Workers ♥ ArrayBuffer

By Eric Bidelman

Engineer @ Google working on web tooling: Headless Chrome, Puppeteer, Lighthouse

As of ~~crbug.com/73313~~, Chrome 13 and FF5 support sending an `ArrayBuffer` (or Typed Array) to/from a Web Worker. For example:

worker.js

```
self.onmessage = function(e) {
  var uInt8Array = e.data;
  postMessage("Inside worker.js: uInt8Array.toString() = " + uInt8Array.toString(
  postMessage("Inside worker.js: uInt8Array.byteLength = " + uInt8Array.byteLengt
};
```

main.html

```
var uInt8Array = new Uint8Array(new ArrayBuffer(10));
for (var i = 0; i < uInt8Array.length; ++i) {
  uInt8Array[i] = i * 2; // [0, 2, 4, 6, 8,...]
}

console.log('uInt8Array.toString() = ' + uInt8Array.toString());
console.log('uInt8Array.byteLength = ' + uInt8Array.byteLength);

worker.postMessage(uInt8Array);
```

Why is this exciting?...binary data!

Instead of the browser serializing your `postMessage()` data to a JSON object, it uses the structured clone algorithm to copy the `ArrayBuffer` to the worker's context, and vice versa. This opens up a real potential for workers that we haven't seen before. That is, being able to easily pass binary data between main app and worker thread.

Typed array I/O makes intense image manipulation, sound processing, and heavy webgl calculations much more feasible. For example, one could read a File as an array buffer or fetch a Blob using XHR2 and pass the result directly to a worker. No more base64 encoding the data :)

In my opinion this is one of those nitpicks workers should have included from the start. It just makes sense.

Last updated July 2, 2018.