Лекция 4 - Расширенные возможности JavaScript

Сегодня мы погрузимся в современные аспекты языка, которые упрощают и улучшают нашу жизнь как разработчиков.

Время быстро меняется, и JavaScript также меняется. Новые версии стандарта ECMAScript добавляют множество новых функций, улучшают старые и устраняют некоторые проблемы.

Краткий экскурс по истории ECMAScript и появление ES6 (ES2015):
ECMAScript - это стандарт языка, на котором основан JavaScript. Первый стандарт был опубликован в 1997 году, но наибольший скачок в его развитии произошел в 2015 году с выходом ES6.

2. Асинхронное программирование

Асинхронность позволяет JavaScript выполнять другие задачи, не дожидаясь завершения предыдущей.

Проблема callback-ов и "Callback Hell":

Когда асинхронные операции зависят друг от друга, мы вынуждены вкладывать callback в callback, что приводит к "аду обратных вызовов".

Что такое Promise? Это объект, представляющий конечное завершение (или отказ) асинхронной операции.

Пример:

javascript

```javascript
const promise = new Promise((resolve, reject) => {
  if (true) {
    resolve('Успех!');
  } else {
    reject('Ошибка!');
  }
});
```

Создание и использование Promise:

Пример:

```javascript
const promise = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Завершено!"), 1000);
});
```

Методы then, catch, и finally:

Эти методы позволяют обрабатывать результат или ошибку и выполнять действия по завершении Promise.

- **then** выполняется после успешного завершения Promise.
- **catch** ловит ошибки.
- **finally** выполняется после завершения Promise, независимо от его результата.

async/await:

Этот синтаксис позволяет работать с асинхронными операциями, как если бы они были синхронными.

Пример:

javascript

```javascript
async function fetchData() {
  const data = await fetch("https://api.example.com/data");
  return data.json();
}
```

Введение в ES6+ (ECMAScript 2015 и далее)

let и const:

let позволяет создавать переменные с блочной областью видимости. const также создает переменную, значение которой нельзя изменить.

- **let** позволяет объявлять переменные с блочной областью видимости.
- **const** объявляет переменные, значения которых не могут быть изменены.

Стрелочные функции:

Они имеют более короткий синтаксис и не имеют своего **this**.

Пример:

javascript

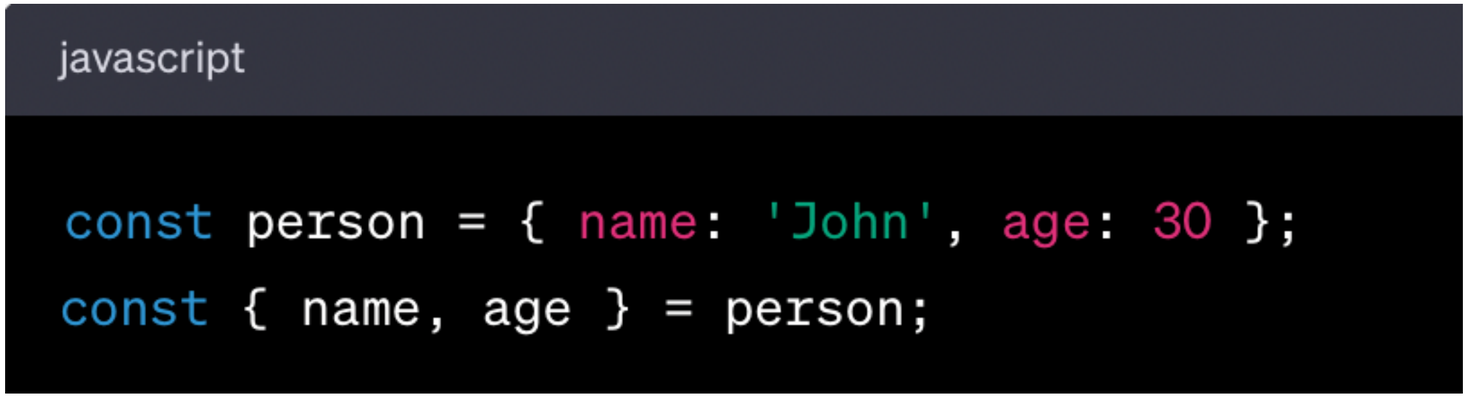
```javascript
const greet = name => `Hello, ${name}!`;
```

Классы:

- Синтаксис классов обеспечивает ясное и структурированное создание объектов и прототипного наследования.

Деструктуризация:

- Упрощает извлечение данных из объектов и массивов.
- **Пример:**



И другие возможности ES6+:

Шаблонные строки: `Hello, ${name}!`

Оператор расширения: ...arr

Параметры по умолчанию: function greet(name = "Guest")

4. Хранение данных в браузере

localStorage:
Позволяет сохранять пары ключ-значение на длительное время. Не исчезает после закрытия браузера.

sessionStorage:
Похож на localStorage, но хранит данные только на протяжении сессии.

Современные возможности JavaScript делают язык еще мощнее и гибким. Рекомендуем продолжить изучение и практику, чтобы быть на волне последних трендов.

Вопросы

1. В чем основное преимущество использования Promise перед callback-ами?
2. Какие основные отличия между var, let и const?
3. Как работает async/await и в чем его преимущества?
4. Как сохранить и извлечь данные из localStorage?
5. Перечислите некоторые из новых возможностей, представленных в ES6.