Outline - Initial Load Performance

- Assets & Build

- Lazy Loading & Deferrable Views

- SSR & SSG

SSR & SSG

- Server-Side Rendering (SSR)

- Hydration (NG >= 16)

- Event Replay (NG >= 18)

- Prerendering

- Hybrid Rendering (NG >= 19)

- Incremental Hydration (NG >= 19)

Agenda

ANGULAR
ARCHITECTS

Server-Side Rendering (SSR)

Server-Side Rendering (SSR)

– Problem: After download rendering on the client takes too long
  – Search Engines may not be abled to index the App correctly

– Identify: After .js files loaded js main thread takes too lon
  – Search Engines don't index correctly

– Solution: Use Angular SSR
  – Page is rendered on the server and then served to the client
  – But only useful for public pages (no user login)

Server-Side Rendering (SSR)

– be careful
  – no document (has to be injected)
  – no localStorage / sessionStorage

– wrapper
  – https://taiga-family.github.io/ng-web-apis/common

Demo

SSR

ANGULAR
ARCHITECTS

Server-Side Rendering (Angular 16)

– New feature called "full application hydration"

Event Replay (Angular 18)

- Problem: *Clicking and interacting with app before hydration*

- Identify: Long server response time when using Universal SSR

- Solution: Event Replay

```
export const appConfig: ApplicationConfig = {
  providers: [
    provideClientHydration(withEventReplay()),
    [...]
  ],
};
```

Prerendering (SSG)

Prerendering (SSG)

– Problem: Server response to slow, page needs to be rendered

– Identify: Long server response time when using Universal SSR

– Solution: Prerender the important pages on the server
  – Built-in Angular Universal since V11
  – Activated by default since V17

– Also works on servers without node.js (e.g. nginx / Apache)!

Demo

SSG

Hybrid Rendering (Angular 19)

- CSR Routes
  - Regular SPA (without SSR)
  - Server serves static files

- SSR Routes
  - Live content + Hydration
  - Server renders the routes

- Pre-rendered routes
  - Built time content + Hydration
  - Server serves built time rendered

```
{
  path: 'charts',
  renderMode: RenderMode.Client,
},

{
  path: 'home',
  renderMode: RenderMode.Server,
},

{
  path: post',
  renderMode: RenderMode.Prerender,
},
```

Lab 05 SSR & SSG

Server Side Rendering & Prerendering

Incremental Hydration

Incremental Hydration (Angular 19)

- ergonomic API
  - known from @defer

- improve performance
  - initial load
  - other CWV

- available in v19

```typescript
export const appConfig: ApplicationConfig = {
  providers: [
    provideClientHydration(
      withIncrementalHydration()
    ),
    [...]
  ],
};
```

hydrate on

- hydrate on immediate (default)
- hydrate on viewport
- hydrate on hover
- hydrate on interaction
- hydrate on timer(4200ms)

hydrate when

- specifies an imperative condition as an expression that returns a bool
    - best used: boolean flag

- if the condition returns to false, the swap is not reverted
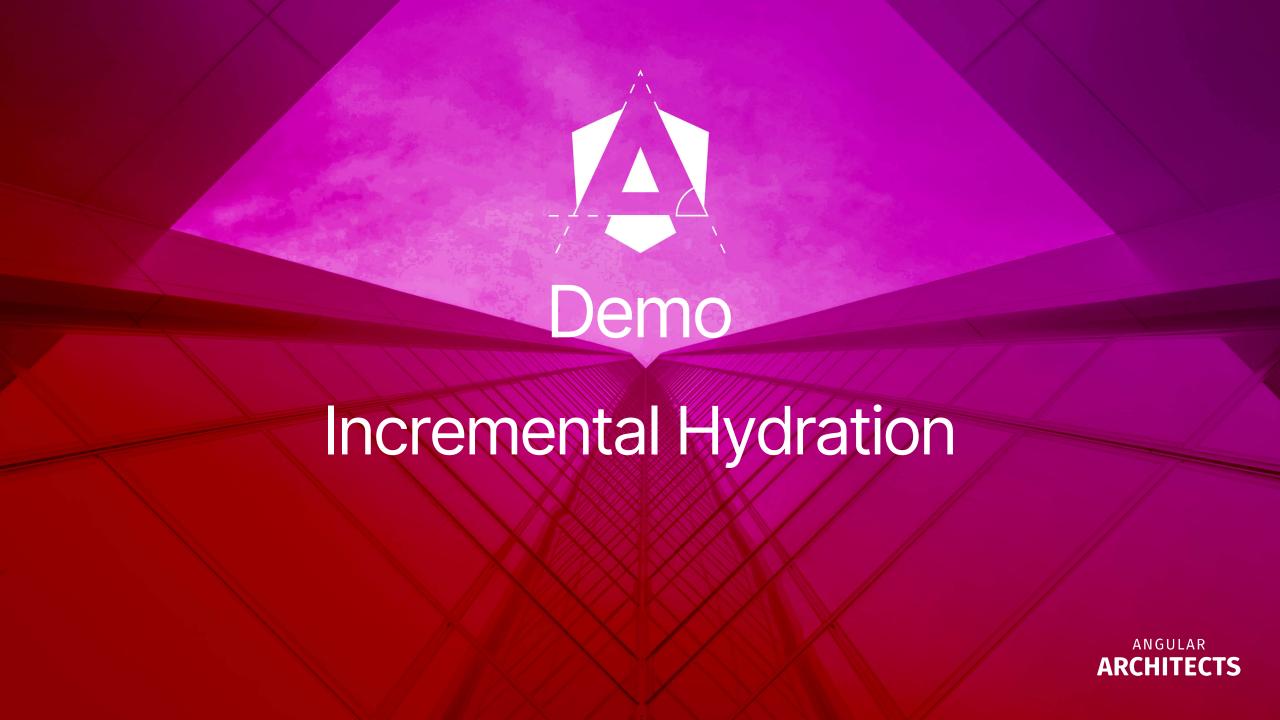   - it is a one-time operation

```
…
@defer (hydrate when condition) {
  <aa-lazy-component />
}
…
```

hydrate never

- component will be rendered but will not be hydrated

- means that even if the application is fully loaded on the client side, the defer block will remain static and not become interactive

```
…
@defer (hydrate never) {
  <aa-lazy-component />
}
…
```

Demo

Incremental Hydration

ANGULAR
ARCHITECTS

Alternative to SSR: Use a (URL) cache

Alternative Solution:

– Of course you could also use an alternative caching solution
  – E.g. Cloudflare or any other CDN

SSR & SSG

– Server-Side Rendering (SSR)

– **Hydration** (NG >= 16)

– Event Replay (NG >= 18)

– **Prerendering**

– Hybrid Rendering (NG >= 19)

– **Incremental Hydration** (NG >= 19)

Recap

References

- Angular Architects Blog
  - [Server-Side Rendering](#) (Blog series)

- Angular Docs
  - [Server-side rendering](#)

Questions?

ANGULAR
ARCHITECTS