ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Performance Tuning

Alex Thalhammer


Turbo Button
HIGHSCREEN
386 SX-33
POWER
HDD
TURBO
RESET

Contents

- Startup Performance
  - Lazy Loading and Preloading
  - (Manual) Build Analyzing & Optimization

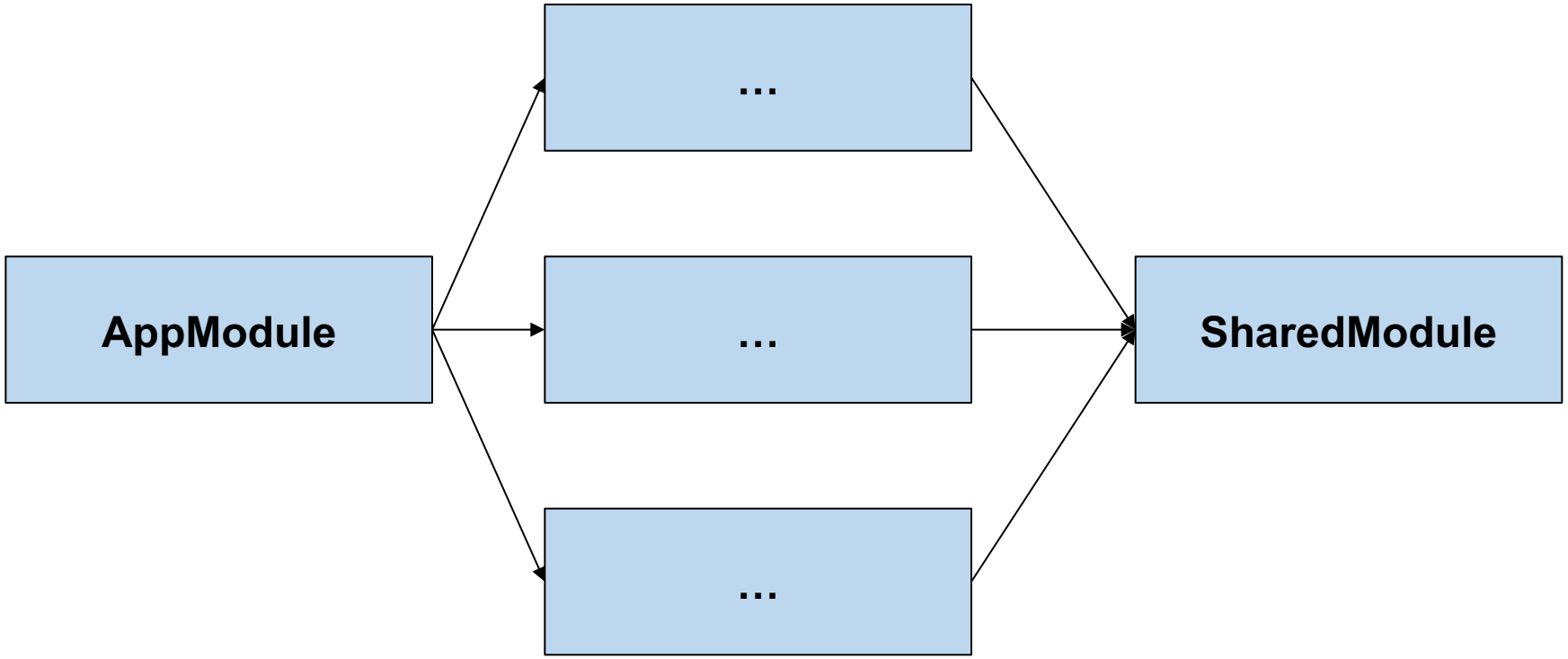
- Runtime Performance
  - Data Binding with ChangeDetection.OnPush

Lazy Loading

Why Lazy Loading?

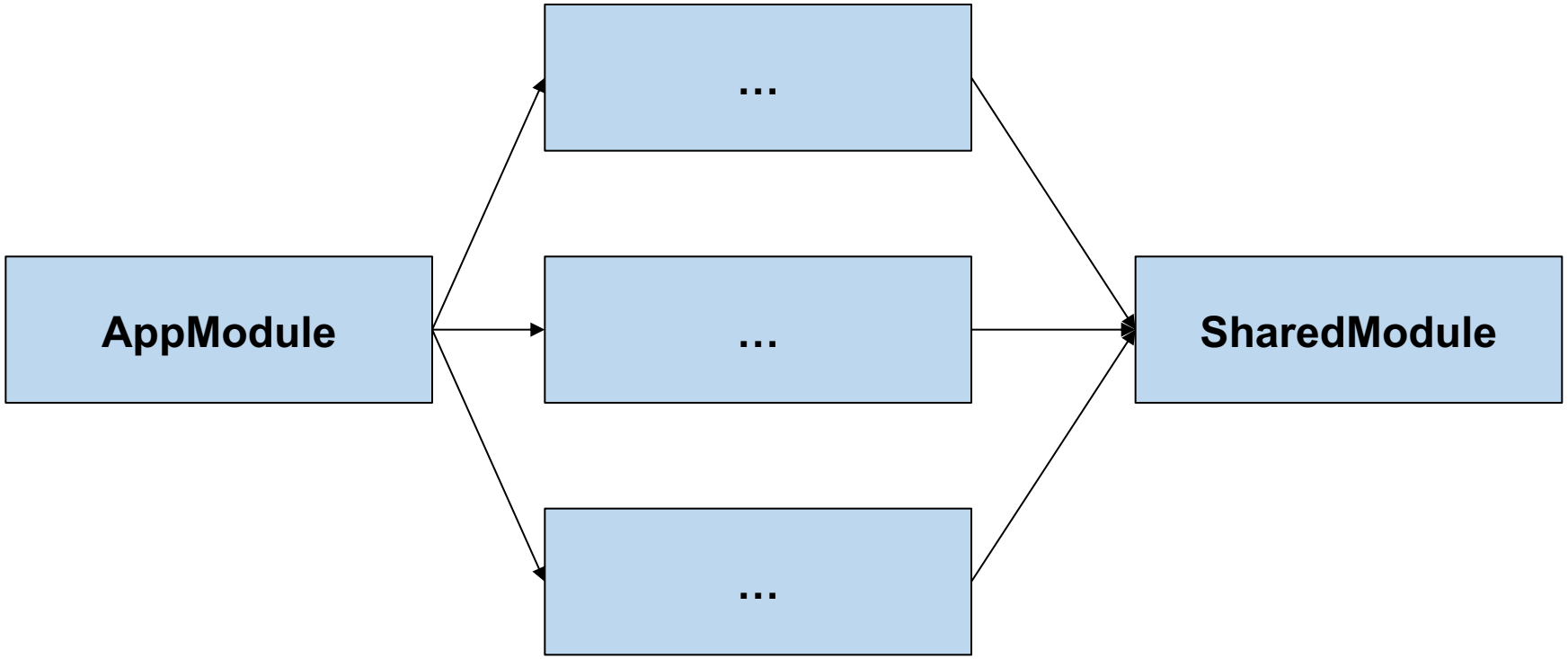
- Improve initial load time (performance → very important!)

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,
        […]
    },
    […]
}
```

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 accessable

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

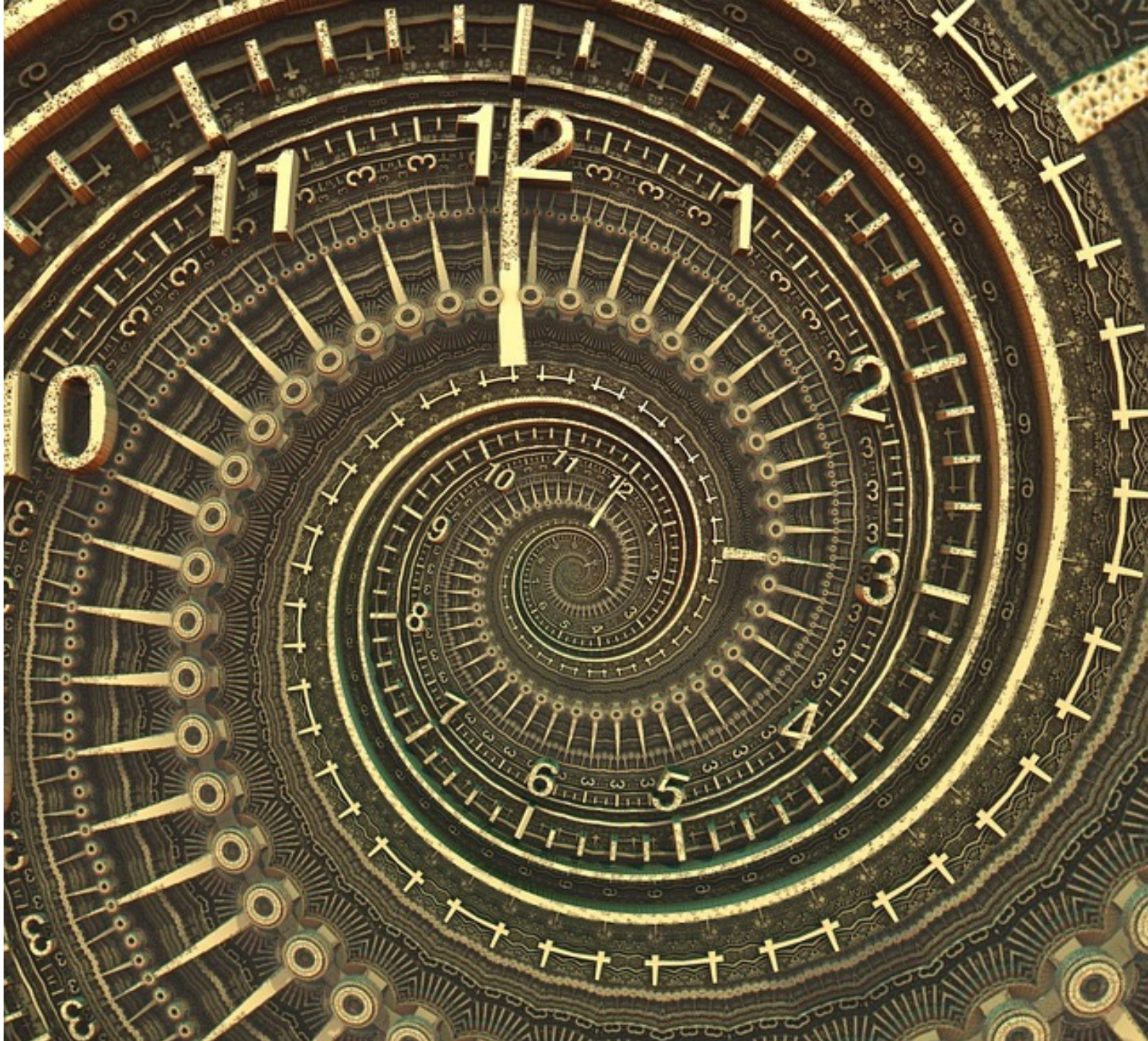
```
…
imports: [
    […]
    RouterModule.forRoot(
        ROUTE_CONFIG,
        { preloadingStrategy: CustomPreloadingStrategy }
    );
]
…
```

LAB

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Ahead of Time (AOT) Compilation



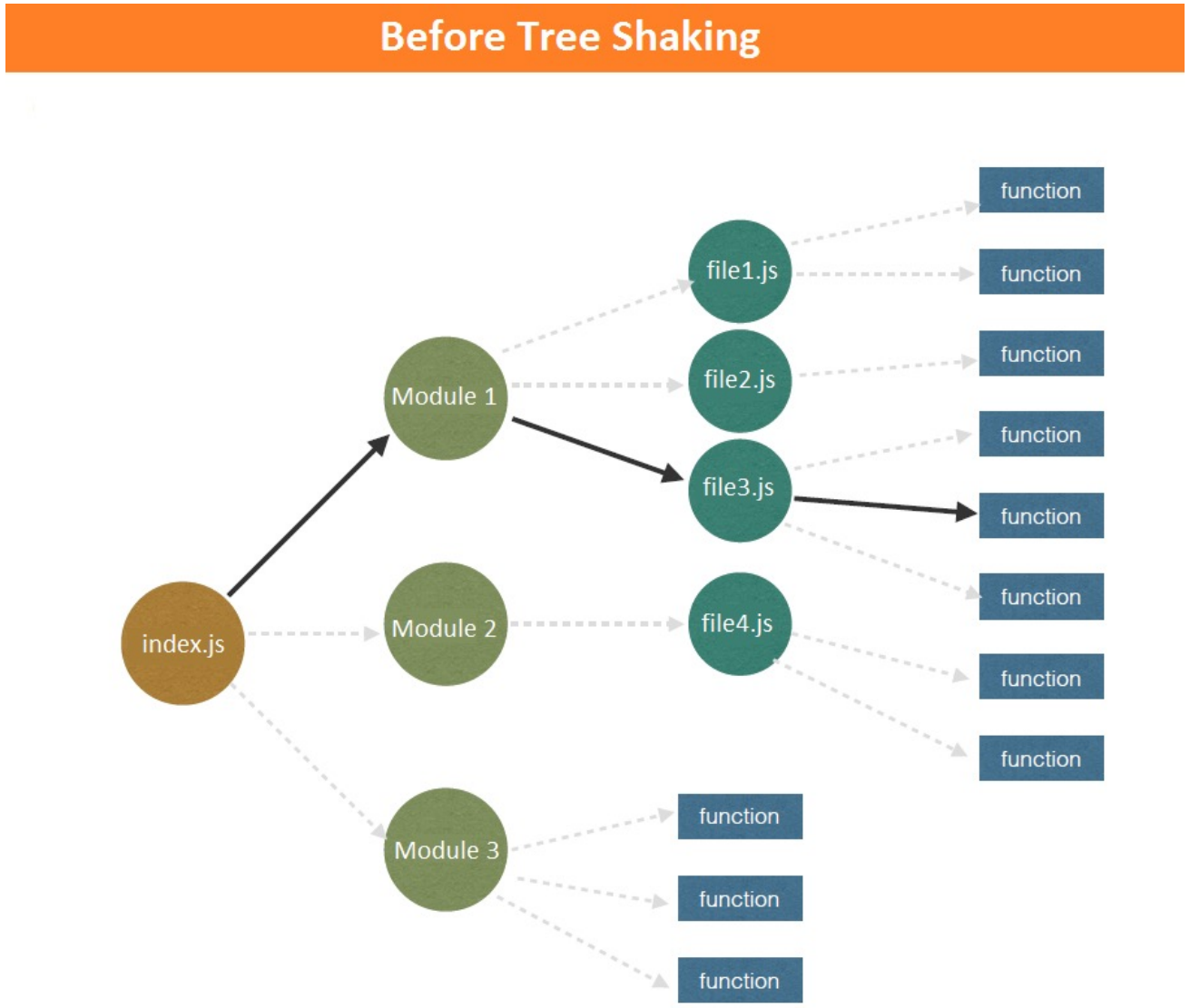
Advantages of AOT

- Ivy makes AOT the default ☺
  - Default for apps since NG 10
  - Default for libs default since NG 12

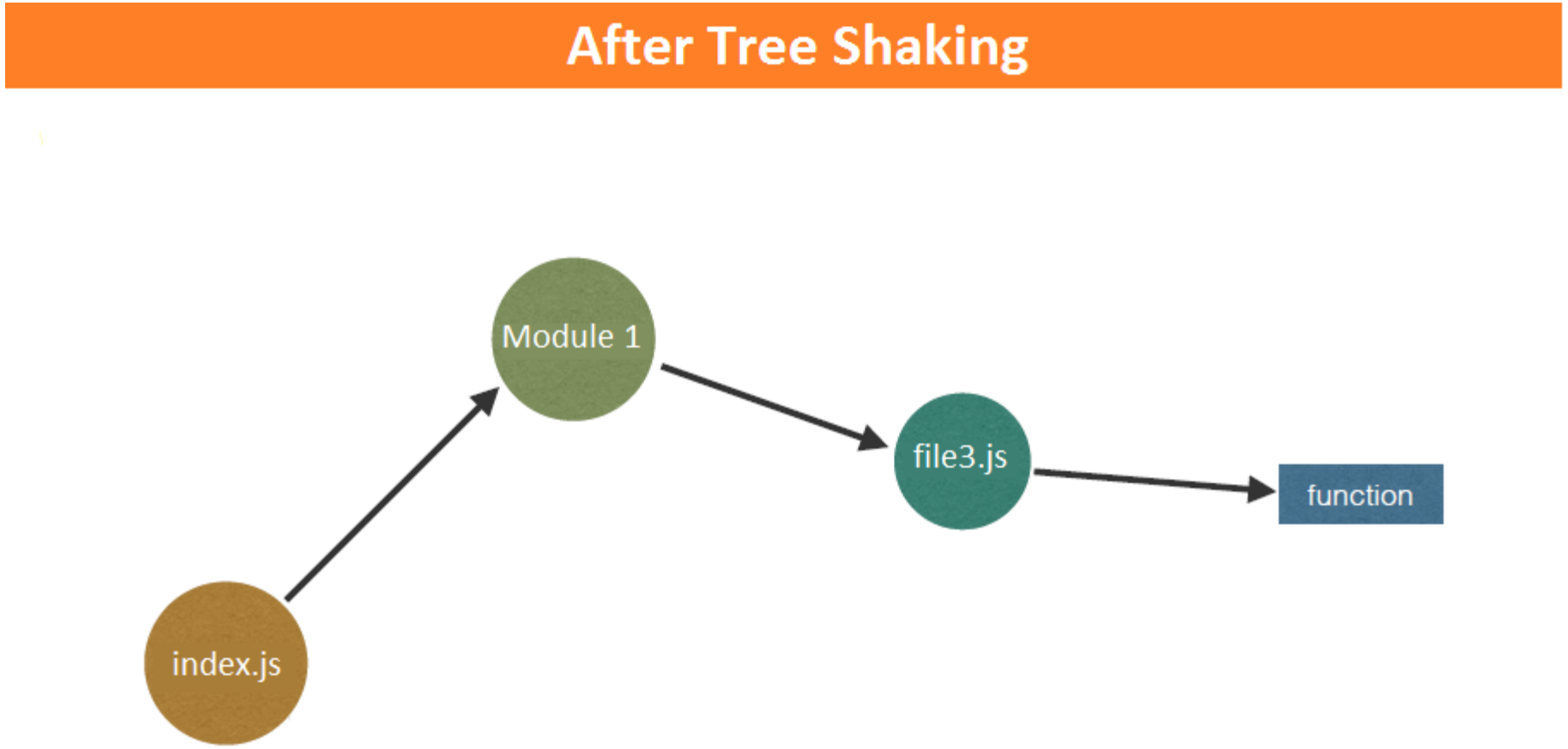
- Tools (e.g. Webpack) can easier analyse the code

- Smaller bundles → better Startup-Performance
  - You don't need to include the compiler!
  - **Tree Shaking**: Remove unused parts of framework & libs

Tree Shaking



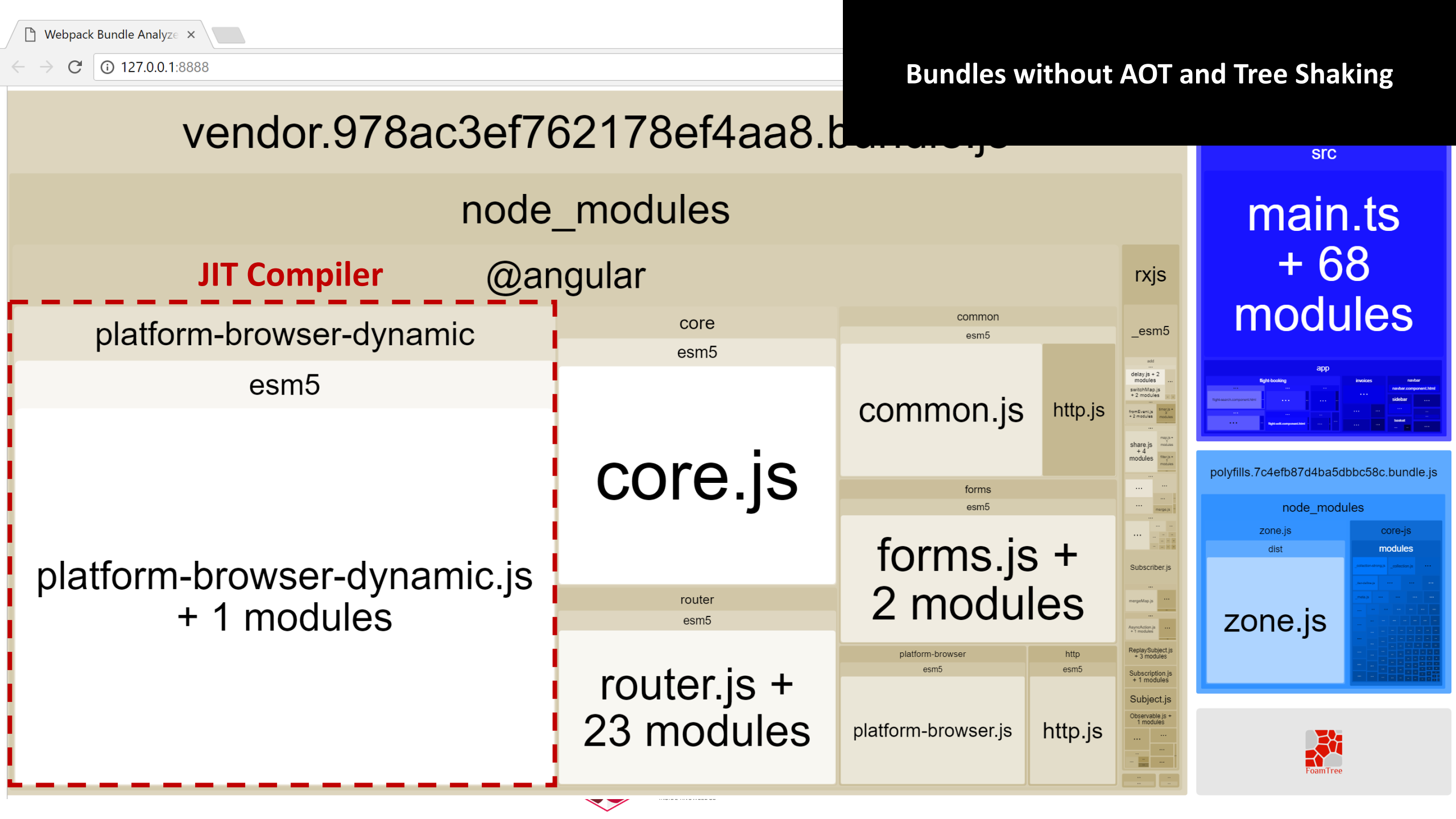
Tree Shaking



Webpack Bundle Analyzer

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

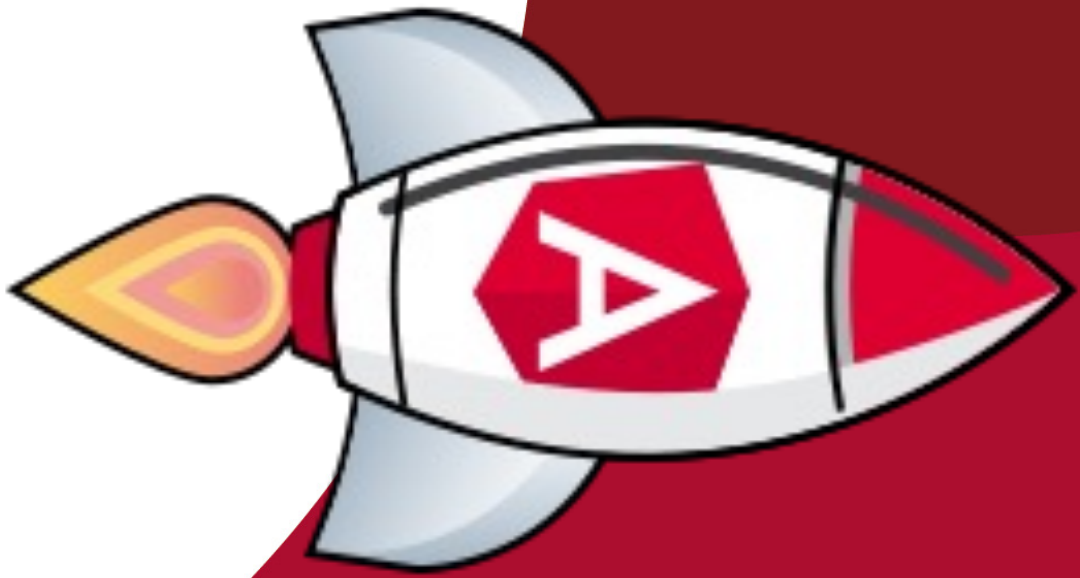

Webpack Bundle Analyze
127.0.0.1:8888
Bundles without AOT and Tree Shaking
vendor.978ac3ef762178ef4aa8.b
node_modules
JIT Compiler
@angular
platform-browser-dynamic
esm5
platform-browser-dynamic.js
+ 1 modules
core
esm5
core.js
router
esm5
router.js +
23 modules
common
esm5
common.js
http.js
forms
esm5
forms.js +
2 modules
platform-browser
esm5
platform-browser.js
http
esm5
http.js
rxjs
_esm5
Subscriber.js
Subject.js
src
main.ts
+ 68
modules
app
polyfills.7c4efb87d4ba5dbbc58c.bundle.js
node_modules
zone.js
dist
zone.js
core-js
modules
FoamTree

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

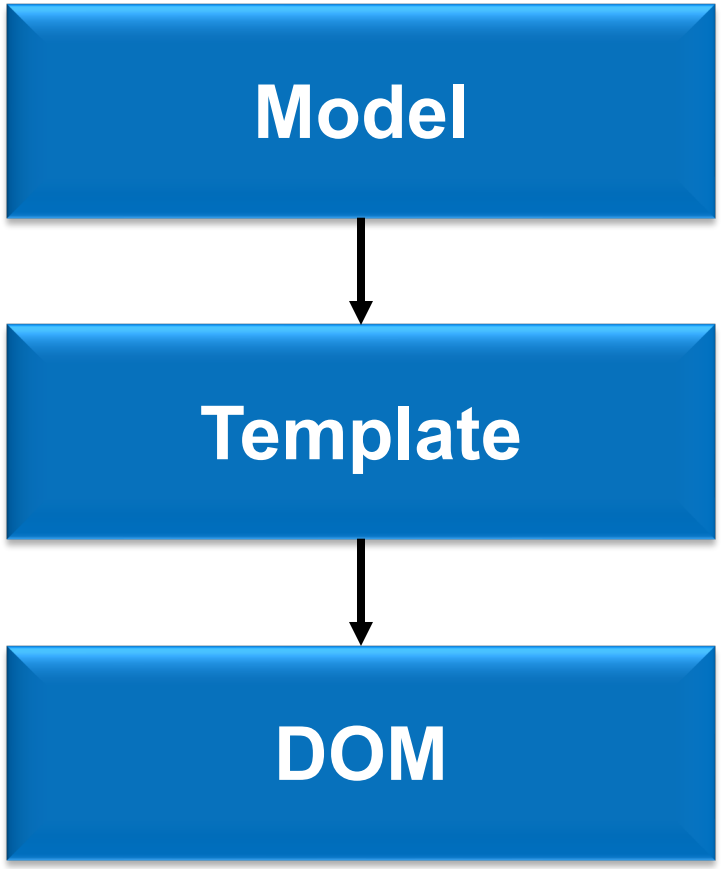
SOFTWARE
ARCHITECT

Change Detection in Angular



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

DOM Rendering



Change Detection

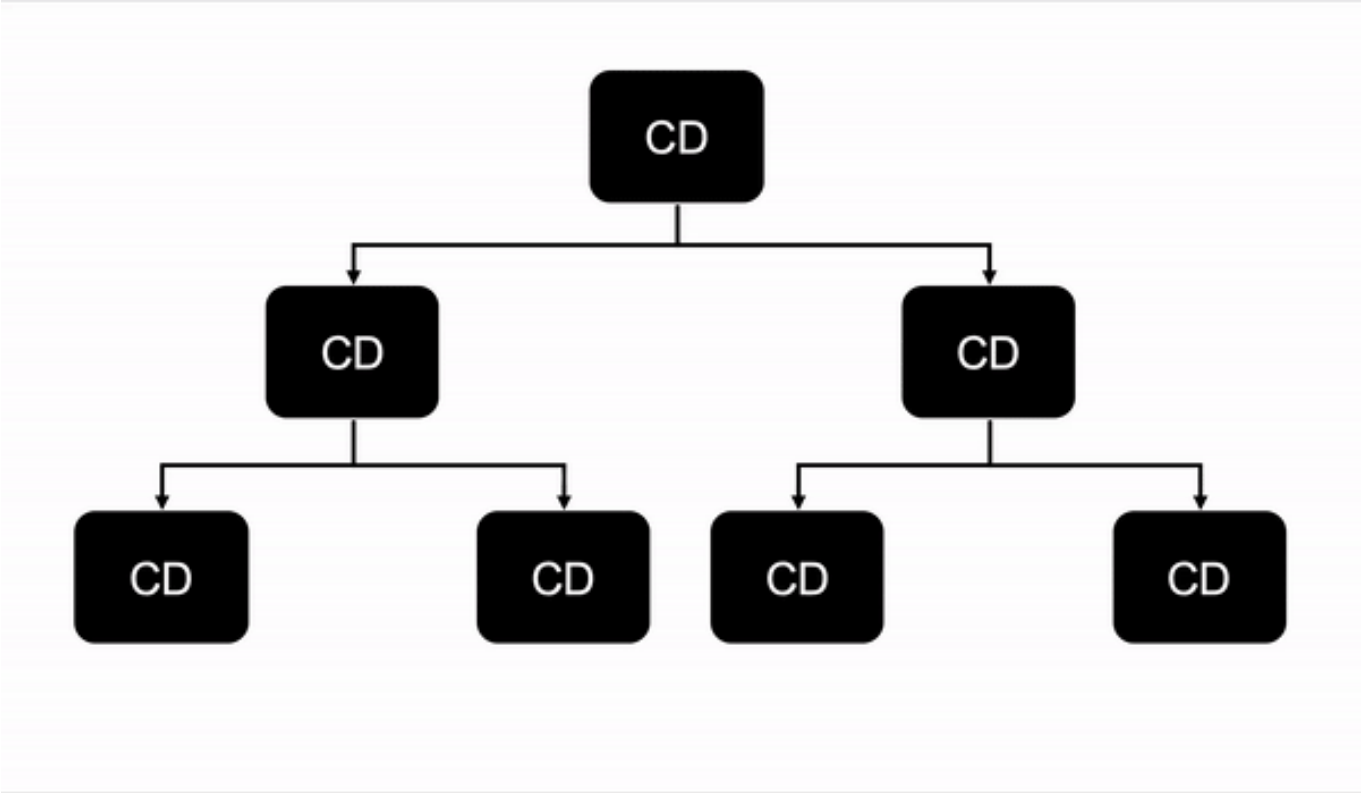
- 1.) User or App changes the model (e.g. @Input() Binding)
- 2.) NG CD checks for every component (from root to leaves) if the corresponding component model has changes and thus its view (DOM) needs to be updated
- 3.) If yes then update / rerender the component’s view (DOM)

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

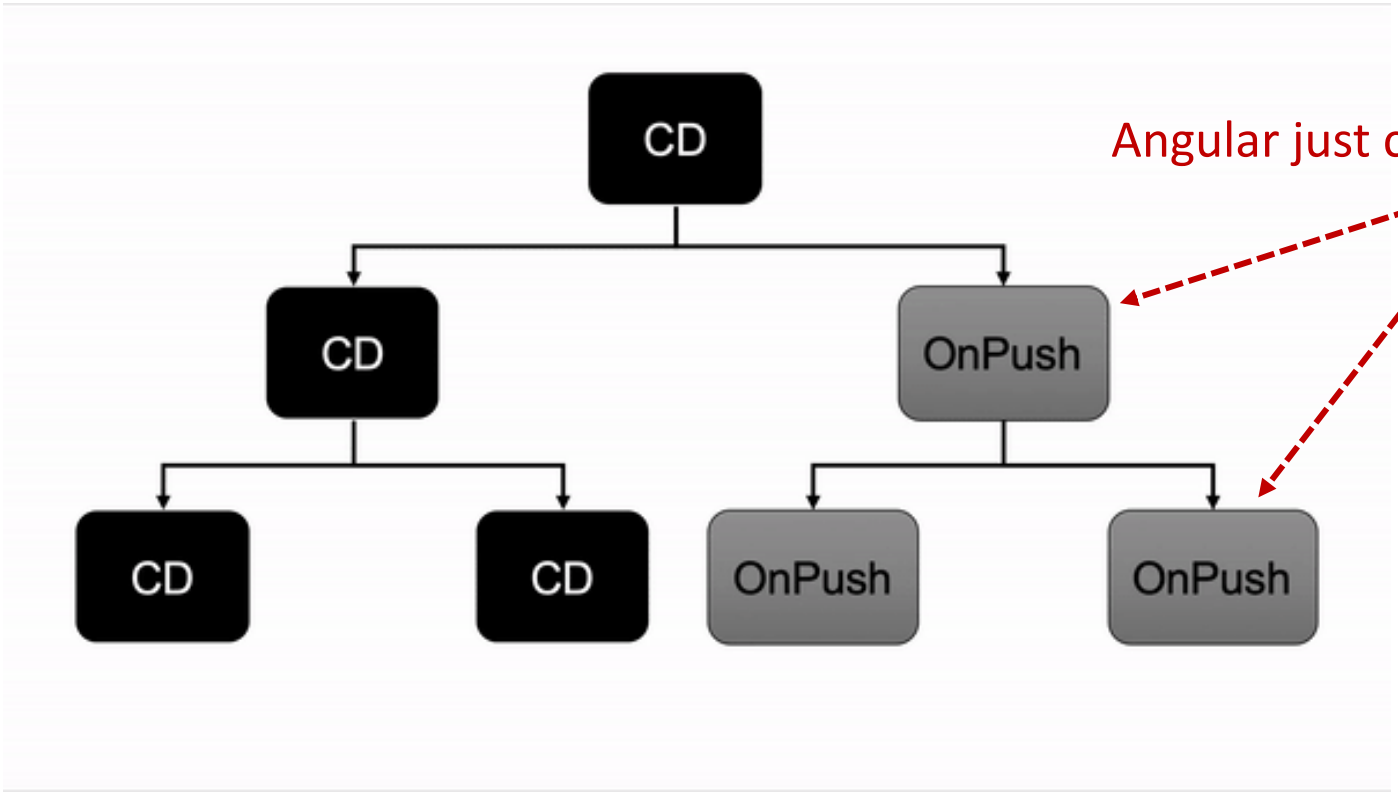
Change Detection – From Root To Leaves



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

Performance-Tuning with OnPush

Change Detection – OnPush Strategy



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

"Notify" about change?

- Change bound data (@Input)
  - OnPush: Angular just compares the object reference!
  - e. g. oldFlight !== newFlight (BTW: like ngOnChanges)
- Raise event within the component and its children (e.g. @Output)
- Emit in a bound observable into the async pipe
  - {{ flights$ | async }}
- Do it manually (cdr.markForCheck())
  - Don't do this at home ;-)
  - Try to avoid this – but there are reasonable cases

Activate OnPush Strategy

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

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

LAB

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Conclusion

Quick Wins

Lazy Loading & Preloading

Build Optimization & Treeshaking

OnPush w/ Immutables and Observables

For a performance deep dive

Check out my special workshop

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