Асинхронный код. Promise

№ урока:	19	**Курс:**	JavaScript Базовый

Средства обучения:	Visual Studio Code Web Browser

Обзор, цель и назначение урока

Научиться использовать Promise для работы с асинхронным кодом. Понять преимущества использования промисов перед использованием функций обратного вызова.

Изучив материал данного занятия, учащийся сможет:

- Использовать функции обратного вызова для организации асинхронного кода.
- Понимать недостатки использования функций обратного вызова для асинхронного кода.
- Использовать Promise для работы с асинхронным кодом.
- Работать с методами then, catch, finally для промисов.
- Использовать цепочки промисов.
- Работать с Promise API и использовать методы all, race, allSettled.

Содержание урока

1. Что такое асинхронный код
2. Использование функций обработанного вызова для асинхронного кода
3. Использование Promise для асинхронного кода
4. Цепочки промисов
5. Обработка ошибок при работе с промисами
6. Promise API

Резюме

- Синхронный код – код, который выполняется последовательно, пока не завершит работу первая операция, вторая операция не запуститься.

- Асинхронный код – код, операции в котором могут выполняется параллельно. Если операция запускается асинхронно, следующая операция может запуститься сразу же после операции, запущенной асинхронно, но так как асинхронная операция выполняется параллельно с другими, необходимо способ обработать результат асинхронной операции в будущем.

- Способы организации асинхронного кода:
  - Callback функции или функции обратного вызова
  - Promise
  - Шаблон Observer

- **Promise** — это объект, который хранит конечный результат отложенной операции. Promise – представляет значение, которое еще не существует.

CyberBionic systematics

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

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Promise может находиться в трех состояниях – fulfilled, rejected, pending.

  Fullfiled – асинхронная операция успешно завершена
  Rejected – асинхронная операция закончена с ошибкой
  Pending – асинхронная операция еще выполняется

  Promise, который перешел в состояние fullfiled или rejected называется settled

- Для определения действия, которое будет запускаться в случае если Promise перешел в одно из состояний, используется функция then(f1, f2) где f1 - функция обратного вызова, которая сработает, если промис перешел в состояние fullfiled, f2 – функция обратного вызова, которая сработает при переходе в rejected.

- Недостаток организации асинхронного кода через callback функции – необходимость использовать дополнительные параметры для всех асинхронных операций и сложность управления определенных последовательностей вызова асинхронных операций из-за необходимости создавать вложенные функции обратного вызова и ухудшать читабельность кода.

- При использовании промисов, можно строить цепочки промисов. Нет необходимости организовывать вложенность кода, что упрощает понимание асинхронного кода.

- Catch – функция для обработки ошибок, которые возникли в цепочке промисов.

- Finally – функция для гарантированного выполнения кода в цепочке промисов, сработает независимо от того, были ли в цепочке ошибки или нет.

- Promise.all(массив) – метод, который позволяет дождаться завершения всех промисов, указанных в параметре и выполнить действие после них.

- Promise.race(массив) – метод, который дождется завершения одного из промисов, указанных в массиве и проигнорирует остальные.

- Promise.allSettled(массив) – дожидается, когда все промисы получат состояние и возвращает новый промис, который в качестве параметра получает массив с данными о состояниях и значениях settled промисов.

Закрепление материала

- В чем разница синхронного и асинхронного кода?
- Опишите принцип использования функций обратного вызова для организации асинхронного кода.
- В чем недостаток использования функций обратного вызова при работе с асинхронным кодом?
- Что такое Promise?
- Назовите основные методы Promise
- В чем разница Promise.all и Promise.race?
- Как можно обработать ошибку в асинхронном коде организованном через Promise

Самостоятельная деятельность учащегося

CyberBionic systematics

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

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Выполните задания в директории Exercises\Tasks\019 Asynchronous Code. Promises. Текст задач, находится в комментариях в тегах script.

Рекомендуемые ресурсы

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

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

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

Promise. Свойства и методы
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Promise

CyberBionic systematics

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

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com