



Replay

Replay is a new debugger for recording and replaying software. Debugging with Replay should be as simple as viewing print statements and more powerful than pausing with breakpoints. Of course, debugging should be collaborative as well!

Issues

Feel free to file any issues you see while recording or replaying.

Getting started

Replay's DevTools is a React app built on top of the Replay protocol. Make sure to install nvm.

Note: We use Yarn 3 for package management - make sure you have Yarn installed globally first:

```
npm i -g yarn
```

Getting started is as simple as:

```
1 # Setup environment variables
2 cp .env.sample .env
3
4 # Install dependencies
5 nvm use
6 yarn install
7
8 # Run dev server (see below)
```

Developing against production

To run DevTools locally, paired with the production Dashboard project:

```
1 yarn dev:prod
```

At this point DevTools will be accessible at localhost:8080 and will proxy any library or authentication requests to the production deployment of the Dashboard project.

Local development

To run both the DevTools and Dashboard projects locally:

```
1 # DevTools (this project)
2 yarn dev:local
3
4 # Dashboard
5 pnpm dev:local
```

At this point the Dashboard will be accessible at localhost:8080 but it will not load recordings. To be able to test the end-to-end interaction of both apps, use localhost:8081. It will serve both Dashboard and DevTools routes.

Next steps

Next, download and install the Replay browser. The browser will allow you to start recording your own replays.

Community

Everybody's welcome to join us on Discord, but please read through our Community Etiquette guidelines first.

Contributing to the project

Anyone is welcome to contribute to the project! If you're just getting started we recommend you start by reading the contributing guide and then check out the "good first issues". If you have questions you can ask in the in the #development channel. (Please do not "@" or direct message people with questions though!)

Running tests:

To run the end-to-end tests make sure that devtools is running locally on port 8080 and run:

```
1 cd packages/e2e-tests
2
3 yarn test
```

Running tests against local builds of the browser If you want to run the tests against a local build of the browser, you'll need to invoke the tests like so:

```
1 RECORD_REPLAY_PATH=~/.devel/gecko-dev/rr-opt/dist/Replay.app
  RECORD_REPLAY_BUILD_PATH=~/.devel/gecko-dev node test/run.js
```

Replace the paths with the appropriate paths to and within `gecko-dev` as appropriate in your environment.

Running tests against local builds of the backend If you want to run the tests against a local build of the backend, you'll need to invoke the tests like so:

```
1 RECORD_REPLAY_SERVER=ws://localhost:8000 RECORD_REPLAY_DRIVER=~/.devel/
  backend/out/macOS-recordreplay.so node test/run.js
```

Replace the paths with the appropriate paths within the `backend` repo as appropriate in your environment.

Installing the trunk launcher This project uses trunk to lint and format its code. Trunk is installed as a dev dependency so you can invoke it as `npm run trunk`. To learn more about `trunk`, check out the documentation.

Linting your changes In most cases you can simply run `trunk check` which will autodetect the changes you have made and lint them. If you would like to only autoformat your changes, you can run `trunk fmt`. More information on using trunk can be found [here](#).

Generating GraphQL types for Typescript See `/src/graphql/README.md` for details.

Deploying

Every commit to the main branch is automatically deployed to production.