

Don't miss out on the action at this years [Chrome Dev Summit](https://developer.chrome.com/devsummit/) (<https://developer.chrome.com/devsummit/>), happening on Nov 10th & 11th. [Register for livestream updates](https://services.google.com/fb/forms/cds2016/) (<https://services.google.com/fb/forms/cds2016/>).

How to convert ArrayBuffer to and from String



By [Renato Mangini](https://developers.google.com/web/resources/contributors#renatomangini)

(<https://developers.google.com/web/resources/contributors#renatomangini>)

Renato is a contributor to WebFundamentals

ArrayBuffers are used to transport raw data and several new APIs rely on them, including [WebSockets](http://www.html5rocks.com/en/tutorials/websockets/basics/) (<http://www.html5rocks.com/en/tutorials/websockets/basics/>), [Web Intents](http://webintents.org) (<http://webintents.org>), [XMLHttpRequest version 2](http://www.html5rocks.com/en/tutorials/file/xhr2/) (<http://www.html5rocks.com/en/tutorials/file/xhr2/>) and [WebWorkers](http://www.html5rocks.com/en/tutorials/workers/basics/#toc-gettingstarted-workercomm) (<http://www.html5rocks.com/en/tutorials/workers/basics/#toc-gettingstarted-workercomm>). However, because they recently landed in the JavaScript world, sometimes they are misinterpreted or misused.

Semantically, an [ArrayBuffer](#)

(https://developer.mozilla.org/en/JavaScript_typed_arrays/ArrayBuffer) is simply an array of bytes viewed through a specific mask. This mask, an instance of [ArrayBufferView](https://developer.mozilla.org/en/JavaScript_typed_arrays/ArrayBufferView) (https://developer.mozilla.org/en/JavaScript_typed_arrays/ArrayBufferView), defines how bytes are aligned to match the expected structure of the content. For example, if you know that the bytes in an ArrayBuffer represent an array of 16-bit unsigned integers, you just wrap the ArrayBuffer in a `Uint16Array` view and you can manipulate its elements using the brackets syntax as if the `Uint16Array` was an integer array:

```
// suppose buf contains the bytes [0x02, 0x01, 0x03, 0x07]
// notice the multibyte values respect the hardware endianness, which is little-en
var bufView = new Uint16Array(buf);
if (bufView[0]===258) {    // 258 === 0x0102
    console.log("ok");
}
bufView[0] = 255;    // buf now contains the bytes [0xFF, 0x00, 0x03, 0x07]
bufView[0] = 0xff05; // buf now contains the bytes [0x05, 0xFF, 0x03, 0x07]
bufView[1] = 0x0210; // buf now contains the bytes [0x05, 0xFF, 0x10, 0x02]
```

One common practical question about `ArrayBuffer` is how to convert a `String` to an `ArrayBuffer` and vice-versa. Since an `ArrayBuffer` is, in fact, a byte array, this conversion requires that both ends agree on how to represent the characters in the `String` as bytes. You probably have seen this "agreement" before: it is the `String`'s character encoding (and the usual "agreement terms" are, for example, Unicode UTF-16 and iso8859-1). Thus, supposing you and the other party have agreed on the UTF-16 encoding, the conversion code could be something like:

```
function ab2str(buf) {
  return String.fromCharCode.apply(null, new Uint16Array(buf));
}
function str2ab(str) {
  var buf = new ArrayBuffer(str.length*2); // 2 bytes for each char
  var bufView = new Uint16Array(buf);
  for (var i=0, strLen=str.length; i < strLen; i++) {
    bufView[i] = str.charCodeAt(i);
  }
  return buf;
}
```

Note the use of `Uint16Array`. This is an `ArrayBuffer` view that aligns bytes of the `ArrayBuffers` as 16-bit elements. It doesn't handle the character encoding itself, which is handled as Unicode by `String.fromCharCode` and `str.charCodeAt`.

Note: A robust implementation of the `String` to `ArrayBuffer` conversion capable of handling more encodings is provided by [the stringencodings library](http://code.google.com/p/stringencodings/) (<http://code.google.com/p/stringencodings/>). But, for simple usage where you control both sides of the communication pipe, the code above is probably enough. A standardized API specification for `String` encoding [is being drafted](http://wiki.whatwg.org/wiki/StringEncoding) (<http://wiki.whatwg.org/wiki/StringEncoding>) by the WHATWG (<http://wiki.whatwg.org/wiki/StringEncoding>) working group.

A popular StackOverflow [question about this](http://stackoverflow.com/questions/6965107/convert-between-strings-and-arraybuffers)

(<http://stackoverflow.com/questions/6965107/convert-between-strings-and-arraybuffers>) has a highly voted answer with a somewhat convoluted solution to the conversion: create a `FileReader` to act as a converter and feed a `Blob` containing the `String` into it. Although this method works, it has poor readability and I suspect it is slow. Since unfounded suspicions have driven many mistakes in the history of humanity, let's take a more scientific approach here. I have [jsperf'ed the two methods](http://jsperf.com/arraybuffer-string-conversion/4) (<http://jsperf.com/arraybuffer-string-conversion/4>) and the result confirms my suspicion and you [check out the demo here](http://www.html5rocks.com/en/tutorials/canvas/performance/embed.html?id=agt1YS1wcm9maWxlcmlhbnR1bGVzdBixrYURDA)

(<http://www.html5rocks.com/en/tutorials/canvas/performance/embed.html?id=agt1YS1wcm9maWxlcmlhbnR1bGVzdBixrYURDA>)

In Chrome 20, it is almost 27 times faster to use the direct `ArrayBuffer` manipulation code on this article than it is to use the `FileReader/Blob` method.

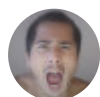
Update, August 2014: The Encoding API specification has matured, and a number of browsers now support it natively. The information in this article still applies for browsers that don't yet support the Encoding API, but the recommended approach is to use the official API wherever possible. See [Easier ArrayBuffer <-> String conversion with the Encoding API](http://updates.html5rocks.com/2014/08/Easier-ArrayBuffer---String-conversion-with-the-Encoding-API) (<http://updates.html5rocks.com/2014/08/Easier-ArrayBuffer---String-conversion-with-the-Encoding-API>) for more details.

6 comments



Add a comment as Devendra Katuke

Top comments



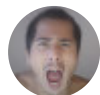
Ling talfi 7 months ago - Shared publicly

The first code block's comments are erroneous.

I believe the correct block would be the following:

[Read more \(10 lines\)](#)

[1](#) · [Reply](#)



Ling talfi 7 months ago (edited) - Shared publicly

I believe there are some typos in the second code block.

The working example for me is the following code:

```
var buf = [72, 69, 76, 76, 79]; // -> HELLO
```

[Read more \(16 lines\)](#) · [Translate](#)

[1](#) · [Reply](#)



Danny Andrews 8 months ago - Shared publicly

Thanks for the tip. But When I try to encode certain `ArrayBuffers` using the `ab2str` method I get 'Range: Uint16Array should be a multiple of 2.'

+1 [1](#) · [Reply](#)



Michael Cohen 8 months ago - Shared publicly

Yes, thanks! Very useful



John Walker 1 year ago - Shared publicly

Thank you Renato, still useful



· Reply



Claudijo Borovic 1 year ago - Shared publicly

Thanks for an interesting post. Seems to be a typo in the example "<" instead of "<"

+2



· Reply

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](http://creativecommons.org/licenses/by/3.0/) (<http://creativecommons.org/licenses/by/3.0/>), and code samples are licensed under the [Apache 2.0 License](http://www.apache.org/licenses/LICENSE-2.0) (<http://www.apache.org/licenses/LICENSE-2.0>). For details, see our [Site Policies](https://developers.google.com/site-policies) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated October 12, 2016.



[Chromium Blog](#)

The latest news on the
Chromium blog



[GitHub](#)

Fork our API code samples and
other open-source projects.



[Twitter](#)

Connect with @ChromiumDev
on Twitter



[Videos](#)

Check out the Web Developer
Relations team's videos



Events

Attend a developer event and
get hacking