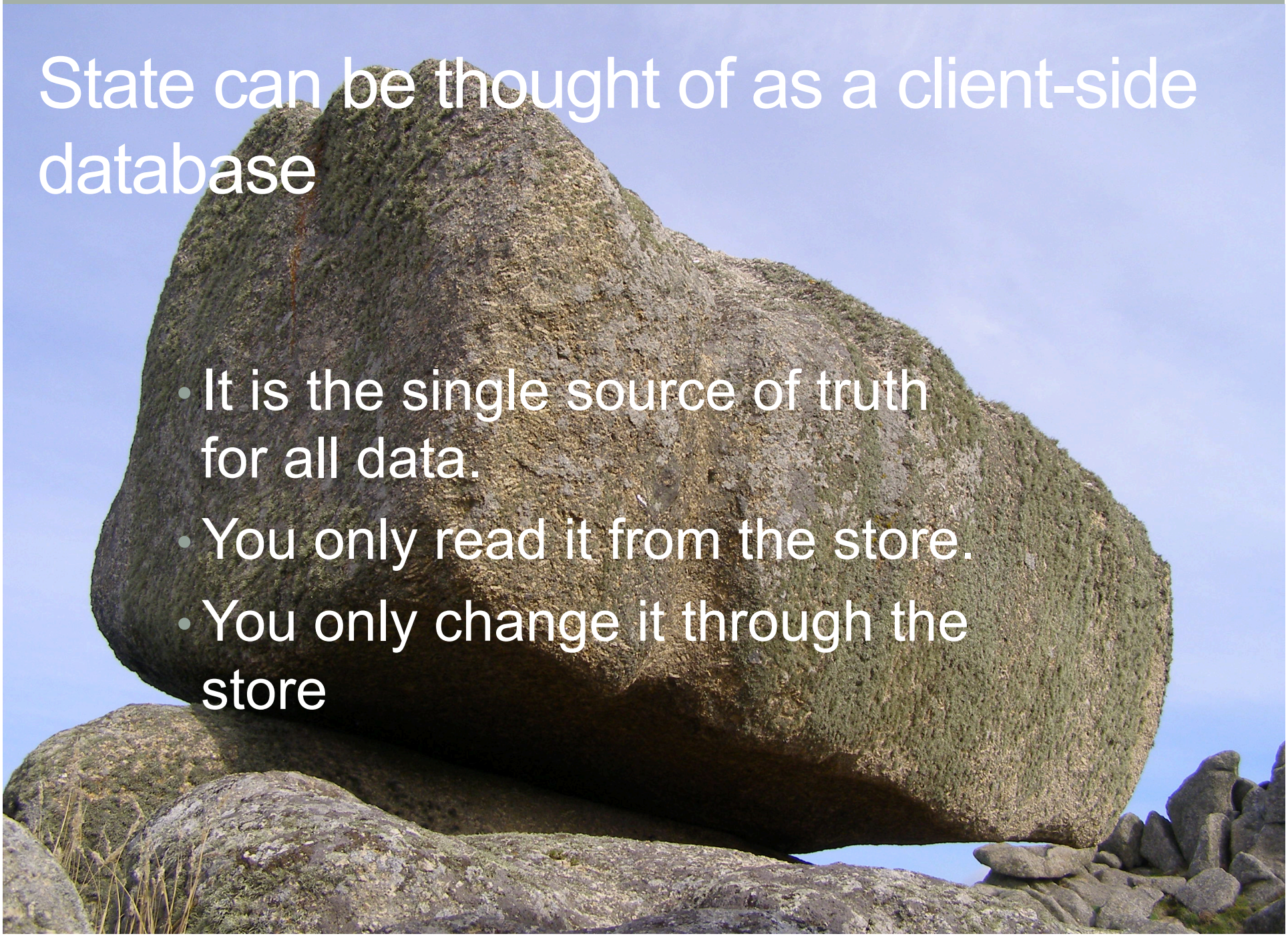
State and Subscriptions

State can be thought of as a client-side database



- It is the single source of truth for all data.
- You only read it from the store.
- You only change it through the store

We should set the initial state when we're creating the store.

```
const initialState = {
  first: "Jo",
  last: "Bennett",
  city: "Tallahassee",
  state: "Florida"
}
export const store =
  createStore(reducer, initialState);
```

- This method is self-documenting.

Hands-on state

What should go in state and what should not?

- Does the data change?
- Should the data be the same between pages/views?
- Should the data be the same when you refresh this page/view?
- If all are yes, then it goes in state. If not, no.

Subscriptions

When state changes, we should redraw the UI

- Different data => different display

- Should we call a redraw() UI function?
- What if we have to change 2 things upon a data change?
- 20?
- 200?
- Do we really want to keep track of all of that?

We can subscribe to every state change

- We know we'll always dispatch an action to change state
- Redux allows us to register X functions to run every time state is changed through a dispatch:

- In the UI:

```
store.subscribe(redraw);
function redraw() {
  // Logic to re-render the UI goes here
}
```

Redux keeps track of all subscriptions no matter where they were registered

- So, you can subscribe in lots of places in the UI and Redux fires them all when the state changes.

- And it doesn't have to be just the UI. Register any function that needs to run when state changes.
  - Upload data to a server
  - Save to local storage
  - Push it through a socket to another client
  - Etc.

Hands-on subscriptions