REDUX MIDDLEWARE

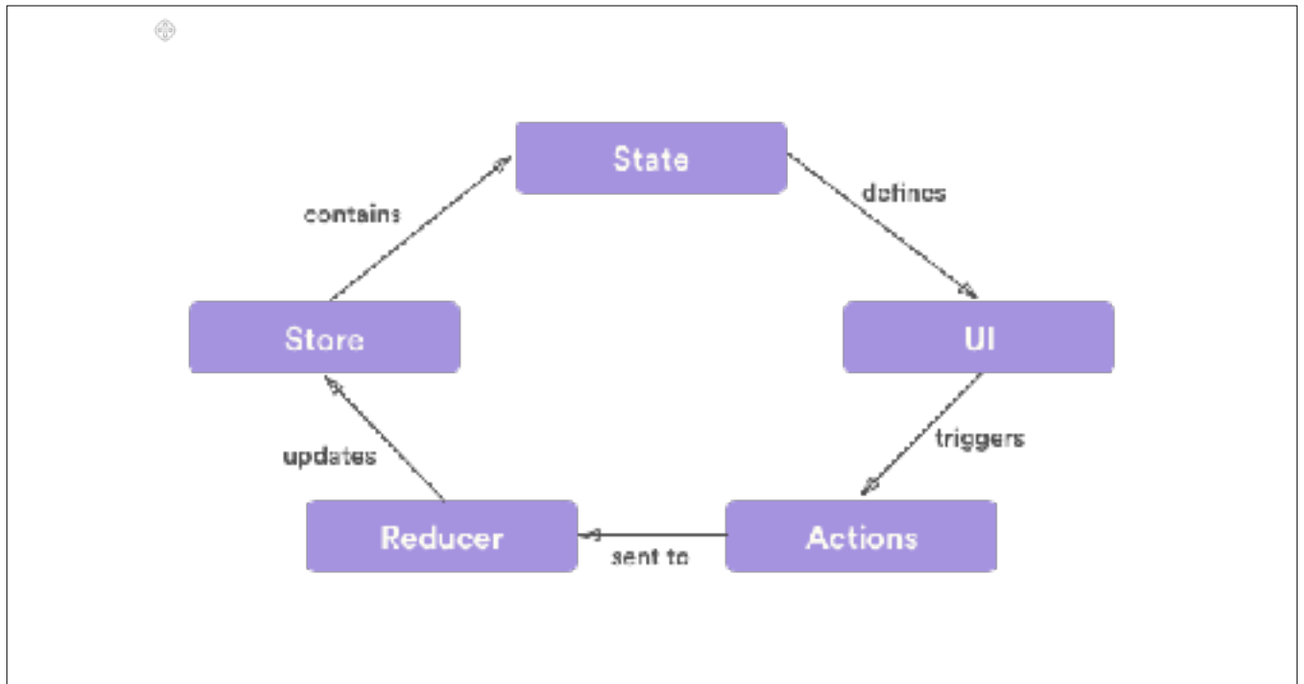
What is a middleware?

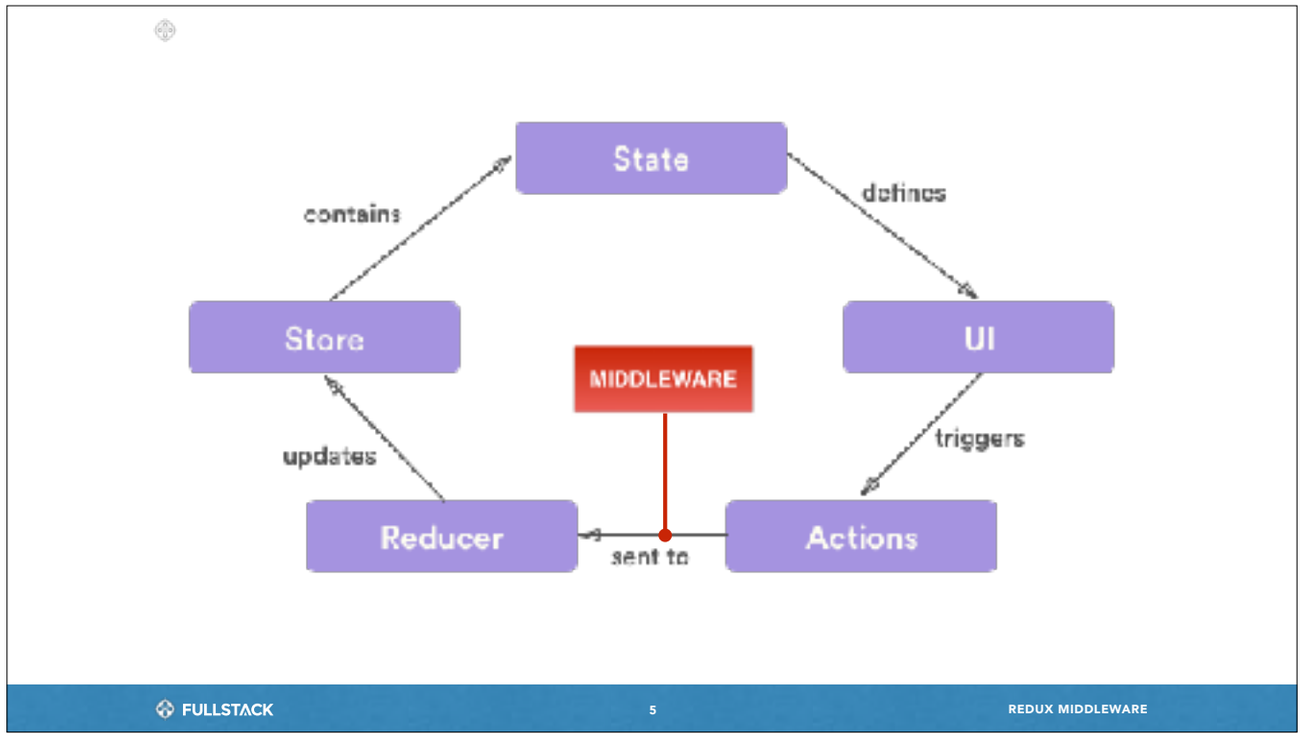
You've seen middleware in action in Express: code you can put between the framework receiving a request, and the framework generating a response. For example, Express middlewares may add logging, static file serving, compression, and more.
Redux middleware solves different problems than Express middleware, but in a conceptually similar way

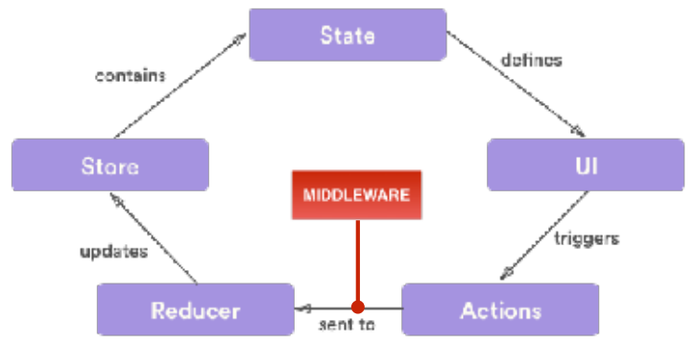
WHAT IS A MIDDLEWARE

- Provides an extension point.
- Middleware code runs between the action being dispatched and reaching the reducer
- Useful for logging, crash reporting, talking to an asynchronous API, routing, and more.

Middleware in redux provides a third-party extension point between dispatching an action, and the moment it reaches the reducer.


State
contains
defines
Store
UI
updates
triggers
Reducer
sent to
Actions

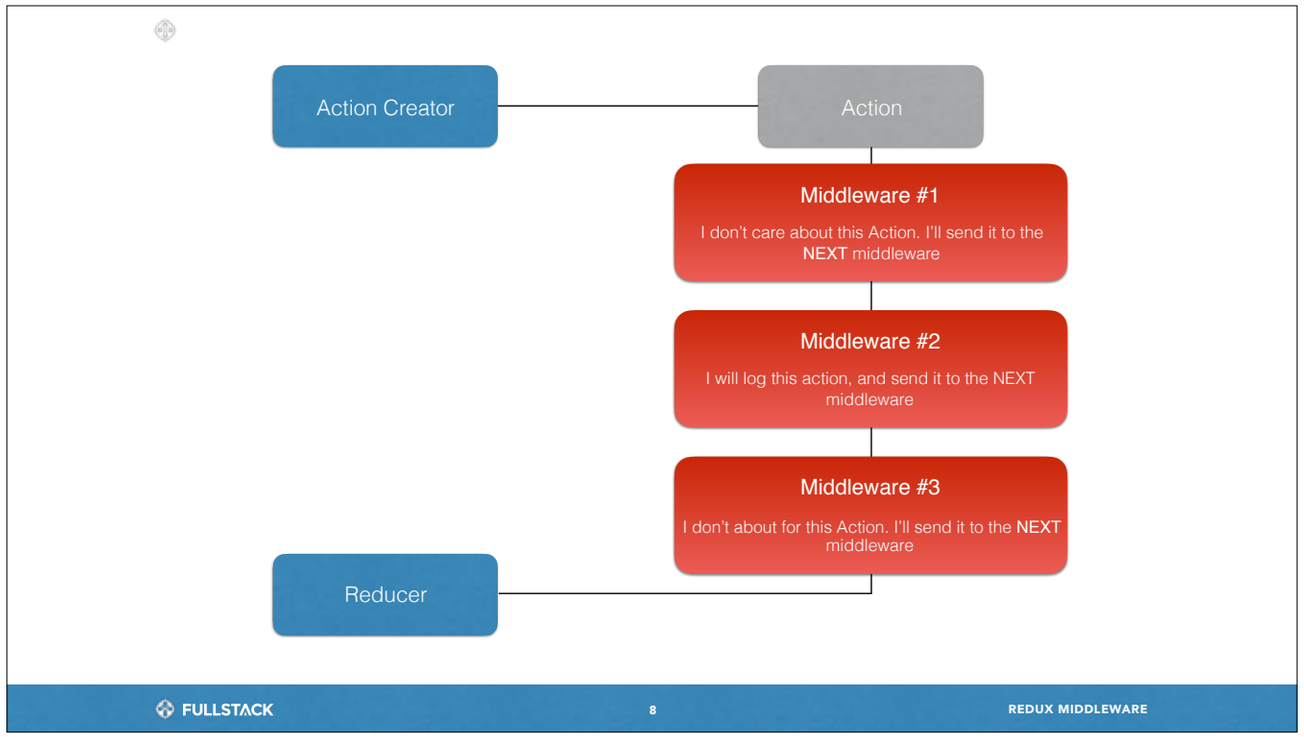

State
contains
defines
Store
MIDDLEWARE
UI
updates
triggers
Reducer
sent to
Actions



- **The Middleware receives the action and can do whatever it wants with it, including modifying or even stopping it.**

MIDDLEWARE BENEFITS

- Middlewares are composable
- Middlewares run independently


Action Creator
Action
Middleware #1
I don't care about this Action. I'll send it to the NEXT middleware
Middleware #2
I will log this action, and send it to the NEXT middleware
Middleware #3
I don't about for this Action. I'll send it to the NEXT middleware
Reducer

MIDDLEWARE FORMAT

- A middleware is a function that receives the store
- It MUST return a function with arg “next”, that itself:
- Must return a function with arg “action”

```
const someMiddleware = store => next => action => {
  // your code here
  return next(action);
}
```

APPLYING MIDDLEWARE

```
const store = createStore(reducer);
```

APPLYING MIDDLEWARE

```
const store = createStore(
  reducer,
  applyMiddleware(someMiddleware)
);
```