[JavaScript] □□□□□

□□□□□ WeakXXX □□□□□□□□□□□□□□□□□□□□□□

-
  - The JavaScript engine may hold references to things which look like they are unreachable (e.g., in closures, or inline caches).
  - Different JavaScript engines may do these things differently, or the same engine may change its algorithms across versions.

WeakRef

> WeakRef □□□ GC □□

WeakMap && WeakSet

> □□□□□□□ WeakMap □□□
> WeakMap □□□

□□□□□□□□□ Map □□value □□□□□ key □□□□□□□ + 1□□□ WeakMap □□key □□ WeakRef □□□□□□□□□□ value □□□ + 1□

WeakMap □□□□□□□□□□□□□□□□DOM□□□key□□□□value□□□□□□□□□□□□□□□□DOM□□□□□□□□□ WeakMap □□□□□□□□□□GC□□□□□□□□□□□
□□ node --expose-gc[1]

WeakSet□□□

```
class IterableWeakMap {
<a class='tag' href="/tags/weakMap">#weakMap</a> = new WeakMap();
<a class='tag' href="/tags/refSet">#refSet</a> = new Set();
// FinalizationRegistry □□□□□□□ key □ GC □□□□ cleanUp □□ refSet □□□□
// @DOC: new FinalizationRegistry((heldValue) => { /*...*/ })
//          register(target, heldValue, ?unregisterToken)
//          unregister(unregisterToken)
<a class='tag' href="/tags/finalizationGroup">#finalizationGroup</a> = new Finalizat

static #cleanup({ set, ref }) {
set.delete(ref);
}

constructor(iterable) {
for (const [key, value] of iterable) {
this.set(key, value);
}
}

set(key, value) {
const ref = new WeakRef(key);

this.<a class='tag' href="/tags/weakMap">#weakMap</a>.set(key, { value, ref });
this.<a class='tag' href="/tags/refSet">#refSet</a>.add(ref);
this.<a class='tag' href="/tags/finalizationGroup">#finalizationGroup</a>.register(ke
set: this.<a class='tag' href="/tags/refSet">#refSet</a>,
ref
}, ref);
}

get(key) {
const entry = this.<a class='tag' href="/tags/weakMap">#weakMap</a>.get(key);
return entry && entry.value;
}

delete(key) {
const entry = this.<a class='tag' href="/tags/weakMap">#weakMap</a>.get(key);
if (!entry) {
return false;
}
```

```
this.<a class='tag' href="/tags/weakMap">#weakMap</a>.delete(key);
this.<a class='tag' href="/tags/refSet">#refSet</a>.delete(entry.ref);
this.<a class='tag' href="/tags/finalizationGroup">#finalizationGroup</a>.unregister(
return true;
}

*[Symbol.iterator]() {
for (const ref of this.#refSet) {
const key = ref.deref();
if (!key) continue;
const { value } = this.<a class='tag' href="/tags/weakMap">#weakMap</a>.get(key);
yield [key, value];
}
}

entries() {
return this[Symbol.iterator]();
}

*keys() {
for (const [key, value] of this) {
yield key;
}
}

*values() {
for (const [key, value] of this) {
yield value;
}
}
}


const key1 = { a: 1 };
const key2 = { b: 2 };
const keyValuePairs = [[key1, 'foo'], [key2, 'bar']];
const map = new IterableWeakMap(keyValuePairs);

for (const [key, value] of map) {
console.log(`key: ${JSON.stringify(key)}, value: ${value}`);
}
// key: {"a":1}, value: foo
// key: {"b":2}, value: bar

for (const key of map.keys()) {
console.log(`key: ${JSON.stringify(key)}`);
}
// key: {"a":1}
// key: {"b":2}

for (const value of map.values()) {
console.log(`value: ${value}`);
}
// value: foo
// value: bar

map.get(key1);
// □ foo

map.delete(key1);
// □ true

for (const key of map.keys()) {
console.log(`key: ${JSON.stringify(key)}`);
}
// key: {"b":2}
```

TLDR;

javascript-memory-management[2]

□□□□

□□□□

Stack: Static memory allocation

- □□□□compile time□
- □□□Stack□□
  - □□□□□□□ primitive values (strings, numbers, booleans, undefined, and null) and references
  - □□□□□□□□□□fixed amount of memory □

Heap: Dynamic memory allocation

- □□□Heap□
- □ objects and functions □□□□□□
- □□□objects □□ primitive values □□□□□ Stack

Garbage collection

Reference-counting garbage collection

- □ no references □□ objects □□□□□□□□ GA
- □□□□ □ Cycles
  - objects □□□□□□□□□□□□□□□□□□ null□□□□□ GA □□
  - □□□□□□ objects □□ heap □□□□□□□□

Mark-and-sweep algorithm

- □□□□□□□□□
- □□□□□□ root object □□□□□□□□□□□□□□
- root object□window□Browsers□ / global□Node□

Memory leaks

Global variables

- □□□□□□ scope □□□□without var / let / const□□□□□□ attach □ root object□
- `use strict;` mode □□□□□□□□
- window.xxx = null;` □□

Forgotten timers and callbacks

- `clearInterval(intervalId);` □□□□□□□
- `element.removeEventListener('click', onClick);` □□□□□□□□
- `element.parentNode.removeChild(element);` □□□□□□

Out of DOM reference

```
const elements = [];
const element = document.getElementById('button');
elements.push(element);

function removeAllElements() {
elements.forEach((item, index) => {
document.body.removeChild(document.getElementById(item.id)); // □□□□ DOM
elements.splice(index, 1); // □□□□ Array □□□□
});
}
```

---

1. https://www.ruanyifeng.com/blog/2017/04/memory-leak.html □

1. https://www.ruanyifeng.com/blog/2017/04/memory-leak.html
2. https://felixgerschau.com/javascript-memory-management

Date: 2024-02-29
Words: 1078
Time to read: 5 mins

Newer Older

2024-02-29
[JavaScript] Symbol

2024-02-28
[JavaScript] features