# JavaScript WeakMap

❮ Previous Next ❯

## The WeakMap Object

A JavaScript **WeakMap** is a collection of key/value pairs where the **keys must be objects**.

A WeakMap holds **weak references** to its keys.

### Example

```
// Create a WeakMap
let myMap = new WeakMap();

// Create an Object
let myObj = {fname:"John", lname:"Doe"};

// Set a WeakMap value
myMap.set(myObj, "player");

// Get the WeakMap value
let type = myMap.get(myObj);
```

Try it Yourself »

## Garbage Collection

JavaScript employs a memory management mechanism known as **Garbage Collection**.

The primary functions are:

- Ensuring efficient use of memory resources
- Reclaim memory occupied by variables that are no longer in use
- Preventing memory leaks

## Weak References

Unlike a regular Map, a WeakMap does not prevent its keys from being garbage collected.

If a key (an object) has no references to it in a program, it becomes eligible for garbage collection.

When the key is garbage collected, its key-value pair is removed from the WeakMap.

### Example

```
let myMap = new WeakMap();
let myObj = {fname:"John", lname:"Doe"};

myMap.set(myObj, "secret");
myObj = null;
// now myObj (and its values) in myMap can be garbage collected
```

## Keys Must Be Objects

Primitive values cannot be used as keys in a WeakMap.

The **keys must be objects** or non-registered symbols.

This restriction is tied to the garbage collection mechanism; primitives are not garbage collected in the same way as objects.

## Tracking Objects

The entries in a WeakMap are weakly held: if the object key becomes unreachable, its mapping is removed automatically.

This is perfect for tracking data about objects without preventing garbage collection.

### Tracking Visitors

```
let text = "";

// Create a WeakMap to store visit counts
const visitsCount = new WeakMap();

// Create Visitor Objects
const John = {name:"John", age:40};
const Paul = {name:"Paul", age:41};
const Ringo = {name:"Ringo", age:42};
const George = {name:"George", age:43};

// Track visits
track(Paul);
track(Ringo);
track(Paul);
track(Paul);
track(John);

// Function to track visitors
function track(visitor) {
  let count = visitsCount.get(visitor) || 0;
  count++;
  visitsCount.set(visitor, count);
  text += visitor.name + ", age " + visitor.age + ", has visited " + count + "
time(s).<br>";
}
```

Try it Yourself »

## Automatic Cleanup

If you remove all references to a visitor object:

### Tracking Visitors:

```
John = null;

// The entry for John is now removed from the WeakMap (persons)
```

## Not Iterable

WeakMaps are **not enumerable**.

You **cannot iterate** over the keys and values with for loops, forEach(), or keys().

You cannot access the size.

## Limited Methods

WeakMap provides a limited set of methods:

| | |
|---|---|
| new WeakMap() | Creates a new WeakMap object |
| get(key) | Gets the value for a key in a WeakMap |
| set(key, value) | Sets the value for a key in a WeakMap |
| delete(key) | Removes an element specified by a key |
| has(key) | Returns true if a key exists in a WeakMap |

## Weak Map Secret Data

### Example

```
// Create WeakMap
const myMap = new WeakMap();

// Private Fields Simulation
class User {
  constructor(name) {
  myMap.set(this, {secret:"hidden data"});
  this.name = name;
  }
  getSecret() {
  return myMap.get(this).secret;
  }
}

const user1 = new User("John");
secret = user1.getSecret();
```

Try it Yourself »

### Example Explained

A WeakMap does not allow iteration.

Outside code can not "discover" what objects are stored inside a WeakMap.

To get the secret, you need the **this** reference that was used in the constructor.

External code has access to user1 and myMap, bot not to the **this** reference inside myMap, unless you explicitly expose it, like via getSecret(), the secret value is unreachable.

## Privacy

WeakMap was intentionally designed for privacy: you can set, get, has, and delete using an object key, but not inspect what is inside.

This was a great tool for simulating private properties in JavaScript classes (before #private fields were added to the language).

## Learn More:

JavaScript Maps

JavaScript Map Methods

JavaScript Map Reference

JavaScript Sets

## Browser Support

`WeakMap` is an ES6 feature.

ES6 is fully supported in all modern browsers since June 2017:

| Chrome | Edge | Firefox | Safari | Opera |
|---|---|---|---|---|
| 51 | 15 | 54 | 10 | 38 |
| May 2016 | Apr 2017 | Jun 2017 | Sep 2016 | Jun 2016 |

❮ Previous Sign in to track progress Next ❯