Асинхронний код. Promise

№ уроку: 16 **Курс:** JavaScript Базовий

Засоби навчання:	Visual Studio Code Web Browser

Огляд, мета та призначення уроку

Навчитися використовувати Promise для роботи з асинхронним кодом. Зрозуміти переваги використання Promise перед використанням зворотного виклику.

Вивчивши матеріал цього заняття, учень зможе:

- Використовувати функції зворотного виклику для організації асинхронного коду.
- Розуміти недоліки використання функцій зворотного виклику для асинхронного коду.
- Використовувати Promise для роботи з асинхронним кодом.
- Працювати з методами then, catch, finally для Promise.
- Використовувати ланцюжки Promise.
- Працювати з Promise API та використовувати методи all, race, allSettled.

Зміст уроку

1. Що таке асинхронний код.
2. Використання функцій зворотного виклику для асинхронного коду.
3. Використання Promise для асинхронного коду.
4. Ланцюжки Promise.
5. Обробка помилок під час роботи з Promise.
6. Promise API.

Резюме

- **Синхронний код** – код, який виконується послідовно. Доки не завершить роботу перша операція, друга операція не запуститься.
- **Асинхронний код** – код, операції у якому можуть виконуватися паралельно. Якщо операція запускається асинхронно, наступна операція може запуститися відразу після операції, запущеної асинхронно. Оскільки асинхронна операція виконується паралельно з іншими, необхідно обробити результат асинхронної операції у майбутньому.
- **Способи організації асинхронного коду:**
  - Callback-функції, або Функції зворотного дзвінка.
  - Promise.
  - Шаблон Observer.
- **Promise** – це об'єкт, який зберігає кінцевий результат відкладеної операції. Promise – значення, яке ще не наявне.
- Promise може бути у трьох станах:

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine
Tel.:0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- **Fullfiled** – асинхронна операція успішно завершена.
- **Rejected** – асинхронна операція закінчена з помилкою.
- **Pending** – асинхронна операція ще виконується.

- Promise, який перейшов у стан fullfiled або rejected називається **settled**.

- Для визначення дії, яка буде запускатися у випадку, якщо Promise перейшов в один зі станів, використовується функція then(f1, f2), де f1 – функція зворотного виклику, яка спрацює, якщо Promise перейшов у стан fullfiled, а f2 – функція зворотного виклику, яка спрацює під час переходу в rejected.

- **Недолік організації асинхронного коду через callback-функції** – необхідність використовувати додаткові параметри для всіх асинхронних операцій і складність управління певних послідовностей виклику асинхронних операцій через необхідність створювати вкладені функції зворотного виклику та погіршувати читабельність коду.

- Під час використання Promise можна будувати **ланцюжки Promise**. Немає потреби організовувати вкладеність коду, що полегшує розуміння асинхронного коду.

- **Catch** – функція обробки помилок, які виникли в ланцюжку Promise.

- **Finally** – функція для гарантованого виконання коду в ланцюжку Promise, спрацює незалежно від того, чи були в ланцюжку помилки, чи ні.

- **Promise.all(масив)** – метод, який дає змогу дочекатися завершення всіх Promise, вказаних у параметрі, та виконати дію після них.

- **Promise.race(масив)** – метод, який дочекається завершення одного з Promise, вказаних у масиві, та проігнорує решту.

- **Promise.allSettled(масив)** – чекає, коли всі Promise отримають стан і повертає новий Promise, який як параметр отримує масив з даними про стани та значення settled Promise.

Закріплення матеріалу

- У чому різниця синхронного та асинхронного коду?
- Опишіть принцип використання функцій зворотного виклику для організації асинхронного коду.
- У чому недолік використання функцій зворотного виклику під час роботи з асинхронним кодом?
- Що таке Promise?
- Назвіть основні методи Promise.
- У чому різниця Promise.all та Promise.race?
- Як можна обробити помилку в асинхронному коді, який організований через Promise?

Самостійна робота учня

Виконайте завдання у директорії Tasks\019 Asynchronous Code. Promises. Текст завдань розташований у коментарях, у тегах script.

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine

Tel.:0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Рекомендовані ресурси

Асинхронний
https://developer.mozilla.org/ru/docs/Glossary/Asynchronous

Синхронний
https://developer.mozilla.org/ru/docs/Glossary/Synchronous

Promise. Визначення
https://developer.mozilla.org/ru/docs/Glossary/Promise

Promise. Властивості та методи
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Promise

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine

Tel.:0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com