NEWS · OCTOBER 27, 2025

Erweitern Sie Ihr Delphi Know-how: Die TypeScript Chance (Teil 1 von 5)

By Dr. Holger Flick

Nach 30 Jahren Delphi-Entwicklung hätte ich nie gedacht, dass ich das schreiben würde. Aber hier sind wir, und ich muss mitteilen, warum das Verstehen von TypeScript, React und Next.js möglicherweise die wichtigste Geschäftsentscheidung ist, die Sie in diesem Jahr treffen.

Warum diese Serie existiert

Lassen Sie mich von vornherein klarstellen: **Delphi ist immer noch phänomenal**. Für Windows Desktop-Anwendungen und datenbankintensive Enterprise-Software bleibt es eine der produktivsten Umgebungen, die je geschaffen wurden. Die VCL ist erprobt, die Database-Komponenten sind unübertroffen, und die Compile-Zeiten sind blitzschnell.

Aber die Welt hat sich verändert. Ihre Kunden fragen nach Web-Anwendungen. Sie wollen auf ihre Systeme von iPads, Chromebooks und Smartphones zugreifen. Sie wollen Anwendungen, die keine Installation erfordern. Sie wollen moderne Benutzeroberflächen, die aussehen wie die Apps, die sie täglich verwenden. Und ehrlich gesagt? Echte Cross-Platform Web-Anwendungen mit Next.js zu erstellen ist nicht nur effizienter – es ist dramatisch kosteneffektiver, als zu versuchen, Delphi für das Web funktionieren zu lassen.

Die Anders Hejlsberg Verbindung

Hier ist etwas, das sofort Ihre Aufmerksamkeit erregen sollte: **TypeScript wurde von Anders Hejlsberg geschaffen**, demselben brillanten Ingenieur, der uns Turbo Pascal und Delphi gab. Als ich das erfuhr, machte alles Sinn.

Denken Sie darüber nach. Das Strong Typing, das wir in Delphi lieben? Es ist da. Interfaces? Check. Generics? Ja. Classes und Inheritance? Absolut. Anders nahm alles, was Delphis Type System robust machte, und brachte es in das JavaScript Ecosystem.

Das ist kein Zufall. Das ist ein Meisterhandwerker, der drei Jahrzehnte Sprachdesign-Erfahrung anwendet, um moderne Probleme zu lösen. Wenn Sie jemals Delphis Type Safety geschätzt haben, die Bugs zur Compile Time abfängt, werden Sie sich mit TypeScript sofort heimisch fühlen.

Die wirtschaftliche Realität, über die Niemand spricht

Lassen Sie uns über Entwicklungskosten sprechen. Ich bin ein Geschäftsinhaber, und das sind viele von Ihnen auch. Hier ist etwas Bemerkenswertes über das moderne Web Development Ecosystem:

Der komplette Development Stack ist kostenlos verfügbar

- **TypeScript**: Kostenlos, Open-Source Sprache mit Enterprise-Grade Tooling
- **React**: Kostenlos, unterstützt von Meta mit massivem Community Support
- **Next.js**: Kostenlos, Production-Ready Framework, das von Fortune 500 Unternehmen verwendet wird

- **Node.js**: Kostenlose Runtime mit einem enormen Ecosystem
- **Visual Studio Code**: Kostenlose, Professional-Grade IDE
- **Alle wesentlichen Libraries und Tools**: Verfügbar ohne Lizenzkosten

Optionale Verbesserungen

- **AI Coding Assistants**: Verfügbar für diejenigen, die sie wollen
- **Premium Hosting**: Viele exzellente kostenlose Tiers verfügbar
- **Advanced Tooling**: Größtenteils kostenlos, mit Premium-Optionen falls gewünscht

Der gesamte Development Stack fügt nichts zu Ihren Pro-Entwickler-Kosten hinzu.

Das bedeutet, Sie können die Fähigkeiten Ihres Teams erweitern und neue Marktmöglichkeiten erkunden ohne zusätzliche Lizenzierungskosten. Ihr Development Budget kann sich auf das konzentrieren, was am wichtigsten ist: Talent, Infrastruktur und Geschäftswachstum.

Aber hier ist das, was wirklich für Ihr Geschäft zählt: Ihre Kunden kümmert es nicht, womit Sie entwickeln. Sie kümmert es, dass ihre Anwendung auf ihrem Telefon, ihrem Tablet und ihrem Laptop funktioniert, ohne etwas installieren zu müssen.

Die Web-Realität: Warum Delphi hier kämpft

Delphi kann technisch Web Development machen. Wir haben WebBroker, WebStencils und verschiedene Frameworks. Aber seien wir ehrlich zueinander – sie fühlen sich an, als würden wir versuchen, ein Desktop Framework etwas tun zu lassen, wofür es nicht wirklich entworfen wurde. Es ist, als würde man einen Formel-1-Wagen verwenden, um ein Feld zu pflügen. Sicher, er kann einen Pflug ziehen, aber dafür wurde er nicht gebaut.

Next.js wurde von Grund auf für moderne Web-Anwendungen entworfen. Seiten laden sofort. Sie funktionieren auf Telefonen, Tablets und Desktops ohne unterschiedlichen Code. Search Engines können Ihren Content finden. Benutzer installieren nichts – sie klicken einfach einen Link.

Ihre Delphi Desktop-Anwendung mag perfekt sein. Aber wenn Ihr größter Kunde sagt "wir brauchen das, um auf jedem Gerät mit einem Web Browser für unser Sales Team zu funktionieren, ohne es auf jedem Gerät installieren zu müssen," was sagen Sie ihm?

Ein vertrautes Beispiel: Strong Typing

Das sollte Ihnen Vertrauen geben. In Delphi schreiben Sie:

```
type
  TCustomer = class
  private
    FName: string;
    FEmail: string;
  public
    property Name: string read FName write FName;
    property Email: string read FEmail write FEmail;
  end;
```

In TypeScript sieht das bemerkenswert ähnlich aus:

```
interface Customer {
  name: string;
  email: string;
}
```

Wenn Sie versuchen, den falschen Type zuzuweisen, fängt TypeScript das ab, bevor Sie den Code überhaupt ausführen. Genau wie Delphi. Die gleiche Compile-Time Safety, auf die Sie sich jahrzehntelang verlassen haben.

TypeScript ist keine völlig andere Welt. Es ist eine vertraute Welt mit einem anderen Akzent.

Warum Sie diese Serie lesen

Ich werde nicht so tun, als würde diese Serie Sie zu einem Next.js-Experten machen. Das wird sie nicht. Was sie tun wird, ist Ihnen zu zeigen, dass der Übergang möglich ist, dass Ihr Delphi-Wissen übertragbar ist, und dass es einen Weg nach vorn für Ihr Geschäft gibt.

Hier ist die Realität: **Die meisten Delphi-Shops benötigen Hilfe bei diesem Übergang**. Sie haben:

- 10, 15, 20 Jahre Business Logic in Delphi Code
- Anwendungen, die perfekt funktionieren und Geld verdienen
- Kunden, die nach Web- und Mobile-Versionen fragen
- Ein Team, das Delphi in- und auswendig kennt, aber nie JavaScript berührt hat

Sie können nicht einfach alles stoppen und von Grund auf neu schreiben. Sie brauchen eine Strategie. Sie müssen verstehen, was möglich ist. Sie müssen wissen, was in Delphi bleiben und was ins Web wechseln soll.

Hier kommt professionelles Consulting ins Spiel. Über diese Serie werde ich Ihnen genug zeigen, um die

Landschaft zu verstehen. Wenn Sie bereit sind, tatsächlich den Schritt zu machen – wenn Sie ein Kundenprojekt haben, das Web-Zugang erfordert, oder wenn Sie Ihre nächste Hauptversion planen – werden Sie erfahrene Hilfe wollen.

Ich helfe Delphi-Entwicklungsshops dabei, ihre Anwendungen zu modernen Web-Plattformen zu migrieren. Nicht indem ich Ihre Investition wegwerfe, sondern indem ich:

- Ihre bestehende Delphi-Anwendungsarchitektur analysiere
- Identifiziere, was ins Web wechseln und was Desktop bleiben sollte
- Einen realistischen Migrationsplan erstelle, der Ihr Geschäft nicht zum Stillstand bringt
- Ihre Delphi-Entwickler in TypeScript und Next.js schule
- Das initiale Framework aufbaue, damit Ihr Team übernehmen kann
- Sicherstelle, dass Ihr Data Layer reibungslos übergeht

Es geht nicht darum, Delphi aufzugeben. Es geht darum, zu erweitern, was Sie Ihren Kunden anbieten kön-

nen. Desktop, wo es Sinn macht. Web, wo sie es brauchen. Ihr Code, Ihre Business Logic, Ihr Wettbewerbsvorteil – erhalten und erweitert.

Wenn Sie das lesen und denken "Ich muss dieses Gespräch über unsere Anwendungen führen," melden Sie sich. Auch wenn Sie noch nicht bereit sind, ein Projekt zu starten, bespreche ich gerne Ihre spezifische Situation und wie eine Migration für Ihr Geschäft aussehen könnte.

Was diese Serie besprechen wird

Über die nächsten vier Artikel werden wir die Konzepte erkunden, die wichtig sind:

Teil 2: TypeScript für Delphi-Entwickler

- Warum sich das Type System so vertraut anfühlt
- Wie Interfaces und Classes übertragen werden
- Was anders ist (und was besser ist)
- Die Kernkonzepte verstehen, ohne in die Implementierung einzutauchen

Teil 3: React Components verstehen

- Components als Bausteine (wie VCL Controls)
- Das Konzept von Props und State
- Wie Event Handling funktioniert
- Warum dieser Ansatz für Web-Anwendungen funktioniert

Teil 4: Das Next.js Application Model

- Wie Projekte strukturiert sind
- Routing und Navigation Konzepte
- Wo Ihre Business Logic lebt
- Wie Databases integriert werden

Teil 5: Migration Strategy und Planung

- Bewertung Ihrer Delphi-Anwendung
- Was ins Web wechselt, was Desktop bleibt
- Planung eines realistischen Übergangs
- Wann professionelle Hilfe hinzugezogen werden sollte

Jeder Teil wird Konzepte mit dem vergleichen, was Sie in Delphi kennen. Das Ziel ist nicht, Ihnen beizubringen, Next.js zu programmieren – das Ziel ist, Ihnen zu helfen zu verstehen, ob das richtig für Ihr Geschäft ist und wie der Migrationspfad aussieht.

Das Fazit für Ihr Geschäft

Sie haben Jahrzehnte in Delphi investiert, und diese Expertise verschwindet nicht. Ihre Anwendungen funktionieren. Ihre Kunden sind zufrieden. Aber der Markt verändert sich, und Sie hören Anfragen, für die Delphi nicht entworfen wurde.

Hier ist, was Sie wissen müssen:

Ihr Delphi-Hintergrund ist ein Vorteil. Sie verstehen bereits Strong Typing, Component Architecture,

Event-Driven Programming und Database Operations. Diese Konzepte ändern sich nicht – nur die Syntax tut es.

Der Übergang ist machbar. Nicht trivial, aber machbar. Mit der richtigen Anleitung kann Ihr Team diese

Technologien lernen. Noch wichtiger, Sie müssen nicht alles auf einmal migrieren.

Der Business Case ist überzeugend. Niedrigere Kosten, größere Reichweite, moderne User Experience und

die Fähigkeit, "ja" zu sagen, wenn Kunden nach Web- oder Mobile-Zugang fragen.

Sie müssen es nicht allein machen. Viele Delphi-Shops haben diesen Übergang erfolgreich mit professioneller Anleitung geschafft. Der Schlüssel ist eine klare Strategie und realistische Erwartungen zu haben.

Nächste Schritte

In Teil 2 werden wir TypeScript selbst betrachten. Sie werden sehen, warum Anders Hejlsbergs Design-Entscheidungen sich vertraut anfühlen. Wir werden das Type System erkunden, Interfaces und Classes betrachten und verstehen, warum Delphi-Entwickler das schneller aufgreifen als Entwickler aus dynamischen Sprachen.

Sie müssen noch nichts installieren. Lesen und verstehen Sie einfach die Konzepte. Wenn Sie entscheiden, dass das die Richtung für Ihr Geschäft ist, dann werden Sie praktisch werden wollen – entweder allein oder mit professioneller Hilfe.

Die Reise von Delphi zu Next.js handelt genauso von Business Strategy wie von Technologie. Lassen Sie uns diese Strategie gemeinsam erkunden.

Möchten Sie Ihr spezifisches Migrationsszenario besprechen? Ich arbeite mit Delphi-Entwicklungsteams zusammen, um Übergänge zu modernen Web-Plattformen zu planen und auszuführen. Egal ob Sie Optionen erkunden oder bereit sind, ein Projekt zu starten, lassen Sie uns darüber sprechen, was für Ihre Anwendungen und Ihr Geschäft Sinn macht. Kontaktieren Sie mich!

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 —