# Map, Set

## Map

## Map – это коллекция ключ/значение, как и Object. Но основное отличие в том, что Map позволяет использовать ключи любого типа.

## *Методы и свойства*:

## new Map([iterable]) – создаёт коллекцию, можно указать перебираемый объект (обычно массив) из пар [ключ,значение] для инициализации.

## map.set(key, value) – записывает по ключу key значение value.

## map.get(key) – возвращает значение по ключу или undefined, если ключ key отсутствует.

## map.has(key) – возвращает true, если ключ key присутствует в коллекции, иначе false.

## map.delete(key) – удаляет элемент по ключу key.

## map.clear() – очищает коллекцию от всех элементов.

## map.size – возвращает текущее количество элементов.

## Примеры:

let map = **new Map**();

map.**set**("1", "str1"); // строка в качестве ключа

map.**set**(1, "num1"); // цифра как ключ

map.**set**(true, "bool1"); // булево значение как ключ

// помните, обычный объект Object приводит ключи к строкам?

// Map сохраняет тип ключей, так что в этом случае сохранится 2 разных значения:

alert(map.**get**(1)); // "num1"

alert(map.**get**("1")); // "str1"

alert(map.**size**); // 3

Map может использовать объекты в качестве ключей.

Каждый вызов map.set возвращает объект map, так что мы можем объединить вызовы в цепочку:

map.set("1", "str1")

.set(1, "num1")

.set(true, "bool1");

## Для *перебора коллекции Map* есть 3 метода:

## map.keys() – возвращает итерируемый объект по ключам,

## map.values() – возвращает итерируемый объект по значениям,

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

* map.forEach()

В отличие от обычных объектов Object, в Map перебор происходит в том же порядке, в каком происходило добавление элементов.

let recipeMap = **new Map**([

["огурец", 500],

["помидор", 350],

["лук", 50]

]);

// перебор по ключам (овощи)

for (let vegetable of recipeMap.**keys**()) {

alert(vegetable); // огурец, помидор, лук

}

// перебор по значениям (числа)

for (let amount of recipeMap.**values**()) {

alert(amount); // 500, 350, 50

}

// перебор по элементам в формате [ключ, значение]

for (let entry of recipeMap) { // то же самое, что и **recipeMap.entries**()

alert(entry); // огурец,500 (и так далее)

}

// выполняем функцию для каждой пары (ключ, значение)

recipeMap.**forEach**((value, key, map) => {

alert(`${key}: ${value}`); // огурец: 500 и так далее

});

**Пример викторины:**

const quiz = **new Map();**

quiz.**set**("вопрос", "Что означает JS?");

quiz.**set**(1, "Journey Start");

quiz.**set**(2, "Journey Script");

quiz.**set**(3, "Java Script");

quiz.**set**(4, "Java Slim");

quiz.**set**("correct", 3);

quiz.**set**(true, "Правильный ответ!");

quiz.**set**(false, "Неверный ответ!");

const question = quiz.**get**("вопрос");

console.log(question);

for **([key, val]** of quiz) {

if (typeof key === "number") {

console.log(key + " => " + val);

}

}

const answer = parseInt(prompt("Введите номер ответа:"));

console.log(answer);

if (quiz.**get**("correct") === answer) {

console.log(quiz.**get**(true));

} else {

console.log(quiz.**get**(false));

}

***Создание Map как массив, из объекта и наоборот***:

При создании Map мы можем указать **массив** (или другой итерируемый объект) с парами ключ-значение для инициализации, как здесь:

// массив пар [ключ, значение]

let map = new Map([

['1', 'str1'],

[1, 'num1'],

[true, 'bool1']

]);

alert( map.get('1') ); // str1

Если у нас уже есть **обычный объект**, и мы хотели бы создать Map из него, то поможет встроенный метод Object.entries(obj), который получает объект и возвращает массив пар ключ-значение для него, как раз в этом формате. Так что мы можем создать Map из обычного объекта следующим образом:

let obj = {

name: "John",

age: 30

};

let map = new Map(**Object.entries**(obj));

Здесь Object.entries возвращает массив пар ключ-значение: [ ["name","John"], ["age", 30] ].

Мы только что видели, как создать Map из обычного объекта при помощи Object.entries(obj).

Есть метод Object.fromEntries, который делает противоположное: получив массив пар вида [ключ, значение], он создаёт из них объект:

let prices = **Object.fromEntries(**[

['banana', 1],

['orange', 2],

['meat', 4]

]);

// prices = { banana: 1, orange: 2, meat: 4 }

let map = new Map();

map.set('banana', 1);

map.set('orange', 2);

map.set('meat', 4);

let obj = **Object.fromEntries(map.entries(**)); // создаём обычный объект (\*)

## Set

Объект Set – это особый вид коллекции: «множество» значений (без ключей), где каждое значение может появляться только один раз.

Его основные методы это:

* new Set(iterable) – создаёт Set, и если в качестве аргумента был предоставлен итерируемый объект (обычно это массив), то копирует его значения в новый Set.
* set.add(value) – добавляет значение (если оно уже есть, то ничего не делает), возвращает тот же объект set.
* set.delete(value) – удаляет значение, возвращает true, если value было в множестве на момент вызова, иначе false.
* set.has(value) – возвращает true, если значение присутствует в множестве, иначе false.
* set.clear() – удаляет все имеющиеся значения.
* set.size – возвращает количество элементов в множестве.

Основная «изюминка» – это то, что при повторных вызовах set.add() с одним и тем же значением ничего не происходит, за счёт этого как раз и получается, что каждое значение появляется один раз.

let set = new Set();

let john = { name: "John" };

let pete = { name: "Pete" };

let mary = { name: "Mary" };

// считаем гостей, некоторые приходят несколько раз

set.add(john);

set.add(pete);

set.add(mary);

set.add(john);

set.add(mary);

// set хранит только 3 уникальных значения

alert(set.size); // 3

for (let user of set) {

alert(user.name); // John (потом Pete и Mary)

}

**Перебор объекта Set**

Мы можем перебрать содержимое объекта set как с помощью метода for..of, так и используя forEach:

let set = new Set(["апельсин", "яблоко", "банан"]);

**for** (let value of set) alert(value);

// то же самое с forEach:

set.**forEach**((value, valueAgain, set) => {

alert(value);

});

Заметим забавную вещь. Функция в forEach у Set имеет 3 аргумента: значение value, потом снова то же самое значение valueAgain, и только потом целевой объект. Это действительно так, значение появляется в списке аргументов дважды.

Это сделано для совместимости с объектом Map, в котором колбэк forEach имеет 3 аргумента.

Set имеет те же встроенные методы, что и Map:

* set.values() – возвращает перебираемый объект для значений,
* set.keys() – то же самое, что и set.values(), присутствует для обратной совместимости с Map,
* set.entries() – возвращает перебираемый объект для пар вида [значение, значение], присутствует для обратной совместимости с Map.

**Перебор Map и Set** всегда осуществляется в порядке добавления элементов, так что нельзя сказать, что это – неупорядоченные коллекции, но поменять порядок элементов или получить элемент напрямую по его номеру нельзя.