hillel IT school

Lesson 36

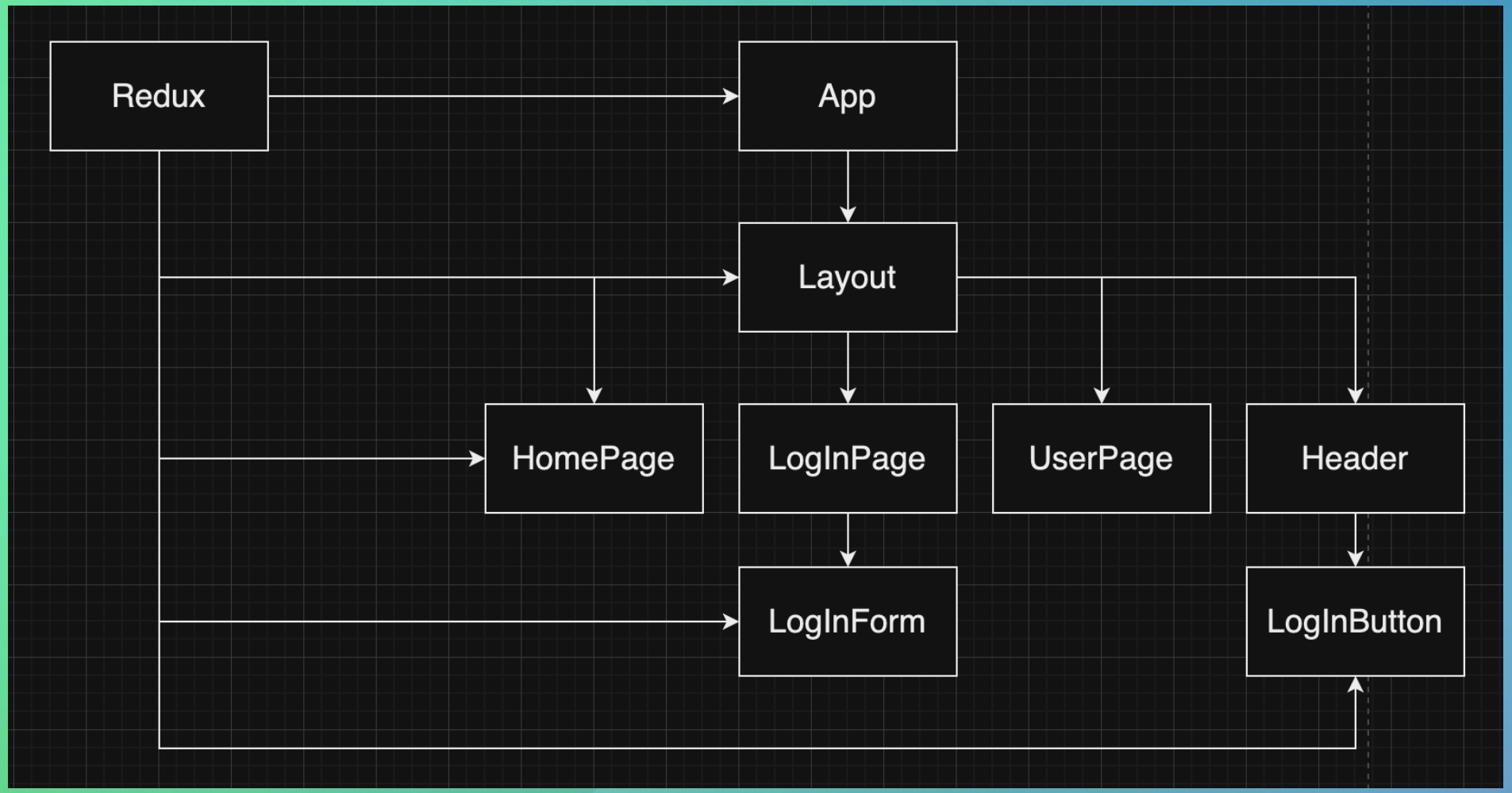
План заняття

- Redux
- Redux thunk

Згадаємо ФП

- Функція вищого порядку - це така функція яка може приймати іншу функцію як аргумент.
- Чиста функція - при отримані одних і тих самих аргументів надає один і той же результат. Також не визиває side effect.
- Імутабельність - незмінність вхідних данних.

Redux concept



Redux introducing

Redux – бібліотека управління станом JS додатків.

Redux - https://redux.js.org/

React-Redux - https://react-redux.js.org/ (офіційна бібліотека для використання Redux у React).

Redux-toolkit - https://redux-toolkit.js.org/ (набір елементів для ефективної розробки Redux).

Терміни:

- Store – сховище даних (деякий банк).
- Actions - дія з сховищем даних (покласти грощі до банку).
- Dispatcher - функція яка виконує дію (мобільний додаток або менеджер який несе грощі у сховище).
- Reducer - мехнізм взаємодії зі сховищем (прописані правила банку).

Redux introducing

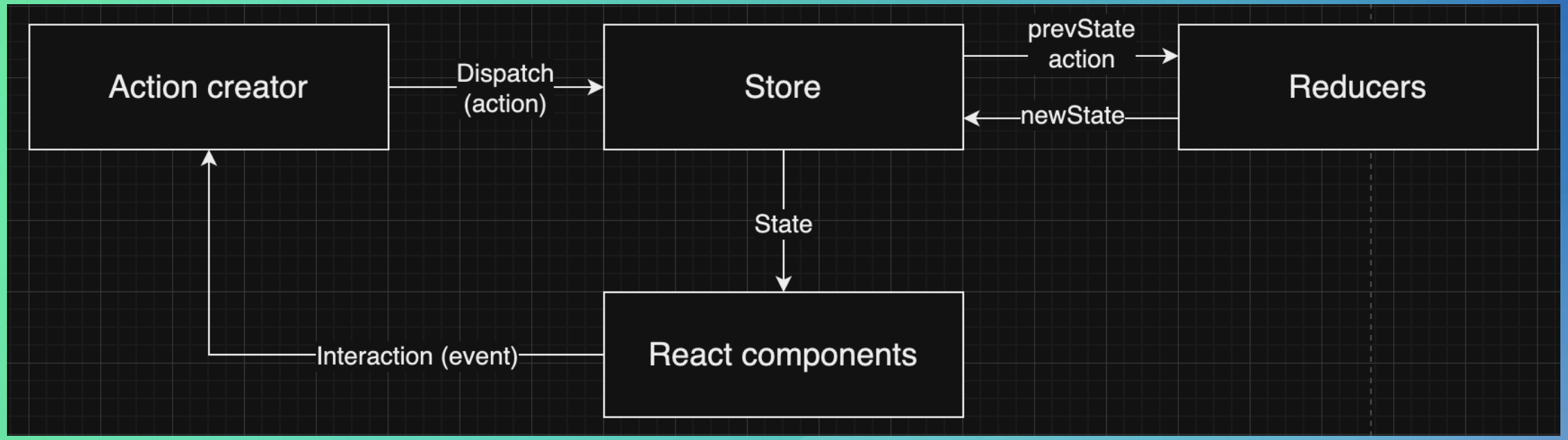
Redux допомагає впоратися зі спільним керуванням станом, але, як і будь-який інструмент, у нього є компроміси. Потрібно вивчити більше концепцій і написати більше коду. Це також додає деяку не пряму взаємодію до вашого коду та просить вас дотримуватися певних обмежень. Це компроміс між короткостроковою та довгостроковою продуктивністю.

Redux більш корисний, коли:

- У вас є велика кількість станів програми, які потрібні в багатьох місцях програми
- З часом стан програми часто оновлюється
- Логіка оновлення цього стану може бути складною
- Програма має кодову базу середнього або великого розміру, і над нею може працювати багато людей

Redux потрібен не всім програмам. Знайдіть час, щоб подумати про те, яку програму ви створюєте, і вирішити, які інструменти найкраще допоможуть вирішити проблеми, над якими ви працюєте.

Redux flow


Action creator
Dispatch
(action)
Store
prevState
action
Reducers
newState
State
React components
Interaction (event)

Store example

hillel IT school

```
const initialUsersState : {users: []}  = {
    users: [],
}

const initialUserInfoState : {name: null, surname: null}  = {
    name: null,
    surname: null
}

const store : {userInfo: {...}, users: {...}}  = {
    users: initialUsersState,
    userInfo: initialUserInfoState
}
```

Action example

hillel IT school

```
// ACTION TYPE
3 usages
export const ADD_USERS = 'ADD_USERS';
// ACTION
no usages
export const action : {payload: string, type: string}  = { type: ADD_USERS, payload: 'example' };
// ACTION CREATOR
no usages
export const addUsers = users => dispatch({ type: ADD_USERS, payload: users }) ;
```

Reducer example

hillel IT school

```
no usages
export const reducer = (state, action) : any & {...}  => {
    switch (action.type) {
        case ADD_USERS: {
            const newStore : any & {...}  = {
                ...state,
                users: action.payload
            }
            return newStore
        }
        // your cases here
    }
}
```

Redux instalation

Базова інсталяція до react

npm install redux react-redux @redux-devtools/extension

Якщо ви плануєте використовувати toolkit

npm install @reduxjs/toolkit react-redux

Якщо вам потрібні деякі пакети з базового Redux

npm install @reduxjs/toolkit react-redux redux

Get started

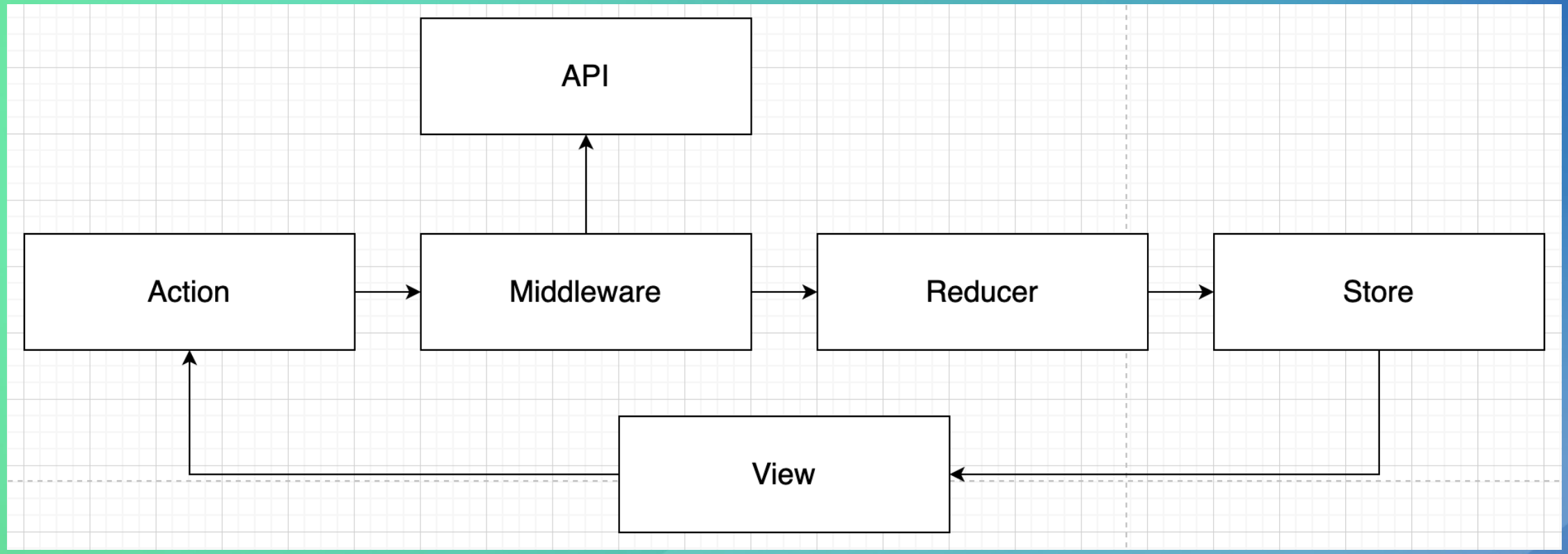
Є два шляхи як розпочати роботу у Redux:

- Олдскул, використовующи базові принципи Redux.
- З використанням redux-toolkit, набору методів які дуже спрощують життя.

Redux methods

- createStore - створює store у redux.
- combineReducers - об’єднує декілька редюсерів у один.
- connect - Використовувався у класових компонентах (сьогодні не розглядаємо).
- useDispatch && useSelector - надають можливість виконувати дії зі store та забирати данні.

Redux middleware



Redux devtools

Спеціальне розширення для браузеру, яке допомагає відслідкувати зміни у Redux, а саме:

- Які action були викликані
- Як змінився state
- Порівнювання попереднього та поточного state
- Та інші...

Github репозіторій https://github.com/reduxjs/redux-devtools

Redux thunk

Слово "thunk" - це програмний термін, який означає "фрагмент коду, який виконує деяку відкладену роботу". Замість того, щоб виконувати певну логіку зараз, ми можемо написати тіло функції або код, який можна використовувати для виконання роботи пізніше.

Зокрема, для Redux «thunks» — це шаблон написання функцій із внутрішньою логікою, які можуть взаємодіяти з методами dispatch і getState з Redux.

Використання thunks вимагає додавання проміжного програмного забезпечення redux-thunk до сховища Redux як частину його налаштування.

Thunks — це стандартний підхід до написання асинхронної логіки в програмах Redux і зазвичай використовується для отримання даних. Однак вони можуть використовуватися для різноманітних завдань і можуть містити як синхронну, так і асинхронну логіку.

Документація https://redux.js.org/usage/writing-logic-thunks

Gitgub репозіторій https://github.com/reduxjs/redux-thunk

Redux thunk

Функція thunk — це функція, яка приймає два аргументи: метод dispatch і метод getState. Функції Thunk не викликаються безпосередньо кодом програми. Замість цього вони передаються в dispatch()

Функція thunk може містити будь-яку довільну логіку, sync або async, і може викликати dispatch або getState у будь-який час.

Подібно до того, як код Redux зазвичай використовує actionCreator для створення об’єктів action для dispatch замість того, щоб писати об’єкти action вручну, ми зазвичай використовуємо actionCreator thunk для створення функцій thunk, які надсилаються. actionCreator thunk — це функція, яка може мати деякі аргументи та повертає нову функцію thunk. Thunk зазвичай закриває будь-які аргументи, передані творцю дії, тому їх можна використовувати в логіці

Redux thunk

Оскільки thunks є інструментом загального призначення, який може містити довільну логіку, їх можна використовувати для різноманітних цілей. Найпоширеніші випадки використання:

- Виведення складної логіки з компонентів
- Створення асинхронних запитів або іншої асинхронної логіки
- Написання логіки, яка потребує відправлення кількох дій поспіль або з часом
- Написання логіки, якій потрібен доступ до getState для прийняття рішень або включення інших значень стану в дію

Thunks — це одноразові функції, які не мають життєвого циклу. Вони також не можуть бачити інші надіслані дії. Таким чином, вони зазвичай не повинні використовуватися для ініціалізації постійних з'єднань, таких як веб-сокети, і ви не можете використовувати їх для відповіді на actions.

Thunks найкраще використовувати для складної синхронної логіки та простої та модерованої асинхронної логіки, як-от створення стандартного запиту AJAX і відправлення дій на основі результатів запиту.

Redux toolkit

У Redux є певі недоліки:

- Дуже багато коду для простих дій: створення об’єкту action, створення функції actionCreator, підключення redux store до компонентів (класвоих).
- Імутабельний state у reducer - слабочитаємий код.

Переваги redux-toolkit:

- Actions, actionCreator - створюються аттоматично.
- Mutable state – immerjs (https://immerjs.github.io/immer/).
- Зручна обробка асинхронних операцій.
- Готовий функціонал для операцій з даними.

Корисна література

Redux in 100 seconds: https://www.youtube.com/watch?v=_shA5Xwe8_4

Redux Tutorial: https://www.youtube.com/watch?v=poQXNp9ItL4

https://react-redux.js.org/api/hooks

Redux Thunk: https://www.digitalocean.com/community/tutorials/redux-redux-thunk-ru

Дякую за увагу