Коллекция Map. Основные недостатки и отличия.

В этом уроке мы познакомимся с ассоциативным массивом, а также разберем его **преимущества и недостатки. Еще научимся работать с ним в языке JavaScript.**

Map как объект

Ассоциативный массив — это абстрактный тип данных, который позволяет хранить пары вида «ключ, значения». Он поддерживает операции добавления пары, а также поиска и удаления по ключу.

JavaScript-объект:

```
const person = {
  firstName: 'Alan',
  lastName: 'Kay',
};

person.firstName; // Alan
person['firstName']; // Alan
```

Преимущества Map

Использование квадратных скобок дает несколько преимуществ:

- Динамическое обращение
- Динамическое обновление
- Итерация

1. **Динамическое обращение**

Можно динамически обращаться к свойствам или ключам через переменную, в которой хранится имя ключа, вместо обращения к свойству через точку. Например:

```
const person = {
  firstName: 'Alan',
  lastName: 'Kay',
};

const key = 'lastName';
person[key]; // Kay
```

Внутри квадратных скобок конструкция [key] ожидает выражение. Это может быть что угодно, например, мы передаем константу key, которая внутри содержит строку. Так мы получаем доступ к внутреннему значению.

Это может происходить в циклических конструкциях, где мы не знаем, с каким ключом работаем на каждой итерации.

2. **Динамическое обновление**

Так же как и с динамическим обращением вместо обновления свойства через точку мы можем делать то же самое через квадратные скобки:

```
const person = {
  firstName: 'Alan',
  lastName: 'Kay',
};

person['firstName'] = 'John';
// person.firstName = 'John';

const propName = 'lastName';
person[propName] = 'King';
```

В этом случае синтаксис практически не меняется.

Мы используем внутри не литерал, а переменную. Она в себе содержит имя, к свойству которого мы обращаемся.

3. **Итерация**

Для итерации по свойствам объекта в JavaScript есть методы, которые находятся не внутри самого объекта.

Чтобы итерировать, можно воспользоваться конструктором Object и вызвать у него метод keys, который возвращает нам ключи переданного объекта:

```
const person = {
  firstName: 'Alan',
  lastName: 'Kay',
};

Object.keys(person).forEach((propName) => console.log(person[propName]));

// Alan
// Kay
```

На выходе конструкции Object.keys(person) получается массив.

Можно использовать любую функцию высшего порядка. Например, forEach, которая проходит поэлементно и рассчитывает, что мы внутри выполним какое-то действие.

Обратиться к нашему объекту, в квадратных скобках указывая propName — переменная, содержащая ключ, который был извлечен из person. Как раз здесь проявляется динамическая природа квадратных скобок и ее удобства. Через точку у нас бы не получилось так сделать.

Есть и другой способ:

```
const person = {
  firstName: 'Alan',
  lastName: 'Kay',
```

```
};

Object.values(person).forEach((value) => console.log(value));

// Alan
// Kay
```

Если нужны не ключи, а значения, то мы можно вызвать метод Object.values() и получить сразу список значений из объекта.

Недостатки Map

У объектов, как и у ассоциативных массивов, есть определенные недостатки, часть из которых довольно критические:

- Дополнительные свойства
- Ключи только строки и символы
- Определение размера

1. **Дополнительные свойства**

У Object есть большое количество свойств, которые даются ему по наследству. Эти свойства в большинстве случаев являются функциями:

```
const obj = {};
console.log(obj.valueOf);
// [Function: valueOf]
```

Например, есть функция valueOf(). При работе со свойствами, особенно в динамическом режиме, мы можем случайно заменить ее. В итоге это приведет к определенным проблемам. Это довольно критическая вещь для определенных типов программ.

Ключи только строки и символы

Допустим, есть такой объект:

```
const obj = { 3: 'value'};
// { '3': 'value' }
```

В итоге тройка превратилась в строку. То есть внутри произошло преобразование. В работе это может быть неудобно.

Определение размера

Object не дает определить размер. Поэтому можно сделать так:

```
Object.keys(obj).length;
```

Через Object.keys(), достаем все ключи и вычисляем длину. Это не очень удобный способ вычислять размер объекта.

Тип Map

Map — это отдельный тип. Если сделать на нем new без переданных аргументов, то можно получить объект, с которым можно дальше работать:

const map = new Map();

При этом мы можем передать аргумент, который внутри станет ключевым значением:

new Map([[key, value], [key2, value2]]

Это массив массива, в котором каждый массив — это элемент из двух пар: ключ и значение. Внутри он превращается в [key, value].

Установка нового значения происходит через set:

```
// Принимается два параметра key и value
map.set('key', 'value');
map.set(10, 'another value');

// Смотрим размер
map.size; // 2

// Получаем значение обратного
map.get('key'); // value
map.get(10); // another value
```

Этот простой интерфейс позволяет работать и не переживать о перечисленных недостатках ранее.

1. **Map как коллекция**

Если необходимо работать с Map как с коллекцией, то для этого он дает из коробки несколько методов:

keys() – возвращает итерируемый объект по ключам,

values() – возвращает итерируемый объект по значениям,

entries() – возвращает итерируемый объект по парам вида [ключ, значение], этот вариант используется по умолчанию в специальном цикле for..of.

свойство forEach:

```
map.forEach((value, key) => console.log(key, value));
```

ДОПОЛНИТЕЛЬНЫЕ МЕТОДЫ С ПРИМЕРАМИ:

1. **свойство size -сколько элементов в Map:**

```
console.log('Количество элементов в map: ', map.size);
```

2. **Поиск элементов с помощью метода has(ключ):**

```
// вернёт true, если map содержит элемент с ключём, 'John'
console.log(map.has('John'));

// вернёт false, если map не содержит элемент с ключём, 'Tapas'
console.log(map.has('Tapas'));
```

3. **Удаление элемента методом delete(ключ):**

```
map.delete('Sam'); // удалит элемент с ключём, 'Sam'.
```

4. **Для удаления всех элементов используйте метод clear():**

```
// Очистить map удалив все элементы
map.clear();

map.size // Вернёт, 0
```

Задания для решения:

1. Даны 4 массива. Необходимо создать коллекцию Map, сделать ключами коллекции эти массивы, а значениями - какие-нибудь строки. Определить количество всех элементов Map.

2. Даны 2 объекта и 2 массива. Необходимо создать коллекцию Map, сделайте ключами коллекции объекты, а значениями - соответствующие массивы. Удалить элемент по ключу.