divotion

tech radar

Spring 2023

Our view on recent developments in our field

3rd edition

Introduction

Our experienced specialists participate every day in the development of many different software projects all over the Netherlands. Every six months our engineers gather to discuss the latest emerging trends and developments in software engineering. We seek to capture these trends in a technology radar. Each edition of the radar shows how these trends change, compared to the previous edition. A shift can indicate that we see a technology becoming more, or less, relevant for certain use cases.

If a trend does not show up in later editions, it signifies that there are no relevant developments or experiences that cause us to shift our assessment. A certain technique or framework that is not currently on the radar does not mean we do not like it or do not use it; it only says that we think it has not moved on the radar.

With this document, we will discuss the shifts we have observed over the past six months. This analysis guides us in deciding which technologies we use and recommend.

Tech Radar

The idea to create a tech radar was initiated by ThoughtWorks. They periodically showcase their view on new trends and development. In addition, they advise all to define their own radar.

At Divotion we fully support that view. Building a tech radar is an instructive and valuable experience, where we mutually share our knowledge and create a common awareness around technology. We believe that specialists should be able to create and maintain their own toolset to perform their job to their best ability. When composing a radar, you facilitate a broad discussion on technology, so organisations can strike the right balance between the risks and rewards of innovation. We can help you to kick-start these discussions in your organisation. Let your teams innovate and inspire each other. Together they can draw up a set of technologies and techniques that can accelerate your business.

Overview

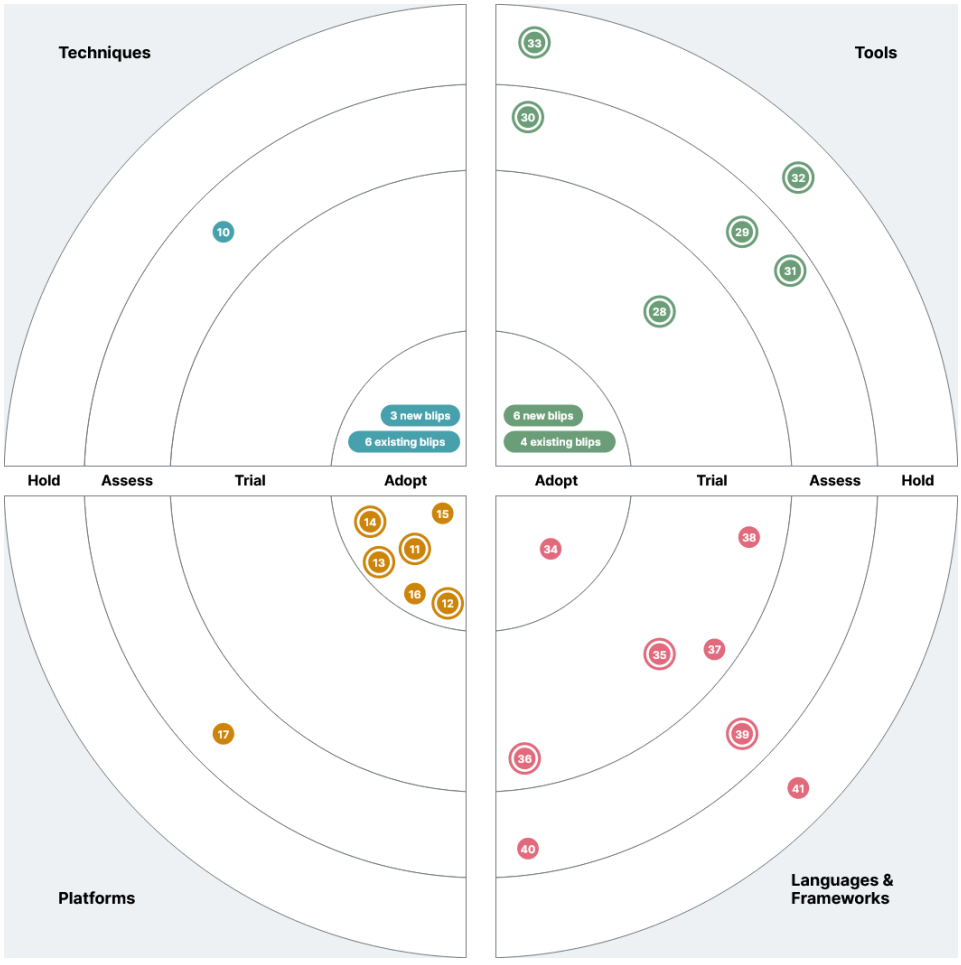
The radar consists of quadrants and rings, containing blips to indicate technologies of interest. The quadrants subdivide the different subjects into categories:

- **Techniques** help developers create better software.
- **Tools** aid in the development and delivery process.
- **Platforms** to deploy and execute programs.
- **Languages and frameworks** that support developers in their daily tasks.

The rings in each quadrant indicate which phase of the adoption cycle we believe a technology is currently at:

- **Adopt**: We recommend using this technology, wherever it fits the requirements.
- **Trial**: We advise to gain experience with this technology, wherever a project allows for a certain degree of risk.
- **Assess**: Interesting topic to learn more about and assess its future impact; yet too early to use in production.
- **Hold**: Do not deploy in a project not already using this technology.

In the subsequent sections we will delve into our views on recent developments in the field of software engineering more closely.



focus on front-end development

Techniques	Tools
ADOPT Design Systems Feature Sliced Design TypeScript BFF Automatic A11y checks Design tokens Monorepo Rendering Patterns Security Scanning Visual Regression Testing **TRIAL** - **ASSESS** HTTP/3 **HOLD** -	**ADOPT** Copilot Non-JS Compilers Redux Toolkit tRPC Turborepo Vitest Playwright PNPM Spectator Vite **TRIAL** Polypane **ASSESS** Rome TanStack Router Turbopack **HOLD** Create React App Jest
Platforms	**Languages & frameworks**
ADOPT Chromatic Datadog Ionic Vercel Deno Progressive Web Apps **TRIAL** - **ASSESS** Cross-platform app development **HOLD** -	**ADOPT** Lit **TRIAL** SvelteKit T3 Stack Qwik Remix **ASSESS** Mitosis Astro **HOLD** Elm

focus on front-end development

Techniques

Adopt

1. Design Systems
2. Feature Sliced Design
3. TypeScript BFF
4. Automatic A11y checks
5. Design tokens
6. Monorepo
7. Rendering Patterns
8. Security Scanning
9. Visual Regression Testing

Assess

10. HTTP/3

Design Systems

ADOPT

Design systems have become more popular in recent years. A well-designed design system can significantly improve an organization's design workflow, efficiency, and consistency, ultimately leading to better user experiences. A design system is designed, managed, and implemented by people. This means that a design system is something that keeps on living and growing.

We recommend the use of a design system to improve the design workflow and consistency of the design. However, setting up a design system requires a lot of time and effort. It is not a one-time thing, it is a long-term investment. This means that it is not suitable for all projects.

Feature Sliced Design

ADOPT

Feature Sliced Design (FSD) is a promising architectural pattern for improving the scalability of front-end projects. In essence, FSD organizes code based on its reusability level, shareability, and domain specificity. This process involves three steps: layers, slices, and segments. Layers represent application layers like shared or app, slices denote feature names, and segments refer to different technical implementations such as UI or API. Modules within a layer can only interact with modules from the layers directly beneath them, and slices within the same layer are restricted from interacting with each other.

We place Feature Sliced Design in the "adopt" category, as it is a crucial architectural pattern that meets modern front-end requirements. Maintaining project scalability and flexibility is essential, such as being able to transfer parts of the codebase to a new team. By adopting this approach, developers can create loosely coupled, scalable code that is easier to understand due to its interaction constraints.

TypeScript BFF

ADOPT

TypeScript can be used as technology to develop a backend for frontend for web applications. A backend for frontend (BFF) is a backend service that simplifies the frontend code by fetching and processing data from multiple APIs and providing a single, simplified API. This allows you to build a backend for frontend that is strongly typed and easier to maintain. TypeScript in combination with NodeJs or Deno is particularly suited to build scalable and efficient real-time applications.

We recommend TypeScript, you can develop a strongly typed backend for frontend that is simpler to maintain. When used in conjunction with NodeJs or Deno, TypeScript is especially well-suited for constructing real-time applications that are both scalable and efficient.

Automatic A11y checks

ADOPT

The Web is designed to work for all people, whatever their hardware, software, language, location, or ability. In order to achieve this the W3C created the Web Content Accessibility Guideline (WCAG). The word "Accessibility" is often shortened to "A11y" (there are 11 letters between the a and the y).

There are three levels of accessibility you can reach. A is for basic accessibility, AA is the global standard, and AAA is the strictest level, for special applications. In the European Union, we need to comply with WCAG 2.1 AA for the web. To create more awareness in your project about these standards you can test your code or UI against these standards. Testing your code can be done by linters and unit tests in your IDE for example. Another approach is to test your DOM output, which represents how a browser processes and presents your website.

Automated A11y testing is a process of using software tools to evaluate the accessibility of a website or digital product. The goal is to identify and fix accessibility issues for users with disabilities. A variety of tools are available for conducting automated A11y checks, such as Axe, Wave, and Lighthouse. These tools scan the website or application for common accessibility issues, such as missing alternative text for images, improper use of headings, and insufficient color contrast.

We recommend automating your A11y tests. It's powerful to run them in your CI/CD pipeline and makes sure that your code is always accessible. Because of the current A11y-regulations for government websites and as of June 2025 for the financial sector, we have to adopt this way of working to make sure that our websites are always accessible. While automated testing can be a quick and effective way to identify issues, it is important to also perform manual testing and involve individuals with disabilities in the testing process. This ensures a comprehensive and inclusive evaluation of accessibility.

Design tokens

ADOPT

Design tokens are values that store design-related properties such as colour, typography, and spacing. They are stored in a central location and can be accessed by various design and development tools. They ensure consistency across products and platforms, and facilitate collaboration between design and development teams. Design tokens provide a shared language for discussing design-related values and makes it easier to update and maintain design systems. They are a powerful tool for creating and managing design systems and can improve efficiency, consistency, and collaboration in workflows.

Design tokens are highly recommended and placed in the "adopt" ring because of their numerous advantages, including improving consistency and collaboration. We suggest starting with design tokens and implementing them in a design system if needed. They provide a shared language for discussing design-related values, making it easier for teams to collaborate.

Monorepo

ADOPT

A monorepo is a single code repository that holds all code and assets for all projects. You can combine both front-end and backend projects in that one repository, but you can also only have a separate monorepo repository for the front-end projects and a monorepo repository for the backend projects.

A monorepo can have its advantages and its disadvantages. We recommend the use of a monorepo if you have a lot of projects that are closely related to each other. By storing multiple projects or components within a single repository, developers can easily share code across projects. This can lead to greater code consistency and easier maintenance. This also directly leads to a bigger code base, which can be a disadvantage.

Setting up a monorepo can be a bit of a challenge. But once you have it set up, it can be a great way to manage your projects. Currently there are a few tools that can help you set up a monorepo. The most popular ones are Nx and Turborepo. These tools can help you keep track of the dependencies between your projects and can help you set up a good developer experience.

Rendering Patterns

ADOPT

Rendering patterns are different strategies for rendering your app. A pattern used by the big frameworks React, Vue, Angular, etc. is that of a Single Page Application (SPA) which uses Client Side Rendering (CSR). The new frameworks like Nuxt, Next, SvelteKit, and Angular Universal allow for Server Side Rendering (SSR) or Static Site Generation (SSG). In React 18 new features like Server Components with streaming just came out. Next.js is implementing Server Components with streaming in their beta version.

When we look at Nuxt, Next, Angular Universal and SvelteKit, we see wide support for SSR. To even reduce Time to Interactive (TTI) more and to improve Developer Experience we see active movement within Rendering Patterns. React now has new improvements to combine SSR with CSR and to have increment updates we therefore place Rendering Patterns in adopt.

Security Scanning

ADOPT

In the world of front-end development there are a lot of external libraries to use. Those can be open source or paid to receive support and updates by a commercial company. All of these libraries can be old or contain vulnerabilities and should be up-to-date to prevent security problems. Your own code can also contain vulnerabilities apart from the used external libraries. To find potential problems a Security Scanner can be used in a CI/CD pipeline to audit the code.

It is essential to be aware of old libraries to update and vulnerabilities in your code or even in the used libraries. To automate the process of scanning you can use tools like SonarQube to scan your own code and use Snyk or Renovate Bot to keep track of used dependencies. There are multiple choices to make and most of them are valid choices. You should at least think about security and choose the Security Scanning option that fits your project and organization.

Visual Regression Testing

ADOPT

Visual regression testing is a process of comparing visual snapshots of a web application or website before and after changes have been made. This technique is useful for identifying any visual discrepancies, such as layout or style changes, that may have been introduced during the development process. By conducting visual regression testing, developers can quickly identify and fix any issues that arise after changes have been made to a website or application. This helps to ensure that the user experience remains consistent and that the site or app functions as intended.

One of the main benefits of visual regression testing is that it allows developers to see exactly what has changed in the application's appearance. This helps to identify any areas where the code has inadvertently broken existing functionality or design. With visual regression testing, developers can ensure that the changes they are making do not have unintended consequences on the application's visual appearance or usability. Overall, this technique helps to improve the quality and reliability of web applications and websites.

HTTP/3

ASSESS

HTTP/3 is a new standard for web browser and server communication. Where HTTP/1 is not efficient in the communication, HTTP/2 and HTTP/3 are much more efficient and faster. HTTP/3 uses a new transport protocol called QUIC and solves some problems with HTTP/2. For example smartphones that switches from Wi-Fi to cellular data, and it prevents the blocking of all other network communication when packets are lost.

HTTP/3 is quite new and is now in the Request for Comments (RFC) phase. Despite it is not an official standard yet, Google already uses it for Gmail in production for example. Currently all mayor modern browsers, except for Safari, support HTTP/3. To use it, the backend infrastructure has to be ready for it. This is not as trivial as it seems, because all reverse-proxies and other infrastructure components (if applicable) have to support it well. We would suggest asking the infrastructure team inside your company to at least have a look into HTTP/3 and find out what is needed to implement it on the infrastructure side. Most likely HTTP/3 will be the new standard and, when implemented well on all sides, it can improve the experience a user has with the web application you developed.

Tools

Adopt
18. Copilot
19. Non-JS Compilers
20. Redux Toolkit
21. tRPC
22. Turborepo
23. Vitest
24. Playwright
25. PNPM
26. Spectator
27. Vite

Trial
28. Polypane

Assess
29. Rome
30. TanStack Router
31. Turbopack

Hold
32. Create React App
33. Jest

Copilot

ADOPT

CoPilot is an AI-powered code assistant developed by GitHub, designed to help front-end developers write efficient and effective code. Supported by ChatGPT, CoPilot provides intelligent suggestions and code examples, seamlessly integrating into your existing workflow. It works with popular code editors, saving you time and energy while building modern and responsive web applications.

We recommend CoPilot for front-end developers due to the benefits in terms of speed, efficiency, and code quality. CoPilot stays up-to-date with the latest front-end technologies and best practices, ensuring teams are always informed about the newest developments. Although there are some potential downsides and risks, such as blindly trusting the generated code and privacy concerns, these can be mitigated by adopting a balanced approach and carefully evaluating your organization's policies. With the right precautions, CoPilot is a valuable addition to your development process and helps you grow as a developer.

Non-JS Compilers

ADOPT

The future of bundling is changing rapidly, with new bundlers emerging and challenging the dominance of tools like Webpack. Modern bundlers are now more advanced, and efficient than ever before, each with their own unique approach. They have come a long way since the early days of simple concatenation and minification.

The use of non-JS compilers is a trend we want to follow. When used with the right tooling it can reach lightning-fast build times of like Turbopack (Rust based), the native ES modules of Vite (Go based), or SWC (Rust based), a tool for transpiling JavaScript code. We think that the future of bundling is in the hands of these tools.

Redux Toolkit

ADOPT

Redux Toolkit can greatly simplify your codebase. This toolkit reduces Redux boilerplate code and improves code organization. TRK Query, a bundled add-on, offers a powerful tool for fetching and caching data. It also supports server-side rendering (SSR), ensuring a consistent initial state on both server and client. The main reason to use Redux Toolkit is to improve the developer experience: Redux Toolkit provides a set of debugging tools that help you trace the flow of data through your Redux store, making it easier to understand how your application's state is changing over time.

We are recommending adopting Redux Toolkit as State management. You can benefit from streamlined development, faster development times, maintainable code, and easier server-side rendering (SSR) state management. Furthermore, it encourages best practices for writing Redux code, including using immutable data structures and reducing unnecessary component re-renders.

tRPC

ADOPT

TypeScript Remote Procedure Call, or tRPC, is a lightweight library. It aims to provide TypeScript interference for building frontend and backend communication. It's an alternative to generating OpenAPI or GraphQL API definitions. The library requires having TypeScript on both the client and the server. Because of tRPC's nature, the ideal application of tRPC is combining a TypeScript "Backend For Frontend" that is located in a Monorepo together with the frontend.

You can adopt tRPC because it offers the benefits of type interference between your frontend and backend. The benefits of type interference are typesafety and autocompletion. Other benefits offered by tRPC are input and output validation, little boilerplating, JavaScript framework agnostic, no extra build/compile steps, and 0 dependencies in client package.

Turborepo

ADOPT

Turborepo is an open-source tool for managing monorepos. Turborepo helps developers manage and organize their monorepos more efficiently by automating tasks like package installation, version management, and cross-project dependencies. It uses a caching system to speed up the development process and reduce the amount of time spent on repetitive tasks. Turborepo is created by the team at Vercel, the company behind Next.js and other popular open-source projects.

We recommend Turborepo for developers who are working on large-scale projects. It's a great tool for managing monorepos that can help simplify the development process and increase productivity for large-scale projects. It is easy to use and can speed up the development process by automating repetitive tasks.

Vitest

ADOPT

Vitest is a test framework written in Rust. It is designed to provide a fast and reliable testing experience for developers. Compared to Jest, Vitest is a newer framework and is still evolving. It has native ES Modules and out-of-the-box TypeScript support, making it easy to use and integrate into projects. Additionally, it is multithreaded and lightweight, ensuring optimal performance and faster live reloads. It's similar to Jest, but with some differences. It is designed to be lightweight and easy to use. Vitest is also highly customizable, allowing developers to tailor it to their specific needs. Another key difference between the two frameworks is that Vitest is built with simplicity in mind, while Jest offers a more comprehensive set of features. However, this also means that Vitest may be easier for beginners to learn and use. Overall, the choice between Vitest and Jest will depend on the specific needs and preferences of the developer or development team.

We recommend Vitest, as a testing library for smaller JavaScript projects, it has several benefits that make it a compelling option for developers. With minimal configuration and setup needed, developers can focus on writing tests rather than worrying about tooling. Furthermore, Vitest builds upon the Jest API, including all Jest testing features, making it highly compatible with Vite projects. However, being in the early phase, Vitest may not have active community support, and its approach of isolating each test to run may impact performance when scaling.

Playwright

ADOPT

Playwright is a fairly new testing and automation framework from Microsoft. It was developed by the same team that worked on the Chrome developer tools and Puppeteer. It supports various programming languages and testing types such as End-To-End, API, UI component, and unit tests all within a single tool and a single API.

Playwright offers both headless and headed modes, without or with browser UI for all browsers and all platforms, including mobile and Safari for Windows. It executes tests in parallel (optionally over multiple machines) partly in the browser and partly in the programming language runtime, making it one of the fastest and feature-rich testing frameworks.

PNPM

ADOPT

PNPM is a package manager specifically designed for Node projects, serving as an alternative to npm and Yarn. The primary distinction between PNPM and its competitors is its unique approach to package storage. To manage packages, PNPM employs a global store and generates symlinks to each project's node_modules folder. During the symlink creation process, it ensures the use of exact package versions as specified in the lockfiles. Furthermore, PNPM mitigates the risk of "phantom dependencies" by mandating that all dependencies are explicitly listed in the package.json file and properly installed.

In conclusion, PNPM offers a more efficient and dependable package management solution by conserving disk space, expediting installation times, and minimizing dependency conflicts. Given these substantial benefits and the improved developer experience, we recommend adopting PNPM for your front-end projects.

Spectator

ADOPT

Spectator is a tool for Angular that helps you with easier testing. It is built on Testbed but without the verbosity of Testbed. Among other things, it makes component, HTTP and router tests easier to set up and is more readable. Spectator provides a way to automatically create mocks for components and providers using ng-mocks. Existing tests made without Spectator can still exist in the codebase and can be migrated to Spectator later on.

We have a positive experience using Spectator with both Jasmine and Jest. It is a tool that improves developer experience, because it makes testing much easier with less code. If you have not tried Spectator in your Angular project we highly recommend to have a look at it.

Vite

ADOPT

Vite is a modern build tool and development server designed for web development, which emphasizes speed and simplicity. It leverages the native ES modules of modern browsers to achieve near-instantaneous builds and fast refreshes during development. Additionally, Vite supports hot module replacement, automatic code splitting, optimized production builds, and a variety of front-end frameworks and tools. Vite was inspired by two other build tools: ESBuild and Rollup. ESBuild is a fast and lightweight build tool that directly transforms JavaScript code, while Rollup is a module bundler that optimizes code for production. Vite combines the best of both tools to provide a streamlined development experience that also optimizes code for production. Overall, Vite's use of ES modules, development server, and a simple API make it a popular choice among web developers looking for an efficient and effective build tool for their projects.

We recommend Vite as an alternative for create-react-app projects build with React and Storybook. It is also already used by frameworks like Vue and Nuxt. Unlike create-react-app, Vite does not build your entire application before serving, instead, it builds the application on demand. It also leverages the power of native ES modules, esbuild, and Rollup to improve development and build time. If you don't want to use a full-blown framework like Next.js or Nuxt, Vite is a great

alternative to keep the flexibility in your project but also improve the development experience.

Polypane

TRIAL

Polypane is a browser made for web developers. It offers a wide range of developer tools for optimizing responsiveness, accessibility, and search engine optimization. This browser also supports Chrome extensions.

We recommend to try out this browser if you're concerned with optimizing your web application for responsiveness, accessibility, and search engine optimization. Having all the tools provided by Polypane together in one browser for this can save developers from minutes to hours per week. Polypane's wide range of tooling covers common challenges like responsiveness, but also very specific challenges like colour blindness.

Rome

ASSESS

Rome is a kit of development tools built from the ground up in Rust with no dependencies. They have a vision where combining the functionality of separate tools will lead to simpler configuration, amazing performance, and adaptability to any stack.

As of now, they have a formatter and a linter. We will be awaiting the release of the remaining tools, consisting of a compiler, bundler, and testing solution. Once those are ready this may prove to be a great stack for developing your applications, but for now we recommend to hold off on it until it has matured.

TanStack Router

ASSESS

TanStack Router is a cutting-edge TypeScript library that offers a fully typesafe router with powerful first-class search-param APIs and client-side cache-aware design. This library supports multiple popular frameworks, including React, Angular, SolidJs, Svelte, and Vue, and builds on established concepts and patterns from NextJs, Remix, and TRPC.

While still in beta, we think TanStack Router is a promising library worth assessing as it can improve your routing logic and provide a more efficient development experience. TanStack Router's seamless integration with various frameworks makes it a promising addition to any project.

Turbopack

ASSESS

Turbopack is a modern, non-JavaScript bundler for frontend assets. It uses Rust, a programming language known for its speed and performance, to bundle and optimize code. Turbopack is designed to be very fast and efficient, which makes it a great choice for large-scale projects with many assets that need to be bundled quickly. It uses a technique called incremental bundling, a cached based strategy to optimize performance.

We think Turbopack has the potential to be a great bundler for large-scale projects, and we're excited to see where it goes. Turbopack is currently still in alpha release, and that is why we don't recommend using Turbopack for production projects just yet.

Create React App

HOLD

Create React App (CRA) was, up until recently, the officially supported way to create a Single-Page-Application (SPA) with React. It comes with an off-the-shelf build setup without any configuration.

We rate CRA with hold since it's no longer officially supported by the React team. And its last release already is from more than a year ago. The advice from the react.dev site is to use a full-stack React framework like Next.js and Remix instead. These all offer solutions to generate starter code for a new web app and come with an off-the-self build setup. Also, starting off with a full-stack React solution makes it easier to offload work to the server (like data fetching and pre-rendering).

Jest

HOLD

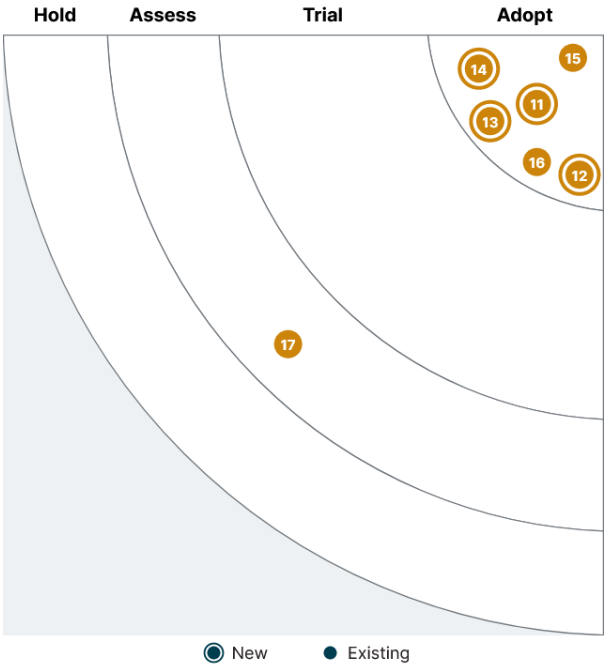
Jest is a powerful and flexible open-source testing framework that makes it easy to write and run tests for JavaScript applications. Jest offers a range of features such as test-driven development, code coverage, mocking, and snapshot testing, making it a comprehensive and efficient tool for testing applications. It is designed to be easy to set up and use, with features like zero-configuration setup and real-time feedback during test runs.

In our opinion, Jest remains a solid and reliable choice for testing JavaScript applications. However, other testing frameworks like Playwright, Vitest, and Cypress, offer unique features and advantages that may make them a better choice for certain use cases. Additionally, these frameworks are continually evolving and improving, making them a viable choice for the future. Jest is primarily used for testing JavaScript code, while Playwright and Cypress are end-to-end testing frameworks. Playwright stands out for its cross-browser testing capabilities, while Cypress offers a unique architecture and excellent debugging features. Vitest aims to position itself as the Test Runner of choice for Vite projects. Ultimately, the choice of testing framework will depend on factors such as the nature of the application, the testing requirements, and the preferences of the development team.

Platforms

Adopt
11. Chromatic
12. Datadog
13. Ionic
14. Vercel
15. Deno
16. Progressive Web Apps

Assess
17. Cross-platform app development



Chromatic

ADOPT

Chromatic uses Storybook as a base and has processes in place to gather UI feedback, execute automatic testing possibilities and documentation, so developers can iterate faster with less manual work.

We would recommend to adopt Chromatic as it helps the communication between different teams and especially between UXers and Developers. It has evolved enough now, especially now that Shopify, Adobe en BBC are also using it. The integrated feedback when visual changes are made gives direct communication where you can pin the feedback directly onto the bug. This helps a lot to know what still needs to be fixed and is much more clear compared to a textual comment without any visual reference. Besides the communication it helps developers to get quickly feedback about the changes they make and catch bugs early in the development cycle.

Datadog

ADOPT

DataDog is a platform for monitoring and security. It's a modern solution for logging, tracing, Synthetics testing and even Real User Monitoring. It can be implemented for any cloud platform or even on-premise by using turn-key integrations, combined with client tooling for nearly any target. It also provides tools for building dashboards within Datadog to gain insight into statistics and user behaviour.

Consider Datadog if you need a logging platform for complex applications where reliability, scaling and gaining insights are essential to growing your business. Especially when you have a complex architecture you will reap the benefits of tools like tracing from the front-end application to your back-end microservices.

Ionic

ADOPT

Ionic is an open-source platform that supports building apps cross-platform based on a single codebase using Web Views to render web technologies. The platform consists of a set of UI components and tools to run, build and deploy applications on web and mobile platforms. Ionic has support for popular web frameworks such as Angular, React, and Vue. Nowadays, the recommended way to build and deploy Ionic applications as native mobile apps is using Capacitor. Capacitor is a cross-platform native runtime. It was developed by the Ionic team to replace Cordova. Capacitor can run web apps natively on iOS, Android, Electron, and Web (using Progressive Web App technology). Capacitor provides a set of core plugins that enables the application to access the latest native SDKs and native APIs on each platform. Like the device file system, camera, and native location services.

The Ionic platform is a modern solution for cross-platform development based on a single codebase. We've added it to our tech radar because of its improvements in recent years. For a couple of years, Vue and React are supported now. And Ionic applications can be deployed to native devices using Capacitor, the modern successor of Cordova. Capacitor is created by the Ionic team using modern web technologies and has an improved developer experience.

Vercel

ADOPT

Vercel is a deployment platform for web applications, built on top of AWS. They are at the forefront of modern web app development. The company Vercel also heavily invests in tooling required for web development, ranging from their state-of-the-art deployment processes and performance optimisations to their continued support for build tooling.

By providing an excellent experience deploying applications using these tools, it's a win-win where we all benefit. The Divotion website runs on Vercel and for good reason. Their deployment process is excellent and functionalities such as feature deployments are essential to our workflow. Their integrations with all popular Git providers make sure to also reflect the build status right next to your commits. You should consider Vercel if you value fast and reliable deployments with little configuration.

Deno

ADOPT

Deno is a modern runtime for JavaScript and WebAssembly powered by Chrome's V8. Created by Ryan Dahl (the creator of Node.js) to solve fundamental design and security flaws and embraces modern best practices like ES Modules and TypeScript. It's very much 'batteries included' with TypeScript support, a built-in linter and code formatter, and IDE integration. As opposed to, Node.js, it offers much broader and spec-compliant support for platform / Web APIs. And it also features significant and broad performance optimizations like TypeScript type checking that's 4.7x faster.

We rate Deno with adopt due to its improved out-of-the-box experience, the much-improved platform / Web APIs, and its significant and broad performance optimizations. Deno offers a modern JavaScript runtime for building "non"-browser applications for the server-side or command-line.

Progressive Web Apps

ADOPT

Progressive Web Apps (PWAs) can be used to enhance your website or to create native-like apps with an icon on the home screen. PWAs are developed by integrating a manifest and service worker into your app. The service worker introduces an additional thread, enabling added functionality. A prime example is caching requests for offline capabilities and quicker load times. The service worker can also interact with the browser through various APIs.

Previously, only Android browsers supported the Push API. However, it is crucial to note that push notifications are now available on iOS 16.4, enabling their use on both platforms. This development significantly increases the relevance of PWAs, as push notifications are essential for marketing and recreating app-like experiences. Therefore, we place PWAs in the "adopt" category.

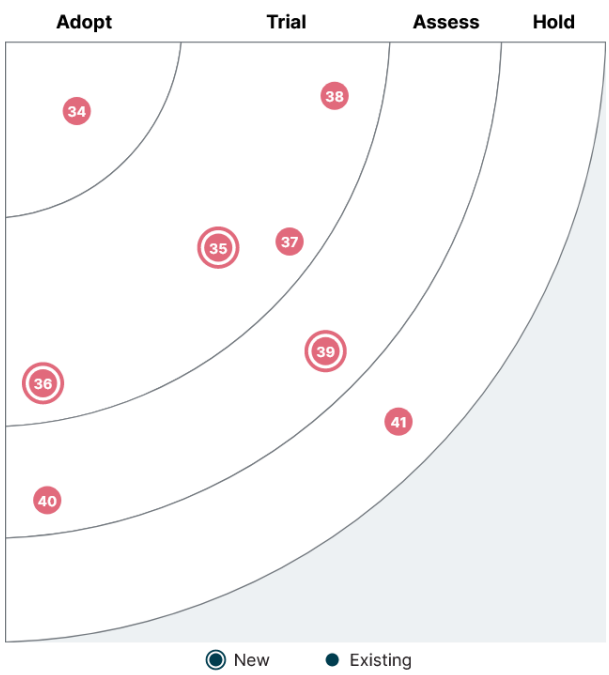
Cross-platform app development

ASSESS

Cross-platform app development allows developers to create apps that work on multiple platforms with a single codebase. Popular frameworks for this type of development include React Native, .NET MAUI (formerly known as Xamarin) and Flutter. These frameworks offer fast development cycles and a range of features and tools to help developers create high-quality apps that work across different devices and operating systems.

Most of the time we see our clients choosing dedicated web and app teams with separate code bases. This of course has its benefits, but having one codebase for multiple platforms seems like the holy grail. It is something to assess when starting a new project.

Languages & frameworks



Adopt
34. Lit

Trial
35. SvelteKit
36. T3 Stack
37. Qwik
38. Remix

Assess
39. Mitosis
40. Astro

Hold
41. Elm

Lit

ADOPT

Lit is a lightweight web UI library that leverages built-in support of web components provided by all modern browsers to build UI components. Instead of abstracting from it, Lit fully embraces the web platform and merely adds functionality that is (currently) missing in the web platform. And Lit in itself does not need any build step due to its usage of standard JavaScript tagged template literals for dynamic HTML template (as opposed to JSX) and `<script type="module">`.

We rate Lit with adopt and consider it a credible alternative to React, Vue, or Angular. Even though it might not be as comprehensive as other solutions, it is powerful enough to build complete web apps (as demonstrated by the ING bank). And every UI component that is build with Lit can also be used in any framework.

NOTE

While React can render Web Components, it cannot easily pass React props to custom element properties or event listeners. But that is easily fixed by creating wrapper React components with an NPM package like `@lit-labs/react`. And once React v19 is released, it should have 100% support for web components (see also https://custom-elements-everywhere.com/#react).

SvelteKit

TRIAL

Powered by Svelte and Vite, SvelteKit is a framework for building a reactive application. It replaces Sapper which was the previous attempt at building a framework for Svelte. It is very flexible and fast, supports various rendering patterns and comes with zero-configuration tooling. SvelteKit being a meta-framework for Svelte already comes with a bundler, router, TypeScript support and various deployment options out of the box.

With its focus on web development being fun and easy, SvelteKit is a promising framework. It will be especially suited for smaller and simpler websites where website speed and development iteration speed are a priority. SvelteKit has already reached a stable release and is actively being worked on, but it will have to prove itself in the longer term before broader adoption.

T3 Stack

TRIAL

The T3 Stack is an opinionated full-stack tech stack consisting of Next.js, TypeScript, tRPC, Prisma, Tailwind and NextAuth.js. This tech stack streamlines the setup of typesafe web applications. It's also provided with the create-t3-app command-line interface. This stack doesn't add everything. For example, you can still choose your own state management libraries or frameworks.

We recommend having a trial of this stack when setting up a new full-stack project because it offers typesafety throughout your entire application. Typesafety in your front-end using TypeScript, typesafety in your backend with tRPC and finally typesafety to your data source using Prisma.

Qwik

TRIAL

Qwik is an HTML-first framework that automatically lazy-loads all JavaScript code. It uses static HTML generated from server-side rendering and ships with less than 1kb on page load. All other JavaScript code is only lazy-loaded on a needed basis. Due to its Qwik optimizer, 'splitting points' for lazy loading are automatically created without relying on the developer to specify any loading boundaries.

We rate Qwik with trial even though it's still relatively young. Its resumable execution model is a huge improvement over having to ship megabytes of JavaScript code to the browser and then hydrating the server state on the client. Additionally, its ecosystem is quite vibrant with third-party support for Storyboard, Vitest, and others. And last but not least... there's Qwik-React which allows the reuse of existing React components inside a Qwik app.

Remix

TRIAL

Remix Focuses on web standards and modern web app UX to build better websites. As a full stack web framework in React it lets you focus on the user interface and work back through web standards to deliver a fast, slick, and resilient user experience. Remix creates a bridge between static site generated websites and dynamic server side rendered content.

We are really interested in Remix as it has the potential to be a great framework. Remix can have the requested content computed and delivered straight away per request. It can also do the compute required to get the requested content and cache it, so everything renders fast for everyone all over the world. This makes Remix a good new framework for high performant React applications.

Mitosis

ASSESS

Mitosis is a modern component library for building web applications, created by the software development company Tonic Systems. It is designed to simplify the process of building complex user interfaces by providing a set of reusable UI components that can be easily integrated into your application.

One of the key features of Mitosis is its ability to compile to other popular frontend frameworks such as React, Vue, and Angular. This means that developers can use Mitosis to build their user interfaces, and then easily compile their code to work with their preferred frontend framework. An advantage of compiling to other frameworks, instead of using web components, is that it can make full advantage of the framework. For example server side rendering is much easier when the components are native to the framework.

Mitosis is currently in the assess ring due to its status as a new technology that is still in beta. It has the potential to be a viable alternative to web components, particularly for those seeking to create a component library that can support multiple frameworks. However, it is not yet stable enough to conduct a full trial, and thus we must monitor its development and progress.

Astro

ASSESS

Astro is a new kind of static site builder for the modern web. It uses a modern server-side templating language that renders directly to HTML & CSS, eliminating heavy JavaScript automatically. Astro uses partial hydration to eliminate dead code from your website and only hydrates your essential, interactive UI. Use a frontend UI component built with React, Preact, Svelte, Vue, SolidJS, AlpineJS or Lit and Astro will automatically render it to HTML at build-time and strip away all JavaScript.

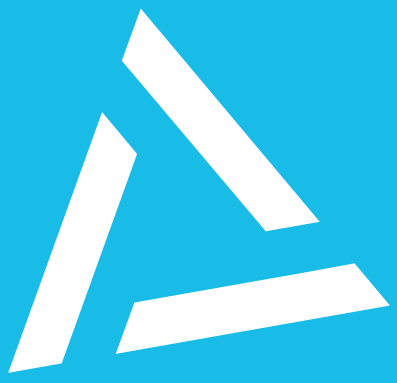
With the rise of popularity among static site generators, we are really interested in the rise of Astro and we think it is worth investigating. Astro is not tied to any particular front-end framework, allowing developers to use their favourite framework or library with ease. This makes it a popular choice for developers who prefer to work with specific frameworks like React, Vue, or Svelte.

Elm

HOLD

Elm, which exists since 2012, is a functional programming language for building front-end web applications and compiles to JavaScript. Because Elm is purely functional, it focuses on immutable values, stateless functions, type inference, et cetera. This makes Elm a reliable language that prevents state issues and related problems.

We would recommend not to use Elm for new projects. Elm does not seem to be officially end of life, but there is not much active maintenance on it. The Elm Core has its last commit in April 2021 and Elm Compiler in September 2022. One of the main reasons to use Elm is immutability but with TypeScript and, for example, Redux for state management, you can achieve the same goals. The Elm language has a relatively steep learning curve and not many developers are familiar with or willing to learn it. However, compared to Elm, many more developers know how to write TypeScript apart from the used library or framework. As we do not see many companies using Elm and there is no active maintenance on the language, we would not implement Elm with future maintenance of an application in mind.



focus on
front-end
development