JavaScript Atomics

Back to JS page

Table of Contents

- JavaScript Atomics
  - Creating a Shared Typed Array
    - Example 1
  - Atomics.load() and Atomics.store()
    - Example 2
  - Atomics.add() and Atomics.sub()
    - Example 3
  - Atomics.exchange()
    - Example 4
  - Atomics.compareExchange()
    - Example 5
  - Document
  - Reference

The Atomics Object

The `Atomics` object provides atomic operations as static methods. They are used with `SharedArrayBuffer` objects to ensure that operations complete without interruption, which is critical for multi-threaded environments (Web Workers).

Atomics operations help prevent race conditions when multiple threads access shared memory.

Creating a Shared Typed Array

To use Atomics, you first need a `SharedArrayBuffer` and a typed array view:

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

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

<h4>Creating a Shared Typed Array</h4>
<p id="demo"></p>

<script>
// Create a SharedArrayBuffer with 4 x 32-bit integers = 16 bytes
const sharedBuffer = new SharedArrayBuffer(16);
const sharedArray = new Int32Array(sharedBuffer);

// Initialize values
sharedArray[0] = 10;
sharedArray[1] = 20;
sharedArray[2] = 30;
sharedArray[3] = 40;

let text = "SharedArrayBuffer created (16 bytes)<br>";
text += "Int32Array length: " + sharedArray.length + "<br><br>";
text += "Initial values: ";
for (let i = 0; i < sharedArray.length; i++) {
  text += sharedArray[i] + " ";
}

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

</body>
</html>
```

Example 1

Result View Example

Atomics.load() and Atomics.store()

`Atomics.load()` reads a value from a shared memory location. `Atomics.store()` writes a value to a shared memory location atomically:

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

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

<h4>Atomics.load() and Atomics.store()</h4>
<p id="demo"></p>

<script>
const sharedBuffer = new SharedArrayBuffer(8);
const sharedArray = new Int32Array(sharedBuffer);

// Store values atomically
Atomics.store(sharedArray, 0, 42);
Atomics.store(sharedArray, 1, 100);

// Load values atomically
const val0 = Atomics.load(sharedArray, 0);
const val1 = Atomics.load(sharedArray, 1);

let text = "Atomics.store() and Atomics.load():<br><br>";
text += "After Atomics.store(sharedArray, 0, 42): " + val0 + "<br>";
text += "After Atomics.store(sharedArray, 1, 100): " + val1 + "<br><br>";

text += "All values: ";
for (let i = 0; i < sharedArray.length; i++) {
  text += Atomics.load(sharedArray, i) + " ";
}

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

</body>
</html>
```

Example 2

Result View Example

Atomics.add() and Atomics.sub()

`Atomics.add()` adds a value at a given position and returns the old value. `Atomics.sub()` subtracts a value and returns the old value:

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

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

<h4>Atomics.add() and Atomics.sub()</h4>
<p id="demo"></p>

<script>
const sharedBuffer = new SharedArrayBuffer(8);
const sharedArray = new Int32Array(sharedBuffer);

sharedArray[0] = 50;
sharedArray[1] = 100;

// Add and subtract atomically
let old1 = Atomics.add(sharedArray, 0, 25);   // 50 + 25 = 75
let old2 = Atomics.sub(sharedArray, 1, 30);   // 100 - 30 = 70

let text = "Atomics.add() and Atomics.sub():<br><br>";
text += "Atomics.add(sharedArray, 0, 25):<br>";
text += "  Old value: " + old1 + "<br>";
text += "  New value: " + Atomics.load(sharedArray, 0) + "<br><br>";

text += "Atomics.sub(sharedArray, 1, 30):<br>";
text += "  Old value: " + old2 + "<br>";
text += "  New value: " + Atomics.load(sharedArray, 1) + "<br><br>";

text += "Final values: " + Atomics.load(sharedArray, 0) + ", " + Atomics.load(sharedArray, 1);

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

</body>
</html>
```

Example 3

Result View Example

Atomics.exchange()

`Atomics.exchange()` exchanges a value at a given position and returns the old value:

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

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

<h4>Atomics.exchange()</h4>
<p id="demo"></p>

<script>
const sharedBuffer = new SharedArrayBuffer(8);
const sharedArray = new Int32Array(sharedBuffer);

sharedArray[0] = 99;

// Exchange the value atomically
let oldValue = Atomics.exchange(sharedArray, 0, 200);

let text = "Atomics.exchange(sharedArray, 0, 200):<br><br>";
text += "Old value: " + oldValue + "<br>";
text += "New value: " + Atomics.load(sharedArray, 0) + "<br><br>";

text += "This is useful for safely swapping values in shared memory.";

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

</body>
</html>
```

Example 4

Result View Example

Atomics.compareExchange()

`Atomics.compareExchange()` exchanges a value only if the current value equals a specified expected value:

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

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

<h4>Atomics.compareExchange()</h4>
<p id="demo"></p>

<script>
const sharedBuffer = new SharedArrayBuffer(8);
const sharedArray = new Int32Array(sharedBuffer);

sharedArray[0] = 50;

// Try to exchange - only succeeds if current value is 50
let result1 = Atomics.compareExchange(sharedArray, 0, 50, 100);
let result2 = Atomics.compareExchange(sharedArray, 0, 50, 200); // Fails (value is now 100)

let text = "Atomics.compareExchange():<br><br>";

text += "Initial value: 50<br><br>";

text += "Attempt 1: compareExchange(0, 50, 100)<br>";
text += "  Expected 50, actual was " + result1 + " (matched!)<br>";
text += "  New value: " + Atomics.load(sharedArray, 0) + "<br><br>";

text += "Attempt 2: compareExchange(0, 50, 200)<br>";
text += "  Expected 50, actual was " + result2 + " (did NOT match!)<br>";
text += "  New value: " + Atomics.load(sharedArray, 0) + " (unchanged)";

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 Atomics