NEWS · JUNE 15, 2026

The One Who Got Away: Anders Hejlsberg, Decades Later

By Dr. Holger Flick



A lot of long-time Delphi developers carry a small, fond nostalgia. We don't talk about it much, but it's there. It's the feeling you get when you read about Anders Hejlsberg shipping yet another world-changing thing — and you remember where that lineage actually started.

For many of us, the personal version of the story begins with Delphi. But Anders' version starts more than a decade earlier, at Borland, with Turbo Pascal.

Where It Started: Turbo Pascal

Before C#. Before TypeScript. Before Delphi, even — Anders Hejlsberg was the man behind **Turbo Pascal**. He wrote the original compiler, and when Borland shipped it in 1983 for next to nothing, it landed like a thunderclap: a fast compiler and an editor bundled together, on a floppy, for the price of a textbook. For a whole generation, that was their first taste of programming actually feeling *good*. Mine included — by the time I found Delphi, I was already a Turbo Pascal kid.

He spent years at Borland as their chief engineer, and in 1995 that work culminated in **Delphi** — Object Pascal with a visual designer, a component model, and a compiler that turned source into a single, blistering-fast native executable while the rest of the industry was still arguing about runtimes. Building Windows applications suddenly felt less like an engineering ordeal and more like craftsmanship. Anders was the chief architect of that, the same way he'd been the architect of Turbo Pascal before it.

Then, in 1996, he left Borland for Microsoft. And for a lot of us, the Pascal story has had a little asterisk on it ever since.

I was reminded of all this by a wonderful interview my friend Thorsten Hindermann pointed me to — Anders sitting down on *The Pragmatic Engineer* to trace his whole career, from his first compiler all the way to how he uses AI today.

It's worth every minute. But watching it, I kept coming back to the same bittersweet thought.

What He Built Next

He built **C#**, and the way he describes it in the interview is pure Anders — a small group of experienced language designers meeting a few hours a week, sweating every detail. That's the same discipline that gave Pascal its clarity, from Turbo Pascal through Object Pascal. C# didn't happen by committee. It happened because someone who'd already designed two languages developers *loved* knew exactly what that took.

Then he gave the JavaScript world **TypeScript** — and at its heart is the most Pascal idea there is: *types*.

JavaScript had spent two decades insisting that dynamic was freedom; Anders quietly bolted a real, static type system onto it and let developers feel the difference. Suddenly the editor could tell you what a value *was*, catch a typo before you ran it, and refactor with confidence. That's not magic — it's what a type system buys you, the same payoff Object Pascal had been delivering for years. Strong typing first; the great tooling falls out of it. Delphi developers had lived that bargain since the '90s.

So when people ask "what would Delphi have been if he'd stayed?" — I think the honest answer is that the question is backwards. The world got *more* of what made Delphi great, just wearing different clothes. C# is Pascal-discipline for the .NET era. TypeScript is Pascal's faith in static types, smuggled into the most dynamic language on earth. He never really left the philosophy. He just kept applying it to bigger and bigger audiences.

The Lineage Is Real

This isn't sentimental hand-waving. The DNA is genuinely traceable:

- **Strong, explicit typing as a feature, not a tax.** Object Pascal believed it. C# believed it. TypeScript was literally an argument *to JavaScript* that explicit types make you faster, not slower. That's a thirty-year-old Delphi argument, finally winning on the web.
- **The compiler and the IDE as one product.** Delphi never treated the editor as an afterthought. Neither did TypeScript. The "language server" model that every modern editor now relies on traces straight back to that conviction.
- **Native, fast, single-artifact output.** It's no accident that the TypeScript team has been rewriting their compiler for raw speed. Anders has *always* cared about performance the way Delphi developers do. Delphi never accepted slow. Neither did he.

When you've spent decades in Delphi, you don't watch the modern dev-tools world and feel like an outsider. You can see the family resemblance everyone else missed.

Then There's AI

The part of the interview that landed hardest for me was Anders on AI — how he uses it, and which *language features* actually make a language friendly to AI-assisted development. He talks about software craft moving away from writing code line by line.

And I couldn't help thinking: the languages best suited to this moment are the ones with unambiguous structure and strong typing. Clear syntax. Explicit contracts. Exactly the qualities Object Pascal has had since before most of today's developers were born. The man who keeps designing languages that AI loves to work with is the same man who designed the one a lot of us never stopped loving.

That's not a coincidence. It's a worldview — and it's a worldview Delphi developers recognize on sight.

Still Family

So no, I don't think Delphi lost Anders Hejlsberg when he walked into Microsoft in 1996. The work just got bigger.

Watch the interview. You'll hear him talk about Turbo Pascal and Delphi not as a footnote on the way to the "important" work, but as the foundation everything else was built on. Because that's what it was. Every developer who has ever felt the joy of a fast compile, a great IDE, and a type system that has their back is, in some small way, living in a world he started building back in Borland's heyday.

We miss having him in the Pascal world. Of course we do. But the best language designer of his generation cut his teeth there — and he's been quietly proving that philosophy right ever since.

That's not a loss. That's a legacy.

Where to find Anders Hejlsberg:

- *X:* @ahejlsberg
- *GitHub:* github.com/ahejlsberg

INDEPENDENT · AD-FREE · READER-SUPPORTED

Free to read. Not free to make.

Every tutorial, article, and line of example code on Holger's Code is published without a paywall and without ads. This page is about what keeps it that way.

Professional work, published free

The content here is held to the same standard as professional client work: every tutorial is built as a real project, tested against current versions, documented step by step, and revisited when the frameworks move on. A single in-depth tutorial routinely takes days — and that's before the bills behind it: servers, storage, build infrastructure, licenses, and the hardware everything is verified on.

If an article or tutorial here has saved you an afternoon of debugging, taught you a technique, or helped you ship, it created real value for you. Supporting it isn't charity — it's paying for professional work you found useful, at whatever price you choose. And if you can't, keep reading anyway. That's what the free part means.

Ways to support

KO-FI

Buy me a coffee

A one-time tip — no account, no subscription, any amount.

BOOKS & COURSES

Buy the paid work

The paid tier of everything published free here — and you get something lasting in return.

SPREAD THE WORD

Share an article

Send a tutorial to a colleague, cite an article, subscribe to the RSS feed.

Nothing here goes behind a paywall, whether you chip in or not. But if something on this site earned its keep today, this is where to say so.

Support on Ko-fi —