

Immer: Immutability made easy

Michel Weststrate - @mweststrate



Example

```
const ADD_TAG = "ADD_TAG"

const state = {
  "4a12d": {
    title: "write proper reducer",
    done: false,
    comments: ["what a mess!", "can be better?"],
    tags: []
  }
}

const reducer = /* what ? */

const nextState = reducer(state, {
  type: ADD_TAG,
  id: "4a12d",
  tag: "needs review"
})

console.log(state["4a12d"].tags.includes("needs review")) // false
console.log(nextState["4a12d"].tags.includes("needs review")) // true
console.log(nextState["4a12d"].comments === state["4a12d"].comments)
```

```
const reducer = (state, action) => {  
  switch (action.type) {  
    case ADD_TAG:  
      return {  
        ...state,  
        [action.id]: {  
          ...state[action.id],  
          tags: [...state[action.id].tags, action.tag]  
        }  
      }  
    default:  
      return state  
  }  
}
```

Oh dear 😱

Spread-hell 🙄

```
// add a tag to a todo
state[action.id].tags.push(action.tag)
```

-- VS --

```
// add a tag to a todo
{
  ...state,
  [action.id]: {
    ...state[action.id],
    tags: [
      ...state[action.id].tags,
      action.tag
    ]
  }
}
```




Come to our gym!



Before

After

Use more libraries! 🤔

```
import update from "immutability-helper"

const reducer = (state, action) => {
  switch (action.type) {
    case ADD_TAG:
      return update(state, {
        [action.id]: {
          tags: {
            $push: [action.tag]
          }
        }
      })
    default:
      return state
  }
}
```

“The syntax takes a little getting used to (though it's inspired by MongoDB's query language)”

A "must-have" for SQL lovers



Query languages for client side state

Making simple look hard

O RLY?

Mr. JavaBean

The worlds worst reducer 🏆

```
const reducer = (state, action) => {  
  switch (action.type) {  
    case ADD_TAG:  
      state[action.id].tags.push(action.tag)  
      break  
  }  
}
```

(what did you expect? I wrote MobX 🤪)

The Good Parts 🙌

Nice and short!

The Bad Parts 🤬



Destroys everything that is pure and beautiful in this world



**Let's fix this with 3
characters 🦾**

```
import p from "immer"

const reducer = p((state, action) => {
  switch (action.type) {
    case ADD_TAG:
      state[action.id].tags.push(action.tag)
      break
  }
})

// all tests pass!
```

Immer 🧐



Immutability in JavaScript made easy

Keeps previous state, produces new next state





Current



immer



Draft



immer



M



Your edits here.

Immer

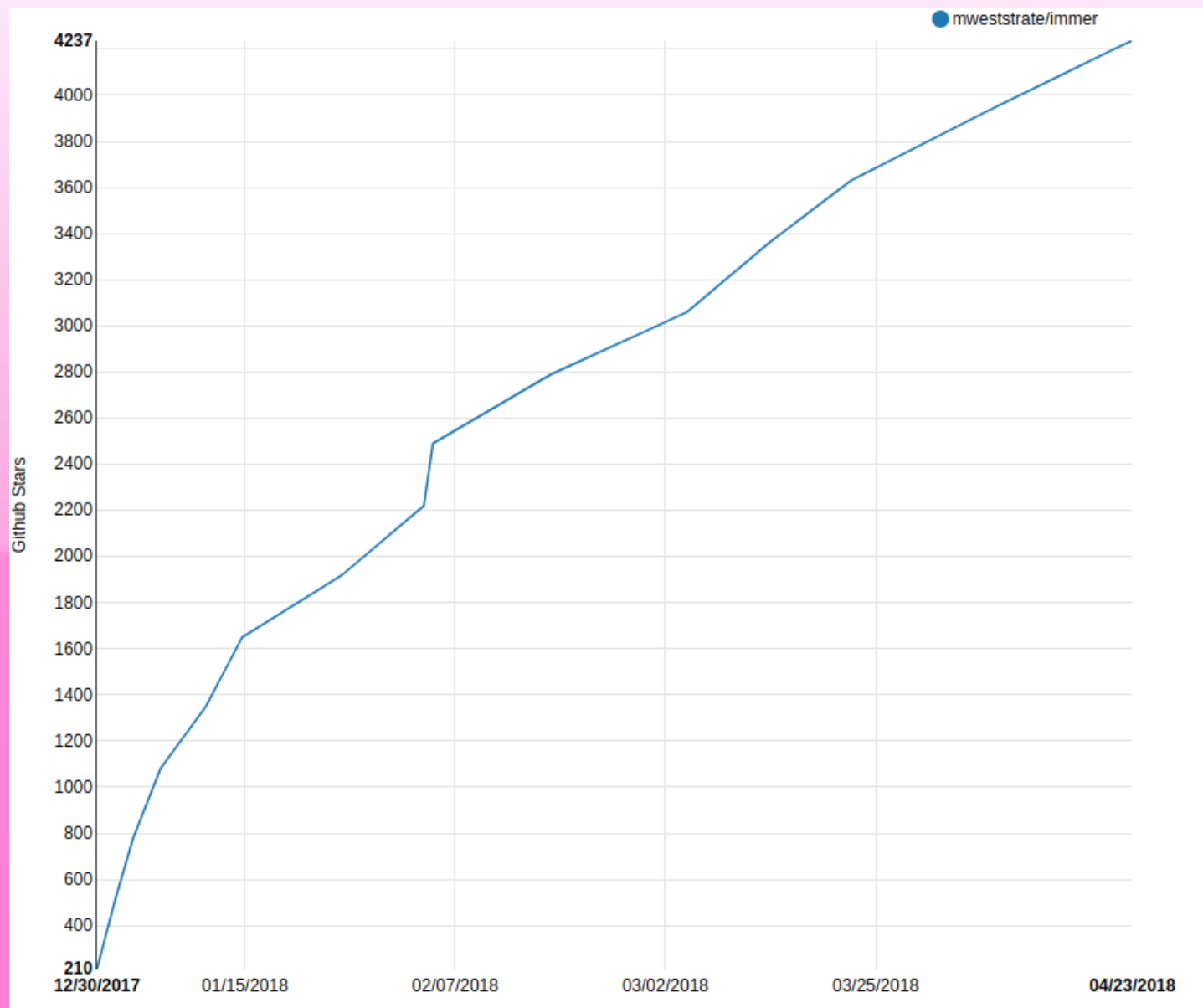
- Secret ingredient: copy-on-write collections
- Developed at Mendix as research project


Immer

1. Straightforward (deep) updates
2. Plain old JavaScript structures
3. Structural sharing
4. Strongly typed
5. 1 function API
6. Small (2KB)
7. Fast
8. Auto freezing
9. Runs on any ES5 environment

Immer

1. Reducers
2. setState
3. Any plain object data tree
4. Used in quite some libraries



 **Kapil Gorge** @kapilgorve · Jan 11

One of my biggest hurdle learning #Redux was writing complex #reducers. @mweststrate made it child's play with Immer. Most Imp - Its all #javascript api. Strongly recommend to #newbie. #reactjs #functionalprogramming @eggheadio egghead.io/lessons/redux-...

1 15 35

↳ Roy Riojas and 2 others Retweeted

 **Chema Roldán** @chemitaxis · Mar 26

I think #immer created by @mweststrate is one of the new best thing that javascript ecosystem has... thanks mate

 **Sebastian Markbåge** @sebmarkbage · Mar 4

Immer looks like a good project. I'm pretty convinced that temporal mutability like this is a good model. As long as it is wrapped in a scope like a reducer, render function, etc.



mweststrate/immer
immer - Create the next immutable state by mutating the current one
github.com

7 57 273

Questions? Find me!