Angular Performance Tuning

Manfred Steyer
SOFTWAREarchitekt.at

Turbo Button

Quick Wins

| Bundling | Minification | enableProdMode() |

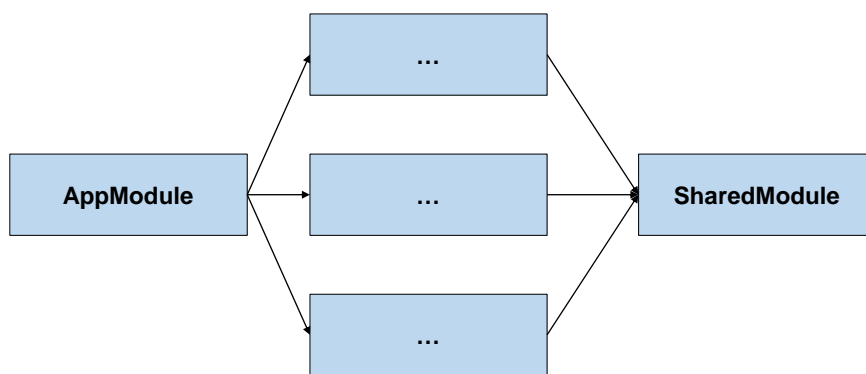SOFTWARE*architekt.at*

Contents

- Lazy Loading and Preloading
- Performance for Data Binding with OnPush
- AOT and Tree Shaking

SOFTWARE*architekt.at*
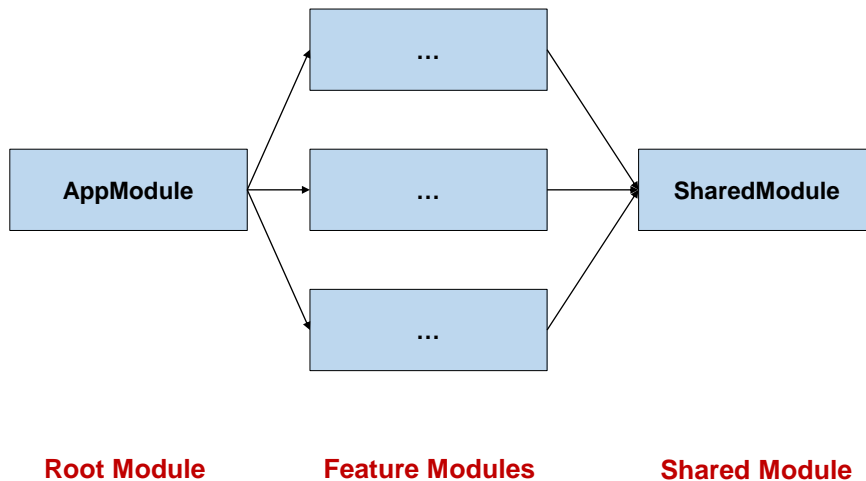
Lazy Loading

Module Structure

Root Module **Feature Modules** **Shared Module**

SOFTWARE*architekt.at*

Lazy Loading

Root Module **Feature Modules** **Shared Module**

SOFTWARE*architekt.at*

Root Module with Lazy Loading

```
const APP_ROUTE_CONFIG: Routes = [
    {
        path: 'home',
        component: HomeComponent
    },
    {
        path: 'flights',
        loadChildren:
            './[…]flight-booking.module#FlightBookingModule'
    }
];
```
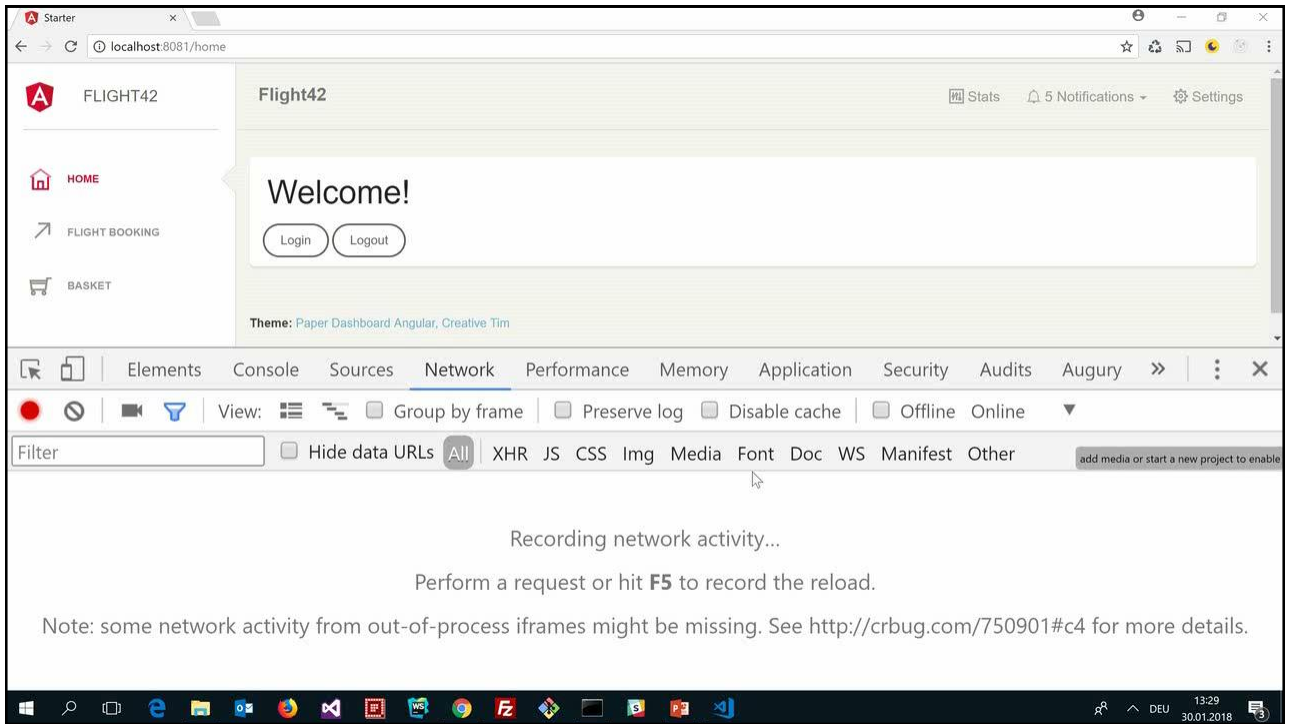
SOFTWARE*architekt.at*

Routes for "lazy" Module

```
const FLIGHT_ROUTES =    [
    {
        path: '',
        component: FlightBookingComponent,
        […]
    },
    […]
}
```

SOFTWARE *architekt.at*

Routes for "lazy" Module

```
const FLIGHT_ROUTES =    [
    {
        path: 'subroute',
        component: FlightBookingComponent,
        […]
    },
    […]
}
```

flight-booking/ subroute

Triggers Lazy Loading w/ loadChildren

SOFTWARE *architekt.at*

Lazy Loading

- Lazy Loading means: Loading it later
- Better startup performance
- Delay during execution for loading on demand

SOFTWARE*architekt.at*

Preloading

Idea

- Module that might be needed later are loaded after the application started
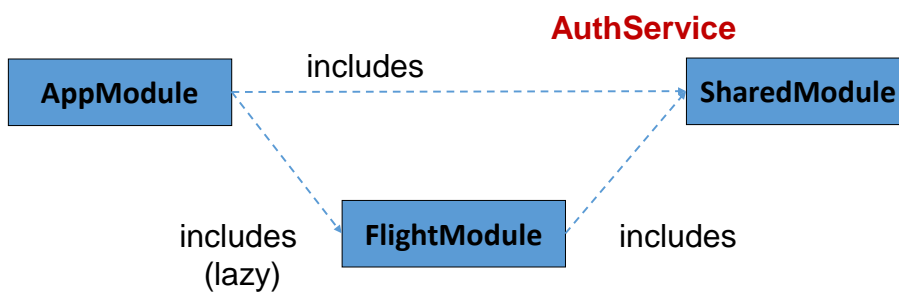- When module is needed it is available immediately

SOFTWARE *architekt.at*

Activate Preloading

```
…
imports: [
    […]
    RouterModule.forRoot(
        ROUTE_CONFIG,
        { preloadingStrategy: PreloadAllModules });
]
…
```

SOFTWARE*architekt.at*

Lazy Loading and Shared Modules

DEMO
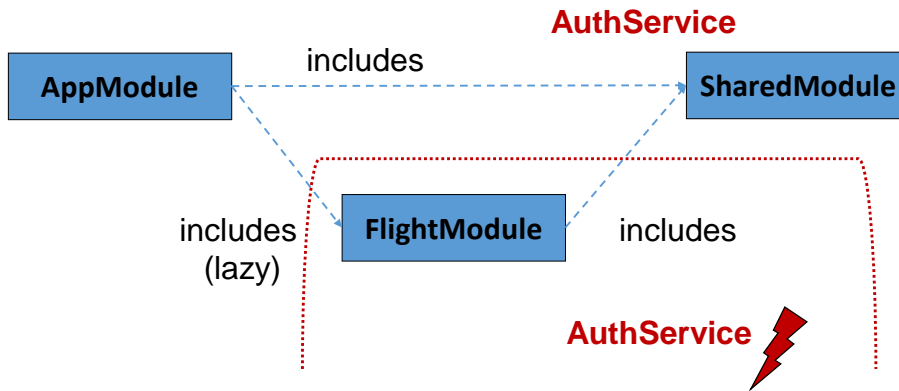
SOFTWAREarchitekt.at

Lazy Loading and Shared Modules

AuthService

AppModule	includes	SharedModule

FlightModule

includes (lazy)

includes

SOFTWAREarchitekt.at

Lazy Loading and Shared Modules

AuthService

AppModule -- includes --> **SharedModule**

includes
(lazy)

FlightModule includes

AuthService ⚡

SOFTWARE*architekt.at*

Lazy Loading and Shared Modules

AuthService

AppModule -- includes --> **SharedModule**

includes
(lazy)

FlightModule includes

Auth~~Service~~ ⚡

SOFTWARE*architekt.at*

Solution

Global Providers like AuthService & Shell
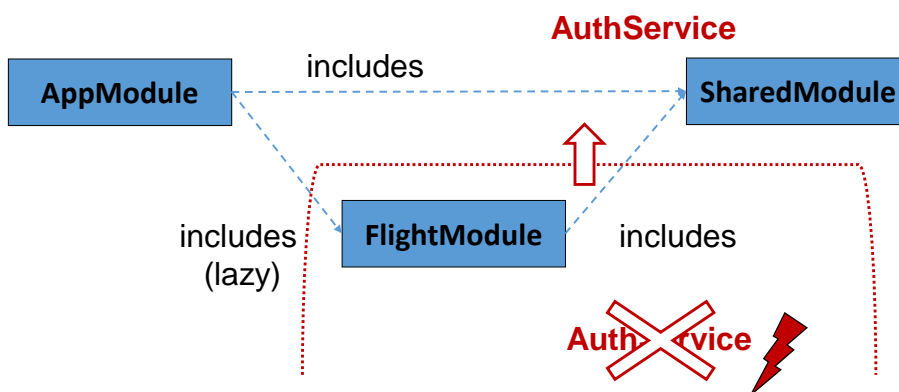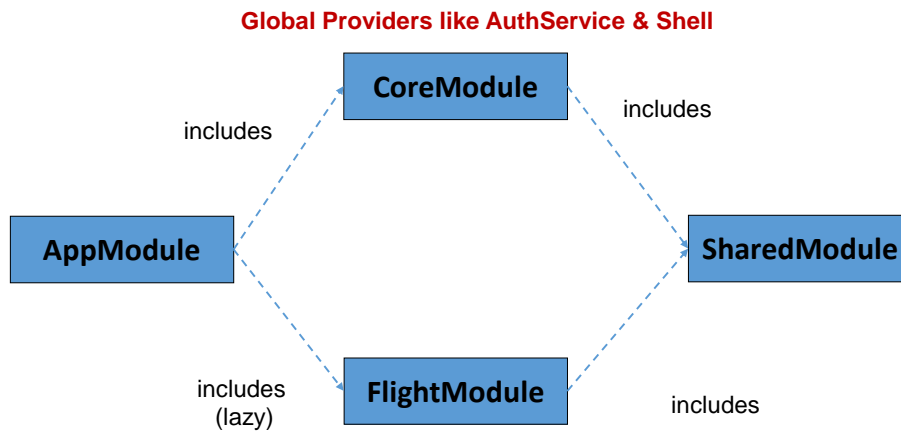
Only import CoreModule into AppModule!

SOFTWARE*architekt.at*

DEMO

SOFTWARE*architekt.at*

Huge CoreModule?
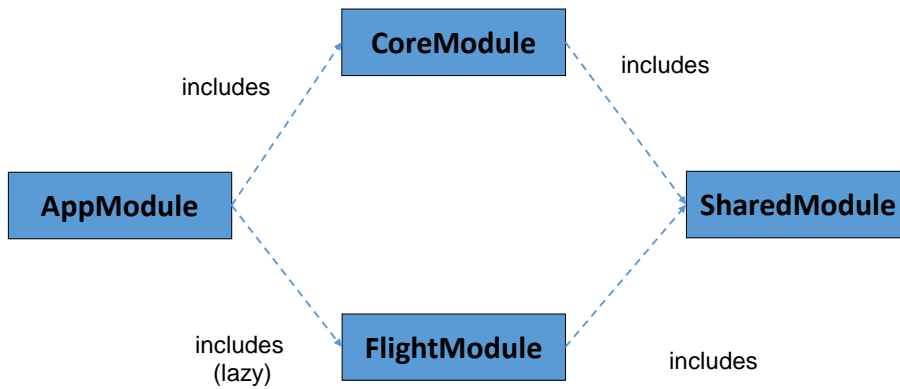
SOFTWAREarchitekt.at
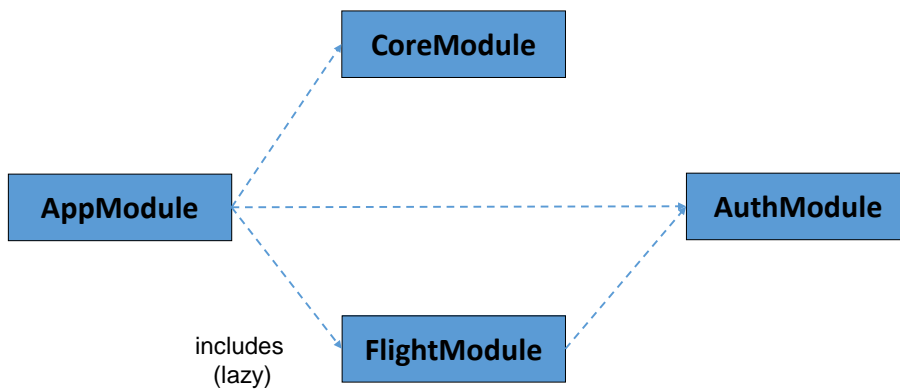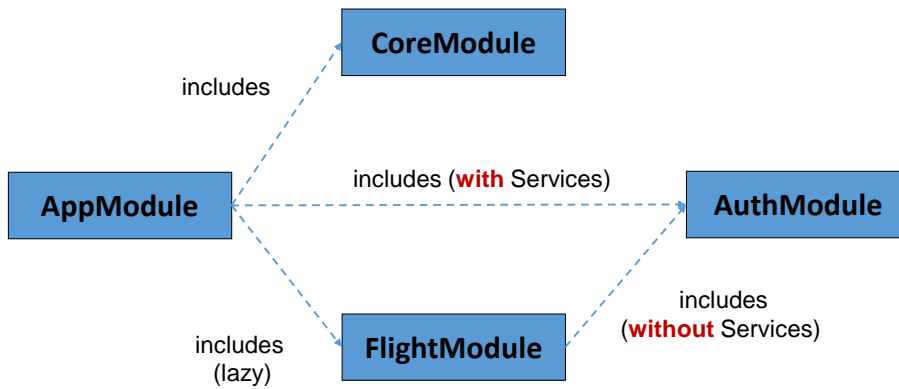
Solution (for Libraries)

SOFTWAREarchitekt.at

Solution (for Libraries)

SOFTWAREarchitekt.at

Auth Module

```
@NgModule({
    […],
    providers: []
})
export class AuthModule {
}
```

SOFTWAREarchitekt.at

Auth Module

```
@NgModule({
    […],
    providers: []
})
export class AuthModule {
    static forRoot(): ModuleWithProviders {
        return {
            ngModule: AuthModule,
            providers: [AuthService, […]]
        }
    }
}
```
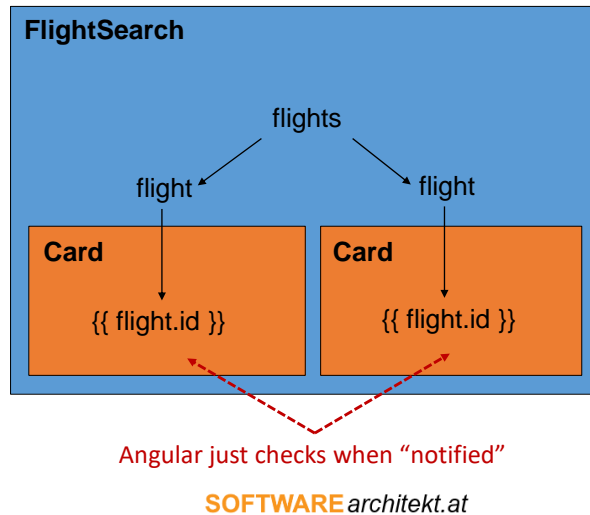
SOFTWAREarchitekt.at

DEMO

SOFTWAREarchitekt.at

Performance-Tuning with OnPush

DEMO

OnPush

FlightSearch

flights

flight   flight

Card   **Card**

{{ flight.id }}   {{ flight.id }}

Angular just checks when "notified"

SOFTWARE *architekt.at*

"Notify" about change?

- Change bound data (@Input)
  - OnPush: Angular just compares the object reference!
  - e. g. oldFlight === newFlight
- Raise Event within the component
- Notify a bound observable
- Trigger it manually
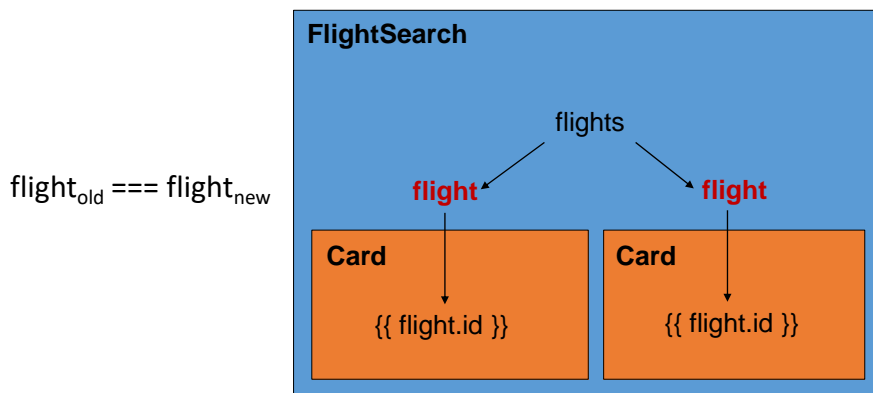 - Don't do this at home ;-)
  - At least: Try to avoid this

SOFTWARE *architekt.at*

Activate OnPush

```
@Component({
      [...]
      changeDetection: ChangeDetectionStrategy.OnPush
})
export class FlightCard {
    [...]
    @Input() flight;
}
```

Change Inputs

$flight_{old} === flight_{new}$

Observables and OnPush

```
<flight-card
    [item]="flight$ | async" [...]>
</flight-card>
```

DEMO 💬

Ahead of Time (AOT) Compilation

Angular Compiler

HTML Template → JavaScript

Template Compiler 💬
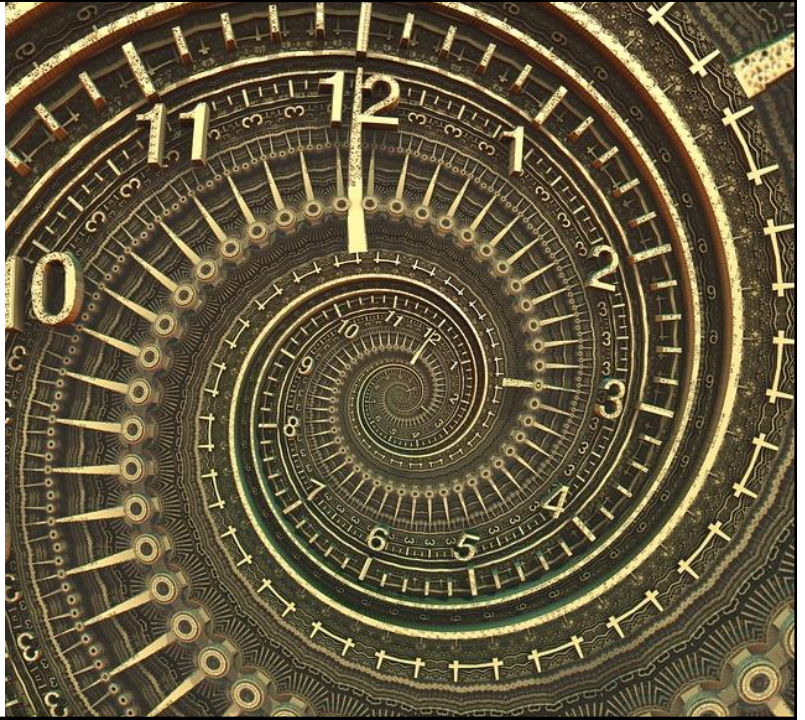
SOFTWAREarchitekt.at
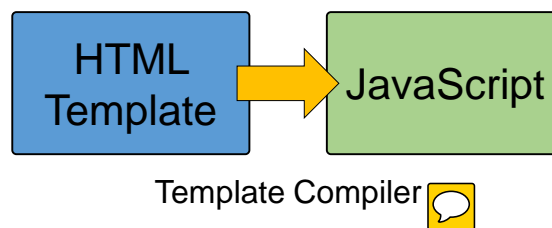
Approaches

- JIT: Just in Time, at runtime
- AOT: Ahead of Time, during build

SOFTWARE*architekt.at*

Advantages of AOT

- Better Startup-Performance
- Smaller Bundles: You don't need to include the compiler!
- Tools can easier analyse the code
  - Remove not needed parts of frameworks
  - Tree Shaking

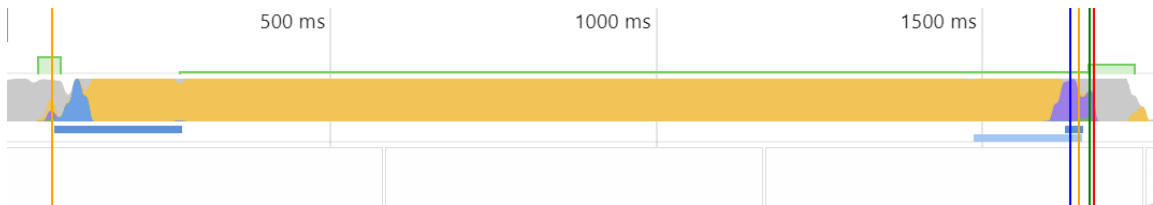**SOFTWARE***architekt.at*

Angular CLI

- ng build --prod
- @ngtools/webpack with AotPlugin
- Soon AngularCompilerPlugin
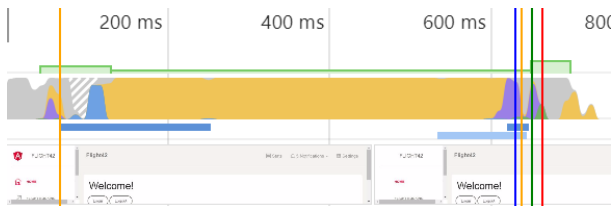- Can be used without CLI too

SOFTWARE*architekt.at*

DEMO

SOFTWARE*architekt.at*

2/27/2018

Flight Search (Prod Build w/o AOT)

Flight Search (Prod Build w/ AOT)

Conclusion

Quick Wins

Lazy Loading and Preloading

OnPush w/ Immutables and Observables

AOT and Tree Shaking

Use the CLI

SOFTWARE*architekt.at*