Angular versus React in 2025: 4 Verrassende Waarheden Die Alles Veranderen

De discussie "Angular vs. React" is al bijna een decennium een constante in de wereld van webontwikkeling. Generaties ontwikkelaars hebben de argumenten aangehoord: de complete, maar rigide structuur van Angular tegenover de flexibele, maar gefragmenteerde vrijheid van React. Deze debatten waren gebaseerd op fundamentele waarheden die lang standhielden. Maar in het technologische landschap van eind 2025 zijn die waarheden niet langer geldig. De spelregels zijn fundamenteel herschreven.

Recente, transformerende updates – met name Angular 21 en React 19.2 – hebben de kernarchitecturen van beide frameworks zo drastisch veranderd dat de oude aannames ronduit misleidend zijn geworden. De keuzes die een architect vandaag maakt, moeten gebaseerd zijn op een nieuwe realiteit. Dit artikel duikt in de vier meest verrassende en impactvolle verschuivingen die elke ontwikkelaar, tech lead en architect moet begrijpen. Dit zijn niet de nuances die je in een changelog vindt, maar de tektonische verschuivingen die bepalen hoe we de komende jaren stabiele, veilige en performante applicaties bouwen.

---

1. De Grote Verrassing: Angular is Nu Koning van de Granulaire Performance

Jarenlang was de mythe onwrikbaar: React, met zijn slimme Virtual DOM, was de standaard voor snelheid. Angular, met het zware en complexe Zone.js, stond bekend als traag, vooral in complexe applicaties. In 2025 is deze realiteit volledig omgekeerd, met name in data-intensieve applicaties.

De nieuwe realiteit wordt gedreven door Angular 21's "Zoneless" architectuur. Door Zone.js volledig te elimineren, heeft Angular een fundamenteel efficiënter reactiviteitsmodel omarmd: **Signals**. Deze technologie, die *Fine-Grained Reactivity* mogelijk maakt, zorgt voor "chirurgische" updates aan de DOM. Architectonisch gezien is dit een verschuiving van O(n) complexiteit (het doorlopen van een componentenboom) naar O(1) complexiteit. Stel je een ledenadministratie voor met een datagrid van duizenden rijen. Als je de status van één lid wijzigt, werken Signals alleen die specifieke cel in de tabel bij. De update is direct en de complexiteit is constant, ongeacht de grootte van de lijst. Dit resulteert niet alleen in superieure update-prestaties, maar ook in een lagere geheugenbelasting door de afwezigheid van een VDOM.

React heeft ondertussen de **React Compiler** geïntroduceerd. Dit is een krachtige innovatie die de noodzaak voor handmatige *memoization* (`useMemo`, `useCallback`) elimineert. Echter, de fundamentele aanpak blijft O(n): de compiler optimaliseert het proces van het

vergelijken van een (virtuele) boomstructuur. Tegelijkertijd introduceert React 19.2 met de **`<Activity>` component** een krachtige tool voor dashboards: het kan componenten (zoals een tabblad) verbergen zonder hun state te vernietigen, wat ideaal is voor het behouden van scrollposities en formulierdata.

De conclusie is verrassend en genuanceerd: voor applicaties met hoogfrequente, granulaire updates in grote datagrids heeft Angular nu een architectonisch voordeel. Voor administratieve dashboards waar het behouden van de state tussen tabbladen cruciaal is, biedt React's `<Activity>` component een significante winst.

---

2. Je AI-Assistent Begrijpt Angular Beter (En Dat Is Revolutionair)

In 2025 zijn AI-codeassistenten zoals GitHub Copilot, geïntegreerd in editors als Cursor, een integraal onderdeel van de ontwikkelworkflow. Verrassend genoeg is er een aanzienlijk verschil ontstaan in hoe effectief deze assistenten zijn voor Angular en React, en Angular heeft een beslissende voorsprong genomen.

De game-changer is een unieke feature in Angular 21: de **Model Context Protocol (MCP) Server**. Deze server, die draait als onderdeel van de Angular CLI, functioneert als een "tolk" tussen de AI en jouw project. Het geeft de AI real-time inzicht in de specifieke structuur van je project, je configuratie en, cruciaal, de allernieuwste officiële Angular-documentatie.

De impact hiervan is enorm. Het lost het hardnekkige 'hallucinatie'-probleem op. De MCP-server dwingt de AI om moderne, best-practice Angular 21-code te genereren: Standalone Components, Zoneless architectuur en op Signals gebaseerde logica. Het zal geen code meer suggereren die leunt op verouderde `NgModules` of `zone.js`-trucs.

In het React-ecosysteem is de situatie anders. Hoewel Copilot zeer capabel is met React, is het getraind op een gigantische dataset die gedomineerd wordt door oudere code (v16-v18). Dit leidt tot een veelvoorkomend probleem: de AI suggereert verouderde patronen, zoals het gebruik van `useEffect` voor data-fetching, in plaats van de moderne server-component- en suspense-patronen van React 19.2. Tenzij de ontwikkelaar zeer specifieke prompts geeft, is de kans groter dat de voorgestelde code niet de modernste aanpak volgt. Angular's MCP-server versnelt niet alleen de leercurve voor modern Angular, maar verhoogt ook proactief de kwaliteit en toekomstbestendigheid van de codebase.

---

3. Security by Default: Waarom Angular's "Vangrails" Belangrijker Zijn Dan Ooit

React 19.2's **Server Actions** illustreren een "veiligheidsparadox". Deze krachtige features vervagen de grens tussen client en server, maar leggen de volledige verantwoordelijkheid voor beveiliging bij de ontwikkelaar. Elke Server Action is in feite een publiek API-endpoint. Als een ontwikkelaar vergeet een gebruikerssessie te valideren, ontstaat er onmiddellijk een ernstig beveiligingslek. Het React-team onderkent dit en werkt aan mitigaties zoals de

experimentele **'Taint' APIs**, die moeten voorkomen dat gevoelige data per ongeluk naar de client lekt, maar dit vereist nog steeds expliciete actie van de ontwikkelaar.

Angular hanteert daarentegen een "defense-in-depth" filosofie. Dit blijkt uit meerdere ingebouwde mechanismen:

1. **Functionele Route Guards:** Deze controleren autorisatie *voordat* een component wordt geladen. Dit voorkomt de beruchte "flash of unauthorized content" die kan optreden in React-apps waar de autorisatiecheck vaak in een `useEffect` plaatsvindt *nadat* de component al is gemount.
2. **De DomSanitizer:** Angular behandelt standaard alle content als onveilig om Cross-Site Scripting (XSS) te voorkomen. Het omzeilen hiervan vereist de expliciete functie `bypassSecurityTrustHtml`, die fungeert als een rode vlag tijdens code-audits. Dit staat in contrast met React, waar je voor dezelfde bescherming afhankelijk bent van externe bibliotheken zoals `DOMPurify`, wat een **"supply chain risk"** introduceert.
3. **Ingebouwde XSRF/CSRF-bescherming:** De `HttpClient` van Angular biedt automatische bescherming tegen Cross-Site Request Forgery, een complexe kwetsbaarheid die in React vaak handmatige configuratie vereist.

Voor applicaties met gevoelige data biedt Angular's 'out-of-the-box' veiligheidsstructuur een fundament dat menselijke fouten minimaliseert en inherent robuuster is.

---

4. De UI-Keuze: "Kant-en-Klaar" versus "Zelf Assembleren"

De ecosystemen van Angular en React hebben tegengestelde paden gekozen voor het bouwen van UI's, een keuze met enorme impact op de productiviteit.

De dominante filosofie in het React-ecosysteem wordt geïllustreerd door **Shadcn/ui**. Dit is geen traditionele bibliotheek, maar een collectie van componenten waarvan je de broncode in je project kopieert. Dit "copy-paste" model biedt maximale flexibiliteit. De keerzijde wordt echter duidelijk bij het bouwen van complexe administratieve applicaties. Voor een geavanceerde datagrid moet je zelf een oplossing "assembleren" door een headless bibliotheek zoals **TanStack Table** te integreren, wat aanzienlijke boilerplate-code en configuratie vereist.

Angular's ecosysteem biedt met **PrimeNG** een fundamenteel andere, "batteries-included" aanpak. Historisch bekritiseerd om zijn rigide styling, heeft PrimeNG dit opgelost door een moderne theming-architectuur te omarmen die gebaseerd is op design tokens en volledige Tailwind CSS-integratie. Het cruciale verschil is dat PrimeNG extreem rijke, kant-en-klare componenten levert. Een datagrid met **out-of-the-box multi-column sortering, complexe filtering (Excel-stijl), en virtuele scrolling** is een kwestie van configuratie, niet van assemblage. Voor teams die de Shadcn-filosofie prefereren, ontstaat er in Angular overigens een alternatief met **SpartanNG**.

Dit resulteert in een duidelijke afweging. Voor een unieke marketingwebsite biedt de ultieme flexibiliteit van React's "zelf assembleren" model voordelen. Maar voor het bouwen van functionele, data-rijke interfaces, biedt de superieure productiviteit van Angular's "kant-en-klaar" ecosysteem een dramatische versnelling van de ontwikkeltijd.

Conclusie

De frontend-wereld van 2025 is niet meer wat hij was. De oude debatten zijn achterhaald. Angular is niet langer de trage, logge kolos, maar een chirurgisch precieze performance-kampioen voor data-intensieve taken. React is niet langer de eenvoudige, lichtgewicht bibliotheek, maar een complex ecosysteem dat krachtige server-integraties biedt ten koste van een hogere beveiligingslast voor de ontwikkelaar. Angular heeft een verrassende voorsprong genomen in AI-ondersteuning, terwijl de keuze voor een UI-bibliotheek een strategische afweging is geworden tussen ultieme flexibiliteit en pure productiviteit. Nu de oude waarheden niet meer gelden, is de vraag niet langer 'welk framework is populairder?', maar 'welk framework biedt de meest voorspelbare en stabiele basis voor de problemen die ik vandaag moet oplossen?'