Apexon

SUCCESS STORY DIGITAL EXPERIENCE

EMPOWERING DEVELOPMENT THROUGH FRAMEWORK-AGNOSTIC COMPONENT LIBRARIES

Apexon streamlines application development with a best-aligned framework to maintain compatibility with a single-source-of-truth component library

This established healthcare organization was struggling to break away from the front-end framework on which its legacy applications were created. The client felt compelled to maintain future development in the same framework as their existing applications to ensure compatibility of their proposed component library.

Apexon recommended a different approach – advising the company to empower its front-end developers to build future applications with whatever framework was best-aligned with the specific application and business requirements.

Named 2021 Winner in the BIG Award for Business

Named in SaaS 1000, Official List Of Top 1000 SaaS Companies, Launches 2Q'18 List

Backed by Asurion

THE RESULTS

KEY OUTCOMES

FLEXIBLE DEVELOPMENT ENVIRONMENT

INCREASED EFFICIENCY; LESS TECHNICAL DEBT

IMPROVED DEVELOPER SATISFACTION

OUR METHODOLOGY

THE DIGITAL LIFECYCLE

Apexon works with companies across the digital lifecycle.

GO DIGITAL

LAUNCH & EXPERIMENT

Enable digital adoption in a quick, and agile manner

Apexon helped create a more flexible development environment - Decoupling the component library and design system from application frameworks offered a highly flexible development environment.

BE DIGITAL

AUTOMATE & ACCELERATE

Build digital infrastructure and foundation for enterprises to scale

Apexon created development environment for easy integration and consumption within target applications.

This new approach also enabled big gains in development efficiency and increased developer satisfaction, not to mention mitigating the company's technical debt.

EVOLVE DIGITAL

BE INTELLIGENT & AUTONOMOUS

Leverage data engineering to make strategic decisions and get digital right every time

THE CHALLENGE

SIMPLIFY THE DEVELOPMENT LIFECYCLE

The client was maintaining several legacy applications and wanted to build out a component library to assist in the development of several new applications. Development teams were not committed to the existing front-end framework for the new applications, but feared component library incompatibly.

Developers simply wanted the option to use whichever front-end framework best suited each project. However, key stakeholders didn't want teams reinventing component libraries every time a framework change was made. Both sides were dedicated to efficiency, but from very different perspectives.

Many organizations were looking to maintain consistent branding across applications using a centralized component library. This would reduce overall feature design and development time. However, with projects located around the world with many different teams, there were bound to be differing opinions about which framework best suited the needs of the team.

THE SOLUTION

A NEW APPROACH TO IMPROVE DEVELOPMENT EFFICIENCY & DEVELOPER RETENTION

Apexon recommended empowering development teams with a framework-agnostic component library with the help of Direflow - a React to Web Component transpiler. While this tool ultimately bound the component library to a front-end framework, the team felt this sacrifice was worth the freedom to build out future applications in whatever framework they felt fit best.

Apexon has found that providing developers the creative freedom that they need improves efficiency and happiness, ultimately yielding accelerated feature timelines and improved developer retention.

With this in mind, the team focused on delivering a universally-applicable component library with three key criteria:

Universally applicable, regardless of application's front-end framework

Easy integration and consumption within target applications

Support for quick and easy enhancements to component library, yet adherent to semantic versioning to mitigate introduction of breaking changes

MARKETING, SALES, & SERVICE EXECUTION

Criteria two and three were the easiest to solve, as the team had abundant prior experience in building out component libraries. Much of the support for consumption and simple enhancements was sourced from the implementation of the component library as a symbolically linked repository using npm link. The team considered options such as Bit, but ultimately determined there to be little value to individual, semantically-versioned components. Publishing and consumption as an npm package afforded a solution naturally conducive to semantic versioning, and paired with some node module best practices, ensured no breaking changes propagated to consuming applications.

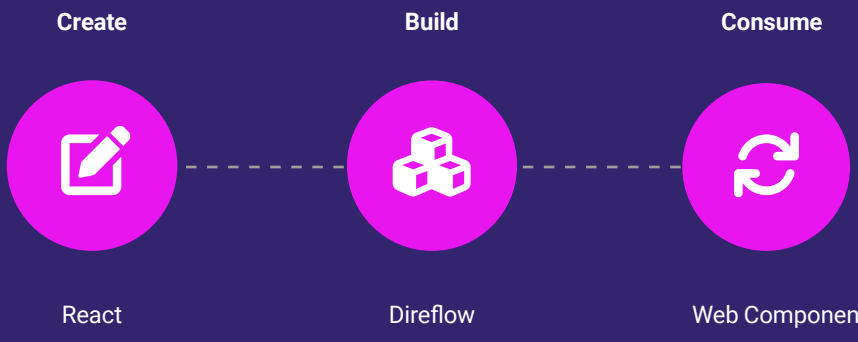
APPLICATION BUILDOUT

Criterion one was the project outlier and required the most deliberation. Digital demographics indicated that the majority of traffic would be sourced from modern devices operating with modern browsers. With this information in hand, the team recognized Web Components as an excellent solution for core users while support for targeted legacy browser versions could be obtained through polyfills. Currently though, developing and maintaining vanilla Web Components is not well supported. Many integrated development environments (IDEs) and tools supporting the concept of component libraries rely on a front-end framework like React or Angular to be in place. If the team hoped to present the component library through something like Styleguidist or Storybook, vanilla Web Components weren't the answer. React's front end framework addressed this issue head on.

HOW IT WORKS

- Use the power of React to create your component
- Build it, then use it and share it as a native Web Component

Source: Direflow



The impact of the new development framework has been significant

A More Flexible Development Environment

Decoupling the component library and design system from application frameworks offered a highly flexible development environment.

Freedom To Choose Tools

Teams no longer felt compelled to architect applications with a front-end framework simply because someone had previously committed them to it. Instead, they were empowered to elect the tools which best matched application requirements and best aligned with their interests.

In addition to the increased flexibility, this new approach also enabled big gains in development efficiency and increased developer satisfaction, not to mention mitigating the company's technical debt.