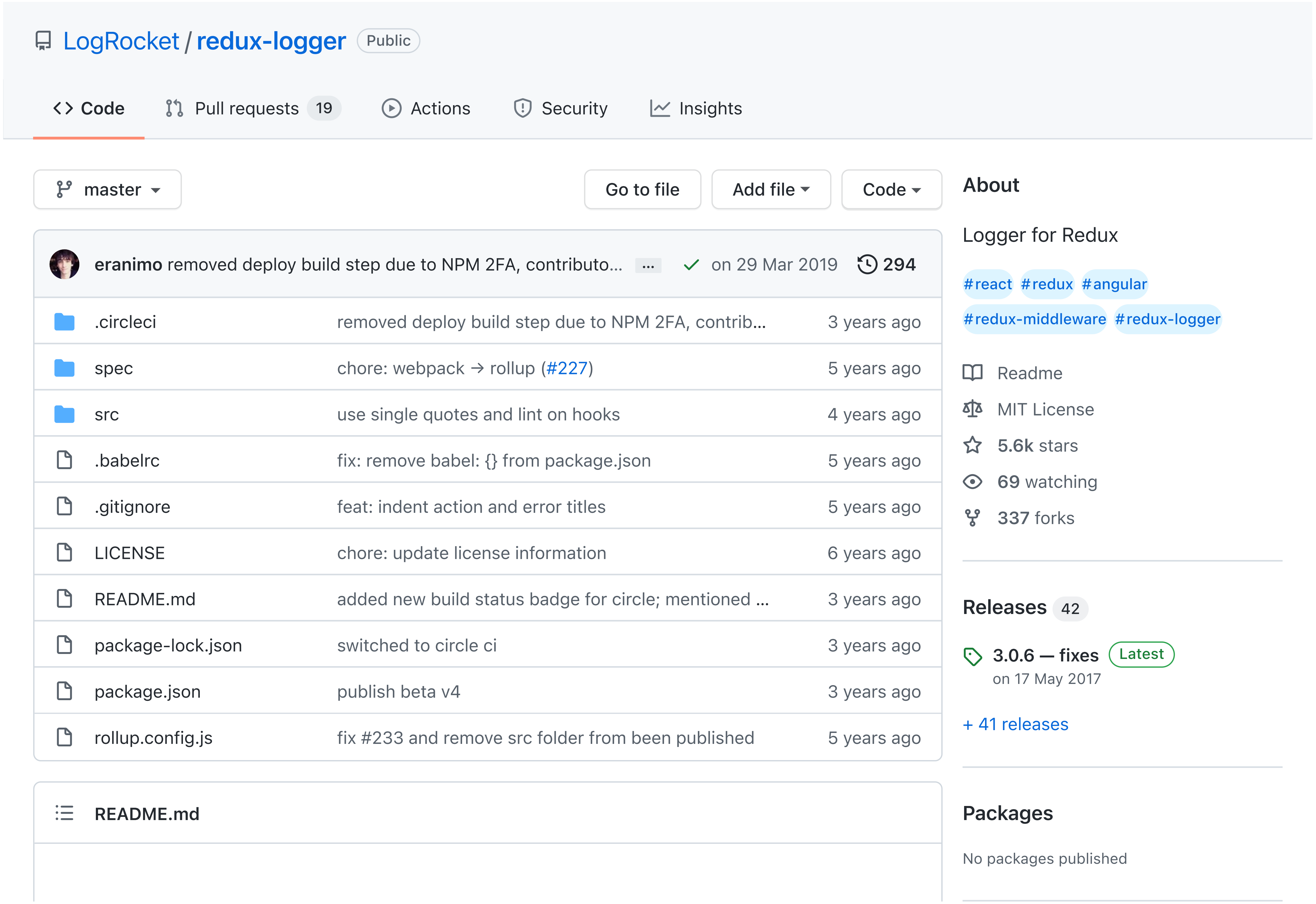

LogRocket / redux-logger Public

eranimo removed deploy build step due to NPM 2FA, contributo... … ✓ on 29 Mar 2019 294

.circleci	removed deploy build step due to NPM 2FA, contrib...	3 years ago
spec	chore: webpack → rollup (#227)	5 years ago
src	use single quotes and lint on hooks	4 years ago
.babelrc	fix: remove babel: {} from package.json	5 years ago
.gitignore	feat: indent action and error titles	5 years ago
LICENSE	chore: update license information	6 years ago
README.md	added new build status badge for circle; mentioned ...	3 years ago
package-lock.json	switched to circle ci	3 years ago
package.json	publish beta v4	3 years ago
rollup.config.js	fix #233 and remove src folder from been published	5 years ago

README.md

About

Logger for Redux

#react #redux #angular #redux-middleware #redux-logger

Readme
MIT License
5.6k stars
69 watching
337 forks

Releases 42

3.0.6 — fixes Latest
on 17 May 2017

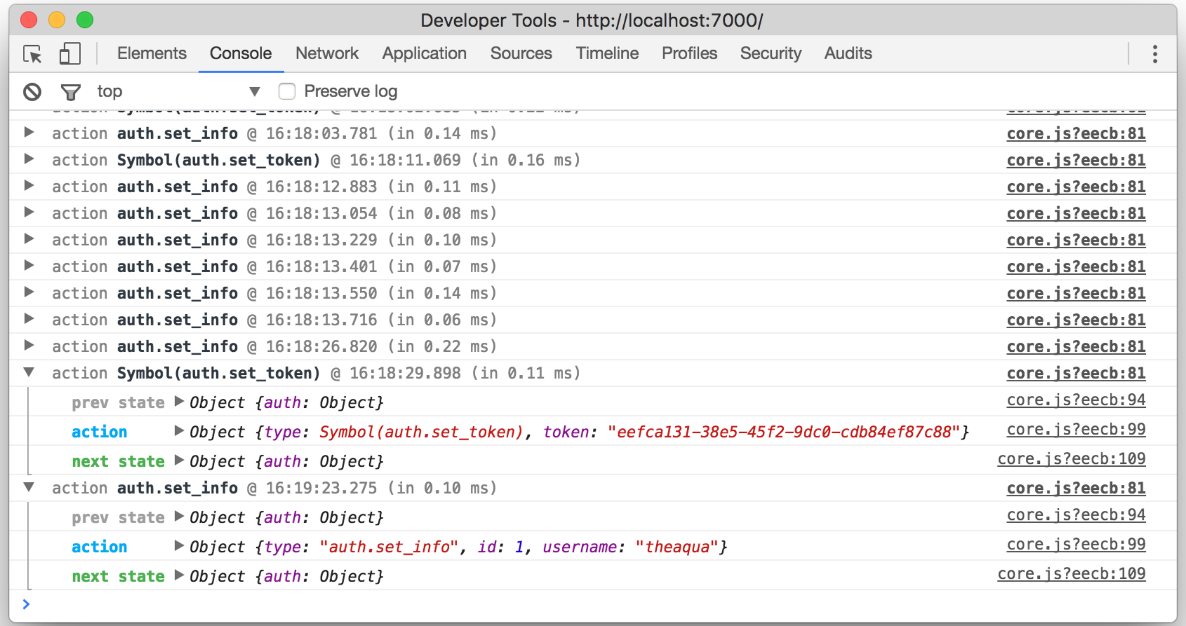
+ 41 releases

Packages

No packages published

Logger for Redux

npm v3.0.6 downloads 3.6M/month PASSED



```
▶ action auth.set_info @ 16:18:03.781 (in 0.14 ms)                                    core.js?eecb:81
▶ action Symbol(auth.set_token) @ 16:18:11.069 (in 0.16 ms)                           core.js?eecb:81
▶ action auth.set_info @ 16:18:12.883 (in 0.11 ms)                                    core.js?eecb:81
▶ action auth.set_info @ 16:18:13.054 (in 0.08 ms)                                    core.js?eecb:81
▶ action auth.set_info @ 16:18:13.229 (in 0.10 ms)                                    core.js?eecb:81
▶ action auth.set_info @ 16:18:13.401 (in 0.07 ms)                                    core.js?eecb:81
▶ action auth.set_info @ 16:18:13.550 (in 0.14 ms)                                    core.js?eecb:81
▶ action auth.set_info @ 16:18:13.716 (in 0.06 ms)                                    core.js?eecb:81
▶ action auth.set_info @ 16:18:26.820 (in 0.22 ms)                                    core.js?eecb:81
▼ action Symbol(auth.set_token) @ 16:18:29.898 (in 0.11 ms)                           core.js?eecb:81
    prev state ▶Object {auth: Object}                                                 core.js?eecb:94
    action     ▶Object {type: Symbol(auth.set_token), token: "eefca131-38e5-45f2-9dc0-cdb84ef87c88"}   core.js?eecb:99
    next state ▶Object {auth: Object}                                                 core.js?eecb:109
▼ action auth.set_info @ 16:19:23.275 (in 0.10 ms)                                    core.js?eecb:81
    prev state ▶Object {auth: Object}                                                 core.js?eecb:94
    action     ▶Object {type: "auth.set_info", id: 1, username: "theaqua"}            core.js?eecb:99
    next state ▶Object {auth: Object}                                                 core.js?eecb:109
>
```

Now maintained by LogRocket!

LogRocket

Used by 327k

+ 327,098

Contributors 49

+ 38 contributors

Environments 1

github-pages Active

Languages

● JavaScript 100.0%

> LogRocket is a production Redux logging tool that lets you replay problems as if they happened in your own browser. Instead of guessing why errors happen, or asking users for screenshots and log dumps, LogRocket lets you replay Redux actions + state, network requests, console logs, and see a video of what the user saw.

For more informatiom about the future of redux-logger, check out the discussion here.

Table of contents

- Install
- Usage
- Options
- Recipes
  - Log only in development
  - Log everything except actions with certain type
  - Collapse actions with certain type
  - Transform Immutable (without `combineReducers`)
  - Transform Immutable (with `combineReducers`)
  - Log batched actions
- To Do
- Known issues (with `react-native` only at this moment)
- License

Install

`npm i --save redux-logger`

Typescript types are also available, via DefinitelyTyped:

`npm i @types/redux-logger`

Usage

```
import { applyMiddleware, createStore } from 'redux';

// Logger with default options
import logger from 'redux-logger'
const store = createStore(
  reducer,
  applyMiddleware(logger)
)

// Note passing middleware as the third argument requires redux@>=3.1.0
```

Or you can create your own logger with custom options:

```
import { applyMiddleware, createStore } from 'redux';
import { createLogger } from 'redux-logger'

const logger = createLogger({
  // ...options
});

const store = createStore(
  reducer,
  applyMiddleware(logger)
);
```

Note: logger **must be** the last middleware in chain, otherwise it will log thunk and promise, not actual actions (#20).

Options

```
{
  predicate, // if specified this function will be called before each action is
  collapsed, // takes a Boolean or optionally a Function that receives `getStat
  duration = false: Boolean, // print the duration of each action?
  timestamp = true: Boolean, // print the timestamp with each action?

  level = 'log': 'log' | 'console' | 'warn' | 'error' | 'info', // console's le
  colors: ColorsObject, // colors for title, prev state, action and next state:
  titleFormatter, // Format the title used when logging actions.

  stateTransformer, // Transform state before print. Eg. convert Immutable obje
  actionTransformer, // Transform action before print. Eg. convert Immutable ob
  errorTransformer, // Transform error before print. Eg. convert Immutable obje

  logger = console: LoggerObject, // implementation of the `console` API.
  logErrors = true: Boolean, // should the logger catch, log, and re-throw erro

  diff = false: Boolean, // (alpha) show diff between states?
  diffPredicate // (alpha) filter function for showing states diff, similar to
}
```

Options description

level (String | Function | Object)

Level of `console`. `warn`, `error`, `info` or else.

It can be a function `(action: Object) => level: String`.

It can be an object with level string for: `prevState`, `action`, `nextState`, `error`

It can be an object with getter functions: `prevState`, `action`, `nextState`, `error`. Useful if you want to print message based on specific state or action. Set any of them to `false` if you want to hide it.

- `prevState(prevState: Object) => level: String`
- `action(action: Object) => level: String`
- `nextState(nextState: Object) => level: String`
- `error(error: Any, prevState: Object) => level: String`

Default: `log`

duration (Boolean)

Print duration of each action?

Default: `false`

timestamp (Boolean)

Print timestamp with each action?

Default: `true`

colors (Object)

Object with color getter functions: `title`, `prevState`, `action`, `nextState`, `error`. Useful if you want to paint message based on specific state or action. Set any of them to `false` if you want to show plain message without colors.

- `title(action: Object) => color: String`

- `prevState(prevState: Object) => color: String`
- `action(action: Object) => color: String`
- `nextState(nextState: Object) => color: String`
- `error(error: Any, prevState: Object) => color: String`

logger (Object)

Implementation of the `console` API. Useful if you are using a custom, wrapped version of `console`.

Default: `console`

logErrors (Boolean)

Should the logger catch, log, and re-throw errors? This makes it clear which action triggered the error but makes "break on error" in dev tools harder to use, as it breaks on re-throw rather than the original throw location.

Default: `true`

collapsed = (getState: Function, action: Object, logEntry: Object) => Boolean

Takes a boolean or optionally a function that receives `getState` function for accessing current store state and `action` object as parameters. Returns `true` if the log group should be collapsed, `false` otherwise.

Default: `false`

predicate = (getState: Function, action: Object) => Boolean

If specified this function will be called before each action is processed with this middleware. Receives `getState` function for accessing current store state and `action` object as parameters. Returns `true` if action should be logged, `false` otherwise.

Default: `null` (always log)

stateTransformer = (state: Object) => state

Transform state before print. Eg. convert Immutable object to plain JSON.

Default: identity function

actionTransformer = (action: Object) => action

Transform action before print. Eg. convert Immutable object to plain JSON.

Default: identity function

errorTransformer = (error: Any) => error

Transform error before print.

Default: identity function

titleFormatter = (action: Object, time: String?, took: Number?) => title

Format the title used for each action.

Default: prints something like `action @ ${time} ${action.type} (in ${took.toFixed(2)} ms)`

diff (Boolean)

Show states diff.

Default: `false`

diffPredicate = (getState: Function, action: Object) => Boolean

Filter states diff for certain cases.

Default: `undefined`

Recipes

Log only in development

```javascript
const middlewares = [];

if (process.env.NODE_ENV === `development`) {
  const { logger } = require(`redux-logger`);

  middlewares.push(logger);
}

const store = compose(applyMiddleware(...middlewares))(createStore)(reducer);
```

Log everything except actions with certain type

```javascript
createLogger({
  predicate: (getState, action) => action.type !== AUTH_REMOVE_TOKEN
});
```

Collapse actions with certain type

```
createLogger({
  collapsed: (getState, action) => action.type === FORM_CHANGE
});
```

Collapse actions that don't have errors

```
createLogger({
  collapsed: (getState, action, logEntry) => !logEntry.error
});
```

Transform Immutable (without `combineReducers`)

```
import { Iterable } from 'immutable';

const stateTransformer = (state) => {
  if (Iterable.isIterable(state)) return state.toJS();
  else return state;
};

const logger = createLogger({
  stateTransformer,
});
```

Transform Immutable (with `combineReducers`)

```
const logger = createLogger({
  stateTransformer: (state) => {
    let newState = {};
    for (var i of Object.keys(state)) {
      if (Immutable.Iterable.isIterable(state[i])) {
        newState[i] = state[i].toJS();
      } else {
        newState[i] = state[i];
      }
    };

    return newState;
  }
});
```

Log batched actions

Thanks to @smashercosmo

```
import { createLogger } from 'redux-logger';

const actionTransformer = action => {
  if (action.type === 'BATCHING_REDUCER.BATCH') {
    action.payload.type = action.payload.map(next => next.type).join(' => ');
    return action.payload;
  }

  return action;
};

const level = 'info';

const logger = {};

for (const method in console) {
  if (typeof console[method] === 'function') {
    logger[method] = console[method].bind(console);
  }
}

logger[level] = function levelFn(...args) {
  const lastArg = args.pop();

  if (Array.isArray(lastArg)) {
    return lastArg.forEach(item => {
      console[level].apply(console, [...args, item]);
    });
  }

  console[level].apply(console, arguments);
};

export default createLogger({
  level,
  actionTransformer,
  logger
});
```

To Do

- [x] Update eslint config to airbnb's
- [ ] Clean up code, because it's very messy, to be honest
- [ ] Write tests
- [ ] Node.js support
- [ ] React-native support

Feel free to create PR for any of those tasks!

Known issues

- Performance issues in react-native (#32)

License

MIT