



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Performance Tuning

Hosted by Alex Thalhammer

Turbo Button



Contents

- Initial Load Performance
 - Lazy Loading and Preloading
 - (Manual) Build Analyzing & Optimization
- Runtime Performance
 - Data Binding with ChangeDetectionStrategy.OnPush

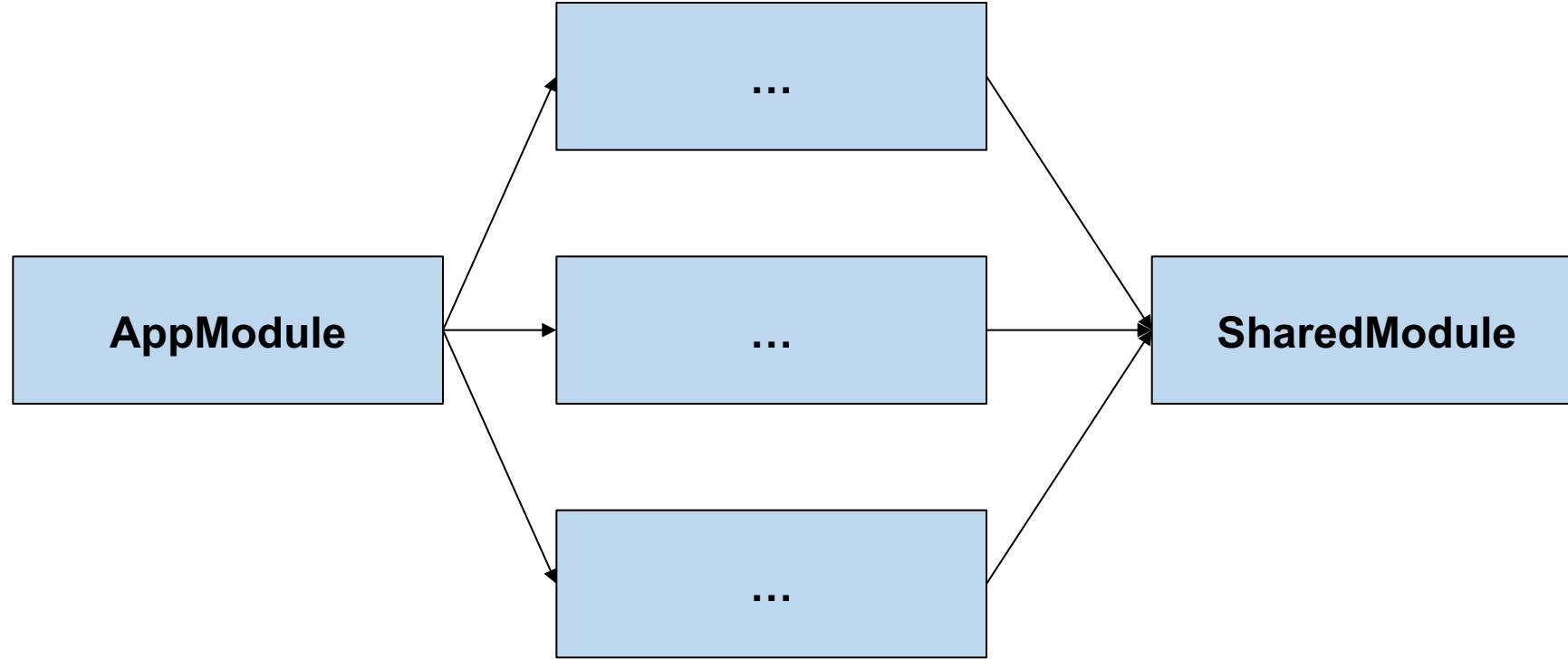
Lazy Loading



Why Lazy Loading?

- Improve initial load time (performance → very important!)

Module Structure

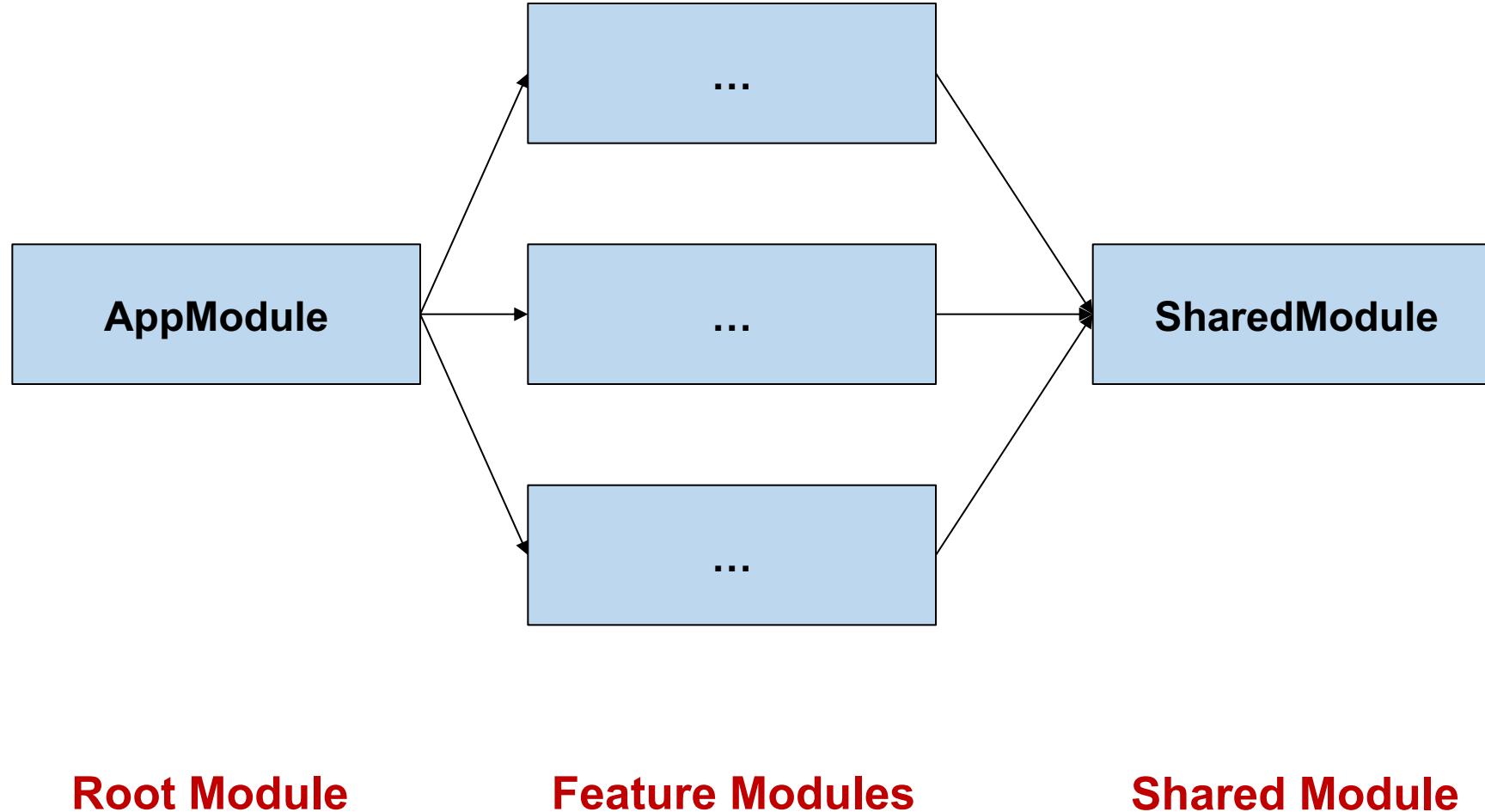


Root Module

Feature Modules

Shared Module

Lazy Loading



Root Module with Lazy Loading

```
const appRoutes: Routes = [
  {
    path: 'home',
    component: HomeComponent
  },
  {
    path: 'flight-booking',
    loadChildren: () => import('./flight-booking/flight-booking.routes')
  }
];
```

Routes for "lazy" Feature Module

```
const flightBookingRoutes: Routes = [  
  {  
    path: 'flight-search',  
    component: FlightSearchComponent,  
    [...]  
  },  
  [...]  
]
```

flight-booking/flight-search

Triggers Lazy Loading w/ loadChildren

Lazy Loading

- Lazy Loading means: Load it later, after startup
- Better initial load performance
- But: Delay during execution for loading on demand

DEMO



Preloading

Idea

- Once the initial load (the important one) is complete load the lazy loaded modules (before they are even used)
- Once the module will come into use it's immediately accessable

Activate Preloading

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

DEMO

Intelligent Preloading with ngx-quicklink

```
...
imports: [
  [...]
  QuicklinkModule,
  RouterModule.forRoot(
    appRoutes,
    { preloadingStrategy: QuicklinkStrategy }
  );
]
...
...
```

<https://web.dev/route-preloading-in-angular/>

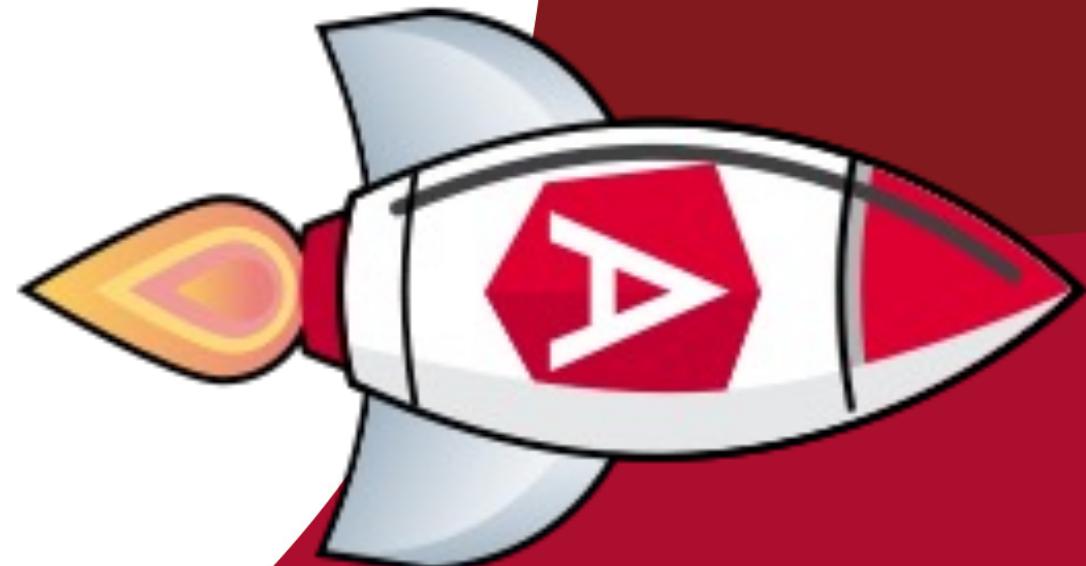
<https://www.npmjs.com/package/ngx-quicklink>

Or CustomPreloadingStrategy

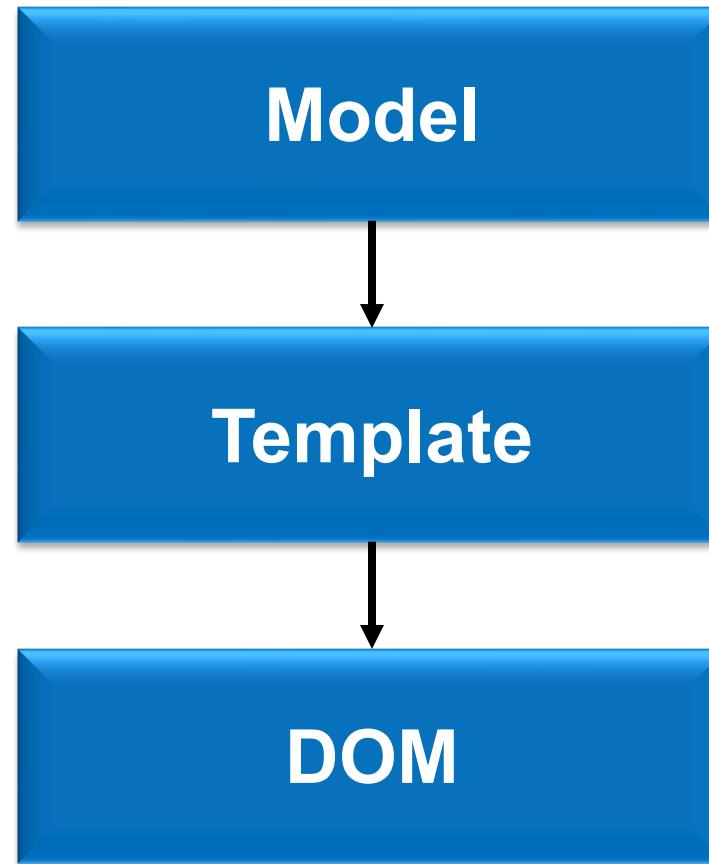
```
...
imports: [
  [...]
  RouterModule.forRoot(
    appRoutes,
    { preloadingStrategy: CustomPreloadingStrategy }
  );
]
...
...
```

LAB

Change Detection in Angular



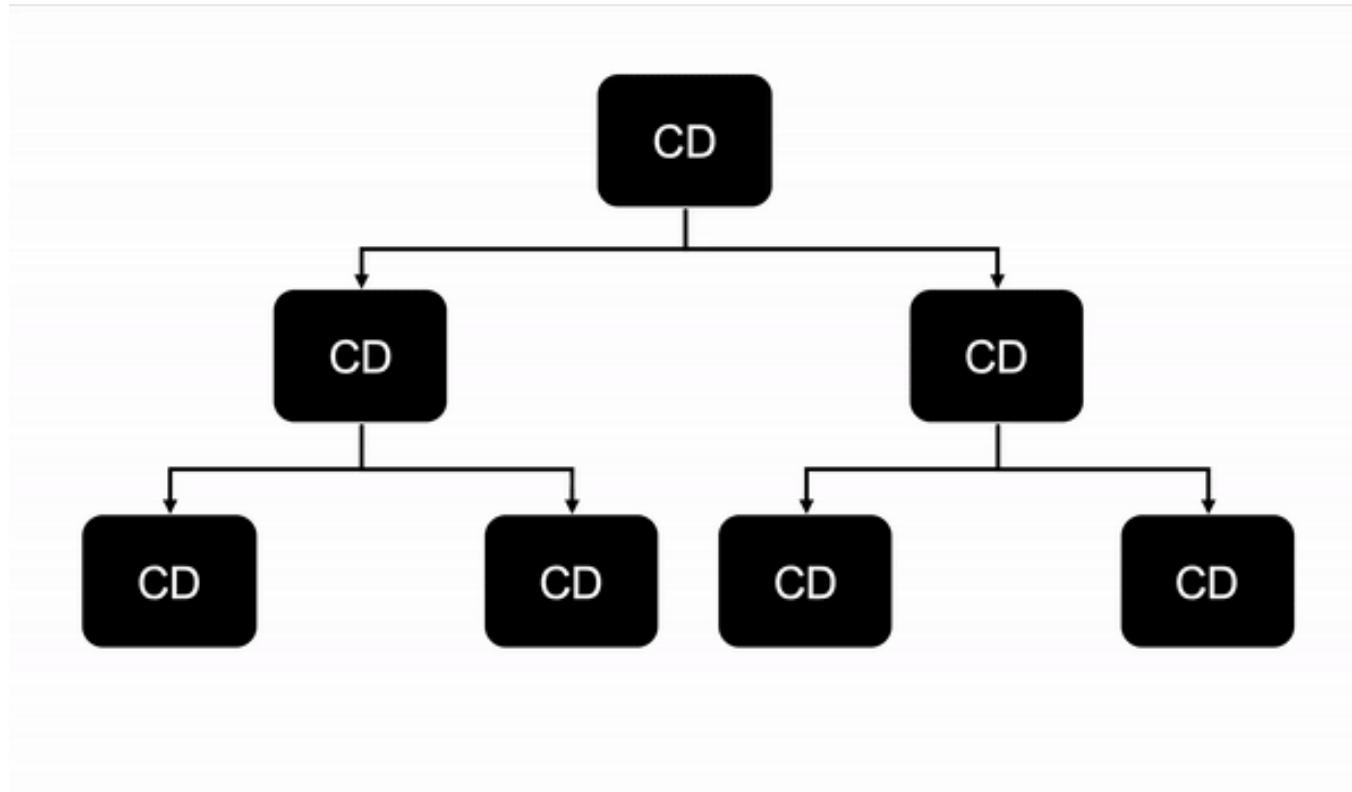
DOM Rendering



Change Detection

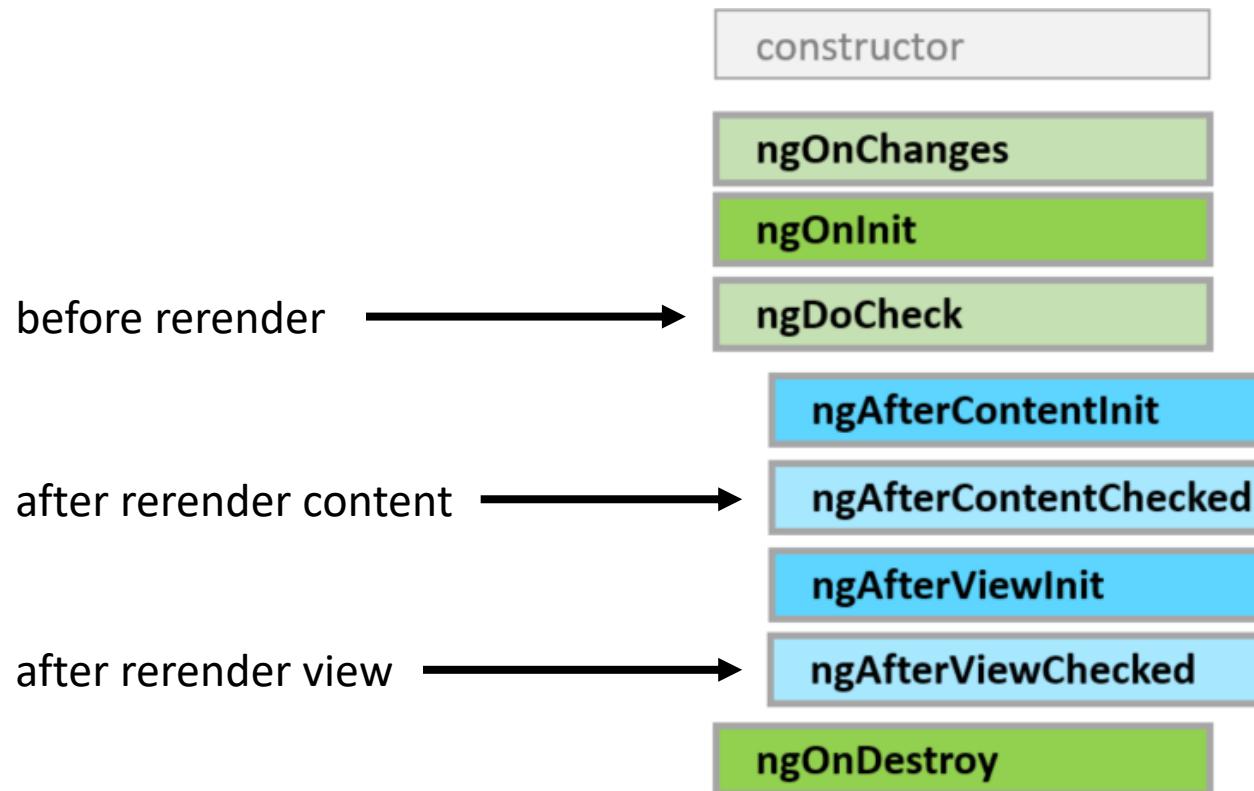
- 1.) User or App changes the model (e.g. input, blur or click)
- 2.) NG CD checks for **every component (from root to leaves)** if the corresponding component model has changes and thus its view (DOM) needs to be updated
- 3.) If yes then update / rerender the component's view (DOM)

Change Detection – From Root To Leaves



<https://mokkapps.de/blog/the-last-guide-for-angular-change-detection-you-will-ever-need/>

Change Detection – Rerender Components



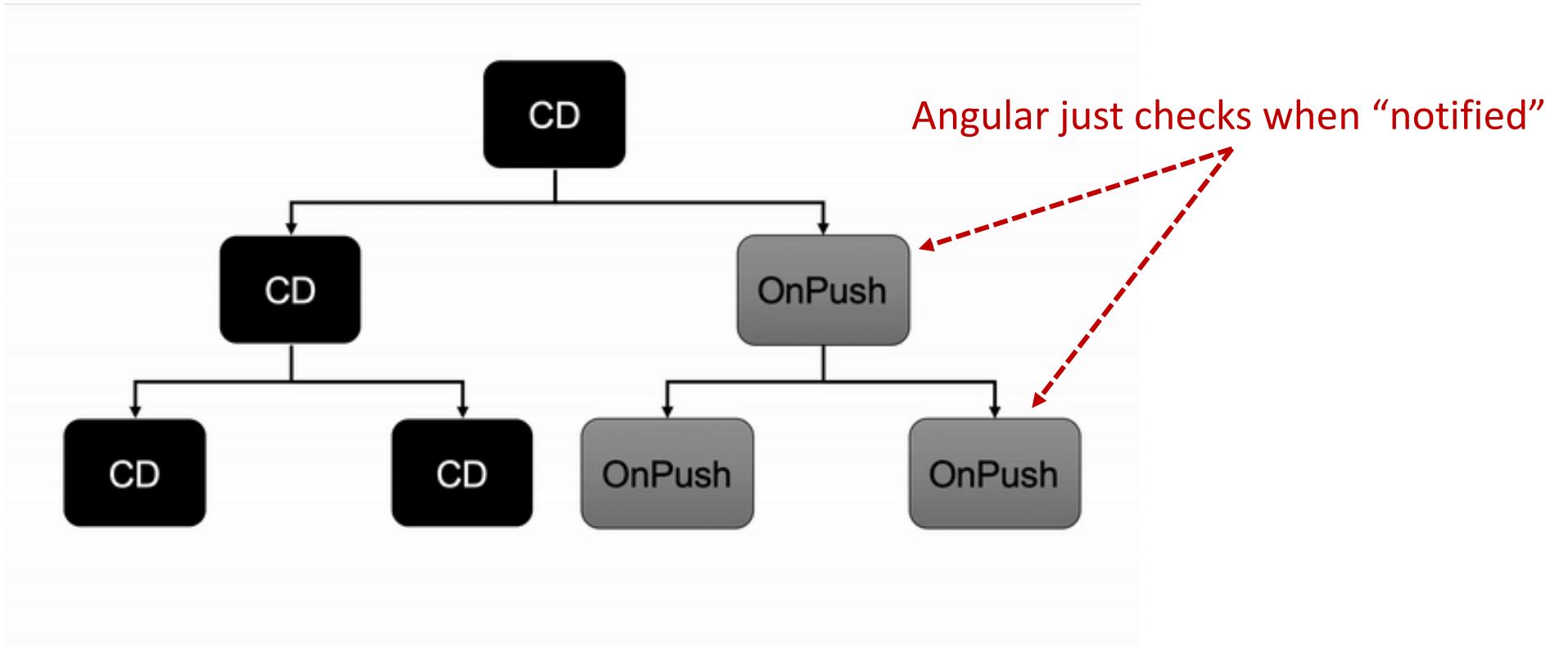
@LX_T

DEMO



Performance- Tuning with OnPush

Change Detection – OnPush Strategy



<https://mokkapps.de/blog/the-last-guide-for-angular-change-detection-you-will-ever-need/>

Activate OnPush Strategy

```
@Component({
  [...]
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class FlightCardComponent {
  [...]
  flight = input.required<Flight>();
}
```

DEMO – ChangeDetection

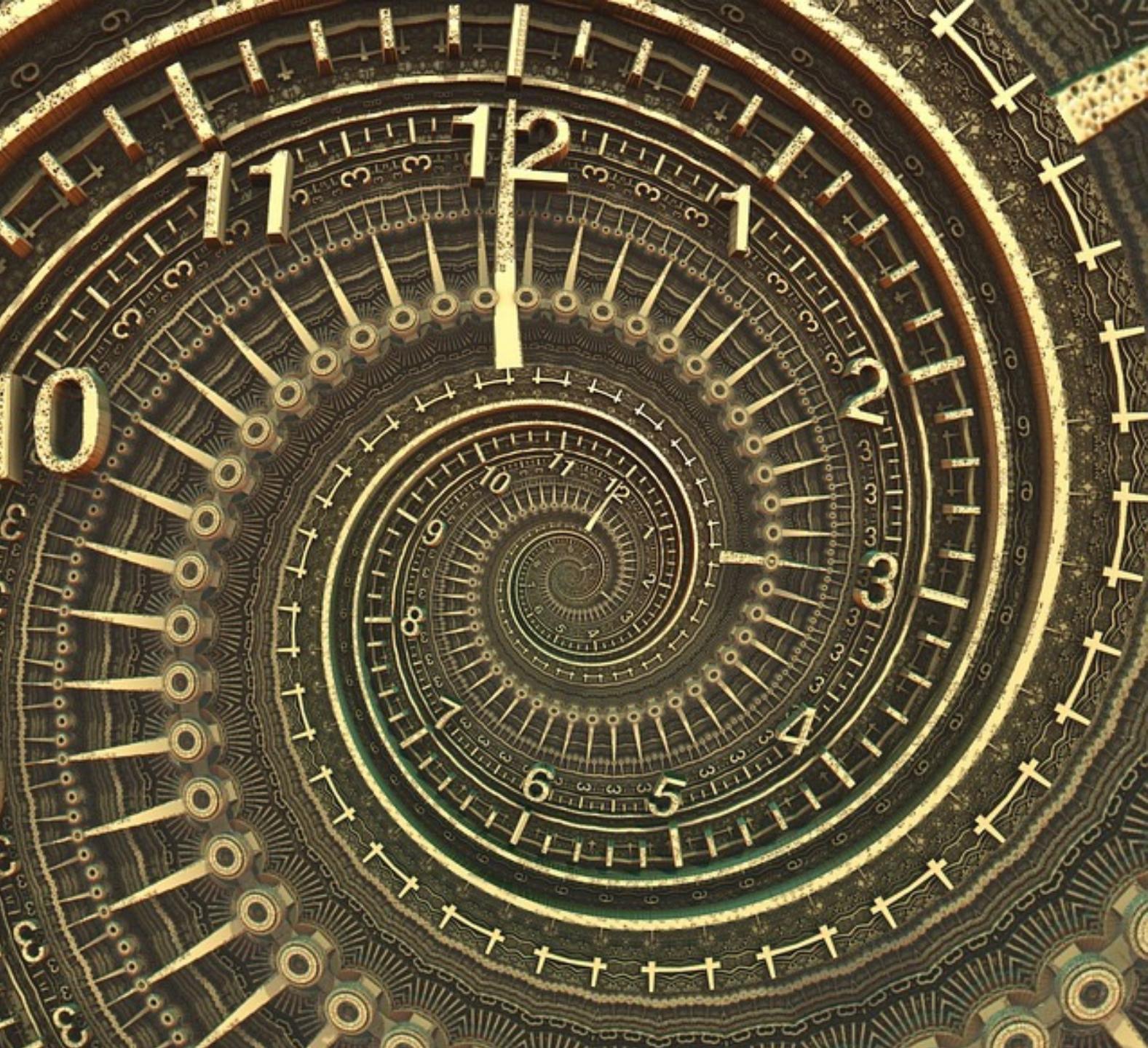
"Notify" about change?

- 1 Change bound data (**@Input**)
 - OnPush: Angular just compares the object reference!
 - e. g. `oldFlight !== newFlight` (BTW: like `ngOnChanges`)
- 2 Raise event within the component and its children (e.g. **@Output**)
- 3 Emit in a bound observable into the `async` pipe | or update a signal
 - `{{ flights$ | async }} | {{ flights() }}`
- 4 Do it manually (`cdr.markForCheck()`)
 - Don't do this at home ;-)
 - But there are reasonable cases (where we can neither use 1 nor 3)

DEMO – ChangeDetection

LAB

Ahead of Time (AOT) Compilation

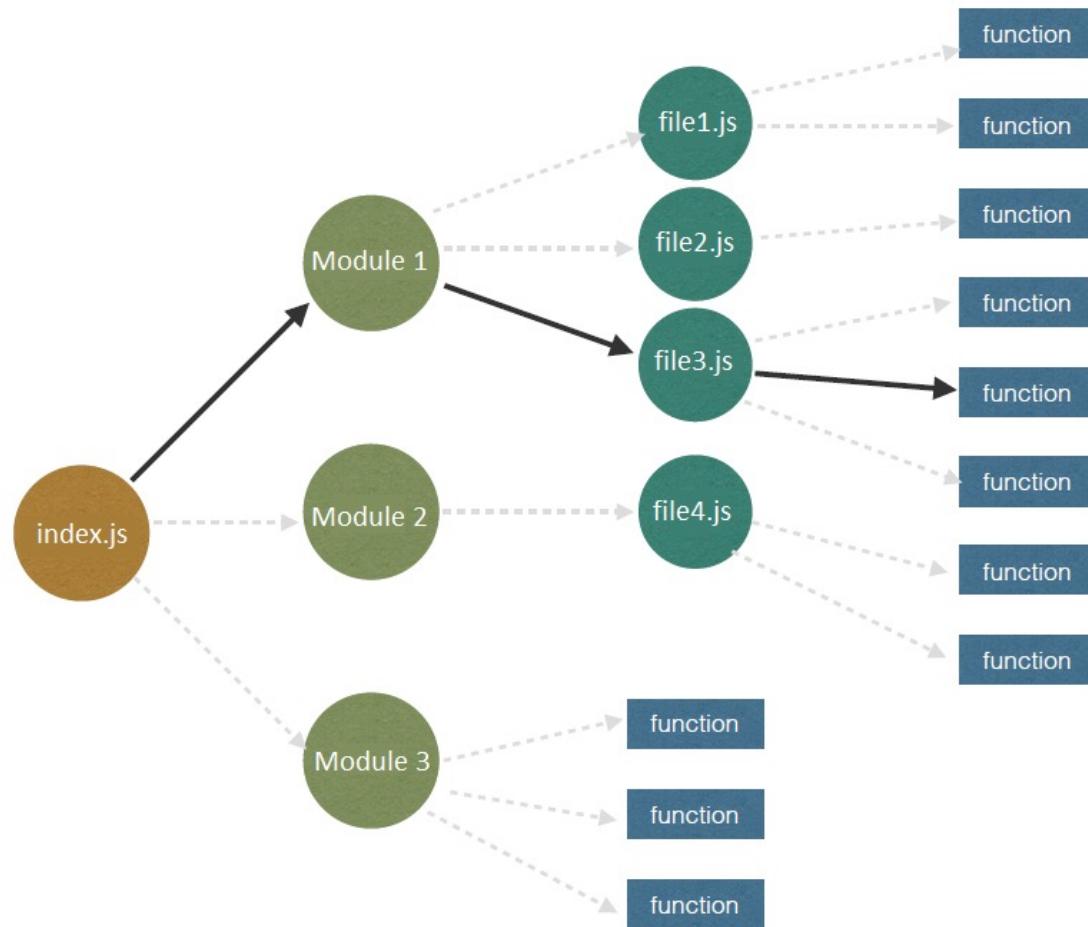


Advantages of AOT

- Ivy makes AOT the default ☺
 - Default for apps since NG 10
 - Default for libs default since NG 12
- Tools (e.g. Webpack) can easier analyse the code
- Smaller bundles → better Startup-Performance
 - You don't need to include the compiler!
 - **Tree Shaking:** Remove unused parts of framework & libs

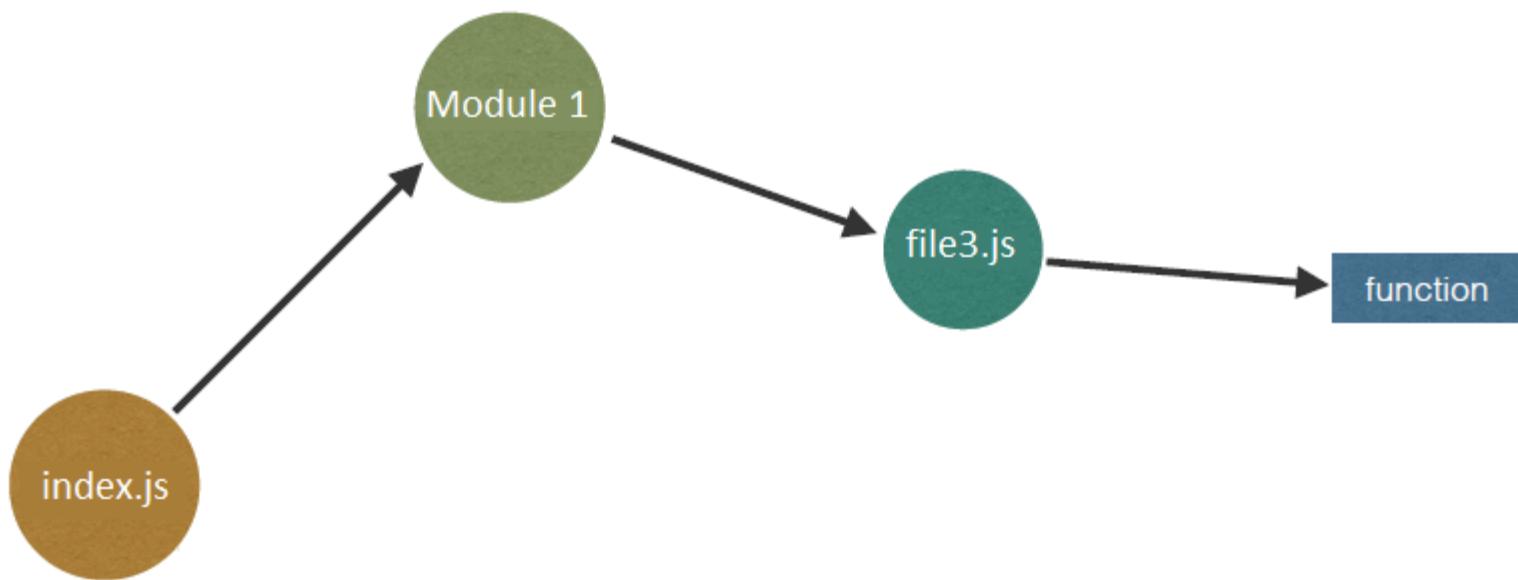
Tree Shaking

Before Tree Shaking



Tree Shaking

After Tree Shaking



@LX_T

Webpack Bundle Analyzer

Bundles without AOT and Tree Shaking

vendor.978ac3ef762178ef4aa8.bundle.js

node_modules

JIT Compiler

@angular

platform-browser-dynamic

esm5

platform-browser-dynamic.js
+ 1 modulescore
esm5

core.js

router.js +
23 modules

common.js http.js

common
esm5forms
esm5forms.js +
2 modulesplatform-browser
esm5http
esm5

platform-browser.js http.js

rxjs

_esm5

main.ts
+ 68
modules

polyfills.7c4efb87d4ba5dbbc58c.bundle.js

node_modules

zone.js

dist

zone.js



DEMO

Conclusion

Quick Wins

Lazy Loading
& Preloading

OnPush w/
Immutables and
Obs. or Signals

Build
Optimization &
Treeshaking



@LX_T

For a performance deep dive
Check out my special workshop

<https://www.angulararchitects.io/schulungen/angular-performance-workshop/>