# Contributing

Pico welcomes contributions from everyone, whether it's bug fixes, new

features or typo fixes.

If you're unsure about anything feel free to

[email me](mailto:maciej@usegripeless.com) with any questions, I'll be happy

to assist you.

# Testing

There are two ways to test Pico's output, one being the provided visual

testing suite and two a web extension that you can test Pico on any real

world website (as long as it doesn't block `eval` (which is most of them so

it's not much of an issue)).

## Development server

To run any of the tests you need to use the development server:

```bash

$ npm run dev

```

Main bundle will be served at `http://localhost:23231/index.js` and

automatically updated whether you change any of the source code.

## Web extension

> Run Pico on any website of your choice.

1. Ensure that you have the development server running

2. Open the web extension (located in `web-ext/` in the root of this

repository) in the browser of your choice - note that it has only been

tested with Firefox, so YMMV. If you have Firefox Nightly installed, for

convenience run `npm run dev:ext` to open the web extension in that

browser - you will have the development browser separate and clean from

your day to day one.

3. With the opened browser, browse onto a website you want to test.

You should see a browser action in your address bar:

![Browser action in address bar](https://github.com/gripeless/pico/blob/master/media/browser-action-address-bar.png?raw=true)

Click on the green puzzle icon to take a screenshot, a new tab should

open with the output contents in SVG form.

## Testing REPL

> Separate cases to test out individual parts of the renderer.

The test suite spawns a real browser (via

[Playwright](https://github.com/microsoft/playwright)) and lets you compare

the output screenshot from Pico to the "real" viewport.

It is designed with really quick iteration in mind and to allow you to

scaffold out potential problems with the library quickly, whether it's

rendering related, CORS related, or anything browser specific related.

The suite is currently only provided in an interactive form, which means

that you have to use your eyes to judge whether the result is suiting or

not.

### Running the REPL

1. Ensure that you have the development server running

2. Add the following to your hosts file:

```

127.0.0.1 test.localhost

127.0.0.1 test2.localhost

127.0.0.1 test3.localhost

127.0.0.1 test4.localhost

127.0.0.1 test5.localhost

```

This lets you test cross-origin related issues without having to do any

complicated local DNS setup.

3. Start the interactive test REPL:

```bash

$ npm run test:interactive

```

An empty browser window should open (in the background if you're on macOS),

this will show the output of any test you run through the console.

### REPL Commands

#### `open`

Opens the browser output again, useful if you happened to close the original output.

#### `run (case name)`

Runs a test with the specified name.

#### `watch (case name)`

Same as `run`, but will update if either the `source/` directory updates or the case.

Press <kbd>q</kbd> to exit the watcher.

#### `all`

Runs all available tests.

### Adding a new test case

The easiest way to add a new case is to start with an existing one and

modify it to suit your needs. For most cases you won't need any special

browser flags or extra servers - I usually just copy `test/cases/hello` into

a new directory in `test/cases`.

1. Create the test's directory in `test/cases`.

For consistency, the name of the test case should be kebab-case.

2. Add your test to `test/cases/index.ts`

After that's done restart your test REPL and you'll be able to run your test

case.