JavaScript WeakMap

Back to JS page

Table of Contents

- JavaScript WeakMap
  - The WeakMap Object
    - Example 1
  - Keys Must Be Objects
    - Example 2
  - Weak References
    - Example 3
  - Not Iterable and Limited Methods
    - Example 4
  - WeakMap Secret Data
    - Example 5
  - Document
  - Reference

The WeakMap Object

A `WeakMap` is a collection of key/value pairs where the keys must be **objects** (not primitives). Values can be any type:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript WeakMap</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>The WeakMap Object</h4>
<p id="demo"></p>

<script>
// WeakMap keys must be objects
const obj1 = { name: "John" };
const obj2 = { name: "Jane" };

const weakMap = new WeakMap();
weakMap.set(obj1, "Engineer");
weakMap.set(obj2, "Doctor");

document.getElementById("demo").innerHTML =
  "WeakMap size (no size property)<br>" +
  "get(obj1): " + weakMap.get(obj1) + "<br>" +
  "get(obj2): " + weakMap.get(obj2) + "<br><br>" +
  "WeakMap keys must be objects";
</script>

</body>
</html>
```

Example 1

Result View Example

Keys Must Be Objects

If you try to use a primitive value (string, number, etc.) as a key in a WeakMap, it causes an error:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript WeakMap</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Keys Must Be Objects</h4>
<p id="demo"></p>

<script>
const weakMap = new WeakMap();

// Keys must be objects
const obj = { id: 1 };
weakMap.set(obj, "Valid");

let text = "set(obj, 'Valid'): OK (object key)<br><br>";

// This would cause an error: Invalid value used as weak map key
try {
  weakMap.set("name", "John");  // TypeError!
  text += "set('name', 'John'): OK?";
} catch(err) {
  text += "set('name', 'John'): ERROR - " + err.message + "<br><br>";
  text += "WeakMap keys MUST be objects";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 2

Result View Example

Weak References

WeakMap holds keys by **weak references**. When a key object is no longer referenced elsewhere, it can be garbage collected:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript WeakMap</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Weak References</h4>
<p id="demo"></p>

<script>
const weakMap = new WeakMap();

let person = { name: "John" };
weakMap.set(person, "Engineer");

let text = "Person object added to WeakMap<br>";
text += "get(person): " + weakMap.get(person) + "<br><br>";

// Remove all references to the object
person = null;

text += "After person = null:<br>";
text += "The object can be garbage collected<br>";
text += "It will be automatically removed from WeakMap<br><br>";
text += "This automatic cleanup is the key benefit<br>";
text += "of WeakMap for memory management";

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 3

Result View Example

Not Iterable and Limited Methods

A WeakMap is **not iterable**. It only has four methods: `set()`, `get()`, `delete()`, and `has()`:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript WeakMap</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Not Iterable and Limited Methods</h4>
<p id="demo"></p>

<script>
const weakMap = new WeakMap();
const obj = { id: 1 };

// The 4 available methods
weakMap.set(obj, "value");
let text = "After set(obj, 'value'):<br>";
text += "get(obj): " + weakMap.get(obj) + "<br>";
text += "has(obj): " + weakMap.has(obj) + "<br><br>";

weakMap.delete(obj);
text += "After delete(obj):<br>";
text += "has(obj): " + weakMap.has(obj) + "<br><br>";

text += "<b>WeakMap limitations:</b><br>";
text += "1. NOT iterable (no forEach, keys, values)<br>";
text += "2. NO size property<br>";
text += "3. Only 4 methods: set(), get(), delete(), has()";

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 4

Result View Example

WeakMap Secret Data

WeakMap can store private data associated with objects without exposing it:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript WeakMap</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>WeakMap Secret Data</h4>
<p id="demo"></p>

<script>
// Store secret data in a WeakMap
const secrets = new WeakMap();

function Person(name, age, secret) {
  this.name = name;
  this.age = age;
  // Store secret data in WeakMap (not directly on object)
  secrets.set(this, secret);
}

const person1 = new Person("John", 30, "secret code: 1234");
const person2 = new Person("Jane", 25, "secret password: abc");

let text = "Person 1: " + person1.name + ", " + person1.age + "<br>";
text += "Person 2: " + person2.name + ", " + person2.age + "<br><br>";

// Access secret data via WeakMap
text += "Secret 1 (via WeakMap): " + secrets.get(person1) + "<br>";
text += "Secret 2 (via WeakMap): " + secrets.get(person2) + "<br><br>";

// Secrets are NOT visible directly on the object
text += "person1.secret: " + person1.secret + " (undefined - private!)<br>";

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 5

Result View Example

Document

Document in project

You can Download PDF file.

Reference

- W3Schools JavaScript WeakMap