blog.angularindepth.com/dynamically-loading-components-with-angular-cli-92a3c69bcd28

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* components be loaded lazily by adding a `data` property
- Write a custom preloading class, in `app.preloader.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`