divotion

tech radar

2025

Our view on recent developments in our field

6th 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 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

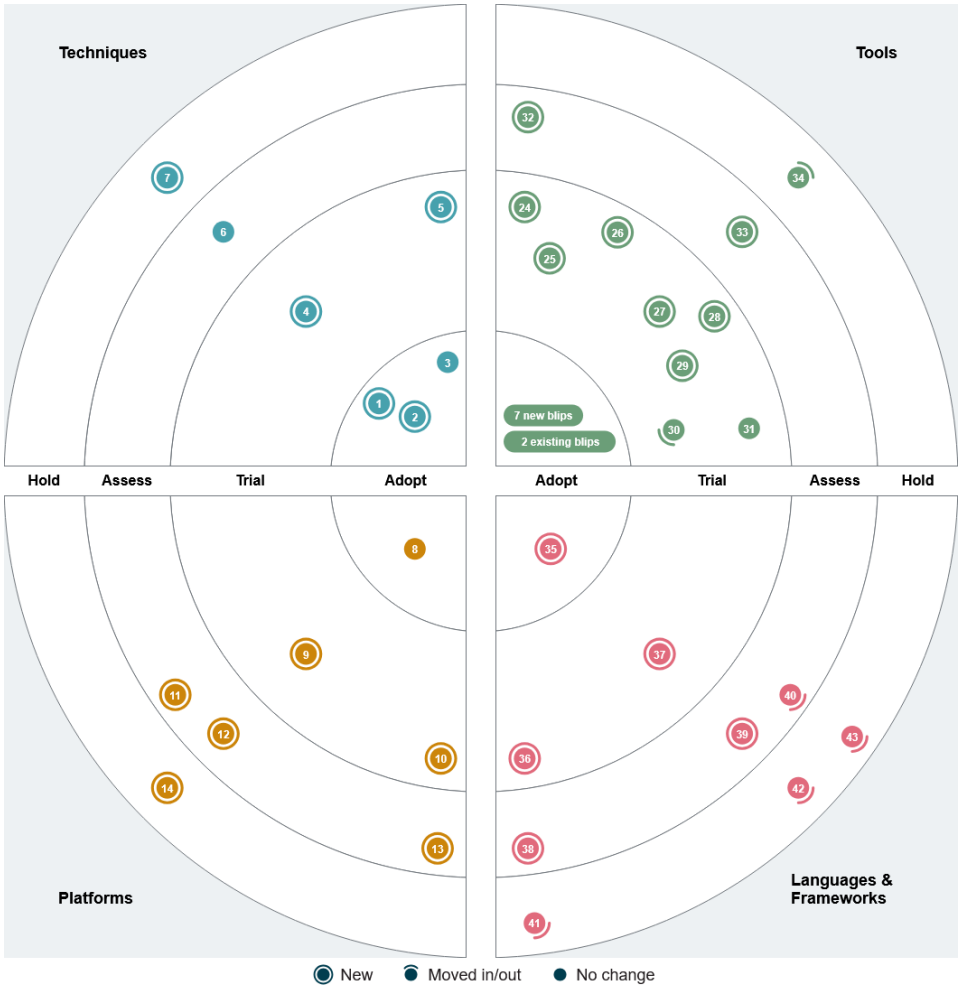
The radar consists of quadrants and rings, containing blips to indicate technologies of interest. The quadrants subdivide the different subjects into categories:

- **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:

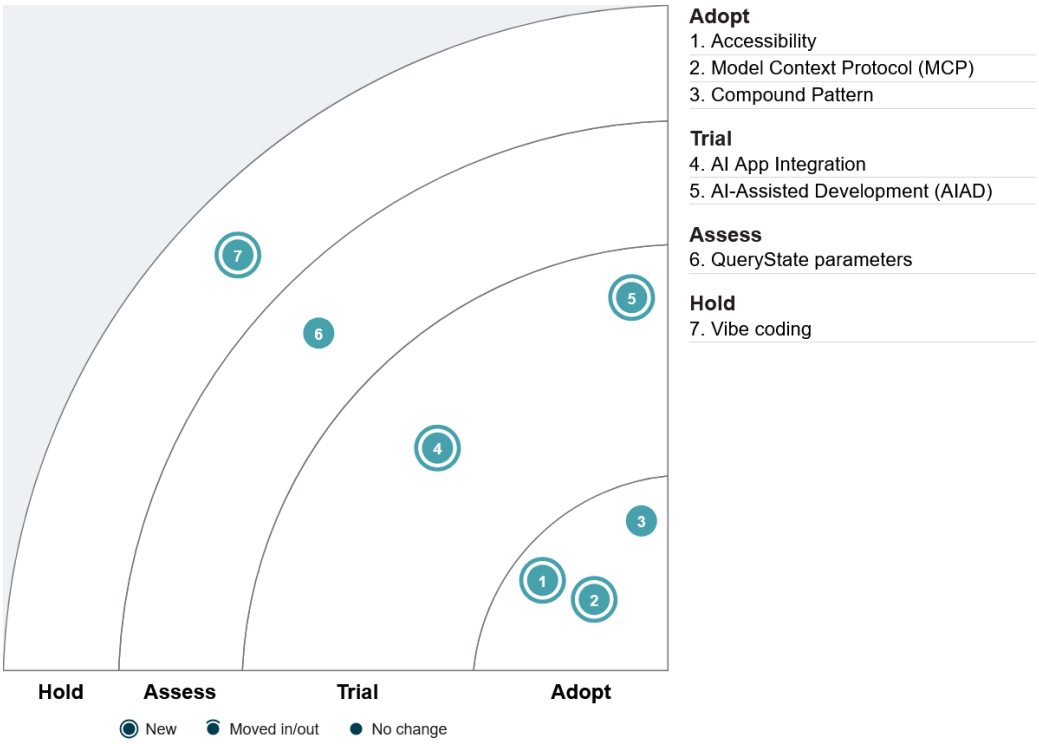
- **Adopt**: We recommend using this technology, wherever it fits the requirements.
- **Trial**: We advise to gain experience with this technology, wherever a project allows for a certain degree of risk.
- **Assess**: Interesting topic to learn more about and assess its future impact; yet too early to 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. Accessibility 2. Model Context Protocol (MCP) 3. Compound Pattern **TRIAL** 4. AI App Integration 5. AI-Assisted Development (AIAD) **ASSESS** 6. QueryState parameters **HOLD** 7. Vibe coding	**ADOPT** 15. Angular DevTools 16. Cursor 17. Dependency cruiser 18. JSR 19. schema-dts 20. Visily 21. WCAG Color contrast checker 22. HTTPie 23. Ollama **TRIAL** 24. ast-grep 25. Detective 26. Gemini Nano 27. GitButler 28. nuqs 29. Warp 30. Claude Code 31. Knip **ASSESS** 32. Builder.io 33. Stitch **HOLD** 34. Supermaven
Platforms	**Languages & frameworks**
ADOPT 8. Progressive Web Apps **TRIAL** 9. Oxc 10. Tauri **ASSESS** 11. Bolt.New 12. Coolify 13. Temporal API **HOLD** 14. iOS WebKit only rendering engine	**ADOPT** 35. React Router v7 **TRIAL** 36. TensorFlow.js 37. Zoneless **ASSESS** 38. Remix v3 39. TanStack Start 40. NgRx SignalStore **HOLD** 41. Angular Material 42. NextJs 43. Leptos

Techniques



Adopt
1. Accessibility
2. Model Context Protocol (MCP)
3. Compound Pattern

Trial
4. AI App Integration
5. AI-Assisted Development (AIAD)

Assess
6. QueryState parameters

Hold
7. Vibe coding

1. Accessibility

ADOPT

Web accessibility ensures that all users, including those with disabilities, can perceive, understand, and navigate the web. This is achieved by designing websites and digital tools to be compatible with assistive technologies, such as screen readers for people with visual impairments, and to be fully operable using a keyboard alone, which is essential for many with motor disabilities. The European Accessibility Act (EAA) reinforces these principles by mandating that a wide range of private-sector digital products and services, including e-commerce, banking, and e-books, must meet specific accessibility requirements. While the EAA focuses on the private sector, government digital products are covered by a separate directive, the EU Web Accessibility Directive, which already requires public-sector websites and mobile apps to be accessible.

Apart from legal requirements, we hope to see more websites and mobile apps become accessible by design, fostering a truly inclusive digital world for everyone. To achieve this, it's crucial that developers gain a deeper understanding of accessibility principles and the tools available to them. By integrating accessibility from the start of the development process, we can build a more equitable and functional web.

2. Model Context Protocol (MCP)

ADOPT

Model Context Protocol (MCP) is a standardized protocol for enabling AI agents to connect with external resources such as databases, APIs, file systems, and custom tools. This provides agents with richer context and awareness of a system's capabilities and available resources. MCP can be used to enhance the development workflow or to enable agent integration in the application that you are building. It allows agents to make more informed decisions and provide more relevant outputs based on the available context.

MCP is already well-supported across the ecosystem. There are 9 SDKs, over 1,000 servers, and more than 70 compatible clients available. While not all agents support every category of functionality, this should not be a blocker to adoption. The protocol is mature and practical for use today and opens the possibilities for better AI agent integration in your applications.

3. 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 Component Pattern earns a spot in our Adopt ring for its ability to create flexible, consistent UIs by centralizing state and simplifying component interactions. It reduces duplication, improves maintainability, and is especially effective in shared components and design systems. Yet despite these strengths, it remains underused. Developers often fall back on prop drilling or one-off setups because they seem quicker or easier to reason about in the moment. But compound components offer a more robust and scalable foundation, especially as apps grow. Due to this, we encourage teams to embrace the Compound Pattern for its long-term benefits in building maintainable and adaptable user interfaces.

4. AI App Integration

TRIAL

Integrating AI into web applications can be done in two primary ways: through external APIs or by running models locally. API-based solutions offer high performance and easy access to powerful models, but involve external dependencies and usage-based costs. Running models locally provides better control over data and avoids recurring fees, although this comes at the cost of higher resource usage and potentially more complex deployment. Both methods enable features like chatbots, smart search, dynamic content generation, interpretation of user input, or user assistance, depending on the model's capabilities and integration level.

We placed AI Integration in the Trial ring because this technology is becoming too impactful to ignore, but demands thoughtful evaluation. It's worth experimenting with both API and local model integration in real use cases, especially where automation or smarter interfaces can offer immediate value. Considerations like performance, privacy, and total cost of ownership will weigh differently for each organization. Some use cases may not yet justify the added complexity or cost. In our experience, even small-scale trials often open the door to valuable improvements and insights. It's time to explore integrating AI into your web application.

5. AI-Assisted Development (AIAD)

TRIAL

AI-Assisted Development refers to the use of AI tools, such as chats, intelligent autocompletion, or full-fledged coding agents, to accelerate and enhance software creation. These systems leverage large language models and domain-specific AI to interpret natural language requirements, support with research, suggest code snippets, flag potential issues, and even generate documentation. The aim is to boost developer productivity, reduce repetitive work, and improve overall code quality, while still keeping humans firmly in control of design and decision-making.

On paper, AI-Assisted Development seems to be the holy grail for businesses and developers alike, bringing the promise of higher productivity and quality. In reality, however, it is not always able to deliver on these promises; there are many concerns about topics such as security, privacy, maintainability, and even productivity. Depending on who is using these tools and what they are used for, they might not always deliver the value we need. When you need to do research, write documentation, or need a "rubber duck", these tools are incredible and can save loads of time. For tasks where the AI contributes to an end-product, or IP, it is recommended to move with caution, as this is where active human oversight is required. As a consequence, using AI for these tasks can often take more time and effort. For these reasons, we recommend using LLM-based tools for what they are intended for: interpreting natural language. At this stage, careful evaluation, limited-scope experiments, and clear human oversight are essential before wider adoption. Try it out, and make sure to be critical of the values it brings: does it actually increase your productivity? Is the quality of the end-product really getting better?

6. QueryState parameters

ASSESS

The use of Query State Parameters is an underappreciated technique that offers significant benefits for usability and persistence. Modern frameworks like React, Angular, Svelte, or Vue make it easy to manage state entirely in memory, especially when paired with state management tools (Redux, Zustand, Pinia, etc.).

Storing state properties and values as search parameters makes application state — such as search queries, filters, pagination, open or selected UI elements, etc. — shareable, bookmarkable, and reload-safe. When state is encoded in the URL (e.g., ?q=laptop&brand=dell&page=2&tab=faq), a user can copy and share the link, and anyone opening it will see the exact same view. This also ensures the state survives a full page refresh without needing extra persistence mechanisms like localStorage. For search-heavy applications, it improves user experience by allowing deep linking to specific content and even enabling search engines to index relevant filtered pages for SEO.

Using this technique serves as a combination of bookmarking and deep linking, and can serve as a key feature in your application. A known challenge is encoding the state in a way that is pretty, readable, and communicates the intent to the user.

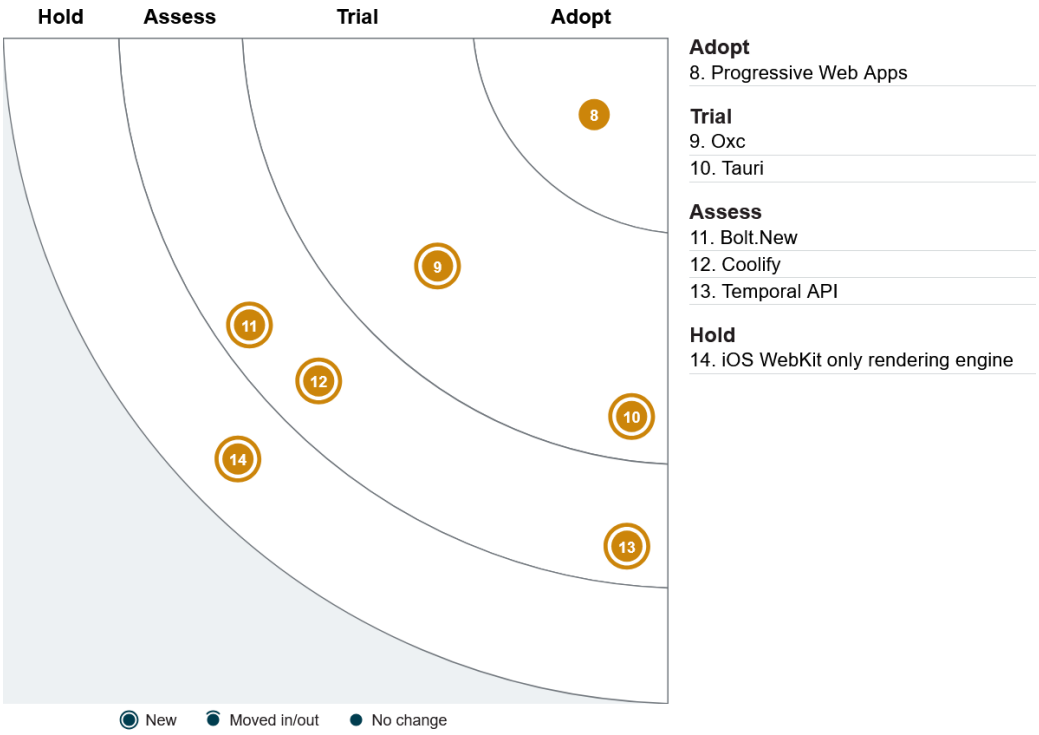
7. Vibe coding

HOLD

Vibe coding refers to the use of Generative AI to generate code based on natural language prompts. Effectively allowing users to "build" applications without deep technical knowledge. It emphasizes communicating desired outcomes to AI, rather than focusing on the technical details of coding. Think of it as describing what you want your app to do, and the AI handles the implementation.

We're putting "Vibe coding" on Hold because it erodes a non-negotiable principle: developer ownership. For our systems to be reliable and maintainable, engineers must take responsibility for the code they ship. Relying on AI to generate complex logic creates a dangerous disconnect, treating the codebase as a "black box" rather than a craft. This approach often proves counterproductive, trapping teams in lengthy cycles of reviewing and debugging AI output, ultimately costing more time than it saves.

Platforms



8. 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.

Progressive Web Apps were already in Adopt in 2023 on our radar, because iOS 16.4 added support for push notifications. This was a major step forward for the technology. However, the journey hasn't been smooth. In early 2024, Apple's beta for iOS 17.4 temporarily removed PWA "Home Screen" functionality in the EU to comply with the Digital Markets Act (DMA). This move sparked a strong backlash from developers and the European Commission. Due to this pressure, Apple quickly reversed its decision and reinstated PWA support in a subsequent iOS 17.4 update in March 2024. This incident underscores that PWAs are here to stay and that the European Commission recognizes their importance as an alternative to native apps.

9. Oxc

TRIAL

Oxc is a high-performance and unified JavaScript / TypeScript toolchain designed to complement Vite and Vitest. It provides components that can be used as either a standalone tool or as a library for building other tools. Its linter (Oxlint) is production-ready and runs 50-100 times faster than ESLint with over 520 built-in rules. And its bundler (Rolldown), currently still in development, will replace both ESBuild and Rollup usage in Vite.

We currently rate Oxc with Trail because it's still under active development. For example, while the Oxlint linter is available, it doesn't yet support TypeScript's "type-aware rules" like ESLint does. To become a truly unified JavaScript/TypeScript toolchain, Oxc also needs a code formatter (like Prettier) and a minifier to complement its Rolldown bundler.

10. Tauri

TRIAL

Tauri is an open-source framework for building cross-platform applications with a Rust backend and a web-based frontend. It's often seen as a more lightweight alternative to Electron because of its key architectural difference: instead of bundling a full web browser with the application, Tauri leverages the operating system's native webview. This approach results in significantly smaller application sizes and faster performance. Tauri provides an extensive API that allows you to interact with the platform's native features, reducing the need to write complex Rust code. This makes it a great tool for developers who want to build cross-platform applications using web technologies without compromising on performance or security.

Tauri v2 further expands its capabilities, making it a compelling alternative to frameworks like Electron for desktop development and Capacitor or Flutter for mobile. Its exceptional flexibility in supporting any frontend framework and targeting multiple platforms makes it an ideal choice for building a wide range of cross-platform applications.

11. Bolt.New

ASSESS

Bolt.new is a web-based AI code platform, created by StackBlitz, that aims to speed up JavaScript-based software development by using generative AI for code suggestions, project scaffolding, and quick prototyping. One of the best features is the ability to build a complete application just by entering a single prompt. This makes you go from an idea to working code in just minutes. This makes it appealing for teams that want to try out new technologies or prototype ideas rapidly. Or if you already have a GitHub repository, you can import it and continue from there. Bolt.new is compatible with well-known frameworks, so you can develop in the framework you prefer. Bolt.diy, the open source and localized version of Bolt.new, allows you to host the AI-driven environment on your own infrastructure. This helps mitigate data privacy and compliance issues.

We have put Bolt.new in the Assess ring since it's a great platform for quick prototyping and quick proof of concepts, enabling you to concentrate on your ideas without the project setup overhead. The browser-based environment lets you get started easily, making it perfect for experimenting with new technologies or proving concepts quickly. We suggest Bolt.new particularly for hackathons, innovation sprints, and early-stage experiments where speed and adaptability outweigh long-term maintainability. Although we are enthusiastic about the platform, questions remain about code quality, maintainability, and integration into existing production deployment workflows. As the platform continues to mature and additional users report their experiences, we will re-evaluate its position on the TechRadar.

12. Coolify

ASSESS

Coolify is an open-source, self-hostable platform that serves as an alternative to PaaS solutions like Heroku, Netlify, and Vercel, and can be deployed to any hosting provider. It enables developers to deploy, manage, and scale containerised applications, such as Docker-based web apps, databases, and services, via an intuitive GUI and automated workflows. Installation is straightforward, after which users gain access to a dashboard through which they can connect Git repositories, handle SSL, set up domains, and manage deployments with ease.

Coolify can provide your teams with a great developer experience, allowing them to ship fast and deploy anything without restrictions, while at the same time keeping full control over your infrastructure and the bills that come with it. Unfortunately, its value for enterprise can be limited due to it only having experimental support for Docker Swarm, and no support for Kubernetes at this time. With active development and strong community interest, it's a promising candidate for teams to assess and evaluate for fit before trying it out or adopting it.

13. Temporal API

ASSESS

Working with JavaScript's `Date` object has long been a source of frustration due to its mutability, clumsy date arithmetic, and wildly unpredictable parsing. This fact is hilariously demonstrated by the jsdate.wtf website. The Temporal API arrives as a modern replacement, solving these core problems with an intuitive and predictable design. It features dedicated objects for specific concepts, makes time zone handling explicit and unambiguous, and is completely immutable. To complete the picture, the Temporal API pairs perfectly with the native `Intl` API, using `Intl.DateTimeFormat` and `Intl.DurationFormat` for powerful, locale-aware formatting.

While the Temporal API is undoubtedly the future, its time hasn't fully arrived, which is why we rate it with Assess. The primary hurdle is its limited browser support; currently, only Firefox offers native implementation. The alternative is shipping a sizable 19.8 KB polyfill, which is a significant cost for production bundles. Our official stance is to wait for baseline support across major browsers before adoption, allowing us to leverage this new API without the polyfill penalty.

14. iOS WebKit only rendering engine

HOLD

For a long time, Apple required all iOS and iPadOS web browsers to use its WebKit rendering engine, the same one that powers Safari. While Apple cited security and privacy, this policy effectively made browsers like Chrome and Firefox little more than "skins," as they couldn't use their own engines. This stifled competition and innovation, preventing third-party browsers from offering unique features or performance boosts. Now, due to regulatory pressure from laws like the EU's Digital Markets Act (DMA), Apple must allow alternative browser engines on iOS in certain regions, which could lead to more diverse and powerful browser experiences.

After temporarily removing Progressive Web Apps (PWAs) support in the EU due to the DMA, Apple faced significant backlash and reinstated the feature. However, a major limitation remains: even with other browser engines now permitted, PWAs are still confined to WebKit. This prevents developers from leveraging features and optimizations from engines like Chromium (Chrome) or Gecko (Firefox). Although the DMA doesn't explicitly mention PWAs, Apple's policy of locking them to WebKit appears to go against the law's core principles of competition and user choice. We hope Apple will stop being maliciously compliant with the DMA and begin to fully support other rendering engines on its operating systems.

Tools

Adopt
15. Angular DevTools
16. Cursor
17. Dependency cruiser
18. JSR
19. schema-dts
20. Visily
21. WCAG Color contrast checker
22. HTTPie
23. Ollama

Trial
24. ast-grep
25. Detective
26. Gemini Nano
27. GitButler
28. nuqs
29. Warp
30. Claude Code
31. Knip

Assess
32. Builder.io
33. Stitch

Hold
34. Supermaven

15. Angular DevTools

ADOPT

The Chrome DevTools plugin for Angular recently received an update that adds full support for Signals, Angular's new reactive primitive designed to simplify state management. Where developers previously had to dig through components and subscriptions to understand state changes, the plugin now provides a clearer and more intuitive view of signal values and how they relate to each other. This helps teams trace reactive state across the application, making debugging and optimization much more straightforward.

In our experience, this update reduces guesswork and increases clarity. It is especially useful for teams that are starting to adopt Signals in more complex codebases. By showing live signal values directly inside the DevTools panel, the plugin helps developers and new team members quickly understand how data flows and responds throughout the app. It may look like a small enhancement, but in practice, it makes a big difference in the day-to-day developer experience.

16. Cursor

ADOPT

Cursor is an IDE that integrates AI assistance directly into the coding workflow, eliminating the need to switch between your editor and external AI tools. Built on the familiar VS Code interface, Cursor provides AI features like intelligent code completion, refactoring suggestions, and natural language code generation.

Cursor is a powerful tool for accelerating development, particularly for features with straightforward logic. It makes prototyping big ideas a breeze, allowing you to quickly get a working model. For tasks that require deep domain knowledge, or for generating code that strictly adheres to your project's coding standards, it can be less effective. We place Cursor in the Adopt ring, as it has proven to be a reliable and active product with a focus on rolling out innovative methods for using AI to enhance developer experience.

17. Dependency cruiser

ADOPT

Dependency Cruiser is a tool for JavaScript and TypeScript that visualizes and validates module dependencies. It helps you understand your project's architecture by creating diagrams that show how your modules are interconnected. You can also validate your codebase against a set of configurable rules, such as forbidding core logic from using UI components or detecting unwanted circular dependencies. To make those rule violations easy to spot, they can be highlighted in the diagrams with a different line color and style.

We have placed Dependency Cruiser in Adopt, since its generated diagrams are an essential tool for gaining architectural insights. The tool proved invaluable when we needed to extract application features into a new, shared library. Using its configurable rules, we created a visual map that precisely highlighted all the dependencies blocking this move, which in turn guided our refactoring strategy.

18. JSR

ADOPT

As far as JavaScript package registries go, npm has long been the dominant player. However, the JavaScript Registry (JSR) is attempting to shake things up. Created by the developers of the Deno JavaScript runtime, this registry embraces modern standards and offers powerful tooling that works for more than just Node.js. JSR has native support for TypeScript, which removes the need to compile code before publishing. It also includes many other useful features. Unlike npm, JSR is a community-driven project overseen by a Governance Board with well-known members from across the JavaScript community.

JSR provides an independent alternative to the Microsoft-owned npm registry and offers control back to the community. This is a significant advantage, but having a registry that's open-source and packed with modern features makes it even better. While many packages are still absent from the JSR registry, that number shrinks each day. We believe JSR is a great package registry for both JavaScript and TypeScript, on both the backend and frontend. New features are added regularly, and we look forward to its future. For legacy packages that still use CommonJS, npm remains a valid choice.

19. schema-dts

ADOPT

Schema-dts is a great tool to help you compose structured data for your web project. It is a collection of TypeScript definitions for Schema.org that makes it easy for you to adhere to the JSON Schema standard. With LLMs on the rise as the new web crawlers and search tools, it is important to make it abundantly clear what data is available and how it is structured for the LLM to understand.

We place schema-dts in the Adopt ring, as it's a nice little schema library that helps you adhere to the specifications.

20. Visily

ADOPT

Visily makes design feel less like a chore and more like creativity unlocked. As an AI-powered wireframing and prototyping tool, it helps you turn ideas from text, sketches, or even screenshots into editable designs in minutes. Visily bridges the early rough wireframes and polished high-fidelity mockups, so teams can move fast without losing quality.

Visily is a great fit for teams that want to turn rough ideas into designs quickly and collaboratively. One of its best features is the ability to transform a screenshot or prompt into a working wireframe, saving hours of work. This makes Visily a smart companion for teams building digital products together.

21. WCAG Color contrast checker

ADOPT

Accessibility is no longer a nice-to-have, it is a core responsibility in modern front-end development. Tools like the WCAG Color Contrast Checker plugin help developers meet accessibility standards more easily by testing color combinations directly in the browser. It checks whether text and background colors meet the minimum contrast ratio requirements set by the Web Content Accessibility Guidelines (WCAG), flagging areas that might cause readability issues for users with visual impairments.

We found this plugin especially helpful in design handoff phases and when reviewing UI changes across different themes or branding updates. Having an instant visual indication of which elements fail to meet the standard keeps accessibility in mind throughout the development process, not just at the end. It is a lightweight tool, but it reinforces a mindset that prioritizes inclusion from the start, which we believe is essential for any serious product team.

22. HTTPie

ADOPT

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 has 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, SOAP, 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, with new features being added regularly.

HTTPie is slowly becoming known as the HTTP client that will make you smile. Though the GUI is still marked as "public beta" on their website, it feels anything but beta. We have been using HTTPie with clients for well over a year now, and could not be happier. Unless you require sophisticated sharing and syncing features, we recommend to start using HTTPie today.

23. Ollama

ADOPT

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

Many organizations are cautious about sharing code with external AI platforms, and for good reasons. A safer approach is to use AI tools locally, and that is where Ollama comes in. While more MacBook users migrate over to ARM-based MacBooks, the performance of Ollama with a local model is good enough for daily use. Ollama also has the capability to run as a server, and the use is then extended to users with less computing power. In our opinion, it's good enough to adopt and use Ollama and use an agent like Continue to integrate it in VSCode and IntelliJ.

24. ast-grep

TRIAL

As a codebase grows, keeping it clean, consistent, and maintainable can feel like an impossible task. The ast-grep tool is a true Swiss Army Knife for managing a codebase, providing a versatile set of capabilities to:

- Find code with structural search patterns based on advanced Abstract Syntax Tree (AST) matching, not just text.
- Enforce consistency with custom, autofixable linting rules to maintain quality standards automatically.
- Automate refactoring on a large scale via its CLI, declarative YAML rules, or programmatically through language bindings.
- Support your stack as a true polyglot, with over 20 languages supported out-of-the-box and easy registration of tree-sitter grammars.

We are placing ast-grep in Trial. While it has proven to be a very useful tool, it is not yet mature enough for wide-scale, unguided adoption. The primary challenge stems from its polyglot matching heuristics, which can cause structural searches to behave inconsistently across different languages. A real-world use case demonstrating this inconsistent behavior is documented on this Gist. This makes simple search patterns potentially error-prone unless they are thoroughly tested against all syntax variations. As a result, achieving reliable results often requires using the more advanced and verbose YAML rule objects, bypassing the simpler CLI approach.

25. Detective

TRIAL

The Detective library, developed by Angular Architects, is an open-source tool designed for code analysis at the architectural level, primarily for TypeScript projects, with seamless integration for Angular and Nx-based applications. It leverages historical data from git logs to uncover hidden patterns in codebases, such as change coupling, where modules are frequently modified together, indicating non-obvious dependencies. It also performs hotspot analysis, identifying complex files with frequent changes that pose higher bug risks, and team alignment analysis to evaluate whether team structures align with module boundaries. By analyzing metrics like churn rate and complexity, Detective generates a hotspot score to prioritize problematic code areas.

Developers should consider trying Detective because it provides actionable insights into codebase health, helping to identify architectural weaknesses that could lead to maintenance challenges or bugs, ultimately improving long-term project maintainability. Its ability to visualize module dependencies and team alignment could improve collaboration and autonomy. For those hesitant to adopt Detective, alternatives like CodeScene by Adam Tornhill offer more comprehensive forensic analysis but come with a commercial cost, while tools like SonarQube focus on static code analysis with broader language support but less emphasis on architectural forensics. Nx's built-in dependency graph provides a similar visualization for monorepos but lacks Detective's focus on change coupling and team alignment, making Detective a lightweight, open-source option for TypeScript developers seeking targeted architectural insights.

26. Gemini Nano

TRIAL

Gemini Nano brings Google's Gemini AI directly onto your device. As the smallest member of the Gemini family, it's designed to run efficiently on phones and laptops without needing a constant internet connection. This means faster responses, more privacy, and AI that works even when you're offline.

Gemini Nano is a great fit for developers who want to embed lightweight AI directly into everyday apps. One of its best features is on-device text generation and summarization, making smart features possible without heavy cloud calls. This makes Gemini Nano a powerful tool for teams building the next wave of AI-driven experiences.

27. GitButler

TRIAL

GitButler is a Git client that is designed around 'virtual branches'. It operates directly on your active working directory, where it will automatically put uncommitted changes into a separate group, while keeping the main branch up-to-date. This allows the developer to work on multiple features without the need to constantly switch between branches or stashing all the changes. With GitButler, we simply change the code, GitButler isolates and groups the changes automatically, and we can choose what we want to push.

We are placing GitButler in Trial. While it has proven to be a very beneficial tool, there is no stable version 1.0.0 at the moment of writing. The current way of working and UI will most likely be tweaked and is actively developed on by the co-founder of GitHub and his team.

28. nuqs

TRIAL

Nuqs is a Type-safe search params state manager for React. The addition of adding this library to the radar serves as a statement to developers who misuse the useState and useEffect hooks, and as a reminder to use a pragmatic approach, using native browser functionalities when possible.

Using query parameters for state management results in SEO-friendly URLs with persistent state that is shareable and feeds directly into your analytics tool of choice. Nuqs is most useful for keeping a persistent state for pagination, filters, the state of UI elements such as tabs, modals, navigation, and much more. But don't use it for sensitive data like user data or security tokens. We place nuqs in the Adopt ring. Be aware that it does not replace state that should not be exposed, such as temporary, internal data flow state, but should be used as an additional solution to expose publicly accessible state to the URL.

29. Warp

TRIAL

Warp is a new, fully customizable, Rust-based terminal that enhances the command-line interaction with AI-powered features and IDE-like editing experience. It offers intelligent suggestions for the command you put in, helps with debugging errors that may pop up, supports multiline editing, and even mouse navigation.

We have put Warp in Trial since it's a great tool to start using, because of its ease of use, the AI-assisted productivity, and collaboration features with others. You do need to be mindful that Warp uses its AI agents via the cloud, and therefore, you must be careful what you share with these agents. Ultimately, using Warp will benefit the workflow of engineers at any level.

30. Claude Code

TRIAL

Claude Code brings AI-powered assistance directly to your development workflow through a command-line interface that lets you delegate coding tasks to Claude without leaving your terminal. This agentic coding tool can help with writing new code, debugging existing issues, refactoring legacy systems, and understanding complex codebases. By integrating seamlessly with the CLI, Claude Code works alongside your existing command-line tools and development workflows, executing multi-step coding tasks autonomously across different programming languages and frameworks.

We placed Claude Code in the Trial ring because AI-enhanced developer tooling represents a significant shift in software development practices, but requires hands-on evaluation to understand its true impact. It's worth experimenting with Claude Code on real development tasks, particularly for complex refactoring or when working with unfamiliar codebases, where AI assistance can provide immediate value. In our experience, even limited trials with AI coding tools often reveal unexpected productivity gains and new approaches to problem-solving. It's time to explore how Claude Code can enhance your development process.

31. 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. Currently, Knip can be used to find unused files, exports, and dependencies. Knip can also be integrated into CI/CD pipelines to automate these checks, ensuring ongoing codebase optimisation.

While Knip is still in its infancy, it is seeing wide adoption, and many open source projects we use every day have started to adopt Knip to keep a grip on their codebase. It is a great tool to streamline both JavaScript and TypeScript projects and to improve the overall code quality and maintainability of the project. Its ability to run both locally and in the CI/CD environment makes it a good fit for large organisations with complex architectures such as monorepos. With features such as auto-fix being shipped, and it working with more and more projects out-of-the-box without configuration, there could not be a better time to give it a spin. At this point in time, it might still require more effort than the value it delivers, so do not get your hopes up just yet.

32. Builder.io

ASSESS

Builder.io is a solution for designing and building web applications in unison. It can connect with various MCP servers and backends, connect to repositories, and the ticket system of your choice.

We had a great developer experience creating UI elements and components in Figma with prompts, then exporting them to Builder.io, which generates code for your project or sends refinements back to Figma. This way, you can synchronize between design and code. Design Tokens are supported, creating consistently styled UI elements, although only available under the Enterprise payment plan. We rate Builder.io as Assess: we're keeping our eyes on this solution to bridge the gap between code and design using AI prompting. The "full-stack" title may soon need to include the fields of prompt engineer, UX architect, and UI designer.

33. Stitch

ASSESS

Stitch is a Google Labs AI experiment that generates desktop and mobile UI designs from text prompts or images. You can refine designs through AI chat and export them to Figma. The exported Figma files provide a good starting point, though manual adjustments are often needed. The tool is free but can be slow to respond.

We place this tool in the Assess ring, as it may work to get you started with a design, but it is not something you can rely on to generate a design that is ready to use.

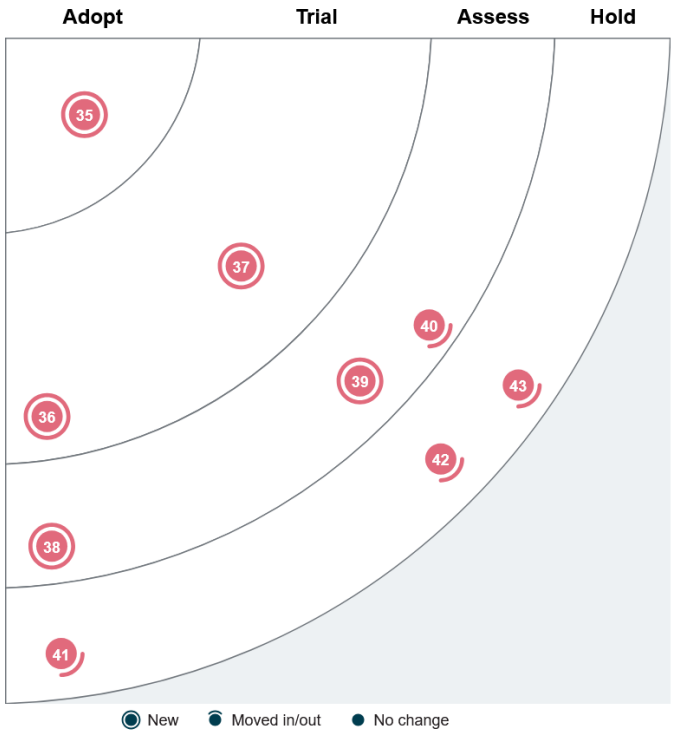
34. Supermaven

HOLD

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.

When Supermaven was introduced, its speed and the ability to adjust code across different files were the big selling points. However, competitors have caught up and surpassed it with powerful command-line interface (CLI) tools and coding agents. In November 2024, Supermaven's CEO announced that they would join Cursor in the development of its code editor. Since then, very few updates have been made to Supermaven, and the JetBrains plugins no longer function in the newer editor versions. We do not suggest using Supermaven as an editor plugin anymore, but instead recommend using other coding assistants that have more capabilities.

Languages & frameworks



Adopt
35. React Router v7

Trial
36. TensorFlow.js
37. Zoneless

Assess
38. Remix v3
39. TanStack Start
40. NgRx SignalStore

Hold
41. Angular Material
42. NextJs
43. Leptos

35. React Router v7

ADOPT

React Router v7 is the latest major release of the popular React routing library, representing a significant evolution beyond traditional client-side routers. The maintainers describe v7 as the next major version of the Remix framework (following Remix v2), reflecting how the Remix full-stack capabilities have now been integrated into React Router. This version introduces an optional "Framework Mode," which brings framework-like features into React Router. With this mode enabled, React Router v7 provides built-in support for data loading and form submissions on routes, type-safe route definitions, and even server-side rendering (SSR) and static site generation. Importantly, you can still use it in a lightweight, "router-only" way as in v6, but the framework option is there whenever you want more out-of-the-box functionality.

We are adopting React Router v7 because it follows the same successful footsteps of Remix, a framework we had already moved to the Adopt ring in our previous tech-radar. In many ways, React Router v7 is Remix under the hood. The Remix team merged their framework directly into it, bringing the two projects together. The maintainers of Remix now officially recommend starting new projects with React Router v7, and existing Remix projects can migrate to it with minimal friction. For us, this is a logical next step that maintains the benefits we are used to while embracing a more flexible and future-ready foundation.

36. TensorFlow.js

TRIAL

TensorFlow.js brings machine learning directly into the browser. As a JavaScript library, it lets developers train and run ML models using only web technologies, without needing Python or server-side infrastructure. This makes it easier to add intelligent features to websites and web apps, accessible to anyone with a browser.

TensorFlow.js is a great fit for developers who want to experiment with AI directly in a browser. One of its best features is running models fully client-side, improving privacy and reducing latency. This makes TensorFlow.js a flexible choice for teams building interactive, AI-powered web experiences.

37. Zoneless

TRIAL

By default, Angular uses Zone.js to monkey-patch async APIs and automatically trigger change detection when needed. This makes development easier, but can add performance overhead and reduce predictability in large or complex web applications.

Since Angular 16, it has been possible to run Angular applications without relying on Zone.js. However, many projects continued using Zone.js because popular libraries had not yet adopted support for zoneless change detection. As of the time of writing, Angular 20 is the latest release, and more libraries are starting to support zoneless applications. We place zoneless in the Trial ring, as it's just out of developer preview since Angular 20.2, and a successful migration still depends on the compatibility of the libraries used in your project. Before migrating to a zoneless setup, teams can prepare by adopting the `ChangeDetectionStrategy.OnPush` strategy.

38. Remix v3

ASSESS

Remix v3 represents a complete reimagining of the framework. The team is moving away from React and rebuilding on a fork of Preact, to create a modern full-stack toolkit grounded in web standards and minimal dependencies. It introduces a model-first approach designed to work well with AI tools, built-in database drivers, a refreshed component library inspired by Reach UI, and a runtime-focused architecture that avoids the need for bundlers or compilers. The result is a framework that aims to be simpler, smaller, and more aligned with the core principles of how the web works.

We place Remix v3 in the Assess ring because it presents a bold vision that aligns with emerging trends in web development. Although not yet ready for production, its shift toward web-native APIs, smaller runtimes, and AI-friendly patterns is promising. It reflects a growing interest in frameworks that reduce complexity and embrace standards over configuration. While Remix v3 is still early in its journey, we are following its progress closely to see if it can deliver on its goals and help shape the next generation of full-stack development.

39. TanStack Start

ASSESS

TanStack Start is a full-stack web framework built on top of Vite and React. It provides a complete solution for building modern web applications with built-in routing, data fetching, and server-side rendering capabilities. Unlike traditional frameworks that bundle everything together, TanStack Start takes a modular approach, allowing developers to use only what they need. The team behind TanStack has proven to be a reliable and active team, with a strong focus on performance and developer experience.

You may choose TanStack Start when you need a fresh start with a modular architecture on a solid foundation, with great developer experience, and maximum runtime performance with a minimal bundle size. But it requires an understanding of multiple TanStack libraries. We place TanStack Start in the Assess ring, since the framework is still in BETA and not ready for production use. Try it out, especially if you already work with many of the other TanStack libraries, but be aware that it is not ready for production yet.

40. NgRx SignalStore

ASSESS

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. However, with the new and experimental Angular Resource API, it becomes possible to handle side-effects like fetching data in an Angular native way. This way, it becomes less relevant to use NgRx for simple use cases. We therefore recommend to use NgRx for complex use cases only and to keep an eye on the development that is going on in the Resource API.

41. Angular Material

HOLD

Angular Material is a component library maintained by the Angular team. This framework brings Google's Material Design system to Angular applications. Providing components like buttons, dialogs, tables, and form controls.

At first, it may seem like a safe choice. An official, stable, and widely used library. But in practice, it often falls short for real-world needs. We recommend avoiding Angular Material in most cases for these reasons:

- Accessibility is limited. It does not fully support WCAG 2.1, which can be a blocker for inclusive design.
- Custom styling is difficult. Many projects require a look beyond Material Design. While design tokens help, customization remains limited compared to other libraries.
- Design mismatches can occur. The components often don't fully match official Material Design specs. This leads to confusion when UX designs in Figma don't align with what developers can build.

42. NextJs

HOLD

Next.js is an open-source, React-based web framework that enables developers to build server-side rendered and static websites with React. It simplifies the development of web applications by providing a structured, full-featured solution that includes file-system routing, automatic code splitting, and hot-reloading, all configured out of the box. It streamlines the creation of modern web applications by focusing on developer experience and performance optimization.

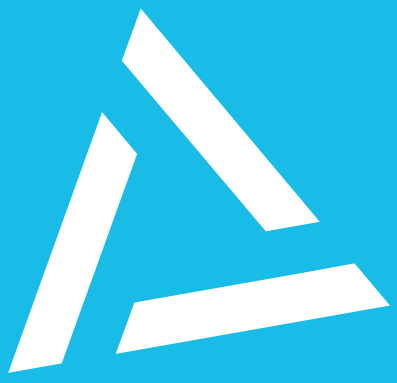
A significant point of contention in the Next.js ecosystem is the tight coupling of the framework with Vercel's platform. Some developers believe that Next.js features are optimized specifically for Vercel's infrastructure, which can lead to a feeling of vendor lock-in. This makes it difficult to deploy Next.js applications on other platforms without losing functionality or experiencing performance issues. While Vercel is well-suited for smaller projects, it may not be the ideal solution for large-scale enterprise applications. Therefore, you might consider alternative frameworks that offer more platform flexibility.

43. Leptos

HOLD

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 Hold ring for good reason. Learning Rust and Leptos requires significant time and effort, and the limited pool of experienced developers makes building expertise difficult for teams. Although Leptos delivers server performance an order of magnitude better than typical JavaScript frameworks, teams without existing Rust expertise will find JavaScript-based solutions more practical. The framework has a smaller ecosystem and steeper learning curve than its alternatives.



focus on
front-end
development