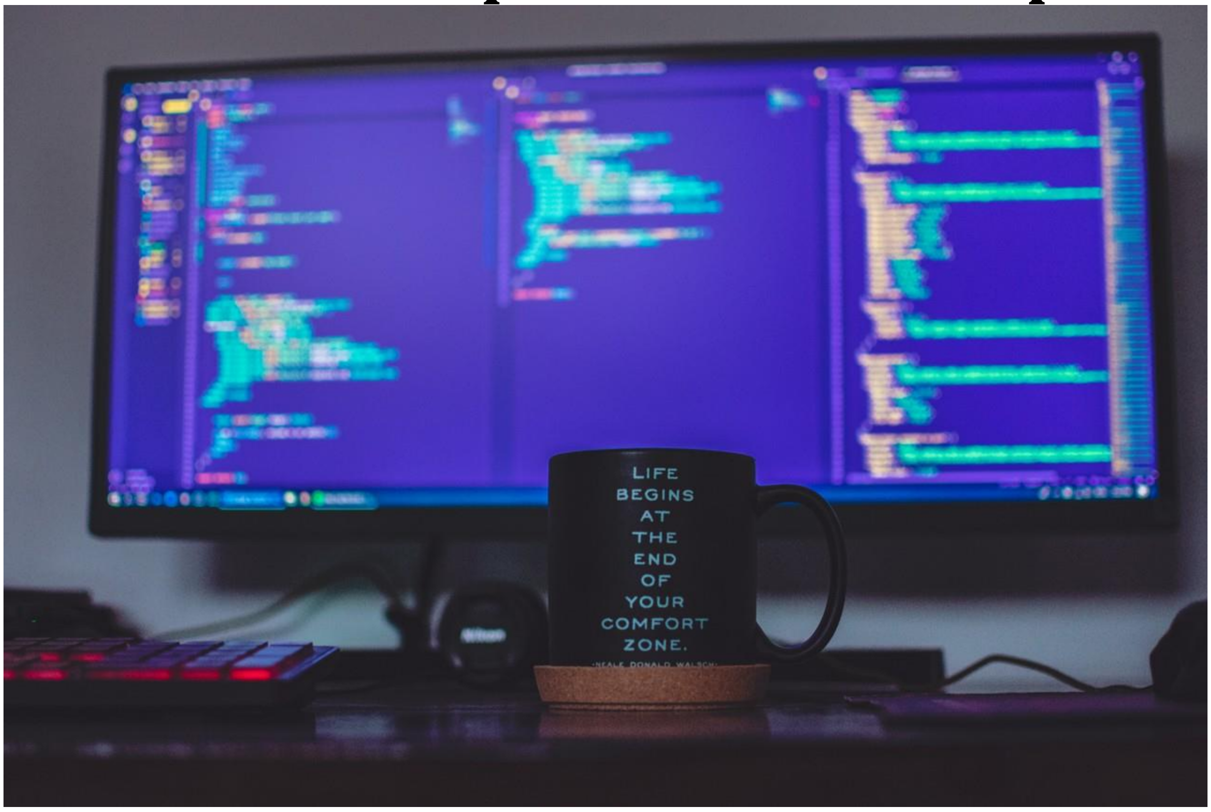
Redux Thunk Explained with Examples



Redux Thunk is middleware that allows you to return functions, rather than just actions, within Redux. This allows for delayed actions, including working with promises.

One of the main use cases for this middleware is for handling actions that might not be synchronous, for example, using axios to send a GET request. Redux Thunk allows us to dispatch those actions asynchronously and resolve each promise that gets returned.

Installation and Setup

Redux Thunk can be installed by running `npm install redux-thunk --save` or `yarn add redux-thunk` in the command line. Because it is a Redux tool, you will also need to have Redux set up. Once installed, it is enabled using `applyMiddleware()`:

```
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
import rootReducer from './reducers/index';

const store = createStore(
  rootReducer,
  applyMiddleware(thunk)
);
```

How to Use Redux Thunk

Once Redux Thunk has been installed and included in your project with `applyMiddleware(thunk)`, you can start dispatching actions asynchronously.
For example, here's a simple increment counter:

```
const INCREMENT_COUNTER = 'INCREMENT_COUNTER';

function increment() {
  return {
    type: INCREMENT_COUNTER
  };
}

function incrementAsync() {
  return dispatch => {
    setTimeout(() => {
      // You can invoke sync or async actions with `dispatch`
      dispatch(increment());
    }, 1000);
  };
}
```

And here's how set up success and failure actions after polling an API:

```
const GET_CURRENT_USER = 'GET_CURRENT_USER';
const GET_CURRENT_USER_SUCCESS = 'GET_CURRENT_USER_SUCCESS';
const GET_CURRENT_USER_FAILURE = 'GET_CURRENT_USER_FAILURE';

const getUser = () => {
  return (dispatch) => {          //nameless functions
    // Initial action dispatched
    dispatch({ type: GET_CURRENT_USER });
    // Return promise with success and failure actions
    return axios.get('/api/auth/user').then(
      user => dispatch({ type: GET_CURRENT_USER_SUCCESS, user }),
      err => dispatch({ type: GET_CURRENT_USER_FAILURE, err })
    );
  };
};
```