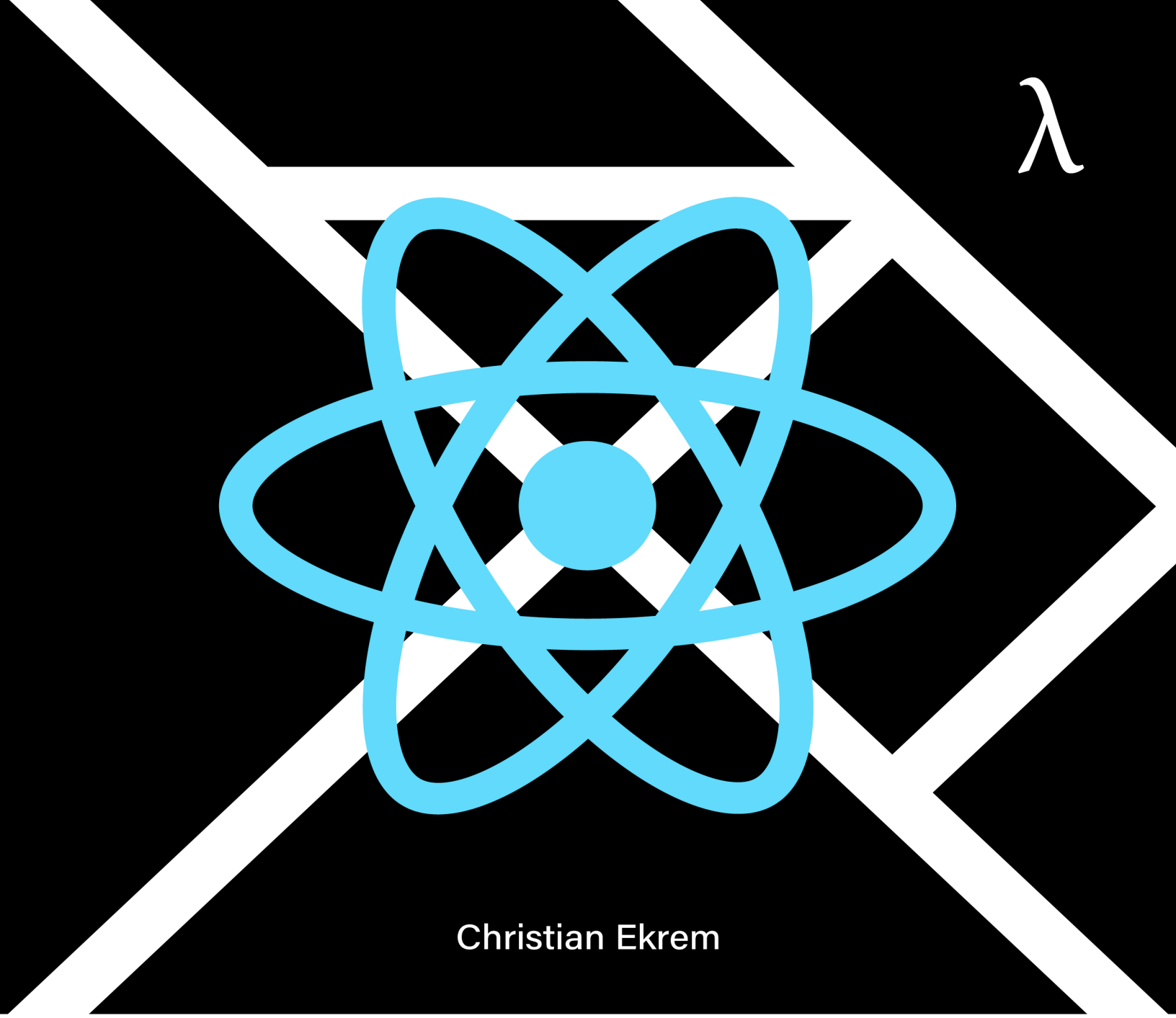

An Elm Primer for React Developers

The Best Way to Learn Real Functional Programming

An Elm Primer for React Developers

The Best Way to Learn Real Functional Programming

Christian Ekrem

This book is available at https://leanpub.com/elm-for-react-devs

This version was published on 2026-05-29

Contents

Praise for An Elm Primer for React Developers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Foreword

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Introduction

If you've spent any time at all using React, chances are you've felt both the library's power and its complexity. Hooks are flexible, TypeScript catches many errors. But runtime surprises still slip through. `useEffect` dependencies are easy to mess up, performance bugs eventually demand strategic (and specific!) `React.memo` and `useCallback` usage, and what started out harmlessly enough easily gets out of hand. I don't think these are signs you're doing React wrong; they're the natural trade-offs of React's flexibility.

Some of my most-read posts are the deep-dive React articles (especially on `React.memo` and reconciliation). A few are used in React courses, and some have even been translated to Korean(!). This popularity reflects a truth: once you step beyond the basics, React's complexity increases dramatically and its benefits can quickly be overshadowed by significant drawbacks.

Elm takes a different approach. It's a *small functional language with one way to build applications*, designed for reliability and clarity. With Elm, you trade flexibility for focus. There's exactly one architecture (The Elm Architecture, or TEA), exactly one way to represent side effects, and a type system that works hard to prevent runtime errors. The result is a developer experience that feels, at least at first, strangely *quiet*. You don't spend hours debugging why your component didn't re-render or why state mutated unexpectedly. Instead, you spend time *modeling* your problem domain.

For React developers, the first encounter with Elm can feel almost too simple. There are no hooks to juggle, no contexts to configure, no class vs. function components. Just a `Model`, an `Update` function, and a `View`. And yet, that simplicity scales to codebases with hundreds of thousands of lines.

So why consider Elm if you already know React?

- **To catch more errors at compile time**. Many React/TypeScript errors show up only in the browser. Elm's compiler is famously strict: if it compiles, it's very likely to work.
- **To simplify mental overhead**. Fewer concepts mean fewer ways to get things wrong. Instead of choosing between a dozen state management solutions, Elm gives you one–TEA.
- **Because it's actually fun**. Elm is genuinely enjoyable to work with. The compiler points you toward fixes, the language encourages clear modeling, and your app tends to just work once it compiles.

But there's a deeper reason to learn Elm, even if you never use it professionally: **it's the fastest way to truly learn functional programming**. Real functional programming, the kind that actually changes how you think about code in any language. Haskell has too many language features and quirks, making it notoriously hard to get started with something practical. Elm, by contrast, is remarkably small and focused (the entire language fits in your head). And crucially, you're working

in a domain you already know: building web UIs. Strict FP discipline plus a domain you already know–that's a surprisingly good combo for actually learning this stuff.

If you've ever wanted to grow as a developer by learning functional programming, Elm is in my honest opinion your shortest path. By far! Especially if you're already familiar with React.

This book is a guided tour for React developers: what Elm looks like, how it compares, and how you might start using Elm–even if only as a single widget inside a React app. By the end, you should have a feel for whether Elm can play a role in your work. Better yet: you will have learned functional programming along the way! Whether you end up actually abandoning React is beside the point (and probably not all up to you).

If React has been your main frontend toolkit, Elm shows you what frontend development feels like with different trade-offs. You get to see what it looks like to write *only functional code*, and to keep all side effects at the boundary where they belong. It may not be for every project, but once you've experienced Elm's compiler, you'll wish more of your codebases worked this way.

And learning Elm will be worthwhile whether you end up using it professionally or not. Especially in this day and age, learning something transferable (not just syntax and recipes) that helps us solve real engineering problems is very valuable.

To give you a taste of what I mean, let me show you the kind of guarantees Elm provides. Throughout this book, you'll see what I call "Elm Hooks" in each chapter–quick previews of how Elm handles that chapter's topic. React has hooks for state management, but Elm keeps you hooked on the language itself.

Try this one. It'll demonstrate two of Elm's main selling points: the friendly compiler and the strict type system (or, good cop / bad cop, if you prefer).

Elm Hook

Given the following Elm code

```elm
type Animal = Cat | Dog | Penguin


animalToString : Animal -> String
animalToString animal =
    case animal of
        Dog ->
            "dog"

        Cat ->
            "cat"

        -- Ooops, forgot about that Penguin!
```

The compiler answers the following:

```
-- MISSING PATTERNS --------------- /path-to-your/src/Main.elm
This `case` does not have branches for all possibilities:

306|>    case animal of
307|>        Dog ->
308|>            "dog"
309|>
310|>        Cat ->
311|>            "cat"

Missing possibilities include:

    Penguin

I would have to crash if I saw one of those. Add branches for them!

Hint: If you want to write the code for each branch later, use `Debug.todo` as a
placeholder. Read <https://elm-lang.org/0.19.1/missing-patterns> for more
guidance on this workflow.
```

Sure beats all those `ensureNever` helpers[1] you've written in TypeScript, huh?

[1] In TypeScript, `ensureNever` (sometimes called `assertNever` or `exhaustive`) is a helper function that takes a value of type `never` and throws if it's ever called. Developers add it as the default case in switch statements to get a compile error when a new union member is added but not handled. The fact that you need a helper function for this tells you something about the difference between opt-in and built-in exhaustiveness checking.

Acknowledgements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Part I: From React to Elm: Getting Started

Chapter 1: Elm: Constraints That Free You

Remember that `Penguin` example from the introduction? That kind of exhaustive checking runs through everything Elm does. Let me show you what this means for everyday development.

Let's say you refactor a type. Say you rename a field from `status` to `orderStatus` (not an unreasonable thing to do). You update the code, run your tests, and ship it, thinking all is well and good. A week later, though, production errors start rolling in: `Cannot read property 'status' of undefined`.

You missed *one* usage. It was buried in an error handler that only runs when a specific edge case triggers. Even with `strict: true` and no `any` types anywhere in the project, TypeScript didn't catch it. Maybe the type was widened through a generic callback, or the old field name was still valid on the structural type. The code compiled without a single red squiggly. Your tests didn't catch it because they didn't quite cover that exact scenario. ESLint was silent because the code was syntactically fine, as far as it could tell.

I'm sure you've experienced something similar. And this is not about being a bad developer or having a sloppy team. You ran the tests, you checked and double-checked your work, and even had that one colleague who's extra strict do the code review! But TypeScript's type safety is opt-in at every layer: you choose to enable strict mode, you choose to avoid type assertions, you choose to handle every case in a switch statement. In a system where soundness requires discipline at every boundary, gaps are a matter of *when*, not *if*. You rely on tools to catch what you forget, but these tools have structural limits baked into their design.

React Recommends, Elm *Requires* and Enables

React and Elm are essentially heading in the same direction, but taking quite different paths to get there.

Look at how React has evolved:

- Hooks moved us toward functional components and immutable state
- Redux brought predictable state management to the mainstream (Bonus point: Redux was actually inspired by Elm!)
- TypeScript went from optional to essential for most serious projects
- Server Components push side effects to the server

Each change pushes React toward functional programming principles. Immutability, pure functions, explicit state management–the React community now considers all of these best practices. No React developer worth their salt has side effects in their `map` or `reduce`, after all! Right?

I've heard more than one React developer say something like: "Good React code in 2026 looks suspiciously like Elm code from 2016."

React recommends functional programming. **Elm requires it**. And the constraints that feel limiting at first? They're what allow you to stop chasing bugs through state mutations and start solving the actual problem.

In React, you can still mutate variables, mix paradigms, create runtime errors. The community discourages it; JavaScript can't stop you.

In Elm, these things are simply impossible[1]. The language won't compile if you try to mutate data. And there's no such thing as throwing an exception–the compiler rejects `Debug.todo` when you build with the `--optimize` flag, so it can't sneak into production.

React's flexibility is a genuine strength. But it comes with a cost: you have to maintain the discipline yourself, and it gets harder as your codebase scales.

When Constraints Give Freedom

This comes up all the time when debugging:

```
1  const user = { name: "Ada", age: 29 };
2  someFunction(user);
3  console.log(user.name); // What's the name now?
```

The answer is simple, and annoying: you can't know without reading `someFunction`. Maybe it mutates the user. Maybe it doesn't. Maybe it mutates it conditionally. Maybe you never get to the console.log at all because of a runtime exception. You have to trace through the code to be sure. TypeScript's `Readonly<T>` helps at the surface level, but deep immutability requires recursive utility types and you basically have to stay on guard all the time (and `as` casts can always punch a hole through your defenses).

I've spent hours debugging issues where data was mutated in unexpected places. A function I thought was safe was actually changing my state without my knowing, and the resulting bug only appeared in specific conditions (and my tests didn't catch it).

Now look at the Elm equivalent:

[1]To be 100% accurate, you *could* generate a runtime error by triggering an underlying JavaScript divide-by-zero exception. An actual division by zero in Elm is not enough, though; you'd have to do `remainderBy 0 {whateverNumber}` or `modBy 0 {whateverNumber}`. And it's technically possible to get a Stack Overflow if you (mis-)use extreme recursion without tail call optimization.

```elm
user = { name = "Ada", age = 29 }

-- This isn't valid Elm syntax, whether inline or in a function:
user.name = "Grace"  -- ERROR: won't compile

-- The right way:
updatedUser = { user | name = "Grace" }  -- Creates a new record
```

The compiler makes mutation impossible. When you pass `user` to a function, you know it comes back unchanged. Not because you trust the function author (that would be too brittle) but because the language prevents it. All mutation. If you've used Rust, you know the distinction between a variable and a mutable variable matters. In Elm, they're *all* immutable.

You stop wondering "who changed this value?" because nothing can. That question simply doesn't exist in Elm.

Debugging Gets Boring (In a Good Way)

In React, when state is wrong, you trace backward: where was this set? What changed it? Did something mutate it accidentally?

In Elm, when state is wrong, you look at your `update` function. That's it. That's the only place state changes. If the state is wrong, the logic in `update` is wrong. No hidden mutations or stale closures. No wondering if some other component changed something, because it couldn't possibly do so.

At my client's production app (a 150k+ lines Elm codebase), we've had entire months with zero runtime exceptions originating from our Elm code[2].

Though the developers on my team are great, that's not the main reason. I'm quite confident this is it: *the compiler catches those errors before the code runs!*

Elm Hook

You know that moment when you refactor a type but forget to update one place that uses it? In Elm, you literally cannot compile until you fix every single usage. The compiler won't let you ship the incomplete refactor. It's impossible to "forget one spot."

Refactoring with Confidence

All of this pays off most when you refactor.

Say you need to add a new state to your application–a `Paused` state for a game, or a `Refreshing` state for data loading. In React, you'd:

[2]JavaScript code communicating through ports can still throw exceptions on its side, though these won't crash the Elm runtime itself. When I say "zero runtime exceptions," I mean zero from the Elm code.

1. Add `'paused'` to your TypeScript union type
2. Search the codebase for places that check the state
3. Update each one, hoping you found them all
4. Test manually, hoping you caught the edge cases
5. Ship and monitor for bugs
6. Hope and/or pray?

(To be fair: TypeScript's discriminated unions help here. But they're opt-in, and escape hatches like `any` and type assertions mean you can never be fully sure you caught everything.)

In Elm, you:

1. Add `Paused` to your union type
2. Try to compile
3. The compiler lists every place that needs updating
4. Fix each one until compiler gives you a green light
5. Deploy

I refactored a complex state machine recently–47 places needed updates. The compiler found all 47. I fixed them one by one. When the code compiled, I deployed it. No bugs or forgotten edge cases. The compiler had verified completeness, and that was it.

This isn't a guarantee of correctness (I can still have logic bugs!) but it *is* a guarantee that every code path handles every state. No gaps. No undefined behavior at runtime.

Architecture You Don't Have to Enforce

If you've read about Clean Architecture or SOLID principles, you know the ideas are sound. Single Responsibility, Dependency Inversion, separation of concerns. Good stuff, and I'm all for it!

But they're also *discipline*. You have to remember to follow them. Code review has to catch violations, and it's way too easy to cut corners when you're rushing.

The Elm Architecture enforces these patterns by default. Not as guidelines, but as requirements. The architecture separates View, Update, and Model–`Browser.sandbox` and `Browser.element` require them as separate functions, so Single Responsibility is just how the code works. Mutation is impossible and side effects go through `Cmd`, which means your functions stay pure whether you're thinking about it or not. And pattern matching forces you to handle every case, so exhaustive handling comes for free.

Where other languages offer SOLID (or whatever acronyms float their boat[3]) as "best practices" you *should* follow if you're disciplined, in Elm they're mandatory. The compiler enforces what code review can't.

[3]Bonus point: you might also have to argue with your peers that following SOLID in particular or being mindful of architecture in general is even worth it, especially in frontend projects. (It is, btw, but the point is not everyone agrees.)

What This Costs You

Elm's strictness has real costs.

The ecosystem is smaller. React has thousands of libraries. Elm has hundreds. You'll find yourself writing more from scratch.

The learning curve is steeper. Functional programming is different if you're coming from JavaScript, and concepts like pattern matching and union types take time to internalize.

Your team needs to learn. Hiring is harder. Onboarding takes longer. Not every developer wants to learn a niche language.

You lose flexibility. Sometimes you just want to mutate a value and move on. Elm won't let you. You have to do it the "right" way, even when the shortcut would probably work.

The language hasn't had a release since 2019. That can look alarming at first glance, but in practice it means remarkable stability. Your Elm code from five years ago still compiles and runs without changes. Whether you see this as a cost or a feature depends on your perspective, but you should know about it going in.

For some projects, these costs aren't worth it. If you're prototyping, exploring, or building something simple, React's flexibility is valuable. You want to move fast, not satisfy a strict compiler. Doubly so if React is already in your bones: when you need speed, familiar tools win. And honestly, for most projects today, React is still the pragmatic choice, and that's perfectly fine.

But for other projects (production applications where bugs are expensive, complex state machines, financial tools, healthcare systems) Elm's guarantees are worth the upfront cost.

And, to be completely honest: At this point I personally prefer Elm even for the occasional whimsical side-project that won't hurt a fly no matter how hard it crashes. As you'll hopefully discover for yourself before too long: Elm is kind of addictive, and fun to work with!

What Elm Teaches You

The best thing about learning Elm is what it does to the rest of your code.

After spending time with Elm, mutable state starts looking suspicious. You begin designing types that prevent bugs instead of just documenting intent. You write better React, better TypeScript, better everything–because the functional thinking becomes yours, not just something a linter enforces.

I genuinely think Elm is the fastest route into functional programming. Not because it teaches you monads and functors[4], but because it makes functional programming impossible to avoid. Try to

[4] Elm never even mentions monads or functors by those names. You'll use them (`Maybe.andThen` is monadic bind) but the language doesn't make you learn the vocabulary to get work done.

mutate a variable? Compiler says no. Try to ignore a case? Compiler says no. Try to sneak in a side effect? Nope. You can't cheat your way around it, so you learn to think functionally.

Compare this to learning FP through Haskell or OCaml. Those languages are powerful, but they're also large and complex. Haskell has lazy evaluation, type classes, monad transformers, dozens of language extensions. By the time you understand enough to build something useful, months have passed.

Elm's syntax fits in your head in a weekend. No classes. No inheritance. No async/await, no promises, no null, no undefined. Just functions, types, and one architecture pattern. That smallness is exactly why you pick it up fast.

And you're building in a domain you already know. If you're coming from React, you understand components, events, state updates, and you're familiar with the DOM. Elm uses different mechanics, but the same concepts. The output is still web, so you can focus on learning functional programming instead of a whole new platform.

Even if you never use Elm professionally, the habits stick. Modeling with types, making illegal states unrepresentable, treating data as immutable–you carry these with you into any language. And if you eventually pick up Haskell, F#, or OCaml, you'll find you already understand the core concepts.

Enough philosophy. Next chapter, we write the same app in React and Elm, side by side.

Chapter 2: The Elm Architecture – A Recipe for Reliable Apps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Recipe: Four Simple Ingredients

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Ingredient 1: Model - Your State Shape

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Ingredient 2: Msg - Things That Can Happen

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Ingredient 3: update - How State Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Ingredient 4: view - Rendering Your State

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Putting It All Together: Counter with Undo

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Messages Describe All Possible Actions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Update Handles Each Case Explicitly

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

View Renders Based on Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Elm Runtime Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What React Developers Already Know

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What Makes TEA Different

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Price of Explicitness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What You Just Learned (The FP Hiding in Plain Sight)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What's Next

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 3: Your First Elm App

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Installing Elm

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Editor Setup

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Setting Up Your Project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What You Just Created

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The obligatory “Hello, world!”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Two Flavors of main

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What's a Program?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The LGTM Generator: Building It Step by Step

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 1: Model First

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 2: Defining What Can Happen

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 3: Implementing Update

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 4: Building the View

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Complete Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Building Your First Elm App

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Compiler-Driven Development in Action

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What You Just Built

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 4: Starting Small: Elm in Your React Codebase

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Widget-by-widget Incremental Adoption Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Integrating One Elm Component into React

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Build Systems and Toolchain Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The StrictMode Gotcha, or "How Not To Render Your Elm Widget Twice"

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Manual Setup

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Scaling This Approach

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

A Real Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Making It Work

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Part II: The Outside World: Randomness, APIs, and JavaScript

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 5: Commands and Randomness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Upgrading from Browser.sandbox to Browser.element

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 1: The Mechanical Refactor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Cmd Type and How It Works

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Adding Randomness to the LGTM Generator

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Building a Random Generator

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Adding a New Message

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Using Random.generate

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Complete Picture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What You Just Learned

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What This Means in Practice

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 6: HTTP and Remote Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

From Random.generate to Http.get

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Setting Up the Server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Installing elm/http

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Shape of HTTP in Elm

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Success | Loading | Error

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Creating the HTTP Request

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Updating init

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Refactoring the GotPhrase Message

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Handling Results in update

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Rendering Different States

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Complete Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Commands Are Just Commands

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What You Just Learned

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 7: JSON Decoders and Type Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Why JSON Needs Decoding in Elm

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Example: Optional fields -> Maybe

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Example: Nested objects and lists

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Example: Reaching deep with `Decode.at`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Supporting JSON in Our LGTM Generator

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 1: Decode Just the Phrase

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 2: Decode the Full Payload

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What Just Happened?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What About Sending JSON?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What You Just Learned

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Further Reading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 8: JavaScript Interop: Ports and Flags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

JavaScript as Infrastructure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Upgrading Our Build for Manual Bootstrapping

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Flags as Program Input

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

A Simple Flag Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Optional Flags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Common Flag Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Ports for Communicating with JavaScript

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Adding Clipboard Support to the LGTM Generator

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step one: adding an outbound port (Elm -> JS) carrying a String

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step two: mapping the `ClickedCopy` `Msg` to our outbound port

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step three: receiving messages in JavaScript land

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step four: sending messages from JavaScript to Elm

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What You Just Learned

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Part III: Building Real Applications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 9: Organizing Files and Modules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Moving Beyond the Single-File Approach?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

JavaScript Pain Point I: “Sneaky Mutations”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

JavaScript Pain Point II: Refactoring is Hard and Quite Dangerous

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Elm is Different

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Let’s Build an Advent Calendar App

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Basic Advent Calendar Spec

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Starting Simple: The Mock Date Version

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Properly Typed Business Logic

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The View Layer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Getting the Real Date: Enter Tasks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What is a Task?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Changes We Made

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Building the Task

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Result

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

When to Split: Modules for Data Structures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Freedom of Safe Refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What We've Learned

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 10: Modules Around Data Structures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Opaque Types: Hiding What Shouldn't Be Seen

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Smart Constructors: The Only Way In

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Working with Opaque Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Going Further: Phantom Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Type Parameters That Don't Exist at Runtime

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Marker Type

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Parse, Don't Validate

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Complete Calendar Module

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Using the Calendar Module

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

One Source of Truth

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What We Gained

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Broader Pattern: State Machines in Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

When to Use Each Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Freedom to Reorganize

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What We've Learned

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 11: Forms and Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

A Simple Contact Form

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Messages for User Input

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Update Function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The View

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Putting It Together

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

When Validation Enters the Picture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Model Expands

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Validation Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Update Function Now Validates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The View Gets Cluttered

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Boilerplate Inventory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Questions Worth Asking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Finding the Right Abstraction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What We Want

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Shape of a Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

A Practical Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

How Validators Work

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Composing Validators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Input Transformation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What This Abstraction Trades Away

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Phantom Types in Action

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What We Learned

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 12: Building a Feedback Wizard

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Problem: A Feedback Widget

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Modeling the Top-Level State

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The OutMsg Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Wizard Module

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Handling OutMsg in the Parent

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Why OutMsg Works

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Native Dialogs and Ports

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Complete Main Module

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Api Module

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Patterns Worth Noting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Nested TEA

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

State as Union Type

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

OutMsg for Composition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Subscriptions Follow State

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What We Learned

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 13: Routing and Navigation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

From Element to Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Modeling Routes as Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Parsing URLs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Building URLs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Application Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Handling Navigation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

A Page Module

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Comparing to React Router

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Complete Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Wrapping up

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 14: State Patterns at Scale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

From Booleans to State Machines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Reversible State Machines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

A Second State Machine: Form Lifecycle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

From Type to Module

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Composing Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Shared State

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Underlying Principle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Wrapping Up

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 15: Testing Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

From Jest to elm-test

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

TDD Where It Shines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Round 1: The Empty List

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Round 2: A Single Element

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Round 3: Two Different Elements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Round 4: Consecutive Equal Elements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Round 5: The Full Test

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Fuzz Testing: Let the Framework Think

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What You Don't Need to Test

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Complete Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Wrapping Up

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 16: The Elm Ecosystem for React Developers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Part Where npm Lies to You (And Elm Doesn't)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What's more: No Lockfile Drama

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Trade-Off

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Popular Packages and Community Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Core Packages (`elm/*`)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Community Standards

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Quality Signals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

elm-review

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

When the Ecosystem Doesn't Have What You Need

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Ports

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Custom Elements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Write It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Ask the Community

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Abandoned Package Problem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Potential Deal-Breakers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Wrapping Up

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Chapter 17: Performance in Practice

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

React.memo, useCallback, and Reconciliation Pain Points

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

How Elm's Virtual DOM Makes Optimization Automatic

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Html.Lazy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Performance by Default vs Performance by Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Bundle Sizes and Compilation Targets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Asset Splitting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Lazy Loading and Code Splitting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

When and How to Optimize Elm Applications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 1: Profile, Don't Guess

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 2: Check Your Update Function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 3: Use Html.Lazy for Expensive Views

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 4: Use Html.Keyed for Lists

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Step 5: Consider the DOM Itself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What You Don't Need to Optimize

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

elm-optimize-level-2

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Wrapping Up

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

About the Author

I'm a Staff Engineer with over 12 years of experience building production systems. My recommendations in this book are battle-tested in one of the world's largest Elm codebases–over 150,000 lines of production code at Lovdata, Norway's leading legal information provider.

Before Lovdata, I spent six years at Vipps MobilePay (Norway's leading mobile payment platform) working across backend development in Go, frontend development in React, Android development in Kotlin, and eventually becoming the first engineer to work across both simultaneously. That range of experience shapes my sense of what actually makes code maintainable.

I've also taught at a special-needs high school and worked as a film director. Both of those taught me how to explain things clearly and meet people where they are–which is exactly what this book aims to do for React developers learning Elm.

Why this book exists: I believe Elm is the fastest way for developers to truly learn functional programming. Whether you adopt Elm professionally or not, learning it will make you a better developer in any language. This book puts that experience to work for React developers.

I write about technology and functional programming at cekrem.github.io[1].

> "Architecture!" the man said, and you'll be glad he did.
>
> Christian Ekrem has worked with Lovdata from March 2025 until May 29, 2026, and in that time he has made a strong and lasting impression as a highly experienced full-stack developer. He combines deep technical knowledge with a genuine passion for clean code, thoughtful architecture, and building systems that are robust, maintainable, and easy for others to understand.
>
> Just as importantly, he is an excellent communicator and teacher. He has a rare ability to explain complex technical ideas clearly, bring people along in architectural discussions, and raise the quality of the whole team through patient guidance and constructive feedback. I would warmly recommend him to any organization looking for a skilled, principled, and collaborative developer.
>
> — **Simon Skrede**, Head of Development at Lovdata[2], home to one of the world's largest Elm codebases

[1] https://cekrem.github.io
[2] https://en.wikipedia.org/wiki/Lovdata

Appendix A: Quick Reference Guide

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

React to Elm Concept Mapping

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Common Patterns Cheat Sheet

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Basic Skeleton

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Cmd Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

JSON Decoder Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Maybe and Result

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Port Communication

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

URL Routing Skeleton

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Common List and Dict Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Type Conversions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Troubleshooting Guide for React Developers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

"Why can't I just mutate the model?"

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Missing a Msg variant in update

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Forgetting to wire subscriptions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

JSON decoder failures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Type annotation confusion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

“Why can’t I just call a function with side effects?”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Common compiler error messages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Appendix B: The Scary Words You Already Understand

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Why Elm Avoids This Terminology

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Functors: Things You Can Map Over

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Applicatives: When You Have Multiple Wrapped Values

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Monads: Chaining Operations That Might Fail

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

The Elm to Haskell Rosetta Stone

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Commands and Tasks: Elm's Approach to Effects

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Where to Go From Here

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Appendix C: Further Reading and Resources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Code Examples from This Book

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Essential Elm Learning Resources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

A Reference Elm SPA: dwayne/elm-conduit

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

What to Look For

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

How to Read It

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Community and Getting Help

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.

Advanced Topics Beyond This Book

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elm-for-react-devs.