How Do I Track Releases?

Tracking Planning and Development

Each release has a corresponding Iteration Plan issue on GitHub. These iteration plans contain a list of **planned work items** as well as **planned release dates**.

We also keep an updated Feature Roadmap which is typically more accurate in terms of what we release.

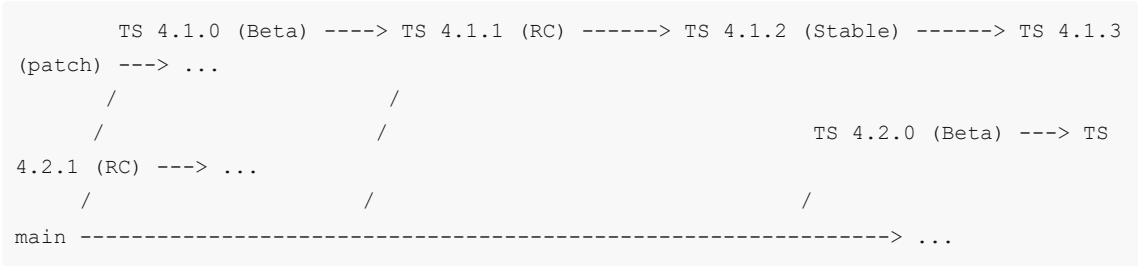
Tracking Actual Releases

The TypeScript team also maintains the official TypeScript Blog where releases are announced.

The official @typescript Twitter account also posts release announcements.

What Are the Stages of a Release?

TypeScript currently has two pre-releases (the Beta and the Release Candidate), followed by a stable "final" release, followed by any number of patches.



How Often Does TypeScript Release?

In general you can expect a release **around every 3 months**. TypeScript's release cadence is relatively regular in this respect, though there may occasionally be some variance due to holidays and other scheduling concerns.

The breakdown across pre-release versions is typically

- Betas are released about 4 weeks after the prior Stable release.
- Release Candidates are released around 6 weeks after the prior Beta.
- Stable releases are released around 2 weeks after the prior Release Candidate.
- Patches are released at any frequency depending on urgency, but we evaluate at the end of every month whether a patch should be published for the current stable release.

What Work Gets Done?

What gets done for the Beta?

During the beta, new features, breaking changes, and bug fixes are prioritized. Breaking changes and features are "front-loaded" so that they can get ample testing in the beta and RC. It also ensures we can roll back changes before there is too much momentum in a release. We strive to get these in as early as possible so that they are available in nightly releases.

We strive not to make breaking changes after the Beta release.

What gets done for the Release Candidate?

Following the beta release, work is done that leads to a Release Candidate (RC). The RC primarily contains bug fixes and often editor features that were not finished in time for the Beta. Editor features are okay to go in here because they are relatively easy to back out in time (given dog-fooding, user expectations, the possibility of flagging, lag in editor rollout, and patch releases).

Once the RC goes out, the team begins focus on the next version of TypeScript.

Very few changes should be expected for a version after the Release Candidate.

What gets done for the Stable/Final release?

High-priority fixes are applied following the Release Candidate for the Stable release.

By default, new bugs that are not regressions from the upcoming or prior version of TypeScript will be addressed in a new minor version following the final release.

What gets done for patch releases?

Patch releases are periodically pushed out for any of the following:

- High-priority regression fixes
- Performance regression fixes
- Safe fixes to the language service/editing experience

These fixes are typically weighed against the cost (how hard it would be for the team to retroactively apply/release a change), the risk (how much code is changed, how understandable is the fix), as well as how feasible it would be to wait for the next version.

Which Version Should I Be Using?

We always appreciate early feedback on new versions of the language and editor support! Depending on how you'd like to test TypeScript and contribute, there are a couple of different options.

Testing Independently

Despite what the name might imply, nightly versions are always the preferred way to test out the current state of TypeScript. While we won't necessarily endorse them for production use, nightlies are always fairly stable and easy to use.

You can download a nightly release via npm (`npm install typescript@next`) and configure your editor support. If you can't commit to updating your build processes yet, but you write code in Visual Studio Code, you can use the TypeScript and JavaScript Nightly extension to at least try out new versions of our editor support.

Testing New Features

If you're willing to try new features, we make Beta releases available. This is often a good period to start providing feedback on new features, and the closer to the beta release that feedback occurs, the easier we can address it before it officially is added to the language.

The longer you wait after a beta, the less actionable your feedback will be. In those cases, we encourage you to switch to the nightly releases.

Testing for High-Priority Bug Fixes

If you are extremely limited in your ability to provide feedback, the Release Candidate is the most stable pre-release we provide. Feedback on the Release Candidate can be addressed, but it is typically limited to high-priority bug fixes, not fundamental changes.

If you intend to provide feedback that will fundamentally affect the design of the language, it should ideally be provided before the RC. If that is the case, do not wait until the release candidate. Use nightly releases to test on your own or within your team/organization!

Staying Up-to-Date

If you can't commit any feedback, just stay up to date on the latest version of TypeScript. Dedicate a day or two every 3 months or so to perform the upgrade.

The following section is targeted more at maintainers and contributors! If you're simply consuming TypeScript, it is probably not relevant.

Release Mechanics

The TypeScript team develops around one central branch: `main`. This branch is used for nightly builds, and is the source of truth. This central branch is always meant to build cleanly.

The Typical Release Schedule

The typical release schedule for a version `X.Y` looks like

1. Create X.Y Beta (X.Y.0) Build for Testing
2. **TypeScript X.Y Beta Release**
3. Create X.Y RC (X.Y.1) Build for Testing
4. **TypeScript X.Y RC Release**
5. Create X.Y Final (X.Y.2) Build for Testing
6. **TypeScript X.Y Final Release**
7. Then every week evaluate whether the following needs to be done:
    1. Cherry-pick prioritized new work into the X.Y branch
    2. Release a new patch version

How Branching Works Around Releases

What does the team have to do to achieve that typical release schedule?

1. Development always just occurs on the `main` branch. This is the default assumption.
2. When we need to create the beta build for TypeScript X.Y, we create a branch called `release-X.Y` and bump the version to `X.Y.0-beta`. On npm, this is published with `--tag beta`.
3. Development for TypeScript X.Y **continues on the `main` branch**.

4. When we need to create an RC build for TypeScript X.Y, we merge the `main` branch into `release-X.Y` and bump the version to `X.Y.1-rc`. On npm, this is published with `--tag rc`.
5. After the RC goes out, the assumption is that all work in `main` will go into TypeScript X.(Y + 1). **Any critical changes will need to be cherry-picked to `release-X.Y`**.
6. When we need to create a build for the stable release version of TypeScript X.Y, we bump the version to `X.Y.2` (with no pre-release version string or tag). On npm, this is published with `--tag latest`.

The Comment Command Sequence

Much of this process is automated by Triggering @typescript-bot to perform tasks, along with a few GitHub actions. Typically, commands to the bot are given in the Iteration Plan comments of a release. The commands roughly occur in the following order:

1. Readying the Beta
    1. `@typescript-bot create release-X.Y` (create the branch)
    2. In the event that changes need to come in after:
        1. `@typescript-bot sync release-X.Y`
        2. Run Update LKG on `release-X.Y`.
2. Readying the RC
    1. `@typescript-bot sync release-X.Y` (sync `main` to `release-X.Y`)
    2. `@typescript-bot bump release-X.Y` (update the version number)
    3. In the event that changes need to come in after:
        1. `@typescript-bot sync release-X.Y`
        2. Run Update LKG on `release-X.Y`.
3. Readying the Stable Release
    1. `@typescript-bot bump release-X.Y` (update the version number)
    2. On relevant PRs early on, run `@typescript-bot cherry-pick this to release-X.Y`
    3. On PRs that look like they will be the last cherry-pick: `@typescript-bot cherry-pick this to release-X.Y and LKG`
    4. Run Update LKG on `release-X.Y` when necessary.

Release Tasks

Every publish, especially the Beta, RC, and Stable releases, must undergo a set of release activities. These release activities are documented here.

Publishing

The publishing process is largely internal at the moment, but it is also largely uninteresting.

The most interesting portion of this is the fact that a Visual Studio build is created, typically 2-3 business days prior to a release, so that a testing team can perform a validation pass. This validation pass sometimes finds regressions in both the core TypeScript experience as well as the TypeScript/JavaScript editing experience.

FAQ

Why do you need to set the patch version on pre-releases?

I think some weirdness around publishing extensions to the Visual Studio marketplace. To accomodate it and stay reasonable, we just version consistently across npm, NuGet, and VS Marketplace.