Actions in Redux

Here, we will compare the withdrawal of money to a fundamental aspect of Redux: Actions. An action is the only way to change the state of the app.

If you're going to get any money from the bank, you're going to have to go in with some intent or action to withdraw money.

If you just walk into the bank and roam about, no one's going to just give you money. You may even end up been thrown out by the security. Sad stuff.

The same may be said for Redux.

Write as much code as you want, but if you want to update the state of your Redux application (like you do with setState in React,) you need to let Redux know about that with an ACTION.

In the same way you follow a due process to withdraw your own money from the bank, Redux also accounts for a due process to change/update the state of your application.

Now, this leads to the Redux principle #2.

> State is read-only: The only way to change the state is to emit an action, an object describing what happened.

What does that mean in plain language?

When you walk to the bank, you go there with a clear action. In this example, you want to withdraw some money.

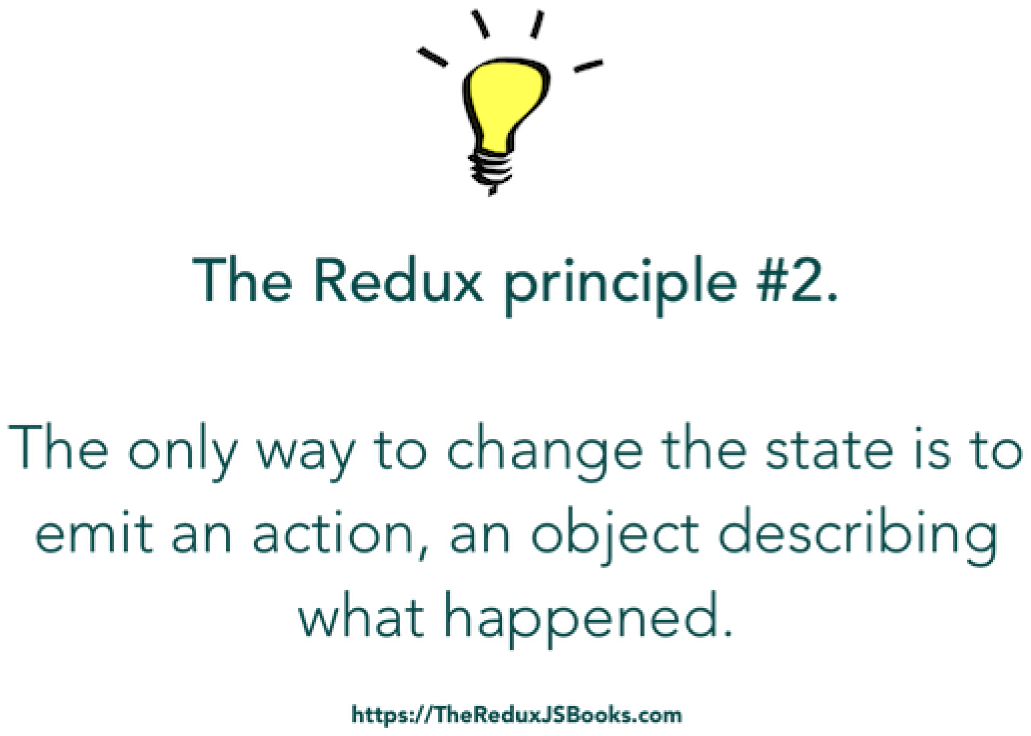
If we chose to represent that process in a simple Redux application, your action to the bank may be represented by an object.

One that looks like this:

```
{
    type: “WITHDRAW_MONEY”,
    amount: “$10,000”
}
```

In the context of a Redux application, this object is called an **action**! It always has a type field that describes the action you want to perform. In this case, **WITHDRAW_MONEY**

Whenever you need to change/update the state of your Redux application, you need to dispatch an action.


The Redux principle #2.

The only way to change the state is to emit an action, an object describing what happened.

https://TheReduxJSBooks.com

Don’t stress over how to do this yet. I’m only laying the foundations here. We’ll sure delve into lots of examples soon.