Redux Saga is a companion library of Redux which helps you manage asynchronous flow of the application

Redux-saga is basically working as frontcontroller in the application.

It uses ES6 feature Generators

What is Function and Generators in js

Functions:

Function is way to completion and there is no way to pause it between execution.

Ex.

Function sayHi(){

//Statements

Return ‘Hi’;

}

* So function can’t stop between statement until execution finish.
* Can’t be return more than one

Generator function

Function\* sayGenerator(){

Return ‘HI’;//

}

* **asterisk** (\*) symbol shows that function is a generator
* It can return multiple return statements
* Returned values will be as iterator

Function\* sayGenerator(){

Yield ‘Hello’;

//Statements

Return ‘HI’;//

}

Two main things are here…

1. Watcher Saga
2. Worker Saga

* Compose Method will combine multiple things to the store.

How to use the Redux-Saga

1. Create the sagamiddleware using createSagaMiddlewware .
2. Create the store and applymiddleware by passing the sagaMiddleware
3. Run the saga using sagamiddleware by passing rootSaga

rootSaga is handling all the activities .

Now apply some helper in rootSaga. Helper are effects like :takeEvery,take ,call , put.

These effects are like object ànd these object called in sagamiddleware and act accordingly.

-------

Now in write the watcher and worker saga in rootsaga

Watcher will listen the action and call the worker accordingly

Watcher saga will listen the action when action will dispatch

//Saga can we in control for any action

Suppose Login click on login many times

//Blocking and non-blocking calls