STEP IT ACADEMY

Занятие №32

Асинхронность. Промисы. Async-Await. Fetch

Что такое асинхронность?

Асинхронность

В JavaScript почти всё выполняется синхронно. То есть идёт сверху вниз.

Но есть также и асинхронные операции, которые должны выполниться позже остальных. И таких операций есть несколько видов. Например:

1. Вызвать функцию позже
2. Поставить слушатель события
3. Асинхронные функции

Промисы

По сути промисы это специальная обертка для асинхронности, которая добавляет нам удобство при написании кода. Объект Promise (обещание) используется для отложенных и асинхронных вычислений.

У него есть три состояния:

pending: Ещё не отправлен

fulfilled: Успешно отправлен

reject: Ошибка

Создание промисов

```
let promise = new Promise(function (resolve, reject) {
  if (5 % 2 !== 0) {
    resolve("YEEEEAH");
  } else {
    reject("baad");
  }
});

console.log(promise);
```

Методы промиса

.then(fn) - Метод принимает колбэк-функции для случаев выполнения обещания.

.catch(fn) - работает только в случае отклонения обещания

.finally(fn) - работает всегда

Async, await

Если мы хотим работать с асинхронными операциями и хотим использовать async, await, то наша функция должна быть асинхронной. То есть если мы в функции ждём какие-то промисы через ключевое слово await, то функцию нужно помечать, как async. Мы не перейдем к следующей строчке пока не выполнится код, где await

```
const delay = ms => {
  return new Promise(r => setTimeout(() => r(), ms))
}

async function fetchAsyncTodos() {
  await delay(2000)

}
```

Давайте подведем итоги!

Чему мы научились?

Что мы использовали?