NEWS · MAY 17, 2024

Cherish the Past, Embrace the Future: Web development with Delphi and TMS WEB Core

By Dr. Holger Flick

Get ready

This week, I watched several technical sessions from Google I/O and React Conf 2024. Two conferences that contain a plethora of information about the latest trends in web development. It's fascinating to see how the web platform has evolved over the years, from the early days of static HTML pages to the modern, dynamic web applications we have today.

As a Delphi developer, I couldn't help but reflect on the evolution of web development and how it intersects with the world of Delphi. While Delphi is primarily known for its strength in building native desktop applications, the rise of web technologies has opened up new possibilities for developers. One such technology that bridges the gap between Delphi and web development is TMS WEB Core.

The difficult learning curve of TMS WEB Core...

Learning TMS WEB Core is a journey that requires time, patience, and dedication. It's a different paradigm from traditional Delphi development, with its own set of challenges and best practices. However, the rewards are well worth the effort. By mastering TMS WEB Core, you can leverage the power of web technologies to create modern, responsive web applications that run in any modern browser.

The tease

During both Google I/O and React Conf, I saw interesting frameworks that had on first sight a lower learning curve than TMS WEB Core. The features shown were amazing. And the best of all: All these tools are free to use. Even for commercial use, there are no royalties to pay.

A new language?

The downside of these frameworks is that they require you to learn a new language. While this can be a barrier for some developers, it's also an opportunity to expand your skill set and explore new horizons. Still, it's a significant investment in time and effort to become proficient in a new language and framework.

JavaScript

With TypeScript, there is a language that is very similar to Object Pascal. However, it is not as easy to implement into your development process as Delphi is. The tooling is not as mature as the Delphi IDE, and the learning curve is steeper than with Delphi. Most tools have to be set up using the command line with dozens of parameters.

The tooling

The tooling is free yet again. Wonderful. However, it is nowhere near as straightforward as the Delphi IDE. The Delphi IDE is a one-stop-shop for all your development needs. You have everything you need at your fingertips, from designing your user interface to writing code, debugging, and deploying your application. With these new frameworks, you have to set up your development environment from scratch, install various tools and dependencies, and configure everything to work together. It's a more complex and time-consuming process that can be daunting for beginners. Also, learning a package manager without having any GUI to guide you through the process is a challenge.

The puzzle pieces that build your development environment

Unlike Delphi, you customize your development environment in React or NextJS picking from packages. These packages determine the language you use, the way you write your code, the way you debug your code, and the way you deploy your application. It's like building a puzzle, where each piece has to fit perfectly to create a complete picture. If one piece is missing or doesn't fit, the puzzle won't be complete, and your development environment won't work as expected.

Puzzle

Be careful what you wish for

A criticism I hear often as a Delphi MVP is that Delphi releases take a long time. However, the Delphi IDE is a mature product that has been refined over decades. It's a stable, reliable, and feature-rich development environment that provides everything you need to build high-quality applications. While it may not have all the

latest bells and whistles of the newest web development frameworks, it offers a level of productivity and ease of use that is hard to match.

To use the bells-and-whistles of a new framework, all the other pieces have to be in place as well. Let me give you an example. I watched a session about new features in React 19. Awesome! I would like to try them. However, to use these features, I have to upgrade my development environment. This means I have to upgrade my NodeJS version, my package manager, the component packages, the language packages, the framework.... This is a significant investment in time and effort that may not pay off in the end.

Here's my list of packages in a two-page web project written in NextJS:

```
"dependencies": {
    "@emotion/react": "^11.11.4",
    "@emotion/styled": "^11.11.5",
    "@mui/icons-material": "^5.15.16",
    "@mui/material": "^5.15.16",
    "better-sqlite3": "^9.6.0",
    "jquery": "^3.7.1",
    "lightbox2": "^2.11.4",
    "next": "^14.2.3",
    "react": "^18.3.1",
    "react-dom": "^18.3.1"
  },
  "devDependencies": {
    "daisyui": "^4.10.5",
    "eslint": "^8",
    "eslint-config-next": "14.2.3",
    "postcss": "^8",
    "tailwindcss": "^3.4.1"
  }
```

In order to upgrade to the latest version of React that was shown at the conference, I need to lift my version of React to 19.0.0. This means I have to upgrade all the other packages as well.

The process should be automatic... however, I have spent about 2 hours to get everything working again. I was not able to enjoy any of the new features because most of the other components were incompatible with the new version of React. I had to downgrade React to 18.3.1 to get everything working again. Further, I could have investigated each and every package used to find out if a beta version compatible with React 19 was available. This would have taken even more time.

The result

- I never was able to use any of the new framework features.
- I lost 2 hours of development time.
- I have to test everything again to make sure it works as expected. Thankfully, I use version control and it is one click.
- I am only able to create a new project with the new features. I can't upgrade my existing projects because of the breaking changes.
- There's no "company" to ask for support. I have to rely on the community to help me out.
- As the community is very active, I would most likely have to upgrade my project every 3 months to keep up with the latest features.

The conclusion

While it's exciting to explore new technologies and frameworks, it's essential to consider the trade-offs. The latest and greatest tools may offer cutting-edge features and performance, but they also come with a learning curve, complexity, and maintenance overhead that can be challenging to manage. Delphi, on the other hand, provides

a stable, reliable, and feature-rich development environment that enables you to focus on building high-quality applications without getting bogged down in the details of setting up and maintaining your development environment.

The solution

Try TMS WEB Core today!

<figure markdown> ![TMS Software](https://www.tmssoftware.com) <figcaption></figcaption> </figure>

The documentation is extensive, and there are many examples to get you started. The learning curve is steep, but the rewards are well worth the effort. I also wrote a book about TMS WEB Core that guides you through the process of building web applications with Delphi. Find more information about it here.

If you need help

I offer consulting services to help you get started with TMS WEB Core. Contact me for more information.

Watch videos

There's a vast quantity of videos available on the TMS Software YouTube channel. Check them out here.

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 —