Exploring JavaScript: Typed Arrays

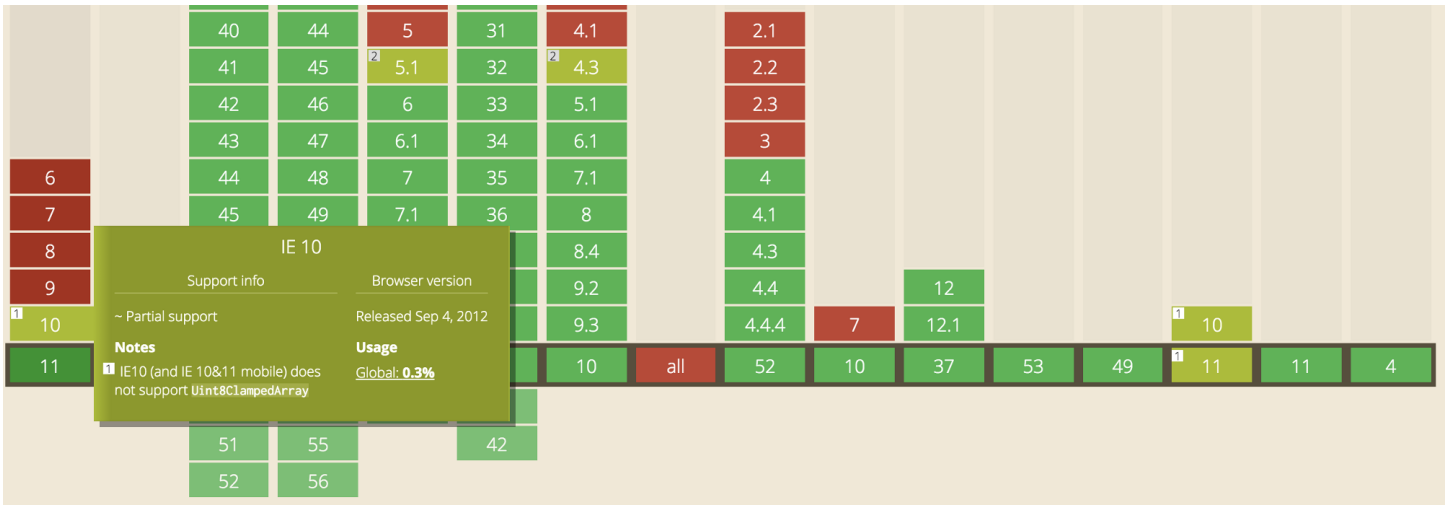
Valerii Iatsko
Oct 17, 2016 · 4 min read

Typed Arrays is a relatively new feature in JavaScript. They are designed to provide an easy way to work with binary data and structures, that was hardly possible before.

Initially, Typed Arrays were introduced for WebGL APIs. The reason behind this is that converting and guessing the type of standard JavaScript array might be too slow, while C API which is backing up WebGL might access already allocated in memory typed array.

Typed Arrays are part of ECMAScript 2015 (also known as ES6) specification.

Browser support



http://caniuse.com/#feat=typedarrays page

The main “problem” with browser support is IE9, which is still quite popular, as well as some browsers handling typed arrays slightly different:

1. IE10 (and IE 10&11 mobile) does not support `Uint8ClampedArray`

2. Safari 5.1 does not support `Float64Array`
3. Firefox 14 and under do not have `DataView` yet.
4. In versions before Safari 6, Typed Arrays are much slower than Array Objects.
5. `ArrayBuffer` has no slice method in IE 10.

Creating typed array

Typed array creation is relatively easy:

```
var arr = new Uint16Array(10);
arr[0] = 0xFFFF;
console.log(arr[0]);
```

Argument we are passing to array constructor is the number of elements in array (in the example above we are creating array of 10 uint16 numbers, which will take 20 bytes of memory).

Typed arrays could be following

Int8Array
size in bytes: 1
description: 8-bit two's complement signed integer
Web IDL type: byte
C type: int8_t

Uint8Array
size in bytes: 1
Web IDL type: 8-bit unsigned integer octet
C type: uint8_t

Uint8ClampedArray
size in bytes: 1
Web IDL type: 8-bit unsigned integer (clamped) octet
C type: uint8_t

Int16Array
size in bytes: 2
Web IDL type: 16-bit two's complement signed integer short
C type: int16_t

Uint16Array
size in bytes: 2
Web IDL type: 16-bit unsigned integer unsigned short
C type: uint16_t

Int32Array
size in bytes: 4
Web IDL type: 32-bit two's complement signed integer long
C type: int32_t

Uint32Array
size in bytes: 4
Web IDL type: 32-bit unsigned integer unsigned long
C type: uint32_t

Float32Array
size in bytes: 4
Web IDL type: 32-bit IEEE floating point number unrestricted float
C type: float

Float64Array
size in bytes: 8
Web IDL type: 64-bit IEEE floating point number unrestricted double
C type: double

JavaScript Operators & Typed Arrays

You might expect that all the operations on typed arrays will be done on bits this array contains. Apparently, that's not true.

Extending previous example:

```
var arr = new Uint16Array(10);

arr[0] = 0xFFFF;
```

```
console.log(~arr[0]); // -65535
```

ArrayBuffer & DataView

ArrayBuffer & DataView are parts of Typed Array implementation.

ArrayBuffer is basically holding the data of Typed Array.

Per example:

```
var fileArrayBuffer = reader.readAsArrayBuffer(file);
```

Here, ArrayBuffer will be holding all the bytes of the file. Sure, file might be in one encoding or another. E. g. to read UTF-8 characters, we need to assign Uint16 array type to this buffer:

```
var arr = new Uint16Array(fileArrayBuffer);
```

Now we can access this data in a regular way:

```
String.fromCharCode(arr[0]); // A
```

DataView is a bit more interesting part of implementation which is providing to read structures kept in binary data.

E. g., you have C structure like this:

```
struct data {
    unsigned int id;
    char[10] username;
}
```

We can read it from js by doing something like this:

```
var buf = new ArrayBuffer(11);
var id = new Uint8Array(buf, 0, 1);
var username = new Uint8Array(buf, 1, 10);
```

Here also alternative API can be used, e. g.:

```
...
var id = buf.getUint8(0);
...
```

A couple of use cases

WebGL

Buffer, pixel data and texture mappings are all using typed arrays. Example for texture mapping:

```
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(textureCoordinates),
gl.STATIC_DRAW);
```

Canvas

Yes, canvas image data is typed array:

```
var uint8ClampedArray = ctx.getImageData(...).data;
```

Useful to replace/analysing colors and for doing other kinds of image manipulation.

WebSockets

Yes, Web Sockets do support transferring binary data and ArrayBuffer representation of data. Enabling it is quiet easy:

```
webSocket.binaryType = 'arraybuffer';
```

Other Browser APIs

File API, XMLHttpRequest, Fetch API, window.postMessage() method and a lot of streaming APIs using typed arrays.

A word about performance

Array Object construction is being done inside JavaScript VM and Array Objects are being allocated inside VM's heap, while typed arrays allocated outside and involving the browser bindings.

Sequential reads/writes to both types of arrays are nearly equally fast, while Array Object will be slower for out-of-the-bounds writes (as it will require reallocation).

Conclusions here:

- Typed Arrays are beneficial when you need long-sustained array with specific size and type, and for use cases mentioned above
- Array Objects are beneficial when you need dynamic array or when you need temporary array (because of allocation method)

Programming JavaScript