What is an action?

Let's dive deeper into the building blocks of Redux.

Actions are JavaScript objects as you can see in the following example:

```
{
   type: LOGIN_USER,
   payload: {
     username: 'sebastian',
      password: '123456'
      }
}
```

Here the action object has two properties:

- *type:* a constant to identify the type of action.
- *payload:* the object which is assigned to this property contains the data which are sent to the store.

Action objects are created by using functions. These functions are called action creators

```
function authUser(data) {
 return {
   type: LOGIN_USER,
   payload: data,
 };
}
```

Here you can see that the only purpose of an action creator function is to return the action object as described.

Calling actions in the application is easy by using the dispatch method:

```
dispatch(authUser(data));
```

What is a Reducer?

Reducers are the most important building block and it's important to understand the concept. Reducers are pure JavaScript functions that take the current application state and an action object and return a new application state:

```
const reducer = (state, action) => {
switch (action.type) {
 case type1:
   return; // the new state
case type2:
   return; // the new state
 default:
   return state;
}


}
```

The important thing to notice here is that the state is not changed directly. Instead a new state object (based on the old state) is created and the update is done to the new state.

What is the Redux Store?

The store is the central objects that holds the state of the application. The store is created by using the `createStore` method from the Redux library:

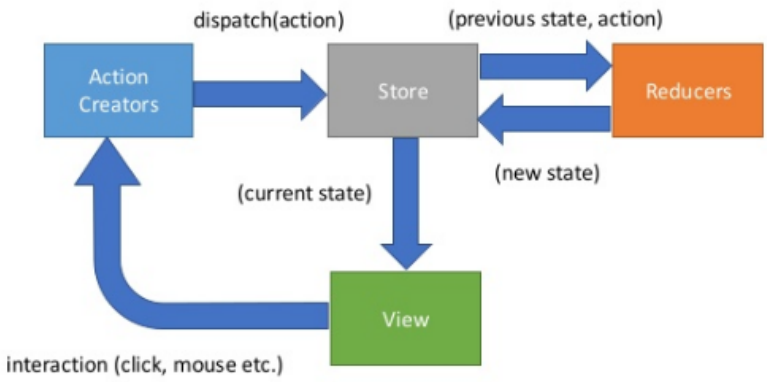
```
import { createStore } from 'redux';
let store = createStore(myReducer);
```

You need to pass in the reducer function as a parameter. Now you're ready to dispatch a action to the store which is handled by the reducer:

Redux data flow

The image below describes the Redux data flow and how every part gets triggered:

Redux Data Flow



Previous
next