5 Release Channels

These docs are old and won't be updated. Go to react.dev for the new React docs.

See Versioning Policy to learn about the React release channels.

React relies on a thriving open source community to file bug reports, open pull requests, and submit RFCs. To encourage feedback we sometimes share special builds of React that include unreleased features.

This document will be most relevant to developers who work on frameworks, libraries, or developer tooling. Developers who use React primarily to build user-facing applications should not need to worry about our prerelease channels.

Each of React's release channels is designed for a distinct use case:

- **Latest** is for stable, semver React releases. It's what you get when you install React from npm. This is the channel you're already using today. **Use this for all user-facing React applications.**
- **Next** tracks the main branch of the React source code repository. Think of these as release candidates for the next minor semver release. Use this for integration testing between React and third party projects.
- **Experimental** includes experimental APIs and features that aren't available in the stable releases. These also track the main branch, but with additional feature flags turned on. Use this to try out upcoming features before they are released.

All releases are published to npm, but only Latest uses semantic versioning. Prereleases (those in the Next and Experimental channels) have versions generated from a hash of their contents and the commit date, e.g. `0.0.0-68053d940-20210623` for Next and `0.0.0-experimental-68053d940-20210623` for Experimental.

The only officially supported release channel for user-facing applications is Latest. Next and Experimental releases are provided for testing purposes only, and we provide no guarantees that behavior won't change between releases. They do not follow the semver protocol that we use for releases from Latest.

By publishing prereleases to the same registry that we use for stable releases, we are able to take advantage of the many tools that support the npm workflow, like unpkg and CodeSandbox.

Latest Channel

Latest is the channel used for stable React releases. It corresponds to the `latest` tag on npm. It is the recommended channel for all React apps that are shipped to real users.

If you're not sure which channel you should use, it's Latest. If you're a React developer, this is what you're already using.

You can expect updates to Latest to be extremely stable. Versions follow the semantic versioning scheme. Learn more about our commitment to stability and incremental migration in our versioning policy.

Next Channel

The Next channel is a prerelease channel that tracks the main branch of the React repository. We use prereleases in the Next channel as release candidates for the Latest channel. You can think of Next as a superset of Latest that is updated more frequently.

The degree of change between the most recent Next release and the most recent Latest release is approximately the same as you would find between two minor semver releases. However, **the Next channel does not conform to semantic versioning.** You should expect occasional breaking changes between successive releases in the Next channel.

Do not use prereleases in user-facing applications.

Releases in Next are published with the `next` tag on npm. Versions are generated from a hash of the build's contents and the commit date, e.g. `0.0.0-68053d940-20210623`.

Using the Next Channel for Integration Testing

The Next channel is designed to support integration testing between React and other projects.

All changes to React go through extensive internal testing before they are released to the public. However, there are a myriad of environments and configurations used throughout the React ecosystem, and it's not possible for us to test against every single one.

If you're the author of a third party React framework, library, developer tool, or similar infrastructure-type project, you can help us keep React stable for your users and the entire React community by periodically running your test suite against the most recent changes. If you're interested, follow these steps:

- Set up a cron job using your preferred continuous integration platform. Cron jobs are supported by both CircleCI and Travis CI.
- In the cron job, update your React packages to the most recent React release in the Next channel, using `next` tag on npm. Using the npm cli:
  ```
  npm update react@next react-dom@next
  ```
  Or yarn:
  ```
  yarn upgrade react@next react-dom@next
  ```
- Run your test suite against the updated packages.
- If everything passes, great! You can expect that your project will work with the next minor React release.
- If something breaks unexpectedly, please let us know by filing an issue.

A project that uses this workflow is Next.js. (No pun intended! Seriously!) You can refer to their CircleCI configuration as an example.

Experimental Channel

Like Next, the Experimental channel is a prerelease channel that tracks the main branch of the React repository. Unlike Next, Experimental releases include additional features and APIs that are not ready for wider release.

Usually, an update to Next is accompanied by a corresponding update to Experimental. They are based on the same source revision, but are built using a different set of feature flags.
Experimental releases may be significantly different than releases to Next and Latest. **Do not use Experimental releases in user-facing applications.** You should expect frequent breaking changes between releases in the Experimental channel.
Releases in Experimental are published with the `experimental` tag on npm. Versions are generated from a hash of the build’s contents and the commit date, e.g. `0.0.0-experimental-68053d940-20210623`.

What Goes Into an Experimental Release?

Experimental features are ones that are not ready to be released to the wider public, and may change drastically before they are finalized. Some experiments may never be finalized — the reason we have experiments is to test the viability of proposed changes.

For example, if the Experimental channel had existed when we announced Hooks, we would have released Hooks to the Experimental channel weeks before they were available in Latest.
You may find it valuable to run integration tests against Experimental. This is up to you. However, be advised that Experimental is even less stable than Next. **We do not guarantee any stability between Experimental releases.**

How Can I Learn More About Experimental Features?

Experimental features may or may not be documented. Usually, experiments aren’t documented until they are close to shipping in Next or Latest.
If a feature is not documented, they may be accompanied by an RFC.
We will post to the React blog when we’re ready to announce new experiments, but that doesn’t mean we will publicize every experiment.
You can always refer to our public GitHub repository’s history for a comprehensive list of changes.