Sets в JavaScript

Sets - это новая структура данных, представленная в стандарте ECMAScript 6 (ES6), которая предоставляет уникальные значения без дубликатов. Они представляют собой коллекции значений, где каждый элемент уникален, и порядок элементов не имеет значения.

```
const mySet = new Set();
mySet.add(1);
mySet.add(2);
mySet.add(3);
console.log(mySet); // Set {1, 2, 3}

const flights = ['Russia', 'USA', 'London',  'London', 'USA', 1, false];
// Внутрь сета передается любой итерируемый объект - это любой объект по которому
// можно проходиться поэлементно. Это может быть массив, это может быть строка
const setFlights = new Set(flights);
console.log(setFlights);    // Set(5) {'Russia', 'USA', 'London', 1, false}
```

Основные характеристики Sets:

1. **Уникальность элементов**: Sets не могут содержать дублирующиеся элементы. Если вы попытаетесь добавить элемент, который уже существует в Set, он будет проигнорирован.
2. **Неупорядоченность**: Элементы в Set не имеют определенного порядка, и вы не можете обратиться к ним по индексу, как в массиве.
3. **Итерируемость**: Sets можно итерировать, перебирая элементы с помощью цикла `for...of` или метода `forEach()`.

Операции с Sets:

- `add(value)`: Добавляет элемент в Set, если его еще нет.
- `delete(value)`: Удаляет элемент из Set.
- `has(value)`: Проверяет, существует ли элемент в Set.
- `size`: Возвращает количество элементов в Set.
- `clear()`: Удаляет все элементы из Set.
- `forEach(callback)`: Итерирует по элементам Set, вызывая функцию обратного вызова для каждого элемента.

```
// size: Возвращает количество элементов в Set.
console.log(setFlights.size);    // 5

// has(value): Проверяет, существует ли элемент в Set.
console.log(setFlights.has('Russia'));    // true
console.log(setFlights.has('s'));    // false

// add(value): Добавляет элемент в Set, если его еще нет
// принимает булево значение, число или строку
setFlights.add('Paris');
setFlights.add('Paris');    // Дубликат будет проигнорирован
console.log(setFlights);    // Set(6) {'Russia', 'USA', 'London', 1, false, 'Paris'}

// delete(value): Удаляет элемент из Set.
setFlights.delete('London');
console.log(setFlights);    // Set(5) {'Russia', 'USA', 1, false, 'Paris'}

// перебор элементов
for (const flight of setFlights) {
  console.log(flight);    // Russia   // USA   // 1   // false   // Paris
}
// или
setFlights.forEach((value) => {
  console.log(value);    // Russia   // USA   // 1   // false   // Paris
});

// Из сета в массив с помощью spread синтаксиса
console.log([...setFlights]);    // (5) ['Russia', 'USA', 1, false, 'Paris']

// Мы можем получить сет объектов, но уникальности у них не будет
const setObj = new Set([{ a: 1 }, { b: 2 }, { b: 2 }]);
console.log(setObj);    // Set(3) {{a: 1}, {b: 2}, {b: 2}}

// Мы можем создать сет не только из массива, но и из строки
console.log(new Set('abcd'));    // Set(4) {'a', 'b', 'c', 'd'}

// Но если мы возьмем неитерабельный объект, то получим ошибку
// console.log(new Set({ a: 1 }));    // Uncaught TypeError: object is not iterable
// console.log(new Set(false));    // Uncaught TypeError: boolean false is not iterable

// clear(): Удаляет все элементы из Set.
setFlights.clear();
console.log(setFlights);    // Set(0) {size: 0}
```

Sets полезны, когда вам нужно хранить уникальные значения и быстро проверять их наличие. Они предоставляют эффективный способ работы с уникальными элементами в JavaScript.