



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Performance Tuning

Alex Thalhammer

Turbo Button



Contents

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

Lazy Loading

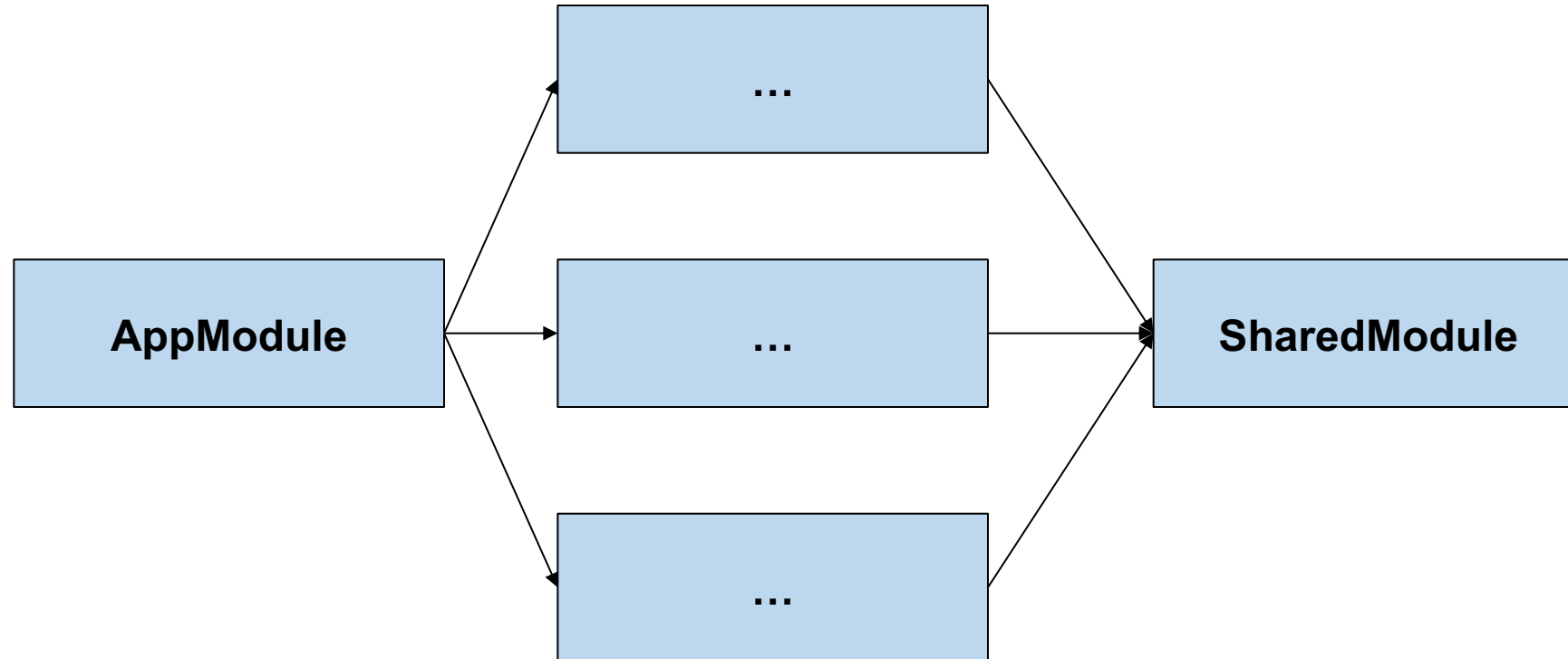


Why Lazy Loading?

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



Module Structure

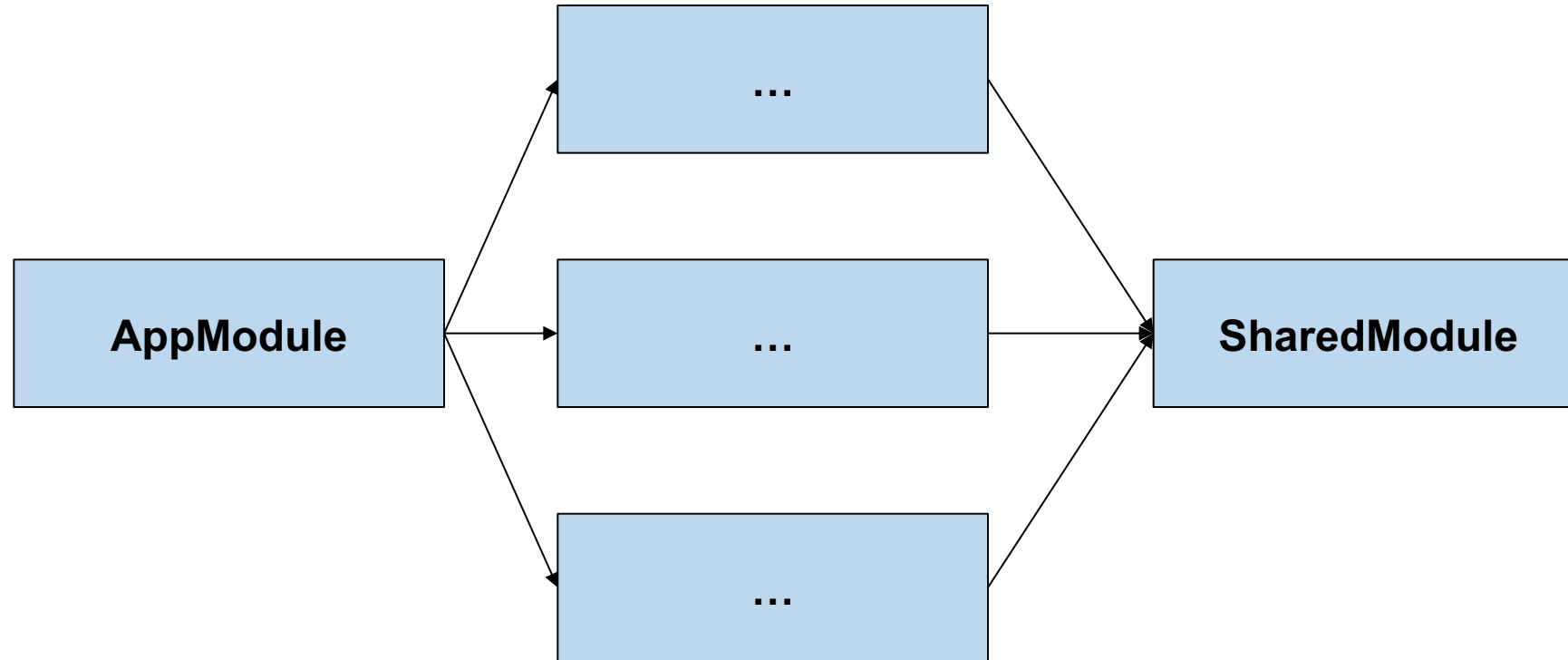


Root Module

Feature Modules

Shared Module

Lazy Loading



Root Module

Feature Modules

Shared Module

Root Module with Lazy Loading

```
const APP_ROUTES: Routes = [  
  {  
    path: 'home',  
    component: HomeComponent  
  },  
  {  
    path: 'flight-booking',  
    loadChildren: () => import('./flight-booking/flight-booking.module')  
      .then(m => m.FlightBookingModule)  
  }  
];
```



Routes for "lazy" Feature Module

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

flight-booking/flight-search

Triggers Lazy Loading w/ loadChildren



DEMO



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Lazy Loading

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



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 accessible

Activate Preloading

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



DEMO



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Intelligent Preloading with ngx-quicklink

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

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

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



Or CustomPreloadingStrategy

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



LAB

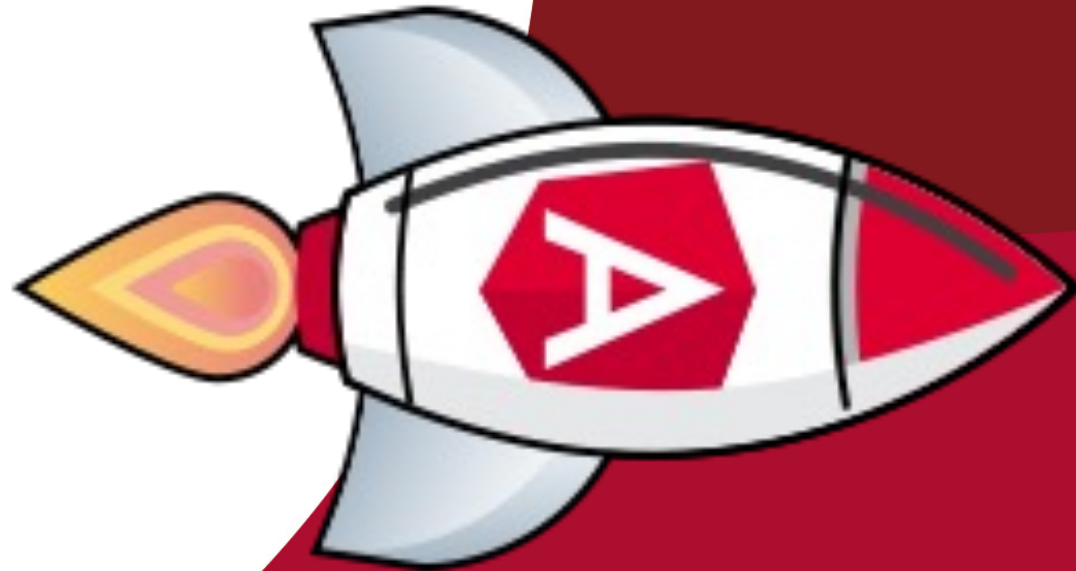


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

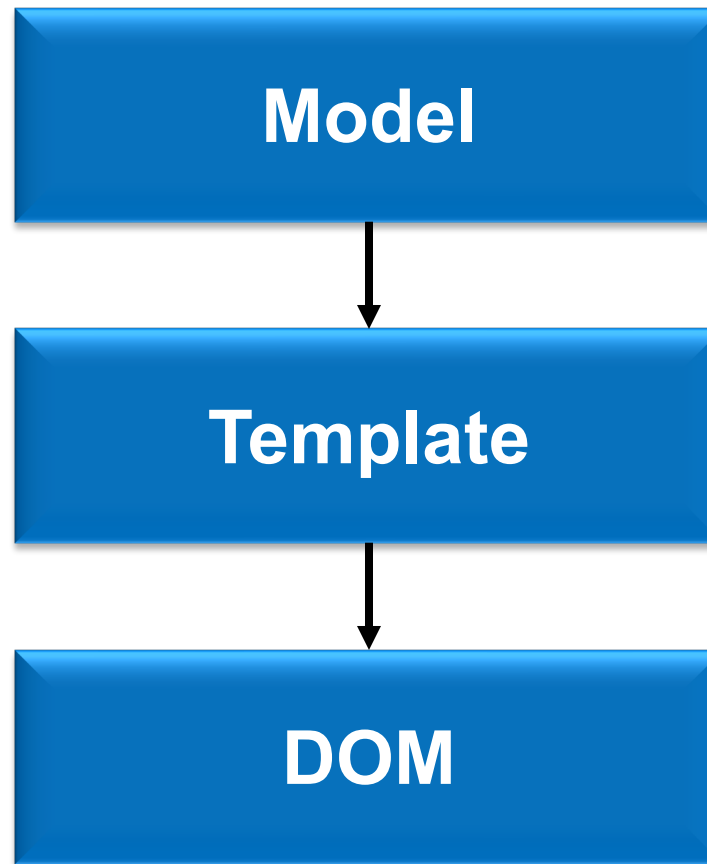


SOFTWARE
ARCHITECT

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)

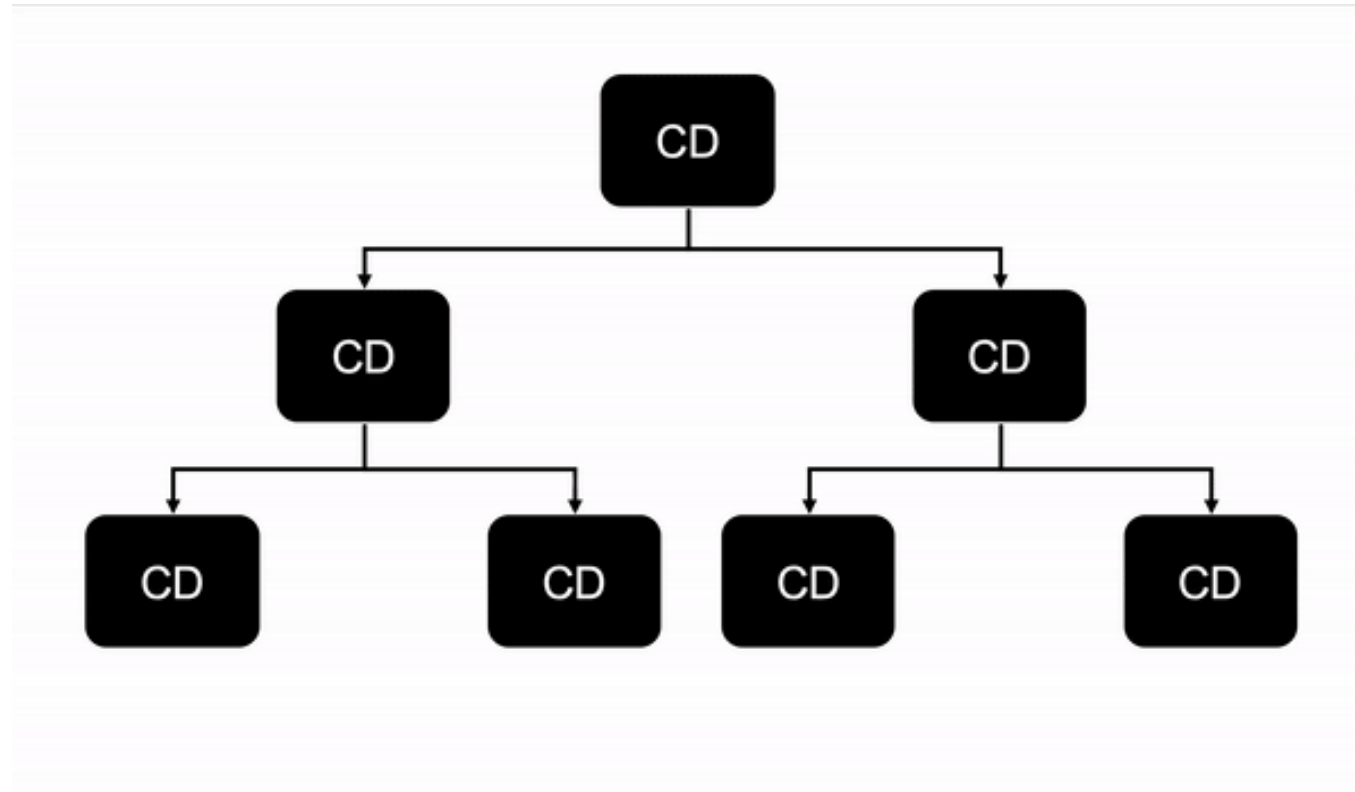


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



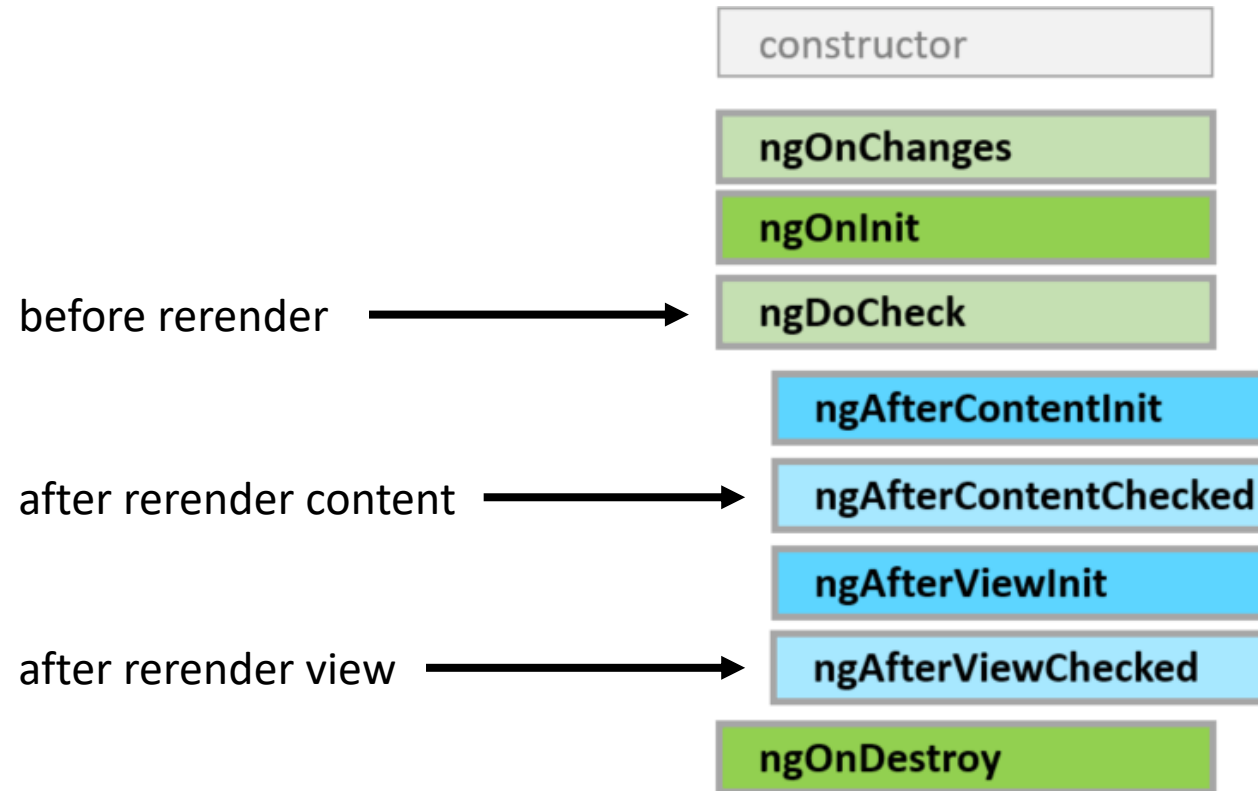
SOFTWARE
ARCHITECT

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



DEMO



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

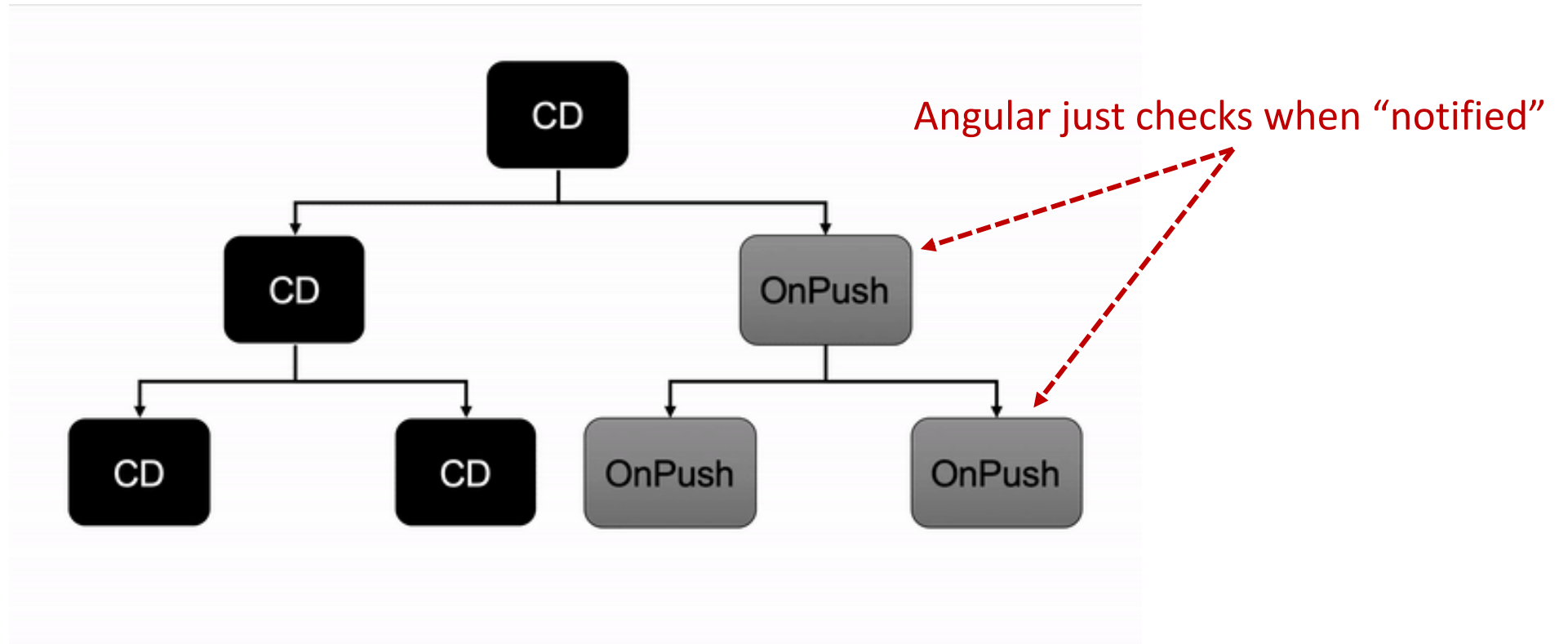


SOFTWARE
ARCHITECT



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 FlightCard {  
  [...]  
  @Input({ required: true }) flight!: Flight;  
}
```



"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 }}` | `{{ flightsSignal() }}`
- 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)



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

DEMO – ChangeDetection



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

LAB

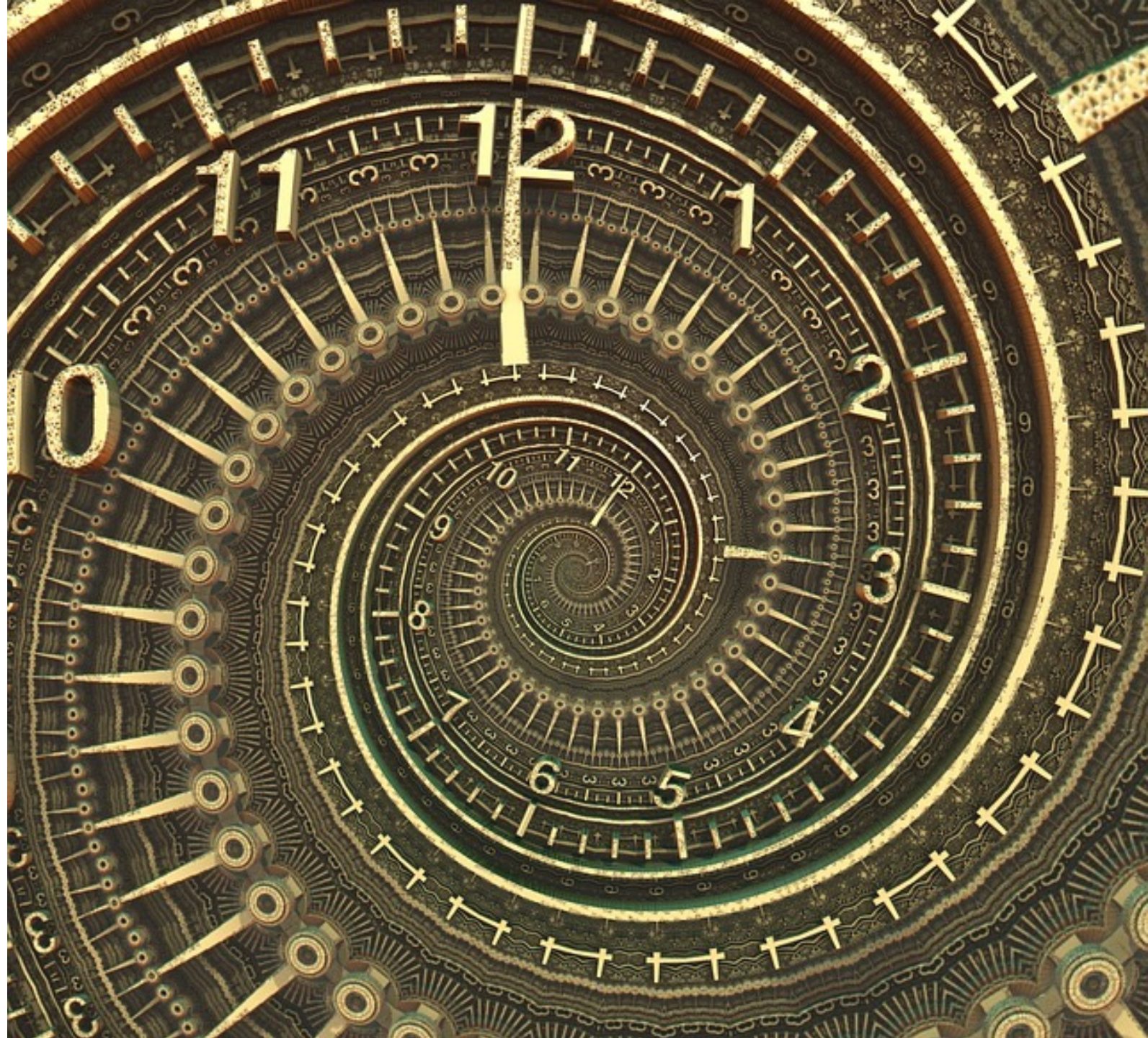


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

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

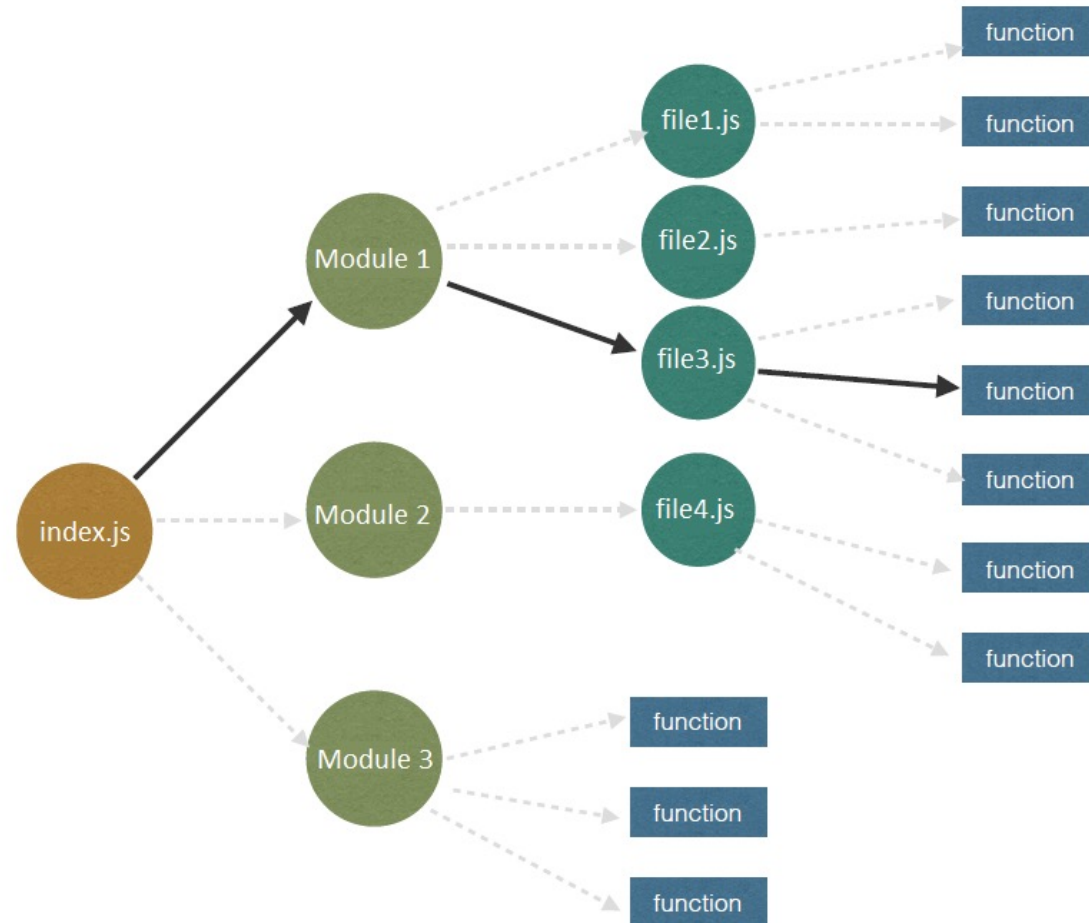


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



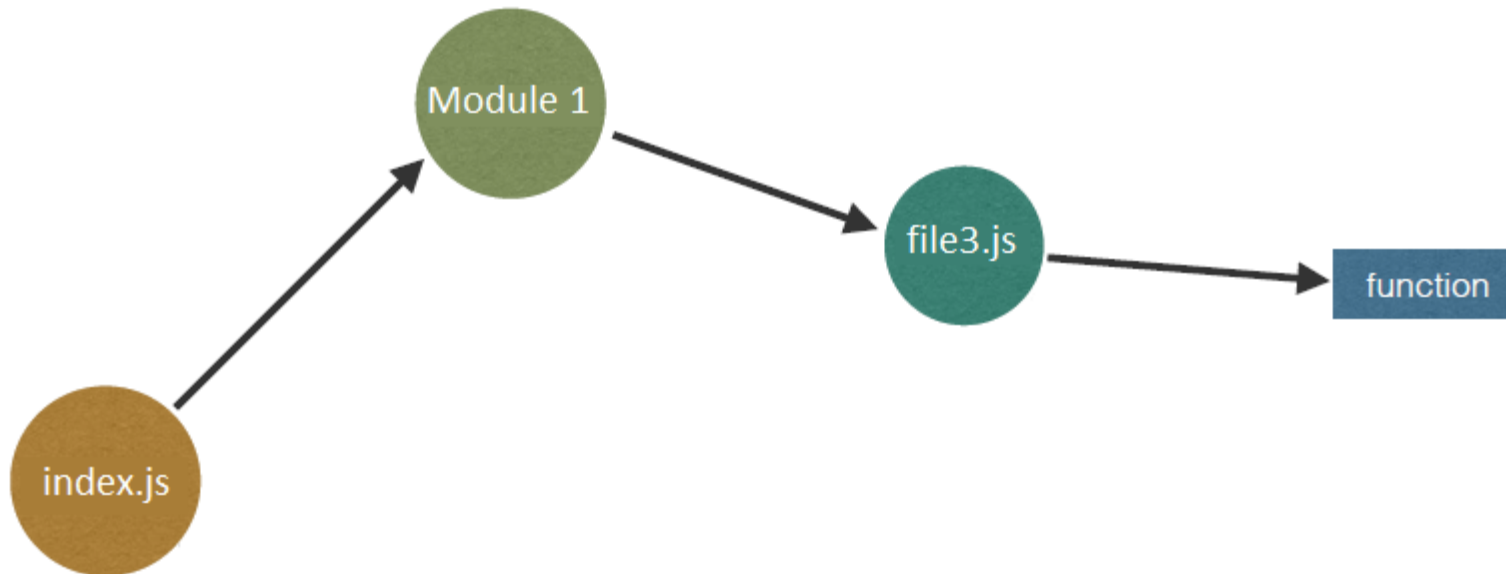
Tree Shaking

Before Tree Shaking



Tree Shaking

After Tree Shaking



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Webpack Bundle Analyzer



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Bundles without AOT and Tree Shaking

vendor.978ac3ef762178ef4aa8.bundle.js

node_modules

JIT Compiler

@angular

platform-browser-dynamic

esm5

platform-browser-dynamic.js
+ 1 modules

core

esm5

core.js

router

esm5

router.js +
23 modules

common

esm5

common.js

http.js

forms

esm5

forms.js +
2 modules

platform-browser

esm5

platform-browser.js

http

esm5

http.js

rxjs

_esm5

add

delay.js + 2

modules

switchMap.js

+ 2 modules

fromEvent.js

+ 2 modules

mergeMap.js

+ 2 modules

share.js

+ 4 modules

merge.js

+ 2 modules

Subscriber.js

+ 1 modules

mergeMap.js

+ 1 modules

AsyncAction.js

+ 1 modules

ReplaySubject.js

+ 3 modules

Subscription.js

+ 1 modules

Subject.js

+ 1 modules

Observable.js

+ 1 modules

src

main.ts
+ 68
modules

polyfills.7c4efb87d4ba5dbbc58c.bundle.js

node_modules

zone.js

dist

zone.js

core-js

modules



FoamTree

DEMO



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Conclusion

Quick Wins

Lazy Loading
& Preloading

OnPush w/
Immutables and
Observables

Build
Optimization &
TreeShaking



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

For a performance deep dive
Check out my special workshop

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

