Async/Await в JavaScript

Async и Await в JavaScript

async и await — синтаксический сахар, введённый в ES2017 (ES8), который позволяет писать асинхронный код в стиле, похожем на синхронный. Это делает код более читабельным и понятным, упрощая работу с промисами.

Async-функции

Ключевое слово async используется для объявления асинхронной функции. Такая функция всегда возвращает промис. Если функция явно не возвращает промис, JavaScript автоматически оборачивает результат в промис.

Пример 1:

```
async function getStarWarsMovies() {
  return 1;
}
console.log(getStarWarsMovies());
// Promise { <state>: "fulfilled", <value>: 1 }
```

Пример 2:

```
async function example() {
  return 'Hello, world!';
}
example().then(result => console.log(result));
// Выведет: "Hello, world!"
```

Await

Ключевое слово await используется для ожидания выполнения промиса. await можно использовать только внутри async-функции.

Пример:

```
async function getStarWarsMovie(id) {
  const response = await fetch(`https://swapi.dev/api/films/${id}/`);
  console.log('ответ получен', response);
```

```
  return response.json();
}
const movies = getStarWarsMovie(1).then((movie) => {
  console.log(movie.title);
});
console.log('результат вызова функции', movies);
```

Порядок вывода:
1. Вызвали функцию, она начала выполнять асинхронную операцию и вернула промис → 'результат вызова функции' Promise
2. Получили ответ API, продолжаем выполнение функции → 'ответ получен' Response
3. Сработал .then() → 'A New Hope'

Почему response, а не промис?

Что возвращает fetch()? Промис, который при разрешении возвращает объект Response.
Как работает await? Ожидает завершения промиса. Когда промис разрешается — возвращается объект Response, а не промис.
Что такое Response? Это объект, содержащий:
- статус ответа (response.status)
- заголовки ответа (response.headers)
- тело ответа (response.json())

try...catch

try...catch позволяет перехватывать ошибки и обрабатывать их, не прерывая выполнение программы.

Синтаксис:

```
try {
  // код
} catch (error) {
  // обработка ошибки
}
```

Пример:

```
try {
  let result = riskyFunction();
  console.log('Result:', result);
} catch (error) {
  console.error('Error occurred:', error.message);
}
```

finally

Блок finally выполняется всегда.

Синтаксис:

```
try {
  // код
} catch (error) {
  // обработка ошибки
} finally {
  // выполняется всегда
}
```

Пример:

```
try {
  let result = riskyFunction();
  console.log('Result:', result);
} catch (error) {
  console.error('Error occurred:', error.message);
} finally {
  console.log('This will always execute');
}
```

Преимущества async/await

1. Читаемость и структура кода: он становится более линейным и легче читаемым. Это помогает избежать цепочек .then().

Async/Await в JavaScript

Пример с промисами:

```
fetch('https://swapi.dev/api/films/1/')
  .then(response => response.json())
  .then(movie => {
    console.log(movie.title);
  })
  .catch(error => {
    console.error('Ошибка:', error);
  });
```

Тот же пример с async/await:

```
async function getMovie() {
  try {
    const response = await fetch('https://swapi.dev/api/films/1/');
    const movie = await response.json();
    console.log(movie.title);
  } catch (error) {
    console.error('Ошибка:', error);
  }
}
getMovie();
```