Міністерство освіти і науки України

Черкаський державний технологічний університет

Звіт

про виконання лабораторної роботи №2 з навчальної дисципліни

«Програмування WEB додатків»

Виконав: студент   
 Групи ПЗС-2044

Оратівський С.І.

Перевірив:

Метелап В.В.

Черкаси

2022

**Тема: React**

**Завдання: Переробити додаток на React**

Посилання на коміт: [SerhiiOrSoratik/internet-shop-Web-Application-Programming at 88a865ea96128216c80547dc97ea8a4a77f2405e (github.com)](https://github.com/SerhiiOrSoratik/internet-shop-Web-Application-Programming/tree/88a865ea96128216c80547dc97ea8a4a77f2405e)

index.js

import React from 'react';  
import ReactDOM from 'react-dom/client';  
import './index.css';  
import App from './App';  
  
const root = ReactDOM.createRoot(***document***.getElementById('root'));  
root.render(  
 <React.StrictMode>  
 <App />  
 </React.StrictMode>  
);

App.js

import './App.css';  
import {Header} from "./Components/Header";  
import {MainPage} from "./Components/MainPage";  
import {useState} from "react";  
  
function App() {  
  
 const [filter, setFilter] = useState('');  
 const [showForm, setShowForm] = useState(false);  
  
  
 return (  
 <>  
 <Header  
 setFilter={setFilter}  
 setShowForm={setShowForm}/>  
 <MainPage  
 filter={filter}  
 showForm={showForm}  
 setShowForm={setShowForm}/>  
 />  
 </>  
 );  
}  
  
export default App;

src/components/MainPage.js

import {useMemo, useState} from "react";  
import {getEntityById, loadData, removeEntityById} from "../Api/Api";  
import {Form} from "./Form";  
  
export const MainPage = (props) => {  
  
 const [allEntities, setEntities] = useState([]);  
 const [editingEntity, setEditingEntity] = useState({});  
  
 useMemo(() => {  
 loadData(props.filter).then(result => setEntities(result));  
 }, [props.filter])  
  
 const updateData = () => {  
 loadData(props.filter).then(result => setEntities(result));  
 }  
  
 const removeById = (id) => {  
 removeEntityById(id).then(() => loadData().then(result => setEntities(result)));  
 }  
  
 const editById = (entity) => {  
 setEditingEntity(entity)  
 props.setShowForm(true)  
 }  
  
 const downloadJsonFile = async (id) => {  
 const data = await getEntityById(id);  
 const json = ***JSON***.stringify(data);  
 let element = ***document***.createElement('a');  
 element.setAttribute('href', 'data:application/json;charset=utf-8,' + encodeURIComponent(json));  
 element.setAttribute('download', `${data.caption}.json`);  
 element.click();  
 }  
  
 const downloadScvFile = async (id) => {  
 const data = await getEntityById(id);  
 const csv = convertToCSV(data);  
 let element = ***document***.createElement('a');  
 element.setAttribute('href', 'data:application/json;charset=utf-8,' + encodeURIComponent(csv));  
 element.setAttribute('download', `${data.caption}.csv`);  
 element.click();  
 }  
  
 const convertToCSV = data => {  
 const keys = ***Object***.keys(data);  
 const values = ***Object***.values(data);  
 let result = keys.join(',');  
 result += '\n';  
 result += values.join(',')  
 return result;  
 }  
  
 return (  
 <>  
 <div id='container'>  
 {props.showForm ? <Form setShowForm={props.setShowForm} setEditingEntity={setEditingEntity}  
 editingEntity={editingEntity} updateData={updateData}/> : null}  
 <div id='catalog'>  
 {  
 allEntities ? allEntities.map(entity =>  
 <div style={{  
 border: '1px solid black',  
 width: '20%',  
 minWidth: '300px',  
 height: 'auto',  
 marginBottom: ' 25px'  
 }}  
 key={entity.id}>  
 <button onClick={() => removeById(entity.id)}  
 style={{  
 transform: ' translate(145px, 5px)',  
 position: 'absolute',  
 height: '25px',  
 width: '25px'  
 }}>x  
 </button>  
 <button onClick={() => editById(entity)}  
 style={{  
 transform: 'translate(115px, 5px)',  
 position: 'absolute',  
 height: 25,  
 width: 25  
 }}>✎  
 </button>  
 <button onClick={() => downloadJsonFile(entity.id)}  
 style={{  
 transform: 'translate(70px, 5px)',  
 position: 'absolute',  
 height: 25,  
 width: 40  
 }}  
 >json  
 </button>  
 <button onClick={() => downloadScvFile(entity.id)}  
 style={{  
 transform: 'translate(25px, 5px)',  
 position: 'absolute',  
 height: 25,  
 width: 40  
 }}  
 >csv  
 </button>  
 <p style={{  
 color: 'gray',  
 fontSize: 12,  
 paddingTop: 10  
 }}>{entity.availability ? ' ' : 'Немає в наявності'}</p>  
 <p>{entity.caption}</p>  
 <p>{entity.price} грн</p>  
 <p style={{wordWrap: 'break-word', color: 'gray'}}>{entity.description}</p>  
 </div>  
 ) : null  
 }  
 </div>  
 </div>  
 </>  
 )  
}

src/components/Header.js

export const Header = (props) => {  
  
 const search = () => {  
 const searchValue = ***document***.getElementById("search").value;  
 props.setFilter(searchValue)  
 }  
  
 return (  
 <div id="header">  
 <h1>Book shop</h1>  
 <div style={{width: '50%'}}>  
 <input style={{width: '15%', height: '88%'}} id="search"/>  
 <button className='form\_button' style={{width: '15%'}} id="searchButton" onClick={() => search()}>  
 Search  
 </button>  
 </div>  
 <button className='form\_button' id="addBook" style={{width: '15%'}} onClick={() => {  
 props.setShowForm(true)  
 }}>Add book  
 </button>  
 </div>  
 )  
}

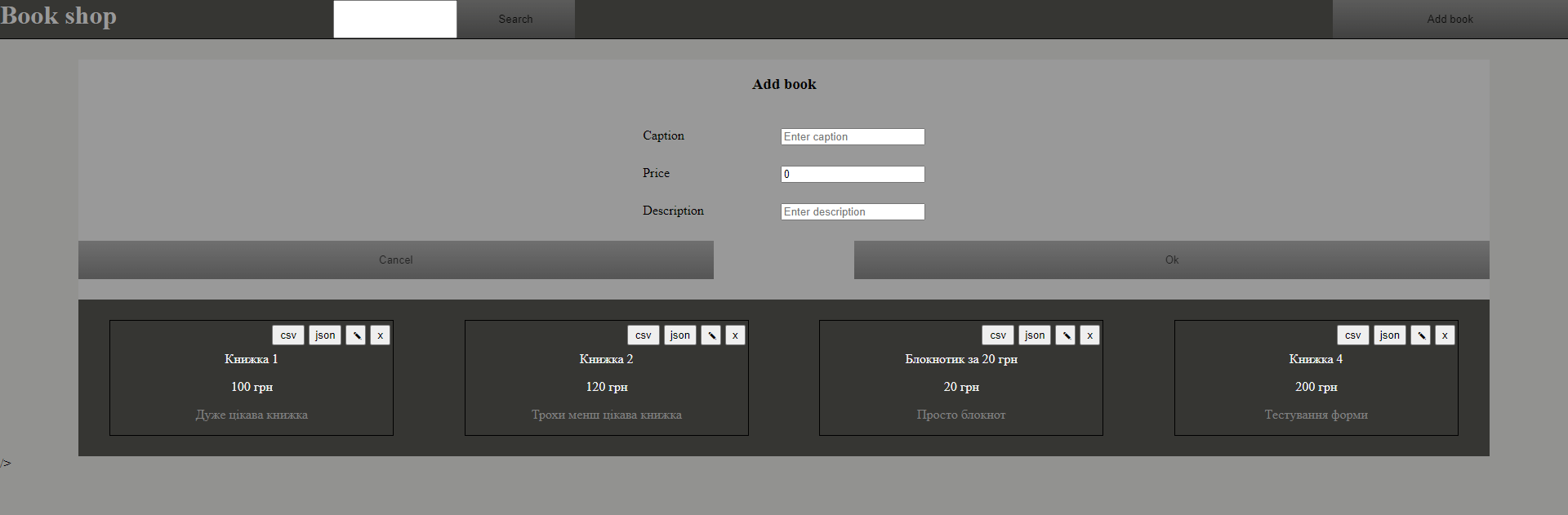
src/components/Form.js

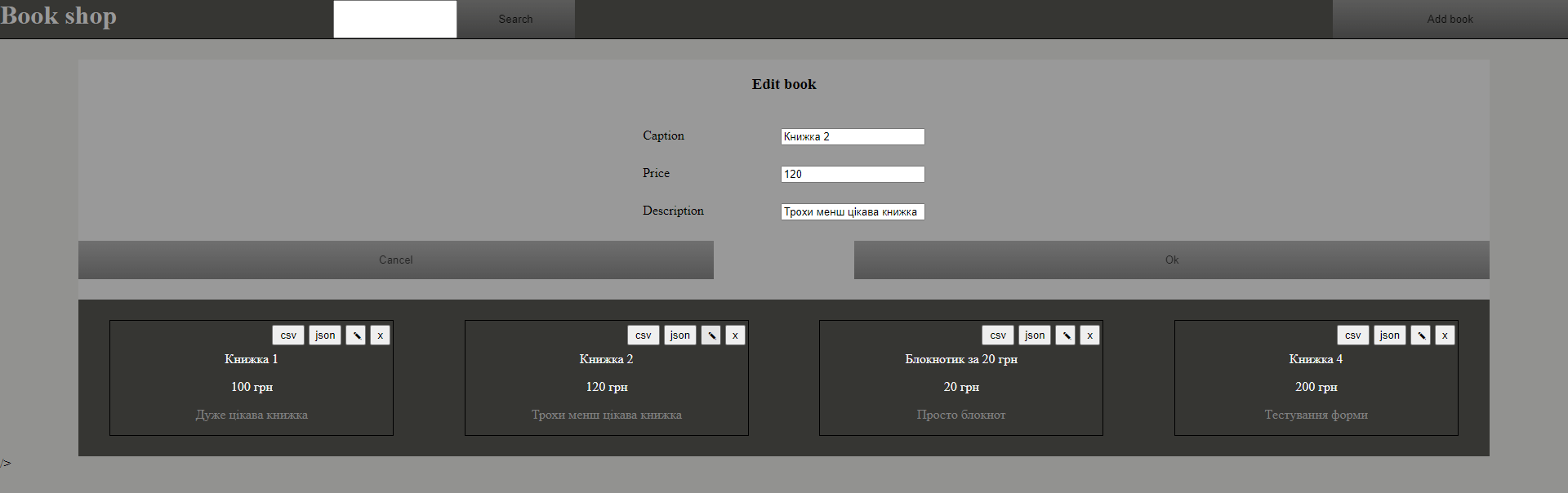
import {useState} from "react";  
import {createEntity, editEntityById} from "../Api/Api";  
  
export const Form = (props) => {  
  
 const [caption, setCaption] = useState(props?.editingEntity?.caption ? props.editingEntity.caption : '')  
 const [price, setPrice] = useState(props?.editingEntity?.price ? props.editingEntity.price : 0)  
 const [description, setDescription] = useState(props?.editingEntity?.description ? props.editingEntity.description : '')  
  
  
 const handleSubmit = (event) => {  
 event.preventDefault();  
 if (props.editingEntity?.id) {  
 editEntityById({caption, price, description}, props.editingEntity.id).then(() => props.updateData())  
 } else {  
 createEntity({caption, price, description}).then(() => props.updateData())  
 }  
 clear();  
 }  
  
 const clear = () => {  
 props.setShowForm(false)  
 setDescription('')  
 setPrice('')  
 setCaption('')  
 props.setEditingEntity({})  
 }  
  
 return (  
 <form id="crForm" encType="application/json" className="form-container" onSubmit={handleSubmit}>  
 <h3>{props.editingEntity?.id ? 'Edit book' : 'Add book'}</h3>  
 <div className="form-group">  
 <label htmlFor="cr\_caption">Caption</label>  
 <input id="cr\_caption" name="caption" placeholder="Enter caption" value={caption}  
 onChange={(e) => setCaption(e.target.value)} required/>  
 </div>  
 <div className="form-group">  
 <label htmlFor="cr\_price">Price</label>  
 <input id="cr\_price" type={"number"} name="price" placeholder="Enter price" value={price}  
 onChange={(e) => setPrice(+e.target.value)} required/>  
 </div>  
 <div className="form-group">  
 <label htmlFor="cr\_description">Description</label>  
 <input id="cr\_description" name="description" placeholder="Enter description" value={description}  
 onChange={(e) => setDescription(e.target.value)}/>  
 </div>  
 <div style={{display: 'flex', justifyContent: 'space-between'}}>  
 <button className="form\_button" value="Cancel" onClick={() => {  
 clear()  
 }}>Cancel  
 </button>  
 <button className="form\_button" type="submit">Ok</button>  
  
 </div>  
 </form>  
 )  
}

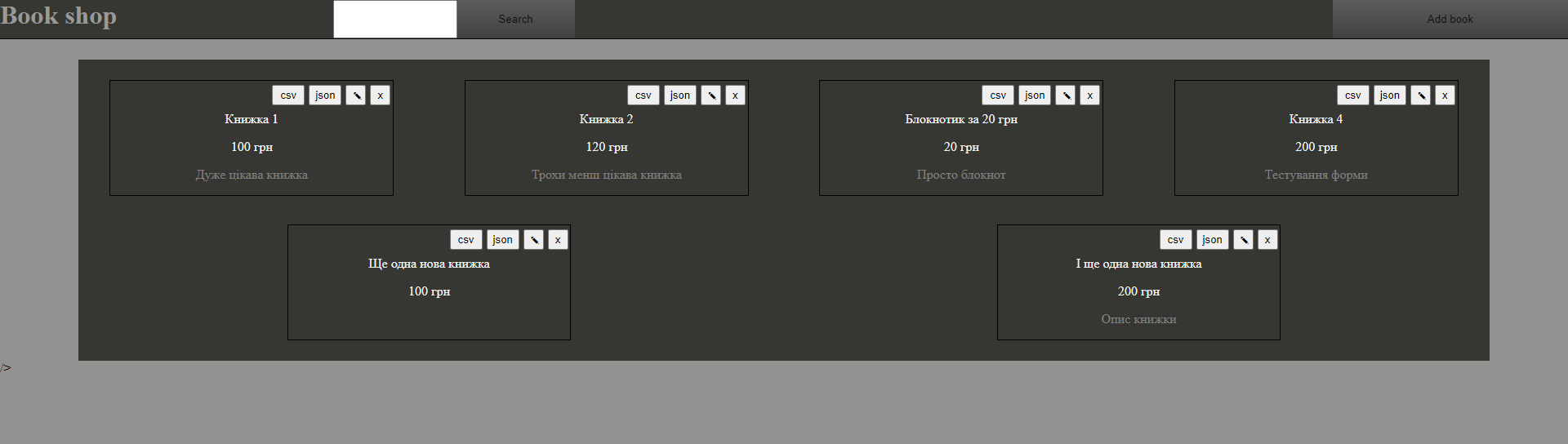
src/api/Api.js

export const loadData = async (searchValue) => {  
 if (searchValue) {  
 return await fetch('http://localhost:3001/product/search/' + searchValue,  
 {  
 method: 'GET',  
 headers: {'Accept': 'application/json'}  
 }).then(data => data.json())  
 .then(result => result)  
 .catch(error => ***console***.log(error))  
 } else {  
 return await fetch('http://localhost:3001/product',  
 {  
 method: 'GET',  
 headers: {'Accept': 'application/json'}  
 }).then(data => data.json())  
 .then(result => result)  
 .catch(error => ***console***.log(error))  
 }  
}  
  
export const removeEntityById = async (id) => {  
 return await fetch('http://localhost:3001/product/' + id,  
 {  
 method: 'DELETE',  
 headers: {'Accept': 'application/json'}  
 }).then(data => data)  
 .catch(error => ***console***.log(error))  
}  
  
export const getEntityById = async (id) => {  
 return await fetch('http://localhost:3001/product/' + id,  
 {  
 method: 'GET',  
 headers: {'Accept': 'application/json'}  
 }).then(data => data.json())  
 .then(result => result)  
 .catch(error => ***console***.log(error))  
}  
  
export const createEntity = async (entity) => {  
 return await fetch('http://localhost:3001/product',  
 {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: ***JSON***.stringify(entity)  
 }).then(data => data.json())  
 .then(result => result)  
 .catch(error => ***console***.log(error))  
}  
  
export const editEntityById = async (entity, id) => {  
 return await fetch('http://localhost:3001/product/' + id,  
 {  
 method: 'PUT',  
 headers: {'Content-Type': 'application/json'},  
 body: ***JSON***.stringify(entity)  
 }).then(data => data.json())  
 .then(result => result)  
 .catch(error => ***console***.log(error))  
}

Огляд результатів:







Питання:

1. Що таке React?

React — це бібліотека JavaScript для створення користувацьких інтерфейсів.

1. Які основні особливості React?

- Віртуальний ДОМ

- JSX

- Однонаправленний поток данных

- Рендеринг на стороні сервера

1. Що таке JSX?

JSX — це розширення синтаксису Javascript, яке дозволяє нам писати HTML у нашому JavaScript для створення React Elements

1. Що таке віртуальний DOM?

- Маніпулювання фактичним DOM дуже затратно

- Віртуальний DOM: React створює віртуальне представлення фактичного DOM в пам'яті

- Ожидаемые зміни спочатку відражаються в оновленій версії Virtual DOM. Обновлений віртуальний DOM порівнюється з його попередньою версією з використанням алгоритму React «diffing», щоб визначити, як краще всього оновити реальний DOM.

1. Що таке компонент в React?

Компонент: незалежний багаторазовий інтерфейс

1. Недоліки функціональних компонентів

- Для незвичайних життєвих циклів ще немає еквивалентів Hook: getSnapshotBeforeUpdate і componentDidCatch

- Старі сторонні бібліотеки можуть бути несумісними з хуками

1. Що викликає оновлення компонента?

- Повторний рендеринг батьків, який може отримати за собою нові реквізити

- SetState ()

1. Що робить setettate () і як він працює?

setState () планує оновлення об'єкта стану компонента. -Когда стан змінюється, компонент відповідає повторним рендерингом.

1. Як запобігти непотрібному повторному рендерингу?

- React.Memo: Компонент вищого порядку, який працює як -React.PureComponent, який використовується для компонентів функцій shouldComponentUpdate: метод життєвого циклу, який приймає наступні реквизити та стан і повертає логічне значення, що вказує, повинен повторно відображатися

1. Що таке ключі?

Ключі є постійними значеннями строк, які однозначно ідентифікують елемент списку серед своїх братів і сестер.

1. Які загальні підходи до повторного використання коду в React?

- Компонент высшего порядка

- Компонент вищого порядку: функція, яка бере компонент React і преобразовує його в інший компонент, обогащаючи його функціональність багатократного використання

- Розширений в сторонніх бібліотеках поширений приклад — підключення React Redux

1. Що таке хуки для React?

Позволяє компонентам функцій використовувати стан та інші попередні функції тільки для класу

1. Чому я повинен використовувати хуки?

Хуки дозволяють просто повторно використовувати логіку зі збереженням стану без багатоуровневих абстракцій, таких як HoC та Render -Props

1. Що таке хук використання штату?

useState — це хук, який дозволяє додати стан React для компонентів функцій

1. Що таке useEffect хук?

useEffect дозволяє виконувати побічні ефекти в компонентах функцій

1. Що таке хук useReducer?

useReducer — це альтернатива useState, яка використовується, коли існує складна логіка стану, яка включає в себе кілька значень, або коли наступне стан залежить від попереднього

1. Що таке props і state в React?

- props — це об'єкт JavaScript, переданий в Компонент, який містить властивості конфігурації

- реквізит невименен принимающим компонентом

- state — це об'єкт JavaScript, керований всередині компонента, який містить його внутрішнє стан.

1. Що таке Redux?

Redux – це бібліотека для керування станом програми

1. Які основні складові Redux?

Actions

- Action – це прості об'єкти Javascript з властивістю type, що вказують, який тип дії виконується

Action відправляються до центрального сховища за допомогою store.dispatch ()

Action зазвичай створюються функціями Action Creator, які при певному введенні генерують відповідну дію.

Перехідники

Reducers – це функції, які приймають Action та поточний стан та повертають результуючий стан.

Reducers повинні бути чистими функціями та не мати побічних ефектів

зберігати

Store - це центральний об'єкт, який підтримує та оновлює стан програми.

Store також займається реєстрацією та скасуванням реєстрації слухачів.

1. Як працює Redux?

- Redux обертається навколо суворого односпрямованого потоку даних

Перше: дія відправляється до магазину через store.dispatch (дія)

- Друге: Redux визначає результуючий стан викликаючи функцію Reducer.

- Третє: кореневий редуктор поєднує виведення кількох редукторів в одне дерево станів.

- Четверте: Store зберігає нове дерево станів та повідомляє зареєстрованих слухачів.

1. Що таке Redux Selector?

- Селектор – це функція, яка приймає стан сховища Redux та повертає отримані дані з цього стану.

1. Що таке Redux Toolkit?

- Redux Toolkit: офіційний самовпевнений набір інструментів -Redux, що дозволяє швидко приступити до роботи з Redux, не переймаючись шаблоном. Поставляється з готовими функціями та утилітами, які допомагають легко налаштувати магазин та створювати дії та редуктори.

1. Що таке контекстний API?

- Як і Redux, Context API – це рішення для керування станом

1. Як вирішити глобальне управління станом за допомогою Hooks?

- Передаючи useContext контекстному об'єкту, компоненти функції можуть підключатися до змін найближчого відповідного провайдера і будуть перемальовуватися під час оновлення

1. Як Context API порівнюється з Redux і коли ви використовуєте один поверх іншого?

Налаштування для Redux вимагає більше додаткової роботи, ніж Context,оскільки Context вбудований у React.

Для більш дрібних варіантів використання більш простих варіантів використання контексту має бути достатньо

Redux дозволяє доступ до проміжного програмного забезпечення, яке запускає функцію після надсилання дії

Redux надає доступ до надійного інструменту налагодження Redux - DevTools, який дозволяє вам подорожувати в часі та крок за кроком бачити зміни, внесені до вашого магазину.

1. Назвіть методи життєвого циклу компонента?

кожен компонент React проходить кілька етапів життєвого циклу, протягом яких викликаються такі методи:

constructor(props): ініціалізація компонента

componentWillMount(): викликається безпосередньо перед рендерингом компонента

render(): метод, який відповідає за рендеринг

componentDidMount(): метод викликається після рендерингу компонента

componentWillUnmount(): викликається перед видаленням компонента з DOM

1. Які методи компонента можуть бути викликані після деяких змін стану?

shouldComponentUpdate(nextProps, nextState): викликається щоразу, коли змінюється стан або props . Оскільки параметри отримують новий об'єкт props та стан. Якщо ці функції повертають true компонент, буде повторно перемальовано

componentWillUpdate(nextProps, nextState): викликається, якщо shouldComponentUpdate повертає true

componentDidUpdate(prevProps, prevState): викликається одразу після оновлення компонента

1. Чому важливо використовувати key для відображення елементів списку?

Ключі допомагають React ідентифікувати, які елементи було змінено, додано або видалено. Ключі надають елементам списку стабільну унікальність. Також використання ключів добре впливає на продуктивність.

1. Як обробляти подію натискання кнопки в React.js?

Передайте функцію обробника у властивість onClick елемента.

1. Що станеться, якщо передати функцію в метод setState?

Ця функція буде викликана з поточним станом як аргумент, ця функція дуже корисна, якщо ви хочете оновити стан цілісно.