МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Задание на разработку учебно-методического комплекса для электронного обучения с использованием современных компьютерных технологий**

**Анализ информационных технологий**

Работу выполнил

студент 309 группы

И.В. Дылько

Проверил преподаватель

канд. тех. наук,

доц. каф. инф. тех.

А.Н. Полетайкин

Краснодар

2022

**Предмет учебного курса:** «Использование и применение коллекций Map, Set, WeakMap, WeakSet в языке JavaScript».

**Цель изучения данного учебного курса:** формирование современных теоретических знаний, приобретение умений и навыков, позволяющих владеть на практике основными приемами использование и применение коллекций Map, Set, WeakMap, WeakSet в языке JavaScript.

**Задачи данного учебного курса:**

* Освоить базовые принципы работы с коллекциями Map, Set, WeakMap, WeakSet в языке JavaScript;
* Приобрести практические навыки взаимодействия с коллекциями Map, Set, WeakMap, WeakSet в языке JavaScript.

**Перечень знаний умений и навыков, приобретаемых при изучении курса:**

* Основные типы коллекций в языке JavaScript;
* Проектирование и создание объектов коллекций и взаимосвязи между ними;
* Углубленные знания по работе со структурами данных в JavaScript.

**Основные результаты деятельности обучения по данному курсу**:

* Углубленные знания по работе со структурами данных в JavaScript.
* Углубленные знания по работе со структурами данных в JavaScript.

**Основные дидактические единицы:**

|  |  |
| --- | --- |
| Раздел курса | Количество академических часов |
| Изучение существующих видов коллекций в языке JavaScript. | 3 |
| Изучение упрощенных видов коллекций на основе изученных в JavaScript. | 3 |

**Список рекомендуемых источников**

1. Современный учебник JavaScript – URL: https://learn.javascript.ru/

2. Справочник по JavaScript – URL: https://developer.mozilla.org/ru/docs/Web/JavaScript

3. Фримен Э., Робсон Э., Изучаем программирование на JavaScript: Питер, 2022 — 660 с.

4. К. Симпсон, You Don′t Know JS – Scope and Closures: O′Reilly, 2014. — 98 с.

5. К. Симпсон, You Don't Know JS: Up & Going. O′Reilly, 2015 — 88 с.

**О курсе**

|  |  |
| --- | --- |
| Раздел курса | Краткое содержание |
| Изучение существующих видов коллекций в языке JavaScript. | В данном разделе вы познакомитесь с коллекциями Map и Set, в которых добавление элементов всегда осуществляется в порядке LIFO, так что нельзя сказать, что это – неупорядоченные коллекции, но поменять порядок элементов или получить элемент напрямую по его номеру нельзя. |
| Изучение упрощенных видов коллекций на основе изученных в JavaScript. | WeakMap и WeakSet используются как вспомогательные структуры данных в дополнение к «основному» месту хранения объекта. Если объект удаляется из основного хранилища и нигде не используется, кроме как в качестве ключа в WeakMap или в WeakSet, то он будет удалён автоматически. Обе этих структуры данных не поддерживают методы и свойства, работающие со всем содержимым сразу или возвращающие информацию о размере коллекции. Возможны только операции на отдельном элементе коллекции. |

**Перечни**

**Перечень инструментальных средств**

1. ПК с ОС поддерживающий современные браузеры (на основе Chromium, Webkit, Quantum) или LTS версию Node.js;
2. Любой текстовый редактор;

**Перечень рекомендуемых источников**

1. Скринкаст по Node.js – URL: https://learn.javascript.ru/screencast/nodejs
2. Крокфорд Д., Как устроен JavaScript, 2019 — 304 с.

3 Бибо Б., Марас И., Резиг Д., Секреты JavaScript ниндзя, 2018 — 544 с.

**Формы итогового контроля**

Итоговый контроль достигается за счёт:

1. Контрольных вопросов после каждого раздела;
2. Задач для самостоятельного решения после каждого раздела;
3. Задач для самостоятельного решения после каждого раздела;
4. Проектного задания по прохождению всех материалов курса;
5. **Изучение объектов для хранения именованных коллекций и массивов для хранения упорядоченных коллекций в JavaScript и используя Map и 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() – создаёт коллекцию.

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 (пару «ключ/значение») по ключу key.

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

map.size – возвращает текущее количество элементов.

Например:

Изображение выглядит как текст

Автоматически созданное описание

Как мы видим, в отличие от объектов, ключи не были приведены к строкам. Можно использовать любые типы данных для ключей.

**map[key] это не совсем правильный способ использования Map**

Хотя map[key] также работает, например, мы можем установить map[key] = 2, в этом случаеmap рассматривался бы как обычный JavaScript объект, таким образом это ведёт ко всем соответствующим ограничениям (только строки/символьные ключи и так далее). Поэтому нам следует использовать методы map: set, get и так далее.

**Map может использовать объекты в качестве ключей.**

Например:

Изображение выглядит как текст

Автоматически созданное описание

Использование объектов в качестве ключей – одна из наиболее заметных и важных функций Map. Это то что невозможно для Object. Строка в качестве ключа в Object – это нормально, но мы не можем использовать другой Object в качестве ключа в Object.

Давайте попробуем заменить Map на Object:

Изображение выглядит как текст

Автоматически созданное описание

Так как visitsCountObj является объектом, он преобразует все ключи Object, такие как john и ben, в одну и ту же строку "[object Object]". Это определенно не то, чего мы хотим.

**Как объект Map сравнивает ключи**

Чтобы сравнивать ключи, объект Map использует алгоритм SameValueZero. Это почти такое же сравнение, что и ===, с той лишь разницей, что NaN считается равным NaN. Так что NaN также может использоваться в качестве ключа.

Этот алгоритм не может быть заменён или модифицирован.

**Как объект Map сравнивает ключи**

Чтобы сравнивать ключи, объект Map использует алгоритм SameValueZero. Это почти такое же сравнение, что и ===, с той лишь разницей, что NaN считается равным NaN. Так что NaN также может использоваться в качестве ключа.

**Перебор Map**

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.

Например:

Изображение выглядит как текст

Автоматически созданное описание

**Используется порядок вставки**

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

Кроме этого, Map имеет встроенный метод forEach, схожий со встроенным методом массивов Array:

Изображение выглядит как текст

Автоматически созданное описание

**Object.entries: Map из Object**

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

Изображение выглядит как текст

Автоматически созданное описание

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:

Изображение выглядит как текст

Автоматически созданное описание

Здесь Object.entries возвращает массив пар ключ-значение: [ ["name","John"], ["age", 30] ]. Это именно то, что нужно для создания Map.

**Object.fromEntries: Object из Map**

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:

Изображение выглядит как текст

Автоматически созданное описание

Мы можем использовать Object.fromEntries, чтобы получить обычный объект из Map.

К примеру, у нас данные в Map, но их нужно передать в сторонний код, который ожидает обычный объект.

Вот как это сделать:

Изображение выглядит как текст

Автоматически созданное описание

Вызов map.entries() возвращает итерируемый объект пар ключ/значение, как раз в нужном формате для Object.fromEntries.

Мы могли бы написать строку (\*) ещё короче:



Это то же самое, так как Object.fromEntries ожидает перебираемый объект в качестве аргумента, не обязательно массив. А перебор map как раз возвращает пары ключ/значение, так же, как и map.entries(). Так что в итоге у нас будет обычный объект с теми же ключами/значениями, что и в map.

**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.

Например, мы ожидаем посетителей, и нам необходимо составить их список. Но повторные визиты не должны приводить к дубликатам. Каждый посетитель должен появиться в списке только один раз.

Множество Set – как раз то, что нужно для этого:

Изображение выглядит как текст

Автоматически созданное описание

Альтернативой множеству Set может выступать массив для хранения гостей и дополнительный код для проверки уже имеющегося элемента с помощью arr.find. Но в этом случае будет хуже производительность, потому что arr.find проходит весь массив для проверки наличия элемента. Множество Set лучше оптимизировано для добавлений, оно автоматически проверяет на уникальность.

**Перебор объекта Set**

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

Изображение выглядит как текст

Автоматически созданное описание

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. Выглядит немного странно, но в некоторых случаях может помочь легко заменить Map на Set и наоборот.

Set имеет те же встроенные методы, что и Map:

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

**Итого**

Map – коллекция пар ключ-значение.

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.

Отличия от обычного объекта Object:

* Что угодно может быть ключом, в том числе и объекты.
* Есть дополнительные методы, свойство size.

Set – коллекция уникальных значений, так называемое «множество».

Методы и свойства:

* new Set(iterable) – создаёт Set, можно указать перебираемый объект со значениями для инициализации.
* 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.

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.

**Контрольные вопросы**

1. Почему Flask – микрофреймворк?
2. Как называется функция создания ответа в функции-представлении?
3. С помощью какой функции можно вызывать отрисовку шаблона страницы?
4. Какой декоратор используется для представлений-обработчиков ошибок?
5. Какая функция позволяет возвращать ответы в формате application/json?

**Упражнения для самостоятельной работы**

1. Создать представления для работы с ошибками или перенаправлениями.

**Решение.**

В файле сделаем нужные импорты и допишем следующий код:

from flask import Flask, render\_template, redirect, abort, url\_for

from pages\_blueprint import page\_views

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

    return redirect(url\_for('picture'))

@app.errorhandler(404)

def error\_handler(error):

    return 'Страницы не существует!', 404

@app.route('/unknown')

def unknown():

    abort(404)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(host='0.0.0.0', port=1234)

1. Создать вложенную структуру шаблонов страниц и подключить их к приложению. Главный шаблон, затем от него наследовать следующий шаблон, и от него наследовать конечную страницу.
2. Создать функцию-представление, которое возвращает ответ в формате application/json.

**2 Использование объектов для хранения массивов и именованных коллекций для хранения коллекций в JavaScript WeakMap и WeakSet**

**WeakMap и WeakSet**

Как мы знаем из главы Сборка мусора, движок JavaScript хранит значения в памяти до тех пор, пока они достижимы (то есть, эти значения могут быть использованы). Например:

Изображение выглядит как текст

Автоматически созданное описание

Обычно свойства объекта, элементы массива или другой структуры данных считаются достижимыми и сохраняются в памяти до тех пор, пока эта структура данных содержится в памяти.

Например, если мы поместим объект в массив, то до тех пор, пока массив существует, объект также будет существовать в памяти, несмотря на то, что других ссылок на него нет.

Например: Изображение выглядит как текст

Автоматически созданное описание

Аналогично, если мы используем объект как ключ в Map, то до тех пор, пока существует Map, также будет существовать и этот объект. Он занимает место в памяти и не может быть удалён сборщиком мусора.

Например:

Изображение выглядит как текст

Автоматически созданное описание

WeakMap – принципиально другая структура в этом аспекте. Она не предотвращает удаление объектов сборщиком мусора, когда эти объекты выступают в качестве ключей.

Давайте посмотрим, что это означает, на примерах.

**WeakMap**

Первое его отличие от Map в том, что ключи в WeakMap должны быть объектами, а не примитивными значениями:

Изображение выглядит как текст

Автоматически созданное описание

Теперь, если мы используем объект в качестве ключа и если больше нет ссылок на этот объект, то он будет удалён из памяти (и из объекта WeakMap) автоматически. Давайте создадим таблицу дисциплин, читаемых в вузе:

Изображение выглядит как текст

Автоматически созданное описание

Сравните это поведение с поведением обычного Map, пример которого был приведён ранее. Теперь john существует только как ключ в WeakMap и может быть удалён оттуда автоматически.

WeakMap не поддерживает перебор и методы keys(), values(), entries(), так что нет способа взять все ключи или значения из неё.

В WeakMap присутствуют только следующие методы:

* weakMap.get(key)
* weakMap.set(key, value)
* weakMap.delete(key)
* weakMap.has(key)

К чему такие ограничения? Из-за особенностей технической реализации. Если объект станет недостижим (как объект john в примере выше), то он будет автоматически удалён сборщиком мусора. Но нет информации, в какой момент произойдёт эта очистка.

Решение о том, когда делать сборку мусора, принимает движок JavaScript. Он может посчитать необходимым как удалить объект прямо сейчас, так и отложить эту операцию, чтобы удалить большее количество объектов за раз позже. Так что технически количество элементов в коллекции WeakMap неизвестно. Движок может произвести очистку сразу или потом, или сделать это частично. По этой причине методы для доступа ко всем сразу ключам/значениям недоступны.

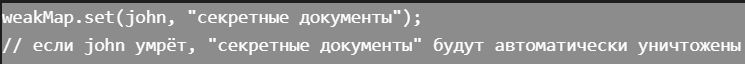
Но для чего же нам нужна такая структура данных?

**Пример: дополнительные данные**

В основном, WeakMap используется в качестве дополнительного хранилища данных.

Если мы работаем с объектом, который «принадлежит» другому коду, может быть даже сторонней библиотеке, и хотим сохранить у себя какие-то данные для него, которые должны существовать лишь пока существует этот объект, то WeakMap – как раз то, что нужно.

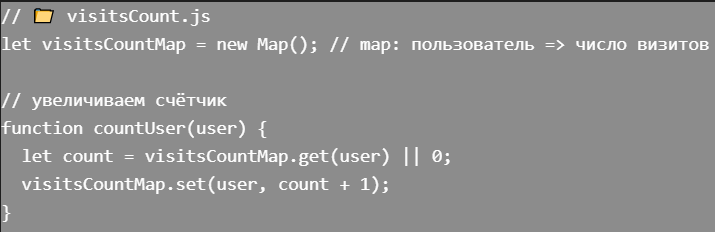
Мы кладём эти данные в WeakMap, используя объект как ключ, и когда сборщик мусора удалит объекты из памяти, ассоциированные с ними данные тоже автоматически исчезнут.



Давайте рассмотрим один пример.

Предположим, у нас есть код, который ведёт учёт посещений для пользователей. Информация хранится в коллекции Map: объект, представляющий пользователя, является ключом, а количество визитов – значением. Когда пользователь нас покидает (его объект удаляется сборщиком мусора), то больше нет смысла хранить соответствующий счётчик посещений.

Вот пример реализации счётчика посещений с использованием Map:



А вот другая часть кода, возможно, в другом файле, которая использует countUser:

Изображение выглядит как текст

Автоматически созданное описание

Теперь объект john должен быть удалён сборщиком мусора, но он продолжает оставаться в памяти, так как является ключом в visitsCountMap.

Нам нужно очищать visitsCountMap при удалении объекта пользователя, иначе коллекция будет бесконечно расти. Подобная очистка может быть неудобна в реализации при сложной архитектуре приложения.

Проблемы можно избежать, если использовать WeakMap:

Изображение выглядит как текст

Автоматически созданное описание

Теперь нет необходимости вручную очищать visitsCountMap. После того, как объект john стал недостижим другими способами, кроме как через WeakMap, он удаляется из памяти вместе с информацией по такому ключу из WeakMap.

**Применение для кеширования**

Другая частая сфера применения – это кеширование, когда результат вызова функции должен где-то запоминаться («кешироваться») для того, чтобы дальнейшие её вызовы на том же объекте могли просто брать уже готовый результат, повторно используя его.

Для хранения результатов мы можем использовать Map, вот так:

Изображение выглядит как текст

Автоматически созданное описание

Многократные вызовы process(obj) с тем же самым объектом в качестве аргумента ведут к тому, что результат вычисляется только в первый раз, а затем последующие вызовы берут его из кеша. Недостатком является то, что необходимо вручную очищать cache от ставших ненужными объектов.

Но если мы будем использовать WeakMap вместо Map, то эта проблема исчезнет: закешированные результаты будут автоматически удалены из памяти сборщиком мусора.

Изображение выглядит как текст

Автоматически созданное описание

**WeakSet**

Коллекция WeakSet ведёт себя похоже:

* Она аналогична Set, но мы можем добавлять в WeakSet только объекты (не примитивные значения).
* Объект присутствует в множестве только до тех пор, пока доступен где-то ещё.
* Как и Set, она поддерживает add, has и delete, но не size, keys() и не является перебираемой.

Будучи «слабой» версией оригинальной структуры данных, она тоже служит в качестве дополнительного хранилища. Но не для произвольных данных, а скорее для значений типа «да/нет». Присутствие во множестве WeakSet может что-то сказать нам об объекте.

Например, мы можем добавлять пользователей в WeakSet для учёта тех, кто посещал наш сайт:

Изображение выглядит как текст

Автоматически созданное описание

Наиболее значительным ограничением WeakMap и WeakSet является то, что их нельзя перебрать или взять всё содержимое. Это может доставлять неудобства, но не мешает WeakMap/WeakSet выполнять их главную задачу – быть дополнительным хранилищем данных для объектов, управляемых из каких-то других мест в коде.

**Итого**

WeakMap – это Map-подобная коллекция, позволяющая использовать в качестве ключей только объекты, и автоматически удаляющая их вместе с соответствующими значениями, как только они становятся недостижимыми иными путями.

WeakSet – это Set-подобная коллекция, которая хранит только объекты и удаляет их, как только они становятся недостижимыми иными путями.

Обе этих структуры данных не поддерживают методы и свойства, работающие со всем содержимым сразу или возвращающие информацию о размере коллекции. Возможны только операции на отдельном элементе коллекции.

WeakMap и WeakSet используются как вспомогательные структуры данных в дополнение к «основному» месту хранения объекта. Если объект удаляется из основного хранилища и нигде не используется, кроме как в качестве ключа в WeakMap или в WeakSet, то он будет удалён автоматически.

**Контрольные вопросы**

1. Какой параметр приложения отвечает за влключение/выключение режима отладки?
2. Какой параметр приложения нужен для задания секретного ключа?
3. Какой параметр приложения содержит путь к настроенному приложению Flask?
4. Что такое blueprint?
5. Какие есть рекомендации по архитектуре приложения Flask?

**Упражнения для самостоятельной работы**

1. Создать параметризированные представления и зарегистрировать их в отдельном blueprint-е.

**Решение**

Создадим в корне приложения файл pages\_blueprint.py следующего содержания:

from flask import Blueprint, render\_template

page\_views = Blueprint('pages\_views', \_\_name\_\_, template\_folder='templates', static\_folder='static')

@page\_views.route('/user/<username>', methods=['POST', 'GET'])

def user\_view(username):

    context = {'string': f'Пользователь {username}' }

    return render\_template('text\_template.html', context=context)

@page\_views.route('/page/<int:page\_id>')

def page\_view(page\_id):

    context = {'string': f'Страница {page\_id}' }

    return render\_template('text\_template.html', context=context)

В главном файле приложения (server.py) напишем следующий код:

from flask import Flask

from pages\_blueprint import page\_views

app = Flask(\_\_name\_\_)

app.register\_blueprint(page\_views)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(host='0.0.0.0', port=1234)

1. Создать и зарегистрировать 2 blueprint-а, а также подключить их к основному приложению.
2. Создать blueprint, который полностью отвечает за получение данных.

**Комплекс тестовых заданий**

**1. Выберите правильное определение Map:**

a) это Map-подобная коллекция, позволяющая использовать в качестве ключей только объекты, и автоматически удаляющая их вместе с соответствующими значениями, как только они становятся недостижимыми иными путями.

b) коллекция пар ключ-значение.

c) За отрисовку шаблонов в Flask отвечает функция render\_template.

d) это Set-подобная коллекция, которая хранит только объекты и удаляет их, как только они становятся недостижимыми иными путями.

**2. Дайте определение WeakMap:**

a) это Set-подобная коллекция, которая хранит только объекты и удаляет их, как только они становятся недостижимыми иными путями.

b) это Map-подобная коллекция, позволяющая использовать в качестве ключей только объекты, и автоматически удаляющая их вместе с соответствующими значениями, как только они становятся недостижимыми иными путями.

c) коллекция пар ключ-значение.

**3** **Методы с которыми работают коллекция Set**



**4. Движок JavaScript хранит значения в памяти до тех пор, пока они достижимы (то есть, эти значения могут быть использованы)?**

1. Да
2. Нет
3. Только простые объекты

**5. Если мы используем объект как ключ в WeakMap, то до тех пор, пока существует WeakMap, будет ли существовать этот объект?**

1. Да
2. нет

**6. Укажите строчку с неверным использованием коллекции weakMap:**

Изображение выглядит как текст

Автоматически созданное описание

**weakMap.set("test", "Whoops");**

**7. Как ведёт себя коллекция WeakSet?**

a) Она полностью аналогична Set, но мы можем добавлять в WeakSet только примитивные значения.

b) Объект присутствует в множестве только до тех пор, пока доступен где-то ещё.

c) Как и Set, она поддерживает методы add, has, delete, size, keys().

**8. Что вернёт вызов alert()?**

Изображение выглядит как текст

Автоматически созданное описание

a) bool1.

b) num1

c) str1

**9** **Что вернёт вызов alert()?**

Изображение выглядит как текст

Автоматически созданное описание

a) 30

b) John

c) name

**10** **Выберете корректную информацию о WeakMap:**

1. ключи в WeakMap должны быть примитивными значениями, а не объектами
2. удаление ссылки на объект не удалит информацию из объекта WeakMap
3. WeakMap поддерживает перебор и методы keys(), values(), entries()
4. ключи в WeakMap должны быть объектами, а не примитивными значениями
5. в WeakMap отсутствует метод .has

**11. Какой метод отсутствует у коллекции Set**

1. add(value)
2. get(value)
3. delete(value)
4. has(value)
5. clear()

**12. Мы хотели бы получить массив ключей map.keys() в переменную и далее работать с ними, например, применить метод .push.**

**Но это не выходит:**

**Почему? Что нужно поправить в коде, чтобы вызов keys.push сработал?**

**Ответ должен быть адресован к 5 строке после знака = (без пробелов):**

Изображение выглядит как текст

Автоматически созданное описание

Array.from(map.keys());

**13. Напишите вызов функции map.get("вставить ответ") для возвращения "num1"**

Изображение выглядит как текст

Автоматически созданное описание**:**

**map.get(1)**

**14. Из созданной коллекции необходимо удалить секретные документы напишите команду без пробелов, которая сделает это.**

Изображение выглядит как текст

Автоматически созданное описание

**john=null;**

**15. Что вернёт вызов alert для следующего кода?**

Изображение выглядит как текст

Автоматически созданное описание**:**

1. 123
2. John
3. {}

**16. Выберете корректные варианты обращения к Object.fromEntries**

Изображение выглядит как текст

Автоматически созданное описание**:**

a) prices.orange

b) prices.meat

c) prices.[“4”]

d) prices.2

**17. Выберите правильное определение Set:**

а) коллекция уникальных значений, так называемое «множество»

b) коллекция пар ключ-значение.

c) это Set-подобная коллекция, которая хранит только объекты и удаляет их, как только они становятся недостижимыми иными путями.

d) это Map-подобная коллекция, позволяющая использовать в качестве ключей только объекты, и автоматически удаляющая их вместе с соответствующими значениями, как только они становятся недостижимыми иными путями.

**18. Выберите правильное определение WeakSet**

a) это Map-подобная коллекция, позволяющая использовать в качестве ключей только объекты, и автоматически удаляющая их вместе с соответствующими значениями, как только они становятся недостижимыми иными путями.

b) это Set-подобная коллекция, которая хранит только объекты и удаляет их, как только они становятся недостижимыми иными путями.

c) коллекция пар ключ-значение.

**19** **Выберете корректные варианта заполнения коллекции WeakSet**

Изображение выглядит как текст

Автоматически созданное описание**:**

1. visitedSet.add(pete);
2. visitedSet.add(john);
3. visitedSet.add("Mary");
4. visitedSet.add(john);
5. visitedSet.add(john.name);

**20. Отличие WeakMap от Map в том, что ключи должны быть примитивными значениями , а не объектами**

a) Нет

b) да

**Количество баллов за задание 1 – 5**

**Макс. балл – 36.**

**Комплекс практических заданий для проверки умений и навыков**

**Задание №1.** Фильтрация уникальных элементов массива Допустим, у нас есть массив arr.

Создайте функцию unique(arr), которая вернёт массив уникальных, не повторяющихся значений массива arr.

Например:

Изображение выглядит как текст

Автоматически созданное описание

P.S. Здесь мы используем строки, но значения могут быть любого типа.

P.P.S. Используйте Set для хранения уникальных значений.

**Критерии оценивания:**

* Вычисление совместимости происходит на стороне сервера и возвращается на страницу – 3б.
* Вычисление происходит прямо на странице – 1 б.

Макс. балл – 3.

**Задание №2.** Отфильтруйте анаграммы Анаграммы – это слова, у которых те же буквы в том же количестве, но они располагаются в другом порядке.

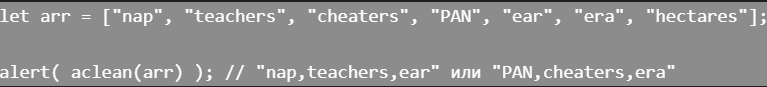
Например:

Изображение выглядит как текст

Автоматически созданное описание

Напишите функцию aclean(arr), которая возвращает массив слов, очищенный от анаграмм.

Например:



Из каждой группы анаграмм должно остаться только одно слово, не важно какое.

**Критерии оценивания:**

* Пользователь видит только свои запросы – 1 б.
* Пользователь видит глобальные запросы – 4 б.

Макс. балл – 4.

**Задание №3**. Хранение отметок "не прочитано"

Есть массив сообщений:

Изображение выглядит как текст

Автоматически созданное описание

У вас есть к ним доступ, но управление этим массивом происходит где-то ещё. Добавляются новые сообщения и удаляются старые, и вы не знаете в какой момент это может произойти.

Имея такую вводную информацию, решите, какую структуру данных вы могли бы использовать для ответа на вопрос «было ли сообщение прочитано?». Структура должна быть подходящей, чтобы можно было однозначно сказать, было ли прочитано это сообщение для каждого объекта сообщения.

P.S. Когда сообщение удаляется из массива messages, оно должно также исчезать из структуры данных.

P.P.S. Нам не следует модифицировать сами объекты сообщений, добавлять туда свойства. Если сообщения принадлежат какому-то другому коду, то это может привести к плохим последствиям.

**Критерии оценивания:**

Система отслеживает правильность ввода цены: целое число – 1 б.

Система реализует хранение товаров, но не может возвращать информацию по номеру чека – 2 б.

Система реализует хранение товаров и возвращает информацию по номеру чека – 3б.

Макс балл – 4.

**Web-страничка курса доступна по ссылке**: