Exploring Angular 18

Unveiling the Future of Angular Development

Angular | Rangle.io

Robyn Dalgleish
Solution Architect
@responsiverobyn

Sumit Arora
Senior Solution Architect
@arorasumit

FRONTEND DEVELOPMENT / JAVASCRIPT / TYPESCRIPT

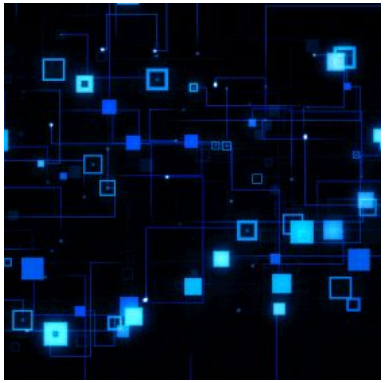
The Angular Renaissance: Why Frontend Devs Should Revisit It

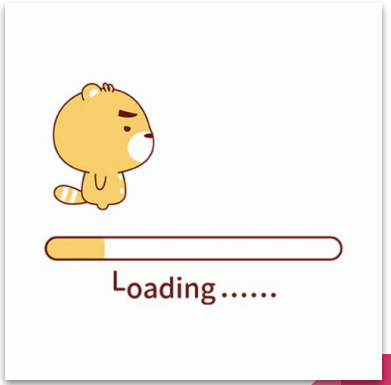
Angular is experiencing a renaissance, according to Progress senior developer advocate Alyssa Nicoll. She explains what that means.

Sep 26th, 2023 8:15am by Loraine Lawson

Photo via Unsplash

+ zone.js





Zoneless Angular

Zoneless Angular

Angular Signals

- Achieve high performance
- Simplified framework concepts
- Easier model infrastructure & dataflow
- More maintainable and reliable

Computed
Angular Signals

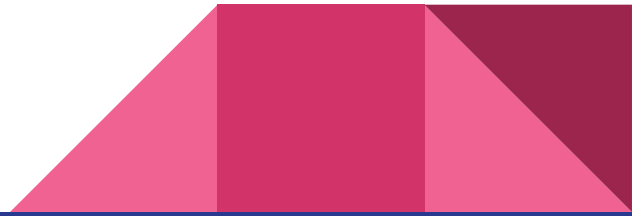
Signal

Writable signals provide an API for updating their values directly

Computed signal are read-only signals that derieve their value from other signals

Effect

An effect is an operation that runs whenever one or more signal values change.



Angular Signals

Signal Input

Signal Queries

New Output

Model Inputs

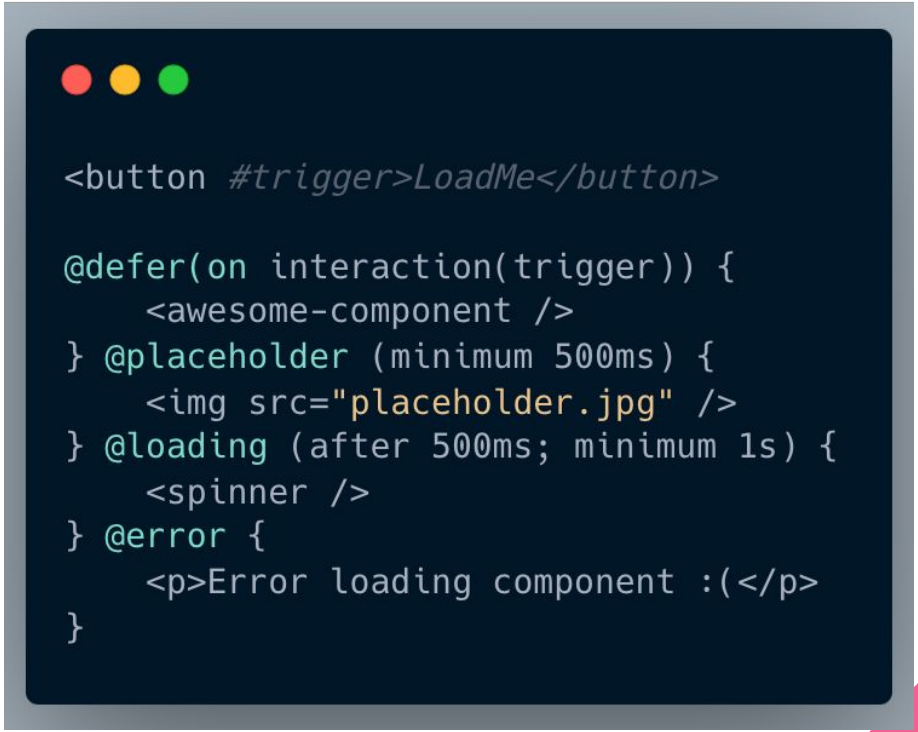
Loading Components

```
export const ROUTES: Route[] = [
  {
    path: 'admin',
    loadComponent: () => import('./admin/panel.component').then(mod =>
mod.AdminPanelComponent)
  {
    path: 'admin',
    loadChildren: () => import('./admin/routes').then(mod => mod.ADMIN_ROUTES)
  },
];
```

Deferred Loading

- Improved Performance
- Enhanced User Experience
- Simplified Code Management
- Optimized Resource Usage
- Better Control Over Loading

Deferred Loading



```
<button #trigger>LoadMe</button>

@defer(on interaction(trigger)) {
    <awesome-component />
} @placeholder (minimum 500ms) {
    <img src="placeholder.jpg" />
} @loading (after 500ms; minimum 1s) {
    <spinner />
} @error {
    <p>Error loading component :(</p>
}
```

New Control Flow

- `@if`
- `@for`
- `@switch`

Default Content

Default Content

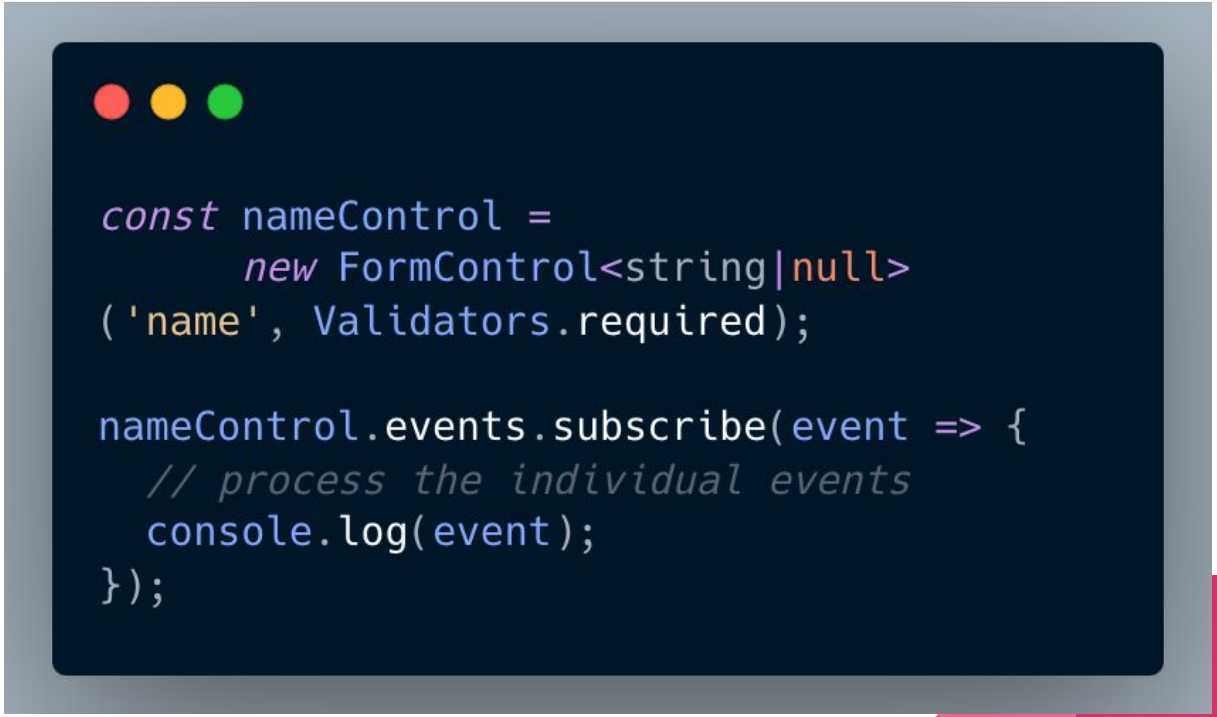
```
<app-header />
```

Default Header

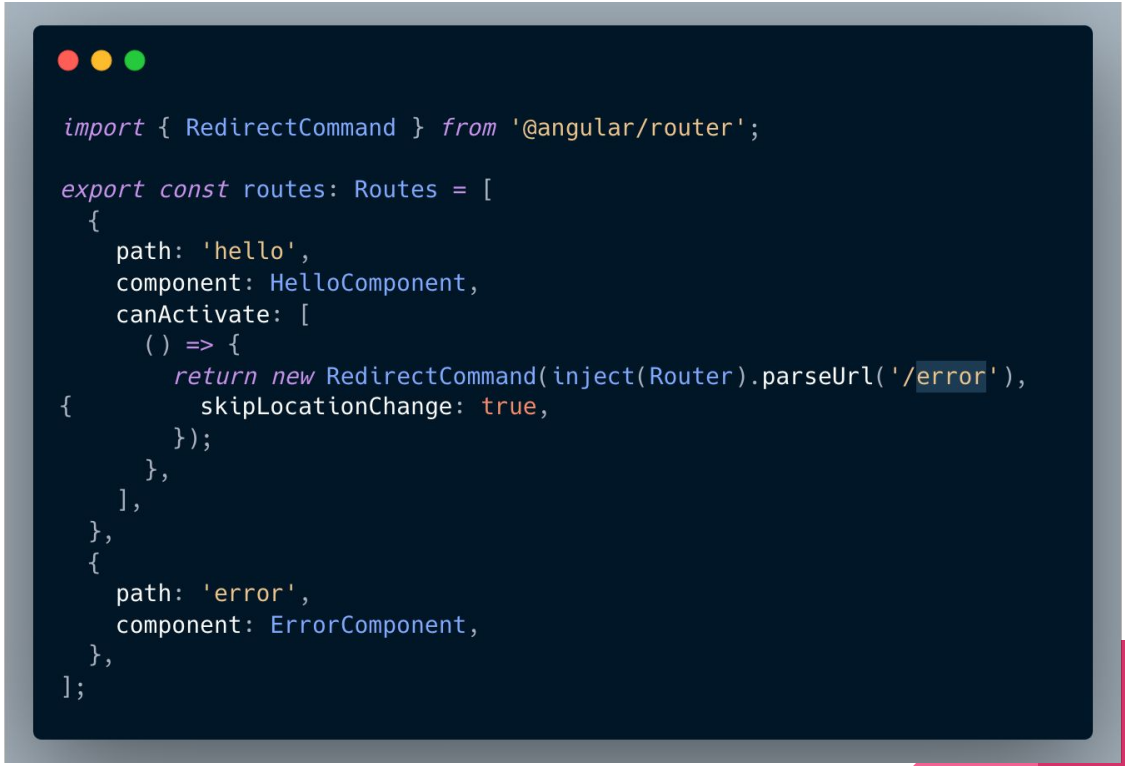
```
<app-header>
   <header>App Header</header>
</app-header>
```

App Header

Unified Form Control Events



Redirect Command



```
import { RedirectCommand } from '@angular/router';

export const routes: Routes = [
  {
    path: 'hello',
    component: HelloComponent,
    canActivate: [
      () => {
        return new RedirectCommand(inject(Router).parseUrl('/error'),
{         skipLocationChange: true,
        });
      },
    ],
  },
  {
    path: 'error',
    component: ErrorComponent,
  },
];
```

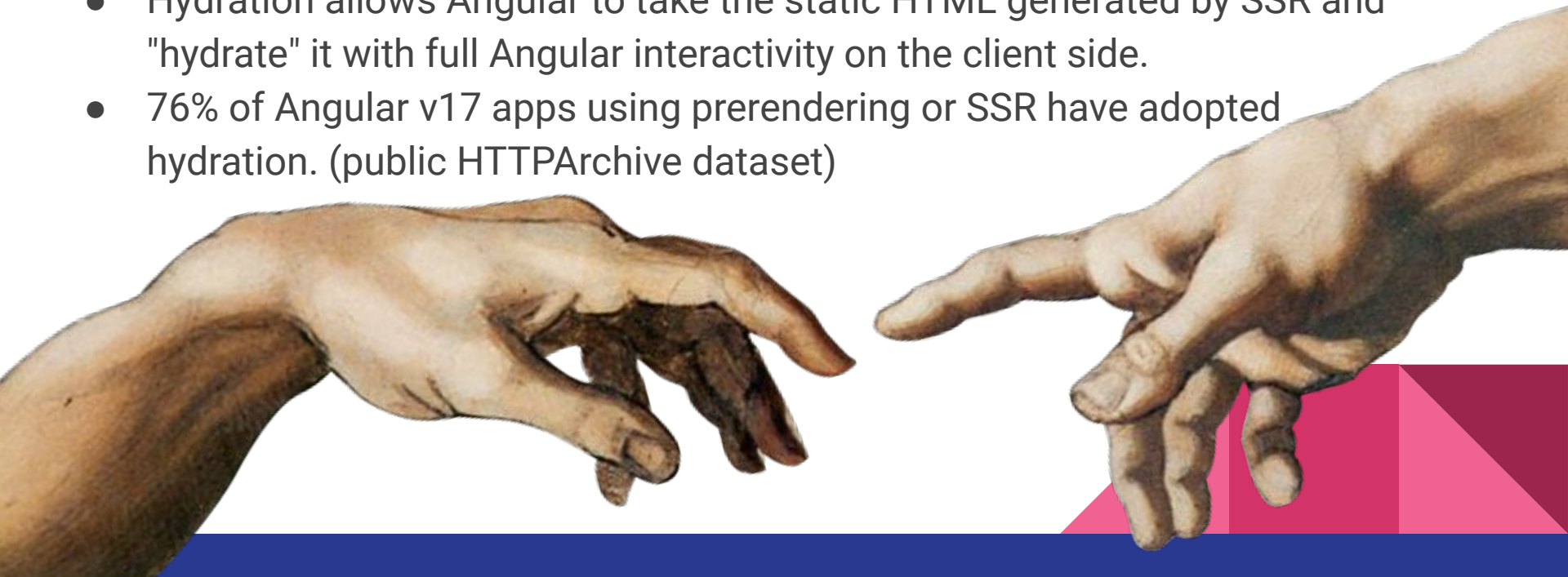
Improved Server-Side Rendering (SSR)

SSR with Angular v2:

- Initial Server-Side Rendering
- Delivery to Client
- Angular Bootstrapping
- Client-Side Interactivity

Improved Server-Side Rendering (SSR) - Angular 17

- Angular 17 introduced hydration, marking a significant leap forward.
- Hydration allows Angular to take the static HTML generated by SSR and "hydrate" it with full Angular interactivity on the client side.
- 76% of Angular v17 apps using prerendering or SSR have adopted hydration. (public HTTPArchive dataset)



Improved Server-Side Rendering (SSR) - Angular 18

- Angular 18 has taken hydration a step further by introducing support for internationalization (i18n) blocks.
- Event replay allows replaying user events (e.g. clicks) that happened on a page before hydration logic has completed.
- Hydration support in Devtools

More Changes

- angular.dev
- Material 3 is now stable (with hydration support!)
- Updated Angular DevTools visualize Angular’s hydration process
- Automated migration to the application builder

What's Next for Angular?

Upcoming Changes

- Partial Hydration
- Server route configuration
- Signal debugging in Angular DevTools
- Improved hot module reload
- Streamed server-side rendering
- Authoring format improvements
- Support two-dimensional drag-and-drop