WeakMap в JavaScript

`WeakMap` - это структура данных в JavaScript, аналогичная `Map`, которая представляет собой коллекцию ключ-значение. Основное отличие `WeakMap` заключается в том, что ключи в `WeakMap` могут быть только объектами, и эти объекты являются слабыми (weak), что означает, что они не удерживаются в памяти и могут быть автоматически удалены сборщиком мусора, если на них больше нет ссылок.

Основные характеристики WeakMap:

1. **Только объекты в качестве ключей**: В `WeakMap` ключами могут быть только объекты, и попытка использования другого типа данных в качестве ключа вызовет ошибку.
2. **Слабые ссылки**: Объекты-ключи в `WeakMap` являются слабыми, что означает, что они не удерживаются в памяти. Если на объект-ключ больше нет ссылок, он будет автоматически удален сборщиком мусора.
3. **Не перечисляемы**: Ключи в `WeakMap` не могут быть перечислены, что означает, что нет методов для получения всех ключей или значений.

Основные методы WeakMap:

- `set(key, value)`: Добавляет пару ключ-значение в `WeakMap`.
- `get(key)`: Возвращает значение, связанное с указанным ключом.
- `has(key)`: Проверяет наличие ключа в `WeakMap`.
- `delete(key)`: Удаляет пару ключ-значение по ключу.

`WeakMap` полезен, когда вам нужно хранить дополнительные данные, связанные с объектами, и при этом не создавать для них ссылок, которые могут мешать сборке мусора. Он часто используется внутри библиотек и фреймворков для управления приватными данными.

Пример использования WeakMap:

```
const weakMap = new WeakMap();

const keyObj = {};
const valueObj = { data: "This is a value object" };

weakMap.set(keyObj, valueObj);

console.log(weakMap.get(keyObj)); // { data: "This is a value object" }

console.log(weakMap.has(keyObj)); // true

weakMap.delete(keyObj);
console.log(weakMap.has(keyObj)); // false
```

Избегание утечек памяти:

```
const weakMap = new WeakMap();

let key = { id: 1 };
weakMap.set(key, 'Value');

key = null; // Удаляем ссылку на ключ
// Сборщик мусора может удалить слабую ссылку на ключ
```

В этом примере, когда переменная `key` устанавливается в `null`, нет больше сильных ссылок на объект `{ id: 1 }`, и сборщик мусора может удалить его из `WeakMap`.

Метаданные для DOM-узлов:

`WeakMap` может использоваться для хранения метаданных для DOM-узлов, не удерживая сильных ссылок на сами узлы, что помогает избежать утечек памяти.

```
const nodeData = new WeakMap();

const element = document.getElementById('myElement');
nodeData.set(element, { meta: 'some metadata' });

// Гарантирует, что метаданные не удерживают ссылку на элемент
element.parentNode.removeChild(element);
```

В этом примере `nodeData` используется для хранения метаданных для DOM-узла, и когда элемент удаляется из DOM, нет больше сильных ссылок на него, и он может быть собран сборщиком мусора.