divotion

tech radar

2024 volume 2

Our view on recent developments in our field

5th tech radar 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

ThoughtWorks initiated the idea of creating a tech radar. They periodically showcase their view on new trends and developments. 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 jobs to the best of their ability. When composing a radar, you facilitate a broad discussion on technology, so organizations can strike the right balance between the risks and rewards of innovation. We can help you to kick-start these discussions in your organization. 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

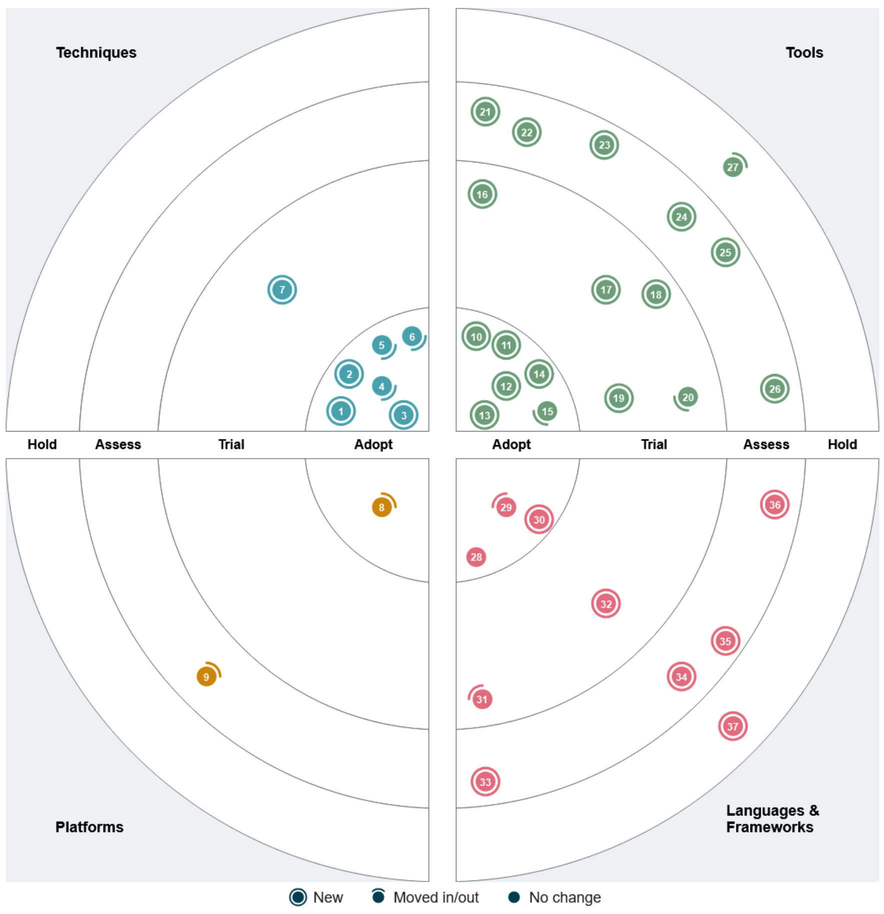
The radar consists of quadrants and rings, containing blips to indicate technologies of interest. The quadrants subdivide the different subjects into categories:

- **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.



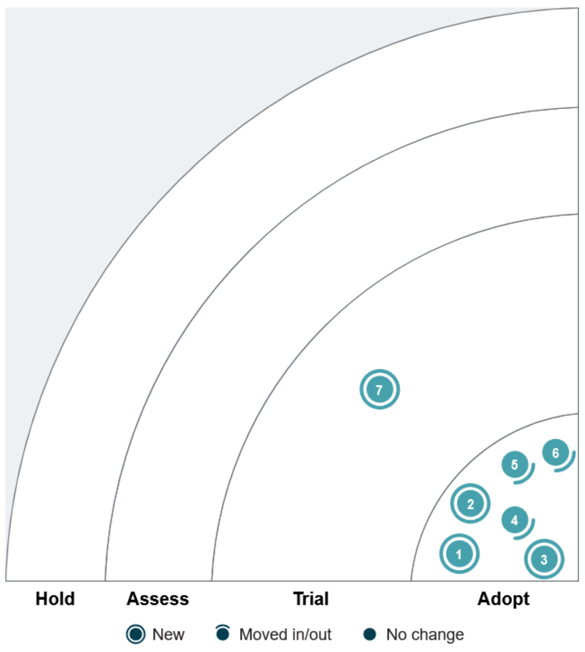
Techniques	Tools
ADOPT 1. Compound Pattern 2. Loader Pattern 3. Micro Frontends 4. Container queries 5. Server Actions 6. WebAuthn **TRIAL** 7. Native Federation **ASSESS** - **HOLD** -	**ADOPT** 10. Eslint 9 11. shadcn/ui 12. Storybook test runner 13. Supermaven 14. Turborepo Code Generation 15. Changesets **TRIAL** 16. Bruno 17. HTTPie 18. Knip 19. Syncpack 20. Guidepup **ASSESS** 21. AI Design conversion 22. Catalyst 23. Claude Sonnet 24. Ollama 25. TruffleHog 26. WebContainers **HOLD** 27. Biome
Platforms	**Languages & frameworks**
ADOPT 8. Bun **TRIAL** - **ASSESS** 9. WebTransport **HOLD** -	**ADOPT** 28. Angular Renaissance 29. Astro 30. Pinia **TRIAL** 31. HTMX 32. NgRx SignalStore **ASSESS** 33. Dioxus 34. Hono 35. Leptos 36. Refine **HOLD** 37. Lodash

Techniques

Adopt
1. Compound Pattern
2. Loader Pattern
3. Micro Frontends
4. Container queries
5. Server Actions
6. WebAuthn

Trial

7. Native Federation



1. Compound Pattern

ADOPT

The Compound Pattern is a design approach, most common in React applications, oriented towards the natural functioning of components together. They will share state and behavior in a more integral and organic way. In contrast with the rigid, isolated, atomically designed components, the Compound Pattern is all about flexibility, enabling components to interact with each other smoothly. This pattern is especially handy when building complex interactive UIs where components need to talk to each other and work together, like in forms with conditional fields or dropdowns with customizable options.

The Compound Pattern offers significant advantages in building flexible and scalable user interfaces, which is why we place it in the "adopt" lane. It can significantly simplify the general structure, reduce code duplication, and thus help to keep clean and manageable codebases thanks to its ability to centralize state management in the parent component. Such a centralized approach will also result in more intuitive APIs for managing component interactions. Moreover, reusing the child components in this pattern makes the pattern fit almost everywhere in a given application, but it provides consistency with flexibility of UI elements. Due to this compound pattern, scalability is enhanced and supported while a project grows.

2. Loader Pattern

ADOPT

Sometimes we need to go full circle before we realize we had it right all along. The loader pattern is a design pattern that has been around since the early days of the web, where you create one or more loaders associated with routes or layouts that are responsible for loading the data your application uses. Loading the data at the top, and passing it down to your application makes it easy to keep track of what data is used where. Without compromising complexity and performance.

We recommend using the loader pattern to populate your app unless your business needs require you to do otherwise. It will allow you to focus on building features instead of odd bugs as a result of caching, de-duplication, and other highly complex solutions.

3. Micro Frontends

ADOPT

Micro Frontends is an architectural style that extends the concept of microservices to the front-end. It allows teams to break down a monolithic front-end application into smaller, more manageable pieces, each representing a self-contained feature or section of the user interface. Each micro front-end can be developed, deployed, and maintained independently, often by different teams using different frameworks or technologies. This approach enables faster development cycles, better scalability, and the ability to innovate more easily without impacting the entire application. Micro Frontends can be integrated into a single UI through various techniques, such as web components or iframes, ensuring that users experience a seamless interface.

We included Micro Frontends in our tech radar because it aligns well with modern front-end development practices prioritizing flexibility and scalability. By adopting Micro Frontends, teams can work more independently, reducing dependencies and bottlenecks between teams. This architecture is particularly valuable for large-scale applications, where different teams may need to iterate quickly on their specific features without waiting for the entire front-end to be updated. It also provides a way to introduce new technologies or frameworks incrementally, making it easier to innovate and future-proof front-end projects. We believe that Micro Frontends offers a clear advantage for organizations looking to maintain agility and keep up with the rapidly changing landscape of front-end development.

4. Container queries

ADOPT

Container queries in CSS is a big step forward in making responsive web design look good on different devices. Instead of changing styles based on the whole screen size, they change styles based on the size of the part of the responsive web design they're in. This new way is more flexible and lets designers create better, more adaptable responsive web designs. You set up container rules, similar to how you use media queries. As of this year, container queries are supported in all major browsers. These queries are really helpful for making parts of responsive web design respond to different sizes, changing how designers approach making flexible layouts, and improving how parts of responsive web design work and look on different screens.

Now is a great time to try out container queries, as they are now fully supported by all major browsers. If you're struggling to create a responsive component with media queries, consider giving container queries a try. Since they're widely supported, we confidently place container queries in the "adopt" category.

5. Server Actions

ADOPT

Several front-end frameworks have incorporated a server-side runtime solution, along with the front-end that runs within the browser. This allows for developing a full-stack application. Using these frameworks, the front-end can be rendered on the server. It will serve HTML to the browser, rather than the classic Single Page App approach. By leveraging this power, data can be loaded and processed on the server. Processing data received from the browser in such a framework on the server-side is called Server Actions. Because these are tightly integrated into the front-end framework, it makes for more elegant solutions with tight integration.

We've put Server Actions into the "adopt" section because they make development easier by enabling server-side logic to be invoked directly from the client, without having to build out complex API layers. Server Actions will empower you to build more elegant solutions for typical problems that arise in an application, such as data validation, state management, and error handling. They have matured to the level of becoming safe to be used in a production environment and are supported by most major front-end frameworks. Overall, Server Actions are a strategic fit for scalable and efficient applications that align with modern trends toward serverless architectures.

6. WebAuthn

ADOPT

Web Authentication (WebAuthn) is a web standard that enables strong authentication with asymmetric cryptography, enabling authentication without user-chosen passwords. Benefits of WebAuthn over password-based authentication include protection against phishing, reduced impact of data breaches, and password (guessing) attacks. WebAuthn is available as a primary authentication option, or as a second factor in combination with a password.

WebAuthn is now fully supported in all major browsers, and we're seeing large companies like Google, Microsoft, and Apple adopting it. The user experience is not yet optimal, with differences in interface between browsers and operating systems, but this is being actively worked on. While users may not yet be familiar with WebAuthn, steps are being taken to increase awareness and adoption. We recommend adopting WebAuthn for new projects and considering it for existing ones requiring strong authentication.

7. Native Federation

TRIAL

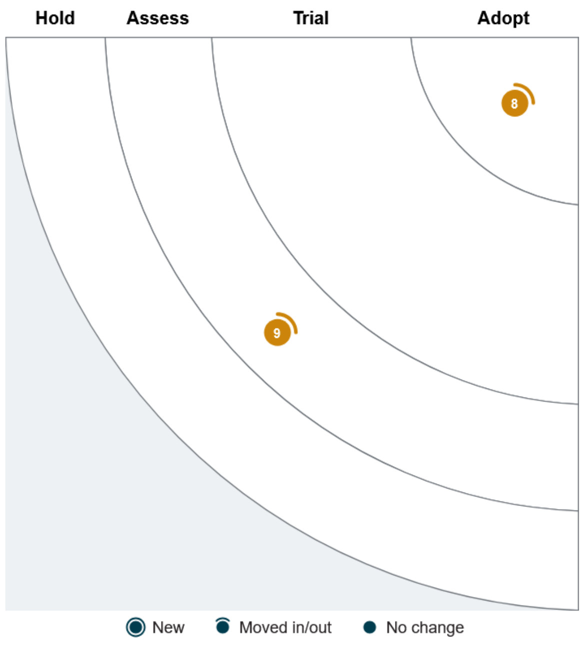
Native Federation is an approach to decoupling larger applications into smaller, independently deployable components or modules, often referred to as Micro Frontends. The key advantage of Native Federation is that it can be implemented in a framework-agnostic manner without relying on Webpack. This flexibility allows it to work seamlessly with a variety of frameworks and build strategies, making it possible to integrate multiple frameworks within a single application.

As the front-end landscape continues to evolve, we recommend considering Native Federation as a viable option. Its flexibility in allowing you to freely choose your technology stack can be an effective way to decouple your applications if you don't want to be tied to one framework. Though it's important to note that other proven technologies, such as Module Federation, might be a better starting point for projects that don't require integrating multiple frameworks. Since Native Federation is relatively new, it should be thoroughly tested before being used in production. However, we expect to see its adaptation increase in the near future.

Platforms

Adopt
8. Bun

Assess
9. WebTransport



8. Bun

ADOPT

Bun is an all-in-one toolkit for JavaScript and TypeScript apps. At its core is the Bun runtime, a fast JavaScript runtime designed as a drop-in replacement for Node.js. It dramatically reduces startup times and memory usage. In addition, Bun also implements a bundler, server framework, test runner, script runner, and NodeJS-compatible package manager.

Since version 1 was released in late 2023 Bun has proven its potential as a batteries included JavaScript and TypeScript runtime. Many projects and companies have battle-tested Bun in production, and it has shown to be able to improve productivity and reduce server costs significantly. We recommend considering adopting Bun as a whole, or only the parts that deliver value to you and your team, as it is a perfect candidate for incremental adoption.

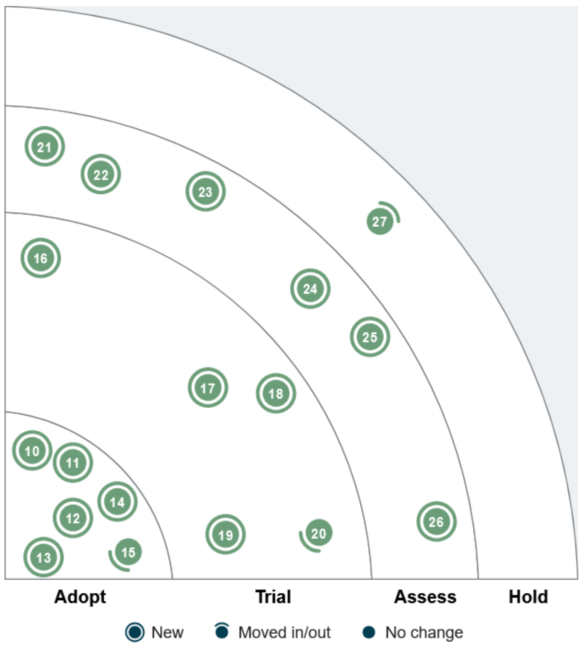
9. WebTransport

ASSESS

WebTransport is an emerging web technology that offers a new way to create bidirectional communication paths between clients and servers. It is built on the QUIC protocol, which is known for its low latency and improved security, making WebTransport a more efficient and secure option compared to WebSockets. One of the main advantages of WebTransport is its capability of managing reliable and unreliable data streams on a single connection. This flexibility allows developers to optimize their applications by choosing the appropriate level of trust for different types of data. This is especially useful in situations like real-time gaming, live streaming, or any application that is sensitive to low latency which guarantees better performance and faster connection setup than traditional TCP-based WebSockets.

WebTransport looks like a promising technique that you should definitely try out. However, since it isn't supported in all browsers, you should not use it in production just yet. Safari does not support it at all and Firefox hasn't implemented all of the specs yet. If you want to try it out, visit socket.io for a nice tutorial.

Tools



Adopt
10. Eslint 9
11. shadcn/ui
12. Storybook test runner
13. Supermaven
14. Turborepo Code Generation
15. Changesets

Trial
16. Bruno
17. HTTPie
18. Knip
19. Syncpack
20. Guidepup

Assess
21. AI Design conversion
22. Catalyst
23. Claude Sonnet
24. Ollama
25. TruffleHog
26. WebContainers

Hold
27. Biome

10. Eslint 9

ADOPT

ESLint is a linting tool for JavaScript and TypeScript that is widely used within the JavaScript community. It is a static code analysis tool that can be used to find and fix problems in your code. ESLint is highly configurable and can be customized to fit the needs of your project, with many plugins and rules available. It can be used to enforce coding standards, find bugs, and improve code quality. ESLint is a popular choice for linting JavaScript and TypeScript code and is often used in combination with Prettier to ensure consistent code formatting.

While ESLint 9 has been around for a while, adoption of this new version has been slow. The removal of the previously deprecated .eslintrc configuration format in ESLint 8 has required all plugins to update their configurations to the new format. This has caused some plugins to fall behind in supporting ESLint 9, which has slowed down the adoption of the new version. However, new tooling from ESLint and the typescript-eslint plugin, introduced a compatibility layer for ESLint 9, making it easier to transition to the new version. Not all plugins or rules will be fully compatible until they are properly updated, but we think it is worth the effort to keep up with the latest version of ESLint. If you are starting a new project, we would recommend using ESLint 9 from the start to take advantage of the latest features and improvements.

11. shadcn/ui

ADOPT

Shadcn/UI is a collection of reusable UI components that you can easily incorporate into your web projects. By providing pre-built, customizable, and accessible components, Shadcn/ui helps accelerate your development process. This approach offers flexibility, allowing you to select only the components you need, without being tied to a specific library or framework. The components are designed with accessibility in mind, making your applications more inclusive for people with disabilities.

The Shadcn/UI components are extensive, which makes it an easy way to kickstart your components. From basic elements like buttons and inputs to more complex components like forms, navigation menus, and data tables. Shadcn/ui has everything you need to build robust and engaging web applications. The collection of components is continuously expanding with new components, ensuring that you have access to the latest features and design trends. We recommend checking it out the next time you need to build a component and adopt it in your project.

12. Storybook test runner

ADOPT

Storybook test runner is an open-source tool that simplifies testing your entire design system in Storybook. Converting all Storybook stories into executable tests enables efficient testing across multiple browsers using a single generic test script. These tests run in a live browser environment and can be executed via the command line or integrated into your CI/CD pipeline. Powered by Jest and Playwright, Storybook integrates with popular testing libraries like React Testing Library.

Storybook test runner is easy to configure and use, allowing you to quickly create and execute smoke tests on all Storybook stories in your design system with minimal effort. As a first-party plugin, it is tightly integrated with Storybook, ensuring seamless compatibility and reliability. Additionally, Storybook provides valuable examples of integrating image snapshot tests and accessibility checks. We recommend adopting Storybook test runner and adding it to your testing strategy.

13. Supermaven

ADOPT

Supermaven is a tool for software development. It integrates into a code editor to provide AI-based code suggestions. A well-known competitor is Copilot, but Supermaven promises to be twice as fast. Integrations are available for VS Code, JetBrains, and Neovim.

We regularly use Supermaven in TypeScript codebases and find that it increases our development speed. Due to how fast Supermaven is, it provides suggestions even before you start typing. It also quickly learns from your input, allowing it to recognize patterns and reduce the amount you need to type. Additionally, Supermaven can assist with writing code, such as logic and tests. All in all, we find Supermaven a valuable addition to our development environment.

14. Turborepo Code Generation

ADOPT

Turborepo features a code generator as a nice tool for automating the creation of files and folders within your monorepo setup. It is almost like a template engine for your projects, in that you can define a pattern to create repetitive code, which often takes up so much time. You can even write your own generators, tailoring them to your project's special needs, which is especially useful for component libraries and Design Systems. It means that instead of creating the same boilerplate every time by hand, you just run a command and you're done!

We've put Turborepo's code generator in the "adopt" section because it simplifies the way you manage large codebases, especially in monorepos. It helps in maintaining projects consistently through generating code, reducing errors, and saving a lot of time. The flexibility to create custom generators means it can be easily adapted to your specific needs, which makes it practical and reliable. It is a mature solution, whose value has been proven, so it's definitely worth making it a core part of your development process.

15. Changesets

ADOPT

The changesets workflow is designed to help when people are making changes, all the way through to publishing. It lets contributors declare how their changes should be released, then we automate updating package versions, and changelogs, and publishing new versions of packages based on the provided information. Changesets focuses on the changes that are being made, rather than the packages that are being changed, and so it is a good fit for monorepos. With their CLI tool, you can use changesets with any package manager, and any CI.

We think that changesets is a great tool when you are working with a monorepo and for that reason we place it in adopt. Changesets help you to keep track of the changes that are being made and to make sure that you are releasing the right versions of packages. It helps you to keep your changelogs up-to-date and to make sure that you are publishing the right versions of packages.

16. Bruno

TRIAL

Bruno is an Opensource API client which you can use to test REST and GraphQL APIs. It is an alternative to the established tools like Postman and Insomnia. Bruno does not use the cloud for storage, but instead stores its collections locally in .bru files, with a kind of yaml syntax. By storing the collections locally, you can easily use it together with GIT version control and therefore collaborate with team members in the same project. It also provides a CLI and IDE plugins for easier usage.

Although Bruno is not yet feature complete (it does not support gRPC or WebSocket for example), we still think you should give it a try and use it in your next project.

17. HTTPie

TRIAL

HTTPie is a promising free, open-source, and lightweight HTTP client, that runs in your terminal, browser, and as a standalone application. For a long time, Postman had been the de facto standard for developing APIs, and applications against those APIs. Over time many developers felt Postman was becoming more and more complex and bloated, which is where alternatives like Insomnia and Nightingale came in, which are more often than not just (not so free) open source clones of Postman. HTTPie was born as an intuitive CLI with which you can send, document, and test requests, whether it be REST, GraphQL, or your own custom HTTP endpoints, and share your collections with your team. Fast forward to today, it has now wrapped its intuitive CLI into a graphical user interface.

Though HTTPie is slowly becoming known as the HTTP client that will make you smile. The GUI is still in beta, and even the CLI might not fulfill all your needs. So, make sure to give it a test run before you make any long-term decisions.

18. Knip

TRIAL

Knip is a tool designed to find and eliminate unused files, exports, and dependencies in a codebase, helping developers maintain a cleaner and more efficient project. It scans your project to find dead code and unnecessary dependencies, reducing bloat and improving performance. Knip can be integrated into CI/CD pipelines to automate these checks, ensuring ongoing codebase optimization.

While Knip is a relatively new tool its adoption is growing quickly, particularly among developers maintaining a large codebase. It is a great tool to streamline both JavaScript and TypeScript projects and improving the overall code quality and maintainability of the project. Currently, Knip can be used to find unused files, exports, and dependencies. The auto-fix feature aims to automate the process of updating or removing files too. This is still in the experimental phase but would be a great addition when implemented. With its ability to run both locally and in the CI/CD environment, there is always a way to keep your project clean.

19. Syncpack

TRIAL

Syncpack is a convenient open-source command-line utility that makes synchronizing the version of packages in large JavaScript monorepos easier to do, which makes dependency management much easier. It will keep everything up-to-date and consistent by automatically bumping your dependencies to a version you specify. Therefore, it reduces version conflicts and incompatibility issues by providing a single, unified way of handling package versions. Syncpack is compatible with a variety of package managers and version control systems right out of the box, making it simple to incorporate into your projects.

We've put Syncpack into the "trial" section because it is less widely adopted or as mature as tools like Renovatebot and Dependabot are today. Compared to these tools, Syncpack prioritizes package consistency above an automated update process. This narrower focus might need some further evaluation to see whether it serves your full needs, especially if you're already using tools that handle dependency updates effectively. Overall, Syncpack is worth experimenting with, but it might not be a complete replacement for other available solutions just yet.

20. Guidepup

TRIAL

Guidepup is a screen reader driver for test automation. A screen reader is an assistive technology that translates digital text and content into synthesized speech or braille output, enabling individuals with visual impairments to access and navigate web content. With Guidepup you can automate screen reader test workflows the same you would for mouse or keyboard-based scenarios. It can be used to create (unit)tests that can check for expected screen reader behavior. This way it is also a self-describing way of documenting the expected screen reader behavior of a component. Guidepup can be used with the screen readers VoiceOver (macOS) or NVDA (Windows). You can also use the "Guidepup Virtual Screen Reader" to be operating system agnostic.

Nowadays, it's becoming more and more important to make the Web more accessible for screen readers. Using Guidepup, you can easily ensure, through automated tests, that your website/app works as expected with an actual screen reader or a virtual screen reader. This can be useful for integration in your CI/CD pipeline. Due to its unique capabilities and the improved importance of accessibility, we've rated Guidepup with Trial. We do not see Guidepup as a replacement for Axe or Pa11y, but as an extra tool to help better convey what the expected behavior of a component or page is.

21. AI Design conversion

ASSESS

Tooling that converts designs into code (and vice versa) is an evolving area in the field of design-to-development workflows. These tools aim to bridge the gap between designers and developers by automating parts of the conversion process. Examples of design to code are Anima, Teleporthq, Locofy, BuilderIO and BuilderX

There are already a few tools capable of converting designs into code, and we believe they have the potential to speed up the initial workflow from design to development. However, we've placed AI design conversion tool in the Assess category because it's still early days for these kinds of tools, and they don't provide a full and workable solution. At this moment we need to test the tools more thoroughly to see if organizations can already benefit from AI design conversion.

22. Catalyst

ASSESS

Catalyst is a React UI kit made by the same people who built Tailwind CSS, that offers pre-built components. It has a clean, modern design that uses Tailwind's utility-first approach and is highly customizable. By copying the components into your project, you can quickly build a modern, responsive UI, without having to install it as a dependency. Catalyst does require a one-time payment to access the components.

We've put Catalyst into the "assess" section because it's a relatively new project. However, it shows real potential as a modern solution for the design and development of UIs. If you're looking for a customizable UI kit that integrates with Tailwind CSS and Headless UI, you should look into Catalyst. Though, keep in mind that Catalyst is not free, so it will be up to you to decide whether it's worth what it offers or consider free alternatives like Shadcn.

23. Claude Sonnet

ASSESS

Claude Sonnet, together with Claude 3.5, makes web development faster by automatically generating code, taking part in the design of UI/UX, and debugging with code reviews. It can also generate and manage content, create interactive documentation, and intelligent chatbots for real user support in real-time. Besides, the feature "Artifacts" allows teams to collaborate in developing generated assets, making web development faster, more efficient, and collaborative.

This AI-powered assistant secures its spot in the "Assess" category because it represents the state-of-the-art in AI's integration into developer workflows. When there are large gains in productivity promised, consideration needs to be given to the effect on skill development and team dynamics. Developers should explore investments in Claude Sonnet for how it can help them enhance their capabilities, particularly around rapid prototyping, documentation, and code review. It lets them play with the technology to visualize some kind of future where AI and human developers work in harmony.

24. Ollama

ASSESS

Ollama is an AI platform designed to provide large language models (LLMs) for different tasks, much like OpenAI's GPT models. It allows developers or users to interact with language models and with the extension CodeGPT in VSCode.

Many organizations are cautious about sharing code with external AI platforms, and for good reasons. AI models, especially large language models (LLMs), are relatively new. Many platforms -especially those on free tiers or default settings- may learn from the data users provide. This raises concerns about inadvertently exposing private code, with little control over how or if it can be removed. A safer approach is to use AI tools locally. While response times on older devices (like Intel MacBook Pros) are currently slow, we expect this to improve as the technology advances. Tools like Ollama are already proving to be promising solutions for enhancing development speed, and will likely become even more valuable in the near future.

25. TruffleHog

ASSESS

TruffleHog is a cyber security tool designed to detect and prevent leaking of secrets in code repositories, infrastructure, and cloud environments. Its primary function is to scan the codebase for sensitive information, such as API keys, credentials, and other secrets, that may have been inadvertently committed or exposed. TruffleHog integrates seamlessly with popular version control platforms like GitHub, GitLab, and Bitbucket, allowing developers to safeguard their code from potential vulnerabilities. For front-end teams, the tool plays a critical role in ensuring that sensitive data, such as API keys used in client-side code, are not unintentionally exposed to the public. Additionally, the tool offers real-time monitoring and can scale with cloud environments, making it suitable for organizations of varying sizes and complexities.

We placed TruffleHog on our tech radar because it addresses a key security challenge, especially in front-end development where API keys and other credentials are often embedded in code. Given the increasing reliance on API-driven front-ends, it's crucial to prevent the accidental exposure of sensitive data in public-facing applications. TruffleHog helps mitigate this risk by scanning for these secrets before they are committed, reducing the potential for front-end security breaches. Its integration with major version control platforms ensures that teams can implement it with minimal disruption to their workflow. For companies focused on front-end development, this tool adds an essential layer of security, making it highly relevant to our ongoing efforts to secure our projects.

26. WebContainers

ASSESS

WebContainers provide the ability to run Node.js environments directly in the browser. Developed by StackBlitz, WebContainers are a way to run Node.js using WebAssembly and browser APIs to provide Node.js functionality in the browser, including a virtual file system and npm integration without server-side infrastructure requirements. The typical use cases for WebContainers are online development environments, coding education platforms, and browser-based Node.js application testing.

This innovative technology earns its place in our "Assess" category, as it has the potential to rewrite the rules of web development environments. They will reduce the barrier to entry for new developers and make collaboration across teams easier by not requiring local setups. However, this requires thoughtful consideration by developers concerning the implications for existing workflows and limitations for large projects. A test drive of WebContainers will let developers experience how browser-based Node.js might supercharge their productivity and collaboration.

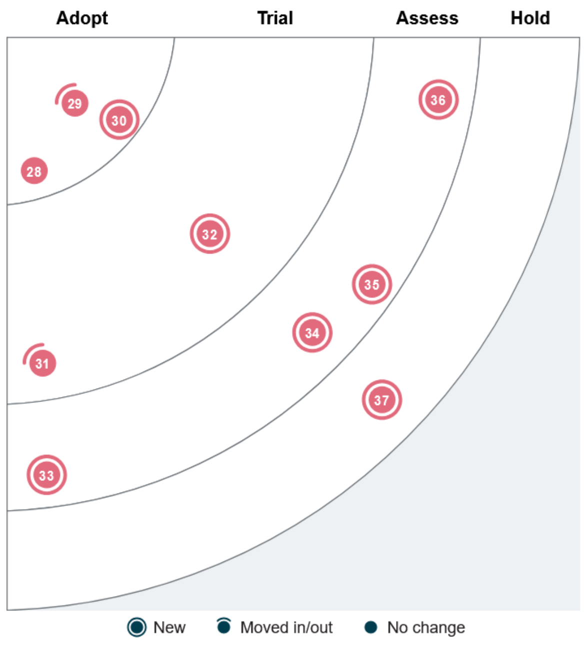
27. Biome

HOLD

Biome, "Toolchain of the Web", promises to be the all-in-one solution for web development. Their Fast Formatter which has 97% compatibility with Prettier and Performant Linter with 268 rules is already available, but the rest of the toolchain is still in development. Biome is built in Rust and aims to be faster and more reliable than the current JavaScript tooling. With their default ruleset, they claim Biome requires no configuration.

Including Rome, Biome’s upstream repository that is no longer in development, the project has been in the works for over 4 years. We’re moving Biome to Hold because we see a lot of potential, but the project is still not anywhere near completion. Lacking language support and the absence of the promised compiler, bundler, and testing solution are the main reasons for this decision. We will keep an eye on the project and update our recommendation when the toolchain is complete.

Languages & frameworks



Adopt
28. Angular Renaissance
29. Astro
30. Pinia

Trial
31. HTMX
32. NgRx SignalStore

Assess
33. Dioxus
34. Hono
35. Leptos
36. Refine

Hold
37. Lodash

28. Angular Renaissance

ADOPT

Since the release of Angular version 17, a lot has changed. The Angular team at Google has introduced significant improvements and modernizations: SSR/SSG, a new control flow for templates, and the previously released standalone components, signals, and the inject function. They have called this their "renaissance" and continue to expand on it with new major releases every six months. In Angular 18, they implemented new control flow syntax, expanded their "signals" API, and are experimenting with zoneless change detection. In recent months, the focus has been on applying these new techniques.

After months of hands-on experience, we still strongly recommend embracing the new Angular features and taking full advantage of them. The new control flow syntax reduces boilerplate code and divides the HTML template into easily recognizable code blocks. Signals help us approach reactive programming in a simpler and more understandable way, making them a good alternative to traditional solutions. We see signals gaining a lot more traction, but it’s still relatively new. Developers should assess its long-term before choosing signals over more established reactive programming techniques. Standalone components eliminate the need for Angular modules, making Angular components easily interchangeable. However, it should be noted that Angular modules can still play a big role in larger-scale applications and can be used in combination with standalone components. Additionally, the inject function simplifies dependency injection, making the code cleaner and more maintainable. When applying these aforementioned improvements, building an Angular application feels intuitive, modern, and fast!

29. Astro

ADOPT

Astro is a content-focused application framework with dead simple APIs that enable all kinds of applications and architectures to be built on top of. No matter which UI framework you use, no matter the scale of your team and application Astro can cater to most of your needs. Either generate a static site, or render your application on the server, Astro will render the HTML for your application, and strip all JavaScript away, hydrating only the parts that require JavaScript to work.

After a strong surge in adoption over the past two years, Astro is now the only framework with a rising line in developer satisfaction in the State of JS survey of 2023. The team behind Astro is actively developing new features to improve the quality of life for developers even more, such as seamless integration with databases and infrastructure. If you are building a new project, or looking for another application framework, we recommend considering Astro as a candidate.

30. Pinia

ADOPT

Released together with Vue3 as the recommended state management library for the setup API. The reactivity API introduced with Vue3 is a very strong foundation for state management, but still requires some expertise in different areas to safely and reliably implement a shared state in your Vue apps. Pinia leverages all the strengths, such as the flexibility and compose-ability of the reactivity API, and provides a tiny framework so you don't have to worry about design patterns, and the dangers of SSR.

We recommend adopting Pinia in your stateful Vue apps, if you haven't already, to create a zero-compromise foundation without having to worry about a lot of the complexity involved.

31. HTMX

TRIAL

HTMX is a powerful library that enables you to add dynamic behavior to your web applications without relying heavily on JavaScript. By using HTMX, you can enhance user interactions by seamlessly updating parts of a webpage in response to user actions like clicks, form submissions, or scrolling. What sets HTMX apart is its simplicity and ease of integration; you don't need to write complex JavaScript code or manage a large JavaScript framework. Instead, you can use HTML attributes to define the behavior, allowing your server to handle most of the logic.

Single Page Applications often communicate with the backend using JSON, but with HTMX this is in most cases just HTML (with possibly more HTMX-enhanced parts). Using HTMX aligns well with traditional server-rendered applications, making it easier to maintain and scale your project without introducing the complexities of modern front-end frameworks like React or Angular. HTMX seems very promising and certainly a library to consider in your next or current project, especially if you see the benefits of Hypermedia and HATEOAS.

32. NgRx SignalStore

TRIAL

NGRX Signal Store is a promising evolution in state management for Angular applications. Building on the robust foundation of the NGRX ecosystem, which has long been the go-to for managing complex state in Angular apps, the Signal Store introduces reactive signals-a new way to think about state and reactivity. This shift addresses some of the pain points of traditional state management approaches, particularly around performance and developer experience.

The Signal Store uses Angular's new signal API. This is part of a move to more reactive and intuitive handling of state changes. Handling state with the Signal Store becomes significantly easier than the previously used RxJS methods. That being said, RxJS can still be used alongside Signal Store for more complex operations. The use of Signal Store significantly reduces boilerplate code and makes the reactive flow of data more straightforward to follow and debug. We recommend running a trial with the NGRX Signal Store in the context of your projects. Start with smaller, non-critical features or applications to get a feel for the API and its benefits. As the technology matures and more community resources become available, it could become a cornerstone of your Angular state management strategy.

33. Dioxus

ASSESS

Dioxus is a modern, Rust-based framework for building highly performant and cross-platform user interfaces (UIs). Leveraging the power and safety of Rust, Dioxus allows developers to create fast and reliable UIs with a focus on simplicity and productivity. It uses a reactive programming model similar to React, where UI components automatically update when the underlying data changes. However, Dioxus goes beyond the web, offering the capability to build applications for various platforms, including desktop, mobile, and web, all from a single codebase. This cross-platform versatility, combined with Rust's memory safety guarantees, makes Dioxus an excellent choice for developers looking to build secure and efficient applications across different environments.

If you are a fan of the Rust programming language and are interested in cross-platform app development, you should definitely check out Dioxus. It can be an alternative to frameworks like React Native and Flutter. Dioxus isn't at version 1.0 yet, so we advise to not use it in production just yet. Still, it looks like a promising framework which can become more relevant in the future.

34. Hono

ASSESS

Hono is a fast, minimalist JavaScript framework for building web applications and APIs. It is built on modern web standards and supports both Node.js and Edge runtimes, such as Cloudflare Workers. The framework has a small footprint and a simple API, with support for routing, middleware, and plugins. Hono is designed for speed and scalability, minimizing the overhead found in more complex frameworks.

We believe that Hono could be an interesting option for setting up lightweight applications with an emphasis on speed, especially for projects targeting edge computing platforms. However, the framework offers fewer features than larger frameworks. Since it is built on web standards, there should still be plenty of options for extension.

35. Leptos

ASSESS

Leptos is a full-stack web framework for building reactive applications in Rust. It’s component-based with automatic UI updates, has server-side and client-side rendering, and compiles to WebAssembly. Leptos provides server functions for communication from client to server and integrates well with Rust backend frameworks. This framework focuses mainly on performance, type safety, and developer experience.

This Rust-based framework lands in the "Assess" category because it challenges conventional wisdom for web development frameworks. Baking together front-end and backend development in Rust represents a forceful proposition, but the requirement for a new language creates a significant pivot. Developers should investigate the ways that performance advantages and full-stack type safety in Leptos might improve development efficiency and application robustness. This is an opportunity to adopt a different way of web development that truly leverages the power of Rust.

36. Refine

ASSESS

Refine is an open-source React-based framework designed to facilitate the development of CRUD (Create, Read, Update, Delete) applications. With a modular and flexible architecture, it allows developers to efficiently integrate with various backend services and third-party tools, such as REST APIs and GraphQL. Its straightforward setup, coupled with comprehensive documentation, makes it a suitable choice for creating scalable and maintainable web applications.

What separates Refine from others is that it provides out-of-the-box components and configurations, especially for CRUD applications and integrations. This sort of usability does reduce development time by a fair fraction, saving teams hours while building and deploying data-driven applications. It also has a huge number of examples, which helps a developer understand and implement common use cases more easily. Although Refine is relatively new, its potential for simplifying development for data driven applications makes it a very strong competitor against other frameworks. Because Refine is relatively new, we have graded Refine as "assess" for now, and we will continue to watch how it evolves and is adopted within the community.

37. Lodash

HOLD

Lodash is a JavaScript utility library that provides many functions for simplifying common programming tasks, particularly with arrays, objects, and strings. The library has a long history intertwined with Underscore.js, the first utility library of its kind. Lodash was created as a fork of Underscore.js to improve performance and add more features. Lodash is still actively maintained and is widely used in the JavaScript community.

Back when Lodash was created, JavaScript was missing many of the utility functions that are now built into the language. However, JavaScript has come a long way since then, and many of the functions provided by Lodash are now available natively in the language. For example, JavaScript now has built-in functions like `Array.prototype.map`, `Array.prototype.filter`, and `Array.prototype.reduce` that can replace many of the functions provided by Lodash. Additionally, modern JavaScript features like arrow functions and destructuring can make code more concise and readable, reducing the need for utility libraries like Lodash. We're moving Lodash to Hold because it's no longer necessary for many projects, and its functions can often be replaced with native JavaScript features.

focus on
front-end
development