# 使用redux-thunk中介軟體處理異步動作

# 本章目標

- 學習thunk樣式
- 學習redux-thunk中介軟體的使用目的
- 學習redux-thunk中介軟體如何處理異步程序

# thunks樣式是什麼

- 一種特別的函式，在其中包裹/封裝同步或異步程序，用於延遲求值的邏輯行為
- 不需依靠任何庫，只用JS語言核心的特性發展，常見用於解決異步流程控制
- thunks有一種樣式會有一個傳參，會是一個callback(回調函數)
- thunks可視為一種「臨時函式」或「形式替換函式」

# thunks様式(同步求值計算)

```
1 const add = (x, y) => x + y
2
3 const thunk = () => add(1, 2)
4
5 // 立即求值
6 add(1, 2)
7 // 延後求值
8 thunk()
```

# thunks樣式(異步求值計算)

```
1  // 異步延後求值
2  const addAsync = (x, y, callback) => {
3    setTimeout(
4      () => callback(x + y), 3000
5    )
6  }
7  // 包裝原本的異步函式，給定要得到3+5的值
8  const thunkAsync = (callback) => {
9    addAsync(3, 5, callback)
10 }
11 // 真實運行，然後打印
12 thunkAsync(
13   (sum) => { console.log(sum) }
14 )
```

# redux-thunk中介軟體是什麼

- 由redux官方設計的一種中介軟體，用於處理異步程序(副作用)
- redux-thunk需要搭配Action Creator(動作建立器)使用，此時Action Creator會是一個返回另一個函式的函式，這個返回的函式才是真正在動作發生時作派送使用的Action Creator
- redux-thunk只是一個運用了thunks樣式，用來協助調用這種轉換函數的中介軟體，程式碼行數非常少

# redux-thunk套用於項目中 - store

```
1 import { Provider } from 'react-redux'
2 import { createStore, applyMiddleware } from 'redux'
3 import thunk from 'redux-thunk'
4
5 const store = createStore(reducer, applyMiddleware(thunk))
```

index.js檔案中，建立store時

# redux-thunk套用於項目中 - 動作建立器

```
export const onItemAdd = payload => ({ type: types.ADD_ITEM, payload })

export const onItemAddAsync = (payload) => {
  return (dispatch) => {
    setTimeout(() => {
      // 延后發送，模擬異步(副作用)程序
      console.log('delay onItemAdd')
      dispatch(onItemAdd(payload))
    }, 2000)
  }
}
```

actions/index.js檔案中，動作建立器使用的檔案

# redux-thunk套用於項目中 - 元件

```
1 const App = ({ items, actions }) => (
2   <div>
3     <ItemInput onItemAdd={actions.onItemAddAsync}/>
4     <ItemList items={items} onItemDel={actions.onItemDelAsync}/>
5   </div>
6 )
```

containers/App.js檔案中，容器元件用的檔案

# Redux中介軟體結論

- redux-thunk只是提供了一種簡便的樣式，用於處理一般的異步運行流程
- 其他常見的熱門中間件還有redux-promise、redux-observable、redux-saga等中介軟體，各自用不同的的樣式來解決異步運行流程
- 中間件可以自行開發，理解它的運作原理與目的後，開發專屬的中介軟體不會太困難