mobx

6.7.0 • Public • Published 22 days ago

Readme

Code Beta

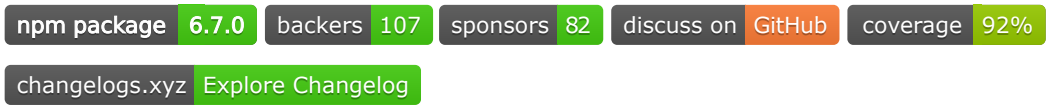
0 Dependencies

3,129 Dependents

232 Versions

MobX

Simple, scalable state management.



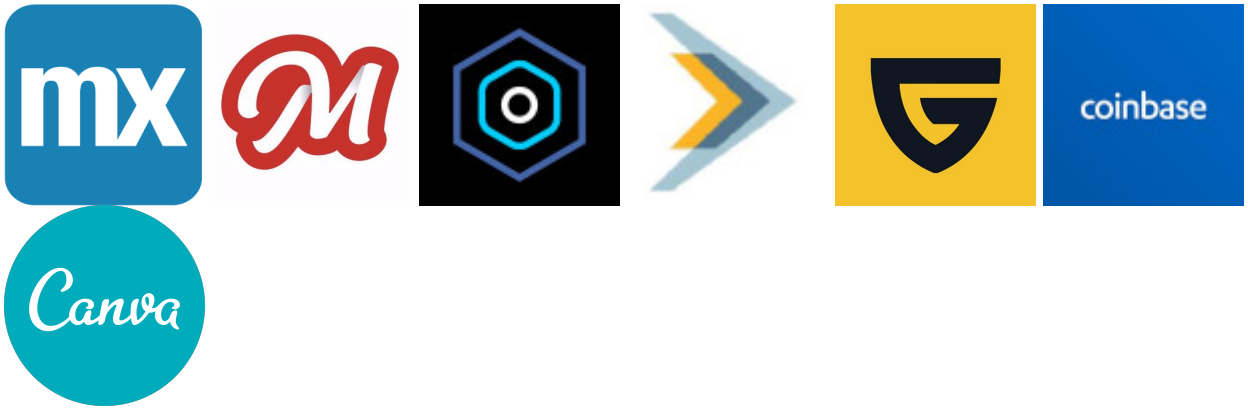
Documentation

Documentation can be found at mobx.js.org.

Sponsors

MobX is made possible by the generosity of the sponsors below, and many other **individual backers**. Sponsoring directly impacts the longevity of this project.

🥇 **Gold sponsors ($3000+ total contribution):**



🥈 **Silver sponsors ($100+ per month):**

🥉 **Bronze sponsors ($500+ total contributions):**

Introduction

Anything that can be derived from the application state, should be. Automatically.

MobX is a battle-tested library that makes state management simple and scalable by transparently applying functional reactive programming. The philosophy behind MobX is simple:

😚

Straightforward

Write minimalistic, boilerplate-free code that captures your intent. Trying to update a record field? Simply use a normal JavaScript assignment — the reactivity system will detect all your changes and propagate them out to where they are being used. No special tools are required when updating data in an asynchronous process.

Effortless optimal rendering

All changes to and uses of your data are tracked at runtime, building a dependency tree that captures all relations between state and output. This guarantees that computations that depend on your state, like React components, run only when strictly needed. There is no need to manually optimize components with error-prone and sub-optimal techniques like memoization and selectors.

Architectural freedom

MobX is unopinionated and allows you to manage your application state outside of any UI framework. This makes your code decoupled, portable, and above all, easily testable.

A quick example

So what does code that uses MobX look like?

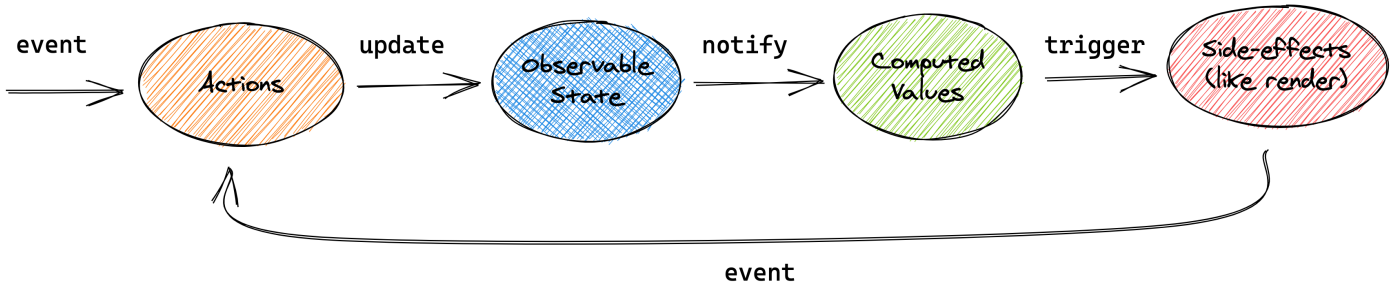
```
import React from "react"
import ReactDOM from "react-dom"
import { makeAutoObservable } from "mobx"
import { observer } from "mobx-react"

// Model the application state.
class Timer {
    secondsPassed = 0
```

```
        constructor() {
            makeAutoObservable(this)
        }

        increase() {
            this.secondsPassed += 1
        }

        reset() {
            this.secondsPassed = 0
        }
    }

    const myTimer = new Timer()

    // Build a "user interface" that uses the observable state.
    const TimerView = observer(({ timer }) => (
        <button onClick={() => timer.reset()}>Seconds passed: {timer
    ))

    ReactDOM.render(<TimerView timer={myTimer} />, document.body)

    // Update the 'Seconds passed: X' text every second.
    setInterval(() => {
        myTimer.increase()
    }, 1000)
```

The `observer` wrapper around the `TimerView` React component will automatically detect that rendering depends on the `timer.secondsPassed` observable, even though this relationship is not explicitly defined. The reactivity system will take care of re-rendering the component when *precisely that* field is updated in the future.

Every event (`onClick` / `setInterval`) invokes an *action* (`myTimer.increase` / `myTimer.reset`) that updates *observable state* (`myTimer.secondsPassed`). Changes in

the observable state are propagated precisely to all *computations* and *side effects* (`TimerView`) that depend on the changes being made.



This conceptual picture can be applied to the above example, or any other application using MobX.

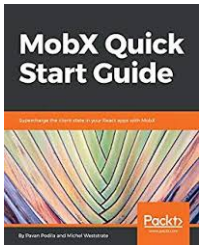
Getting started

To learn about the core concepts of MobX using a larger example, check out **The gist of MobX** page, or take the **10 minute interactive introduction to MobX and React**. The philosophy and benefits of the mental model provided by MobX are also described in great detail in the blog posts **UI as an afterthought** and **How to decouple state and UI (a.k.a. you don't need componentWillMount)**.

Further resources

- The MobX cheat sheet (£5) is both useful and sponsors the project
- 10 minute interactive introduction to MobX and React
- Egghead.io course, based on MobX 3
- The MobX awesome list – a long list of MobX resources and example projects

The MobX book



The **MobX Quick Start Guide** ($24.99) by **Pavan Podila** and **Michel Weststrate** is available as an **ebook**, **paperback**, and on the **O'Reilly platform** (see **preview**).

Videos

- Introduction to MobX & React in 2020 by Leigh Halliday, *17 min*.
- ReactNext 2016: Real World MobX by Michel Weststrate, *40 min*, slides.
- CityJS 2020: MobX, from mutable to immutable, to observable data by Michel Weststrate, *30 min*.
- OpenSourceNorth: Practical React with MobX (ES5) by Matt Ruby, *42 min*.
- HolyJS 2019: MobX and the unique symbiosis of predictability and speed by Michel Weststrate, *59 min*.
- React Amsterdam 2016: State Management Is Easy by Michel Weststrate, *20 min*, slides.
- {🚀} React Live 2019: Reinventing MobX by Max Gallo, *27 min*.

Credits

MobX is inspired by reactive programming principles, which are for example used in spreadsheets. It is inspired by model–view–viewmodel frameworks like **MeteorJS's Tracker**, **Knockout** and **Vue.js**, but MobX brings *transparent functional reactive programming* (TFRP, a concept which is further explained in the **MobX book**) to the next level and provides a standalone implementation. It implements TFRP in a glitch-free, synchronous, predictable and efficient manner.

A ton of credit goes to **Mendix**, for providing the flexibility and support to maintain MobX and the chance to prove the philosophy of MobX in a real, complex, performance critical applications.

Keywords

mobx **mobservable** **observable** **react-component** **react** **reactjs** **reactive** **model** **frp** **functional-reactive-programming** **state management** **data flow**

Install

```
npm i mobx
```

Repository

github.com/mobxjs/mobx

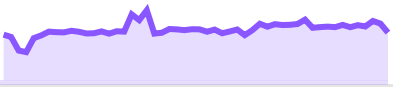
Homepage

mobx.js.org/

Fund this package

Weekly Downloads

851,714



Version

6.7.0

License

MIT

Unpacked Size

3.99 MB

Total Files

130

Issues

19

Pull Requests

8

Last publish

22 days ago

Collaborators






Try on RunKit

Report malware

Support

Help

Advisories

Status

Contact npm

Company

About

Blog

Press

Terms & Policies

Policies

Terms of Use

Code of Conduct

Privacy