Connecting with Frontend Applications

This guide explains how to establish a connection between a frontend single-page application (SPA) — such as those built with React, Vue, or Angular and a Keycloak identity provider using the official Keycloak JavaScript adapter. It walks through the necessary setup, configuration, and execution of a protected login flow.

Variables

Certain parameters must be provided to integrate a frontend application with Keycloak. Below is a breakdown of each required variable, its purpose, and where to find it. Here's what each variable represents:

Variable	Description	Purpose
`URL`	Full Keycloak realm URL (e.g., https://your-domain/realms/your-realm)	The base endpoint for authentication, token requests, and user info
`CLIENT_ID`	Client ID from the Keycloak Admin Console	Identifies the SPA in Keycloak
`REALM`	The realm name where the client is defined	Defines the identity space
`REDIRECT_URI`	The URL where the frontend app should return after login	Must be registered in Keycloak as a Valid Redirect URI

These values can be found under **Clients > [Your Client] > Settings** in the Keycloak Admin Console.

Prerequisites

Install Node.js and NPM

Check if Node.js is installed:

```
node -v
```

If not, download and install from https://nodejs.org.

Set Up Frontend Project

Create a frontend project using your framework of choice. For example:

- **React:**

```
npx create-react-app keycloak-app
cd keycloak-app
```

- **Vue:**

```
npm init vue@latest
cd keycloak-app
```

- **Angular:**

```
ng new keycloak-app
cd keycloak-app
```

Then install the Keycloak JS adapter:

```
npm install keycloak-js
```

Code

Create a file named keycloak.js inside your src/ directory with the following content:

```
import Keycloak from "keycloak-js";

const keycloak = new Keycloak({
  url: "https://your-keycloak-domain",
  realm: "your-realm",
  clientId: "your-client-id",
});

export default keycloak;
```

Then update your app’s entry point (App.js, main.js, or main.ts) to initialize Keycloak:

Example (React - App.js):

```
import React, { useEffect, useState } from "react";
import keycloak from "./keycloak";

function App() {
  const [authenticated, setAuthenticated] = useState(false);

  useEffect(() => {
    keycloak.init({ onLoad: "login-required" }).then((auth) => {
      setAuthenticated(auth);
    });
  }, []);

  if (!authenticated) return <div>Loading...</div>;

  return (
    <div>
      <h1>Welcome, {keycloak.tokenParsed?.preferred_username}</h1>
      <p>You have accessed a protected frontend app using Keycloak.</p>
    </div>
  );
}

export default App;
```

Notes for Vue and Angular

- In Vue, you can wrap keycloak.init() inside a plugin and gate your app rendering using the onReady() hook.
- In Angular, use route guards (CanActivate) to protect routes based on Keycloak session state.

Execution

1. Replace all placeholders in the config with actual values from your Keycloak setup.
2. Start your frontend application:

```
npm start
```

3. Open your browser and navigate to:

```
http://localhost:3000
```

4. The Keycloak login page will appear. After authentication:
    - You’ll be redirected back to your SPA
    - The user info will be displayed, indicating successful integration

---

Revision #2
Created 17 June 2025 10:05:01 by kaiwalya
Updated 17 June 2025 10:07:48 by kaiwalya