



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Performance Tuning

Alex Thalhammer

Turbo Button



Masterplan

- New performance workshop
- 2-3days
- Under construction 😊

15facts in 15slides

Topics

Tools

InitialLoad

Runtime



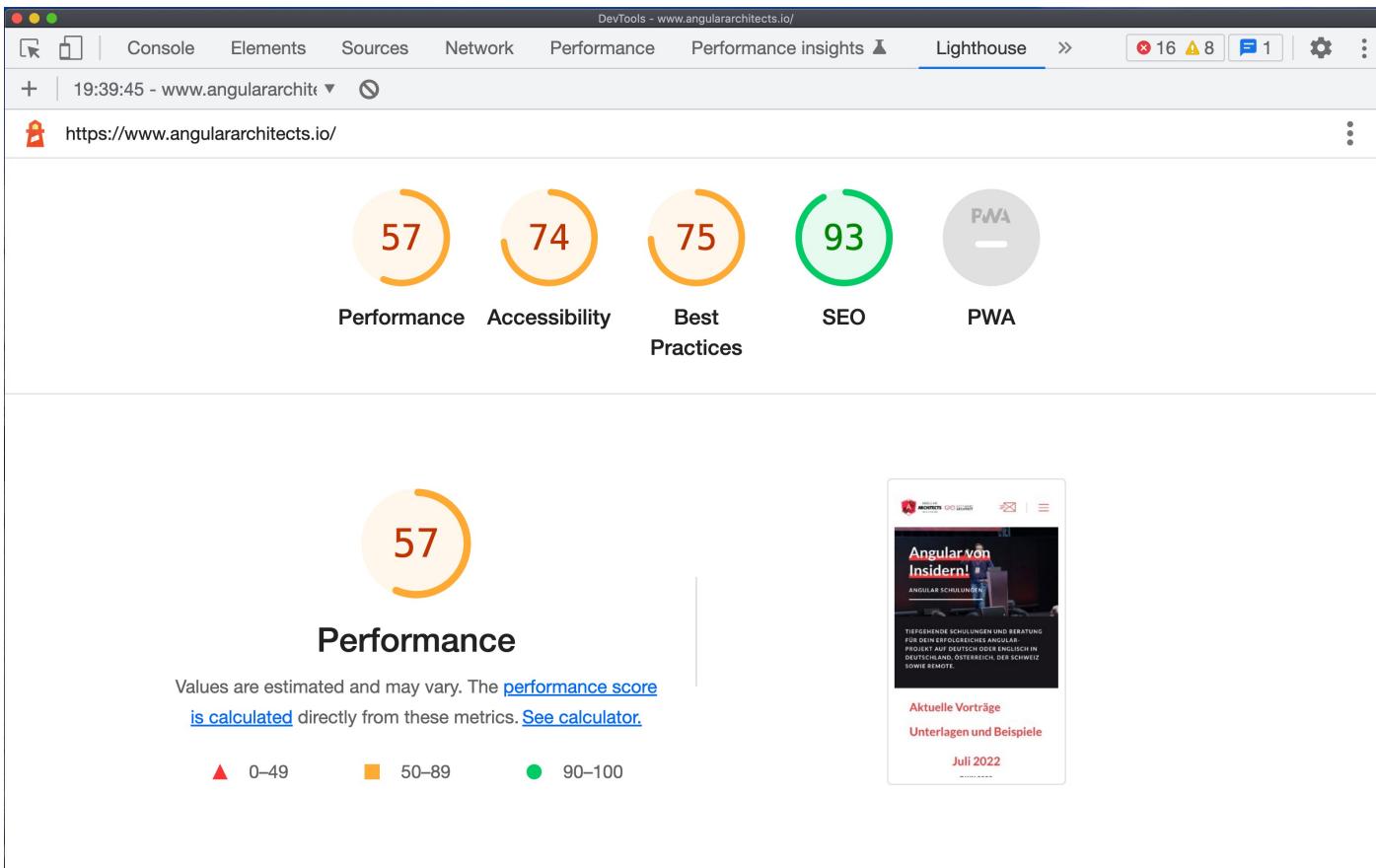
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT



#1: Google Chrome Lighthouse / PageSpeed

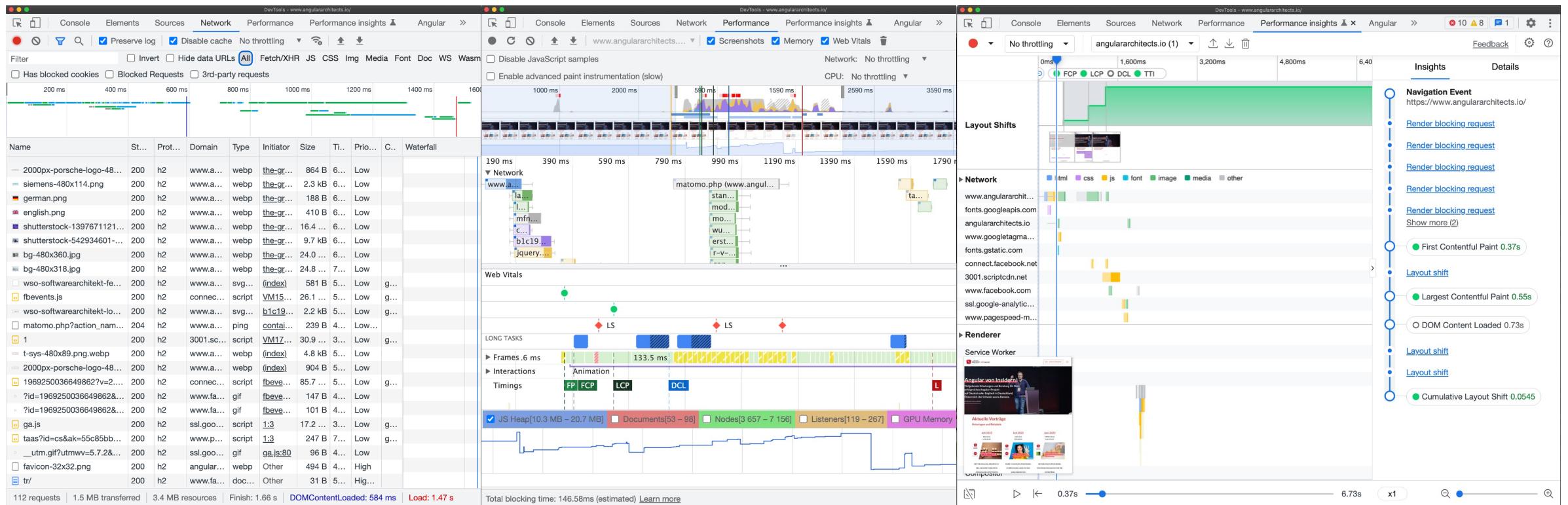


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

#2: Google Chrome DevTools



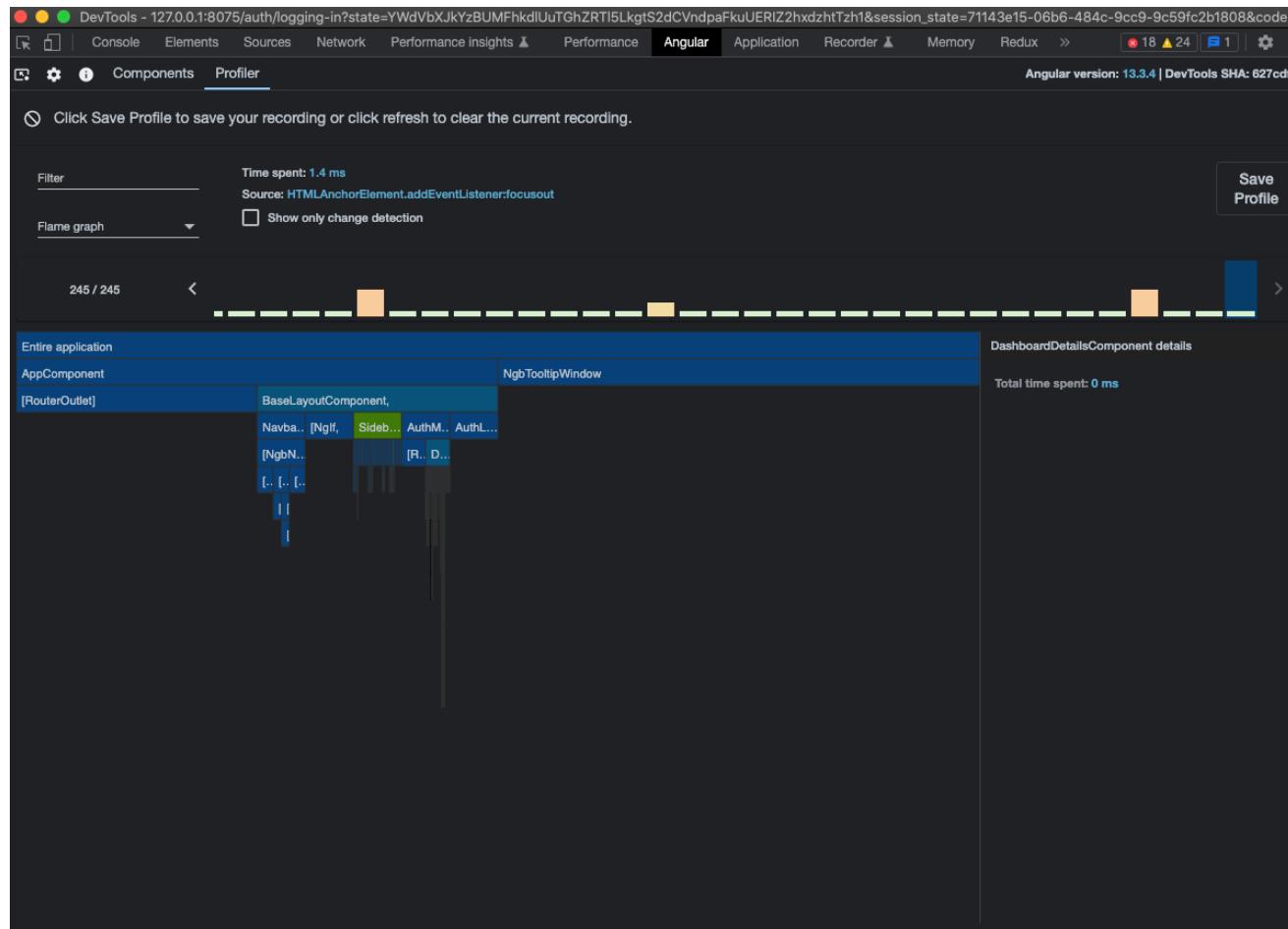
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE

ARCHITECT

#3: Angular DevTools Profiler



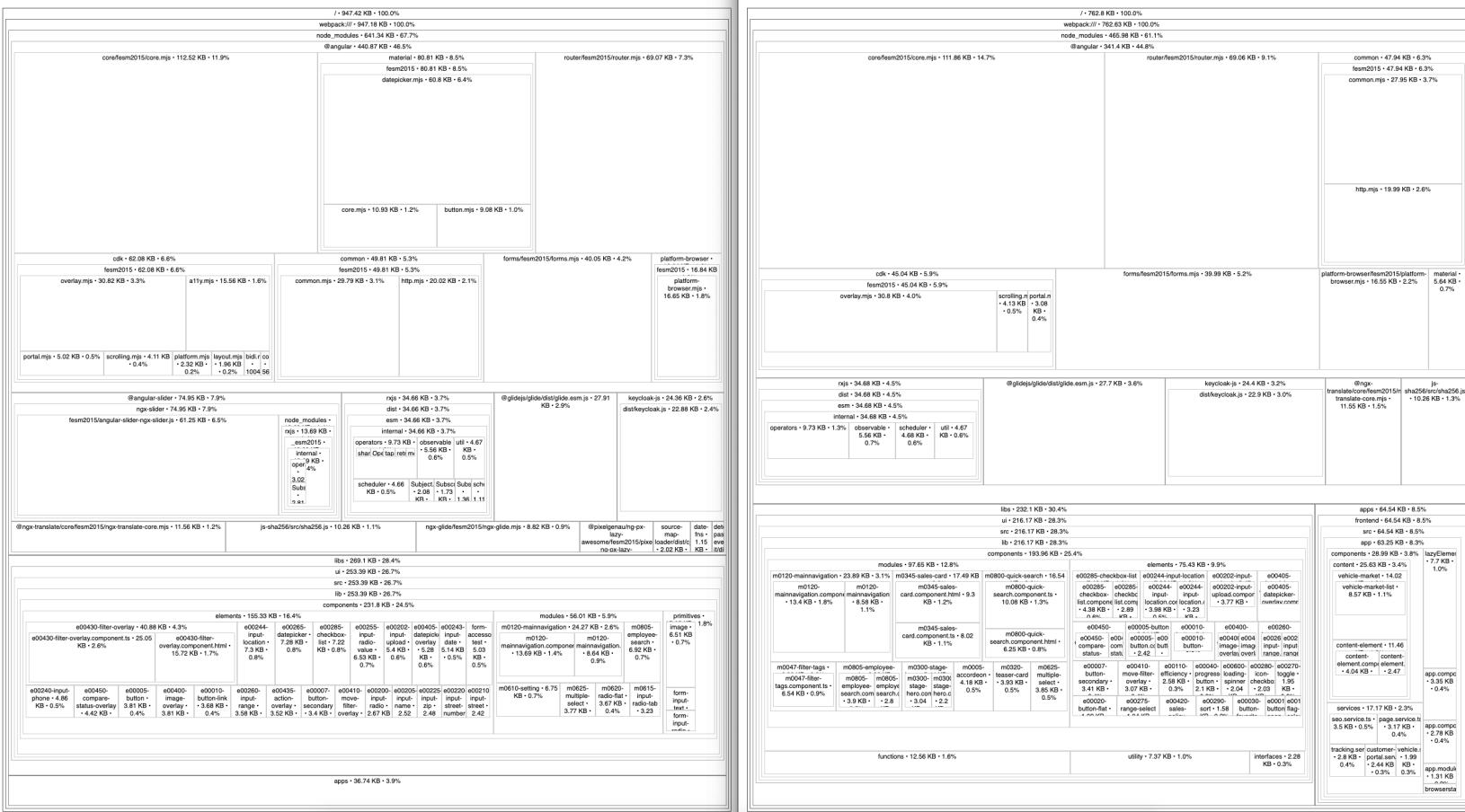
ANGULAR
ARCHITECTS

INSIDE KNOWLEDGE

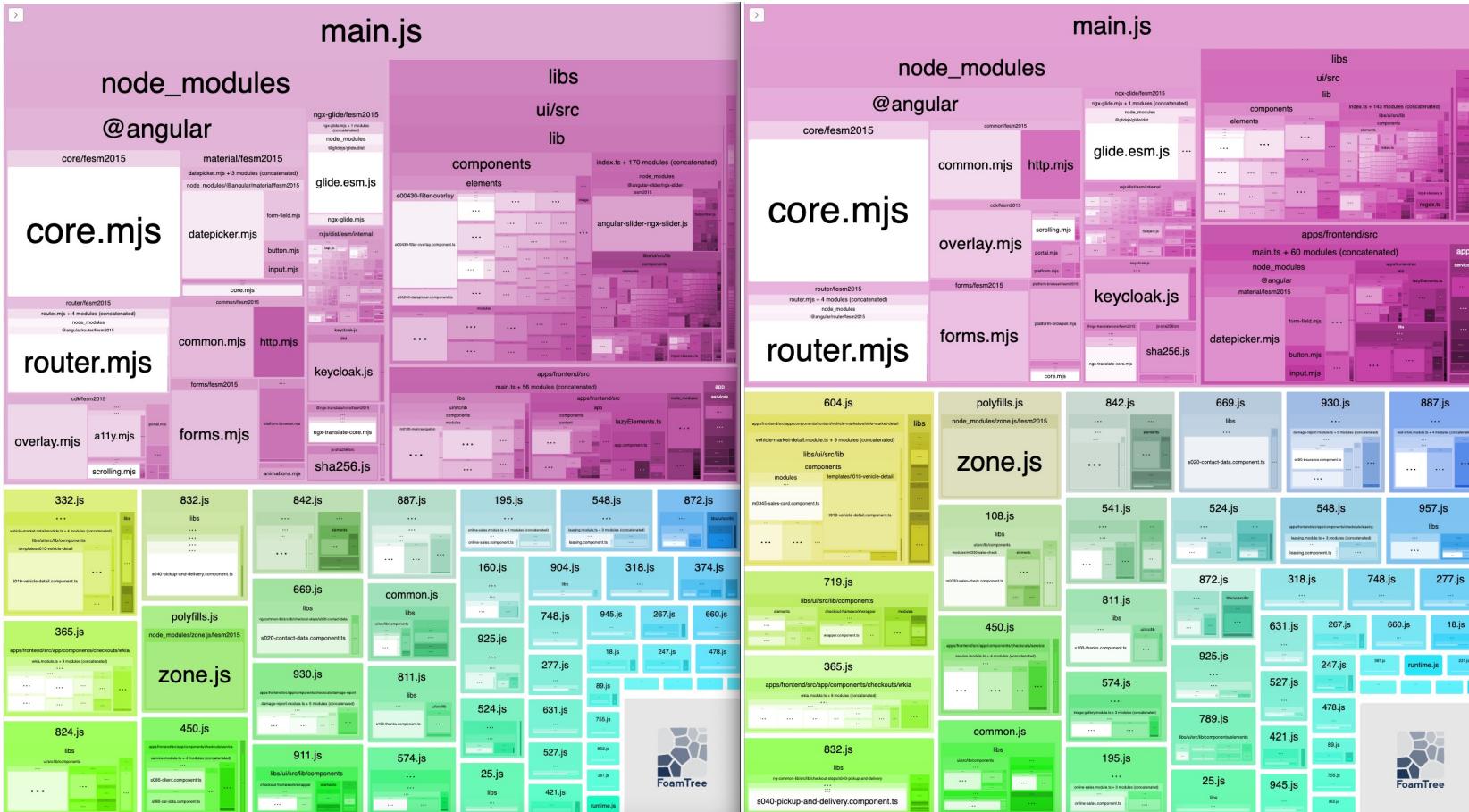


SOFTWARE
ARCHITECT

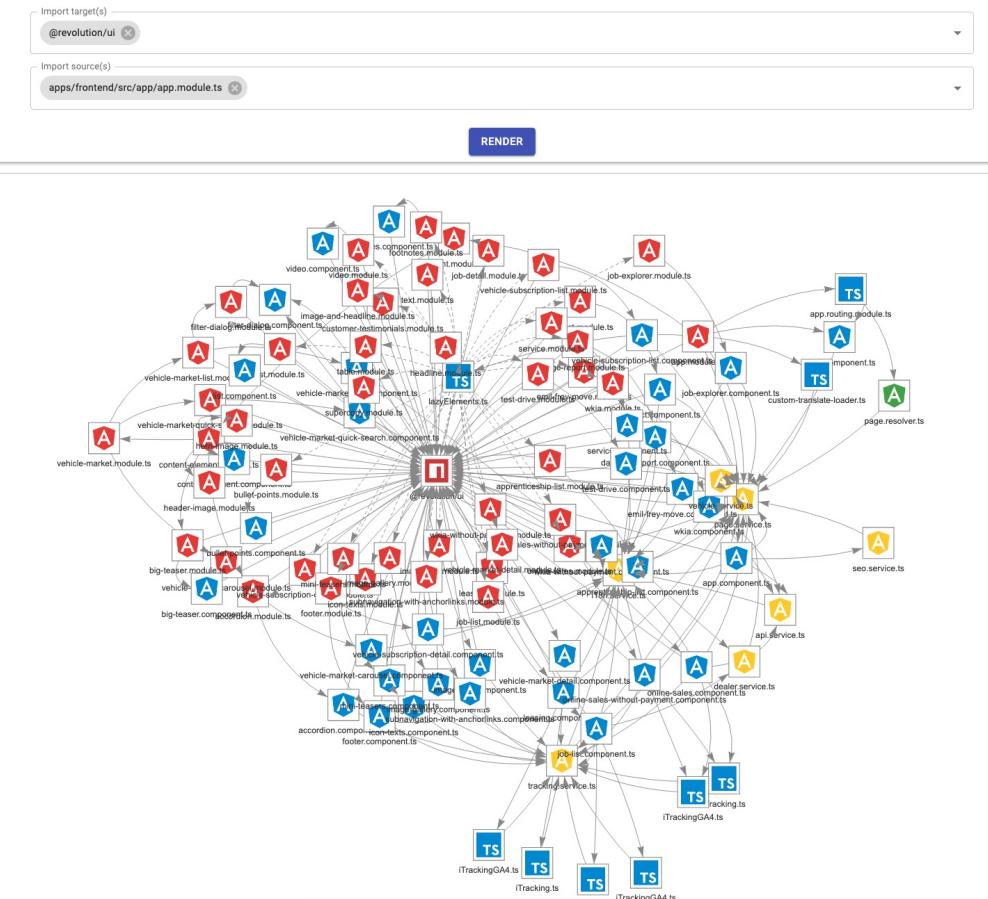
#4: Sourcemap Explorer (more accurate)

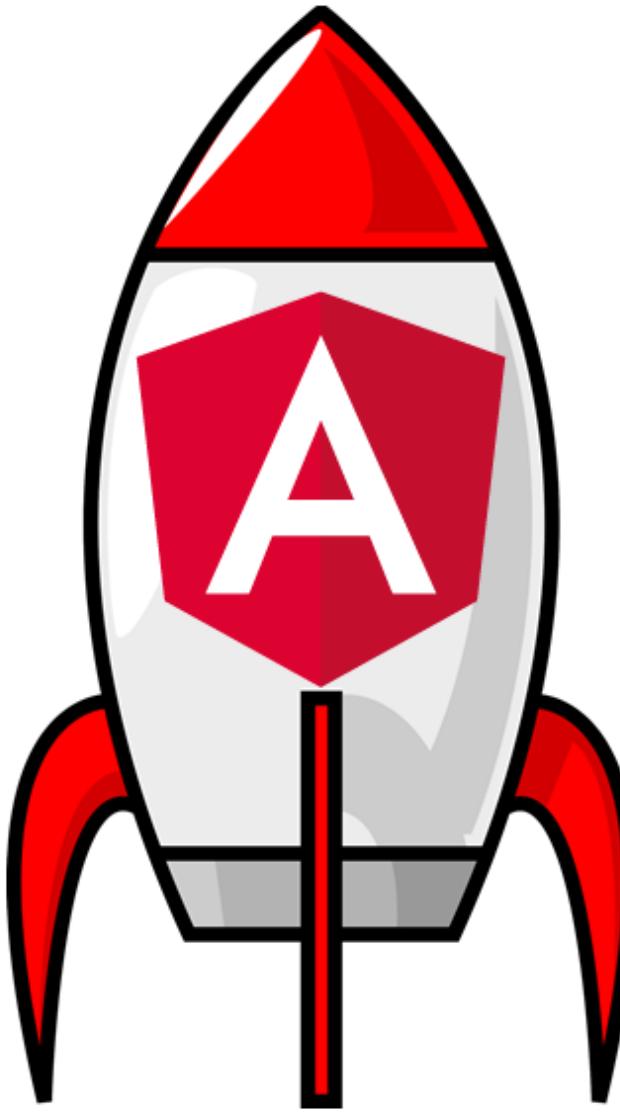


#4.5: Webpack Bundle Analyzer (colorful)



#5: rx-angular/import-graph-visualizer





#6: Use web performance best practices

- Problem: *Slow infrastructure & too large assets / waiting for responses*
- Identify: Using Lighthouse & PageSpeed / seeing loading spinners
- Solution:
 - Use performant infrastructure (CDN?), cache static files, use service workers...
 - Optimize assets (use correctly sized images and srcsets (responsive sizes), use .webp & vector graphics where it makes sense)
 - Optimistic updates

#7: Avoid large 3rd party libs / CSS frameworks

- Problem: *Importing large 3rd party libraries that are not treeshakable*
- Identify: Find with sourcemap or wba
 - E.g. moment & lodash
- Solution: Remove or replace that lib / framework
 - E.g. date-fns & lodash-es

Lazy Loading



#8: Use Lazy Loading a lot - but carefully ;-)

- Problem: *Loading too much source (libs / components) at startup*
- Identify: Not using lazy loading throughout the App
- Solution: Implement lazy loading wherever you can
 - Maybe use a CustomPreloadStrategy if App is very big
 - **But** don't lazyload the initial feature, because it will be delayed then ;-)

#9: Load everything below the fold lazily

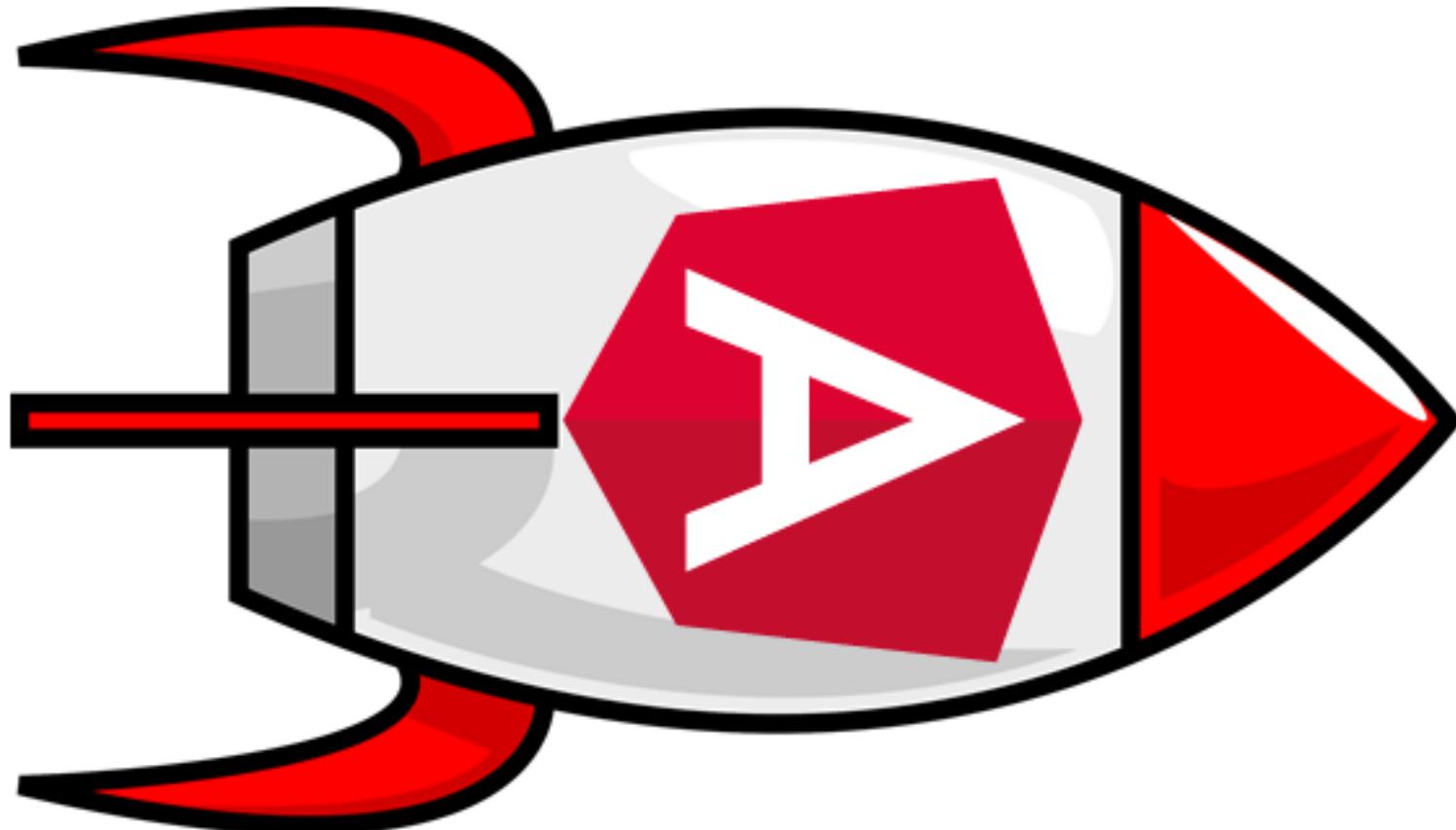
- Problem: *Bad PageSpeed Score*
- Identify: Initial load is too slow (Using Lighthouse / PageSpeed)
- Solution: Use custom lazy loading of content below the fold

#10: Server-side rendering (Angular Universal)

- Problem: *After download rendering on the client takes too much time*
- Identify: After .js files have been loaded js main thread takes too long
- Solution: Use Angular Universal so that the page is rendered on the server and then served to the client

#10.5: Prerender important routes (Universal)

- Problem: *Server response takes too long cos page has to be rendered*
- Identify: Long server response time when using Universal SSR
- Solution: Prerender the important pages on the server and then serve it rendered to the user
 - Built-in Angular Universal since V11





Performance-Tuning with OnPush



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

#11: O.B. change detection

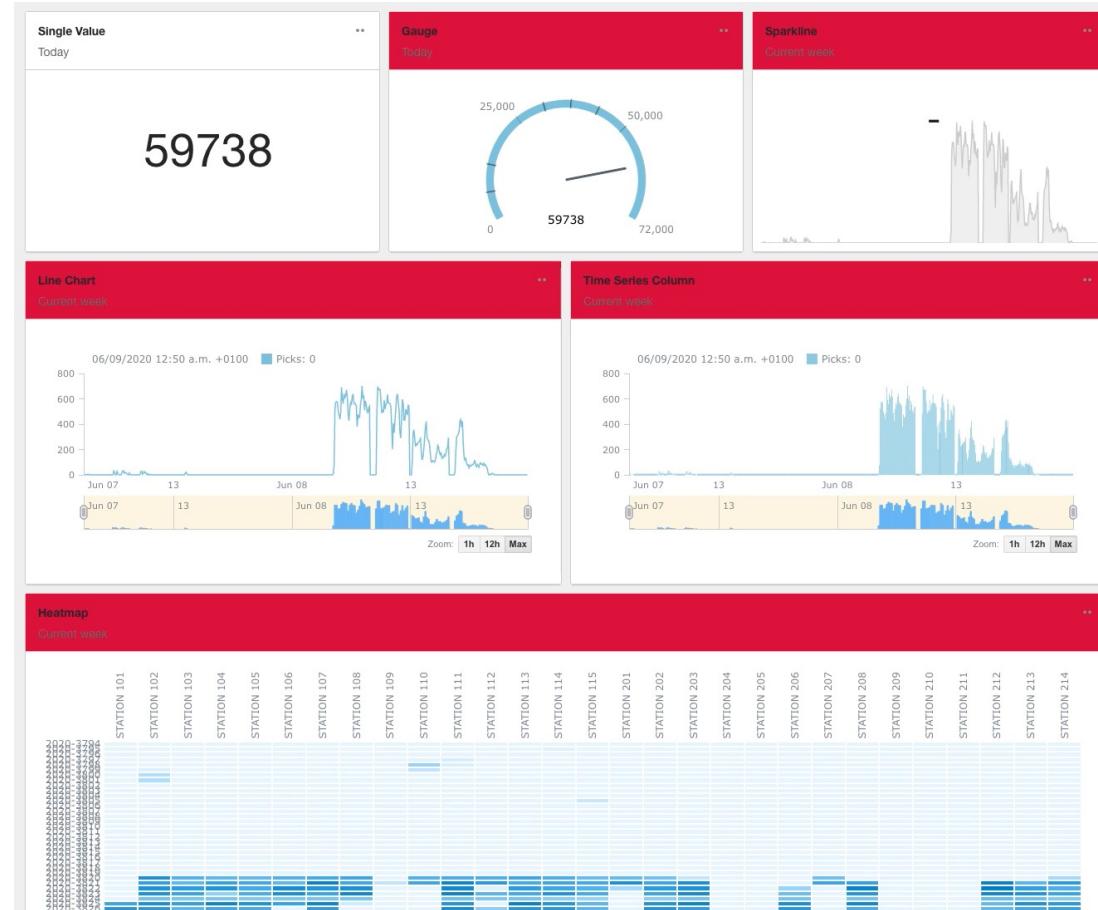
- Problem: *Local state change triggers change detection in other comps*
- Identify: Use the infamous `blink()` or the Angular DevTools Profiler
 - E.g. Input field keydown triggers change detection in other components
- Solution: `ChangeDetectionStrategy.OnPush` as default

```
1  "schematics": {  
2      "@schematics/angular:component": {  
3          "styleext": "scss",  
4          "changeDetection": "OnPush"  
5      }  
6  },  
  
angular.json hosted with ❤ by GitHub
```

#11.5: detectChanges() vs markForCheck()

- Use **cdr.detectChanges()** to trigger CD immediately when you've updated the model after angular has run it's change detection, or if the update hasn't been in Angular world at all
- Use **cdr.markForCheck()** to mark for check in next CD cycle if you're using **OnPush** and you're bypassing the ChangeDetectionStrategy by mutating some data or you've updated the model inside a **setTimeout**

#12: Zone pollution by 3rd party libs (charts)



ANGULAR
ARCHITECTS

INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

#12: Zone pollution by 3rd party libs (charts)

- Problem: *Callbacks that trigger redundant change detection cycles*
- Identify: Use the infamous `blink()` or the Angular DevTools Profiler
 - E.g. `MouseEvent` listeners, `setTimeout` or `requestAnimationFrame()`
- Solution: Run outside of NG Zone
 - Inject (`private ngZone: NgZone`)
 - Call `this.ngZone.runOutsideAngular(() => doStuff)`
- Alternative: Using `cdr.detach()` for components

#12.5: ChangeDetectorRef API, once more

`detectChanges`

- Runs Change Detector for the component and its children
- It runs CD once also for the component which is detached from the component tree

`markForCheck`

- It marks component and all parents up to root as dirty
- In next cycle Angular runs CD for marked components

`reattach`

- Re-attaches the component in the change detection tree
- If parent component's CD is detached, it won't help, so make sure to run `markForCheck` with `reattach`

`detach`

- Detaches the component from the change detection tree
- Bindings will also not work for the component with detached CD

`checkNoChanges`

- Changes the component and its children and throws error if change detected



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

#13: Optimization with state or Flags

- Problem: *Redundant calculations for conditions*
- Identify: Methods being executed in ***ngIf** statements
- Solution: Use StateManagement like Subjects or use boolean flags or strings, that only change when they should

#13.5: Optimization with Angular Pipes

- Problem: *Redundant calculations for content or formatting*
- Identify: Methods being executed in string interpolations in the template or similar things slowing change detection cycles
- Solution: Use (pure) Angular Pipes

#14: Using trackBy in ngFor

- Problem: *Angular will replace all items in *ngFor upon changes*
- Identify: Easy - search for "*ngFor"
- Solution: Use the trackBy function

```
<li *ngFor="let dashboard of dashboards; trackBy: trackByDashboardId"  
    trackByDashboardId(index: number, item: Dashboard): number {  
        return item.id;  
    }  
}
```

#15: Large component trees

- Problem: *Too many (100+) components are loaded*
- Identify: Lots of components slowing down frame rate
- Solution: On demand component rendering
 - E.g. Pagination or Angular CDKs <cdk-virtual-scrolling-component>

One more thing

- Unsubscribe from all Observables in your App
 - Except Angular Router Params

Any questions or remarks?

