RedwoodJS Contributing Workshop 🚀

Resources

Community

👉 This is your next step! There's a vibrant, helpful community standing by to guide you toward your next step, whatever it may be:

- Introduce yourself — just say hi!
- Say that you're interested in contributing and, if you know, what you'd like to help with.
- Ask a question — how can people help you take the next step?

Forums
https://community.redwoodjs.com

Chat
https://discord.gg/jjSYEQd

Start with the #contributing categories in each.

Contributing Guides and Docs

This is the material we covered in the Workshop.

Contributing Doc + Guide: Start Here
https://redwoodjs.com/docs/contributing

How to Contribute to the Framework Packages: This is what we covered in the Workshop
redwoodjs/redwood/CONTRIBUTING.md

How to Contribute to the CLI Package: Example of Doc specific to the Package we Updated
redwoodjs/redwood/packages/cli/README.md

Getting Started with Redwood and GitHub (and git)

These are the foundations for contributing, which we didn't cover in the Workshop.

The Redwood Tutorial
The best, and most fun, way to learn Redwood and the underlying tools and technologies

- Part 1: https://redwoodjs.com/tutorial
- Part 2: https://redwoodjs.com/tutorial2/welcome-to-redwood-part-ii-redwood-s-revenge

Redwood Docs
https://redwoodjs.com/docs/introduction

Redwood Cookbooks
https://redwoodjs.com/docs/cookbook

GitHub (and Git)

Understandably, diving into Git and the GitHub workflow can feel intimidating if you haven't experienced it before. There's a lot of great material to help you learn. And, most of all, I'm sure you'll quickly get the hang of it and rockin' commits and pushes with the best of 'em.

- Introduction to GitHub (overview of concepts and workflow)
- First Day on GitHub (including Git)
- First Week on GitHub (parts 3 and 4 might be helpful)

The Characteristics of Contributor

More than committing code, this is about human collaboration and relationship. Our community mantra is "By helping each other be successful with Redwood, we make the Redwood project successful." We have a specific vision for the effect this project and community will have on you — it should give you superpowers to build+create, progress in skills, and help advance your career.

So *who* do you need to become to achieve this? Specifically, what characteristics, skills, and capabilities will you need to cultivate through practice? Here are my suggestions:
- Empathy
- Gratitude
- Generosity

All of these are applicable in relation to both others *and* yourself. The goal of putting them into practice is to create *trust* that will be a catalyst for *risk-taking* (another word to describe this process is "learning"!). Those are the ingredients necessary for productive, positive collaboration.

And you thought all this was just about opening a PR 🤣 Yes, it's a super rewarding experience. But that's just the beginning!

What makes for a good Pull Request?

In general, we don't have a formal structure for PRs. Our goal is to make it as efficient as possible for anyone to open a PR. But there are some good practices, which are definitely flexible. Just keep in mind that after opening a PR there's more to do before getting to the finish line:

1) Reviews from other contributors and maintainers
2) Update code and, after maintainer approval, merge-in changes to the `main` branch
3) Once PR is merged, it will be released and added to the next version Release Notes with a link for anyone to look at the PR and understand it.

Some tips and advice:

- **Connect the dots and leave a breadcrumb**: link to related Issues, Forum discussions, etc. Help others follow the trail leading up to this PR.
- **A Helpful Description**: What does the code in the PR do and what problem does it solve? How can someone use the code? Code sample, Screenshot, Quick Video… Any or all of this is so so good.
- **Draft or Work in Progress:** You don't have to *finish* the code to open a PR. Once you have a start, open it up! Most often the best way to move an Issue forward is to *see* the code in action. Also, often this helps identify ways forward before you spend a lot of time polishing.
- **Questions, Items for Discussion, Etc.**: Another reason to open a Draft PR is to ask questions and get direction via review.
- **Loop in a Maintainer for Feedback and Review**: ping someone with an `@`. And nudge again in a few days if there's no reply. We appreciate it and truly don't want the PR to get lost in the shuffle!
- **Next Steps:** Once the PR is merged, will there be a follow up step? If so, link to an Issue. How about Docs to-do or Docs to-merge?

The best thing you can do is look through existing PRs, which will give you a feel for how things work and what *you* think is helpful.

Example PR

If you're looking for an example of "what makes a good PR", look no further than this one by Kim-Adeline:
Convert component generator to TS #632

Not every PR needs this much information. But it's definitely helpful when it does!

What should I work on?

Even if you know the mechanics, it's hard to get started without a starting *place*. My best advice is this — dive into the Redwood Tutorial, read the docs, build your own experiment with Redwood. Along the way, you'll find typos, out-of-date (or missing) documentation, code that could work better, or even opportunities for improving and adding features. You'll be engaging in the Forums and Chat and developing a feel for priorities and needs. This way, you'll naturally follow your own interests and sooner than later intersect "thinks you're interested in" + "ways to help improve Redwood".

There are other more direct ways to get started as well. I've outlined a few categories that people find helpful.

Roadmap to Redwood v1 and GitHub Issues

Over the next few months, our focus is to achieve a v1.0.0 release of Redwood. You can follow the status via the Roadmap Doc, which links to GitHub Project boards with associated tasks.

Eventually, you end up on Redwood's GitHub Issues page. Here you'll find open items that need help, which are organized by labels. There are three labels for you to focus on:

1. Good First Issue: these items are more likely to be an accessible entry point to the Framework. It's less about skill level and more about focused scope.
2. Help Wanted: these items especially need contribution help from the community.
3. v1 Priority: to reach Redwood v1.0.0, we need to close *all* Issues with this label.

The sweet spot is a "v1 Priority" Issue that's either a "Good First Issue" or "Help Wanted". Yes, please!

Create an Issue

Anyone can create a new Issue. If you're not sure that your feature or idea is something to work on, start the discussion with an Issue. Describe the idea and problem + solution as clearly as possible, including examples or pseudo code if applicable. It's also very helpful to `@` mention a maintainer or Core Team member that share the area of interest.

Just know that there's a *lot* of Issues that shuffle every day. If no one replies, it's just because people are busy. Reach out in the Forums, Chat, or comment in the Issue. We intend to reply to every Issue that's opened. If yours doesn't have a reply, then give us a nudge!

Lastly, it can often be helpful to start with brief discussion in the community Chat or Forums. Sometimes that's the quickest way to get feedback and a sense of priority before opening an Issue.

Join a Contributing Project

There are several self-contained contributing projects being discussed in the Forums. These are high-priority needs that already have a defined scope and (mostly) don't require working on the Framework codebase directly. The top three right now are:

- Redesign RedwoodJS's 404 Not Found page
- RedwoodJS Splash Page
- Redesign Redwood's Error Page

If you want to join a project, join the discussion in the respective thread

Workflow: Local Development to PR

Definitions

Redwood "Project"

I refer to the codebase of a Redwood application as a **Project**. This is what you install when you run `yarn create redwood-app <path-to-directory>`. It's the *thing* you are building with Redwood.

Lastly, you'll find the template used to create a new project (when you run create redwood-app) here in GitHub: redwoodjs/redwood/packages/create-redwood-app/template/

I refer to this as the **CRWA Template or Project Template**.

Redwood "Framework"

The Framework is the codebase containing all the packages (and other code) that is published on NPMjs.com as `@redwoodjs/<package-name>`. The Framework repository on GitHub is here: https://github.com/redwoodjs/redwood

Development tools

These are the tools I use and recommend.

VS Code
Download VS Code
This has quickly become the de facto editor for JavaScript and TypeScript. Additionally, we have added recommended VS Code Extensions to use when developing both the Framework and a Project. You'll see a pop-up window asking you about installing the extensions when you open up the code.

GitHub Desktop
Download GitHub Desktop
You'll need to be comfortable using Git at the command line. But the thing I like best about GitHub Desktop is how easy it makes workflow across GitHub <> GitHub Desktop <> VS Code. You don't have to worry about syncing permissions or finding things. You can start from a repo on GitHub.com and use Desktop to do everything from "clone and open on your computer" to returning back to the site to "open a PR on GitHub".

iTerm and Oh-My-Zsh
There's nothing wrong with Terminal (on Mac) and bash. (If you're on Windows, I highly recommend using Git for Windows and Git bash.) But I enjoy using iTerm (download) and Zsh much more (use Oh My Zsh). Heads up, you can get lost in the world of theming and adding plugins. I recommend keeping it simple for awhile before taking the customization deep dive 😉

Local Development Setup

Step 1: Redwood Framework

1. **Fork the Redwood Framework** into a personal repo
2. Using GitHub Desktop, **open the Framework Codebase** in a VS Code workspace
3. Commands to "**start fresh**" when working on the Framework
   - `yarn install`
     This installs the package dependencies in /node_modules using Yarn package manager. This command is the same as just typing `yarn`. Also, if you ever switch branches and want to make sure the install dependencies are correct, you can run `yarn install --force` (shorthand `yarn -f`).
   - `git clean -fxd`
     You'll only need to do this if you've already been developing and want to "start over" and reset your codebase. This command will permanently delete *everything* that is .gitignored, e.g. /node_modules and /dist directories with package builds. When switching between branches, this command makes sure nothing is carried over that you don't want. (Warning: it will delete .env files in a Redwood Project.)
4. **Create a new branch** from the `main` branch
   First make sure you've pulled all changes from the remote origin (GitHub repo) into your local branch. (If you just cloned from your fork, you should be up to date.) Then create a new branch. The nomenclature I use is `<my_initials>-description-with-hyphens`, e.g. `dsp-add-eslint-config-redwood-toml`. I use VS Code or GitHub Desktop to manage branches. You can also do this via the CLI git checkout command.

Step 2: Test Project

There are several options for creating a local Redwood Project to use during development. Choose an option below that meets your needs based on functionality and codebase version.

1. **Clone the Redwood Tutorial App repo**
   This is the codebase to use when starting the Redwood Tutorial Part 2. It is updated to the latest version and has the Blog features. This is most often what I use for local development.
2. **Install a fresh project:** `yarn create redwood-app <path/to/project>`
   If you just need a fresh installation 1) using the latest version template codebase and 2) without any features, then just install a new Redwood project.
3. **Install a fresh project using the local Framework template code:**
   `yarn babel-node packages/create-redwood-app/src/create-redwood-app.js <path/to/project>`
   Sometimes you need to create a project that uses the Template codebase in your local branch of the Framework, e.g. your changes include modifications to the CRWA Template and need to be tested. Running the command above is exactly the same as `yarn create redwood- app …`, only it runs the command from your local Framework package using the local Template codebase.
4. 👉 **Use a build script to create a test project:** (*recommended)*
   From the Framework root directory, run `yarn build:test-project <path/to/directory>`
   This command installs a new project using the Template codebase from your current Framework branch, it then adds Tutorial features, and finally it initializes the DB (with seed data!). It should work 90% of the time and is the recommended starting place.

Note: All the options above currently set the language to JavaScript. If you would like to work with TypeScript, you can add the option `--typescript` to either of the commands that run the create-redwood-app installation.

Step 3: Link the local Framework with the local test Project

Once you work on the Framework code, you'll most often want to run the code in a Redwood app for testing. However, the Redwood Project you created for testing is currently using the latest version packages of Redwood published on NPMjs.com, e.g. @redwoodjs/core

So we'll use the Redwood-tools "Link" command to connect our local Framework and test Projects, which allows the Project to run on the code for Packages we are currently developing.

Run this command from the CLI in your test Project:
`run yarn redwood-tools link <path/to/framework-directory>`

For my example, I ran `yarn rwt link ../redwoodjs-redwood`. `rwt` is an alias for redwood-tools. And the Framework codebase was in the directory `redwoodjs-redwood`.

You'll see a lot of processes run for a minute or two as the packages are built and linked to your test Project. Once the success message appears, "Go forth and contribute", the Framework packages have been linked to your test Project and are ready to use!

You don't have to re-start the Link process — it watches for saved changes to your framework code and re-runs the build and link for the respective package. You'll see the process console update each time.

All done? You're ready to kill the link process with "ctrl + c", which will then prompt you to run: `yarn rwt unlink`

Step 4: Framework Package(s) Local Testing

Within your Framework directory, use the following tools and commands to test your code:

1. Build the packages: `yarn build`
    a. to delete all previous build directories: `yarn build:clean`
2. Syntax and Formatting: `yarn lint`
    a. to fix errors or warnings: `yarn lint:fix`
3. Run unit tests for each package: `yarn test`
4. Run through the Cypress E2E integration tests: `./tasks/run-e2e`

All of these checks are included in Redwood's GitHub PR Continuous Integration (CI) automation. However, it's good practice to understand what they do by using them locally. The E2E tests aren't something I use every time anymore (because it takes a while), but you should learn how to use it because it comes in handy when your code is failing tests on GitHub and you need to diagnose.

Step 5: Open a PR 🚀

You've made it to the fun part! It's time to use the code you're working on to create a new PR into the Redwood Framework `main` branch.

I use GitHub Desktop to walk through the process of:

- committing my changes to my development branch
- Publishing (pushing) my branch and changes to my GitHub repo fork of the Redwood Framework
- Opening a PR requesting to merge my forked-repo branch into the Redwood Framework `main` branch

Refer to the section above "What makes for a good Pull Request?" for advice on opening your PR.

Note: Make sure you check the box to “allow project maintainers to update the code” (I’m not sure about the specific description used). This helps a PR move forward more quickly as branches always need to be updated from `main` before we can merge.

When is my code “ready” to open a PR?

Most of the action, communication, and decisions happen within a PR. A common mistake new contributors make is *waiting* until their code is “perfect” before opening a PR. Assuming your PR has *some* code changes, it’s great practice to open a Draft PR (setting during the PR creation), which you can use to start discussion and ask questions. PRs are closed all the time without being merged, often because they are replaced by another PR resulting from decisions and discussion. It’s part of the process. More importantly, it means collaboration is happening!

What isn’t a fun experience is spending a whole bunch of time on code that ends up not being the correct direction or is unnecessary/redundant to something that already exists. This is a part of the learning process. But it’s another reason to open a draft PR sooner than later to get confirmation and questions out of the way before investing time into refining and details.

When in doubt, just try first and ask for help and direction!

Original Collaborative Notes Created During Workshop

Resources

Docs, Where to find Things + Help, Tools, Etc.

What makes for a good Pull Request?

How to get started?

Join the Redwood Community (forums and chat)

- https://community.redwoodjs.com

- https://discord.gg/jjSYEQd

Development to PR Workflow

What should I work on?

David To Do:
- Project listed on the Forum
- Upcoming Competitions

Local Dev Set Up

Commands and Tips

Git clean -fxd ?

Testing & QA: Locally and CI Automation

Pull Request

Questions

Feedback