TypeScript
Final Report

Practical Course:
Contributing to an Open-Source Project

Jonas Hübotter
jonas.huebotter@tum.de

Technical University Munich
March 16, 2021

Contents

1. Introduction **1**

2. Contributions **2**
2.1. Error reporting 2
2.2. Type safety 2
2.3. Type inference 2
2.4. Other work 3

3. Communication **3**
3.1. Quantitative analysis 4

4. Case study **6**
4.1. Development environment 6
4.2. Development process 7
4.3. Iteration on the public pull request 8
4.4. Summary and comparison to other contributions 9

5. Feedback and suggestions **10**

6. Conclusion and reflection **11**

A. Pull request overview **18**

1. Introduction

TypeScript[1] is a programming language that extends JavaScript by adding types. It is funded by Microsoft and primarily developed by a dedicated team of Microsoft engineers. TypeScript is one of Microsoft's first ventures into open-source. Development on TypeScript began privately at Microsoft in 2010 [26]. With the first release in 2012, TypeScript was made available freely under the Apache-2.0 License[2]. Initially, the code was hosted on Microsoft's forge website CodePlex before being moved to GitHub in 2014 [4]. New feature releases of TypeScript are published every other month [5].

Today, the TypeScript GitHub repository[3] has more than 69 thousand stars making it the 20th most starred repository on GitHub [52]. The `typescript` package on NPM has over 19 million weekly downloads [57]. With more than 21 thousand dependent packages, TypeScript is the 24th most depended-upon package on NPM [50]. According to GitHub, almost 3.2 million public repositories depend on GitHub [56].

TypeScript's genesis was the surge in JavaScript development in the late 2000s as JavaScript runtimes became increasingly efficient. With the rise of single-page applications and Node.js supporting large JavaScript backends, the development and maintenance of large-scale JavaScript projects became a challenge [3]. Anders Hejlsberg, one of TypeScript's lead developers, argued that refactoring becomes practically impossible using a programming language without static types and strong static analysis [8].

The TypeScript project has particular significance as it marks a turning point in Microsoft's attitude towards open-source. The former CEO of Microsoft, Steve Ballmer, famously compared copyleft licenses to a "cancer that attaches itself in an intellectual property sense to everything it touches" [1] and complained about a lack of accountability in open-source software [2]. However, In recent years with the success of TypeScript and other open-source efforts at Microsoft, this attitude has changed. Today, the company is the single most significant contributor to open-source in the world [25]. According to Hejlsberg, the team developing TypeScript at Microsoft knew that they were only "going to appeal to the JavaScript community ... by being open source", but also noted that at the time, Microsoft was" very ambivalent" about open-source and even" afraid" of it [26].

In this report, I critically reflect on my contributions and my communication with members of the TypeScript team and external contributors. Then, I examine systemic features of the development process on an individual and a project level. The report ends with suggestions towards increasing accessibility and engagement to grow community support and contributions further.

[1] TypeScript website
[2] TypeScript license
[3] TypeScript repository

2. Contributions

This section gives a brief overview of all my contributions to TypeScript. My contributions can be divided into three broad categories: contributions towards *more accurate and concise error reporting*, contributions towards *increased type safety* by strengthening existing or adding new type checks, and lastly, contributions towards *better type inference*. During the practical course, I used a project board to track my work [47]. A complete overview of my pull requests is shown in Appendix A.

2.1. Error reporting

- I worked on an improvement that reduced the error message's length when a type is not assignable to a type parameter [41, 37].
- I also improved the error message when one should omit the `typeof` operator by making it more specific [38]. I published a pull request that implemented the improvement less than a day after the original issue was created and opened to external contributors [29]. Ultimately, we decided not to merge my changes as the creator of the original issue, Daniel Rosenwasser, realized that there was already another ongoing effort that prevented this error in the first place [39].

2.2. Type safety

- The very first issue I worked on was an improvement to the type checks for the `in` operator[4] [22]. Because it was a breaking change of type checks on a relatively common operator, we had to make quite a few design decisions [17]. I discuss my work and conversations with the TypeScript team in greater detail as part of the case study in Section 4.
- I worked on an improvement that expanded the checks for uncalled functions to all operands of disjunctions [7, 36]. Previously, the compiler checked only the rightmost argument of a disjunction.
- Initially, my pull request mentioned in the previous point also expanded uncalled function checks to negations. Following the request of a reviewer [42], I extracted this improvement into a separate issue [34] and pull request [35].

2.3. Type inference

- When indexing the intersection type where one member is a tuple with an index that exceeded that tuple's length, the resulting type is expected to be `undefined`. I fixed a bug that led the resulting type of such an indexing operation

[4] `in` operator

to be the union over all members of the tuple type, effectively treating the tuple as an infinitely long array [44, 32].

- I also worked on a more significant undertaking that improved the type inference for generic mapped types[5] [51, 46]. Nathan Fenner, an external contributor, supported this effort by reviewing my pull request and suggesting changes [45].

2.4. Other work

- While first familiarizing myself with TypeScript's development environment, I noticed that the documentation on inspecting changes to the test baselines in the contributing guidelines was outdated [49]. I, therefore, created an issue and a pull request to update them [18, 19].

- I also worked on an issue improving the specificity of the event type of the `readystatechange` event [20]. Even though the original issue was tracked in the TypeScript repository, I realized the change had to be made in the `TypeScript-DOM-lib-generator` repository[6] while working on this improvement. This the repository where the DOM types are specified. My pull request was not merged due to a design decision trading performance for accuracy [43].

3. Communication

I now discuss my communication with other contributors throughout the contributing process. To that end, I first debate general aspects of communication alongside specific examples. This is followed in Section 3.1 by a quantitative analysis of the iteration speed of my contributions.

The public documents like the contribution guidelines are very concise, up-to-date, and cover the entire contribution process. Consequently, there was no point where I was unsure how to act as a contributor. I only found one passage on comparing test baselines that could be improved [18]. Likewise, the direct communication was always concise, inclusive, and either friendly or neutral. This was true regardless of whether communicating with external or internal contributors. I did not come across any violation of Microsoft's Code of Conduct that remained unaddressed[7].

Moreover, direct conversations always communicated intent clearly and efficiently. An excellent example of this is Daniel Rosenwasser's comment on my pull request, which had to be closed as there was already another ongoing effort solving the issue:

[5] Mapped Types
[6] `TypeScript-DOM-lib-generator` repository
[7] Microsoft Code of Conduct

> Thanks for the PR @jonhue! However, since I filed that issue yesterday, I've realized that maybe #24738 is a better direction, where the type-checker will automatically promote well-known symbols to being unique. If we merge this in, we may have to back-out this change afterwards. [39]

Due to sometimes infrequent responses examined further in Section 3.1, I occasionally considered reaching out privately to the TypeScript team members. However, I ultimately decided to keep all communication public since we concluded in our meeting on creating a FLOSS project that in open-source projects, public communication is generally preferable to private communication [23, 31].

3.1. Quantitative analysis

This section gives an overview of the iteration speed of my contributions and shows how specific features of issues and their related pull requests influence their iteration speed.

3.1.1. Method

In general, three features are used to differentiate between issues:

- An issue is *approved* once it was confirmed and labeled by a member of the TypeScript team. If it was labeled with `help wanted`, external contributors may submit pull requests [49]. At this stage, issues are commonly also assigned to a milestone.
- An issue is *scheduled* (or a milestone bug) once it was assigned to a milestone that is not the `Backlog` milestone. This may be an upcoming release of TypeScript, like in the case of the issue examined in Section 4.
- An issue is *assigned* once it was assigned to a member of the TypeScript team. Commonly all scheduled issues are also assigned. Unscheduled issues are sometimes assigned too, but usually only after a pull request was opened.

In the following, I refer to these features as *states*. Once a pull request was opened, a bot automatically shares the states of the issue addressed by the pull request with the pull request. We can therefore extend these issue states to their related pull requests.

The main quantities examined are the time between meaningful actions (*response time*) and the percentage of meaningful actions at all responded to (*response rate*). The *total response rate* refers to the percentage of threads (issues and pull requests) where all actions have been responded to. Time differences are measured in days. A *meaningful action* could be either a group of messages, a review, or updating some pull request characteristic like adding a label or moving the pull request within a

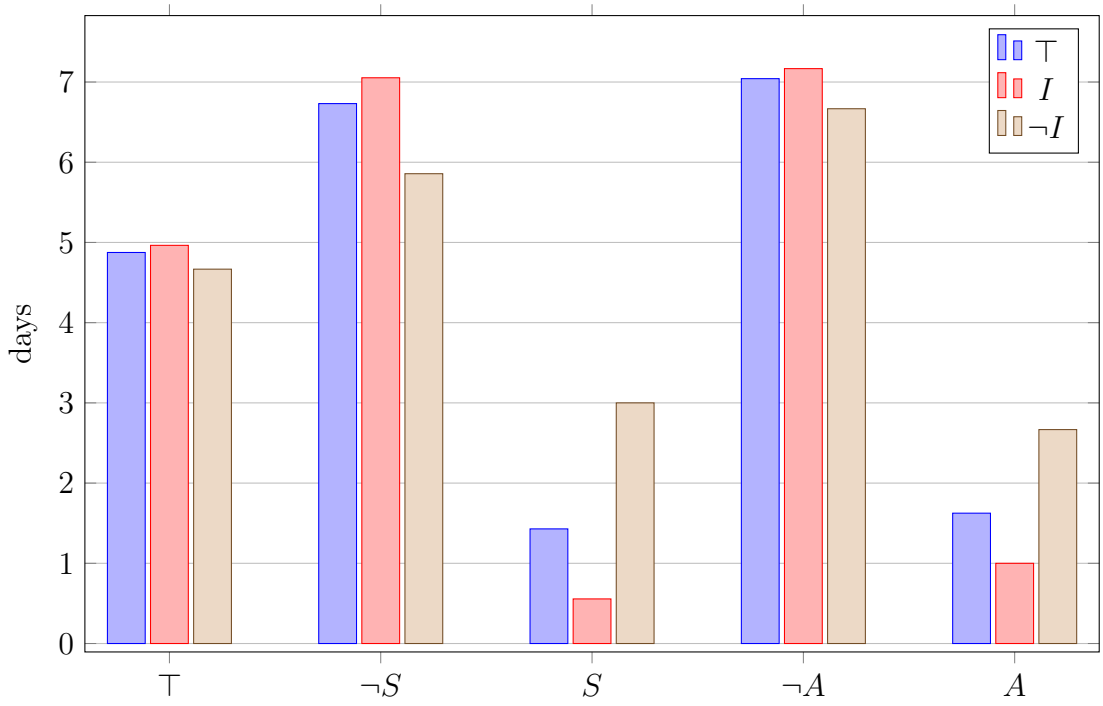

Figure 1: Average response time by state
($\top \sim$ any, $S \sim$ scheduled, $A \sim$ assigned, $I \sim$ immediate)

project board[8]. Further, meaningful actions are divided into *immediate* responses to the pull requests author and remaining responses.

3.1.2. Results

The average response time among all 40 analyzed responses was 4.9 days and is shown in Figure 1. 8 responses of issues were analyzed with an average response time of 2.1 days. In contrast, the remaining 32 responses to pull requests had an average response time of 5.5 days which was even higher when only counting the 29 responses by the TypeScript team (6 days). Despite the small sample size, it can be concluded that there is a significant disparity in how fast the TypeScript team reacts to issues when compared to pull requests which are likely explained by an effort to prevent the build-up of a backlog of issues.

Further, it can be seen that there is no substantial difference in iteration speed between scheduled and assigned tickets, with around 1.5 days on average between responses. On the other hand, unscheduled and unassigned tickets perform significantly worse, with an average response time of more than seven days and almost nine days for pull requests. So on average, the iteration speed of scheduled and assigned tickets is about five times faster.

[8]The complete list of meaningful actions

The average response time of the 28 immediate responses (5 days) is slightly higher than the remaining responses' average response time (4.7 days). However, a fascinating observation is that the immediate responses (0.6 days and one day, respectively) to scheduled and assigned tickets are faster than the average response (1.4 days and 1.6 days, respectively). This reverses when considering unscheduled tickets where immediate responses on average take more than a day longer. Likewise, for unassigned tickets, immediate responses take longer on average than other responses.

One attempt at explaining this asymmetry leads to the following insight: The non-immediate responses are influenced less by the ticket status than immediate responses as they have a higher dependency on internal development processes of the TypeScript team. Once a team member is assigned to a ticket or a ticket was scheduled, there appears to be a considerably higher perceived responsibility incentivizing quicker responses. For unscheduled and unassigned tickets, once they are close to being merged, the internal processes do not take as much longer as the reduced perceived responsibility increases the immediate response time.

The average response rate of the 16 examined threads was 81%, while the total response rate was 63%. On average, the response rate was slightly higher in pull requests than in issues (83% compared to 78%). On the other hand, the total response rate was higher in issues than in pull requests (71% compared to 56%). This discrepancy is explained by the higher number of overall actions in pull requests that diminish the effect of single actions that did not receive a response.

Pull requests were open an average of 30.4 days before they were closed, assuming open pull requests were closed on the 16th of March 2021. Of the nine pull requests considered, four were closed. On average, my pull requests received 9.8 comments. Totaling 44 comments, the longest conversation revolved around strengthening type checks for the `in` operator [22], which is discussed in the following Section 4.

4. Case study

In this case study, I examine the work process and communication with the TypeScript team, as exemplified by my work on improving the type checks for the `in` operator [22]. I begin by describing the TypeScript development environment, which is an essential part of the contribution process. Then, I discuss specific aspects of this particular contribution and how my approach generalized to other contributions. Subsequently, in examining the iteration process, I debate the communication as part of the pull request and the design decisions made along the way. Finally, I summarize how this contribution compared to my other contributions.

4.1. Development environment

TypeScript is bootstrapped; that is, TypeScript itself is written in TypeScript. Therefore, cloning, setting up development tools, and installing dependencies is fast and

well-documented [49]. I examine two main aspects of the development environment: the development environment when working on the source files of TypeScript and when writing tests.

The core of the TypeScript compiler is implemented in relatively few files in `src/compiler`. The fundamental parts like scanning, parsing, type-checking, and code generation essentially live in single isolated files. Therefore, these files are rather large. The type-checking and type-inference algorithm, which was the subject of most of my contributions, is implemented in `checker.ts` which spans more than 41 thousand lines of code [54]. This file and code structure comes with some drawbacks that are discussed further in Section 5. However, one immediate consequence is that regardless of which particular editor is used, running static analysis on an edited file is very slow.

The tests can be divided into two categories. Some automated tests are run on every single commit and are part of the TypeScript repository [49], and there are user tests that run on actual TypeScript implementations. These latter tests are invoked on pull requests as needed by TypeScript team members and run by a bot [53].

A test case is a single TypeScript source file. When a test case is run, TypeScript generates multiple files, including the generated JavaScript, the inferred type of every expression, and the errors produced by the compiler [49]. These files are tracked by Git under `tests/baselines/reference`. A local test run produces updated baselines in `tests/baselines/local` which can then be compared to the tracked baselines and accepted if the differences are valid [49]. A launch configuration for VS Code is also provided, which can be used to launch test cases directly from the editor [49]. Possible improvements to this test system are discussed in Section 5.

4.2. Development process

Now, I discuss the specifics of the given issue and how I implemented the first solution in greater detail. The problem was to ensure that the right operand of the `in` operator is not a primitive type at runtime as the ECMAScript[9] spec requires this operand to be an object [48]. If a primitive type is given, a runtime error is thrown. There are, however, a multitude of nuances between allowing any value as the right operand and absolute insurance that the right operand can never be primitive. The compiler already caught the case where the right operand is a primitive literal (like in the expression `key in 42`). The ticket was aimed at detecting when a value was used as the right operand whose type *could* be primitive, for example, because its type was an unconstrained type parameter.

This specific issue already had an ongoing conversation where Andrew Branch, the TypeScript team member assigned to the issue, already indicated which function a potential fix would need to change [10]. This was incredibly helpful. For other issues, the main difficulty was determining *where* a change should go rather than *what* the

[9]ECMAScript is the programming language specification that is implemented by JavaScript [15]

change should be. I generally began by writing a test case that reproduced the problem by throwing an error (all problems affecting type inference or type checking can be specified in such a way). Then, I was able to identify where this error is added in the type checker and finally use call traces to determine where the change had to be made. Naturally, this approach works better with type checking problems where the problematic area is *closer* to the place where the error is added. However, this approach is far from ideal for type inference problems where the problematic area may be very far away from this place. I encountered this problem while working on a ticket that aimed to fix a discrepancy in the handling of a type and its corresponding single-constituent union type in scenarios where this type was inferred from a concrete generic type [33].

Once I determined the problematic area, I made changes until the reproducible test case behaved as expected. Then I ran the code on an increasing number of tests. I quickly realized how delicate the behavior of the type inference and type checking code is. Because the type checker and type inference algorithm are very interrelated and there are many different contexts in which code is run, even a minimal change can have a tremendous effect on the compiler's behavior. For example, when I restricted the type of the right operand of the `in` operator too much, many test cases were failing even if they had not explicitly tested type checking of the `in` operator.

4.3. Iteration on the public pull request

The issue examined here was already assigned to a member of the TypeScript team, namely Andrew Branch, and scheduled for the 4.2 release of TypeScript [22]. As was shown in Section 3.1, this issue could be expected to receive more traction than others.

The TypeScript team tracks pull request reviews with a dedicated `PR Backlog` project board on GitHub, illustrated in Figure 2 [58]. New pull requests are added manually to `Not started`. During the iteration process, pull requests cycle between `Waiting on author` and `Needs review`. Once a pull request is close to a merge and approved by the primary reviewer, it is moved to `Needs merge` from where it is automatically moved to `Done` once it was merged.

During the iteration process of the examined issue, we had to make two crucial design decisions [17]. Namely, we had to find an accurate and concise error message and tune the type check's strictness. The latter, in particular, required multiple rounds of iteration. This lengthy discussion revolved because we had to balance type safety and break as little code as possible. My initial solution worked by ensuring the right operand of an `in` expression was not assignable to a primitive type, like `number` or `string` [16]. A limitation with narrowing type parameters would have made it difficult to enforce these more vigorous type checks in practice [6]. Therefore, members of the TypeScript team suggested a more conservative fix that only ensures that the resolved constraint of the right operand's type is not assignable to a primitive

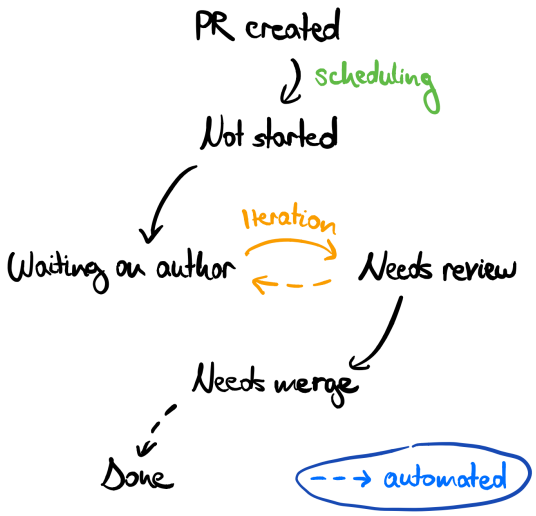

Figure 2: PR Backlog process

[11]. These solutions sound similar at first. However, the latter is a negative check ensuring that the type of the right operand does not explicitly extend a primitive. Simultaneously, the former solution is a positive check that the type of the right operand cannot possibly extend a primitive. Ultimately, the decision to opt for weaker type safety reaffirms TypeScript's focus on usability and unobtrusiveness [24].

A comment by Ryan Cavanaugh, the lead developer of TypeScript, on the expression `typeof val === 'object' && '__isMaybe' in val`, which lead to an error with my initial implementation, exemplifies this discussion:

> This error is *actually correct* — `val` could be `null` — but boy are we going to get bug reports about this. Maybe we need to special-case the error message. [13]

Because this was such a wide-ranging change, we had to widen test coverage in some additional iteration rounds [12]. While opening the pull request, I forgot to allow maintainers to edit my branch. Consequently, my pull request was closed immediately before being merged so that Andrew Branch could make a few final changes to the error message [27].

4.4. Summary and comparison to other contributions

The work on this issue has been delightful. The conversation was always friendly, comments were concise, and intentions were communicated clearly. It was a tremendous help that the reviewers included examples in their comments, like when Andrew

Branch was suggesting to widen the test coverage [12]. Moreover, it was rewarding to see this change featured prominently in the announcement of the 4.2 release [40].

Most importantly, however, the iterations were fast-paced. This gave everyone a sense that this issue was moving in the right direction. An average *immediate response time* of 0.7 days compared to an average immediate response time across all contributions of 5 days exemplifies this observation. The *response rate* was 100% compared to an overall response rate to pull requests of 83% (see Section 3.1).

5. Feedback and suggestions

A well-documented and precise contribution process is perhaps the most critical feature of an open-source repository regarding reducing friction for contributors. This process already works very well within the TypeScript project today. One specific feature of how contributions are handled is that incoming issues are processed rapidly and labeled accurately with up to 127 labels [55]. If an issue does not meet the criteria outlined in the contribution guidelines or will not be worked on, it is closed quickly. This policy is very effective in reducing noise.

Additionally, the number of labels and their active usage make issues and pull requests very searchable. All of the above means in practice that one rarely finds an issue that is not still prevalent, and it is easy to restrict searches to a very narrow domain of interest. Further, by requiring the use of the TypeScript playground[10], it is straightforward for contributors to reproduce issues and determine whether they are a regression.

Anders Hejlsberg claimed in an interview in 2019 that since its move to GitHub in 2014, TypeScript has not only been open-source but also doing its "entire development process in the open" [9]. While it is true that most conversations around the development of tracked issues are open, feature-planning is still done privately by the TypeScript team. Opening this process, for example, by allowing the community to vote on new features, would almost certainly increase community engagement. Additionally, as of now, almost all new features are implemented by the TypeScript team. Alongside a more transparent tracking of planned features, opening the development of new features to the community could further increase the contributor excitement. In an interview last year, Ryan Cavanaugh acknowledged this:

> The main challenge we see there is that adding features to the TypeScript codebase is actually easy and fixing bugs is really hard. People are a lot more excited to add features than fix bugs because it's fun. Who can blame them? Figuring out what we can do about that and encouraging people to help us out on the things we need more help with would be a community challenge. [24]

[10] TypeScript playground

Starting to work on a project as big as the TypeScript compiler might seem somewhat daunting for many potential contributors. This is where mentoring and a place to ask questions can help. In our conversation with Stephan Kulla on his experience with Serlo, we learned that onboarding people to a large project and continuously mentoring them increases their excitement about their work [21]. The TypeScript Discord[11] is offering precisely that, but it could be featured more prominently in the project's Readme and Contributing Guidelines.

As Ryan Cavanaugh also pointed out in the mentioned interview, "rather than growing the team, [they] just hope to grow the community to support TypeScript" [24]. To support a growing community, however, the processes must sustain a growing number of contributions without growing the team. With the discussed PR Backlog serving as an example, the team already has such processes in place [58]. As shown in Section 3.1, for some contributions, interactivity is relatively low. Increasing the iteration speed would likely increase the happiness amongst contributors. To that end, one first potential step could be to automate more steps of the PR Backlog workflow by extending the already existing TypeScript bot to automatically move pull requests within the PR Backlog project once an author addressed review feedback.

Finally, to increase contributions, it is vital to lower the barrier to entry and reduce friction during development. Especially for a large and arguably complicated project like TypeScript, increasing code quality is critical. Besides, a document that explains each module's purpose would make it easier for first-time contributors to understand where to start. The most significant potential of making development easier lies with improving debugging and the test performance because (as discussed in Section 4.2) this is where contributors spent the most time during development. Nonetheless, especially the latter is difficult to improve. This is because TypeScript relies on a large number of behavioral tests to test for correct behavior. Each pull request and fixed issue adds a new behavioral test, but only very rarely are tests removed. As there is a potentially infinite number of behavioral tests, defining a strategy of limiting test execution will become increasingly important. Viable approaches might be better structuring tests to reduce duplication or extracting more classes of rarely broken tests to be only run once before a merge.

6. Conclusion and reflection

Throughout the past months, I found it very enjoyable to work on TypeScript. I had many interesting conversations with both members of the TypeScript team and other external contributors. It was fascinating and gratifying to work on features like type checking and type inference that are immensely interrelated and eventually see my contributions improve the static analysis used in the development of many programs.

Often, infrequent conversations made precise communication even more critical.

[11] TypeScript Discord

While not being great in that regard in the beginning, I improved that aspect over the past months. Inspired by an article from developers at Netlify, I started to review my pull requests immediately after opening them and revisited them once a team member was assigned to ensure that my comments were understandable [14]. While working with uncalled function checks, I also learned that it is best to keep the scope of pull requests to an absolute minimum to reduce the number of iterations in almost all cases.

I certainly plan to continue contributing to TypeScript, particularly in the area of type narrowing. Recently, Anders Hejlsberg opened a pull request addressing the problem with narrowing generic union types, an issue that I was also looking into over the past months and that was mentioned previously in Section 4.3 [6, 28]. Building on this, I opened an issue to improve further the strictness of type checks on the `in` operator, which I am excited to continue to work on [30].

Over the last few years, TypeScript exemplified Microsofts journey towards more open-source development. TypeScript has come very far from where it started when published in 2012, but it is still on its journey towards a development process that is entirely open. In the coming years, it will be necessary to increase community participation further. As one of the most significant open-source projects with growing adoption and a growing number of people wanting to contribute, it remains essential to update development processes to sustain that growth.

References

[1] Thomas C Greene. *Ballmer: 'Linux is a cancer'*. June 2001. URL: https://www.theregister.com/2001/06/02/ballmer_linux_is_a_cancer/ (visited on 03/07/2021).

[2] Joe McKendrick. *Ballmer: Open Source is Not Trustworthy*. Oct. 2003. URL: https://rcpmag.com/articles/2003/10/22/ballmer-open-source-is-not-trustworthy.aspx (visited on 03/07/2021).

[3] Mary Jo Foley. *Who built Microsoft TypeScript and why*. Oct. 2012. URL: https://www.zdnet.com/article/who-built-microsoft-typescript-and-why/ (visited on 03/07/2021).

[4] Jonathan Turner. *New Compiler and Moving to GitHub*. July 2014. URL: https://devblogs.microsoft.com/typescript/new-compiler-and-moving-to-github/ (visited on 03/07/2021).

[5] Daniel Rosenwasser. *TypeScript's New Release Cadence*. Mar. 2017. URL: https://devblogs.microsoft.com/typescript/typescripts-new-release-cadence/ (visited on 03/07/2021).

[6] Kevin Ryan. *Generics extending unions cannot be narrowed - TypeScript Issue #13995*. Feb. 2017. URL: https://github.com/microsoft/TypeScript/issues/13995 (visited on 03/07/2021).

[7] Matt Bierner. *Uncalled function checks only works with single conditional - TypeScript Issue #35584*. Dec. 2019. URL: https://github.com/microsoft/TypeScript/issues/35584 (visited on 03/07/2021).

[8] David Cassel. *A Conversation with the Creators Behind Python, Java, TypeScript, and Perl*. Apr. 2019. URL: https://thenewstack.io/a-conversation-with-the-creators-behind-python-java-typescript-and-perl/ (visited on 03/07/2021).

[9] Anders Hejlsberg. *BEHIND THE CODE: The one who created languages*. Feb. 2019. URL: https://youtu.be/tmOmFfcA9us?t=625 (visited on 03/15/2021).

[10] Andrew Branch. *'in' should not operate on primitive types - TypeScript Issue #41317 (comment)*. Dec. 2020. URL: https://github.com/microsoft/TypeScript/issues/41317#issuecomment-740121850 (visited on 03/07/2021).

[11] Andrew Branch. *'in' should not operate on primitive types - TypeScript Pull Request #41928 (comment)*. Dec. 2020. URL: https://github.com/microsoft/TypeScript/pull/41928#pullrequestreview-551865735 (visited on 03/07/2021).

[12] Andrew Branch. *'in' should not operate on primitive types - TypeScript Pull Request #41928 (comment)*. Dec. 2020. URL: https://github.com/microsoft/TypeScript/pull/41928#issuecomment-749195594 (visited on 03/07/2021).

[13] Ryan Cavanaugh. *'in' should not operate on primitive types - TypeScript Pull Request #41928 (comment)*. Dec. 2020. URL: https://github.com/microsoft/TypeScript/pull/41928#issuecomment-743356094 (visited on 03/07/2021).

[14] Leslie Cohn-Wein, Kristen Lavavej, and swyx. *Feedback Ladders: How We Encode Code Reviews at Netlify*. Mar. 2020. URL: https://www.netlify.com/blog/2020/03/05/feedback-ladders-how-we-encode-code-reviews-at-netlify/ (visited on 03/15/2021).

[15] ECMA. *ECMA-262*. June 2020. URL: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/ (visited on 03/11/2021).

[16] Jonas Hübotter. *'in' should not operate on primitive types - TypeScript Pull Request #41928 (comment)*. Dec. 2020. URL: https://github.com/microsoft/TypeScript/pull/41928#issue-536763325 (visited on 03/07/2021).

[17] Jonas Hübotter. *'in' should not operate on primitive types by jonhue - TypeScript Pull Request #41928*. Dec. 2020. URL: https://github.com/microsoft/TypeScript/pull/41928 (visited on 03/07/2021).

[18] Jonas Hübotter. *Add information on gulp diff to the contributing guidelines - TypeScript Issue #41991*. Dec. 2020. URL: https://github.com/microsoft/TypeScript/issues/41991 (visited on 03/07/2021).

[19] Jonas Hübotter. *Add information on gulp diff to the contributing guidelines - TypeScript Pull Request #42031*. Dec. 2020. URL: https://github.com/microsoft/TypeScript/pull/42031 (visited on 03/07/2021).

[20] jurosh. *Event type for readystatechange is not specific enough - TypeScript Issue #41775*. Dec. 2020. URL: https://github.com/microsoft/TypeScript/issues/41775 (visited on 03/07/2021).

[21] Stephan Kulla. *Motivation in Open Source Projects. Personal Communication as part of the Open-Source Lab Course*. Dec. 2020. URL: https://www21.in.tum.de/teaching/osp/WS20/index.html#stephan-kulla-motivation-in-open-source-projects.

[22] Pierre-Antoine Mills. *'in' should not operate on primitive types - TypeScript Issue #41317*. Oct. 2020. URL: https://github.com/microsoft/TypeScript/issues/41317 (visited on 03/07/2021).

[23] *Personal Communication as part of the Open-Source Lab Course*. Nov. 2020. URL: https://www21.in.tum.de/teaching/osp/WS20/index.html#tuesday-10.11..

[24] Ryan Donovan. *Talking TypeScript with the engineer who leads the team*. June 2020. URL: https://stackoverflow.blog/2020/06/15/talking-typescript-with-ryan-cavanaugh/ (visited on 03/07/2021).

[25] Tom Warren. *Microsoft: we were wrong about open source*. May 2020. URL: https://www.theverge.com/2020/5/18/21262103/microsoft-open-source-linux-history-wrong-statement (visited on 03/07/2021).

[26] Liam Tung. *TypeScript creator: How the programming language beat Microsoft's open-source fears*. Sept. 2020. URL: https://www.zdnet.com/article/typescript-creator-how-the-programming-language-beat-microsofts-open-source-fears/ (visited on 03/07/2021).

[27] Andrew Branch. *'in' should not operate on primitive types - TypeScript Pull Request #42288*. Jan. 2021. URL: https://github.com/microsoft/TypeScript/pull/42288 (visited on 03/07/2021).

[28] Anders Hejlsberg. *Improve narrowing of generic types in control flow analysis - TypeScript Pull Request #43183*. Mar. 2021. URL: https://github.com/microsoft/TypeScript/pull/43183 (visited on 03/15/2021).

[29] Jonas Hübotter. *'typeof' on a well-known symbol in a computed property name has a poor error - TypeScript Pull Request #42530*. Jan. 2021. URL: https://github.com/microsoft/TypeScript/pull/42530 (visited on 03/07/2021).

[30] Jonas Hübotter. *Ensure 'in' does not operate on primitive types - TypeScript Issue #43210*. Mar. 2021. URL: https://github.com/microsoft/TypeScript/issues/43210 (visited on 03/15/2021).

[31] Jonas Hübotter. *Generics extending unions cannot be narrowed - TypeScript Issue #13995 (comment)*. Jan. 2021. URL: https://github.com/microsoft/TypeScript/issues/13995#issuecomment-766642401 (visited on 03/11/2021).

[32] Jonas Hübotter. *Indexing tuple intersection type beyond length produces unexpected type - TypeScript Pull Request #42602*. Feb. 2021. URL: https://github.com/microsoft/TypeScript/pull/42602 (visited on 03/07/2021).

[33] Jonas Hübotter. *Inferred type is not the same as what's passed in original generic type - TypeScript Issue #42516 (comment)*. Mar. 2021. URL: https://github.com/microsoft/TypeScript/issues/42516#issuecomment-797430614 (visited on 03/15/2021).

[34] Jonas Hübotter. *Uncalled function checks don't work with negation - TypeScript Issue #43096*. Mar. 2021. URL: https://github.com/microsoft/TypeScript/issues/43096 (visited on 03/07/2021).

[35] Jonas Hübotter. *Uncalled function checks don't work with negation - TypeScript Pull Request #43097*. Mar. 2021. URL: https://github.com/microsoft/TypeScript/pull/43097 (visited on 03/07/2021).

[36] Jonas Hübotter. *Uncalled function checks only works with single conditional - TypeScript Pull Request #42835*. Feb. 2021. URL: https://github.com/microsoft/TypeScript/pull/42835 (visited on 03/07/2021).

[37] Jonas Hübotter. *Unnecessary elaboration about not being assignable to type parameters - TypeScript Pull Request #42952*. Feb. 2021. URL: https://github.com/microsoft/TypeScript/pull/42952 (visited on 03/07/2021).

[38] Daniel Rosenwasser. *'typeof' on a well-known symbol in a computed property name has a poor error - TypeScript Issue #42523*. Jan. 2021. URL: https://github.com/microsoft/TypeScript/issues/42523 (visited on 03/07/2021).

[39] Daniel Rosenwasser. *'typeof' on a well-known symbol in a computed property name has a poor error - TypeScript Pull Request #42530 (comment)*. Jan. 2021. URL: https://github.com/microsoft/TypeScript/pull/42530#pullrequestreview-578642953 (visited on 03/07/2021).

[40] Daniel Rosenwasser. *Announcing TypeScript 4.2*. Feb. 2021. URL: https://devblogs.microsoft.com/typescript/announcing-typescript-4-2/#stricter-in-checks (visited on 03/15/2021).

[41] Daniel Rosenwasser. *Unnecessary elaboration about not being assignable to type parameters - TypeScript Issue #42849*. Feb. 2021. URL: https://github.com/microsoft/TypeScript/issues/42849 (visited on 03/07/2021).

[42] Nathan Shively-Sanders. *Uncalled function checks only works with single conditional - TypeScript Pull Request #42835 (comment)*. Mar. 2021. URL: https://github.com/microsoft/TypeScript/pull/42835#issuecomment-788963071 (visited on 03/07/2021).

[43] Orta Therox. *Event type for readystatechange is not specific enough - TypeScript-DOM-lib-generator Pull Request #969 (comment)*. Jan. 2021. URL: https://github.com/microsoft/TypeScript-DOM-lib-generator/pull/969#issuecomment-784344870 (visited on 03/07/2021).

[44] tomblind. *Indexing tuple intersection type beyond length produces unexpected type TypeScript Issue #42557*. Jan. 2021. URL: https://github.com/microsoft/TypeScript/issues/42557 (visited on 03/07/2021).

[45] Nathan Fenner. *[P in keyof T]: T[P] not accepting inferred base type via extends - TypeScript Pull Request #42382 (comment)*.

[46] Jonas Hübotter. *[P in keyof T]: T[P] not accepting inferred base type via extends - TypeScript Pull Request #42382*.

[47] Jonas Hübotter. *Project board*. URL: https://github.com/users/jonhue/projects/2 (visited on 03/07/2021).

[48] *in operator*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/in (visited on 03/07/2021).

[49] Microsoft. *TypeScript Contributing Guidelines*. URL: https://github.com/microsoft/TypeScript/blob/master/CONTRIBUTING.md (visited on 03/07/2021).

[50] *Most depended upon packages*. URL: https://www.npmjs.com/browse/depended (visited on 03/07/2021).

[51] Rodrigo Nascimento. *[P in keyof T]: T[P] not accepting inferred base type via extends - TypeScript Issue #37670*.

[52] *Repositories Ranking*. URL: https://gitstar-ranking.com/repositories (visited on 03/07/2021).

[53] *TypeScript bot*. URL: https://github.com/typescript-bot/TypeScript/pulls (visited on 03/07/2021).

[54] *TypeScript checker.ts*. URL: https://github.com/microsoft/TypeScript/blob/master/src/compiler/checker.ts (visited on 03/07/2021).

[55] *TypeScript labels*. URL: https://github.com/microsoft/TypeScript/labels (visited on 03/15/2021).

[56] *TypeScript Network Dependents*. URL: https://github.com/microsoft/TypeScript/network/dependents (visited on 03/07/2021).

[57] *typescript NPM package*. URL: https://www.npmjs.com/package/typescript (visited on 03/07/2021).

[58] *TypeScript PR Backlog*. URL: https://github.com/microsoft/TypeScript/projects/13 (visited on 03/07/2021).

A. Pull request overview

Tag	Days until first review	Days until close	Comments	Number of participants[12]
`TypeScript#43097`	-	-	0	1
`TypeScript#42835`	13	-	11	2
`TypeScript#42602`	-	-	3	2
`TypeScript#42382`	11	-	14	3
`TypeScript#42952`	9	9	1	2
`TypeScript#42530`	0	6	10	2
`TypeScript-DOM-lib-generator#43097`	1	-	4	3
`TypeScript#42031`	1	17	1	4
`TypeScript#41928`	0	32	44	5

[12]excluding bots