Shared Array Buffers and Atomics

Roland Guijt

INDEPENDENT SOFTWARE DEVELOPER AND TRAINER | MVP

@rolandguijt rolandguijt.com

Overview

Refresher: Web workers

Refresher: ArrayBuffer and typed arrays

SharedArrayBuffer

Atomics

Web Workers

Bring task parallelism to JavaScript

Conceived by W3C

Started by main thread

Each worker has it's own global environment

Nothing is shared between main thread and worker thread

Can do all kinds of tasks as long as it doesn't involve the DOM

Using a Web Worker

```
let worker = new Worker("someTask.js");
worker.postMessage(workOnThisData);
worker.onMessage = (event) => {
    //process data posted back by worker in event.data
}
```

Inside the Web Worker

```
self.addEventListener('message', (event) => {
    //get event.data == posted data by main thread
    //process data
    self.postMessage(processedData);
});
```

Sharing Data

Structured cloning

Transferables

SharedArrayBuffer

Binary Data in JavaScript

Since ES2015 with ArrayBuffer and Typed Arrays

New APIs in JavaScript

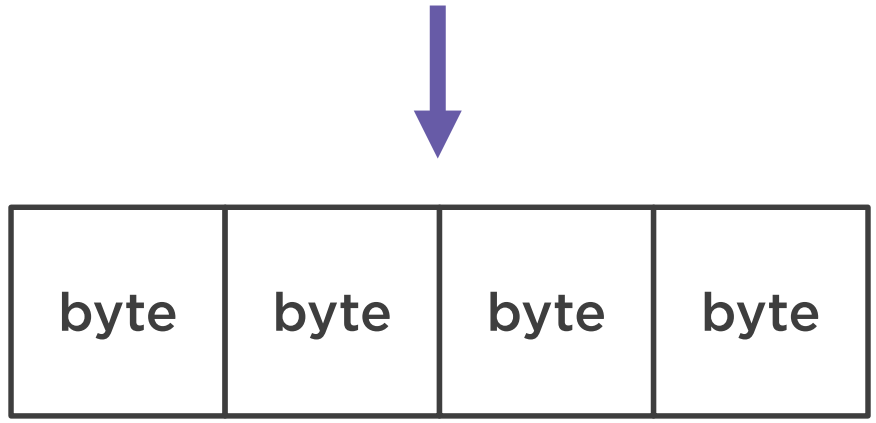
Reduce memory footprint

Optimize data transfer

Other

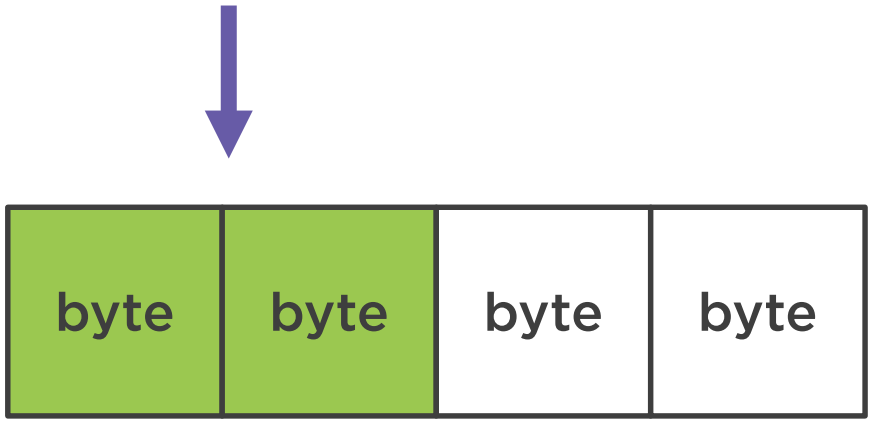
Array Buffer

```
let buffer = new ArrayBuffer(4);
```



Typed Array

```
let view = new Int16Array(buffer);
View[0] = 1000;
```



http://kangax.github.io/
compat-table/es2016plus/

SharedArray Buffer

Enable the sharing of binary data across threads

Web workers can work on the underlying binary data simultaneously

Sharing Data Other Than Integers

TextEncoder/TextDecoder

stringview.js

FlatJS

Waiting Until Data Becomes Available

```
while (uInt8View[6] === 0);
//start processing
```

Waiting Until Data Becomes Available

```
const waitValue = uInt8View[6];

while (waitValue === 0);

//start processing
```

Atomics

Safe access to a SharedArrayBuffer

Wait and continue mechanism

Atomic operations

Global variable Atomics

Load

```
while (Atomics.load(uInt8View, 6) === 0);
//start processing
```

Store

```
Atomics.store(uInt8View, 6, 1);
```

Atomic Operations

```
Atomics.add(uInt8View, 6, 1);
```

Summary

- Web workers enable parallelism and are isolated from other threads
- An ArrayBuffer instance represents a piece of memory: binary data
- Typed arrays are needed to access data in an ArrayBuffer
- The SharedArrayBuffer is an ArrayBuffer for which the underlying data can be shared across threads
- The Atomics global variable provides a collection of tools to work with the SharedArrayBuffer