

Beyond the Redirect

Modernizing Keycloak for the TypeScript Era

Erik Jan de Wit



edewit



social.linux.pizza/@edewit



blog.nerdin.ch

Keycloak Nightmare DX

The Old Enterprise Way:

- Fighting with clunky Freemarker Java templates just to change a CSS style.
- Hard to integrate into your application
- A backend security "black box" that frontend developers try to avoid at all costs.



Default Keycloak Theme

"Looks secure, feels like 2005."

Why "Better Auth" Wins Hearts

The Bait:

- **Magic Type Inference:** Instant autocomplete for roles and permissions right inside the IDE.
- **Native UI Control:** No redirects; the login forms are standard components inside the application.

The Catch:

You sacrifice standalone, battle-hardened architecture, multi-realm isolation, and OIDC protocol compliance just for better DX.

Typesafety

```
// The DX developers fell in love with:  
const session = await auth.getSession();  
  
if (session.user.role === "admin") {  
    // typed out-of-the-box!  
}
```

Components



daveycodez

daveycodez@example.com



```
import { UserButton } from "@components/auth/user/user-button";

export function UserButtonDemo() {
  return <UserButton />;
}
```

Let's bring this to keycloak

Introducing keycloak-headless

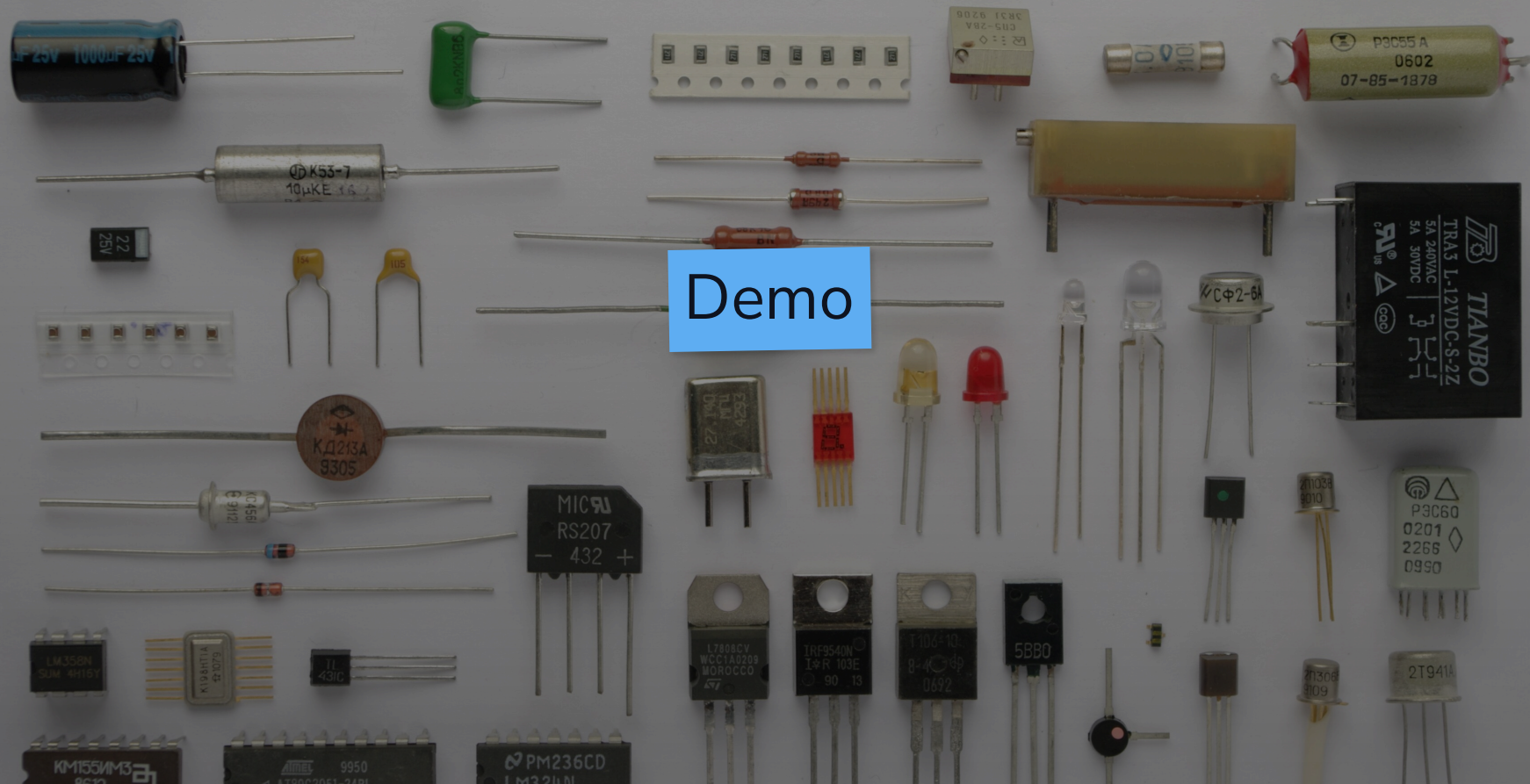
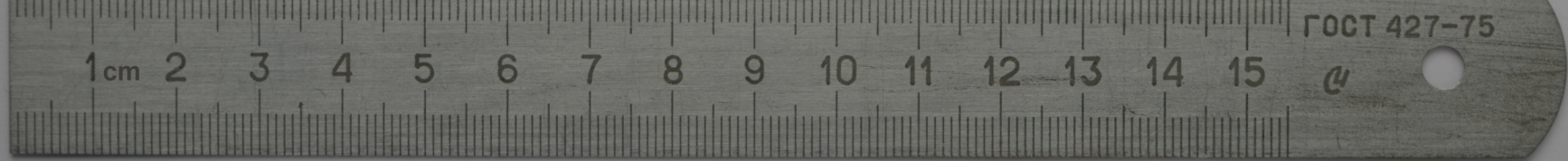
```
import "keycloak-headless/provider";
import { KeycloakProvider, useAuth } from "keycloak-headless/react";

export function App() {
  return (
    <KeycloakProvider
      url="https://keycloak.example/auth/" realm="myrealm" clientId="my-spa"
    >
      <kc-login-button>
        <button>Login</button>
      </kc-login-button>

    </KeycloakProvider>
  );
}
```



That's nice, but what about other frameworks



Login UXD

Freemarker nobody likes it!

Why can't I use the framework I already know

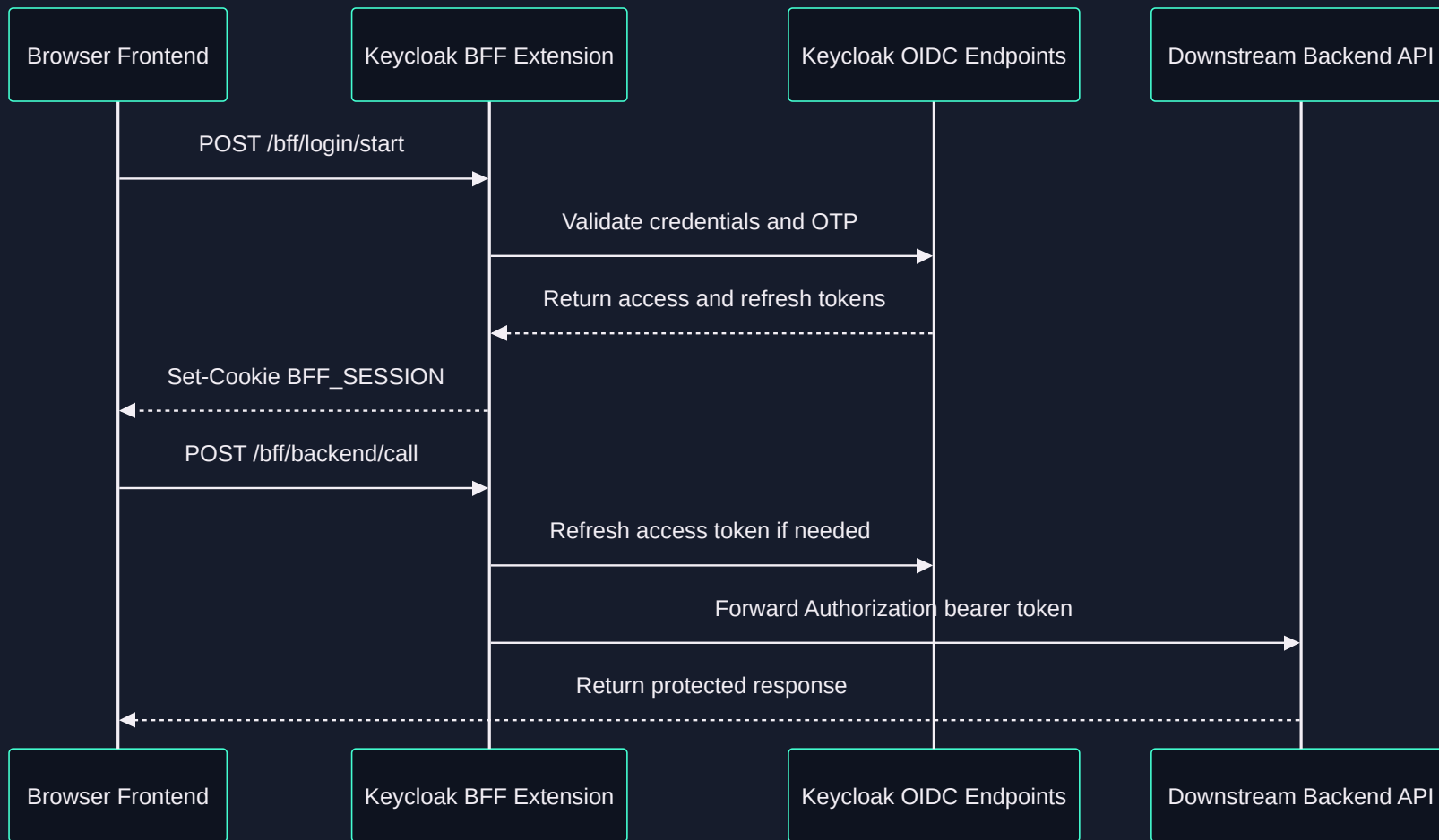
Introducing login-ui-base

- a reusable foundation for customizing Keycloak login UIs
- It provides two modes:
 - client-side rendering
 - Qute for server-side
- A CLI that scaffolds child themes
- Local dev workflows with live reload

Backend-for-Frontend

Keycloak extension that gives frontend apps a secure BFF auth layer.

- Uses ``BFF_SESSION`` cookie so the browser does not manage tokens directly
- Supports login, OTP, session check, logout, and backend proxy
- Adds CSRF validation on browser ``POST`` routes
- Demo stack: ``examples/docker`` -> ``docker compose up --build``



Security Trade-Offs

Security gains

- Access and refresh tokens stay server-side instead of in browser storage
- Centralized token refresh and backend call policy enforcement in one layer
- Built-in CSRF origin checks on sensitive browser `POST` endpoints

Trade-offs and new risks

- Session cookie theft becomes high impact unless HTTPS + secure cookie handling are strict
- BFF is now a high-value target with access to token exchange and backend propagation
- Extra operational burden: session store hardening, client secret management, proxy/header policy tuning
- Password grant style login (`username` / `password` to BFF) is a weaker model than code flow + PKCE

Demo

IBM Kubecost

IBM

Agentic AI that scales with you

Power trusted AI
with better data and

Open Source Projects

 github.com/edewit/keycloak-headless

 github.com/edewit/keycloak-react

 github.com/edewit/login-ui-base

 github.com/edewit/Backend-for-Frontend

Overview site

 blog.nerdin.ch/keycloak-frontend-tools