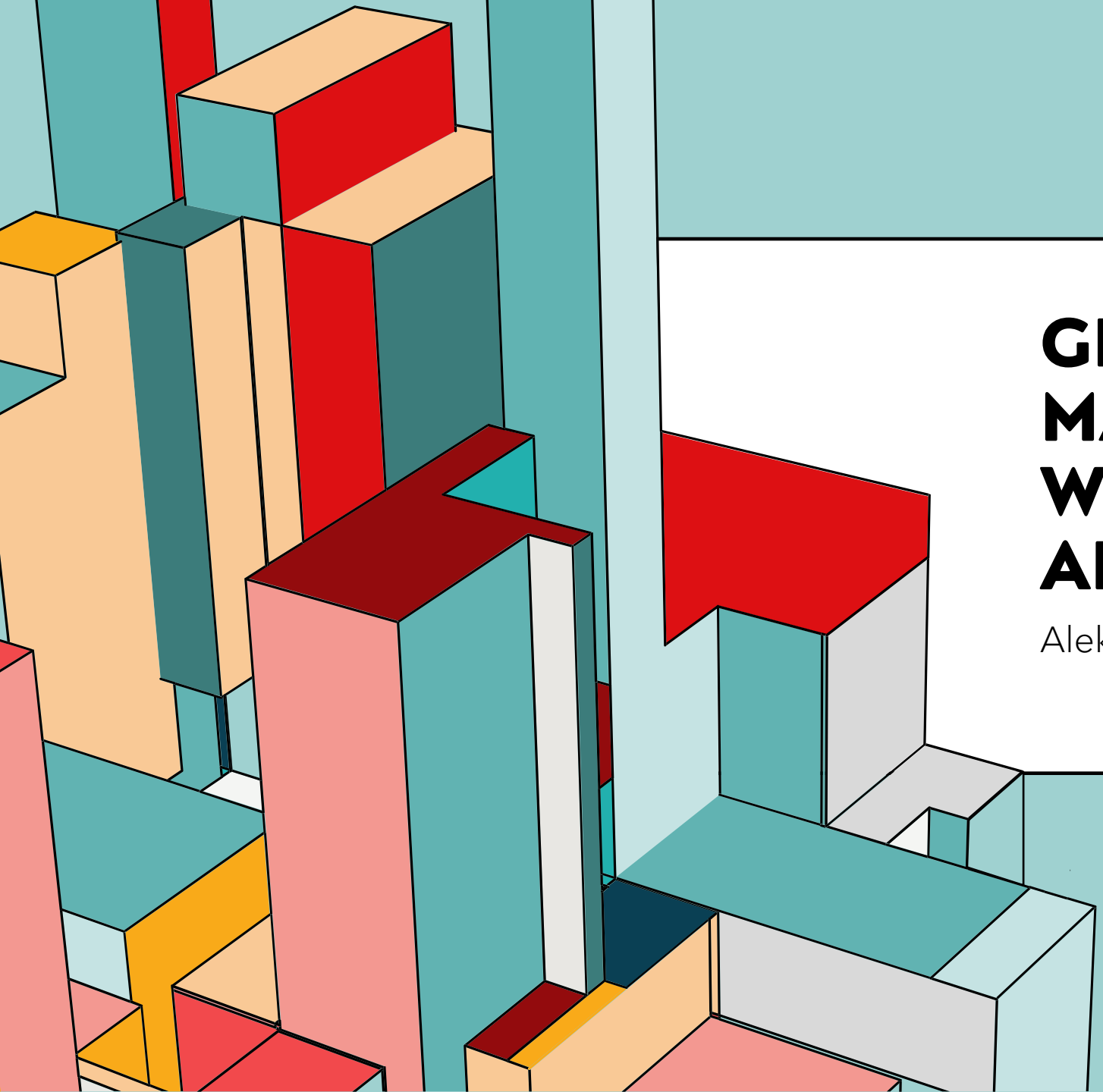


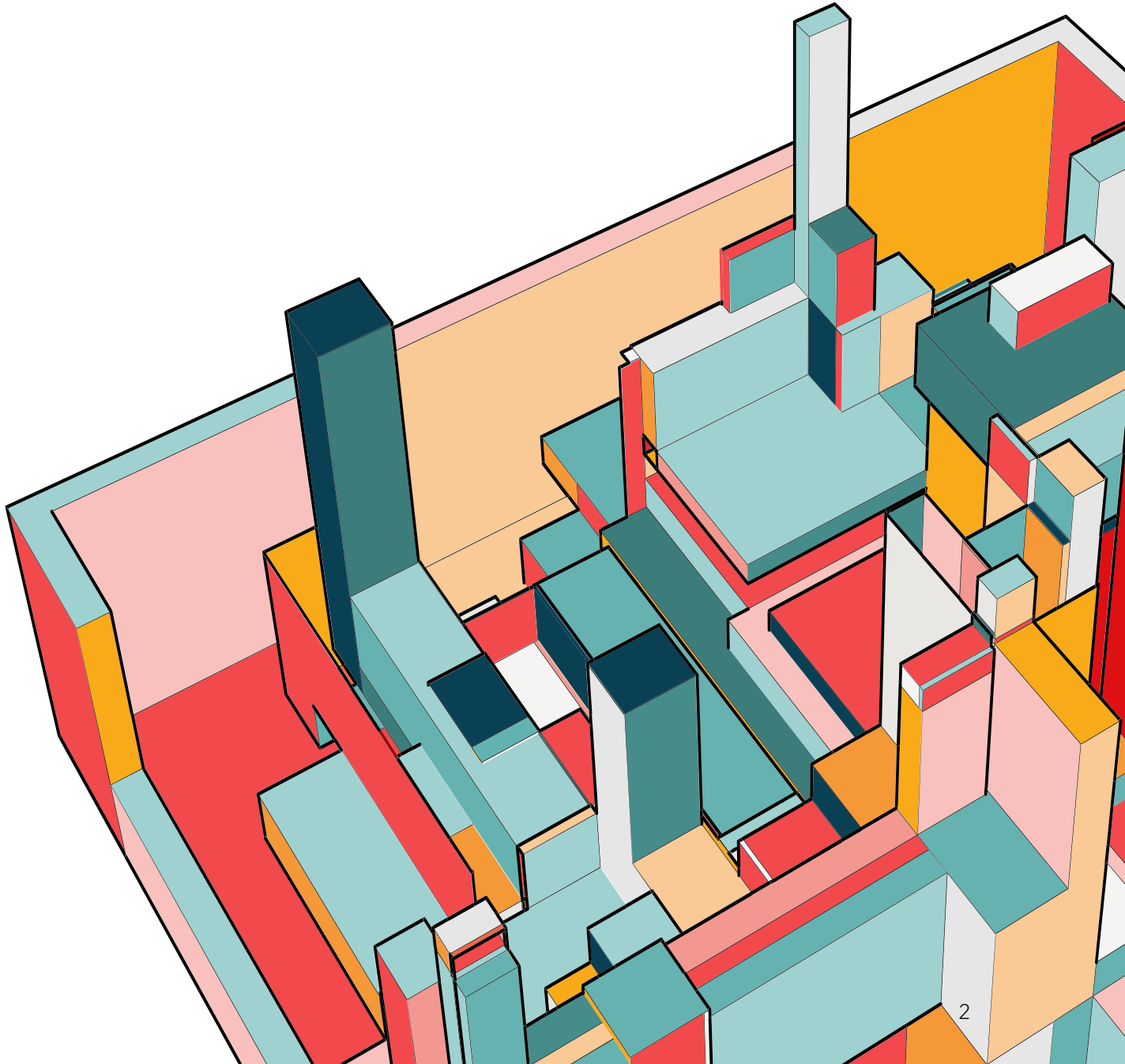
GLOBAL STATE MANAGEMENT WITH CONTEXT API IN REACT

Aleksi Alhola

GLOBAL STATE MANAGEMENT

React state management is implemented with useState and useReducer hooks at component level. However, if these states need to be accessed in child components, they need to be passed as props.

Most of the times React's global state is instead implemented via 3rd party libraries, like Redux, instead. However, React's own API already has all the tools required for global state management.



PROBLEMS WITH REDUX

3rd PARTY LIBRARIES

Although Redux is highly maintained and supported, there is always a risk that it will enter EOL state and is no longer supported.

NPM CONFLICTS

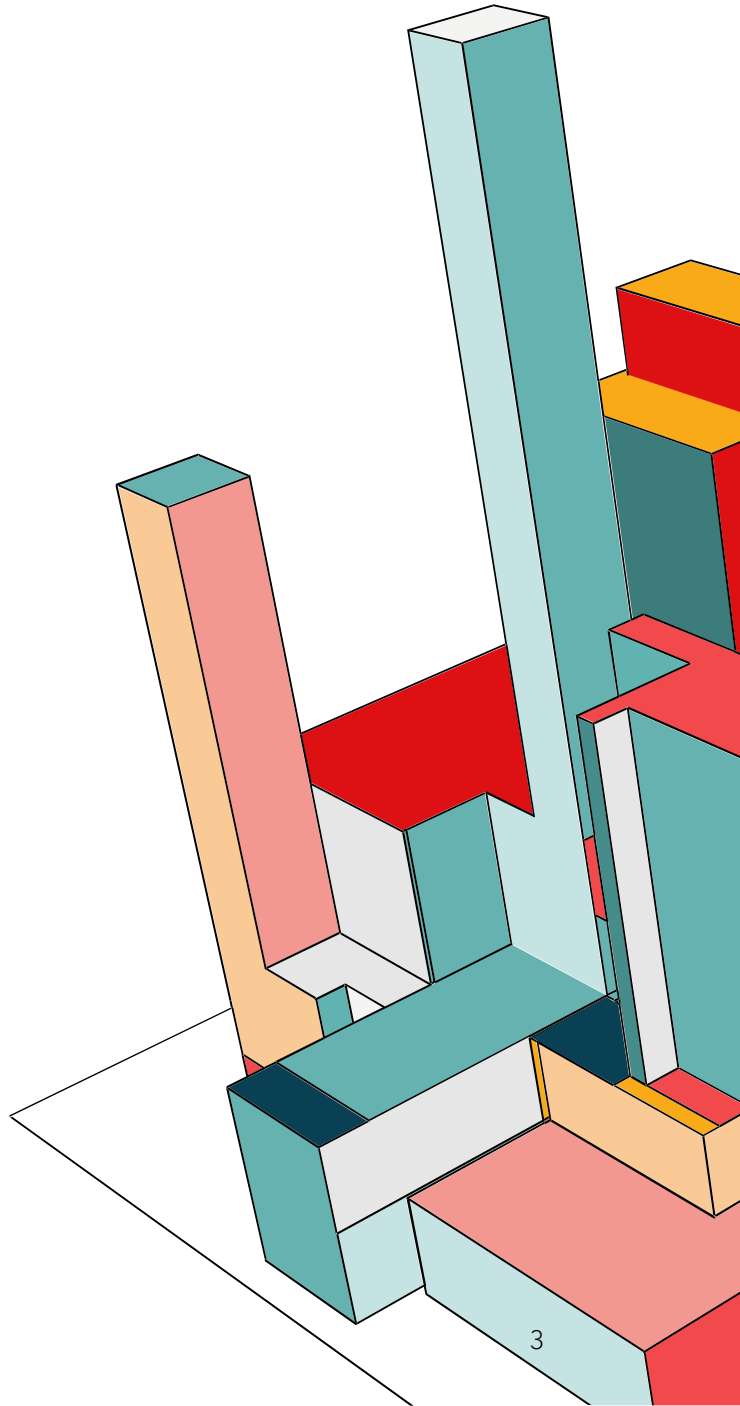
3rd party libraries have the hassle for creating conflicts when new releases are published.

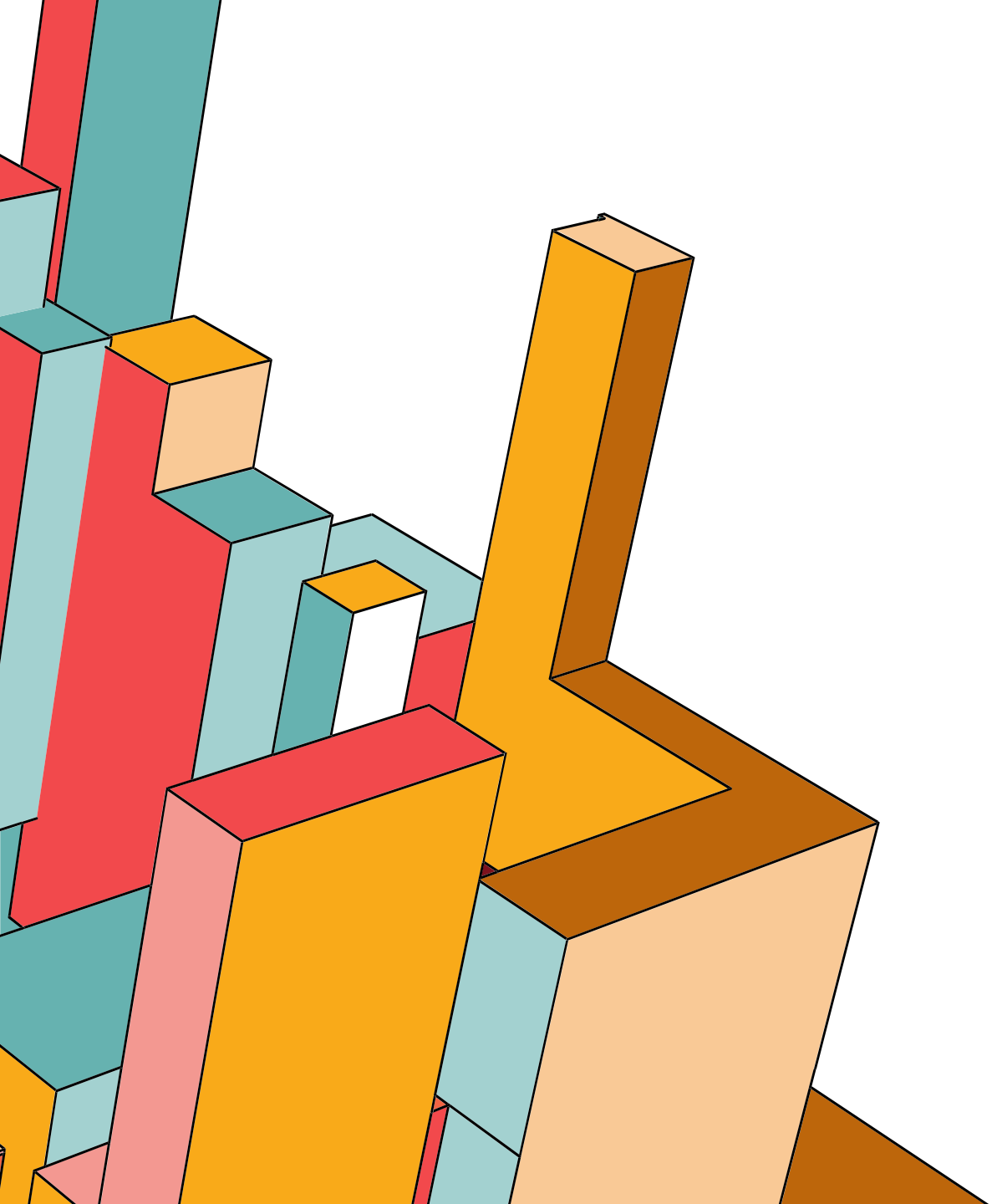
BLOAT

3rd party libraries might use some external libraries that you don't need, increasing the directory size. Also, if you deploy to Docker, the installation time will take longer.

READABILITY

Chaining functions with sagas and using the effect functions can lead to spaghetti code.





BENEFITS OF REDUX

LESS BOILERPLATE CODE

Usage of Redux reduces the amount of source code if used correctly.

SECURITY

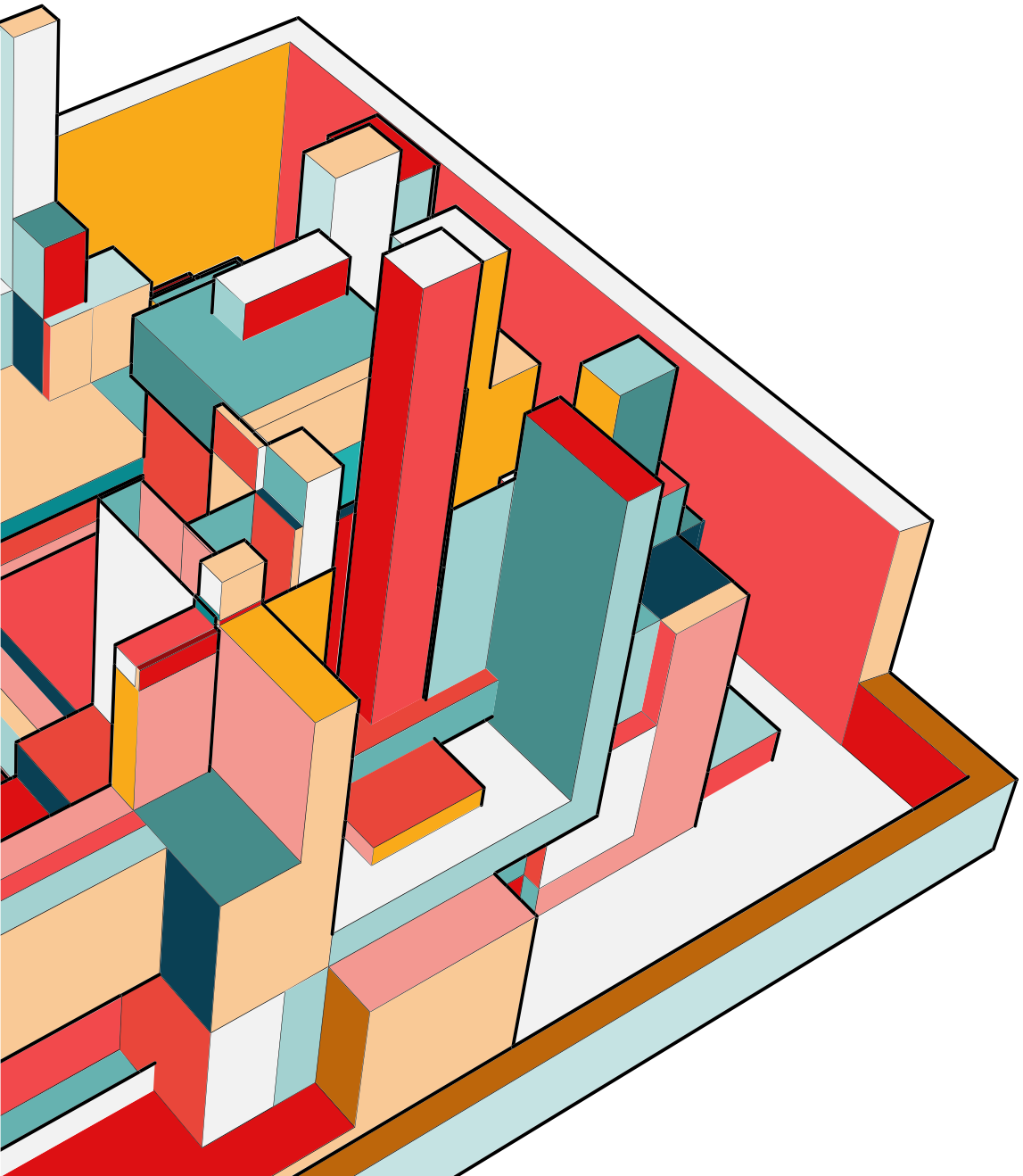
Redux state is immutable out of the box and can also be encrypted for extra security.

EASY TO USE

With Redux toolkit and sagas almost all of the use-cases are already covered.

Redux developer tools

Following the state mutations with the devtools is very easy.



WHAT IS CONTEXT API?

Context is used to pass values through the component tree without passing it as props.

This is made possible with createContext and useContext hooks, which create and subsribe to the Context respectively. Any mutation to the Context is automatically received by the components subscribed to it.

Context does not handle state management by itself, it only does dependency injection by default.

https://reactjs.org/docs/context.html

BENEFITS OF CONTEXT API

ALREADY INCLUDED

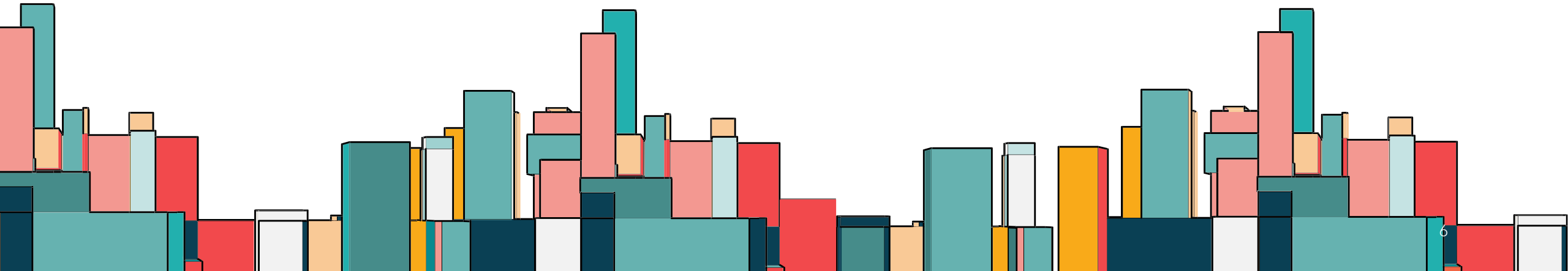
Context ships together with React, so no need to install additional libraries.

MORE CONTROL

Gives the developer "full" control over the state management.

MULTIPLE STORES

Can create stores even at component level, e.g if each page on the website have their own state that doesn't need to be accessed elsewhere. (Possible on Redux too, it's just not how it is intended to be used)



PROBLEMS WITH CONTEXT API

LOTS OF BOILERPLATE

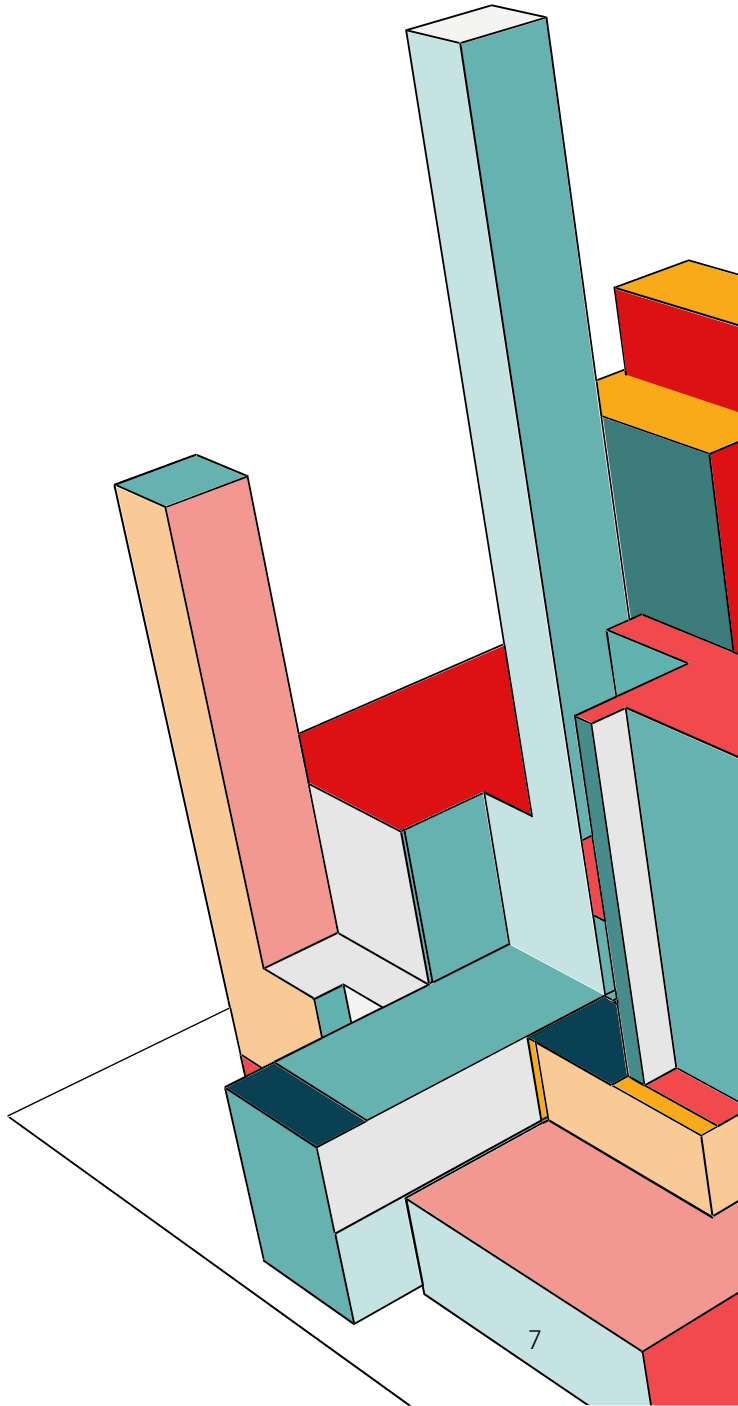
This depends of the implementation and scale, but generally requires a lot of boilerplate code to be used efficiently (especially for complex states).

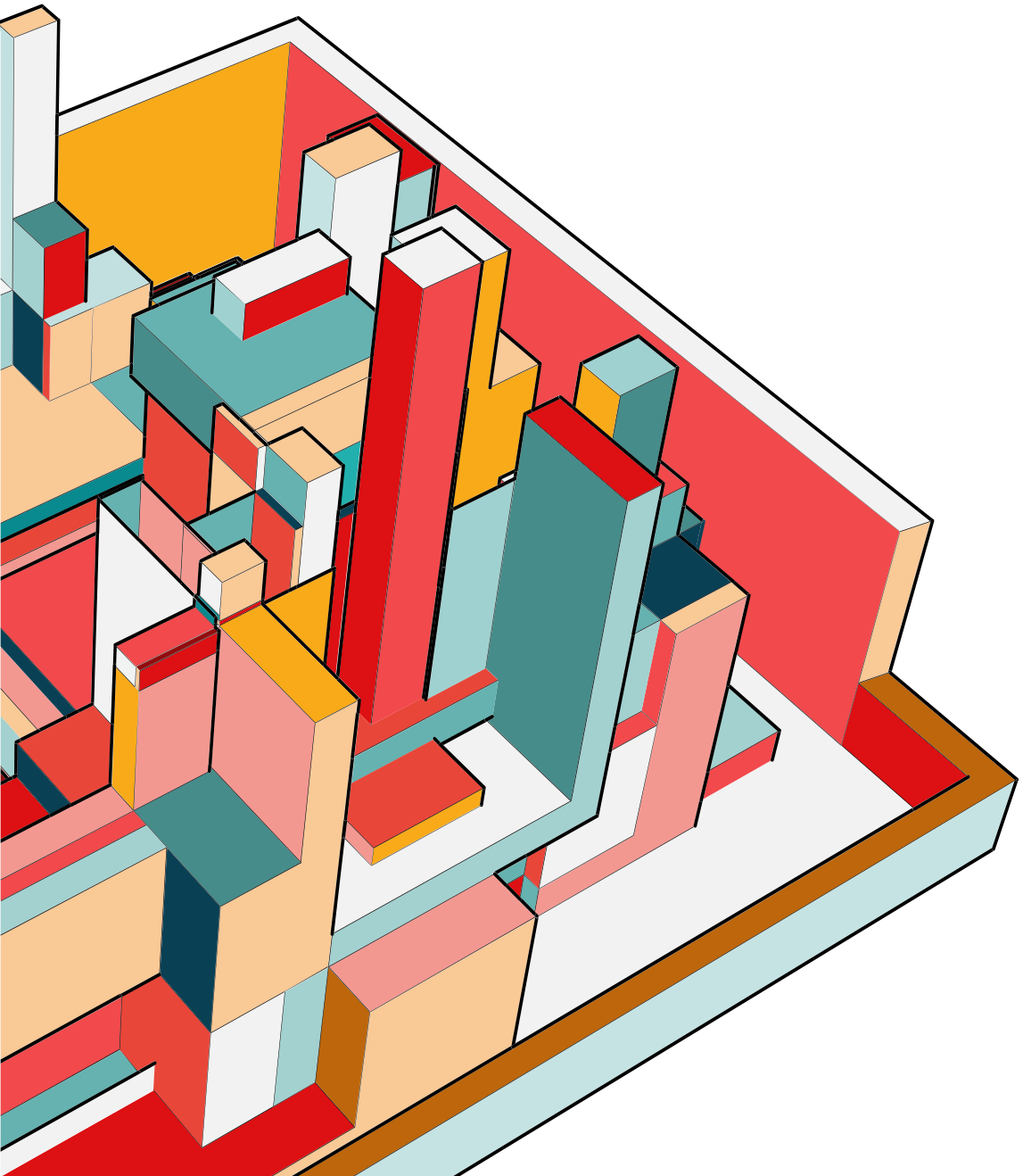
LACK OF SECURITY

If the state objects are not frozen, the state can be mutated directly. Additionally, if an object is used as state and the object is mutated, this does not trigger re-render (the state needs to be reassigned to trigger re-render).

UNNECESSARY RE-RENDERS

Because the change in Context triggers re-render whenever the state of it changes, every component subscribed to it re-renders when the state is updated (unless handled correctly with separate logic).





HOW TO USE CONTEXT EFFECTIVELY?

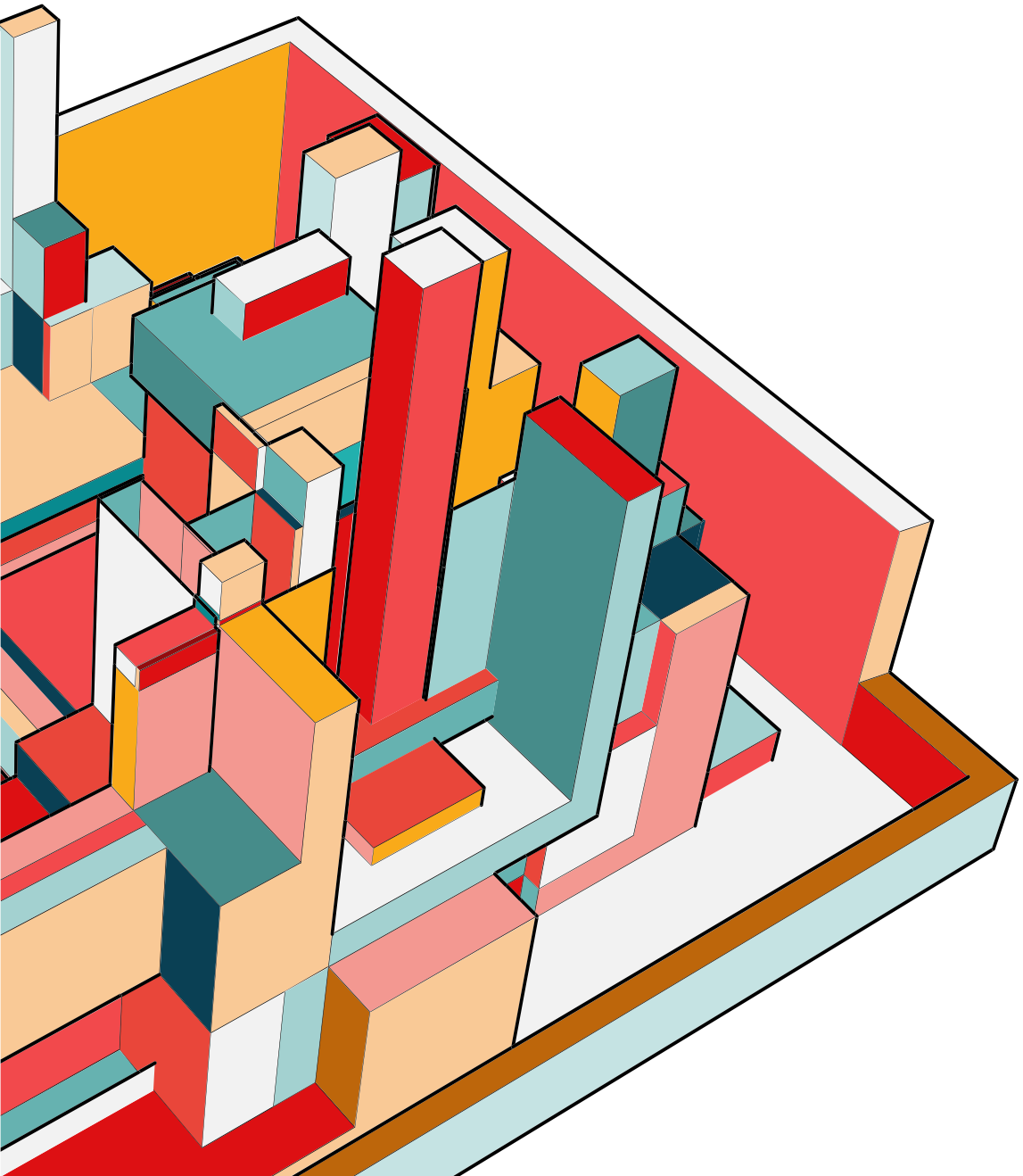
By its own Context is not that amazing. It has its use when used instead of useState in deeply nested components, but over all most of these cases are already handled within the global state.

What makes it powerful is its use together with other React hooks (useReducer, useMemo, useCallback, useSyncExternalStore) and possibility of creating your own custom hooks.

This provides a way for the developer to create the best use-case for the state logic for their apps needs.

TASKS

https://github.com/KnowitJSTS Guild/context-api-with-hooks



FINAL WORDS

The provided examples for the usage of Context were wholly created by myself to prove that React does not really need any additional libraries for state management.

I have not yet found myself in a situation where proper usage of Context API would not have been sufficient for the application.

Both Redux and Context have their good and bad sides, and I acknowledge that Redux is very powerful tool for larger and more complicated applications.

My suggestion would be to try out Context for global state management, but Redux is sort of industrial standard so perhaps it's best to stick with it for now.