Svelte for new developers

Rich Harris

This short guide is designed to help you — someone who has looked at the tutorial and wants to start creating Svelte apps, but doesn't have a ton of experience using JavaScript build tooling — get up and running.

If there are things that don't make sense, or that we're glossing over, feel free to raise an issue or suggest edits to this page that will help us help more people.

If you get stuck at any point following this guide, the best place to ask for help is in the chatroom.

First things first

You'll be using the *command line*, also known as the terminal. On Windows, you can access it by running **Command Prompt** from the Start menu; on a Mac, hit `Cmd` and `Space` together to bring up **Spotlight**, then start typing `Terminal.app`. On most Linux systems, `Ctrl-Alt-T` brings up the command line.

The command line is a way to interact with your computer (or another computer! but that's a topic for another time) with more power and control than the GUI (graphical user interface) that most people use day-to-day.

Once on the command line, you can navigate the filesystem using `ls` (`dir` on Windows) to list the contents of your current directory, and `cd` to change the current directory. For example, if you had a `Development` directory of your projects inside your home directory, you would type

```
cd Development
```

to go to it. From there, you could create a new project directory with the `mkdir` command:

```
mkdir svelte-projects
cd svelte-projects
```

A full introduction to the command line is out of the scope of this guide, but here are a few more useful commands:

- `cd ..` — navigates to the parent of the current directory

- `cat my-file.txt` — on Mac/Linux (`type my-file.txt` on Windows), lists the contents of `my-file.txt`
- `open .` (or `start .` on Windows) — opens the current directory in Finder or File Explorer

Installing Node.js

Node is a way to run JavaScript on the command line. It's used by many tools, including Svelte. If you don't yet have it installed, the easiest way is to download the latest version straight from the website.

Once installed, you'll have access to three new commands:

- `node my-file.js` — runs the JavaScript in `my-file.js`
- `npm [subcommand]` — npm is a way to install 'packages' that your application depends on, such as the svelte package
- `npx [subcommand]` — a convenient way to run programs available on npm without permanently installing them

Installing a text editor

To write code, you need a good editor. The most popular choice is Visual Studio Code or VSCode, and justifiably so — it's well-designed and fully-featured, and has a wealth of extensions (including one for Svelte, which provides syntax highlighting and diagnostic messages when you're writing components).

Creating a project

We're going to follow the instructions in part two of The easiest way to get started with Svelte.

First, we'll use npx to run degit, a program for cloning project templates from GitHub and other code storage websites. You don't have to use a project template, but it means you have to do a lot less setup work. You will need to have Git installed in order to use degit. (Eventually you'll probably have to learn Git itself, which most programmers use to manage their projects.)

On the command line, navigate to where you want to create a new project, then type the following lines (you can paste the whole lot, but you'll develop better muscle memory if you get into the habit of writing each line out one at a time then running it):

```
npx degit sveltejs/template my-svelte-project
cd my-svelte-project
npm install
```

This creates a new directory, `my-svelte-project`, adds files from the sveltejs/template code repository, and installs a number of packages from npm.

Open the directory in your text editor and take a look around. The app's 'source code' lives in the `src` directory, while the files your app can load are in `public`.

In the `package.json` file, there is a section called `"scripts"`. These scripts define shortcuts for working with your application — `dev`, `build` and `start`. To launch your app in development mode, type the following:

```
npm run dev
```

Running the `dev` script starts a program called Rollup. Rollup's job is to take your application's source files (so far, just `src/main.js` and `src/App.svelte`), pass them to other programs (including Svelte, in our case) and convert them into the code that will actually run when you open the application in a browser.

Speaking of which, open a browser and navigate to http://localhost:8080. This is your application running on a local *web server* (hence 'localhost') on port 8080.

Try changing `src/App.svelte` and saving it. The application will reload with your changes.

Building your app

In the last step, we were running the app in 'development mode'. In dev mode, Svelte adds extra code that helps with debugging, and Rollup skips the final step where your app's JavaScript is compressed using Terser.

When you share your app with the world, you want to build it in 'production mode', so that it's as small and efficient as possible for end users. To do that, use the `build` command:

```
npm run build
```

Your `public` directory now contains a compressed `bundle.js` file containing your app's JavaScript. You can run it like so:

```
npm run start
```

This will run the app on http://localhost:8080.

Next steps

To share your app with the world you'll need to *deploy* it. There are many ways to do so — some are listed in the `README.md` file inside your project.