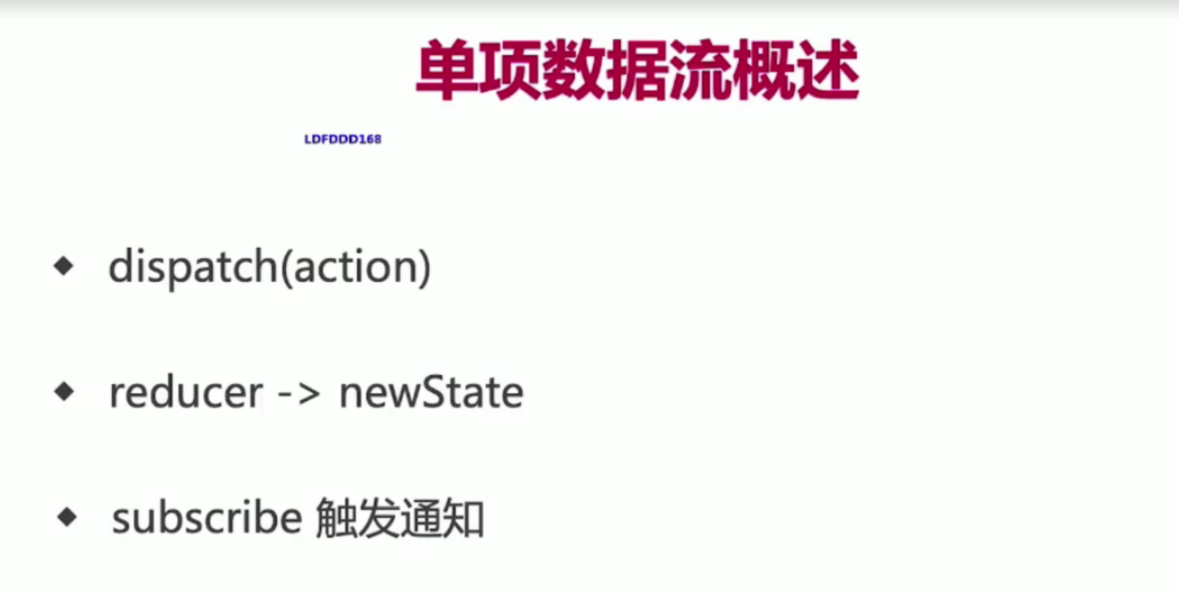
# redux

## 基本概念



单项数据流概述

LDFDDD168

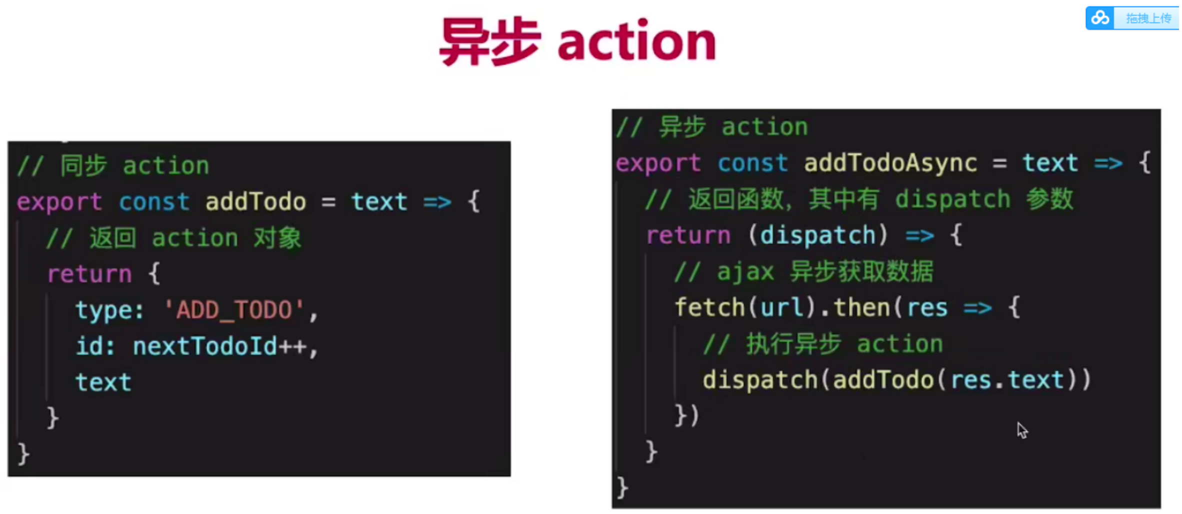
- dispatch(action)
- reducer -> newState
- subscribe 触发通知

## react-dedux

Provider

connect

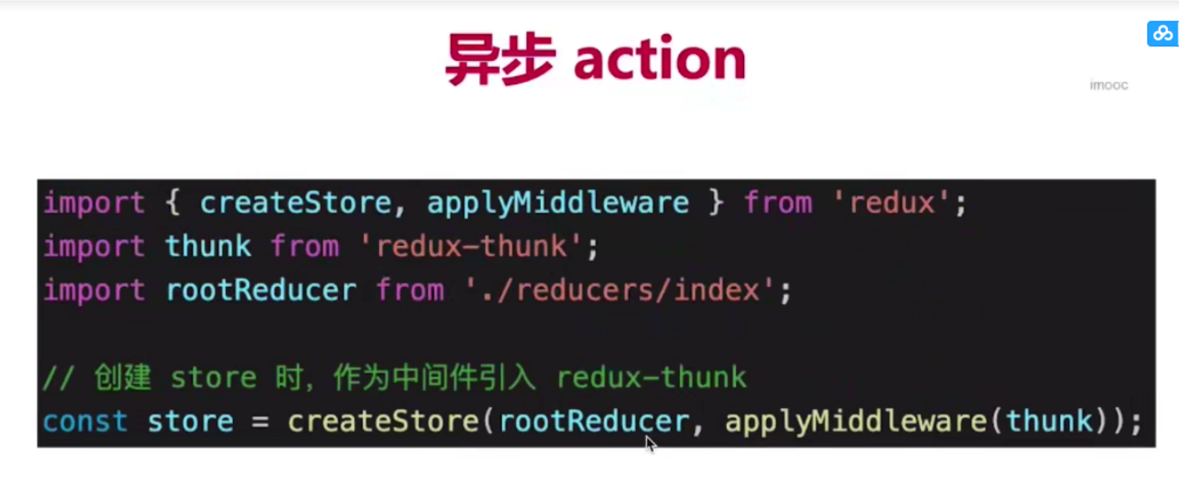
## 异步action



异步 action

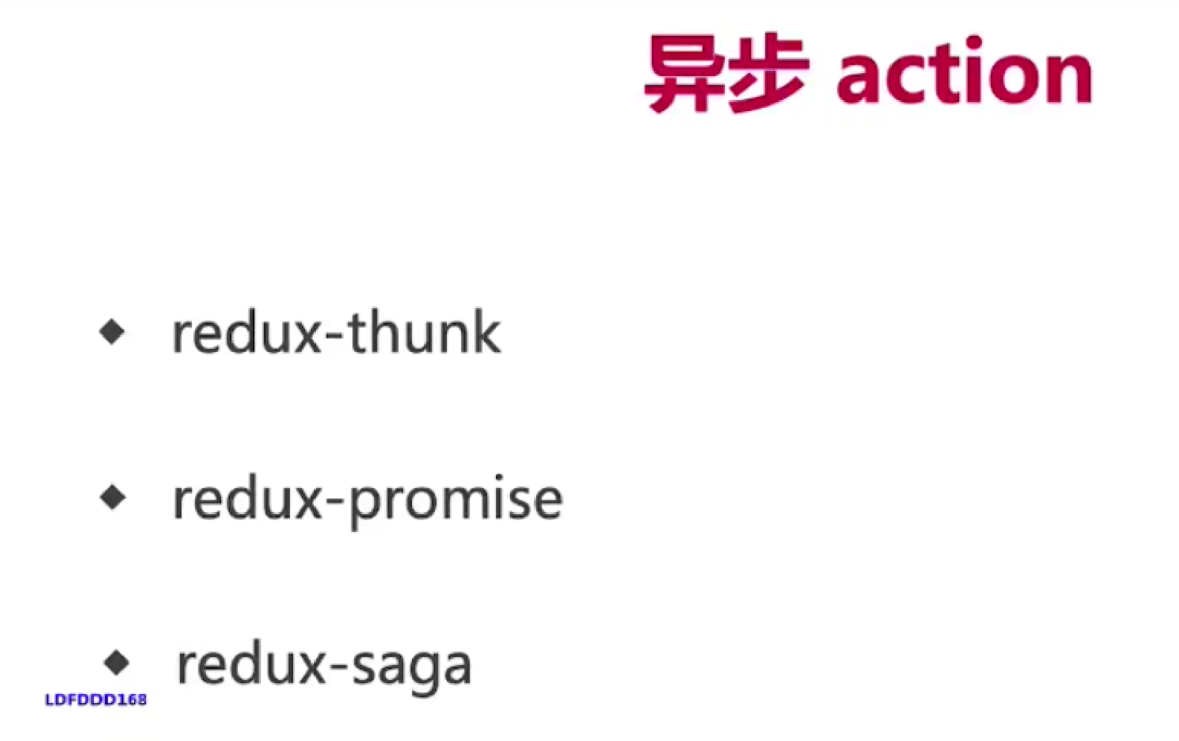
```
// 同步 action
export const addTodo = text => {
  // 返回 action 对象
  return {
    type: 'ADD_TODO',
    id: nextTodoId++,
    text
  }
}
```

```
// 异步 action
export const addTodoAsync = text => {
  // 返回函数，其中有 dispatch 参数
  return (dispatch) => {
    // ajax 异步获取数据
    fetch(url).then(res => {
      // 执行异步 action
      dispatch(addTodo(res.text))
    })
  }
}
```



## 异步 action

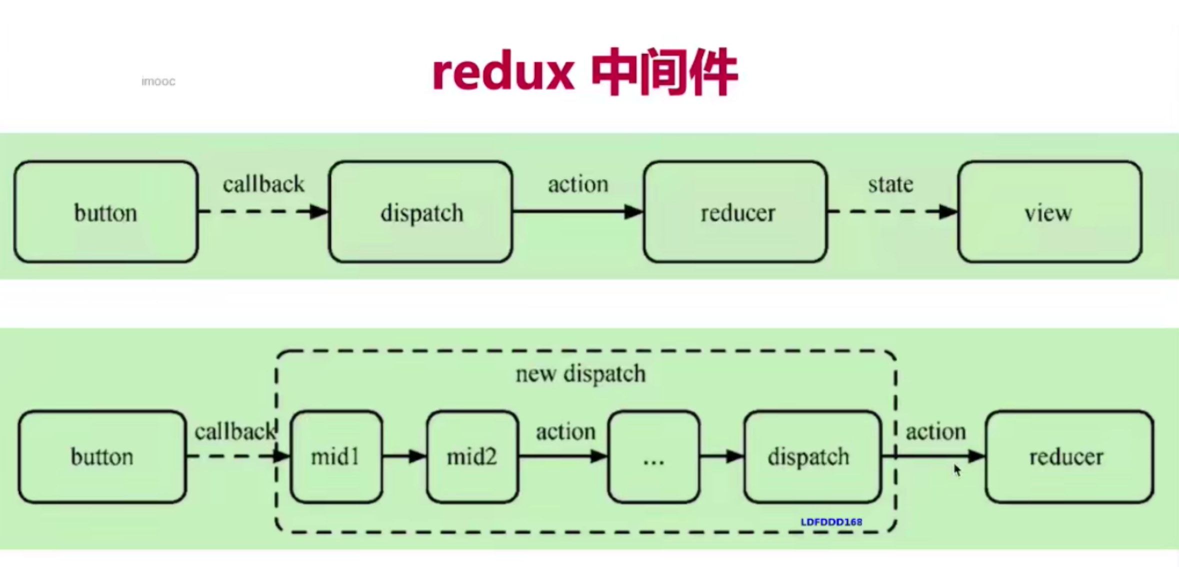
```
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
import rootReducer from './reducers/index';

// 创建 store 时，作为中间件引入 redux-thunk
const store = createStore(rootReducer, applyMiddleware(thunk));
```



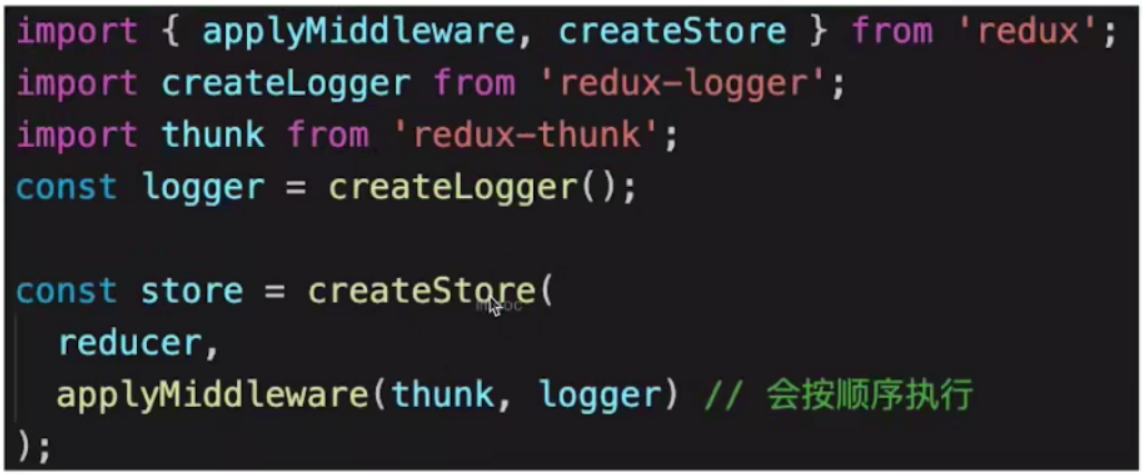
## 异步 action

- redux-thunk
- redux-promise
- redux-saga

## redux中间件

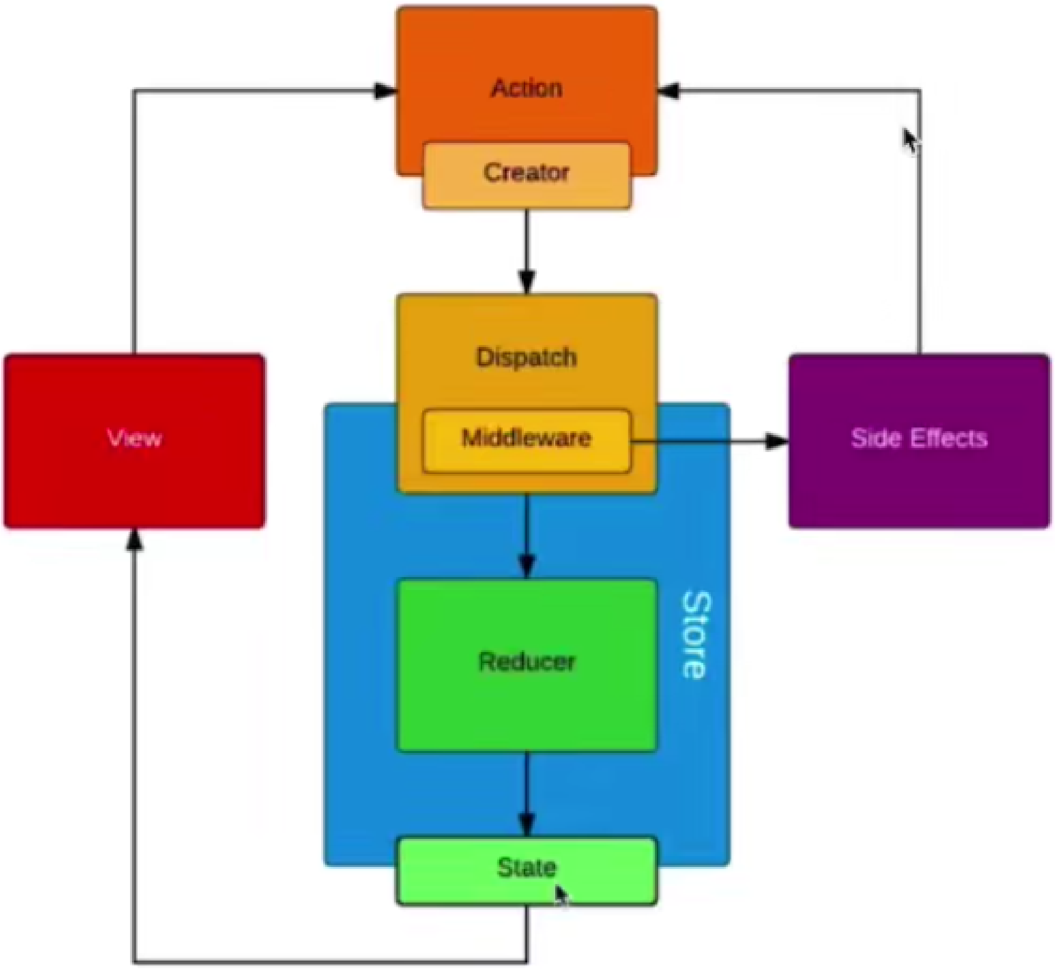


# redux 中间件 – 示例



```
import { applyMiddleware, createStore } from 'redux';
import createLogger from 'redux-logger';
import thunk from 'redux-thunk';
const logger = createLogger();

const store = createStore(
  reducer,
  applyMiddleware(thunk, logger) // 会按顺序执行
);
```

# redux 数据流图


Action
Creator
Dispatch
Middleware
View
Side Effects
Reducer
Store
State