> Эта страница была переведена с английского языка силами сообщества. Вы тоже

М

Коллекции

Этот раздел содержит обзор коллекций `Set` и словарей `Map (en-US)` - встроенных структур данных с доступом по ключу.

Словари

Тип Map

`Map (en-US)` - реализация простого ассоциативного массива (словаря). Он содержит данные в виде набора пар ключ/значение(ключи уникальны) и предоставляет методы для доступа и манипулирования этими данными.

Также как и объект, словарь позволяет

- получать значение по ключу, а также проверять наличие ключа
- добавлять/удалять пары ключ/значение
- перезаписывать значение по ключу (ключи уникальны).
- итерироваться по ключам

Словари, как специализированная структура данных, имеют существенные преимущества по сравнению с объектами:

- Ключи словаря могут быть любого типа (а не только строки).

- Словарь хранит свой размер (не надо вычислять).
- Натуральный порядок обхода элементов (в порядке добавления) с помощью `for...of` .
- Словарь не подмешивает ключи из прототипа (в отличие от объекта).

В следующем примере приведены основные операции со словарём:

```
JS
```

```
var sayings = new Map();
sayings.set("dog", "woof");
sayings.set("cat", "meow").set("elephant", "toot");
//вызов функции .set возвращает Map, поэтому set можно объединять в цепочки

sayings.set("dog", "гав-гав"); // заменить значение по ключу

sayings.size; // 3
sayings.get("fox"); // undefined
sayings.has("bird"); // false
sayings.delete("dog");

for (var [key, value] of sayings) {
  console.log(key + " goes " + value);
}
// "cat goes meow"
// "elephant goes toot"
```

Больше примеров и полное описание на странице справочника `Map (en-US)` .

Тип WeakMap

`WeakMap` это специальный вид словаря, ключами которого могут быть только объекты, причём ссылки на них в WeakMap являются *слабыми* (не учитываются сборщиком мусора (garbage collector GC)).

> Примечание: Интерфейс `WeakMap` совпадает с `Map` , единственное отличие - ключи `WeakMap` нельзя итерировать (т.е. нельзя получить список ключей). Это понятно, поскольку в таком случае возникла бы неопределённость с достоверностью этого списка в зависимости от состояния garbage collection.

Больше примеров, полное описание, а также обсуждение "Зачем *WeakMap*?" на странице справочника `WeakMap`.

Отметим, что `WeakMap`, в частности, может элегантно использоваться для упаковки приватных данных или деталей реализации. Следующий пример из статьи Nick Fitzgerald "Hiding Implementation Details with ECMAScript 6 WeakMaps". Приватная часть сохраняется как значение в `privates` и имеет время жизни такое же как и сущность класса. Сам класс и его методы публичны; прочее недоступно извне модуля:

```
JS
```

```
const privates = new WeakMap();

export class Public() {

  constructor() {
    const me = {
    // Приватные данные идут здесь
    };
    // 'me' будет освобождён вместе с 'this' !!!
    privates.set(this, me);
  }

  method () {
    const me = privates.get(this);
    // Сделайте что-нибудь с приватными данными в 'me'...
  }
}
```

Коллекции

Тип Set

`Set` реализация коллекции - структура данных, которая содержит список уникальных элементов в порядке их добавления.

В следующем примере приведены основные операции по работе с коллекцией `Set`:

```
JS
```

```
var mySet = new Set();
mySet.add(1);
mySet.add("some text");
mySet.add("foo");

mySet.has(1); // true
mySet.delete("foo");
mySet.size; // 2

for (let item of mySet) console.log(item);
// 1
// "some text"
```

Больше примеров и полное описание на странице справочника `Set`

Преобразования между Array и Set

Можно создать `Array` из Set с помощью `Array.from` или используя `spread operator`.

В свою очередь, конструктор `Set` может принимать `Array` в качестве аргумента.

> Примечание: Поскольку `Set` структура работает с уникальными значениями, любые повторяющиеся элементы из Array будут проигнорированы.

JS

```
Array.from(mySet);
[...mySet2];

mySet2 = new Set([1, 2, 3, 4]);
```

Сравнение Array и Set

Словари, как специализированная структура данных, имеют существенные отличия по сравнению с массивами:

- `Set.has` работает быстрее чем `Array.indexOf` .
- можно удалять элементы по значению (а не по индексу как массивах).
- `NaN` обрабатывается корректно.

- поддерживается уникальность значений.

Тип WeakSet

`WeakSet` это специальный вид коллекции, элементами которой могут быть только объекты. Ссылки на эти объекты в WeakSet являются *слабыми* (не учитываются сборщиком мусора (garbage collector GC)).

> Примечание: Элементы `WeakSet` уникальны и могут быть добавлены только один раз, также как и в `Set`.

Основные отличия от `Set`:

- `WeakSet` это коллекция *объектов* (примитивные значения не могут быть добавлены).
- `WeakSet` *нельзя итерировать*. А также нельзя получить список (итератор) элементов.

Использование `WeakSet` достаточно специфическое. Пользуясь тем, что они не могут создавать утечек памяти, в них можно, например, безопасно помещать ссылки на DOM-элементы.

Больше примеров и полное описание на странице справочника `WeakSet`

Проверка на равенство в Map и Set

Сравнение на равенство ключей в `Map` objects или объектов в `Set` основано на "same-value-zero algorithm ":

- алгоритм сравнения в целом совпадает с оператором `===`.
- `-0` и `+0` считаются равными (в отличие от `===`).
- `NaN` считается равным самому себе (в отличие от `===`).

This page was last modified on 7 авг. 2023 г. by MDN contributors.