Redux без копипаста

#митап_за_чашкой_чая

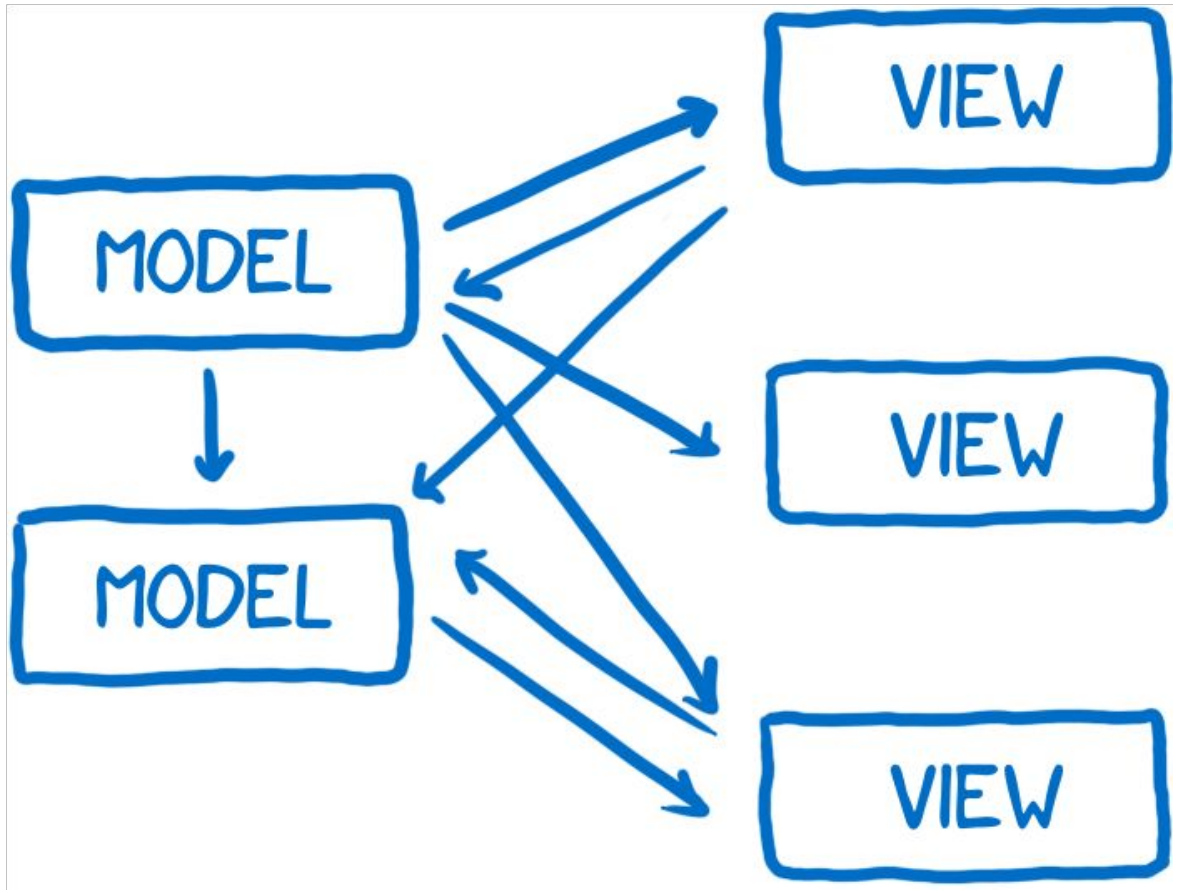
Pavel Uvarov,
Software Engineer at Epam Ryazan

pavel_uvarov@epam.com

spiderpoul

poul_uvarov

Redux is a predictable state container

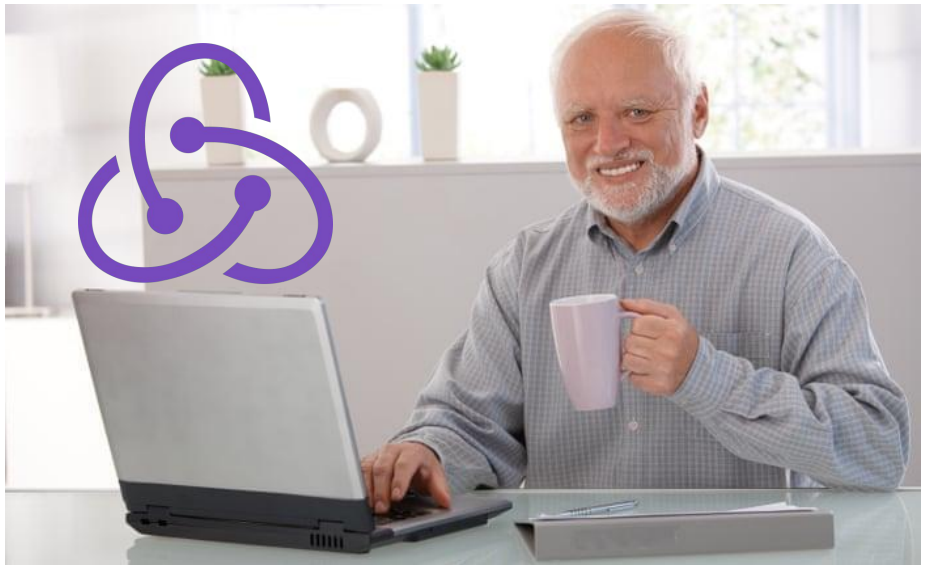

VIEW
MODEL
VIEW
MODEL
VIEW

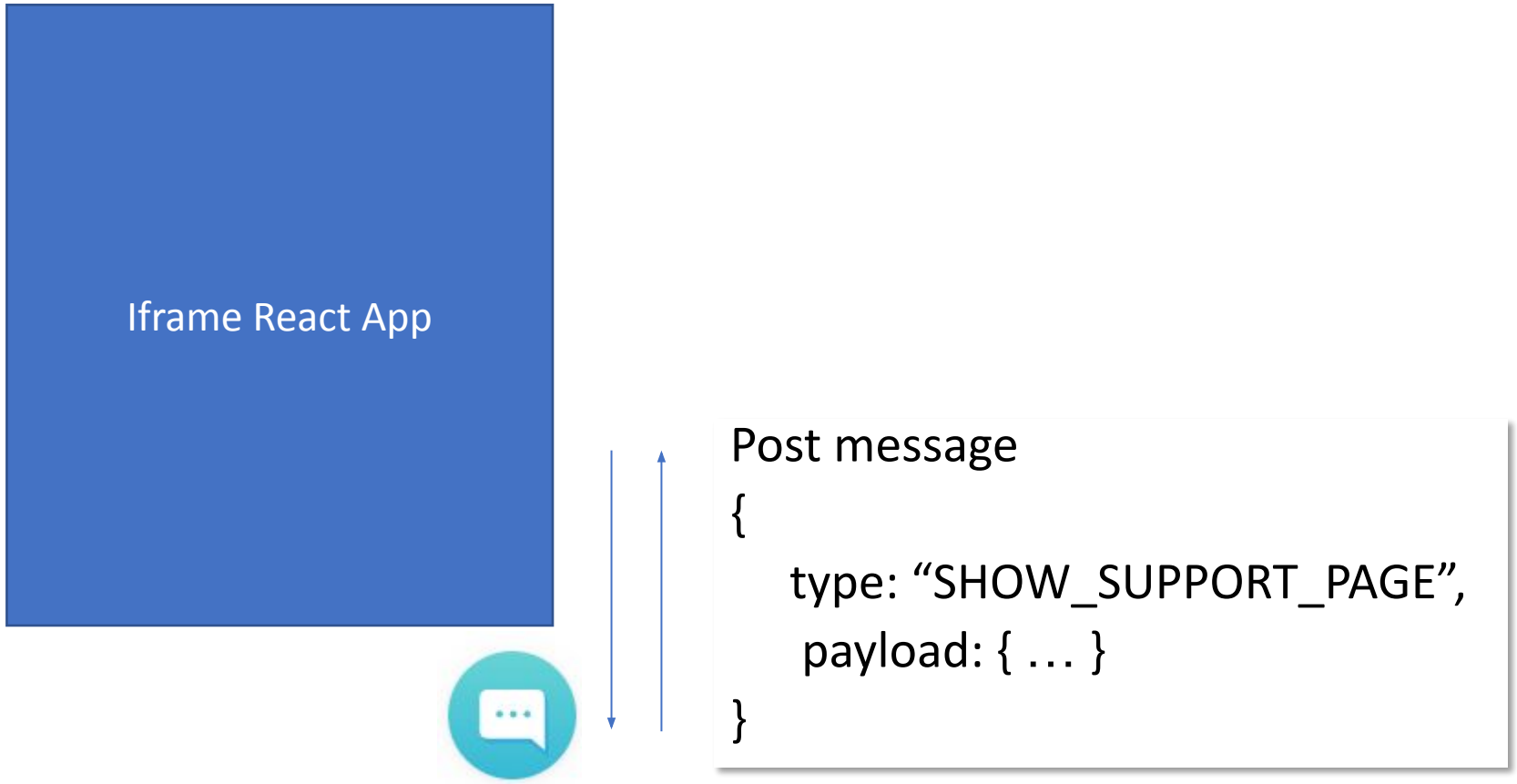
What I like about Redux?

- Predictable state management
- Global state
- Immutability
- A lot of middleware
- Easy to use
- Easy to test
- Time traveling
- Dev tools
- Big community

What we don’t like about Redux

- "Configuring a Redux store is too complicated"
- "I have to add a lot of packages to get Redux to do anything useful"
- "Redux requires too much boilerplate code"



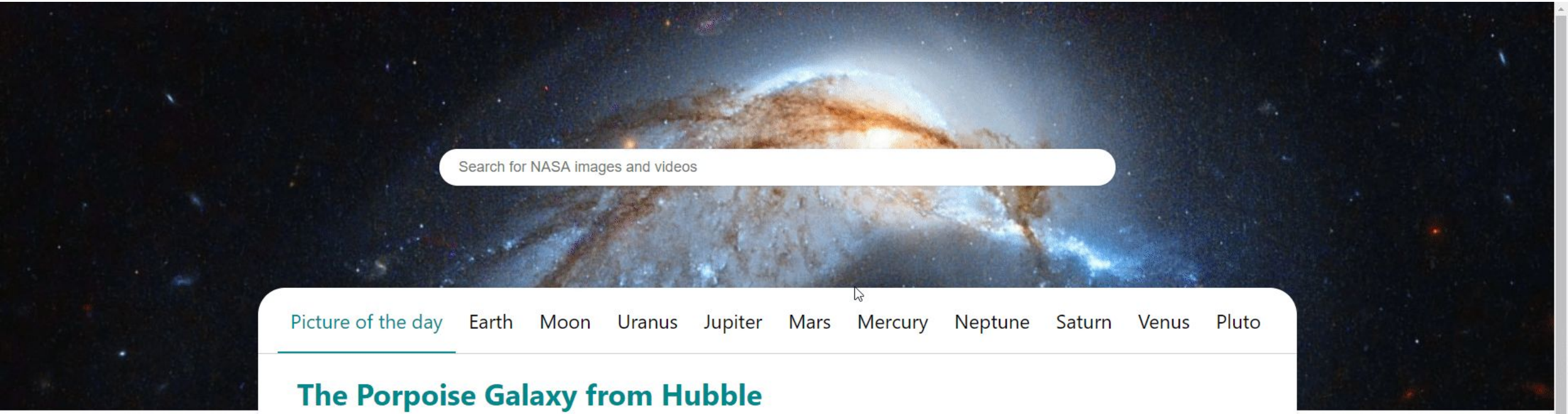

Iframe React App
Post message
{
type: “SHOW_SUPPORT_PAGE”,
payload: { … }
}

This tool will change the way you work with Redux

- 2-3x less code
- Strictly typed out of the box
- Reducer and action creation all in one
- You won’t hate Redux so much after

Special thanks to Yakov Zhmurov

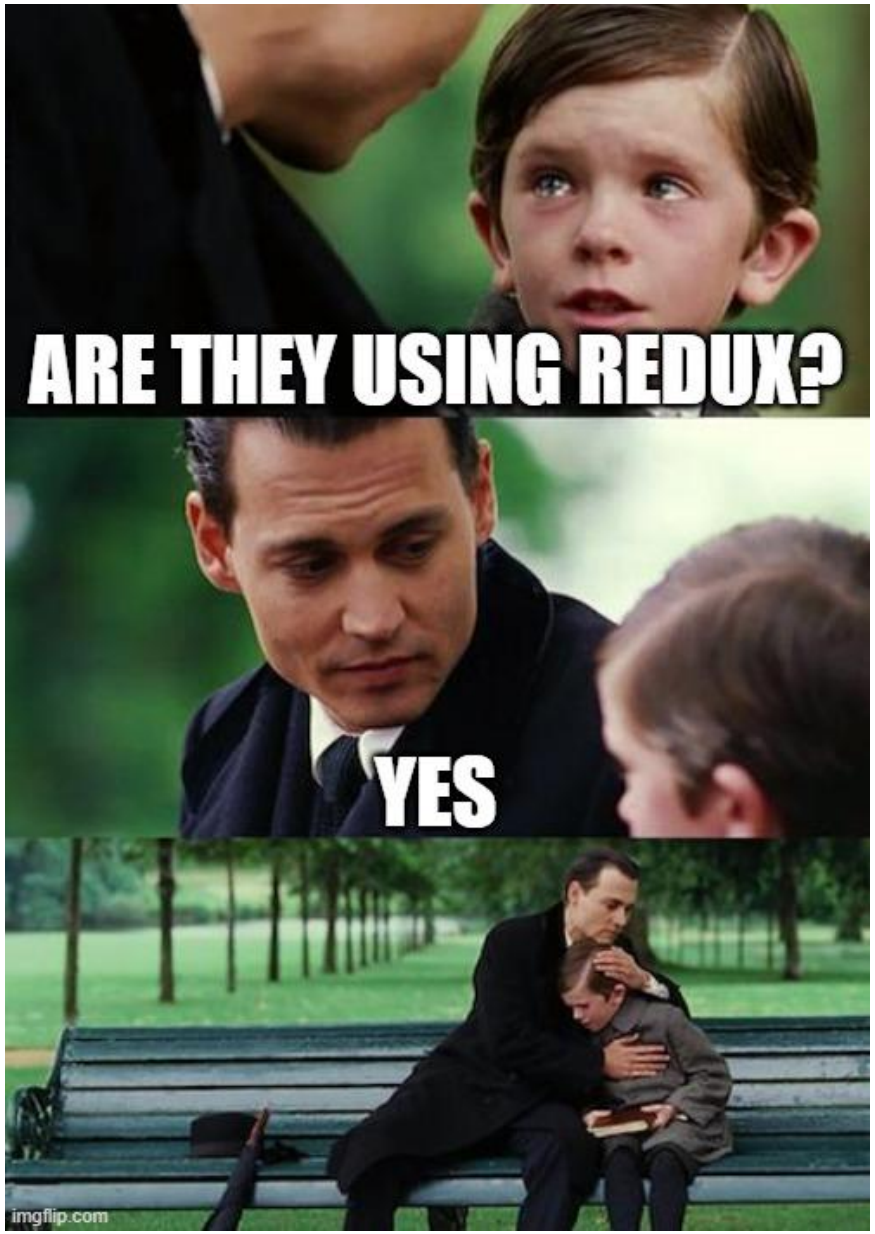
React Redux Typescript + NASA API (nasa-app-react.netlify.app)



Picture of the day | Earth | Moon | Uranus | Jupiter | Mars | Mercury | Neptune | Saturn | Venus | Pluto

The Porpoise Galaxy from Hubble

What's happening to this spiral galaxy? Just a few hundred million years ago, NGC 2936, the upper of the two large galaxies shown, was likely a normal spiral galaxy -- spinning, creating stars -- and minding its own business. But then it got too close to the massive elliptical galaxy NGC 2937 below and took a dive. Dubbed the Porpoise Galaxy for its iconic shape, NGC 2936 is not only being deflected but also being distorted by the close gravitational interaction. A burst of young blue stars forms the nose of the porpoise toward the right of the upper galaxy, while the center of the spiral appears as an eye. Alternatively, the galaxy pair, together known as Arp 142, look to some like a penguin protecting an egg. Either way, intricate dark dust lanes and bright blue star streams trail the troubled galaxy to the lower right. The featured re-processed image showing Arp 142 in unprecedented detail was taken by the Hubble Space Telescope last year. Arp 142 lies about 300 million light years away toward the constellation, coincidently, of the Water Snake (Hydra). In a billion years or so the two galaxies will likely merge into one larger galaxy.


ARE THEY USING REDUX?
YES
imgflip.com

Too much boilerplate…

```
export const SEARCH_PAGE_NASA_REQUEST = 'SEARCH_PAGE_NASA_REQUEST';
export const SEARCH_PAGE_NASA_SUCCESS = 'SEARCH_PAGE_NASA_SUCCESS';
export const SEARCH_PAGE_NASA_ERROR = 'SEARCH_PAGE_NASA_ERROR';

export const SEARCH_PAGE_SET_SEARCH = 'SEARCH_PAGE_SET_SEARCH';
```

```
export const searchPageRequest = (): PageRequest => ({
    type: SEARCH_PAGE_NASA_REQUEST,
});

export const searchPageSuccess = ({ data }): PageSuccess => ({
    type: SEARCH_PAGE_NASA_SUCCESS,
    payload: {
        data,
    },
});

export const setSearch = ({ search }: { search: string }): SetSearch => ({
    type: SEARCH_PAGE_SET_SEARCH,
    payload: {
        search,
    },
});

export const searchPageError = ({ error }): PageError => ({
    type: SEARCH_PAGE_NASA_ERROR,
    payload: {
        error,
    },
});
```

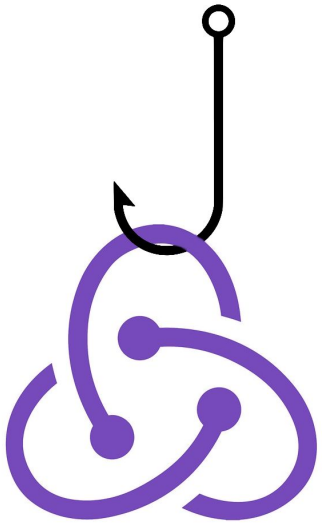
```
const getState = (state: AppState) => state.searchPage;

export const selectData = createSelector(getState, (state) => state.data);

export const selectSearch = createSelector(getState, (state) => state.search);

export const selectIsLoading = createSelector(
    getState,
    (state) => state.isLoading
);
```

```
export const fetchSearchData = ({ search }: { search: string }) => async (
    dispatch
) => {
    dispatch(searchPageRequest());
    try {
        const res = await axios.get(`${API_URL}?q=${search}`);
        dispatch(searchPageSuccess({ data: res.data?.collection }));
    } catch (error) {
        searchPageError({ error });
    }
};
```

```
const SearchReducer = (state = initState, action: ActionsTypes) ⇒ {
    switch (action.type) {
        case SEARCH_PAGE_NASA_REQUEST: {
            return {
                ...state,
                isLoading: true,
            };
        }
        case SEARCH_PAGE_NASA_SUCCESS: {
            return {
                ...state,
                isLoading: false,
                data: action.payload.data,
            };
        }
        case SEARCH_PAGE_SET_SEARCH: {
            return {
                ...state,
                isLoading: false,
                search: action.payload.search,
            };
        }
        case SEARCH_PAGE_NASA_ERROR: {
            return {
                ...state,
                isLoading: false,
                error: action.payload.error,
            };
        }
        default:
            return state;
    }
};
```

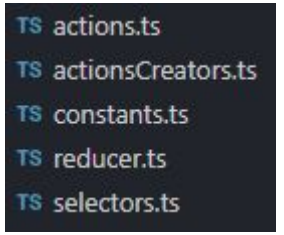
Less pain with React Redux Hooks

```
function SearchPage() {
    const dispatch = useDispatch();
    const data = useSelector(selectData);
    const search = useSelector(selectSearch);
    const searchDebounce = useDebounce(search);
    const isLoading = useSelector(selectIsLoading);
    const querySearch = useQuerySearch();

    useEffect(() => {
        dispatch(fetchSearchData({ search: searchDebounce }));
    }, [dispatch, searchDebounce]);

    useEffect(() => {
        if (!search && querySearch) {
            dispatch(setSearch({ search: querySearch }));
        }
    }, []);

    return data?.items?.length || isLoading ? (
        <CardGrid items={data?.items} isLoading={isLoading} />
    ) : (
        <Placeholder>No results found.</Placeholder>
    );
}

export default SearchPage;
```



Even more with Typescript...

```
You, 18 days ago | 1 author (You)
interface PageRequest {
    type: typeof SEARCH_PAGE_NASA_REQUEST;
}

You, 18 days ago | 1 author (You)
interface PageSuccess {
    type: typeof SEARCH_PAGE_NASA_SUCCESS;
    payload: {
        data: LibraryResponse;
    };
}

You, 18 days ago | 1 author (You)
interface SetSearch {
    type: typeof SEARCH_PAGE_SET_SEARCH;
    payload: {
        search: string;
    };
}

You, 18 days ago | 1 author (You)
interface PageError {
    type: typeof SEARCH_PAGE_NASA_ERROR;
    payload: {
        error: any;
    };
}
```




LET'S CREATE A REDUX STORE
NO PROBLEMS
imgflip.com

Reducing Boilerplate

Redux is in part inspired by Flux, and the most common complaint about Flux is how it makes you write a lot of boilerplate. In this recipe, we will consider how Redux lets us choose how verbose we'd like our code to be, depending on personal style, team preferences, longer term maintainability, and so on.

#Actions

Actions are plain objects describing what happened in the app, and serve as the sole way to describe an intention to mutate t
dispatch is not boilerplate, but

There are frameworks claiming t
terms of being predictable, this i
plain object actions, it is impossil
reloading with time travel. If you

```
const ADD_TODO = 'ADD_TODO'
const EDIT_TODO = 'EDIT_TODO'
const REMOVE_TODO = 'REMOVE_TODO'

export const addTodo = makeActionCreator(ADD_TODO, 'text')
export const editTodo = makeActionCreator(EDIT_TODO, 'id', 'text')
export const removeTodo = makeActionCreator(REMOVE_TODO, 'id')
```

There are also utility libraries to aid in generating action creators, such as redux-act and redux-actions. These can help reduce boilerplate code and enforce adherence to standards such as Flux Standard Action (FSA).

redux-act / redux-actions

```
You, 6 minutes ago | 1 author (You)
export interface SearchPageSuccessPayload {
    data: LibraryResponse;
}

You, 6 minutes ago | 1 author (You)
export interface SetSearchPayload {
    search: string;
}

You, 6 minutes ago | 1 author (You)
export interface SearchPageErrorPayload {
    error: string;
}

export const searchPageError = createAction<SearchPageErrorPayload>(
    'SEARCH_PAGE_NASA_SUCCESS'
);      You, 6 minutes ago • Uncommitted changes
export const searchPageRequest = createAction('SEARCH_PAGE_NASA_REQUEST');
export const searchPageSuccess = createAction<SearchPageSuccessPayload>(
    'SEARCH_PAGE_NASA_ERROR'
);
export const setSearch = createAction<SetSearchPayload>(
    'SEARCH_PAGE_SET_SEARCH'
);
```

```
const initState: SearchPageState = {
    data: null,
    isLoading: true,
    error: null,
    search: '',
};

export default handleActions<SearchPageState>(
    {
        [searchPageRequest]: (state) ⇒ ({ ...state, isLoading: true }),
        [searchPageSuccess]: (state, payload: SearchPageSuccessPayload) ⇒ ({
            ...state,      You, 10 minutes ago • Uncommitted changes
            data: payload.data,
        }),
        [searchPageError]: (state, payload: SearchPageErrorPayload) ⇒ ({
            ...state,
            error: payload.error,
        }),
        [setSearch]: (state, payload: SetSearchPayload) ⇒ ({
            ...state,
            search: payload.search,
        }),
    },
    initState
);
```

Solutions to reduce boilerplate

Search Page store (Typescript)

	Lines (with Prettier)	Symbols	Additional size (kB)
raw redux	152	3844	-
redux-act	99 (1,5x less)	2680 (1,4x less)	143
redux-actions	98 (1,5x less)	2695 (1,4x less)	84.5

Redux Toolkit (@reduxjs/toolkit)

createSlice - a function that accepts an initial state, an object full of reducer functions, and a "slice name", and **automatically generates action creators** and action types that correspond to the reducers and state.

```
function createSlice({
    // An object of "case reducers". Key names will be used to generate actions.
    reducers: Object<string, ReducerFunction | ReducerAndPrepareObject>
    // The initial state for the reducer
    initialState: any,
    // A name, used in action types
    name: string,
    // An additional object of "case reducers". Keys should be other action types.
    extraReducers?:
    | Object<string, ReducerFunction>
    | ((builder: ActionReducerMapBuilder<State>) => void)
})
```

@reduxjs/toolkit - createSlice

```
You, 2 minutes ago | 1 author (You)
export interface SearchPageSuccessPayload {
    data: LibraryResponse;
}

You, 2 minutes ago | 1 author (You)
export interface SetSearchPayload {
    search: string;
}

You, 2 minutes ago | 1 author (You)
export interface SearchPageErrorPayload {
    error: string;
}

type ReducerSlice<Payload = any> = CaseReducer<
    SearchPageState,
    PayloadAction<Payload>
>;
```

```
export const fetchSearchData = ({ search }: { search: string }) => async (
    dispatch
) => {
    dispatch(actions.loadRequest({}));
    try {
        const res = await axios.get(`${API_URL}?q=${search}`);
        dispatch(actions.loadSuccess({ data: res.data?.collection }));
    } catch (error) {
        actions.loadError({ error });
    }
};
```

```
const loadRequest: ReducerSlice = (state) => ({
    ...state,
    isLoading: true,
});

const loadSuccess: ReducerSlice<SearchPageSuccessPayload> = (
    state,
    { payload }
) => ({
    ...state,
    data: payload.data,
});

const loadError: ReducerSlice<SearchPageErrorPayload> = (
    state,
    { payload }
) => ({
    ...state,
    error: payload.error,
});

const setSearch: ReducerSlice<SetSearchPayload> = (state, { payload }) => ({
    ...state,
    search: payload.search,
});

const { actions, reducer } = createSlice({
    initialState,
    name: 'SEARCH_PAGE',
    reducers: {
        loadRequest,
        loadSuccess,
        loadError,
        setSearch,
    },
});
```

Solutions to reduce boilerplate

Search Page store (Typescript)

	Lines (with Prettier)	Symbols	Package size
raw redux	152	3844	-
redux-act	99 (1,5x less)	2680 (1,4x less)	143 kB
redux-actions	98 (1,5x less)	2695 (1,4x less)	84.5 kB
@reduxjs/toolkit	103 (1,5x less)	2311 (1,7x less)	1.36 MB

@reduxjs/toolkit – createSlice + createAsyncThunk

```
import { createAsyncThunk } from '@reduxjs/toolkit';

export const fetchSearchData = createAsyncThunk(
    'SEARCH_PAGE_DATA',
    async (search: string) => {
        const res = await Axios.get(`${API_URL}?q=${search}`);
        return res.data?.collection as LibraryResponse;
    }
);
```

dispatch(**fetchSearchData**('sun'))

```
const setSearch: ReducerSlice<SetSearchPayload> = (state, { payload }) => ({
    ...state,
    search: payload.search,
});

const { actions, reducer } = createSlice({
    initialState,
    name: 'SEARCH_PAGE',
    reducers: {
        setSearch,
    },
    extraReducers: (builder) => {
        builder.addCase(fetchSearchData.pending, (state) => ({
            ...state,
            isLoading: false,
        }));
        builder.addCase(fetchSearchData.fulfilled, (state, action) => ({
            ...state,
            data: action.payload,
            isLoading: false,
        }));
        builder.addCase(fetchSearchData.rejected, (state, action) => ({
            ...state,
            error: action.payload,
            isLoading: false,
        }));
    },
});
```

Redux Toolkit Type safety

```
const usersSlice = createSlice({
  name: 'users',
  initialState,
  reducers: {
    // fill in primary logic here
  },
  extraReducers: builder => {
    builder.addCase(fetchUserById.pending, (state, action) => {
      // both `state` and `action` are now correctly typed
      // based on the slice state and the `pending` action creator
    })
  }
})
```

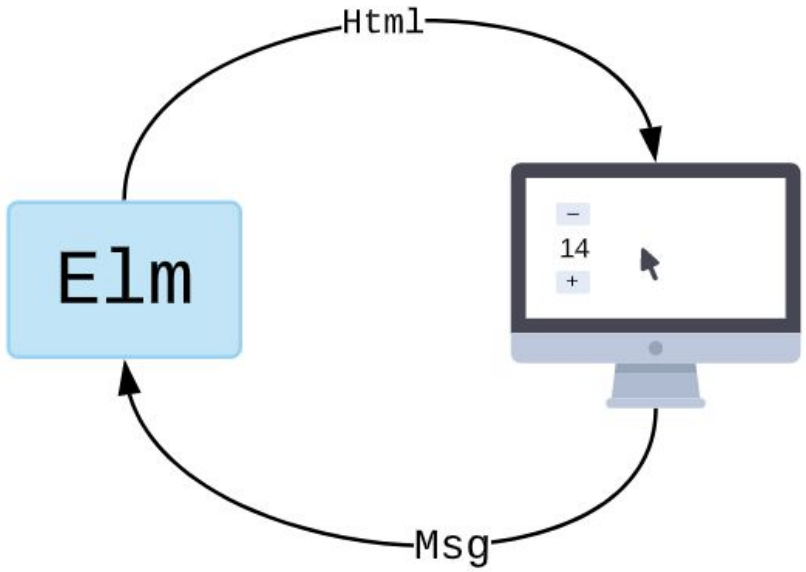
Solutions to reduce boilerplate

Search Page store (Typescript)

	Lines (with Prettier)	Symbols	Package size
raw redux	152	3844	-
redux-act	99 (1,5x less)	2680 (1,4x less)	143 kB
redux-actions	98 (1,5x less)	2695 (1,4x less)	84.5 kB
@reduxjs/toolkit	80 (1,9x less)	2189 (1,8x less)	1.36 MB

Elm is a functional language that compiles to JavaScript.

It has a strong emphasis on simplicity and quality tooling.



State management with Elm

```
type alias Model =
  { name : String
  , password : String
  , passwordAgain : String
  }


init : Model
init =
  Model "" "" ""
```

```
type Msg
  = Name String
  | Password String
  | PasswordAgain String


update : Msg -> Model -> Model
update msg model =
  case msg of
    Name name ->
      { model | name = name }

    Password password ->
      { model | password = password }

    PasswordAgain password ->
      { model | passwordAgain = password }
```

```
view : Model -> Html Msg
view model =
  div []
    [ viewInput "text" "Name" model.name Name
    , viewInput "password" "Password" model.password Password
    , viewInput "password" "Re-enter Password" model.passwordAgain PasswordAgain
    , viewValidation model
    ]


viewInput : String -> String -> String -> (String -> msg) -> Html msg
viewInput t p v toMsg =
  input [ type_ t, placeholder p, value v, onInput toMsg ] []
```

ELM:

```
update : Msg -> Model -> Model
```

JS:

const update = action => State => NewState

Wouldn’t that be great….

```
const initialState = {
    name: ‘’,
    password: ‘’
}

const setName = ({ value }) => state => ({ …state, name: value })

const setPassword = ({ value }) => state => ({ …state, password : value
})
```

Wouldn’t that be even greater….

```
const initialState = {
        name: ‘’,
        password: ‘’
}
const setName = ({ value }) => state => ({ …state, name: value })
const setPassword = ({ value }) => state => ({ …state, password : value })
```

Reducer

Action Creators

Types inference

Redux-blaze

```
import { buildReducer } from "redux-blaze";
import { combineReducers } from 'redux'

export const { actionCreators, reducer: filtersReducer } = buildReducer(initialState, {
    setMySearch: ({ search }) /* <- payload */ => state => ({ ...state, search }),
    setCategory: ({ category }) => state => ({ ...state, category }),
    setSort: ({ sort }) => state => ({ ...state, sort }),
}, {
    prefix: 'MY_FILTER',
});
```

```
// dispatch an action:
dispatch(actionCreators.setCategory({category: 'my category'}))
```

{
type: “MY_FILTER_SET_CATEGORY”
category: ‘my category’
}

redux-blaze buildReducer

```
const { bind, actionCreators, reducer } = buildReducer(
    initialState,
    {
        loadRequest,
        loadSuccess,
        loadError,
    },
    { prefix }
);
```

- reducer – ready to use autogenerated reducer
- actionCreators – ready to use strictly typed actions
- bind – return function for binding actions to dispatch

Easy to create High-order Reducers

```
const loadRequest: Reducer<{}> = () ⇒ (s) ⇒ ({ ...s, isLoading: true });

const loadSuccess: Reducer<{ data: TModel }> = ({ data }) ⇒ (s) ⇒ ({
    ...s,
    data,
    isLoading: false,
    error: false,
});

const loadError: Reducer<{ error: any }> = ({ error }) ⇒ (s) ⇒ ({
    ...s,
    error,
    isLoading: false,
});

const { bind, actionCreators, reducer } = buildReducer(
    initialState,
    {
        loadRequest,
        loadSuccess,
        loadError,
    },
    { prefix }
);
```

```
const fetchData = (...args) ⇒ async (dispatch) ⇒ {
    const { loadError, loadRequest, loadSuccess } = bind(dispatch);
    loadRequest({});
    try {
        const data = await fetcher(...args);
        loadSuccess({ data: mutate(data) });
    } catch (error) {
        loadError({ error });
    }
};
```

createReduxFetcher

```ts
import { API_URL } from './constants';
import { createReduxFetcher } from '../createReduxFetcher';
import { LibraryResponse } from '../../models';
import Axios from 'axios';

export const EarthFetcher = createReduxFetcher<LibraryResponse>({
    fetcher: () => Axios.get(API_URL),
    getState: (state) => state.earthPage,
    mutate: (res) => res.data?.collection,
    prefix: 'EARTH_PAGE',
});      You, 18 days ago • add redux-blaze

export default EarthFetcher.reducer;
```

```ts
import React from 'react';

import CardGrid from '../../components/ImagesGrid';
import { EarthFetcher } from '../../store/EarthPage/reducer';

function EarthPage() {
    const { data, isLoading } = EarthFetcher.useData();

    return <CardGrid items={data?.items} isLoading={isLoading} />;
}

export default EarthPage;
```

react-redux-blaze - createReduxFetcher

```
export const JupiterFetcher = createReduxFetcher<LibraryResponse>({
    fetcher: () ⇒ Axios.get(API_URL),
    getState: (state) ⇒ state.jupiterPage,
    mutate: (res) ⇒ res.data?.collection,
    prefix: 'JUPITER_PAGE',
});
```

```
function JupiterPage() {
    const { data, isLoading } = JupiterFetcher.useData();

    return <CardGrid items={data?.items} isLoading={isLoading} />;
}
```

- All data keeps in redux global store
- Strictly typed
- Autogenerated reducer
- Handles loading and error states
- **useData** hook
- Built in data selector
- Prefetching data

Another example “Earth page”

Search for NASA images and videos

Picture of the day | Earth | Moon | Uranus | Jupiter | Mars | Mercury | Neptune | Saturn | Venus | Pluto

A Waterspout in Florida

What's happening over the water? Pictured here is one of the better images yet recorded of a waterspout, a type of tornado that occurs over water. Waterspouts are spinning columns of rising moist air that typically form over warm water. Waterspouts can be as dangerous as tornadoes and can feature wind speeds over 200 kilometers per hour. Some waterspouts form away from thunderstorms and even during relatively fair weather. Waterspouts may be relatively transparent and initially visible only by an unusual pattern they create on the water. The featured image was taken in 2013 July near Tampa Bay, Florida. The Atlantic Ocean off the coast of Florida is arguably the most active area in the world for waterspouts, with hundreds forming each year. Experts Debate: How will humanity first discover extraterrestrial life?

```
import axios from 'axios';
import {
    earthPageError,
    earthPageRequest,
    earthPageSuccess,
} from './actionsCreators';

import { API_URL } from './constants';

export const fetchEarthData = () => async (dispatch) => {
    dispatch(earthPageRequest());
    try {
        const res = await axios.get(API_URL);
        dispatch(earthPageSuccess({ data: res.data?.collection }));
    } catch (error) {
        earthPageError({ error });
    }
};
```

```
}
            data,
        },
    });

export const earthPageError = ({ error }): Pa
    type: EARTH_PAGE_NASA_ERROR,
    payload: {
        error,
    },
});

export type ActionsTypes = PageRe
```

```
export const EARTH_PAGE_NASA_REQUEST = 'EARTH_PA
export const EARTH_PAGE_NASA_SUCCESS = 'EARTH_PA
export const EARTH_PAGE_NASA_ERROR = 'EARTH_PAGE
```

```
const initState: EarthPageState = {
    data: null,
    isLoading: true,
    error: null,
};

const EarthReducer = (state = initState, action: ActionsTypes) => {
    switch (action.type) {
        case EARTH_PAGE_NASA_REQUEST: {
            return {
                ...state,
                isLoading: true,
            };
        }
        case EARTH_PAGE_NASA_SUCCESS: {
            return {
                ...state,
                isLoading: false,
                data: action.payload.data,
            };
        }
        case EARTH_PAGE_NASA_ERROR: {
            return {
                ...state,
```

```
const getState = (state: AppState) => state.earthPage;

export const selectData = createSelector(getState, (state) => state.data);

export const selectIsLoading = createSelector(
    getState,
    (state) => state.isLoading
);
```

Redux-blaze

```
export const SearchFetcher = createReduxFetcher<LibraryResponse>({
    fetcher: (search) => Axios.get(`${API_URL}?q=${search}`),
    getState: (state) => state.searchPage.model,
    mutate: (res) => res?.data?.collection,
    prefix: 'SEARCH_PAGE',
});
```

```
const SearchFiltersInitialState: SearchFiltersState = {
    search: '',
};

const setSearch = (a: { search: string }) => (s): SearchFiltersState => ({
    ...s,
    search: a.search,
});

const clearSearch = () => setSearch({ search: '' });
```

```
export const SearchPageFilters = buildReducer(
    SearchFiltersInitialState,
    {
        setSearch,
        clearSearch,
    },
    { prefix: 'SEARCH_PAGE_FILTERS' }
);
```

```
export default combineReducers({
    filters: SearchPageFilters.reducer,
    model: SearchFetcher.reducer,
});
```

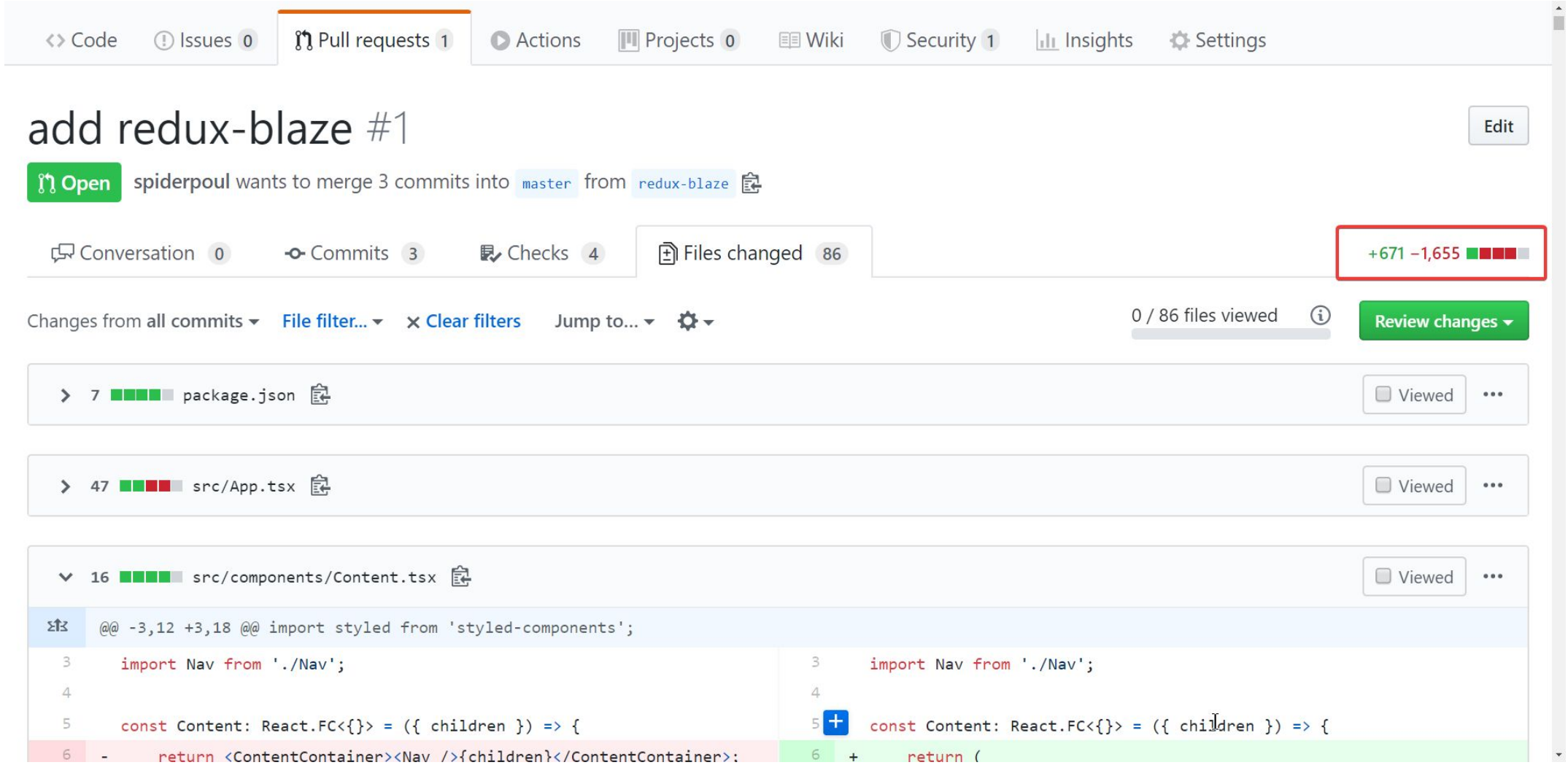
```
function SearchPage() {
    const dispatch = useDispatch();
    const search = useSelector(selectSearch);
    const searchDebounce = useDebounce(search);
    const querySearch = useQuerySearch();

    const { setSearch } = SearchPageFilters.bind(dispatch);

    const { data, isLoading } = SearchFetcher.useData(searchDebounce);

    useEffect(() => {
        if (!search && querySearch) {
            dispatch(setSearch({ search: querySearch }));
        }
    }, []);

    return data?.items?.length || isLoading ? (
        <CardGrid items={data?.items?.slice(0, 10)} isLoading={isLoading} />
    ) : (
        <Placeholder>No results found.</Placeholder>
    );
}
```

Solutions to reduce boilerplate

Search Page store (Typescript)

	Lines (with Prettier)	Symbols	Package size
raw redux	152	3844	-
redux-act	99 (1,5x less)	2680 (1,4x less)	143 kB
redux-actions	98 (1,5x less)	2695 (1,4x less)	84.5 kB
@reduxjs/toolkit	80 (1,9x less)	2189 (1,8x less)	1.36 MB
redux-blaze	**43** **(3,5x less)**	**1309** **(2.9x less)**	22 kB

After refactoring with redux-blaze



createSlice vs redux-blaze

Pros:

- Customization

Cons:

- Overcomplicated typing for TS
- Poor type inference
- Complicated configs
- Read a lot of documentation before use

Redux-blaze

- Easy to understand and use
- Easy to use with Typescript
- Intellisense works even with plain JS!
- 2-3x less code
- Strictly typed out of the box
- Light-weight!


```
import axios from 'axios';
import {
    earthPageError,
    earthPageRequest,
    earthPageSuccess,
} from './actionsCreators';

import { API_URL } from './constants';

export const fetchEarthData = () => async (dispatch) => {
    dispatch(earthPageRequest());
    try {
        const res = await axios.get(API_URL);
        dispatch(earthPageSuccess({ data: res.data?.collection }));
    } catch (error) {
        earthPageError({ error });
    }
};
```

```
): PageRequ
,

{ data }):
,

error }): P
```

```
    payload: {
        error: any;
    };
}

export type ActionsTypes = PageRequest | Pa
```

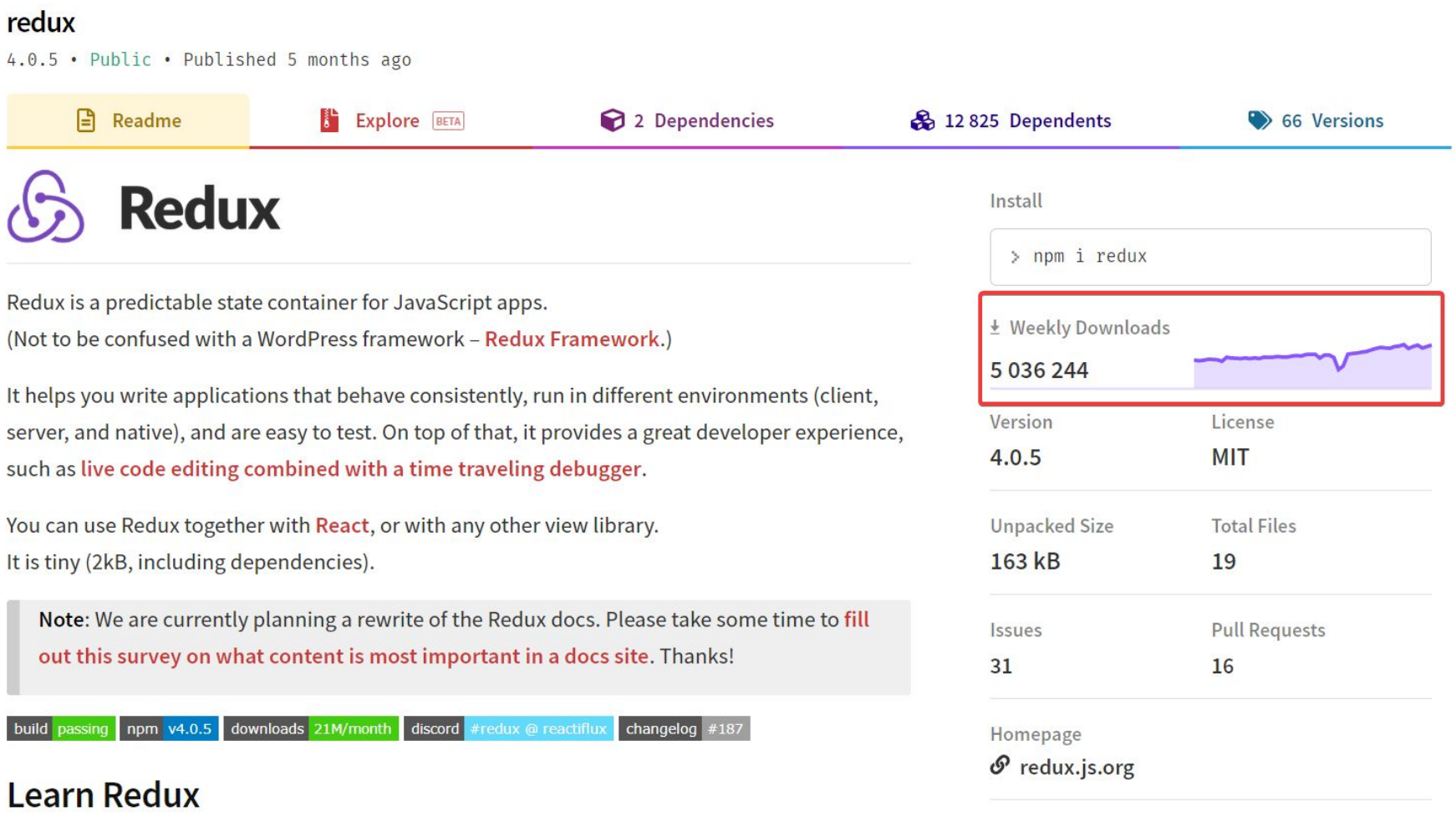
```
const initState: EarthPageState = {
    data: null,
    isLoading: true,
    error: null,
};

const EarthReducer = (state = initState, action: ActionsTypes) => {
    switch (action.type) {
        case EARTH_PAGE_NASA_REQUEST: {
            return {
                ...state,
                isLoading: true,
            };
        }
        case EARTH_PAGE_NASA_SUCCESS: {
            return {
                ...state,
                isLoading: false,
                data: action.payload.data,
            };
        }
        case EARTH_PAGE_NASA_ERROR: {
            return {
```

```
const getState = (state: AppState) => state.earthPage;

export const selectData = createSelector(getState, (state) => state.da

export const selectIsLoading = createSelector(
    getState,
    (state) => state.isLoading
);
```

```
export const SearchFetcher = createReduxFetcher<LibraryResponse>({
    fetcher: (search) => Axios.get(`${API_URL}?q=${search}`),
    getState: (state) => state.searchPage.model,
    mutate: (res) => res?.data?.collection,
    prefix: 'SEARCH_PAGE',
});
```

```
export const SearchPageFilters = buildReducer(
    SearchFiltersInitialState,
    {
        setSearch,
        clearSearch,
    },
    { prefix: 'SEARCH_PAGE_FILTERS' }
);
```

Is Redux alive?



redux

4.0.5 • Public • Published 5 months ago

Readme | Explore BETA | 2 Dependencies | 12 825 Dependents | 66 Versions

Redux

Redux is a predictable state container for JavaScript apps.
(Not to be confused with a WordPress framework – Redux Framework.)

It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. On top of that, it provides a great developer experience, such as live code editing combined with a time traveling debugger.

You can use Redux together with React, or with any other view library.
It is tiny (2kB, including dependencies).

> **Note**: We are currently planning a rewrite of the Redux docs. Please take some time to fill out this survey on what content is most important in a docs site. Thanks!

build passing | npm v4.0.5 | downloads 21M/month | discord #redux @ reactiflux | changelog #187

Learn Redux

Install

```
> npm i redux
```

Weekly Downloads
5 036 244

Version 4.0.5
License MIT

Unpacked Size 163 kB
Total Files 19

Issues 31
Pull Requests 16

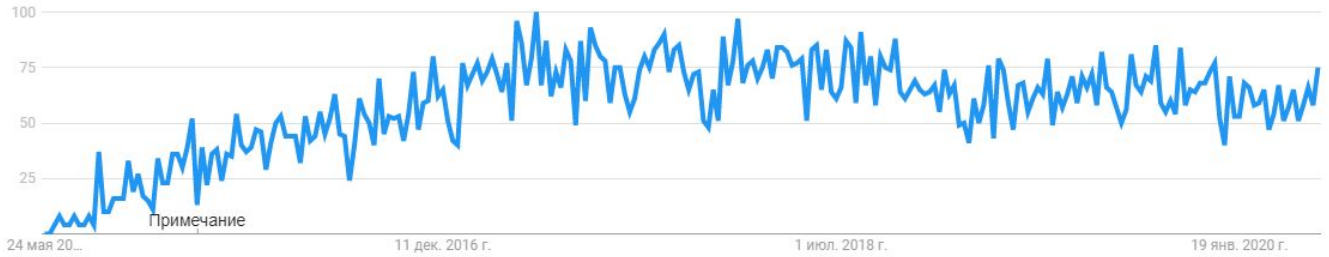
Homepage
redux.js.org

redux js
Поисковый запрос

+ Сравнить

По всему миру | Последние 5 лет | Все категории | Веб-поиск

Динамика популярности

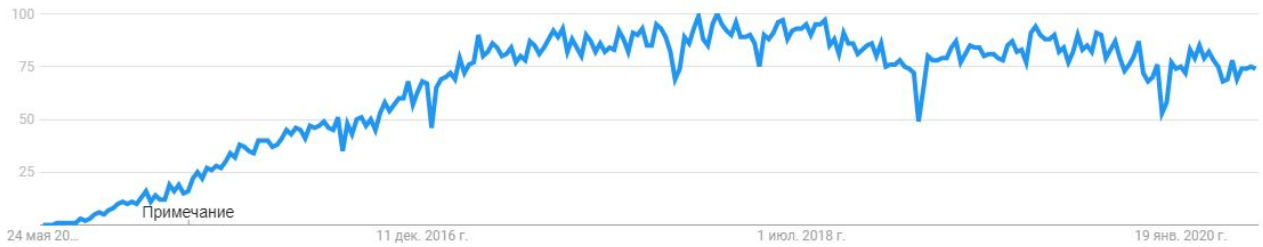


redux react
Поисковый запрос

+ Сравнить

По всему миру | Последние 5 лет | Все категории | Веб-поиск

Динамика популярности



Should we use Redux in 2020 for React Apps?

I’d use Redux (with redux-blaze) for:

- complex apps (a lot of forms, filters and etc.) and teams more 4-5 members
- Instead of multiple React Contexts

If no reason for Redux:

- React Hooks
- Hooks for Fetching (SWR, React Query)

redux-blaze
https://github.com/spiderpoul/redux-blaze
https://github.com/spiderpoul/react-redux-blaze

Nasa App
https://nasa-app-react.netlify.app/
https://github.com/spiderpoul/nasa-app-react/tree/redux-blaze
https://github.com/spiderpoul/nasa-app-react/tree/redux

https://github.com/zeit/swr
https://github.com/tannerlinsley/react-query

Спасибо за внимание!

комьюнити Нижнего Новгорода