Swisscom Design System – Server-Side-Rendering

Students

Tim Gamma

Natalia Gerasimenko

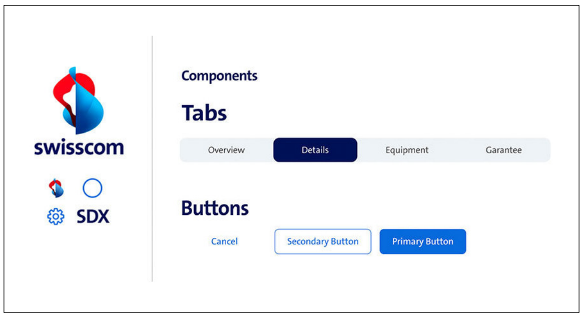
Introduction: Swisscom has its own design system, the Swisscom Digital Experience (SDX). It consists of a component library, UX principles, design guidelines, documentation, and rules. In this design system, Swisscom offers web components, reusable styled building blocks, build using StencilJS. This research project investigates the integration of Server-Side rendering (SSR) with these components. When built with StencilJS and combined with SSR, web components are not interactive and do incorporate incorrect CSS styling. The project focuses on addressing these issues. Additionally, it seeks solutions for integrating these web components within the frameworks Angular and Next.js.

Approach: The methodology involved a series of experiments to tackle the challenges in implementing SSR with StencilJS. These experiments included resolving interactivity deficits and rendering inconsistencies within Stencil's SSR framework, as well as exploring the integration of StencilJS web components in Angular and Next.js frameworks. The performance aspects of SSR were inspected in a simple Node.js environment using Express.js as a server and within an Angular application. The aim was to comprehensively evaluate the efficiency and responsiveness of SSR in different contexts.

Result: The findings revealed that manually adding ESM scripts post rendering resolved the interactivity issues. Incorrect CSS styling, linked to bugs in StencilJS, was addressed through a CSS-grid based workaround. The integration of StencilJS web components with Angular was achieved by defining Angular output targets and pre-rendering the components. Integrating these components into Next.js presented significant challenges. The investigation revealed that, in its current state, Next.js is not compatible with StencilJS. This necessitates the development of react wrapper generator aimed at encapsulating Stencil components within React components, enabling seamless integration into the Next.js framework. Regarding performance, SSR generally demonstrated a faster response than client-side rendering, particularly in the first contentful paint. However, this performance advantage was not consistent across all scenarios.

The outcome of this project showed that SSR is possible with StencilJS web components. However, we identified a number of remaining issues with the current tooling for Stencil SSR support, necessitating the development of custom solutions and workarounds. While the integration of StencilJS web components within Angular and Next.js is achievable, it demands significant setup and configuration, especially for Next.js. Nevertheless, as framework compatibility improves, the efficiency and user experience offered by SSR with StencilJS web components are expected to become more pronounced, highlighting the promising future of this technology in web development.

Example of Swisscom's web components
https://sdx.swisscom.com



Advisor
Prof. Dr. Markus Stolze

Subject Area
Internet Technologies and Applications