**Розділ 1.**

**3**. Навіщо потрібно використовувати React?

І так, давайте розберемо цей простий приклад. Ви можете знайти код цього простого веб сайту, веб додатку, який прикріплений до цього відео.

На цьому веб сайті є якийсь контент і коли я натискаю на кнопку “видалити”, я отримую ось такий popup window - спливаюче віконце, на якому я можу натиснути або на “відміну” або на “ок”. Ці дві кнопки, звичайно, ніякої роботи не виробляють.

Коли я натискаю “видалити” з`являється такий шар, який покриває всю веб сторінку. Як бачимо, вона затемнюється, з'являється overlay.

Коли я натискаю на одну із кнопок: “відміна” або на “ок”, цей overlay зникає, так само як і popup віконце. Все що тут відбувається, контролюється за допомогою мови Javascript.

В індексі HTML файлу, є такий простий код, прості елементи. Також, у мене є якісь стилі. В принципі, вся інтерактивність відбувається за допомогою файлів script.js. На самому верху у мене є код прикріплення EventListener кнопки. Тобто, кнопка, яка відображається спочатку на веб сайті, от, до цієї кнопки “ видалити”.

Коли ця кнопка натиснута, запускається функція showModalHandler. Тут запускається декілька рядків з Javascript коду, які створюють декілька елементів. Як бачимо, “div”, далі “paragraph”, також кнопки “cancel” або “confirm”.

Для конфігурації елементів, також додається якийсь текст, якісь CSS класи, а також додається EventListener для цих кнопок, як бачимо. І тут, ми запускаємо при події функцію closeModalHandler.

За допомогою цієї функції, ми видаляємо оцей overlay, який накладається на веб сайт, також, функція showModalHandler. Є код, який комбінує всі ці елементи, кнопки і відображає їх на веб сторінці.

Тобто, це Javascript код, який відповідає за ось цю появу модального віконця, popup window, і, також накладання overlay на весь веб сайт.

Якби, все працює, але у нас для ось такої простої дії відображення простого елемента на веб сторінці, ми використовуємо досить велику кількість коду.

І, на даний момент, тут відображається всього лише одна задача у цьому додатку “Мої задачі”. Відображається одна задача вивчити React. І, якщо б ми, додали ще декілька задач, тобто, якщо б ми в індексі html ось тут додали ще декілька “div”, окрім вивчення React ще якісь задачі.

Або, наприклад, ми б використовували код для відображення контенту на цьому елементі після вилучення з якихось API, тоді цей код, звичайно, збільшився б в рази. Він став би набагато більш складно читаним і складним.

Наприклад, я не зміг би вибирати за допомогою querySelector button кнопку, так якби у нас було багато кнопок на веб сторінці, багато елементів.

І тому, нам потрібно було б переконуватися, що для різних задач відкриваються різні модальні віконця. Коли ми підтверджуємо видалення, то саме конкретна задача повинна бути видалена.

Зараз, у демо версії цього додатку, не відбувається видалення, але якщо це б відбулося, то це було б набагато складніше. І тому, хоча Javascript справляється з цією задачею, іноді у деяких простих випадках можна використати просто Javascript.

Але цьому є межа. Одна з причин - це те, що, якщо ви використовуєте Javascript, ви повинні писати код для кожного кроку, який повинен відбуватися.

Тобто, ось тут, ми повинні створити елемент, далі, створити його контент, далі, потрібно додати якісь класи, далі, нам потрібно додати EventListener, далі, ми повинні вказати, що повинно відбуватися, коли цей EventListener спрацює.

Тобто, кожний окремий крок повинен бути врахованим і описаним. Такий спосіб програмування і відображення чогось за допомогою цієї програми на екрані називається імперативним. Тобто, ми просто вказуємо дії за дією, послідовно крок за кроком.

Покроково. І цьому є межа. І навіть, якби не було цієї межі, тим паче, ми як розробники, повинні турбуватися про всі ці дрібні деталі.

Тобто, ми повинні писати код низького рівня. Наприклад, для створення кнопки, для встановлення контенту цієї кнопки. І як висновок, ми якби, винаходимо колесо, знову і знову.

Ми знову і знову, повторюємо одні і ті самі дії, виконуємо одні і ті самі задачі. І тепер, давайте розглянемо цей додаток, але створене за допомогою React.

Тобто, тут відбувається все те саме, та сама інтерактивність, але код виглядає трішки інакше. Знову ж, цей код буде також прикріплений до цього відео і тут звичайно код може виглядати для вас трішки більш складним, тут більше файлів.

Але все це звичайно, ми розберемо в курсі. Тобто, для чого нам потрібні всі ці файли? Ось у цьому файлі App.js, знаходиться головний код, за допомогою якого відображається ось цей контент на екрані.

І тут, у нас знаходиться кастомний HTML компонент. Тобто, кастомний HTML елемент і React насправді, це компоненти. Тобто, React, якби розбиває ваш додаток на невеликі будівельні блоки, на невеликі компоненти.

Кожен будівельний блок, будівельний компонент, має невелику чітку задачу. Тому ваш код за допомогою React буде набагато кращим у підтримці цього коду і також, в управлінні цього коду.

І React, буде піклуватися про те, щоб відобразити щось на веб сторінці і також на комбінуванні і взаємодії всього вашого коду.

Якщо ми роздивимось ось цей код кастомного HTML елементу, який знаходиться в файлі відповідно за назвою Todo.js, тобто задача, то тут знову ви можете побачити HTML код для відображення цієї задачі, точніше це відображається у popup window, модальне віконце, у якому ми підтверджуємо видалення якоїсь задачі.

І, звичайно зараз, цей код багато в чому для вас незрозумілий, але я ще раз повторю, ми все це з вами розберемо і ви навчитеся за цей курс розбиратися у всьому цьому на рівні спеціаліста.

Тим паче, ось цей контент, точніше, цей код відповідає за відображення ось цього контенту, а також за відображення ось цього модального віконця.

І хоча, зараз, на даний момент, ви не розумієте всього того, що тут відбувається і може бути дивним, що у нас знаходиться HTML код в Javascript файлі, у файлі Todo.js.

Незважаючи на всі ці дивності, я думаю вам зрозуміло, що тут нема ось цих от покрокових інструкцій рядок за рядком, що потрібно робити.

Замість цього ми визначаємо щось, що ми хочемо отримати в кінцевому результаті і також, у нас тут є якісь прості інструкції, які обробляються React для зміни того, що відображається на веб сторінці.

І, замість всіх цих низькорівневих конструкцій покрокових, коли ми говоримо Javascript що потрібно зробити, все це буде написано за нас або визначено, як завгодно, бібліотека React. Нам не потрібно робити це самостійно.

Тобто, ми як реактор розробники, працюємо на більш високому рівні. Нас не цікавлять низькорівневі деталі. І звичайно, це дозволяє створювати складні UI набагато простіше і легше.

Саме тому, ми і використовуємо бібліотеку React.js. Ця бібліотека дозволяє створювати сучасні складні UI набагато простіше.

І ця бібліотека дозволяє нам це робити, надаючи нам високорівневий синтаксис, у якому ми пишемо код, використовуючи декларативний підхід сфокусований на компонентах. Тобто, ми визначаємо, що ми хочемо бачити на екрані, на веб сторінці. Ми створюємо кастомні HTML елементи і все інше за нас робить бібліотека React.

І тому, наприклад, якщо ми хочемо створити ще одну задачу, ми можемо це зробити дуже легко. Ви це знову побачите це трохи пізніше. Саме тому, фреймворки або бібліотеки, або технології подібні React дуже популярні у сучасній розробці.