Angular Performance Tuning

Manfred Steyer
SOFTWARE*architekt.at*

Quick Wins

Bundling

Minification

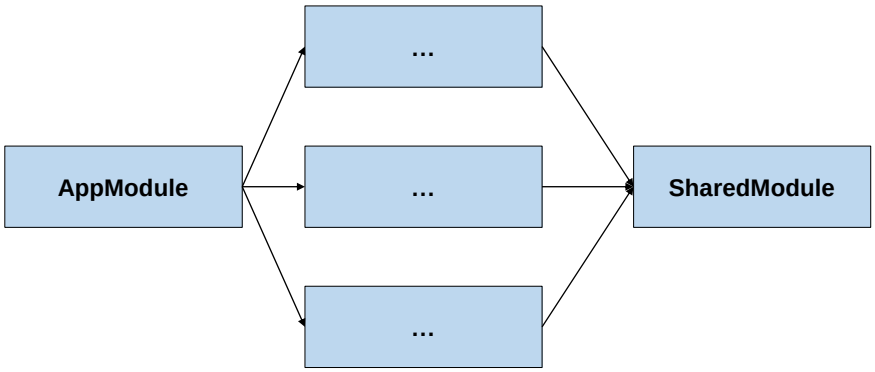
enableProdMode()

Contents

- Lazy Loading and Preloading
- Performance for Data Binding with OnPush
- AOT and Tree Shaking
- Caching with Service Worker
- Server Side Rendering

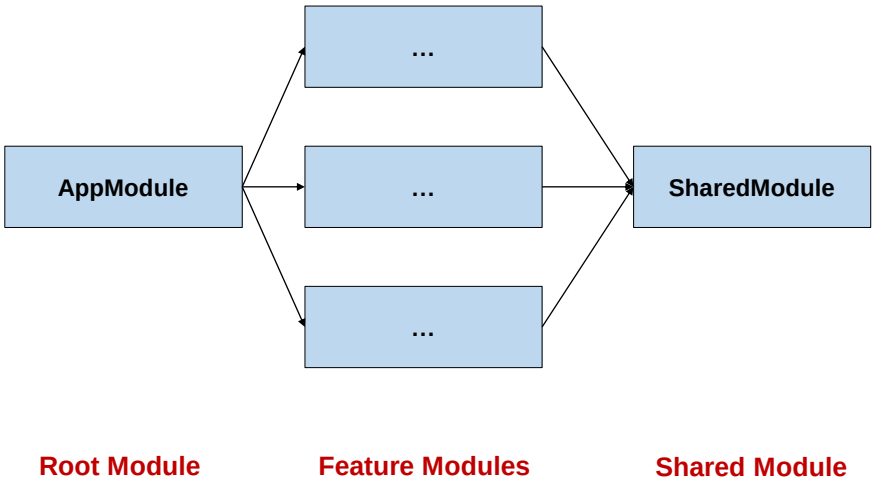
Lazy Loading

Module Structure



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

Lazy Loading



Root Module with Lazy Loading

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

DEMO

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

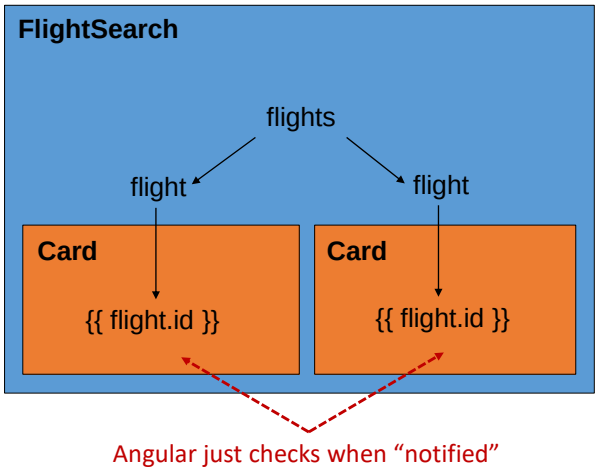
Activate Preloading

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


Performance-Tuning with OnPush

DEMO

OnPush


FlightSearch
flights
flight
flight
Card
{{ flight.id }}
Card
{{ flight.id }}
Angular just checks when “notified”

"Notify" about change?

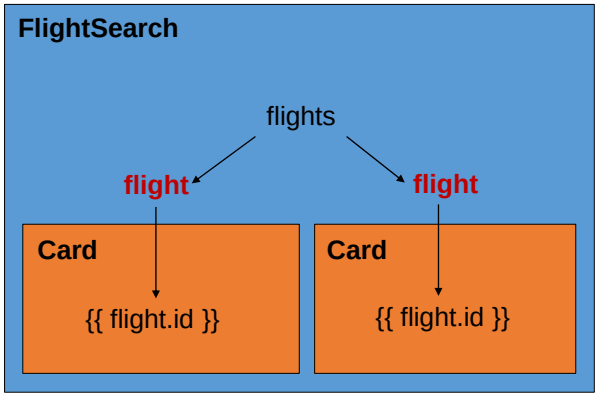
- Change bound data (@Input)
  - OnPush: Angular just compares the object reference!
  - e. g. oldFlight === newFlight
- Raise Event within the component
- Notify a bound observable
- Trigger it manually
  - Don't do this at home ;-)
  - At least: Try to avoid this

Activate OnPush

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

Change Inputs

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



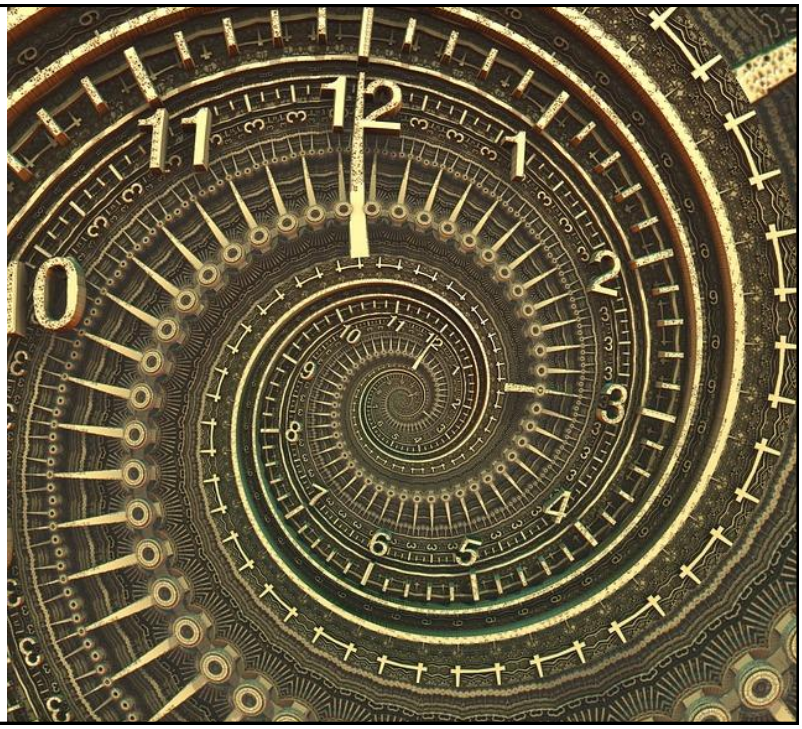
Observables and OnPush

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

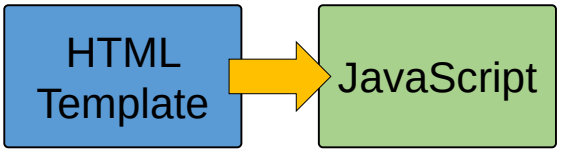
DEMO

SOFTWARE architekt.at

Ahead of Time (AOT) Compilation



Angular Compiler



Template 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 not needed parts of frameworks
  - Tree Shaking

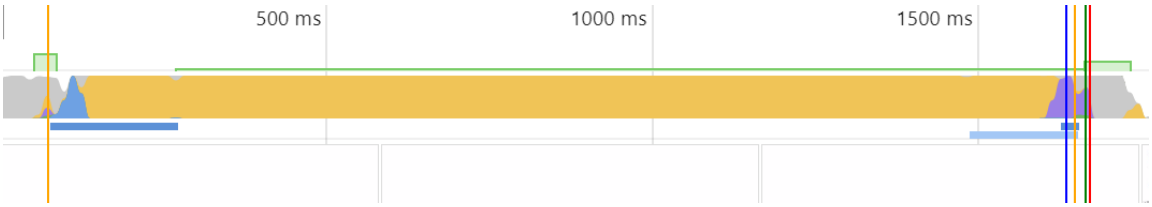
Angular CLI

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

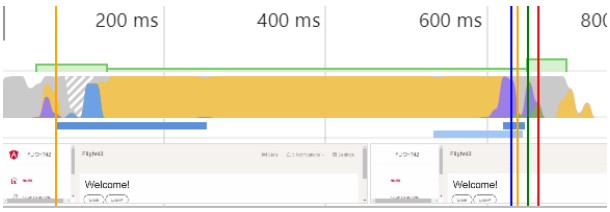
DEMO

SOFTWARE architekt.at

Flight Search (Prod Build w/o AOT)



Flight Search (Prod Build w/ AOT)

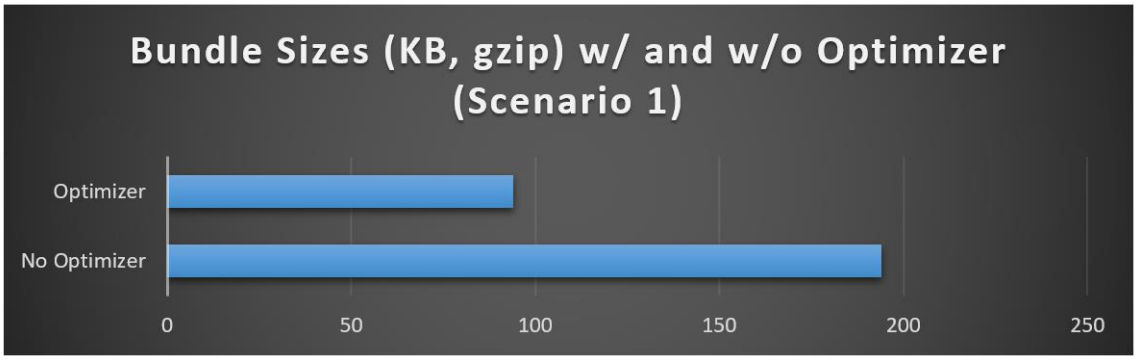


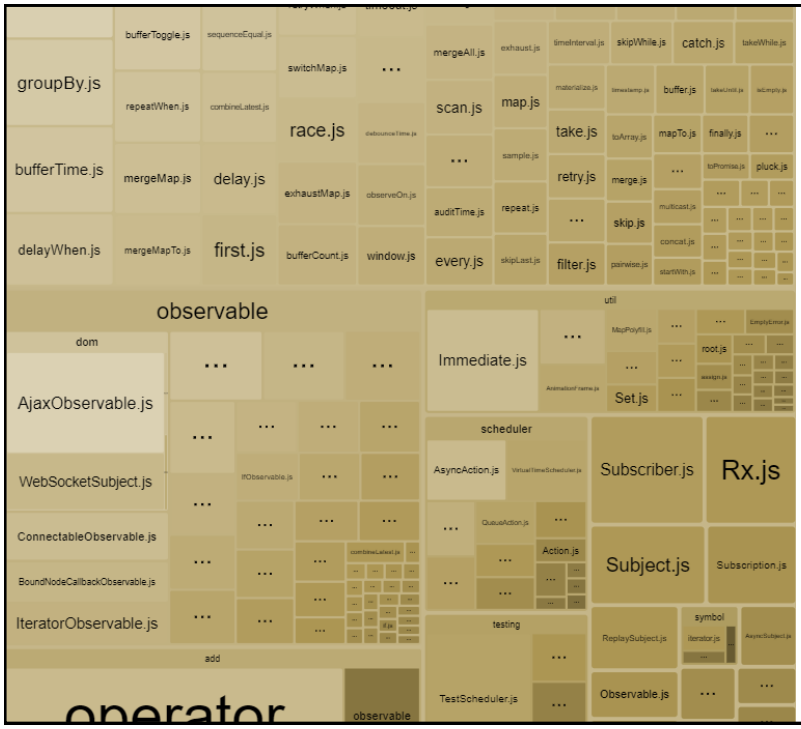
Challenges

- Most tree shaking tools are conservative
- They just remove code when they are 100% sure
- Very often, they aren't sure :-)
- Solution: Angular Build Optimizer
- Rewrites compiled code
- Currently: Experimental

SOFTWARE architekt.at

Sample Application w/ Angular Material





RxJS is huge

RxJS

- We don't need every operator/ method
- But: Methods are not tree-shakable for CLI/ webpack and others
- Solution: Just import methods you need
  - Imports are patching Observable

Import Operators

```
import 'rxjs'; // Import all --> Bad!
```

```
this.http.get<Flight>(url)
         .filter(f => f.price < 300)
         .map(f => toFlightOffer(f))
         .subscribe(…);
```


Import Operators

```
import 'rxjs/add/operator/map'; // Patch Observable --> add map
import 'rxjs/add/operator/filter';
```

```
this.http.get<Flight>(url)
         .filter(f => f.price < 300)
         .map(f => toFlightOffer(f))
         .subscribe(…);
```

Better Alternative: Pipeable Operators (aka lettable Operators)

```
import { map, filter } 'rxjs/operators';
```

```
this.http.get<Flight>(url)
         .pipe(
             filter(f => f.price < 300),
             map(f => toFlightOffer(f))
         )
         .subscribe(…);
```

Since RxJS 5.5

SOFTWARE architekt.at

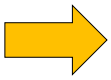
Removing Whitespaces

Removing Whitespaces

  Hello
World!

Removing Whitespaces

  Hello \n
World!

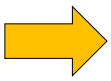


```
//Pseudo Code
createNode('p', ' Hello World');
createNode('TEXT-NODE', '\n');
createNode('p', 'World!');
```

Template Compiler

Removing Whitespaces

  Hello \n
World!



//Pseudo Code
createNode('p', ' Hello World');
createNode('TEXT-NODE', '\n');
createNode('p', 'World!');

Template Compiler

Removing Whitespaces

```
@Component({
    selector: 'flight-card',
    templateUrl: './flight-card.component.html',
    styleUrls: ['./flight-card.component.css'],
    preserveWhitespaces: false
})
export class FlightCardComponent {
    […]
}
```

New in Angular 5 and 4.4!

Caching with Service Worker

What are Service Workers?

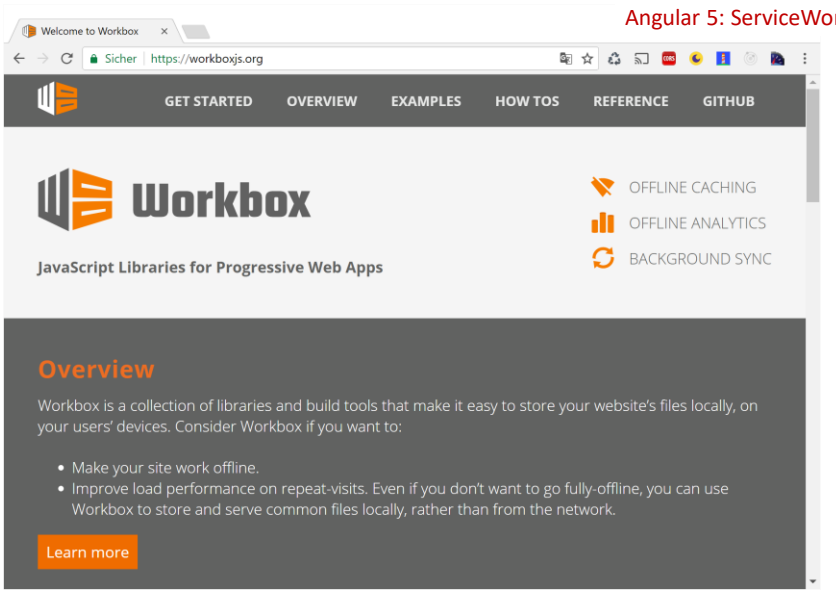
- Background Tasks
- Web App installs them
- Are activated and deactivated on demand

Service Worker und Caching/ Offline

- Intercept requests
- Decide how to respond (Cache, Network)
- Same Origin Policy
- Caching Patterns
  - Cache only
  - Network only
  - Try cache first, then network
  - Try network first, then cache
  - etc.

SOFTWARE architekt.at

Angular 5: ServiceWorkerModule



Service Worker with Workbox (sw.js)

```
importScripts('./assets/workbox-sw.js');

const workboxSW = new WorkboxSW();
```


Service Worker with Workbox (sw.js)

```
importScripts('./assets/workbox-sw.js');

const workboxSW = new WorkboxSW();

const networkFirst = workboxSW.strategies.networkFirst();
const cacheFirst = workboxSW.strategies.cacheFirst();
```

Service Worker with Workbox (sw.js)

```
importScripts('./assets/workbox-sw.js');

const workboxSW = new WorkboxSW();

const networkFirst = workboxSW.strategies.networkFirst();
const cacheFirst = workboxSW.strategies.cacheFirst();

workboxSW.router.registerRoute(
                  new RegExp('^http:\/\/www.angular.at\/api\/'), networkFirst);

workboxSW.router.registerRoute(/./, cacheFirst);
```

SOFTWARE architekt.at

Browser Support

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			29						
			45						
			48					4.3	
		45	49			8.4		4.4	
8		46	50			9.2		4.4.4	
11	13	47	51	9.1	38	9.3	8	50	50
	14	48	52	10	39				
		49	53	TP	40				
		50	54						

In Development

In Development

SOFTWARE architekt.at

DEMO

SOFTWARE architekt.at

Server Side Rendering

Why Server Side Rendering?

Prerender 1st Page

Start up performance

Consumer

renderModuleFactory

Available since Angular 4.0

```
[…]
renderModuleFactory(moduleFactory, {
    document: indexFileContentsAsString,
    url: options.req.url
})
.then(string => {
     […]
});
[…]
```

DEMO

Challenges

Angular 5:
Server Side DOM Simulation (partly)

- Other conditions
- Separate Services for Server and Client-Seite
- Renderer abstracts DOM
- 3rd parts libs

More about this in my Medium Account

- Configuration Details, Samples etc.
- https://medium.com/@ManfredSteyer/angular-performance-tuning-article-series-6e3c33707b25

Conclusion

Quick Wins	Lazy Loading and Preloading	OnPush w/ Immutables and Observables
AOT and Tree Shaking	Caching w/ Service Worker	Server Side Rendering