



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Performance Tuning

angular-architects.io

Turbo Button



Quick Wins

Bundling

Minification

enableProdMode()

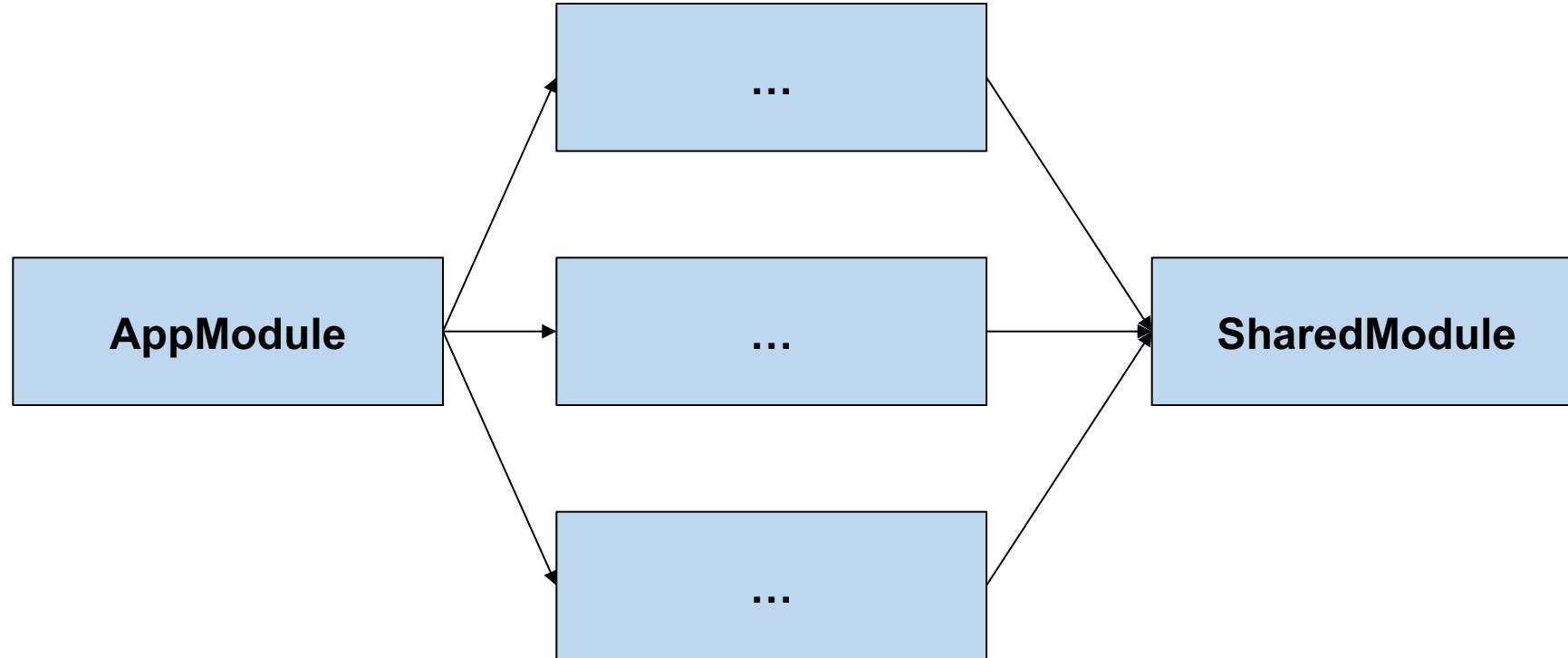
Contents

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

Lazy Loading



Module Structure

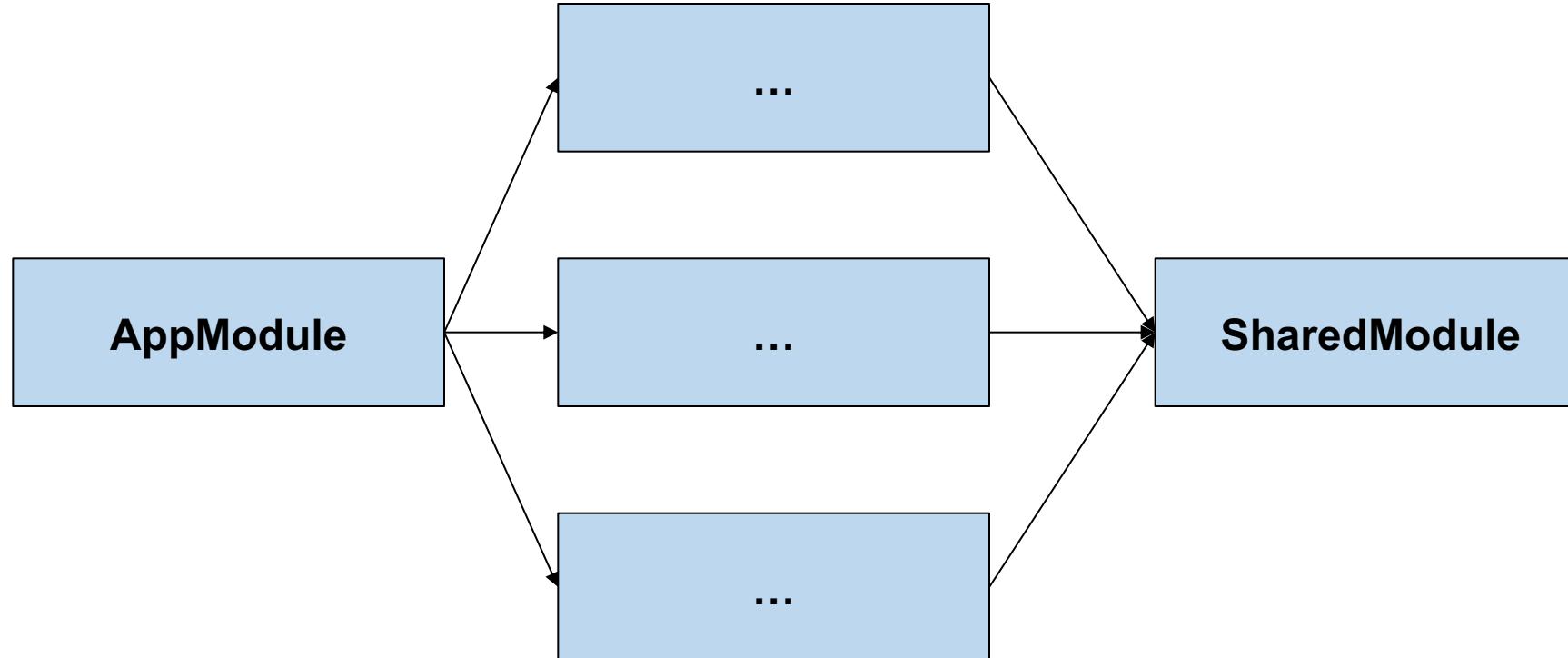


Root Module

Feature Modules

Shared Module

Lazy Loading



Root Module

Feature Modules

Shared Module

Root Module with Lazy Loading

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

Routes for "lazy" Module

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

Routes for "lazy" Module

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

flight-booking/ subroute

Triggers Lazy Loading w/ loadChildren

Lazy Loading

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

Preloading



Idea

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

Activate Preloading

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



Performance- Tuning with OnPush

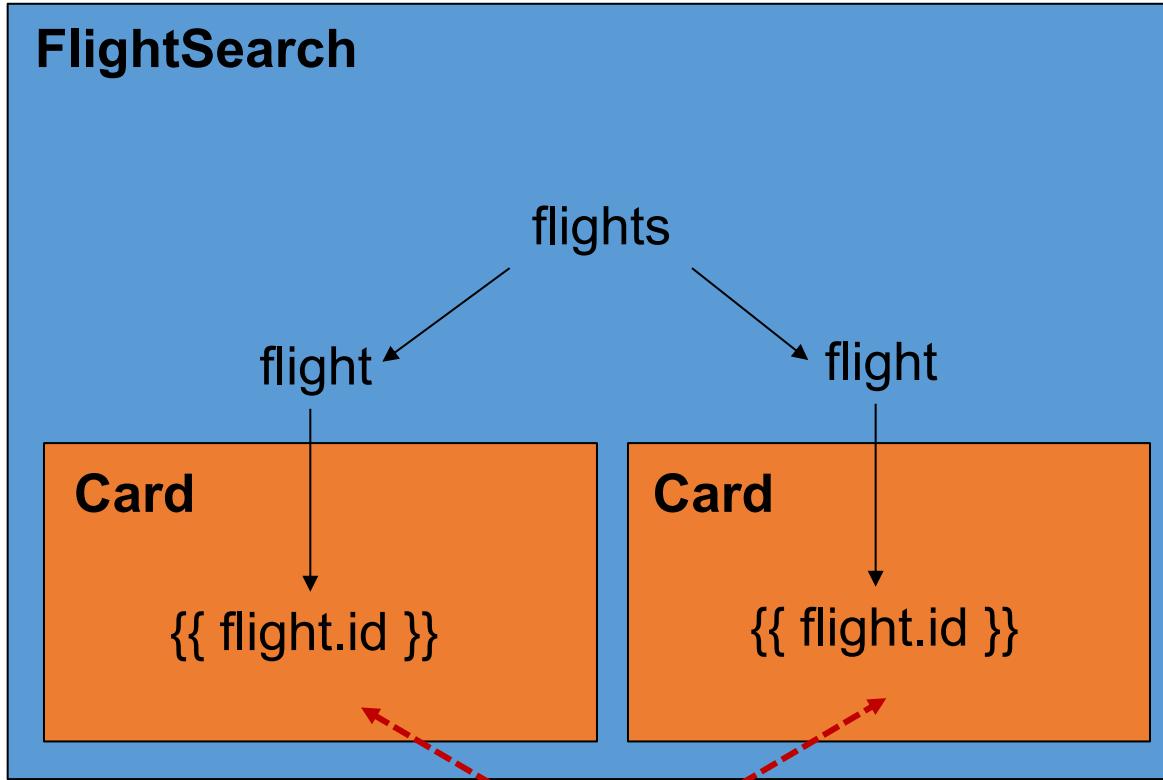
DEMO



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



OnPush



Angular just checks when “notified”



ANGULAR
ARCHITECTS

INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

"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
 - {{ flights\$ | async }}
- Trigger it manually
 - Don't do this at home ;-)
 - At least: Try to avoid this

Activate OnPush

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

DEMO



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE





Server Side
Rendering

Server Side Rendering

- Instead of the empty index.html that sits in our root dir

```
<!DOCTYPE html>
]<html>
]  <head>
|    <meta charset="utf-8" />
|    <title>Flight App</title>
|    <base href="/" />
]  </head>
]  <body>
|    <flight-app></flight-app>
]  </body>
]</html>
```



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



Server Side Rendering

- Each route is being compiled on the server via express.js
- On visit of the URL serve the rendered html to the client

Server Side Rendering - Advantages

- Further decrease initial page load time
 - No processing/rendering on client
- Make URLs (routes) crawlable for
 - Other search engines (not Google)
 - Social Media platforms

Server Side Rendering - Setup

- `ng add @nguniversal/express-engine`
- Deploy it on an node/express.js server

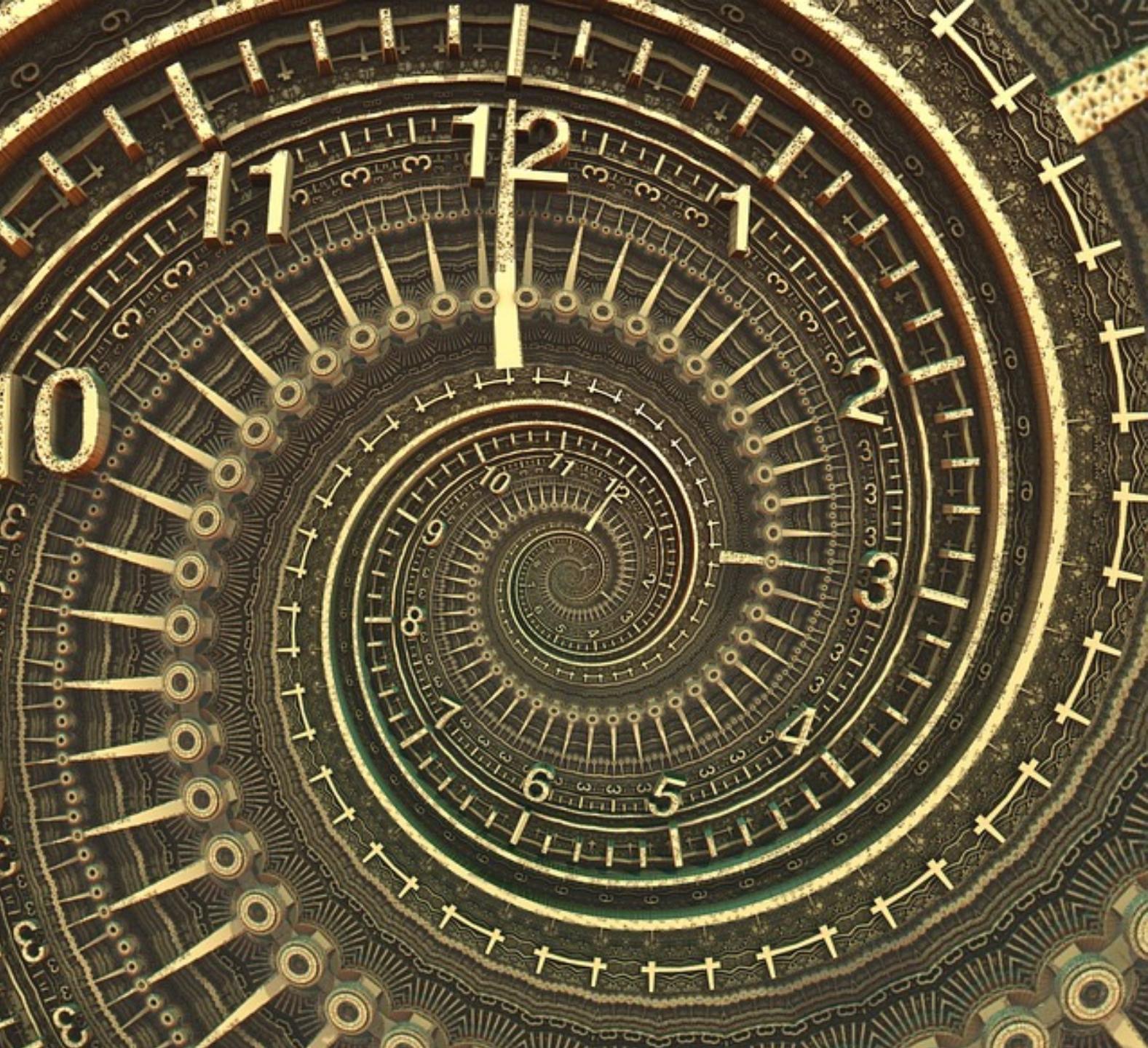
DEMO



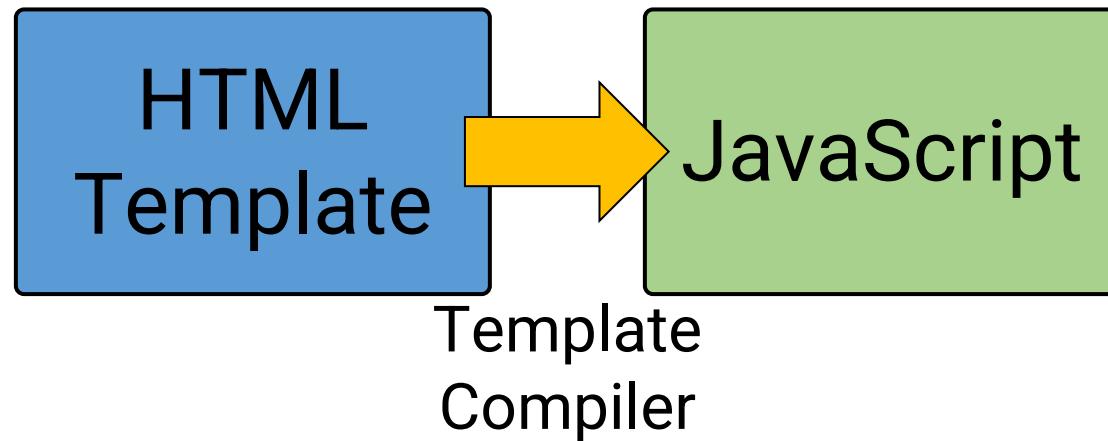
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



Ahead of Time (AOT) Compilation



Angular Compiler



Approaches

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

Advantages of AOT

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

Angular CLI

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

Production Mode without AOT (no lazy loading)

FLIGHT42 Flight42

HOME Flight Search Passenger Search

FLIGHT BOOKING

Flight Search

Elements Console Sources Network **Performance** Memory Application Security Audits Augury > ... X

Screenshots Memory

1000 ms 2000 ms 3000 ms 4000 ms 5000 ms 6000 ms 7000 ms

FPS CPU NET

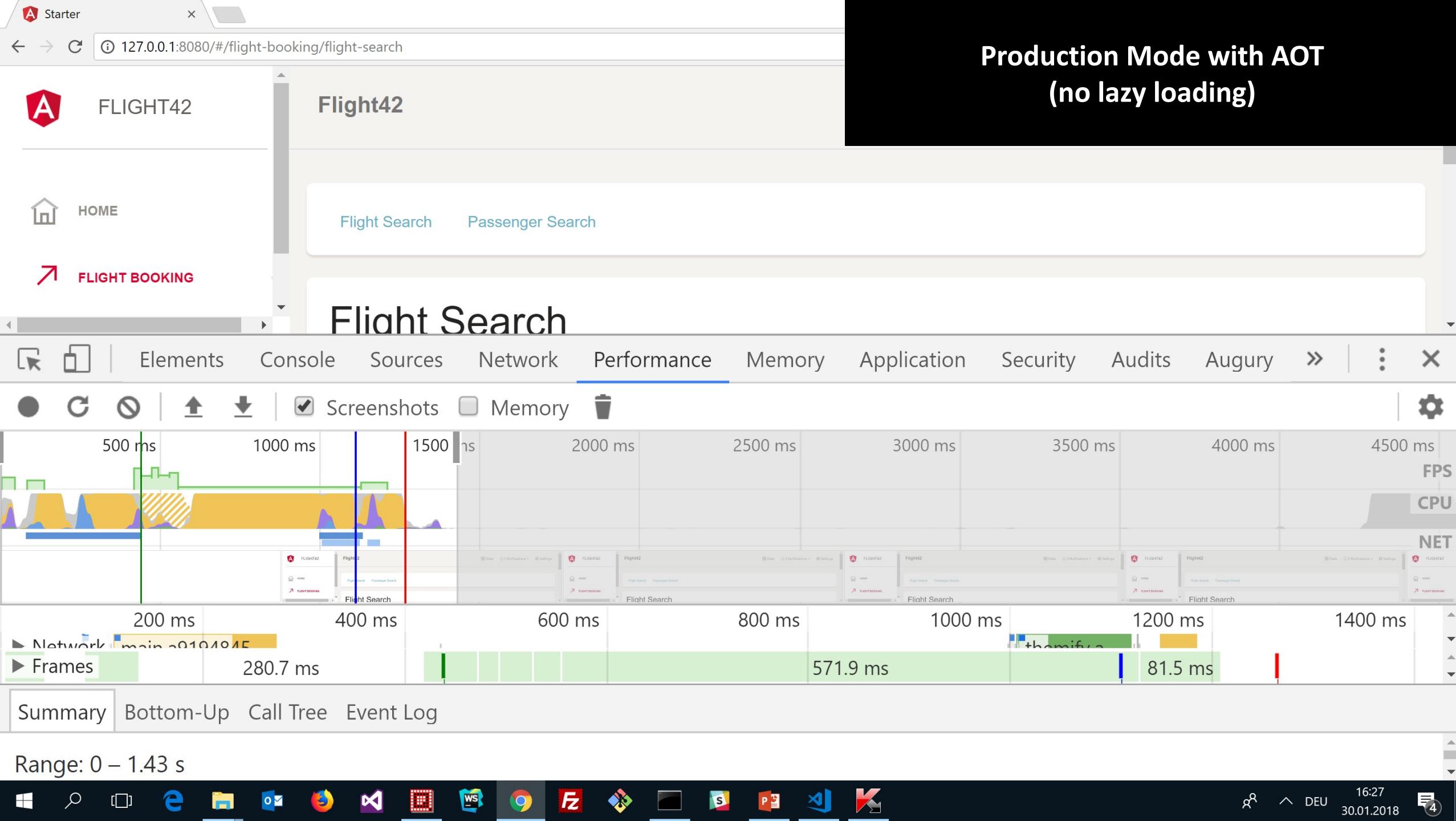
1000 ms 2000 ms 3000 ms 4000 ms 5000 ms 6000 ms 7000 ms

Network Frames 2234.0 ms

Summary Bottom-Up Call Tree Event Log

Range: 0 – 6.93 s

16:30 DEU 30.01.2018 4

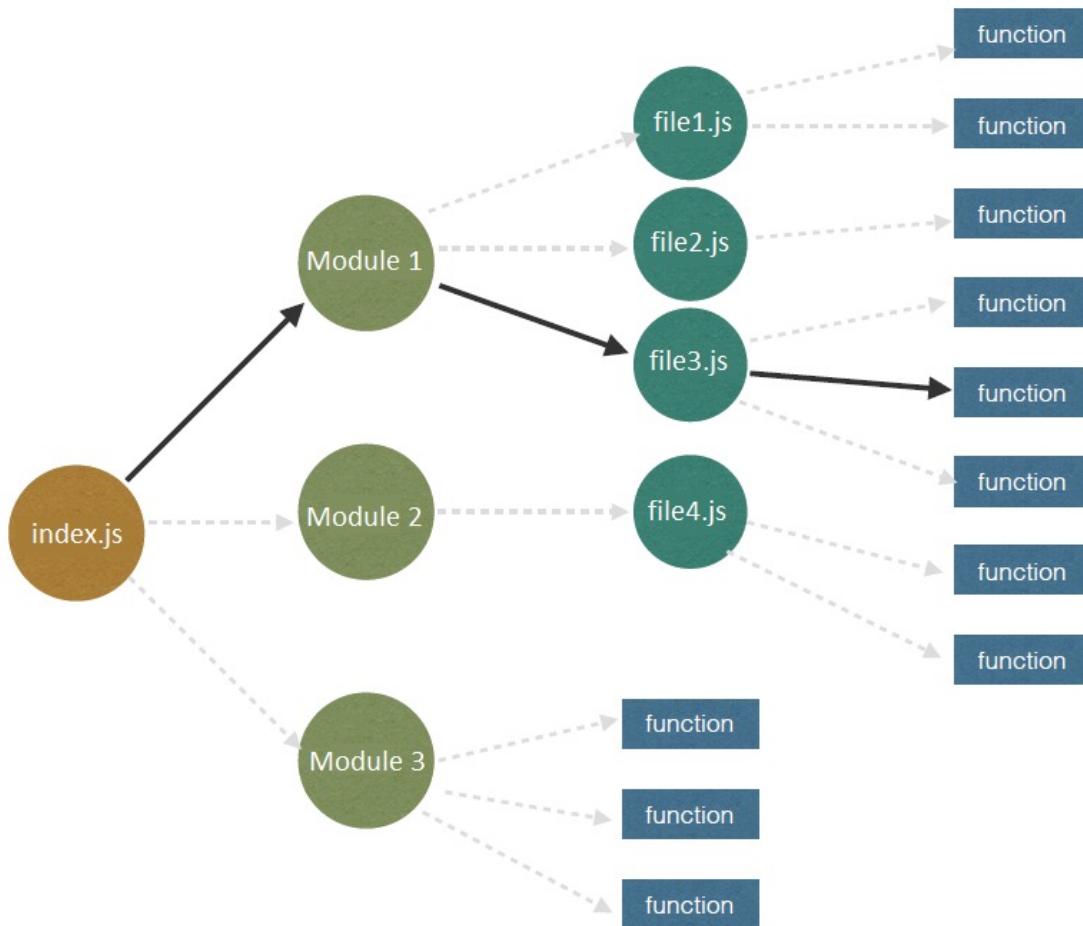


Ivy makes AOT the default 😊

- Ivy also does a lot of under the hood optimization
- No breaking changes, nothing to do from our side 😊
- Angular ViewEngine itself was not tree-shakable
- Angular Ivy is tree-shakable 😊
- Default since NG 10, for libs default since NG 12

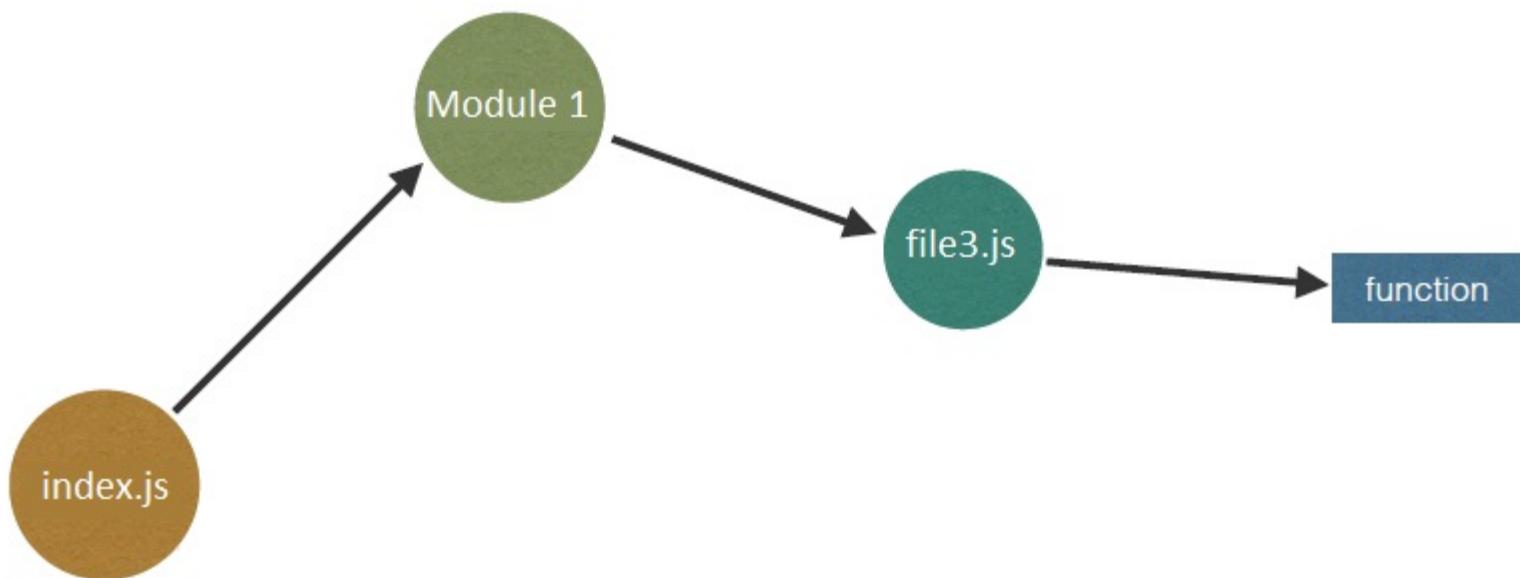
Tree Shaking

Before Tree Shaking



Tree Shaking

After Tree Shaking



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

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
and
Preloading

OnPush w/
Immutables and
Observables

AOT and Tree
Shaking



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



For performance deep dive

Watch this (starting at 8:30):

https://drive.google.com/file/d/15fmyedJPYSOIv_0YvFtg26XGS8tZpZ03/view

Repo: <https://github.com/jeffbcross/victor-videos/>