



ANGULAR
ARCHITECTS

NG POLAND 2025

<https://LXT.dev>



Alexander Thalhammer

Deferrable Views and Incremental Hydration

Alexander Thalhammer, ANGULARarchitects.io

Good news!



Not about AI 🎉

Live demos 🖐️

Huge rectangles!



But first...



Please imagine



Imagine a perfect winter day in the alps

- it's cold but you're wearing your new gear   
- the sky is blue and sun is shining  
- 50cm of fresh powder  
- but then ...



... you
have
to
queue!



Why Performance Optimization?



Happy Users

Happy Devs



What kind of Optimization?

general web
performance

Angular
specific

initial load

runtime

Angular Initial Load Optimization



router-based:
loadChildren
loadComponent

template-driven:
@defer

RenderModes:
SSR, SSG

incremental
hydrate



Please raise hands for



Angular

SSR

Angular & SSR



Challenges

node?

login?

dynamic?

knowledge!



@LX_T → favorite topics



Performance 🚀

Accessibility ♿

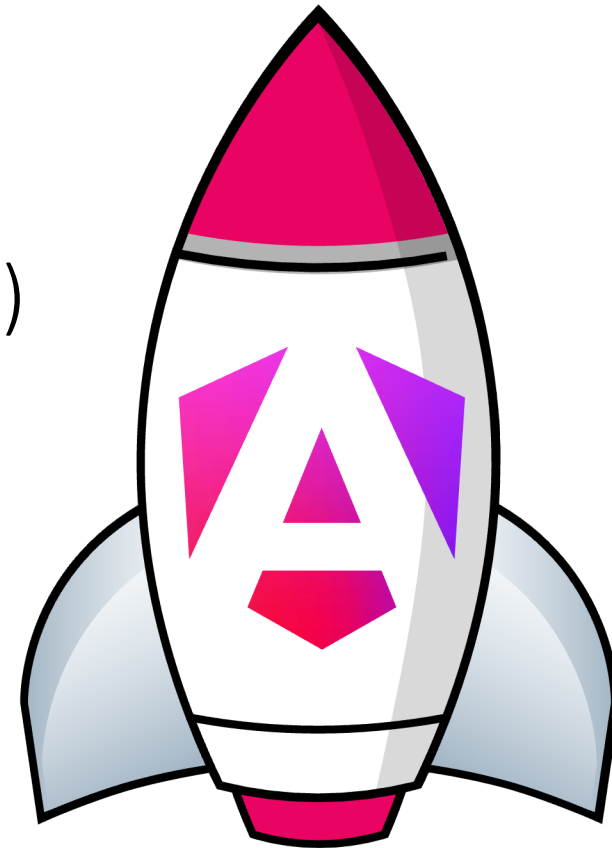
Best Practices 📈

Refactoring 😅



Agenda

- 1) Deferrable Views (NG $\geq$ 17)
- 2) Incremental Hydration (SSR & NG $\geq$ 19)





ANGULAR
ARCHITECTS



First things first

First things first: Terminology

- Lazy Loading === Defer Loading?
 - Lazy Loading for router-based lazy loading using `loadChildren()` / `loadComponent()`
 - Defer Loading for template-driven lazy loading using `@defer` blocks
- Hydration → attach JS (event handlers) to static HTML
- CSR → client-side rendering, SSR, SSG → static site generation
- above-the-fold vs below-the-fold

First things first: Measure Performance

- measure your Angular apps initial load performance
- make improvements measurable
- how to measure? 🤔

Test Angular App with Lighthouse DevTools

- `ng b` (production config)
- `[npx] serve dist/[app-name]/browser [-s]`
- open <http://localhost:3000> in incognito and run Lighthouse
- do 3-5 audits and note down the median of the metrics (FCP, LCP & INP ...)



ANGULAR
ARCHITECTS

@defer



Deferrable Views with @defer (v17)

- alternative to router-based Lazy Loading in Angular
 - to reduce initial load (size and time) → improve performance
 - without dedicated route
- very good DX compared to hand-crafted solutions with **ViewContainerRef**
- only works with **standalone components**

Digression – Angular Migrations (mandatory!)



standalone

build w/
esbuild & vite

signal inputs
& queries

control flow

lazy-loaded
routes

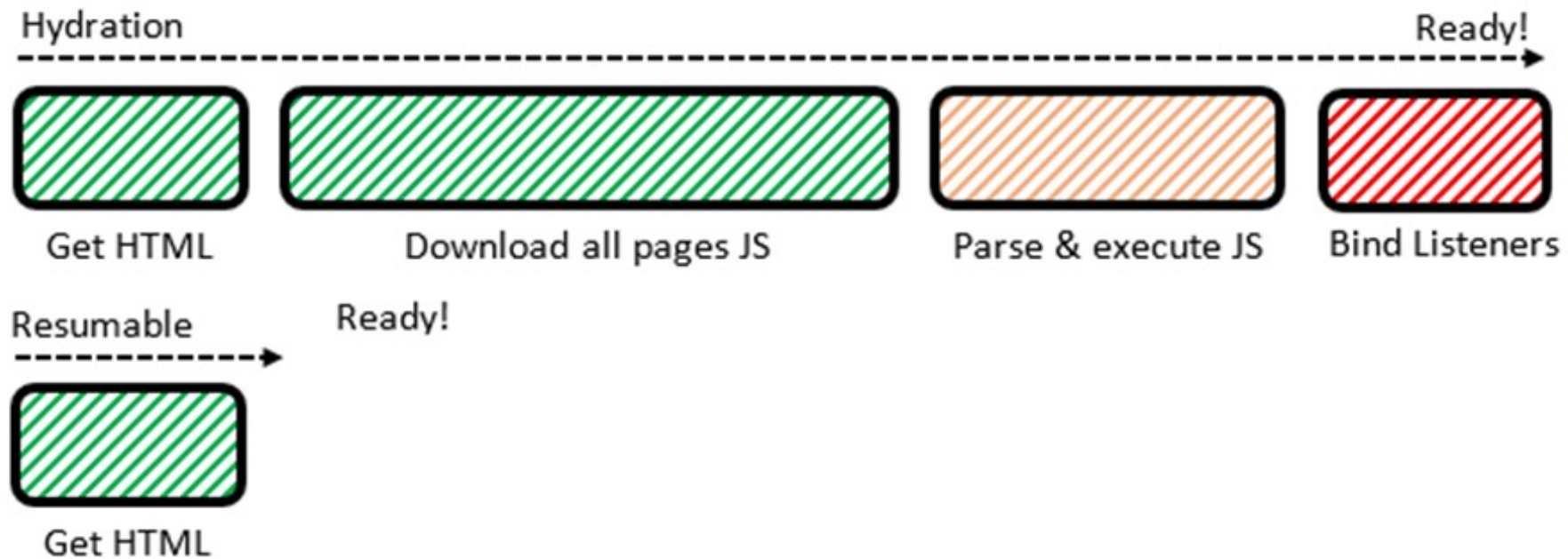
cleanup imports &
self-closing tags &
ngClass & ngStyle



Angular Docs – Migration

<https://angular.dev/reference/migrations>

Why Incremental Hydration?



<https://www.acldigital.com/blogs/zero-hydration-qwik-future-web>

Deferrable Views – Scenarios

- below-the-fold content
- **widgets** that are used on multiple routes (router-based cannot be used)
- lazy loading of **large 3rd-party** dependencies (e.g., charts, tables, etc.)
- not best for **above-the-fold** content (initially the placeholder is shown)

Deferrable Views – Trigger Intro

- triggers (lazy) loading of the component
 - e.g. on viewport (similar to ngOptimizedImage)

```
@defer (on viewport) {  
  <aa-lazy-component />  
}
```

Deferrable Views – Trigger Details

- **on**
 - immediate
 - idle (*default*)
 - viewport (like ngOptimizedImage)
 - hover (mouseover & focusin)
 - interaction (click & keydown)
 - timer(4200ms) (use case → demo)
- **when**
 - boolean symbol or
 - Signal<boolean> (or function)
- *both are one-way ticket*
- combination possible

Deferrable Views – Prefetch

- with **prefetch**, the deferred JavaScript can be preloaded
 - similar to the PreloadingStrategy used in router-based lazy loading
- this speeds up the loading process when the **@defer** trigger is activated

```
@defer (on viewport; prefetch on idle) {  
  <aa-lazy-component />  
}
```

Deferrable Views – Placeholder & Loading

- @placeholder
- @loading
- @error

```
@defer (on viewport; prefetch on idle) {  
  <aa-lazy-component />  
} @placeholder (minimum 500ms) {  
    
} @loading (after 500ms; minimum 1s) {  
    
} @error {  
  <p>Why do I exist?</p>  
}
```

Demo: @defer

<https://github.com/L-X-T/ng-poland-ssr-ih>
ng s --no-hmr

Deferrable Views – Best Practices

- on immediate (priority) or
- on idle (default trigger)
- on viewport (scroll)

- be careful with nested **@defer** blocks
 - they have to use different triggers

- cautious when **@defer** is triggered above-the-fold during the initial load
 - no **layout shifts** should occur (poor UX) → use placeholders with same size

Deferrable Views – Combination with SSR



- forces **CSR** → combine with SSR for user-dependent content (e.g., price)
- in combination with SSR, the **@placeholder** is rendered on the server
 - or nothing if no **@placeholder** is provided
- only on the client the **@defer** blocks' comp. view is rendered
 - thereby enforcing client-side rendering
- only with Incremental Hydration comp. view is pre-rendered on the server





ANGULAR
ARCHITECTS

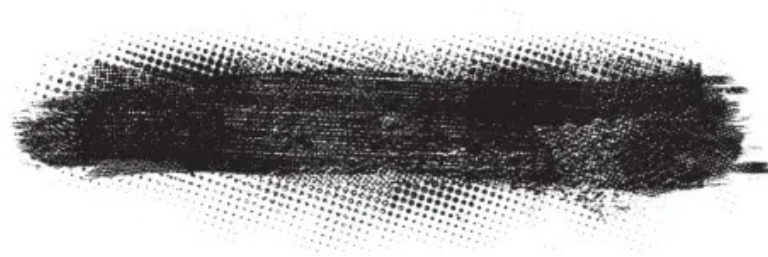
hydrate on



Incremental Hydration (v19)



The *FUTURE* is



incremental hydration



Incremental Hydration (v19)



- based upon Hydration (v16), @defer (v17) and Event Replay (v18)
- enables **even better** initial load experience, because
 - less JavaScript is shipped → partial lazy loading of JS on demand
- @defer blocks now can also be used above-the-fold, because
 - the comp. view (instead of @placeholder, rendered on the server) is shown



Incremental Hydration – Scenarios

- above-the-fold content (now usable)
- dynamic (JS-heavy) content
- products in a online shop
- actions, forms, tables, charts, ...

Incremental Hydration – Trigger

- same triggers like `@defer`
- `on` (immediate, idle, viewport, hover, interaction)
- `when` (boolean symbol, `Signal<boolean>` (or function))
- `never` (for static content without interactivity)

Demo: Incremental Hydration

<https://github.com/L-X-T/ng-poland-ssr-ih>

ng s --no-hmr

Incremental Hydration – Best Practices

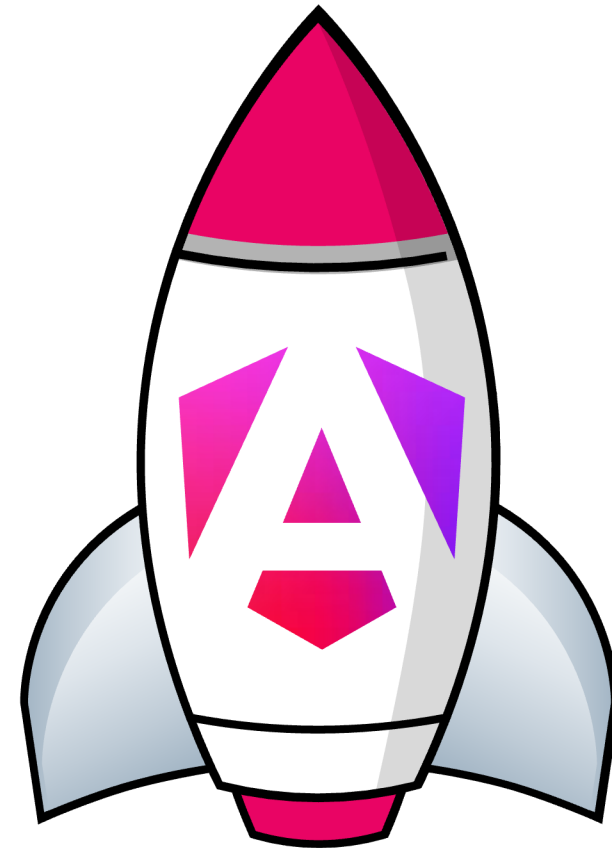
- on viewport (priority) or
- on hover (my favorite)
- on interaction

- use it for **landing pages!**

- initial load blazingly fast 😊

Conclusion @defer & hydrate on

- 1) Faster **Angular** Apps (FCP, LCP & INP)
- 2) Better **UX makes** Users **happy** 😊
- 3) Devs **happy** 😊





ANGULAR
ARCHITECTS

Why aren't you using it?



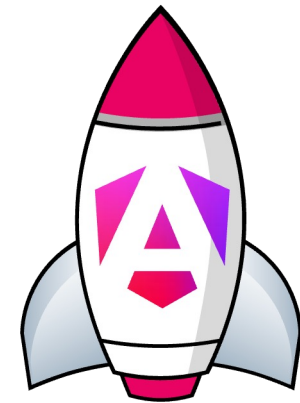
Thank you for the attention



[X] [@LX_T](#)  *Angular Updates*

[web] <https://LXT.dev>

 *Slides & Source*



Workshops
and Consulting

<https://angulararchitects.io>

