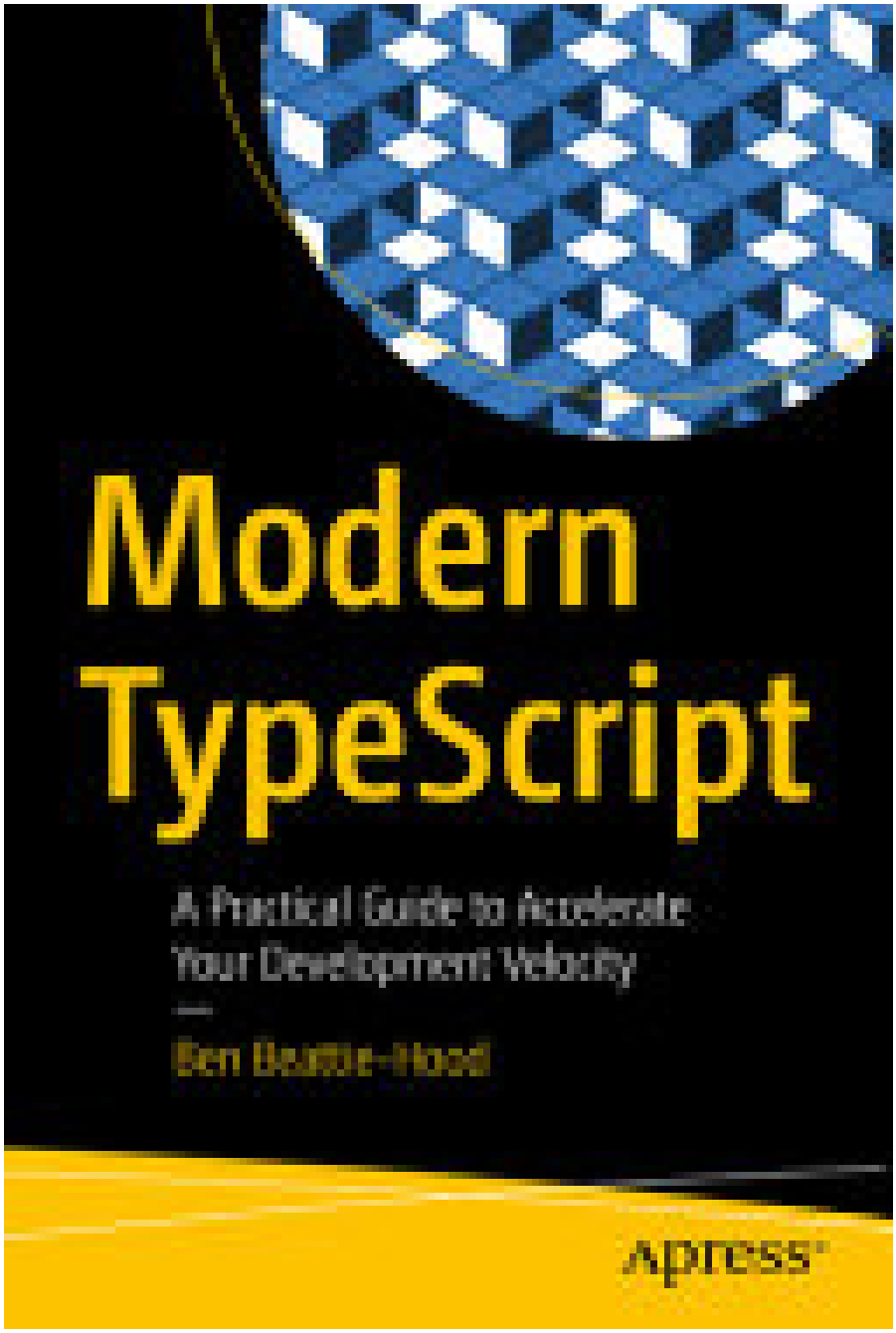
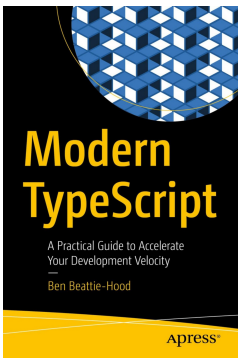
Modern TypeScript: A Practical Guide to Accelerate Your Development Velocity 1st Edition Ben Beattie-Hood

Visit to download the full and correct content document:
https://ebookmeta.com/product/modern-typescript-a-practical-guide-to-accelerate-your-development-velocity-1st-edition-ben-beattie-hood/



More products digital (pdf, epub, mobi) instant download maybe you interests ...

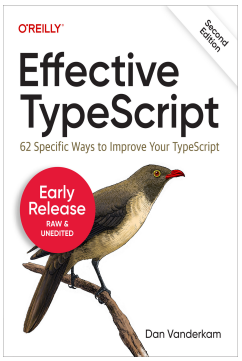


Modern TypeScript 1 / converted Edition Ben Beattie-Hood

https://ebookmeta.com/product/modern-typescript-1-converted-edition-ben-beattie-hood/

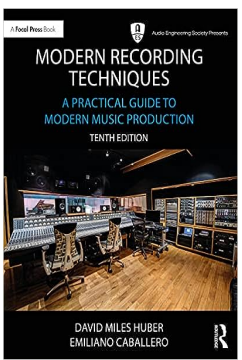
Azure Modern Data Architecture. A Guide to Design and Implement a Modern Data Solutions 1st Edition Ben Zahra

https://ebookmeta.com/product/azure-modern-data-architecture-a-guide-to-design-and-implement-a-modern-data-solutions-1st-edition-ben-zahra/



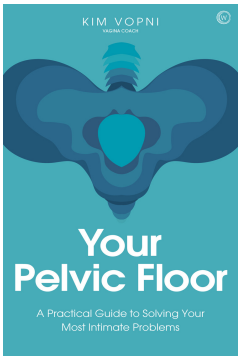
Effective TypeScript: 62 Specific Ways to Improve Your TypeScript, 2nd Edition Dan Vanderkam

https://ebookmeta.com/product/effective-typescript-62-specific-ways-to-improve-your-typescript-2nd-edition-dan-vanderkam/



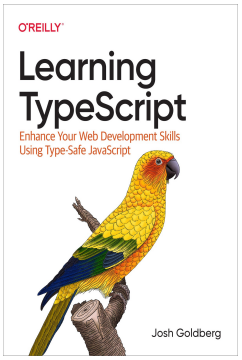
Modern Recording Techniques: A Practical Guide to Modern Music Production, 10th Edition David Miles Huber

https://ebookmeta.com/product/modern-recording-techniques-a-practical-guide-to-modern-music-production-10th-edition-david-miles-huber/



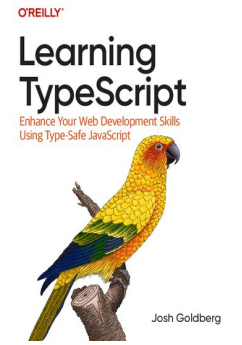
Your Pelvic Floor A Practical Guide to Solving Your Most Intimate Problems 1st Edition Vopni Kim

https://ebookmeta.com/product/your-pelvic-floor-a-practical-guide-to-solving-your-most-intimate-problems-1st-edition-vopni-kim/



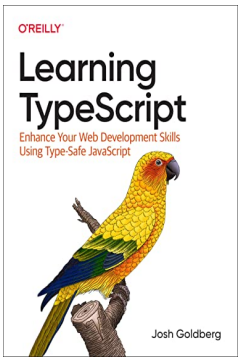
Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript Josh Goldberg

https://ebookmeta.com/product/learning-typescript-enhance-your-web-development-skills-using-type-safe-javascript-josh-goldberg-2/



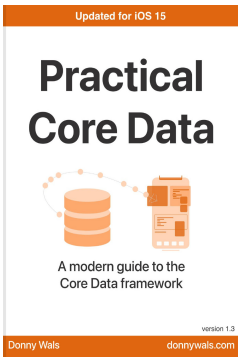
Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript Josh Goldberg

https://ebookmeta.com/product/learning-typescript-enhance-your-web-development-skills-using-type-safe-javascript-josh-goldberg/



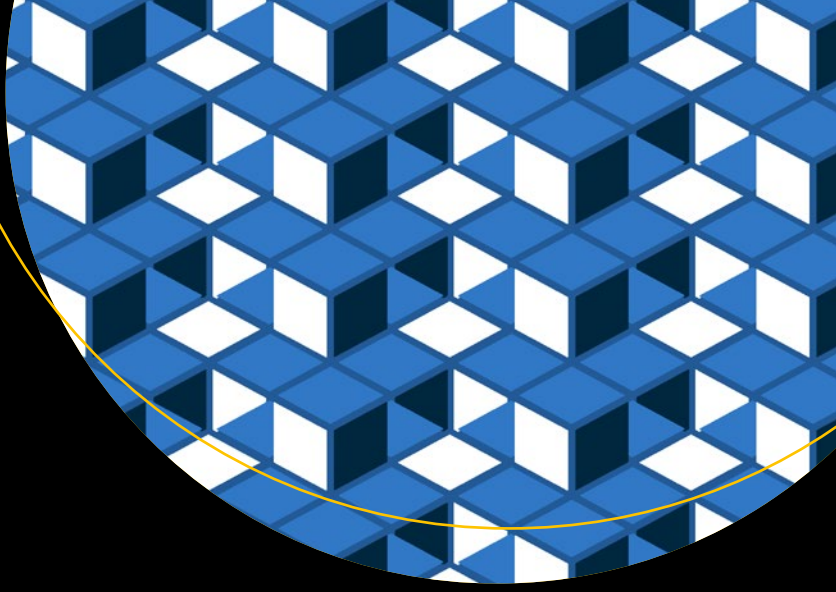
Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript 1st Edition Josh Goldberg

https://ebookmeta.com/product/learning-typescript-enhance-your-web-development-skills-using-type-safe-javascript-1st-edition-josh-goldberg/



Practical Core Data: A modern guide to the Core Data framework Donny Wals

https://ebookmeta.com/product/practical-core-data-a-modern-guide-to-the-core-data-framework-donny-wals/



Modern TypeScript

A Practical Guide to Accelerate Your Development Velocity

Ben Beattie-Hood

Apress®

Modern TypeScript

A Practical Guide to Accelerate Your Development Velocity

Ben Beattie-Hood

Apress®

Modern TypeScript: A Practical Guide to Accelerate Your Development Velocity

Ben Beattie-Hood
Melbourne, VIC, Australia

ISBN-13 (pbk): 978-1-4842-9722-3 ISBN-13 (electronic): 978-1-4842-9723-0
https://doi.org/10.1007/978-1-4842-9723-0

Copyright © 2023 by Ben Beattie-Hood

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: James Robinson-Prior
Development Editor: James Markham
Editorial Assistant: Gryffin Winkler

Cover designed by eStudioCalamar

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub. For more detailed information, please visit https://www.apress.com/gp/services/source-code.

Paper in this product is recyclable

Table of Contents

About the Author

Ben Beattie-Hood is a principal software engineer and professional mentor with over 20 years of industry experience. He currently specializes in front-end technology, technical strategy, system design, and development and training in TypeScript, React, and related technologies.

Ben is passionate about evolvable systems, about creating learning organizations, and about how ideas are formed and communicated. He has given a wide range of talks covering product development, event sourcing microservices, event storming practice, modern database internals, functional programming, front-end design and build, as well as coaching in TypeScript as a velocity tool.

Ben lives with his beautiful wife and two children in Melbourne, Australia, where he loves to hike and travel.

About the Technical Reviewer

Oscar Velandia is a front-end developer at Stahls. He has worked the past four years with Typescript, transitioning projects from Vanilla JavaScript to TypeScript, creating MVP projects, and working on large TypeScript, Next.js, and React code bases.

CHAPTER 1

How TypeScript Came to Be

Introduction

As we start our TypeScript journey, we delve into the background of this powerful programming language. To truly grasp the capabilities of TypeScript, it is essential to understand its roots and the context in which it evolved. We'll take a brief trip through the history of JavaScript and ECMAScript and how these played into the development of TypeScript in late 2012. This brief background will help us get both a clearer sense of the problems TypeScript solves as well as the reasoning behind some of the evolution of TypeScript itself. Understanding these key points, around the need for more scalable velocity, and how testing and docs can be delivered in the simplest possible way to solve this will give us a strong foundation for fully understanding the direction and outcomes of the TypeScript language.

History

To truly understand TypeScript, it is important to understand some of the background of the language. We want to get onto the fun stuff with types, but this background is necessary, so I'll keep it short and to the point.

© Ben Beattie-Hood 2023
B. Beattie-Hood, *Modern TypeScript*, https://doi.org/10.1007/978-1-4842-9723-0_1

Because when looking into TypeScript, you'll undoubtedly come across references to both JavaScript and ECMAScript and their versions – so how do all these fit together? So let's rewind the clock to get a quick overview.

The first version of JavaScript was released in 1995, and it quickly gained popularity among web developers. In 1996, Microsoft introduced their own version of JavaScript called JScript. And so in 1997, JavaScript was submitted to the European Computer Manufacturers Association (ECMA) in an effort to standardize. This resulting standard was called ECMAScript and was released in 1999.

So the important thing to note here is that ECMAScript is a standard, not a language. That means that it defines *how* features like objects, functions, variables, closures, operators, error handling, etc., *all work and interoperate*, but it doesn't define the actual implementation for them. JavaScript then became the first *implementation* of ECMAScript, defining the syntax and being implemented in runtimes in early browsers.

So ECMAScript v1 (also known as ES1) was released in 1997 and was implemented by concurrent versions of JavaScript. And like any new specification, features missing from the ECMAScript specification quickly began to be found, and so ECMAScript v2 (ES2) was released shortly after in 1998 and ECMAScript v3 (ES3) in 1999.

At this point, the specification allowed for single language files to import values into a global memory space, but things were pretty rudimentary. The next phase of work would be to define how modules worked – how large parcels of code could interoperate without using a global variable store. However, with contention between primary contributors such as Adobe and Mozilla Foundation, eventually the next iteration (ES4) was postponed as a leap that was too great at the time. And without tooling like modules to manage complexity, most projects stabilized on a single or few, manually crafted, monolithic js-file approach, and the industry eventually settled onto using frameworks like jQuery (2006) to facilitate basic-to-midscale UI interactions.

Then in 2008–2009, a series of major breakthroughs occurred in nearly domino effect, which changed the course of the JavaScript, and ECMAScript, ecosystem massively (Figure 1-1).

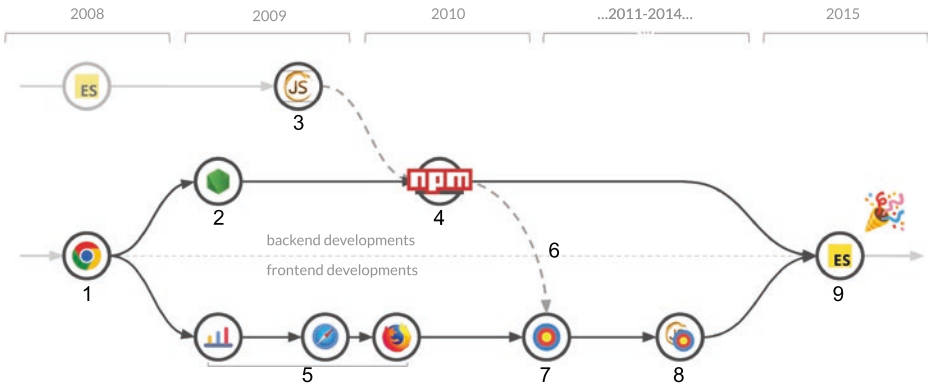


Figure 1-1. *Sequence of major breakthroughs in the JS ecosystem*

1. In 2008, Google released the Chrome browser, with a new, faster JavaScript runtime engine called V8.
2. In 2009, Chrome's new V8 engine was picked up by a team led by Ryan Dahl as the basis for a new server runtime they called Node.js. This ran plain JavaScript, but from a CLI or system service, to support non-front-end tasks.
3. Meanwhile, an independent team of volunteers had been working on an alternative to the stalled ECMAScript module problem – a new JavaScript module workaround they called CommonJS.
4. As Node.js rapidly grew, they needed a way to solve ECMAScript's missing modules problem and so picked up the CommonJS as a suitable workaround. This allowed them to then create the Node Package Manager (NPM) in 2010, as a way of sharing Node.js modules.

5. During all this time, V8's significant increase in browser performance triggered further improvements in browsers such as Safari and Firefox and therefore product potential. This renewed interest in JavaScript as a platform for delivering user experiences.

6. Larger, more sophisticated user experiences in front end were needed, and so the community began using NPM as a way of sharing front-end packages too.

7. This triggered a new front-end module system called Asynchronous Module Definition (a.k.a. AMD, via RequireJS), and with this standard, a new wave of front-end development began to increase with exponential rapidity.

8. To allow module systems to work on both the browser *and* on Node.js, a third module system grandiosely titled Universal Module Definition was released, which basically packaged both AMD and CommonJS formats in a single bundle.

9. The renewed development, as well as increasing sharding of front- and back-end module systems, coupled with the hugely increased contribution and investment in the JavaScript ecosystem, saw the reconvening of the ECMAScript group and eventual creation of the ECMAScript module system (ES Modules, or ESM) in 2015.


We explore these module formats in more detail in the "Modules" section in Chapter 8.

Phew – that's a lot to take in! But the main thing you need to know is that during 2009–2010, there was a huge upsurge in the JavaScript ecosystem. This continued to have exponential growth, until NPM became the largest and most active package repository in the world and still continues as such today, 2× larger than *all other public package ecosystems combined* and with now over 32k packages published or updated every month (Figure 1-2).

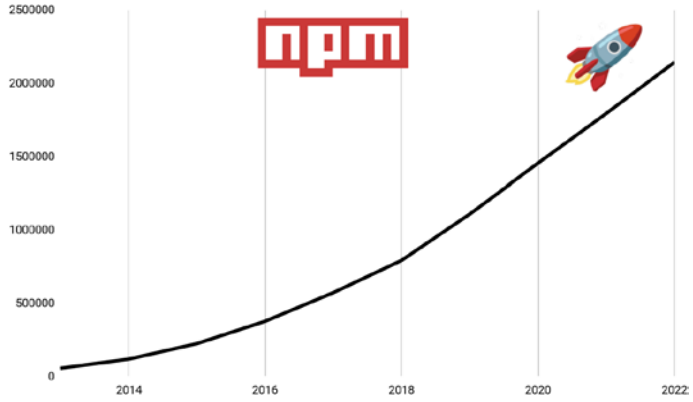


Figure 1-2. *Total number of packages on NPM over recent years*

The Problem

It would be right to say that the JavaScript ecosystem, to put it mildly, is flourishing. To keep up with the rapid development, NPM packages often depend on each other – each package providing specialism in discrete areas, with best-of-breed packages ever emerging. In one recent study, it was found that over 60% of packages had a dependency chain greater than

three layers deep; and another found that over 61% of packages could be considered abandoned with dependent ecosystems likely to need to swap them for replacements.

So it comes with the territory in JavaScript that one will be managing dependencies, constantly working out what packages are compatible with others, and trying to test against flaws. But if we have to face continually updating packages, how will we know that an update will still work?

Scalable Velocity

Scalable velocity is important in software development because it enables teams to maintain a consistent pace of development and predictable expectations, even as the project grows in complexity and size. But with an ever-shifting foundation, businesses faced needing to mitigate the risk of continual unknown compatibility.

Fortunately, there are two tools built for ensuring scalable velocity: **Testing** and **Docs**.

Testing Allows Scale

The first tool to protect against churn from ecosystem velocity is exhaustive testing. Ignoring for a moment the problems that overtesting can cause (calcification and friction; we'll come back to these), there's no denying that we can feel confident updating a whole batch of packages if we have tests that ensure our program still works afterward.

And so 2010–2011 saw the rapid increase of JavaScript testing frameworks, such as Jasmine and Mocha. These provided a standardized way to write tests against JavaScript products and as such a way to ensure teams to retain velocity at scale.

But tests are a double-edged sword – they can help certainty and thereby speed development; but exhaustive testing can slow development where tests need large-scale review and updating after changes, and often unit and E2E tests replicate product code and cause changes to effectively need to be written twice.

Docs Allow Scale

Another aspect of maintaining scalable velocity is the need to educate consumers on the changes and how to best use the updated systems. Documentation therefore is our second tool in mitigating difficulty onboarding new versions of packages, by supplying training and deeper understanding for consuming devs.

And so, likewise, 2010–2011 saw the rise of generative or simplified tooling like Dox and the much increased adoption of existing systems like JSDoc.

But writing documentation takes time, and documentation is out of date as soon as you update a product. Reading documentation also takes time and can be hard to convince consumers to do. So returns on documentation can be limited.

Types Provide Both Testing and Docs

So we have two solutions, both imperfect. But if we combine them, can we get something more ideal?

Static testing is the practice of evaluating code without running it. Types perform compatibility checking within your code and between packages and so provide the simplest form of static testing; and types can work as documentation, inlined into dev IDEs and guaranteed to be up to date.

And therefore in October 2012, Microsoft released TypeScript, an at-build-time tool for static testing JavaScript. TypeScript was an implementation of ECMAScript, which surfaced the underlying dynamically asserted types of JavaScript. Using TypeScript, developers didn't have to work out the types of variables and functions in order to understand the code, but instead their IDEs could refer to TypeScript's inline type lints, allowing structural types to be defined explicitly in the code. Running TypeScript checked the contracts for the developer, both providing live compatibility tests (also known as type checks) as well as inline documentation for the structures. This was the best of both solutions in action.

Since then, nearly all large-scale projects have migrated onto TypeScript as a tool for providing documentation and static testing at scale with low friction. Teams using the language have found it much easier, and more reliable, to stay abreast of constant changes in dependencies, and the JavaScript ecosystem has grown faster and even more successfully than before.

Summary

In this chapter, we explored the history of JavaScript and ECMAScript and how these gave birth to a need for scalable velocity and how this was then addressed through types in TypeScript. We saw how TypeScript's ability to perform compatibility checking and provide inline type lints offered the best of both worlds, allowing teams to manage dependencies more confidently and efficiently. And we saw how, as a result, TypeScript adoption skyrocketed, with large-scale projects flocking to leverage its benefits for documentation and static testing.

The stage now set, we are now primed with a better appreciation of the language to delve deeper into TypeScript's core features and dive into the world of structural and computed typing. By harnessing TypeScript's power, you will be able to tackle complex projects, and lean into JavaScript's thriving ecosystem, with complete confidence.

CHAPTER 2

Getting Started with the Developer Experience

Introduction

In this chapter, we will set up your TypeScript development environment and explore the essential tools and configurations needed to streamline your TypeScript projects. A well-configured development environment is crucial for maintaining scalable velocity and effectively managing dependencies.

You will learn how to set up TypeScript from scratch with a no-frills approach, allowing you to run TypeScript files directly from the command line. We'll walk through the process of creating a simple TypeScript file, execute it using the ts-node package, and then leverage Visual Studio Code's debugging capabilities to pause execution and inspect your code during runtime.

We'll then explore a more realistic setup for TypeScript projects, one that leverages best-of-breed packages from the JavaScript ecosystem. We'll introduce you to powerful build and management tools like NX, which

© Ben Beattie-Hood 2023
B. Beattie-Hood, *Modern TypeScript*, https://doi.org/10.1007/978-1-4842-9723-0_2

automate the configuration process, keeping your projects up to date with industry best practices. These tools will empower you to focus on building your applications without worrying about tedious configuration tasks.

By the end of this chapter, you will have a solid understanding of how to set up a TypeScript project for any enterprise need, to build robust and maintainable applications.

Environment and IDE

Currently, most TypeScript and JavaScript builds are done using Node.js, so we'll use that here. To install it, visit `https://nodejs.org/en/download`, and choose the correct version for your operating system.

And for this book, I'll assume you will be using Visual Studio Code. This small, extensible IDE is the de facto standard for most web development and comes with built-in support for TypeScript. It too is simple to install – just visit `https://code.visualstudio.com/download`, and again choose the correct version for your operating system.

No-Frills TypeScript Setup

To get started with a vanilla, zero additions, no-frills setup of TypeScript, create a new folder, open a terminal or command-line there (I'll refer to this as a terminal from now on), and enter the following:

Listing 2-1. A vanilla, no-frills setup of TypeScript

```
npm install typescript ts-node
```

This will create a package.json and package-lock.json for you, and install both TypeScript and ts-node (a package that allows you to run TypeScript directly from Node.js). Now, let's try it out.

Create a folder called "src", and create a file called main.ts within it. Add the following code to the file:

Listing 2-2. /src/main.ts

```
console.log('Hello world!');
```

Now, back in your terminal, type the following and press enter:

Listing 2-3. Running TypeScript from the terminal

```
npx ts-node src/main.ts
```

Congratulations, you've just run your first TypeScript file from the command line!

Debugging TypeScript

Running TypeScript as shown in the preceding examples is fine, but you'll also need to be able to debug TypeScript in order to build at scale. Visual Studio Code comes with support for this, so let's set this up now.

In Visual Studio Code, navigate to debug panel (1); press the "create a launch.json file" link (2) (Figure 2-1).

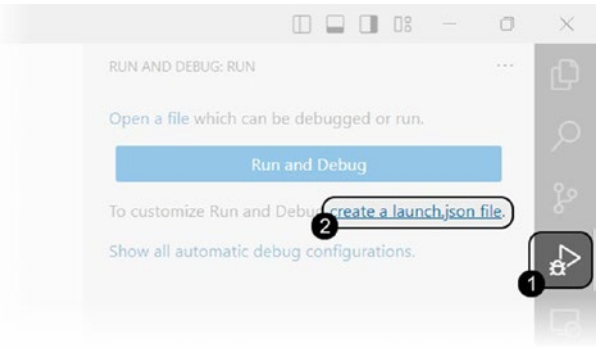


Figure 2-1. *Creating a launch.json file*

Choose “Node.js” (3) from the pop-up list of debugger options (Figure 2-2).

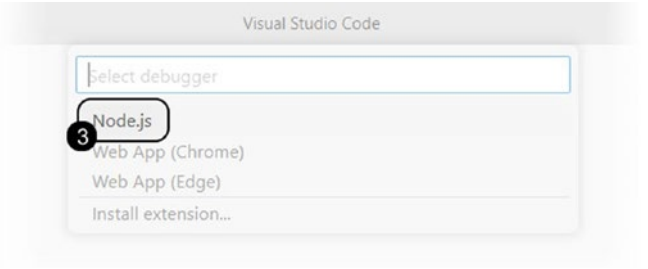


Figure 2-2. *Selecting Node.js as a debugger*

And lastly edit the automatically created “launch.json” file to include the “runtimeArgs” details as follows:

Listing 2-4. Visual Code launch.json

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "node",
            "request": "launch",
            "name": "Launch Program",
            "runtimeArgs": [
                "-r",
                "ts-node/register"
            ],
            "skipFiles": [
                "<node_internals>/**"
            ],
```

```
            "program": "${file}"
        }
    ]
}
```

That's it! Now navigate to a line in your main.ts file that you'd like execution to pause on, and press the F9 key on your keyboard, or click in the left gutter, to create a breakpoint. Now, press the F5 key on your keyboard to run your program, and you'll see the IDE pause on your breakpointed line.

If you want a quick scratch pad for trying out TypeScript code outside your project, you might want to check out the "Quokka.js" extension for Visual Studio Code. This extension will allow you to run TypeScript files live as well as set up live tests against them. It's an easy way to explore the language more, and I've found it an invaluable tool. The free online "TypeScript Playground" (`www.typescriptlang.org/play`) is another similar resource.

Another useful extension to try in Visual Studio Code, while getting into TypeScript, is "Pretty TypeScript Errors." This extension will provide a more readable breakdown of error messages, as well as links to further information and sometimes a video on how to solve them.

Looking good! We can build, run, and debug. And we could go further on this, as there are a lot more options we can pass to TypeScript via config to use it more effectively. We'll get into these later (see Build ➤ Compiler Options); but rather than digging deeper and deeper into those now, let's just pause for a minute and take a reality check.

TypeScript Within a More Realistic Setup

Although you can build and run TypeScript files directly via the CLI as shown previously, TypeScript is honestly better used as part of a larger toolchain – leveraging best-of-breed packages from the JavaScript

ecosystem to build large scalable websites, back ends, or native apps. Most books will suggest you clone a custom repo created by the author, pre-set for these configurations; but to be frank, that's not how I'd expect you to use TypeScript in industry. Instead, let's now set up TypeScript as part of a toolchain in the best possible way that will be easy to keep up to date.

Sounds good? But with the JavaScript ecosystem moving so quickly, how are we to keep continually up to date on all the "best practice" configurations and NPM packages for our particular use cases?

Thankfully, the ecosystem has matured to a state where this can be done for you by a couple of useful build and management tools – you no longer need to retain a full-time job learning about build caching, module bundle splitting, and the like – these can take care of these tasks for you.

The first of these is NX. Let's try it out now. Create a new folder, open a terminal there, and enter the following:

Listing 2-5. Installing NX

```
npm install --save-dev @nrwl/next
```

See how we didn't need to install TypeScript or ts-node, or configure either of these to "best practice"? Now, let's use NX to configure our TypeScript project to build a web app, built on Next.js using React and TypeScript:

Listing 2-6. Creating a Next.js+TypeScript+React site using one of NX's built-in templates

```
npx nx g @nrwl/next:app my-new-app
```

Now, without having to configure or maintain it, we have a fully working repo for Next.js using React and TypeScript, with additional build and test tooling out of the box, ready to go. And on top of that, we also don't have to maintain our repo configuration as tooling packages, configurations, and practices come and go out of date.

Honorable Mentions

NX and Turborepo provide a solid basis for frictionless, enterprise-scale development. If you're interested in pushing the boundaries a bit further, some other current tooling worth being aware of at present with TypeScript is listed as follows, for further research:

- Turborepo (https://turbo.build/): An alternative to NX, developed by Vercel – the company behind Next.js, a leading React framework and contributor to the React core libraries. It's a little more bare-bones than NX currently, but if you're targeting Next.js/React, it is a more canonical tooling with many useful features.
- ESLint (https://eslint.org/, https://typescript-eslint.io/): Although TypeScript will check most of your types, there are still rules developed by the community that can assist further for type safety. We'll cover this tool in more detail in the "Linting" section in Chapter 8, later in this book.
- Bun (https://bun.sh/): A highly optimized alternative to Node.js, Deno, and others, which includes built-in support for TypeScript with no compilation required.
- Civet (https://civet.dev/): A meta-language built on top of TypeScript that provides many functional concepts not yet accepted by the W3C or the TC39 steering committee, such as pattern matching, pipelines, infix operators, chained comparisons, returning conditionals, and much more.

- RedwoodJS (https://redwoodjs.com/): A batteries-included alternative to Next.js and others that integrates best-of-breed for each front-end tooling category and provides some simple meta-language that reduces client-server boilerplate.

Summary

In this chapter, we learned how to get started with TypeScript, including how to set up a simple project and how to use a templating tool for more advanced use cases.

We covered the basics of TypeScript, such as installing the **`tsc`** CLI tool, creating a configuration file, and compiling and debugging TypeScript code. Running and debugging TypeScript files from the terminal and Visual Studio Code allowed us to gain confidence in executing TypeScript code efficiently.

We also explored more advanced use cases by using the templating tool **`NX`** to generate your TypeScript configuration and allow easy running and debugging of your app. These tools automated the configuration process, sparing us from the need to retain a full-time job learning about various build tasks and module bundling. By leveraging these tools, we created a fully functional Next.js web app, powered by React and TypeScript, without any manual configuration.

To enhance our development experience, we explored useful extensions like "Quokka.js" and "Pretty TypeScript Errors," which provided us with live TypeScript code execution, better error messages, and improved code readability. We also reviewed some additional tools that reduce errors and/or speed development, in ESLint, Bun, Civet, and RedwoodJS.

As we move forward in this book, armed with a well-configured development environment, we'll dive deeper into TypeScript's core concepts and advanced features. By mastering the tools and setup presented in this chapter, you have a solid springboard for successful TypeScript development, building scalable and maintainable applications within the JavaScript ecosystem.

CHAPTER 3

TypeScript Basics

Introduction

In this chapter, we will explore the basics of TypeScript's structural typing system, often known as "duck typing," and how it differs from the hierarchical types offered by traditional object-oriented programming.

Structural typing asserts that the shape or structure of an object is more critical than its instance type or class. JavaScript uses runtime structural typing, and TypeScript supports these structural types by allowing the developer to specify them as contracts pre-build via explicit lint annotations. TypeScript then uses these annotations to check that the inferable type of a value matches the developer's expectations.

In this chapter, we will explore the ways we can use TypeScript to define these "minimum contract" types for values and how structural typing allows various objects to fulfill that contract in a more maintainable and scalable way than traditional inheritance and polymorphism.

We will also explore the concepts of type inference, narrowing and widening, and the interpretation of types via an inbuilt mechanism called "control-flow analysis." We will look at optionality, what to do when you have unknown types, and then begin a deeper dive into more advanced concepts such as simple value vs. parameterized types.

© Ben Beattie-Hood 2023
B. Beattie-Hood, *Modern TypeScript*, https://doi.org/10.1007/978-1-4842-9723-0_3

By the end of this chapter, you will have a deep understanding of structural typing and its applications in TypeScript. You will be equipped with the knowledge and tools to ensure type safety, harnessing the power of TypeScript while embracing the dynamic nature of JavaScript.

Structural Typing

Dynamic languages such as JavaScript use a concept called "structural typing." This is more commonly called "duck typing." Figure 3-1 shows the usual obtuse and confusing picture people will often point you toward when you ask "why is it called duck typing?"

Figure 3-1. *Is this a rabbit or a duck?*

Not very helpful, I agree. But the preceding image actually looks like a duck *and* a rabbit (depending on what you're looking for) – so the point here is that it's somewhat *both* things at the same time, like Schrödinger's cat, being both alive and dead simultaneously. And the analogy goes that you'll only really be able to determine if it's one or the other type if someone adds additional detail, like legs or nose.

What the duck-or-rabbit analogy, or "duck typing" nickname, is getting at is that the **type** or the **class** of an object is *less* important than the **methods it defines**.

I find an easier way to visualize it as shown in Figure 3-2.

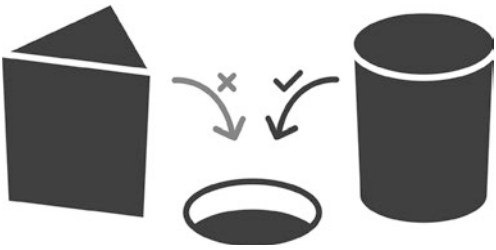


Figure 3-2. *The shape of the hole defines the contract*

In the preceding image, whatever is round can fit into the hole. It doesn't matter if it's

If it fits the "contract" delineated by the round hole, it can successfully fit in the round hole (Figure 3-3).

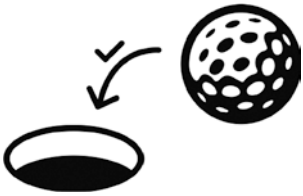


Figure 3-3. *Other items that match the same shape are accepted too*

So that's the way you can think of structural types: **structural types define a minimum shape (or structure) of a value**. And it doesn't matter what the value is, what it inherits from, or anything else really – if it fits the required minimum contract, it's accepted. And while this may seem overly simplistic, in practice, it turns out to be much more powerful, and scalable, than alternative approaches.

This power can take some getting used to, if you come from an object-oriented programming background and are more familiar with using inheritance and polymorphism. So let's illustrate the power of this with a different example, this time with code. Take the following types:

Listing 3-1. Limits of polymorphism, 1

```
interface Toy {
  manufacturer: string
  height: number
  width: number
}
```

```
interface Food {
  height: number
  width: number
  expiry: date
}
```

```
interface Painting {
  height: number
  width: number
  style: string
}
```

Let's say we wanted to put all those things into a cardboard box and wanted to know if they'd all fit. If we were using polymorphism, we'd have to inherit them all from some sort of "ThingWithASize" base class and then define the function like this:

Listing 3-2. Limits of polymorphism, 2

```
class Toy extends ThingWithASizeAbstractType { ... }
class Chair extends ThingWithASizeAbstractType { ... }
class Painting extends ThingWithASizeAbstractType { ... }
```

```
function addToBoxPolymorphically(thing:
ThingWithASizeAbstractType) {
  // ...
}
```

Yuck. And we start to hit problems here, finding that polymorphism isn't really helping us, because in another context (say, storing Purchasables, or Favorites), we can see that those three things *don't* really inherit from a common class – what they inherit from really depends on the context.

A polymorphic approach leads into having strange object taxonomies, trying to force various conflicting contexts and needs into a single polymorphic inheritance tree. Polymorphism also requires that the abstract class inherited is already defined; so any later extension can often require a rewrite, or to try to predict all eventualities in your inheritance tree.

Figure 3-4. *Polymorphism problem: Which base class should we inherit the items from?*

To avoid this problem of inheritance, structural typing proposes instead, "hey, if whatever you're assigning to this value fits this minimum contract, then we're all good" – thereby removing the need for weird arbitrary taxonomies of conflicting contexts.

Making types only a "minimum contract" for what is required by any given context, structural typing also allows for some additional bonus features, such as function, intersection, and union types becoming possible – we'll come back to these later.

JavaScript Types Are Structural Contracts

Happily, JavaScript is structurally typed. But confusingly, it does this by using *runtime* typing of values, only asserting them when a value is required:

Listing 3-3. JavaScript runtime typing

```
function addToBoxInJavaScript(thing /* can pass anything in
                                    here */) {
  // No requirements up here
  console.log(`Adding ${thing}`);

  // JavaScript's runtime structural type assertion
  // happens here:
  // the minimum contract of height & width are asserted at
  // retrieval
  const height = thing.height;
  const width = thing.width;

  // Maybe further requirements down here
  // ...etc...
}
```

You can pass any value into the function in the preceding JavaScript code snippet, and the JavaScript runtime will only assert the value meets the "minimum contract" of the type when the code evaluates a code line where the value is further interrogated.

Allowing for structural typing is powerful, but JavaScript's do-it-at-runtime provision of this makes it tricky to refactor, as the runtime structural contracts become buried in the code, adding cognitive load.

TypeScript provides a way to specify these underlying runtime structural contracts as explicit up-front lint assertions, inline, within your JavaScript code. These TypeScript types thereby allow easier static analysis – thus simpler refactoring and thereby aiding your velocity when working at scale.

Adding Explicit Types to JavaScript

TypeScript is based on ECMAScript, providing a way to add inline lint annotation for the structural contracts – TypeScript is just JavaScript with type assertions added inline. And so, all the TypeScript compiler needs to do – aside from the hard work of asserting the structural contracts – is strip out the type lint, and the code is then valid JavaScript for normal use, bundling, etc. (In practice, the TypeScript compiler does a fair bit more than this, but it's a good mental model for now.)

Let's have a look at a very simple example:

Listing 3-4. A simple const

```
const fortyTwo = 42;
```

If you hover over the variable "fortyTwo" in your editor, you'll see a tooltip containing this:

Listing 3-5. TypeScript's analysis of our const

```
› const fortyTwo: 42
```

In the tooltip, the ": 42" suffix isn't the value – it's actually the TypeScript type constraint. Because we've used a constant, TypeScript is saying that the *only* value that is the right shape for this box is the value 42 itself. Let's try the same thing manually with a nonconstant by entering the following in our editor instead:

Listing 3-6. An equivalent let literal instead

```
let fortyTwo: 42 = 42;
```

This works. Now, if you try to assign the value again, it'll only work if the value assigned matches the "minimum contract" of our type constraint:

Listing 3-7. Attempting to reassign our constrained let literal

```
let fortyTwo: 42 = 42;
fortyTwo = 42; ✔ // fits the contract, so this doesn't error
fortyTwo = 58; ✖ // whereas here we're protected from the
                 // invalid value
```

This preceding narrow type constraint is called a "literal" type – a type whose value is *literally* the same as the type definition. These literal types can be used with all primitives:

Listing 3-8. Other primitive literals

```
// boolean literal types:
let yes: true = true;
let no: false = false;

// string literal types:
let red: '#990000' = '#990000';
```

We can also loosen (a.k.a. "widen") these structural constraints a bit too. The following allows the variable "age" to be assigned anything that is a number (therefore, it is no longer a literal type):

Listing 3-9. Widening our constraints

```
let age: number = 42;
age = 58; ✔ // yay! doesn't error
age = 'hello'; ✕ // phew, still protected from non-numbers
```

The same goes for other types:

Listing 3-10. Widening our constraints using other types

```
// boolean value types:
let isArchived: boolean = false;
isArchived = true; ✔ // doesn't error

// string value types:
let color: string = '#990000';
color = '#333333'; ✔ // doesn't error

// variable-length arrays:
const names: string[] = [ 'Ashley', 'Ted', 'Kim', 'Dave' ]; ✔
names[0] = 'Veronica'; ✔
names[0] = 42; ✕
```

So structural types allow us to specify that *only* values that fulfill the minimum contract of the type specified can be assigned. This isn't limited to primitives as shown in previous examples; we can also define custom shapes.

We can define a custom array, like this:

Listing 3-11. Custom arrays

```
// fixed-type array:
const values: [number, number, string, ...string[]] =
    [41, 32, 'hi']; ✔
values[0] = 4; ✔
values[8] = 9; ✘
values[8] = 'something'; ✔

// fixed-length array, also known as a 'tuple' (pronounced
   tuh-pl):
const coords: [number, number] = [41, 32]; ✔
coords[0] = 38; ✔
coords[8] = 14; ✘
```

And we can even define a custom object type, like this:

Listing 3-12. Custom object types

```
interface Employee {
    name: string
    employeeNumber: number
    dateOfBirth: Date
    address: {
      street: string
      postcode: string
    }
    isArchived: boolean
}
```

Once defined, we can use our custom object type to add a structural contract to a value like this:

Listing 3-13. Using our types as structural contracts

```
✔ // doesn't error
const amy: Employee = {
    name: "Amy",
    employeeNumber: 123,
    dateOfBirth: new Date(2000, 9, 1),
    address: {
        street: "123 Street",
        postcode: 1234
    },
    isArchived: false
}

✖ // this will error
const ted: Employee = {
    name: "Ted",
    employeeNumber: 123
}
```

In this way, structural contracts allow us to define the **minimum structure** that is required for value assignment.

This method of using types to define the minimum structure required can be used in other places too, such as function parameters:

Listing 3-14. Using TypeScript on function parameters

```
// Simple parameters:
function greet(firstName: string, lastName: string): string;
// or, using a lambda style:
// const greet = (firstName: string, lastName: string):
string => ...;

greet("Ted", "Smith") ✔ // doesn't error
greet(12345, "Smith") ✕ // this will error

// And complex parameters:
function draw(shape: {
    type: string, coords: [number, number] }): void;
// or, using a lambda style:
// const draw = (shape: {
    type: string, coords: [number, number] }) void => ...;

draw({
    type: 'square',
    coords: [32, 48]
});  ✔ // doesn't error

draw({
    type: 'square',
    coords: 'top left'
});  ✕ // this will error
```

Optionality

In the preceding **`Employee`** type, we might have one or two fields that we want to specify the type of *only* if they are present. We call these optional fields. We can use a shorthand "**?**" optionality modifier to indicate them:

Listing 3-15. Allowing some fields to be optional

```
interface Employee {
    name: string
    employeeNumber: number
    dateOfBirth?: Date      👈 // Optional
    address?: {             👈 // Optional
        street: string
        postcode: string
    }
    isArchived?: boolean    👈 // Optional
}

✔ // doesn't error
const amy: Employee = {
    name: "Amy",
    employeeNumber: 123
}

✖ // this will error
const ted: Employee = {
    name: "Ted",
    employeeNumber: 123,
    dateOfBirth: 'August'
}
```

This allows us to skip some fields if left undefined – while also ensuring that if they are present, they are the right type.

And we can use this optionality modifier in function definitions and class constructors too:

Listing 3-16. Allowing some parameters to be optional

```
// Optional function parameters
function greet(name: string, age?: number): string;

✔ // both these are fine:
greet('Homer', 42);
greet('Marge');

// Optional constructor parameters
class Person {
    constructor(name: string, email?: string, age?: number);
}

✔ // all these are fine:
new Person('John', 'john@example.com', 31);
new Person('Fi', 'fi@example.com');
new Person('Jane');
```

If you've got sharp eyes, you will have spotted that the optional parameters in the preceding examples are only ever trailing parameters. This is because the parameters are provided by calling code "ordinally" (in sequence), rather than by name – and so it would be impossible to write valid calling code that worked for the following:

Listing 3-17. Optional parameters have to be last

```
// The following will error...
function greet(name: string, age?: number, dob: Date): string; ✖
```

```
// ...because it's impossible to know if the following is
// correct or a typo:
greet('John', new Date()); 🤔
```

Therefore, optional parameters must always be at the end of the function call, and so TypeScript includes a check for this as part of the type analysis.

All The Sugar

Over time, ECMAScript has added additional syntactic sugar to simplify common or complex tasks. And for all the ECMAScript features, TypeScript includes support for inferring their types out of the box. Let's pause for a moment and look at some of the more advanced ECMAScript features to explore the extent of the type safety TypeScript offers.

Array and Object Destructuring, Spread, and Rest

TypeScript provides type safety for array and object destructuring by analyzing the structure of the destructured variables and inferring their types from the matching types of the original objects or arrays, even using the ECMAScript rest operator (a.k.a. "**...**").

Listing 3-18. Destructuring arrays and object types

```
// For arrays:
const [name, age, ...others] = ["John", 30, true, "Red"];
 ❯    const name: string;
 ❯    const age: number;
 ❯    const others: [boolean, string];
```

```
// For objects:
const { name, age, ...others } =
   { name: "John", age: 30, isActive: true, color: "Red" };
 ❯    const name: string;
 ❯    const age: number;
 ❯    const others: { isActive: boolean, color: string };
```

Did you notice that TypeScript infers the preceding destructured values as nonliterals (e.g., **`string`**, **`boolean`**, **`number`**, etc.), rather than as the literals as we had in our original "const" example we started with? That's because, although the outer variable is a constant, the inner array items or object fields could be reassigned – so essentially it is like they are assigned with **`let`**. Instead, we can assign them as **`const`**, and the result is more expected:

Listing 3-19. Destructuring arrays and object types more narrowly

```
// For arrays:
const [name, age, ...others] =
    ["John", 30, true, "Red"] as const;
 ❯    const name: "John";
 ❯    const age: 30;
 ❯    const others: [true, "Red"];

// For objects:
const { name, age } =
   { name: "John", age: 30, isActive: true, color: "Red" }
   as const;
 ❯    const name: "John";
 ❯    const age: 30;
 ❯    const others: { isActive: true, color: "Red" };
```

This works for the spread on the assignee side too:

Listing 3-20. Types can be persisted even through spreads

```
// For arrays:
const lettersArray = ['b2', 'c3', 'd4', 'e5'] as const;
const newLettersArray = ['a1', ...lettersArray, 'f6'] as const;
 ❯    const newLettersArray: readonly [
    "a1", "b2", "c3", "d4", "e5", "f6"];

// For objects:
const lettersObject = { b: 'b2', c: 'c3', d: 'd4', e: 'e5' }
as const;
const newLettersObject = { a: 'a1', ...lettersObject, f: 'f6' }
as const;
 ❯    const newLettersObject: {
        readonly a: "a1", readonly b: "b2", readonly c: "c3",
        readonly d: "d4", readonly e: "e5", readonly f: "f6",
      };
```

Tuple destructuring in types is a technique we use in Chapter 6. See **Flatten** and **UrlParameters** for examples.

Another random document with no related content on Scribd:

to the officer commanding at the Hsi-Hua Gate, ordering them to cease firing until her return to the Forbidden City.

11th Day of the 6th Moon (7th July).—Yü Lu has sent in a ridiculous memorial, reporting the capture of four camels, as well as the killing of many foreigners, in Tientsin. Jung Lu has advised him to cease attacking the foreign Settlements. Talking of Jung Lu, I hear that Tung Fu-hsiang recently hired a Manchu soldier to assassinate him, but, instead of doing so, the man betrayed the plot to Jung Lu. This soldier turns out to be a brother of that En Hai who slew the foreign devil (Baron von Ketteler), and Tung thought therefore that he would gladly do anything to assist in destroying the Legations. But he is a clansman of Jung Lu's banner, and, like Yü Kung-ssŭ, whom Mencius called the best archer in Wei, "he could not bear to slay the old Chief who had taught him the arts of war." Jung Lu has again memorialised the Old Buddha, reminding her of that well-known saying in the Spring and Autumn annals,[87] which lays down that the persons of foreign Envoys are always inviolate within the territories of any civilised State. This attack on the Legation, he says, is worse than an outrage; it is a piece of stupidity which will be remembered against China for all time. Her Majesty appeared to think that, because a small nation like the Transvaal could conquer a great Power like England, China must necessarily be even more successful in fighting the whole world; but there was no analogy between the two cases. If peace were to be made at once, the situation might still be saved; but if the Legations were demolished, there must be an end of Manchu rule. He warned Her Majesty solemnly, and she appears to be gradually coming to look at things from his point of view. These Boxers can certainly talk, but they do very little.

Bad news has reached the palace to-day of the fighting around Tientsin, and Her Majesty is most anxious about it, though she still refuses to believe that the foreign brigands can possibly enter Peking.

15th Day of the 6th Moon (11th July).—My neighbour Wen Lien, Comptroller-General of the Imperial Household, tells me that the Old Buddha is in a furious rage. She finds the heat trying, and yesterday

she turned on the Heir Apparent and snubbed him badly for impertinence; he had asked if he might be permitted to escort her to Jehol, leaving the Emperor to settle matters with his foreign friends in Peking. One of the young eunuchs tried to mollify her by reporting, whenever the report of a gun was heard, that another foreign devil had been killed, but as the Old Buddha observed, “there has been enough firing for the past few weeks to kill off every foreigner in China several times, and so far there is hardly anything to show for it.”

17th Day of the 6th Moon (13th July).—Jung Lu asked Her Majesty yesterday what she would do if the Boxers were defeated, and if Peking were captured by the foreigners. In reply, she quoted to him the words of Chia Yi, a sophist of the Han dynasty, in reference to the Court’s diplomatic dealings with the Khan of the Hans:—

> “If the Emperor wishes to gain the allegiance of other countries, he can only do so by convincing their rulers that he possesses the three cardinal virtues of government, and by displaying the five allurements.
>
> These allurements are: (1) Presents of chariots and rich robes, to tempt the eye; (2) rich food and banquets, to tempt the palate; (3) musical maidens, to tempt the ear; (4) fine houses and beautiful women, to tempt the instinct of luxury; and (5) the presence of the Emperor at the table of the foreign ruler, to tempt his pride.
>
> The three cardinal virtues of government are: (1) to simulate affection; (2) to express honeyed sentiments; and (3) to treat one’s inferiors as equals.”

Two years ago, said the Empress, she had invited the foreign ladies to her Court, and had noticed their delight at the reception she gave them, although she well knew that their sympathies were with the Emperor, and against her. She would again allure them to her side with rich gifts and honeyed words.[88]

20th Day of the 6th Moon (16th July).—Bad news from Yü Lu; Tientsin has been captured by the foreigners, who now swarm like locusts. Not one of the Grand Councillors dared to carry the news to Her Majesty, so Prince Tuan went in boldly, and informed her that the foreign devils had taken the city, because the Boxers had been negligent in the performance of their prescribed rites; Peking, however, would always be perfectly safe from invasion. Early this morning Jung Lu had informed the Old Buddha that he had ascertained beyond doubt that the document, which purported to come from the Foreign Ministers, demanding her abdication, was a forgery. It had been prepared by Lien Wen-chung, a Secretary of the Grand Council, at Prince Tuan's orders. The Old Buddha was therefore in no soft mood; angrily she told Prince Tuan that, if the foreigners entered Peking, he would certainly lose his head. She was quite aware of his motives; he wanted to secure the Regency, but she bade him beware, for, so long as she lived, there could be no other Regent. "Let him be careful, or his son would be expelled from the palace, and the family estates confiscated to the throne." His actions had indeed been worthy of the dog's[89] name he bore. Prince Tuan left the palace, and was heard to remark that "the thunderbolt had fallen too quickly for him to close his ears."

Jung Lu has won over all the military commanders except Tung Fu-hsiang and his staff, and they have come to a general understanding that the bombardment of the Legations must cease. Jung Lu has explained, as his reason for not allowing the heavy artillery to be used, that it would inevitably have inflicted serious damage on the Imperial shrines and the Ancestral temple.

The Old Buddha is sending presents to the Legations, water-melons, wine, vegetables, and ice, and she has expressed a wish that Prince Ch'ing should go and see the Foreign Ministers.

They say that Hsü Ching-ch'eng is secretly communicating with the Legations.

A messenger with twelve dispatches from the Legations was captured to-day and taken to Prince Chuang's Palace. Three of the twelve were in cipher and could not be translated by the Tsung Li

Yamên interpreter, but from the others it was learned that the foreigners had lost over a hundred killed and wounded and that their provisions were running very low.

Chi Shou-ch'eng has gone to T'ai-Yüan fu to see Yü Hsien, his father-in-law. The latter has memorialised the Throne, reporting that he cunningly entrapped all the foreigners, cast them into chains and had every one decapitated in his Yamên. Only one woman had escaped, after her breasts had been cut off, and had hidden herself under the City wall. She was dead when they found her.

Rain has fallen very heavily to-day. Liu Ta-chiao brought me 8 lbs. of pork from the Palace kitchen, and I sent a large bowl of it to my married sister. Towards evening a detachment of cavalry, with several guns, passed my door. They were Li Ping-heng's men, on their way to mount these guns on a platform above the Forbidden City wall, as a precaution against sorties by the foreigners. There has been heavy firing all night, and it is reported that foreign devils have been seen in the neighbourhood of the Ha-Ta Men.

21st Day of the 6th Moon (17th July).—A lovely day. I walked over to call on Prince Li and Duke Lan. The latest rumour is that Yü Lu's troops are in flight and harrying the country side. They are said to be clamouring for their pay, which is months in arrears, and have plundered both Tungchou and Chang Chia-wan most thoroughly. Both the eastern gates of the City are now kept closed, and the northern gate (Anting men) is only opened occasionally.

Yang Shun, the gate-keeper, has returned from his home at Pao-ti hsien, east of Peking, where he reports things fairly quiet.

Li Ping-heng's troops are reported to have won a great victory and driven the barbarians to the sea. Nevertheless, heavy firing was heard to the south-eastward this afternoon.

Duke Lan has gone out with a large force of Boxers to search for converts reported to be in hiding in the temple of the Sun.

27th Day of the 6th Moon (23rd July).—This morning Yüan Ch'ang and Hsü Ching-ch'eng handed in the third of their Memorials against the Boxers, in which they recommend the execution of several

members of the Grand Council. Their valour seems to be more laudable than their discretion, especially as the Old Buddha is disposed once more to believe in the Boxers as the result of Li Ping-heng's audience with her yesterday. He came up from Hankow, and has now been appointed joint Commander, with Jung Lu, of the army of the North. He confidently assured her of his ability to take the Legations by storm, and repeatedly said that never again would the tutelary deities of the Dynasty suffer her to be driven forth, in humiliation, from her capital.

I went across to Duke Lan's house this morning and found Prince Tuan and Li Ping-heng there. They were busy planning a renewed attack on the Legations, and Li was strongly in favour of mining from the Hanlin Academy side. He has advised the Empress Dowager that a mine should be sprung, as was done lately at the French Cathedral, and he is convinced that in the ensuing confusion the foreigners would be easily overwhelmed.

After reading the latest Memorial of Hsü and Yüan, the Old Buddha observed, "These are brave men. I have never cared much for Hsü, but Yüan behaved well in 1898 and warned me about K'ang Yu-wei and his plotting. Be that as it may, however, they have no business to worry me with these persistent and querulous questions. The Throne itself is fully competent to judge the character of its servants, and it is a gross misconception of duty for 'the acolyte to stride across the sacred vessels and show the priest how to slaughter the sacrificial beasts.'[90] Desiring to deal leniently with the Memorialists, I command that my censure be communicated to them and that they take heed to refrain in future from troubling my ears with their petulant complainings."

3rd Day of the 7th Moon (28th July).—The Old Buddha places much confidence in Li Ping-heng. Yesterday he and Kang Yi discovered that the word "to slay," in Her Majesty's Decree ordering the extermination of all foreigners, had been altered to "protect" by Yüan Ch'ang and Hsü Ching-ch'eng. I have just seen Kang Yi, and he says that Her Majesty's face was divine in its wrath. "They deserve the punishment meted out to Kao Ch'u-mi,"[91] she said,

“their limbs should be torn asunder by chariots driven in opposite directions. Let them be summarily decapitated.” An Edict was forthwith issued, but no mention is made in it of the alteration of the Decree, as this is a matter affecting the nation’s prestige; the offenders are denounced only for having created dissensions in the Palace and favoured the cause of the foreigner. Both were executed this morning; my son, En Ming, witnessed their death. It is most painful to me to think of the end of Yüan Ch’ang, for he had many sterling qualities; as for Hsü, I knew him in the days when we were colleagues at the Grand Secretariat, and I never had a high opinion of the man. His corruption was notorious. Just before the sword of the executioner fell, Yüan remarked that “he hoped that the Sun might soon return to its place in the Heaven, and that the usurping Comet might be destroyed.” By this he meant that Prince Tuan’s malign influence had led the Empress Dowager to act against her own better instincts. Duke Lan, who was superintending the execution, angrily bade him be silent for a traitor, but Yüan fearlessly went on, “I die innocent. In years to come my name will be remembered with gratitude and respect, long after you evil-plotting Princes have met your well-deserved doom.” Turning then to Hsü, he said, “We shall meet anon at the Yellow Springs.[92] To die is only to come home.” Duke Lan stepped forward as if to strike him, and the headsman quickly despatched them both.

8th Day of the 7th Moon (3rd August).—I have had much trouble with my eldest son to-day. He has been robbing me lately of large sums, and when I rebuked him he had the audacity to reply that my duty to the Throne would make my suicide a fitting return for the benefits which I have received at its hands.

Li Ping-heng has gone to the front to rally the troops and check the foreigners’ advance. He has impeached Jung Lu but the Old Buddha has suppressed the Memorial. The Emperor thanked Jung Lu for his services, and the Commander-in-Chief replied that he of all the servants of the Throne never expected to receive praise from His Majesty, considering the events of the past two years.[93]

11th Day of the 7th Moon (5th August).—The Old Buddha has commanded Jung Lu to arrange for escorting the foreigners to Tientsin, so that the advance of the Allies may be stopped. In this connection, I hear that not many days ago, —— persuaded Ch'i Hsiu to have a letter sent to the Foreign Ministers, inviting them to come, without escort of troops, to an interview with the Tsung Li Yamên, his idea being to have them all massacred on the way. Ch'i Hsiu thought the suggestion excellent, but, although several letters have been sent proposing it, the Ministers decline to leave the Legations. Meanwhile, there have been several fresh attacks on the Legations during the past few days.

A foreign devil, half naked, was found yesterday in Hatamen Street. He kowtowed to everyone he met, high class or low, imploring even the rag-pickers to spare his life and give him a few cash. "We shall all be massacred soon," he said, "but I have done no wrong." One of Jung Lu's sergeants seized him and took him to the Commander-in-Chief's residence. Instead of decapitating him, Jung Lu sent him back. This shows, however, the desperate straits to which the foreigners are reduced.

15th Day of the 7th Moon (9th August).—Bad news from the South. Yü Lu's forces have been defeated and the foreigners are approaching nearer every day. The Old Buddha is meditating flight to Jehol, but Jung Lu strongly urges her to remain, even if the Allies should enter the City. Duke Lan scoffs at the idea of their being able to do so. One comfort is that, if they do come, they will not loot or kill. I remember well how good their discipline was forty years ago. I never stirred out of my house and not one of the barbarians ever came near it. We had a little difficulty about getting victuals, but the foreigners hardly came into the city, and did us no harm.

16th Day of the 7th Moon (10th August).—My old colleague, Li Shan, whose house adjoins the French Cathedral, has been accused of making a subterranean passage and thus assisting the foreigners with supplies. He has been handed over to the Board of Punishments by Prince Tuan, without the knowledge of the Empress Dowager, together with Hsü Yung-yi and Lien Yuan. Prince Tuan has long had a grudge against Hsü for having expressed disapproval of

the selection of the Heir Apparent. As to Lien, they say that his arrest is due to ——, and his offence is that he was on terms of intimacy with Yüan Ch'ang. All three prisoners were decapitated this morning. Hsü Yung-yi was older than I am (seventy-nine) and his death is a lamentable business indeed. But he went to his death calmly and without complaint when he learned that the Empress Dowager knew nothing of the matter and that it was Prince Tuan's doing alone. "The power of the usurper," said he, "is short-lived. As for me, I am glad to die before the foreigners take Peking." The Old Buddha will be very wrath when she hears that two Manchus have thus been put to death. Li Shan and Jung Lu were old friends.

A certain General Liu, from Shansi, assured the Empress this morning that he would undertake to demolish the Legations in three days, and this would so alarm the allies that their advance would certainly be stopped. A furious bombardment has just begun.

The Boxers have proved themselves utterly useless. I always said they never would do anything.

18th Day of the 7th Moon (*12th August*).—The foreigners are getting nearer and nearer. Yü Lu shot himself with a revolver on the 12th at Ts'ai Ts'un. He had taken refuge in a coffin shop, of all ill-omened places! His troops had been utterly routed thrice, at Pei Tsang, Yang Ts'un and at Ts'ai Ts'un. Li Ping-heng reached Ho-hsi wu on the 14th, but in spite of all his efforts to rally our forces, the two divisional leaders, Chang Ch'un-fa and Ch'en Tsê-lin, refused to fight. Li Ping-heng therefore took poison. Jung Lu went to-day to break the news to the Old Buddha: sovereign and Minister wept together at the disasters which these Princes and rebels have brought upon our glorious Empire. Jung Lu refrained from any attempt at self-justification; he is a wise man. The Old Buddha said she would commit suicide and make the Emperor do the same, rather than leave her capital. Jung Lu besought her to take his advice, which was to remain in Peking and to issue Decrees ordering the decapitation of Prince Tuan and his followers, thus proving her innocence to the world. But she seems to cling still to a hope that the supernatural powers of the Boxers may save Peking, and so the furious bombardment of the Legations continues.

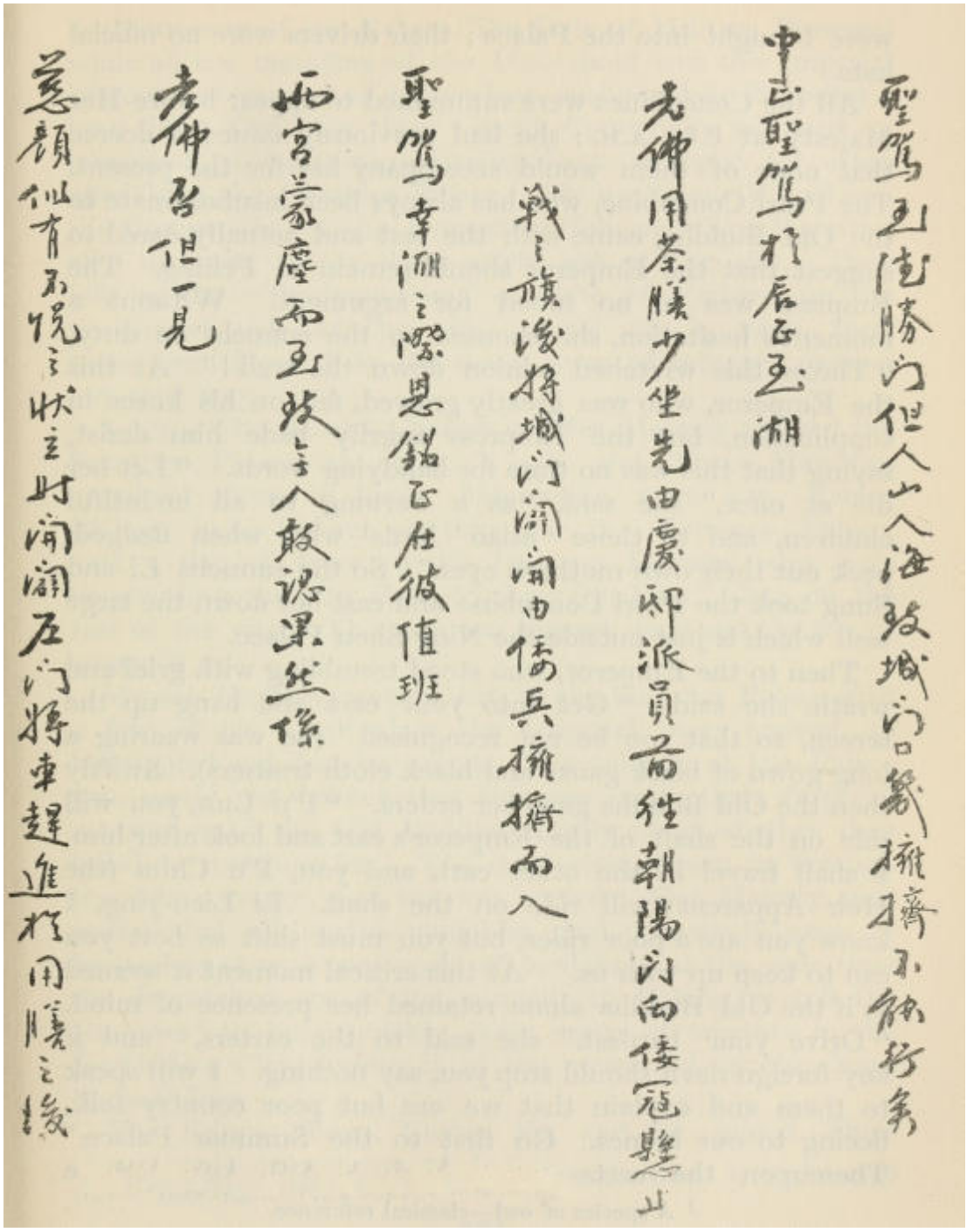
Eight audiences have been given to-day to Jung Lu and five to Prince Tuan. All the other members of the Grand Council sat with folded hands, suggesting nothing.

20th Day (*14th August*), 5 P.M.—Tungchou has fallen and now the foreigners have begun to bombard the city. The Grand Council has been summoned to five meetings to-day in the Palace of Peaceful Longevity: Her Majesty is reported to be starting for Kalgan. At the hour of the Monkey (4 P.M.) Duke Lan burst into the Palace, unannounced, and shouted, “Old Buddha, the foreign devils have come!” Close upon his footsteps came Kang Yi, who reported that a large force of turbaned soldiery were encamped in the enclosure of the Temple of Heaven. “Perhaps they are our Mahommedan braves from Kansuh,” said Her Majesty, “come to demolish the Legations?” “No,” replied Kang Yi, “they are foreign devils. Your Majesty must escape at once, or they will murder you.”

Later, midnight.—There has just been an Audience given to the Grand Council in the Palace, at which Kang Yi, Chao Shu-ch’iao and Wang Wen-shao were present. “Where are the others?” said the Old Buddha. “Gone, I suppose, everyone to his own home, leaving us here, Mother and Son,[94] to look after ourselves as best we may. At all events, you three must now accompany me on my journey.” Turning to Wang Wen-shao, she added:—“You are too old, and I could not bear the thought of exposing you to such hardships. Make such speed as you can and join me later.” Then to the other two she said, “You two are good riders. It will be your duty never to lose sight of me for an instant.” Wang Wen-shao replied, “I will hasten after Your Majesty to the best of my ability.” The Emperor, who seemed surprisingly alert and vigorous, here joined in, “Yes, by all means, follow as quickly as you can.” This ended the audience, but the actual hour of Her Majesty’s departure remains uncertain. Jung Lu’s attendance was impossible because he was busy trying to rally our forces.

21st Day (*15th August*).—Wen Lien tells me that the Old Buddha arose this morning at the Hour of the Tiger (3 A.M.) after only an hour’s rest, and dressed herself hurriedly in the common blue cloth garments of a peasant woman, which she had ordered to be

prepared. For the first time in her life, her hair was done up in the Chinese fashion. “Who could ever have believed that it would come to this?” she said. Three common carts were brought into the Palace; their drivers wore no official hats.



Facsimile of a Fragment of the Diary.

All the Concubines were summoned to appear before Her Majesty at 3.30 A.M.; she had previously issued a decree that none of them would accompany her for the present. The Pearl Concubine, who has always been insubordinate to the Old Buddha, came with the rest and actually dared to suggest that the Emperor should remain in Peking. The Empress was in no mood for argument. Without a moment's hesitation, she shouted to the eunuchs on duty, "Throw this wretched minion down the well!" At this the Emperor, who was greatly grieved, fell on his knees in supplication, but the Empress angrily bade him desist, saying that this was no time for bandying words. "Let her die at once," she said, "as a warning to all undutiful children, and to those 'hsiao' birds[95] who, when fledged, peck out their own mother's eyes." So the eunuchs Li and Sung took the Pearl Concubine and cast her down the large well which is just outside the Ning Shou Palace.

Then to the Emperor, who stood trembling with grief and wrath, she said: "Get into your cart and hang up the screen, so that you be not recognised" (he was wearing a long gown of black gauze and black cloth trousers). Swiftly then the Old Buddha gave her orders. "P'u Lun, you will ride on the shaft of the Emperor's cart and look after him. I shall travel in the other cart, and you, P'u Chün (the Heir Apparent) will ride on the shaft. Li Lien-ying, I know you are a poor rider, but you must shift as best you can to keep up with us." At this critical moment it seemed as if the Old Buddha alone retained her presence of mind. "Drive your hardest," she said to the carters, "and if any foreign devil should stop you, say nothing. I will speak to them and explain that we are but poor country folk, fleeing to our homes. Go first to the Summer Palace." Thereupon the carts started, passing out through the northern gate of the Palace (The Gate of Military Prowess) while all the members of the Household and the Imperial Concubines prostrated themselves, wishing their Majesties a long life. Only the three Grand Councillors followed on horseback, a rendezvous having been arranged for other officials at the Summer Palace. My neighbour Wen Lien, the Comptroller of the Household, followed their Majesties at a distance, to see them safely out of the city. They left by the "Te-sheng-men," or Gate of Victory, on the

north-west side of the city, where for a time their carts were blocked in the dense mass of refugees passing out that way.

4 P.M.—The Sacred Chariot of Her Majesty reached the Summer Palace at about 8 A.M. and Their Majesties remained there an hour. Meanwhile, at 6 A.M., Prince Ch'ing, just before starting for the Summer Palace, sent a flag of truce to the Japanese Pigmies who were bombarding the city close to the "Chi Hua" Gate on the east of the city. The gate was thrown open and the troops swarmed in.

My son En Ming was on duty at the Summer Palace with a few of his men, when the Imperial party arrived, all bedraggled and dust-begrimed. The soldiers at the Palace gate could not believe that this was really their Imperial mistress until the Old Buddha angrily asked whether they failed to recognise her. The carts were driven in through the side entrance, and tea was served. Her Majesty gave orders that all curios, valuables, and ornaments were to be packed at once and sent off to Jehol; at the same time she despatched one of the eunuchs to Peking to tell the Empress[96] to bury quickly every scrap of treasure in the Forbidden City, hiding it in the courtyard of the Ning Shou Palace.

The Princes Tuan, Ching, Na, and Su joined Their Majesties at the Summer Palace; a few Dukes were there also, as well as Wu Shu-mei and Pu Hsing of the higher officials. About a dozen Secretaries from the different Boards, and three Clerks to the Grand Council, accompanied the Court from this point. General Ma Yu-k'un, with a force of 1,000 men escorted Their Majesties to Kalgan, and there were, in addition, several hundreds of Prince Tuan's "Heavenly Tiger" Bannermen, fresh from their fruitless attacks on the Legations. Jung Lu is still endeavouring to rally his troops.

I have just heard of the death of my old friend, Hsü T'ung, the Imperial Tutor and Grand Secretary. He has hanged himself in his house and eighteen of his womenfolk have followed his example. He was a true patriot and a fine scholar. Alas, alas! From all sides I hear the same piteous story; the proudest of the Manchus have come to the same miserable end. The betrothed of Prince Ch'un, whom he

was to have married next month, has committed suicide, with all her family. It is indeed pitiful.[97]

Thus, for the second time in her life, the Old Buddha has had to flee from her Sacred City, like the Son of Heaven in the Chou Dynasty, who “fled with dust-covered head.” The failure of the southern provinces to join in the enterprise has ruined us. Prince Tuan was much to blame in being anti-Chinese. As Confucius said, “By the lack of broad-minded tolerance in small matters, a great design has been frustrated.” After all, Jung Lu was right—the Boxers’ so-called magic was nothing but child’s talk. They were in reality no stronger than autumn thistledown. Alas, the bright flower of spring does not bloom twice!

DAUGHTERS OF A HIGH MANCHU OFFICIAL OF THE COURT.

My wife and the other women, stupidly obstinate like all females, intend to take opium. I cannot prevent them from doing so, but, for myself, I have no intention of doing anything so foolish. Already the foreign brigands are looting in other quarters of the city, but they will

never find my hidden treasure, and I shall just remain here, old and feeble as I am. My son, En Ch'u, has disappeared since yesterday, and nearly all my servants have fled. There is no one to prepare my evening meal.

(*Here the Diary ends. The old man was murdered by his eldest son that same evening; all his women folk had previously taken poison and died.*)

Vermilion Decree of H.M. Kuang Hsü, 24th day, 12th Moon of 25th year (January, 1900), making Prince Tuan's son Heir Apparent.

"In days of our tender infancy we succeeded by adoption to the Great Inheritance, and were favoured by the Empress Dowager, who graciously 'suspended the curtain' and administered the Government as Regent, earnestly labouring the while at our education in all matters. Since we assumed the reins of government, the nation has passed through severe crises, and our sole desire has been to govern the Empire wisely in order to requite the material benevolence of Her Majesty as well as to fulfil the arduous task imposed on us by His late Majesty.

"But since last year our constitution has been sore-stricken with illness, and we have undergone much anxiety lest the business of the State should suffer in consequence. Reflecting on the duty we owe to our sacred ancestors and to the Empire, we have therefore besought Her Majesty to administer the Government during the past year. Our sickness has so far shown no signs of improvement, and it has prevented us from performing all the important sacrifices at the ancestral shrines and at the altars of the gods of the soil.

"And now at this acute crisis, the spectacle of Her Majesty, labouring without cease in the profound seclusion of her Palace, without relaxation or thought of rest, has filled us with dismay. We can neither sleep nor eat in the anxiety of our thoughts. Reflecting on the arduous labours of our ancestors

from whom this great Heritage has descended to us, we are overwhelmed by our unfitness for this task of government. We bear in mind (and the fact is well known to all our subjects) that when first we succeeded by adoption to the Throne, we were honoured with a Decree from the Empress Dowager to the effect that so soon as we should have begotten an heir, he should become the adopted son of His Majesty T'ung-Chih. But our protracted sickness renders it impossible for us to hope for a son, so that His late Majesty remains without heir. This question of the succession is of transcendent importance, and our grief, as we ponder the situation, fills us with feelings of the deepest self-abasement, and renders illusive all hope of our recovery from this sickness.

"We have accordingly prostrated ourselves in supplication before our Sacred Mother, begging that she may be pleased to select some worthy person from among the Princes of the Blood as heir to His Majesty T'ung-Chih, in order that the Great Inheritance may duly revert to him. As the result of our repeated entreaties Her Majesty has graciously consented, and has appointed P'u Chün, son of Prince Tuan, as heir by adoption to His late Majesty. Our gratitude at this is unbounded, and obediently we obey her behests, hereby appointing P'u Chün to be Heir Apparent and successor to the Throne. Let this Decree be made known throughout the Empire."

Seldom has history seen so tragically pathetic a document. It was not only a confession of his own illegality and an abdication, but his death-warrant, clear writ for all men to read. And the poor victim must perforce thank his executioner and praise the "maternal benevolence" of the woman whose uncontrollable love of power had wrecked his life from the cradle.

Memorial from the Censorate at Peking to the Throne at Hsi-an, describing the arrest of En Hai, the murderer of the German Minister, Baron von Ketteler.[98]

This Memorial affords a striking illustration of the sympathy which animated, and still animates, many of those nearest to the Throne in regard to the Boxers and their anti-foreign crusade, and their appreciation of the real sentiments of the Empress Dowager, even in defeat. It also throws light on the Chinese official's idea of heroism in a soldier.

> "A spy in Japanese employ, engaged in searching for looted articles in the pawnshops of the district in Japanese military occupation, found among the unredeemed pledges in one shop a watch bearing Baron von Ketteler's monogram. The pawnbroker said that it had been pledged by a bannerman named En Hai, who lived at a carters' inn of the Tartar city. This spy was a man named Te Lu, a writer attached to the Manchu Field Force, of the 8th squad of the 'Ting' Company. He went at once and informed the Japanese, who promptly sent a picquet to the inn mentioned. Two or three men were standing about in the courtyard, and the soldiers asked one of them whether En Hai was there. 'I'm the man,' said he, whereupon they took him prisoner. Under examination, En was perfectly calm and showed no sort of emotion. The presiding Magistrate enquired 'Was it you who slew the German Minister?' He replied 'I received orders from my Sergeant to kill every foreigner that came up the street. I am a soldier, and I only know it is my duty to obey orders. On that day I was with my men, some thirty of them, in the street, when a foreigner came along in a sedan chair. At once I took up my stand a little to the side of the street, and, taking careful aim, fired into the chair. Thereupon the bearers fled: we went up to the chair, dragged the foreigner out, and saw that he was dead. I felt a watch in his breast pocket and took it as my lawful share; my comrades appropriated a revolver, some rings and other articles. I never thought that this watch would lead to my detection, but I am glad to die for having killed one of the enemies of my country. Please behead me at once.'

“The interpreter asked him whether he was drunk at the time. He laughed and said, ‘Wine’s a fine thing, and I can put away four or five catties at a time, but that day I had not touched a drop. Do you suppose I would try to screen myself on the score of being in liquor?’ This En Hai appears to have been an honest fellow; his words were brave and dignified, so that the bystanders all realised that China is not without heroes in the ranks of her army. On the following day he was handed over to the Germans, and beheaded on the scene of his exploit. We, your Memorialists, feel that Your Majesties should be made acquainted with his meritorious behaviour, and we therefore report the above facts. We are of opinion that his name should not be permitted to fall into oblivion, and we trust that Your Majesties may be pleased to confer upon him honours as in the case of one who has fallen in battle with his face to the foe.”

XVIII

IN MEMORY OF TWO BRAVE MEN

The Memorial of the Censors given in the last chapter, recording the arrest and execution of the Manchu soldier who shot the German Minister defenceless in his chair, took occasion to congratulate the Empress and the nation on possessing such brave defenders; and to do the man justice, he met his end with a fine courage. But with fuller knowledge and a clearer insight, the scholars of the Empire might well put forward claims to real heroism, moral courage of the rarest kind, in the case of Yüan Ch'ang and Hsü Ching-ch'eng, the two Ministers who, as we have shown, so nobly laid down their lives for what they knew to be their country's highest good. So long as China can breed men like these, so long as the Confucian system contains moral force sufficient to produce Stoic scholars of this type, the nation has no cause to despair of its future. We make no apology for insisting on the claims of these two men to our grateful admiration, or for reproducing their last Memorials, in which they warned the Old Buddha of her folly, and, by denouncing the Boxers, braved all the forces of anarchy and savagery which surged about the Dragon Throne. Already their good name stands high in the esteem of their countrymen. *Et prevalebit*: their courage and unselfish patriotism have been recognised by their canonisation in the Pantheon of China's worthies, under an Edict of the present Regent.

Shortly after their execution the following circular letter *pour faire part* was addressed by the sons of Yüan Ch'ang to the relatives and friends of the family:—

Notice sent by the Yüan family to their relatives regarding the death of Yüan Ch'ang, September, 1900.

After the usual conventional formulæ of grief and self-abasement, this circular letter proceeds as follows:—

> “We realise that it was because of his outspoken courage in resisting the evil tendencies of the times that our parent met his untimely death, and we now submit the following report of the circumstances for the information of our relatives and friends.
>
> “When, in the 5th Moon of this year, the Boxer madness commenced, our late father, in his capacity as a Minister of the Foreign Office, felt extremely anxious in regard to the situation, and his anxiety was shared by his colleague, Hsü Ching-ch’eng. On three occasions when the Princes and Ministers were received in audience, my father expressed his opinion to the Throne that the Boxers were utterly unreliable. ‘I have been in person,’ he said, ‘to Legation Street, and have seen the corpses of Boxers lying on all sides. They had most certainly been shot, proving that their unholy rites availed them nothing. They should be exterminated and not used as Government forces.’ On hearing this advice, the Emperor, turning to Hsü Ching-ch’eng, enquired whether China is strong enough to resist the foreigners or not, and other questions bearing on the position of the Foreign Powers abroad. Hsü replied without hesitation that China was far too weak to think of fighting the whole world. His Majesty was so much impressed by what he had heard that he caught hold of Hsü by the sleeve and seemed much distressed. Hsü sorrowfully left the presence, and proceeded with our father to draft the first of their joint Memorials.
>
> “Later on, when the bombardment of the Legations was in full swing, our father observed to Hsü, ‘This slaughtering of Envoys is a grave breach of all international law. If the Legations are destroyed and the Powers then send an expedition to avenge them, what will become of our country? We must oppose this folly, you and I, even at the risk of our lives.’ So they put in their second Memorial, which never