NEWS · JUNE 15, 2026

Der, der uns davonkam: Anders Hejlsberg, Jahrzehnte später

By Dr. Holger Flick



Viele langjährige Delphi-Entwickler tragen eine kleine, liebevolle Nostalgie mit sich herum. Wir reden nicht oft darüber, aber sie ist da. Es ist dieses Gefühl, wenn man liest, dass Anders Hejlsberg schon wieder etwas Weltveränderndes ausgeliefert hat — und man sich daran erinnert, wo diese Linie eigentlich begann.

Für viele von uns beginnt die persönliche Version der Geschichte mit Delphi. Aber Anders' Version beginnt mehr als ein Jahrzehnt früher, bei Borland, mit Turbo Pascal.

Wo alles begann: Turbo Pascal

Vor C#. Vor TypeScript. Sogar vor Delphi — Anders Hejlsberg war der Mann hinter **Turbo Pascal**. Er schrieb den ursprünglichen Compiler, und als Borland ihn 1983 für einen Spottpreis auf den Markt brachte, schlug das ein wie ein Donnerschlag: ein schneller Compiler und ein Editor in einem Paket, auf einer Diskette, zum Preis eines Lehrbuchs. Für eine ganze Generation war das der erste Moment, in dem sich Programmieren tatsächlich *gut* anfühlte. Für mich auch — als ich Delphi entdeckte, war ich längst ein Turbo-Pascal-Kind.

Er verbrachte Jahre bei Borland als deren Chefingenieur, und 1995 gipfelte diese Arbeit in **Delphi** — Object Pascal mit einem visuellen Designer, einem Komponentenmodell und einem Compiler, der Quelltext in eine einzige, rasend schnelle native Executable verwandelte, während der Rest der Branche noch über Runtimes stritt. Windows-Anwendungen zu bauen fühlte sich plötzlich weniger wie eine technische Tortur an und mehr wie

Handwerkskunst. Anders war der Chefarchitekt dieses Werks, so wie er zuvor der Architekt von Turbo Pascal gewesen war.

Dann, 1996, verließ er Borland in Richtung Microsoft. Und für viele von uns trägt die Pascal-Geschichte seither ein kleines Sternchen.

An all das erinnerte mich ein wunderbares Interview, auf das mich mein Freund Thorsten Hindermann aufmerksam gemacht hat — Anders im Gespräch bei *The Pragmatic Engineer*, wo er seine gesamte Karriere nachzeichnet, von seinem ersten Compiler bis hin dazu, wie er heute mit KI arbeitet.

https://www.youtube.com/watch?v=K-Xv8D8NjTk

Es lohnt sich jede Minute. Aber beim Zuschauen kam ich immer wieder auf denselben bittersüßen Gedanken zurück.

Was er danach baute

Er baute **C#**, und die Art, wie er es im Interview beschreibt, ist Anders in Reinform — eine kleine Gruppe erfahrener Sprachdesigner, die sich ein paar Stunden pro Woche trifft und über jedem Detail brütet. Es ist dieselbe Disziplin, die Pascal seine Klarheit gab, von Turbo Pascal bis Object Pascal. C# entstand nicht per Komitee. Es entstand, weil jemand, der bereits zwei Sprachen entworfen hatte, die Entwickler *liebten*, genau wusste, was dafür nötig war.

Dann schenkte er der JavaScript-Welt **TypeScript** — und in dessen Kern steckt die pascaligste Idee überhaupt: *Typen*. JavaScript hatte zwei Jahrzehnte lang darauf bestanden, dass Dynamik Freiheit bedeute; Anders schraubte in aller Stille ein echtes, statisches Typsystem darauf und ließ die Entwickler den Unterschied spüren. Plötzlich konnte der Editor Ihnen sagen, was ein Wert *ist*, einen Tippfehler abfangen, bevor Sie das Programm starten, und mit Zuversicht refaktorieren. Das ist keine Magie — das ist genau das, was ein Typsystem einbringt, derselbe Gewinn, den Object Pascal seit Jahren lieferte. Erst starke Typisierung; das großartige Tooling folgt daraus. Delphi-Entwickler lebten diesen Handel schon seit den 90ern.

Wenn also jemand fragt: „Was wäre aus Delphi geworden, wenn er geblieben wäre?" — dann lautet die ehrliche Antwort meiner Meinung nach, dass die Frage verkehrt herum gestellt ist. Die Welt bekam *mehr* von dem, was Delphi großartig machte, nur in anderen Kleidern. C# ist Pascal-Disziplin für die .NET-Ära. TypeScript ist Pascals Glaube an statische Typen, eingeschmuggelt in die dynamischste Sprache der Welt. Er hat die Philosophie nie wirklich verlassen. Er hat sie nur auf immer größere Zielgruppen angewandt.

Die Abstammungslinie ist real

Das ist keine sentimentale Schwärmerei. Die DNA lässt sich tatsächlich nachverfolgen:

- **Starke, explizite Typisierung als Feature, nicht als Steuer.** Object Pascal glaubte daran. C# glaubte daran. TypeScript war buchstäblich ein Plädoyer *an JavaScript*, dass explizite Typen einen schneller machen, nicht langsamer. Das ist ein dreißig Jahre altes Delphi-Argument, das endlich im Web gewinnt.
- **Compiler und IDE als ein Produkt.** Delphi hat den Editor nie als Nebensache behandelt. TypeScript ebenso wenig. Das „Language Server"-Modell, auf das sich heute jeder moderne Editor stützt, führt in direkter Linie auf diese Überzeugung zurück.
- **Native, schnelle Ausgabe in einem einzigen Artefakt.** Es ist kein Zufall, dass das TypeScript-Team seinen Compiler gerade für rohe Geschwindigkeit neu schreibt. Anders war Performance schon *immer* so wichtig, wie sie Delphi-Entwicklern wichtig ist. Delphi hat langsam nie akzeptiert. Er auch nicht.

Wer Jahrzehnte in Delphi verbracht hat, schaut nicht als Außenseiter auf die moderne Dev-Tools-Welt. Man erkennt die Familienähnlichkeit, die alle anderen übersehen haben.

Und dann ist da noch KI

Der Teil des Interviews, der mich am meisten getroffen hat, war Anders zum Thema KI — wie er sie einsetzt und welche *Sprachfeatures* eine Sprache tatsächlich freundlich für KI-gestützte Entwicklung machen. Er spricht davon, dass sich das Software-Handwerk vom zeilenweisen Schreiben von Code wegbewegt.

Und ich musste unweigerlich denken: Die Sprachen, die für diesen Moment am besten geeignet sind, sind die mit eindeutiger Struktur und starker Typisierung. Klare Syntax. Explizite Verträge. Genau die Eigenschaften, die Object Pascal hatte, bevor die meisten heutigen Entwickler geboren wurden. Der Mann, der immer wieder Sprachen entwirft, mit denen KI gerne arbeitet, ist derselbe Mann, der die Sprache entwarf, die viele von uns nie aufgehört haben zu lieben.

Das ist kein Zufall. Das ist eine Weltanschauung — und es ist eine Weltanschauung, die Delphi-Entwickler auf den ersten Blick wiedererkennen.

Immer noch Familie

Also nein, ich glaube nicht, dass Delphi Anders Hejlsberg verloren hat, als er 1996 bei Microsoft anfing. Die Arbeit wurde nur größer.

Sehen Sie sich das Interview an. Sie werden hören, wie er über Turbo Pascal und Delphi spricht — nicht als Fußnote auf dem Weg zur „wichtigen" Arbeit, sondern als das Fundament, auf dem alles Weitere gebaut wurde. Denn genau das war es. Jeder Entwickler, der je die Freude eines schnellen Compile-Vorgangs, einer großartigen IDE und eines Typsystems, das ihm den Rücken freihält, gespürt hat, lebt in einem kleinen Stück der Welt, die er damals in Borlands Glanzzeit zu bauen begann.

Wir vermissen ihn in der Pascal-Welt. Natürlich tun wir das. Aber der beste Sprachdesigner seiner Generation hat dort sein Handwerk gelernt — und er beweist seither in aller Stille, dass diese Philosophie richtig war.

Das ist kein Verlust. Das ist ein Vermächtnis.

Wo Sie Anders Hejlsberg finden:

- *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 —