Startel

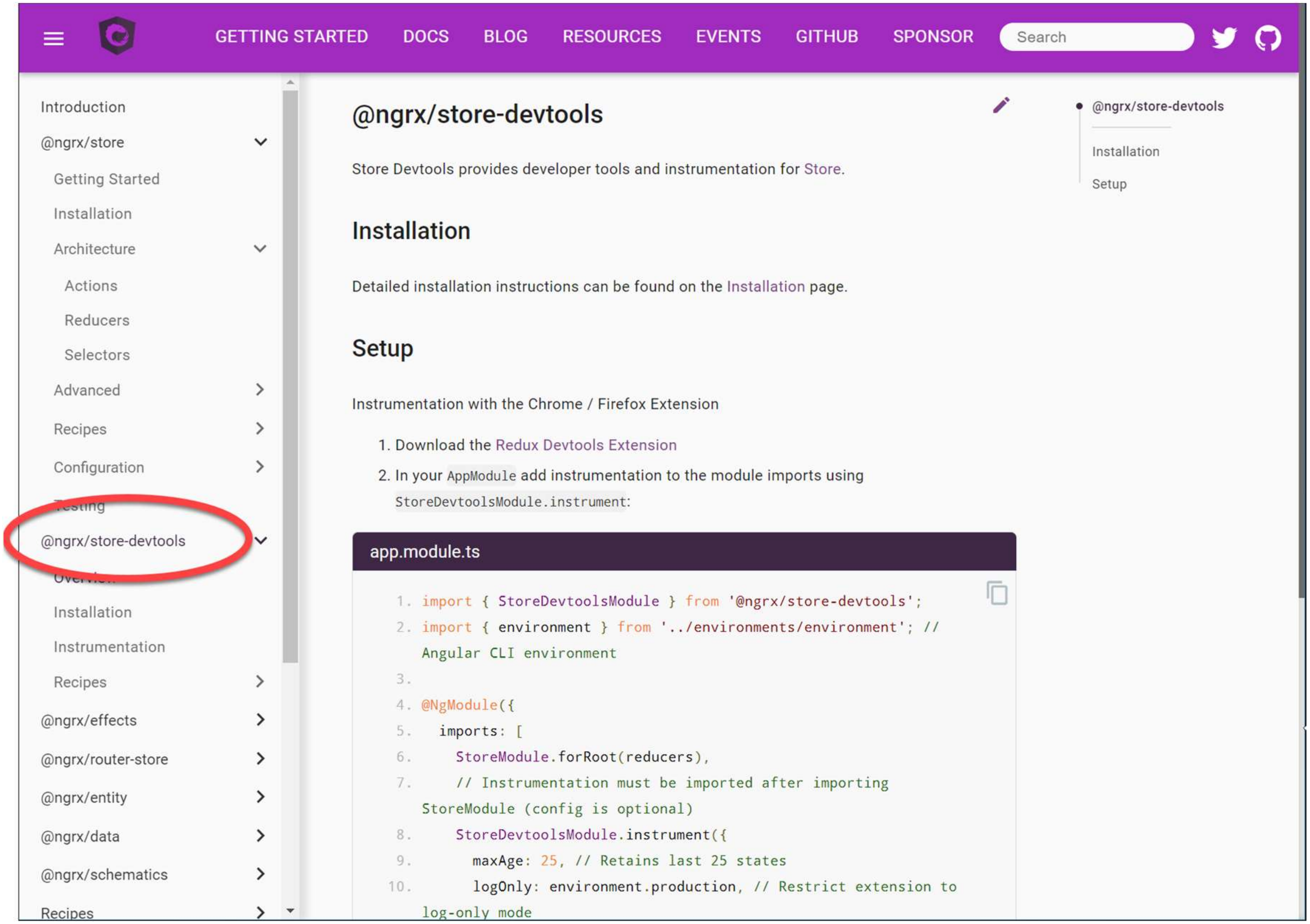
Store introspection with `@ngrx/store-devtools`

Peter Kassenaar
info@kassenaar.com

What is @ngrx/store-devtools?

- Analyzing, inspection and debugging your State Management Store
- Time Travelling options

https://ngrx.io/guide/store-devtools



@ngrx/store-devtools

Store Devtools provides developer tools and instrumentation for Store.

Installation

Detailed installation instructions can be found on the Installation page.

Setup

Instrumentation with the Chrome / Firefox Extension

1. Download the Redux Devtools Extension
2. In your `AppModule` add instrumentation to the module imports using `StoreDevtoolsModule.instrument`:

app.module.ts

```
import { StoreDevtoolsModule } from '@ngrx/store-devtools';
import { environment } from '../environments/environment'; // Angular CLI environment

@NgModule({
  imports: [
    StoreModule.forRoot(reducers),
    // Instrumentation must be imported after importing StoreModule (config is optional)
    StoreDevtoolsModule.instrument({
      maxAge: 25, // Retains last 25 states
      logOnly: environment.production, // Restrict extension to log-only mode
```

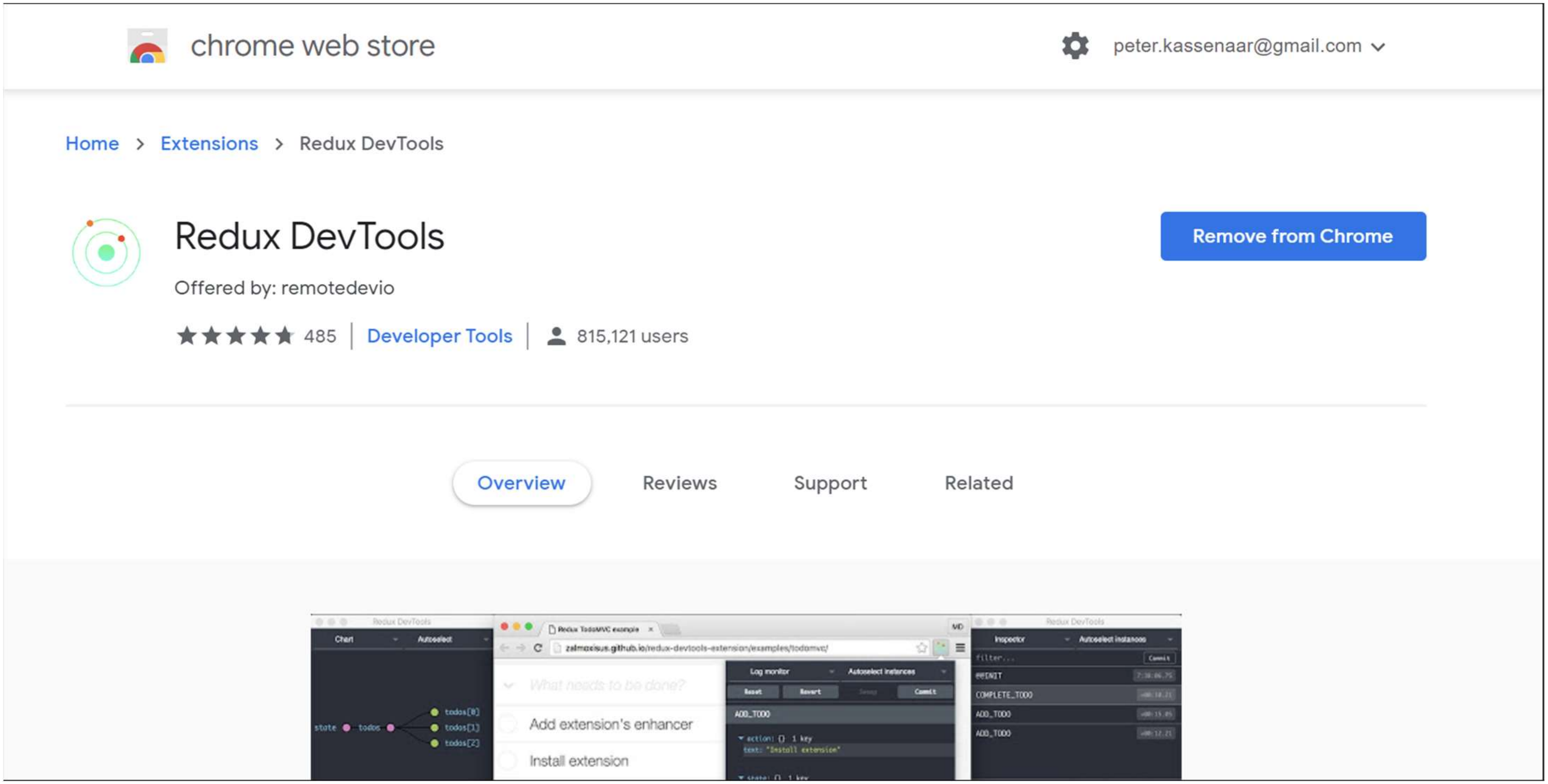
Two Parts

1. *Chrome/Firefox Extension/Add-On*

2. *Installation in your project*

1. Install in Chrome

One-time installation of Redux DevTools Extension



chrome web store

peter.kassenaar@gmail.com

Home > Extensions > Redux DevTools

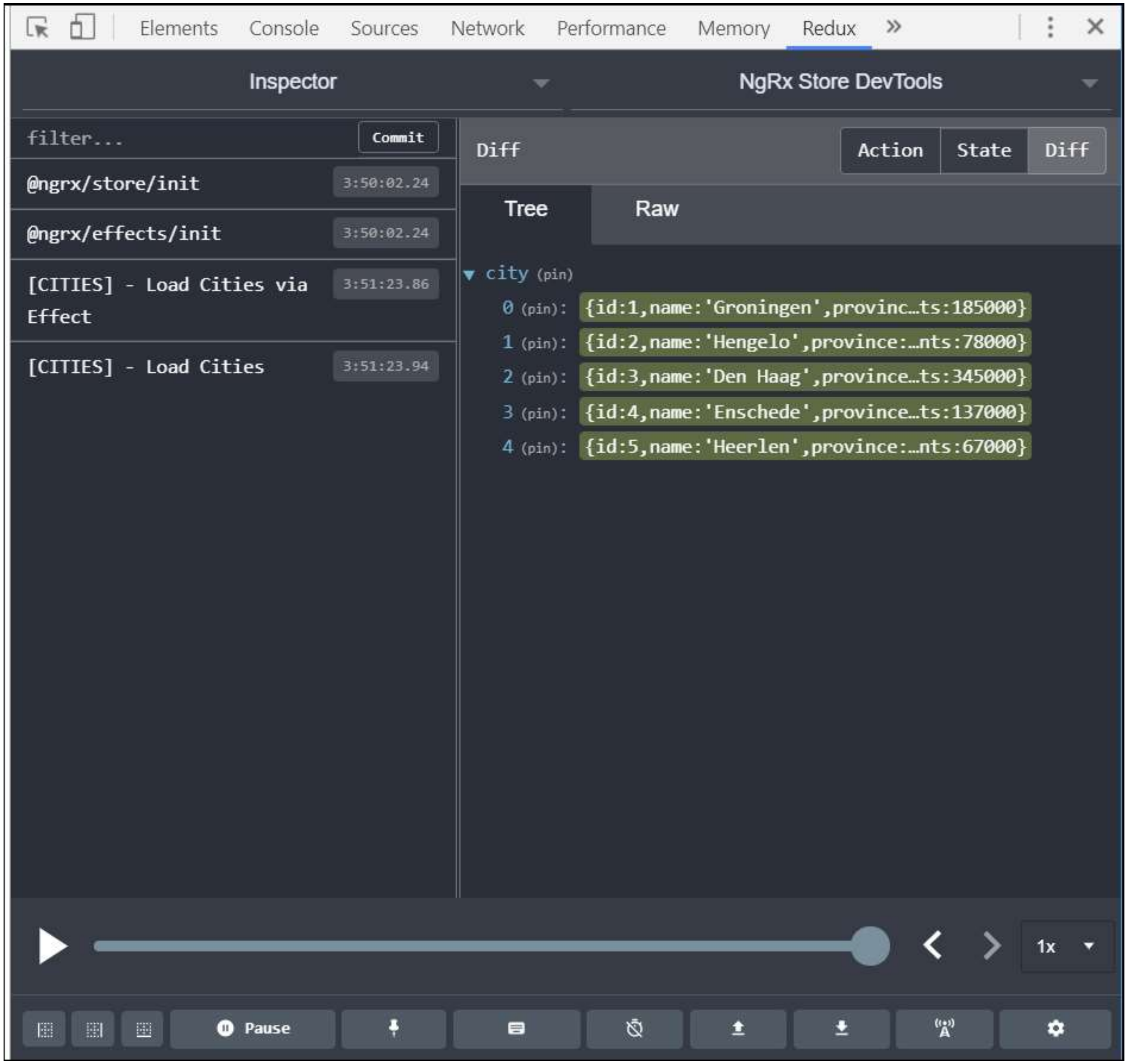
Redux DevTools

Offered by: remotedevio

★★★★★ 485 | Developer Tools | 815,121 users

https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd?hl=en

Chrome Extension

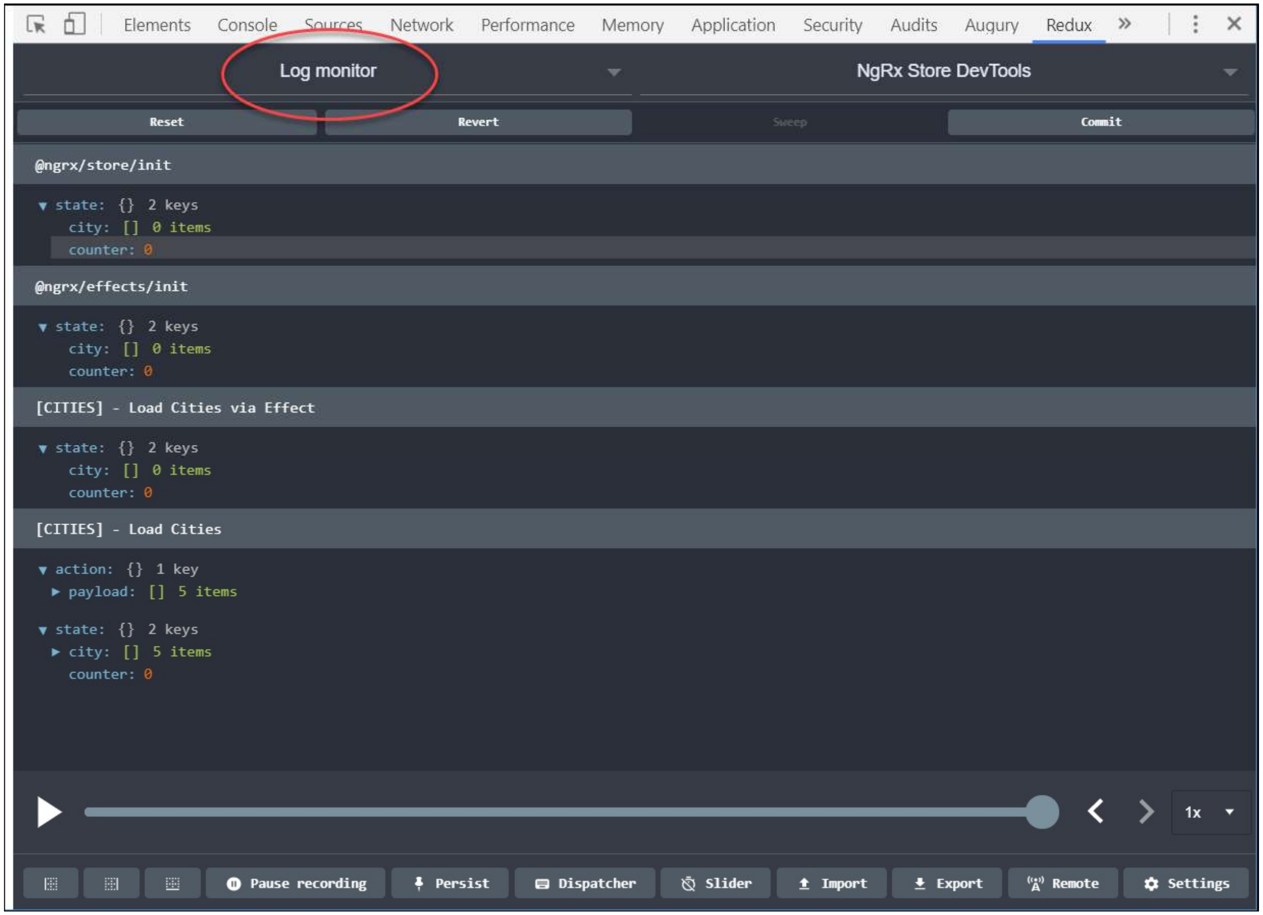


```
@ngrx/store/init                      3:50:02.24
@ngrx/effects/init                    3:50:02.24
[CITIES] - Load Cities via            3:51:23.86
Effect
[CITIES] - Load Cities                3:51:23.94
```

```
▼ city (pin)
    0 (pin): {id:1,name:'Groningen',provinc…ts:185000}
    1 (pin): {id:2,name:'Hengelo',province:…nts:78000}
    2 (pin): {id:3,name:'Den Haag',province…ts:345000}
    3 (pin): {id:4,name:'Enschede',province…ts:137000}
    4 (pin): {id:5,name:'Heerlen',province:…nts:67000}
```

Options

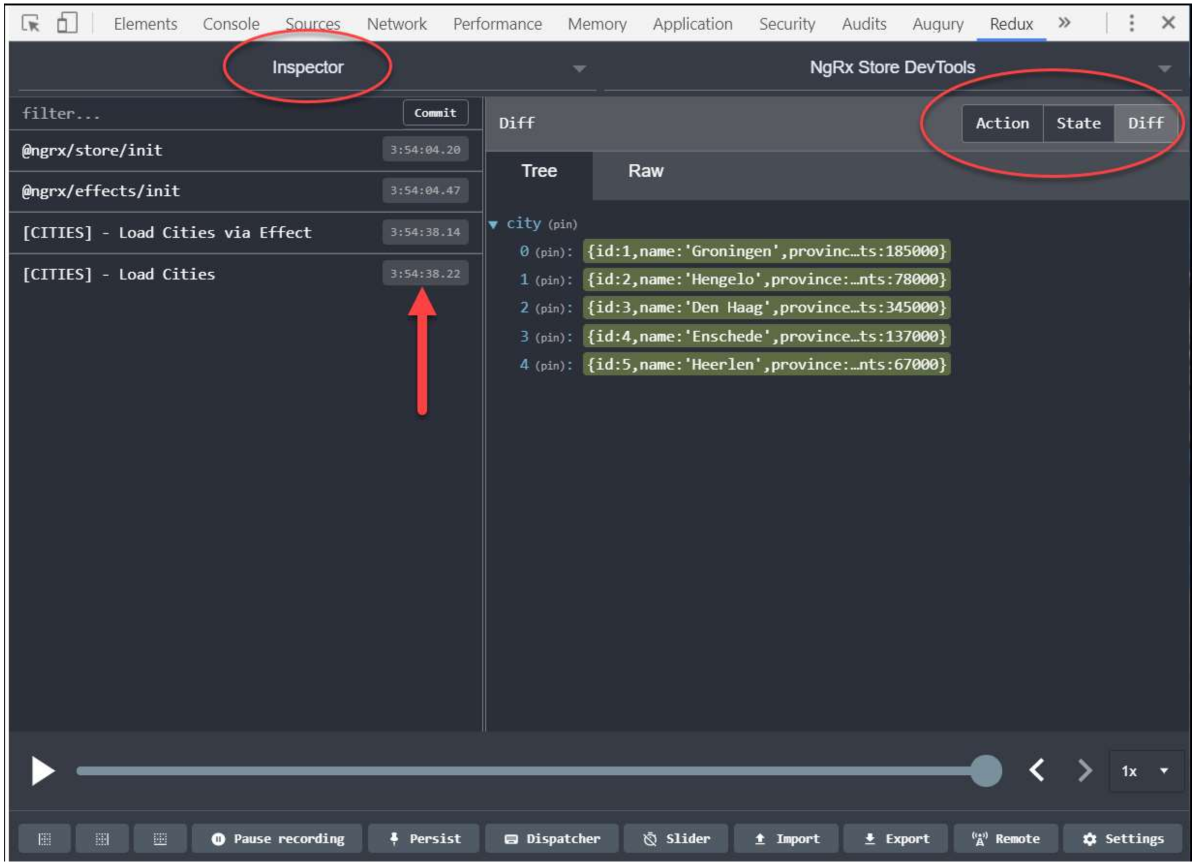
- Log Monitor
  - Log actions from/to the store and see changes in Real Time
  - Reset/Revert actions



```
@ngrx/store/init
▼ state: {} 2 keys
   city: [] 0 items
   counter: 0
@ngrx/effects/init
▼ state: {} 2 keys
   city: [] 0 items
   counter: 0
[CITIES] - Load Cities via Effect
▼ state: {} 2 keys
   city: [] 0 items
   counter: 0
[CITIES] - Load Cities
▼ action: {} 1 key
 ▶ payload: [] 5 items
▼ state: {} 2 keys
 ▶ city: [] 5 items
   counter: 0
```

Inspector

- Filter Actions
- Jump/Skip actions
- Inspect `Action`, `State` or `Diff` compared to previous State

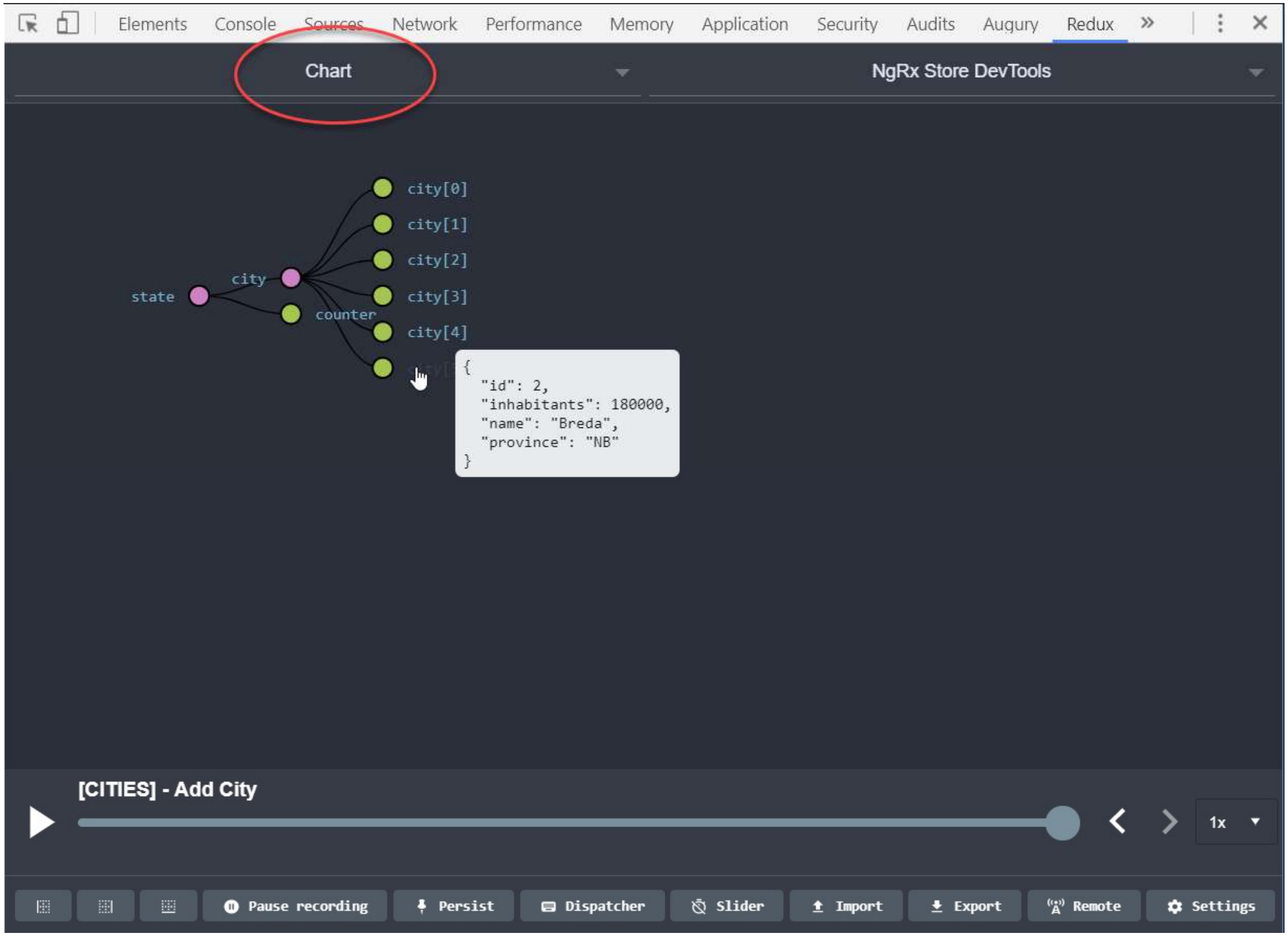


```
@ngrx/store/init                     3:54:04.20
@ngrx/effects/init                   3:54:04.47
[CITIES] - Load Cities via Effect    3:54:38.14
[CITIES] - Load Cities               3:54:38.22
```

```
▼ city (pin)
  0 (pin): {id:1,name:'Groningen',provinc…ts:185000}
  1 (pin): {id:2,name:'Hengelo',province:…nts:78000}
  2 (pin): {id:3,name:'Den Haag',province…ts:345000}
  3 (pin): {id:4,name:'Enschede',province…ts:137000}
  4 (pin): {id:5,name:'Heerlen',province:…nts:67000}
```

Chart

- Inspect flowchart of Actions, flow of data through the Store

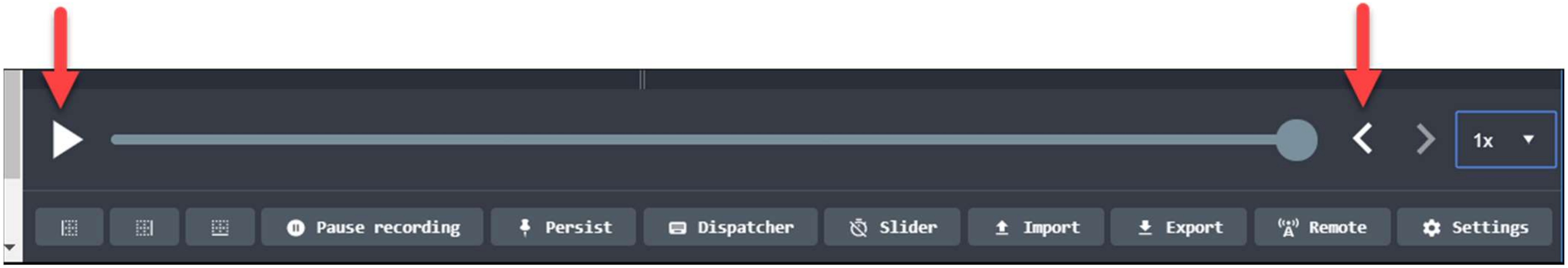


```
{
  "id": 2,
  "inhabitants": 180000,
  "name": "Breda",
  "province": "NB"
}
```

[CITIES] - Add City

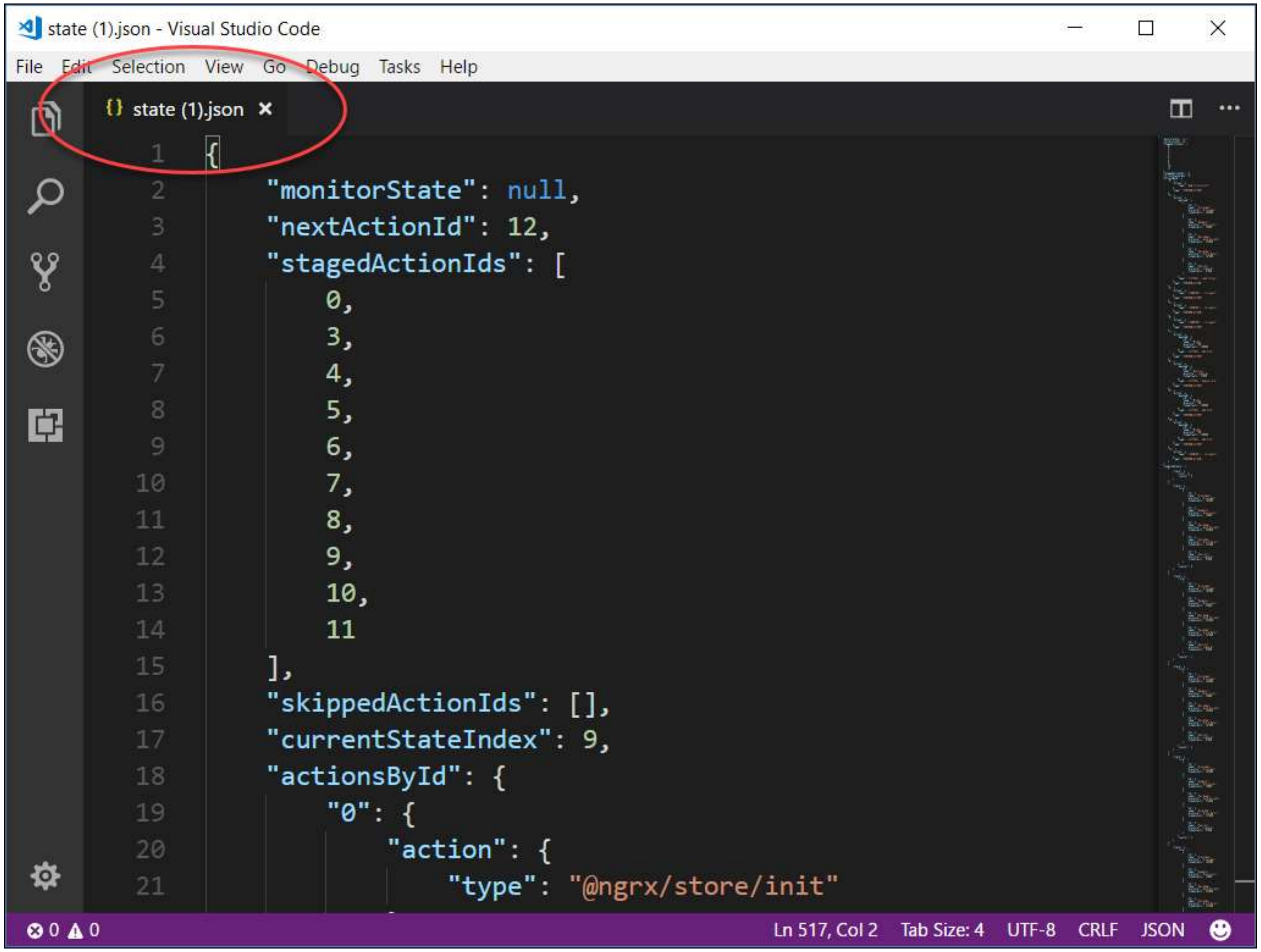
Slider

- (Re)play various actions and see changes in app and in state
- Jump to begin/end of timeline



Import/export State

- Save State changes to .json-file and import them to replay steps



```
{
    "monitorState": null,
    "nextActionId": 12,
    "stagedActionIds": [
        0,
        3,
        4,
        5,
        6,
        7,
        8,
        9,
        10,
        11
    ],
    "skippedActionIds": [],
    "currentStateIndex": 9,
    "actionsById": {
        "0": {
            "action": {
                "type": "@ngrx/store/init"
```

2. Installation

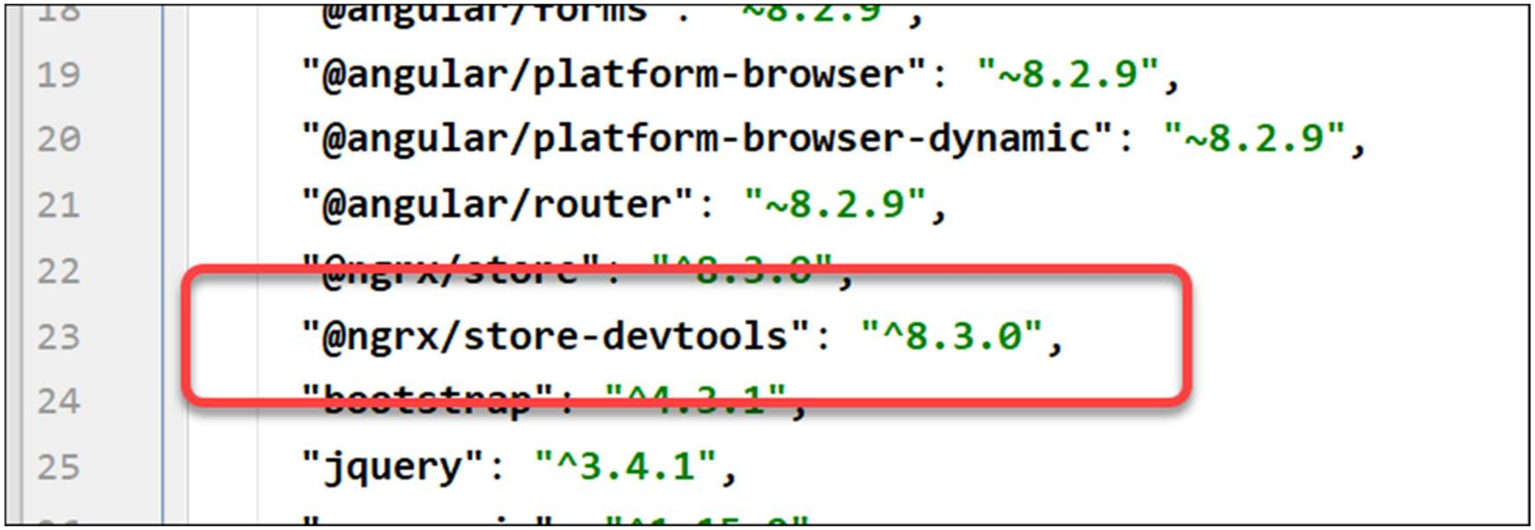
Install in Google Chrome *and* in project

Installation in Project

```
npm install @ngrx/store-devtools --save
```

OR:

```
ng add @ngrx/store-devtools
```



```
    "@angular/forms": "~8.2.9",
    "@angular/platform-browser": "~8.2.9",
    "@angular/platform-browser-dynamic": "~8.2.9",
    "@angular/router": "~8.2.9",
    "@ngrx/store": "^8.3.0",
    "@ngrx/store-devtools": "^8.3.0",
    "bootstrap": "^4.3.1",
    "jquery": "^3.4.1",
```

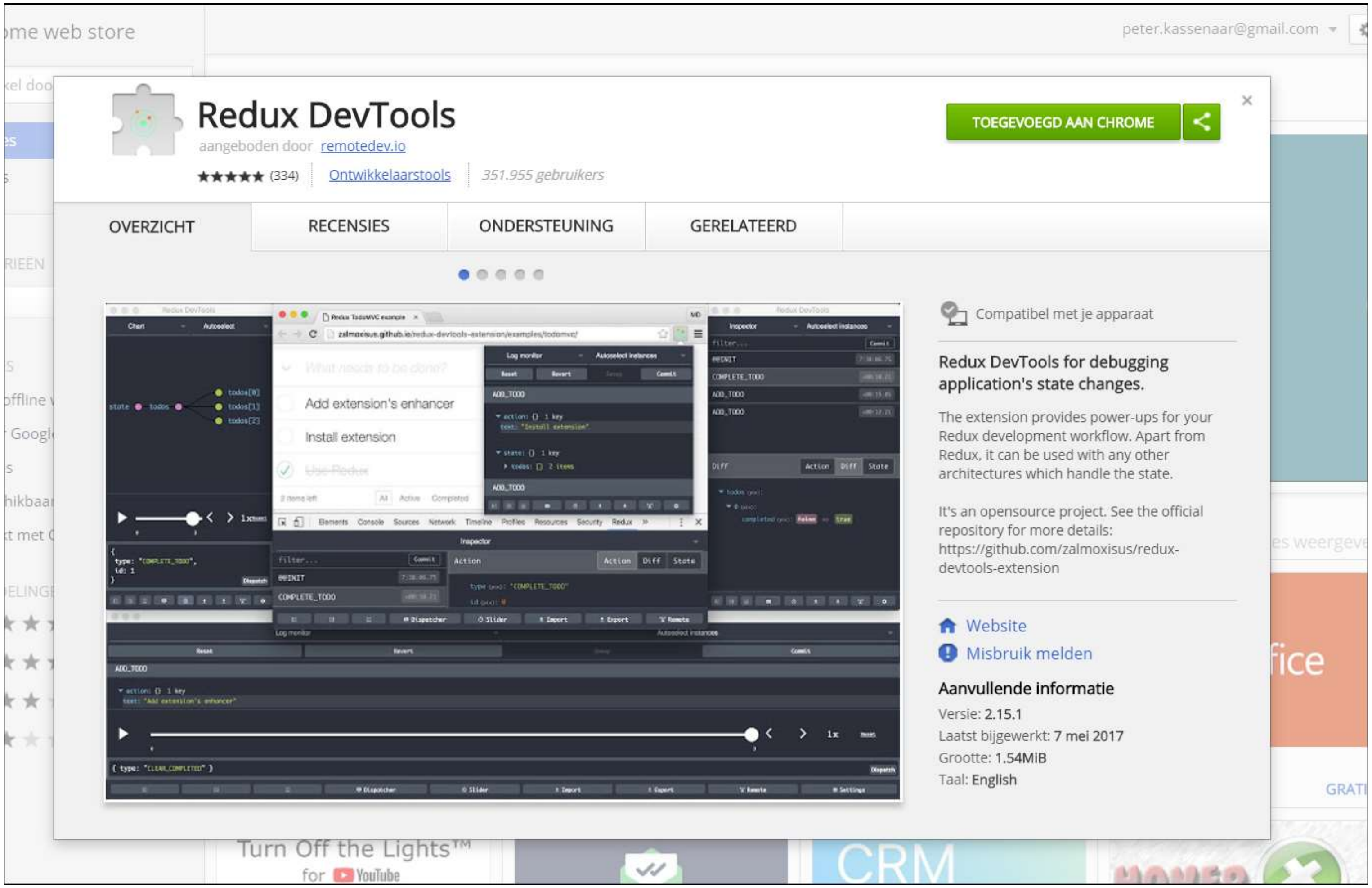
Installation in app.module.ts

```
// app.module.ts
import {StoreDevtoolsModule} from '@ngrx/store-devtools';

@NgModule({
   …
   imports     : [
      …
      StoreDevtoolsModule.instrument({
            maxAge: 10}) // Redux Devtools debugging in Chrome
   ],
})
export class AppModule {
}
```

Install Redux DevTools

- Add from `chrome://extensions`



Redux DevTools

aangeboden door remotedev.io

★★★★★ (334) | Ontwikkelaarstools | *351.955 gebruikers*

Compatibel met je apparaat

Redux DevTools for debugging application's state changes.

The extension provides power-ups for your Redux development workflow. Apart from Redux, it can be used with any other architectures which handle the state.

It's an opensource project. See the official repository for more details: https://github.com/zalmoxisus/redux-devtools-extension

Website

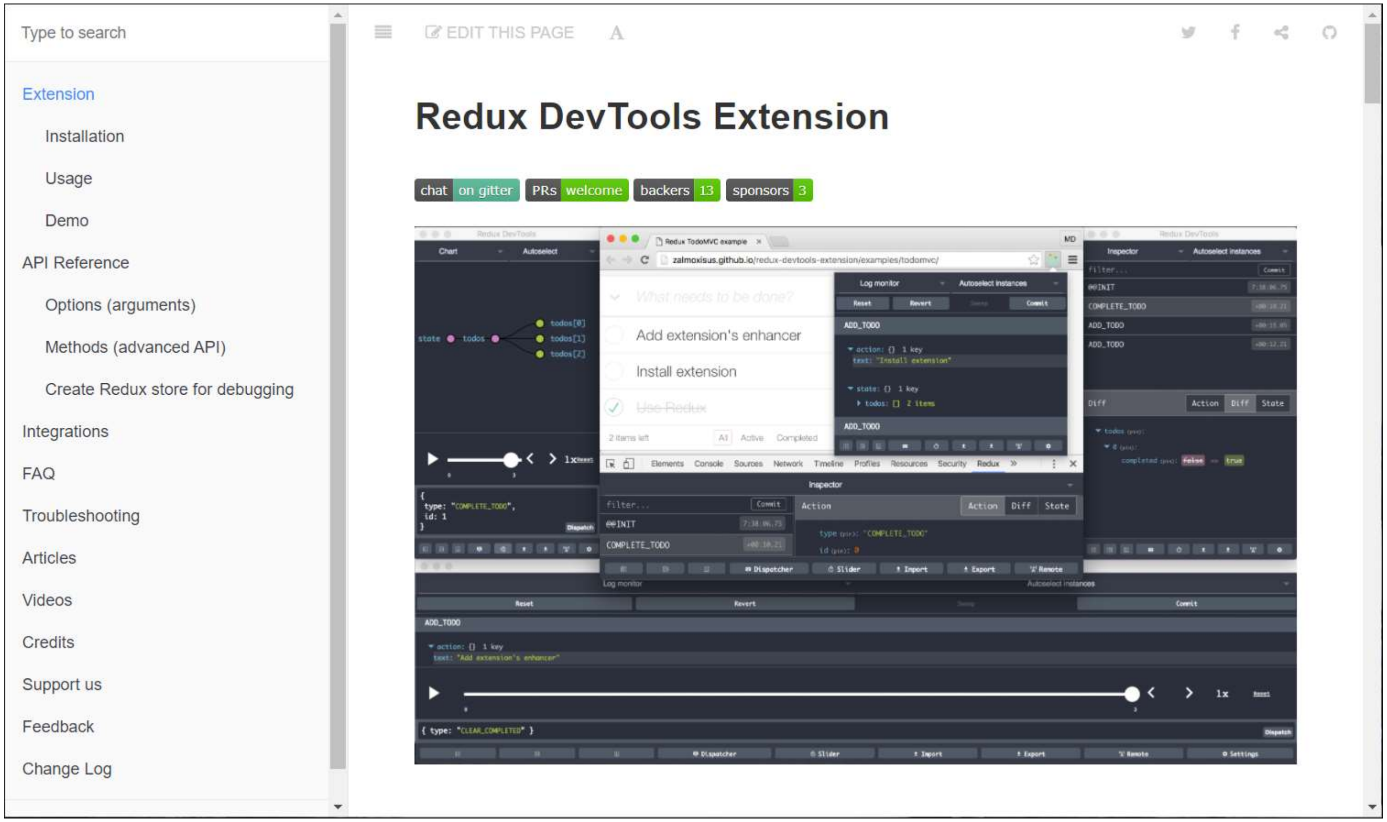
Misbruik melden

Aanvullende informatie

Versie: 2.15.1
Laatst bijgewerkt: 7 mei 2017
Grootte: 1.54MiB
Taal: English

More info

- http://extension.remotedev.io/
- NOT specific to Angular – to be used with all Redux-like stores!



Extension

- Installation
- Usage
- Demo

API Reference

- Options (arguments)
- Methods (advanced API)
- Create Redux store for debugging

Integrations

FAQ

Troubleshooting

Articles

Videos

Credits

Support us

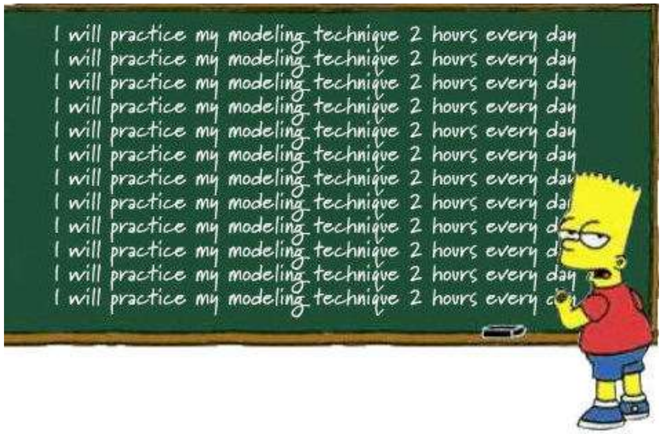
Feedback

Change Log

Redux DevTools Extension

Workshop

- Use your own project, or start from one of the Store projects, for example `/200-ngrx-simple-store`
- Add Redux DevTools extension to Chrome (one-off action)
- Add `@ngrx/store-devtools` to current project via `npm` or `ng add`
- Activate devtools in `app.module.ts`
- Experiment with different views
  - Log Monitor
  - Inspector
  - Chart
- Export a State Tree to `.json` file, reload the project and import it again. See how this works


I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every da
I will practice my modeling technique 2 hours every da
I will practice my modeling technique 2 hours every d
I will practice my modeling technique 2 hours every d
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every d