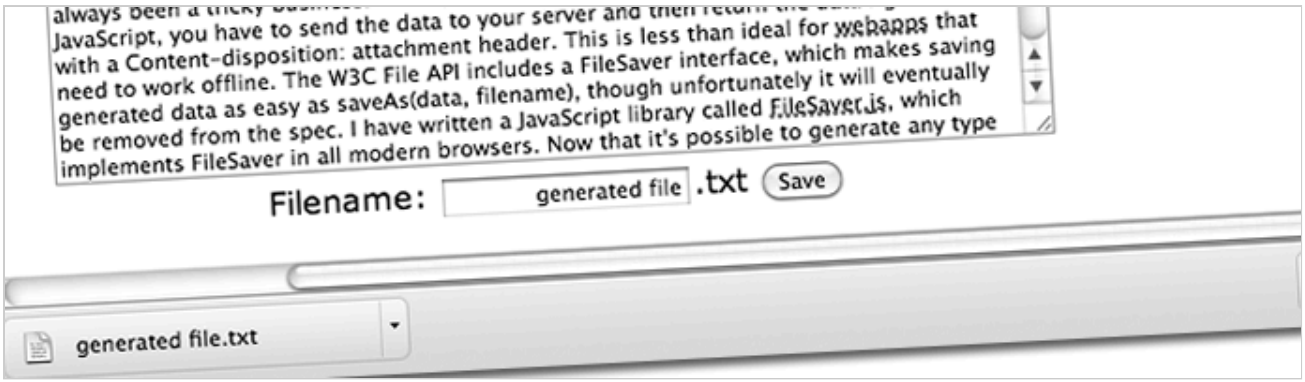
Saving generated files on the client-side

By Eli Grey
Eli is a Developer

Have you ever wanted to add a “Save as…” button to a web app? Whether you're making an advanced WebGL-powered CAD web app and want to save 3D object files or you just want to save plain text files in a simple Markdown text editor, saving files in the browser has always been a tricky business. Usually when you want to save a file generated with JavaScript, you have to send the data to your server and then return the data right back with a `Content-disposition: attachment` header. This is less than ideal for web apps that need to work offline. The W3C File API includes a `FileSaver` interface, which makes saving generated data as easy as `saveAs(data, filename)`, though unfortunately it will eventually be removed from the spec. I have written a JavaScript library called FileSaver.js, which implements `FileSaver` in all modern browsers. Now that it's possible to generate any type of file you want right in the browser, document editors can have an instant save button that doesn't rely on an online connection. When paired with the standard HTML5 `canvas.toBlob()` method, FileSaver.js lets you save canvases instantly and give them filenames, which is very useful for HTML5 image editing webapps. For browsers that don't yet support `canvas.toBlob()`, Devin Samarin and I wrote canvas-toBlob.js. Saving a canvas is as simple as running the following code:

```
canvas.toBlob(function(blob) {
  saveAs(blob, filename);
});
```

I have created a demo of FileSaver.js in action that demonstrates saving a canvas doodle, plain text, and rich text. Please note that saving with custom filenames is only supported in browsers that either natively support `FileSaver` or browsers like Google Chrome 14 dev and Google Chrome Canary, that support <a>.download or web filesystems via `LocalFileSystem`.



View FileSaver.js demo

How to construct files for saving

First off, you want to instantiate a `BlobBuilder`. The `BlobBuilder` API isn't supported in all current browsers, so I made BlobBuilder.js which implements it. The following example illustrates how to save an XHTML document with `saveAs()`.

```
var bb = new BlobBuilder();
bb.append((new XMLSerializer).serializeToString(document));
var blob = bb.getBlob("application/xhtml+xml;charset=" + document.characterSet);
saveAs(blob, "document.xhtml");
```

Not saving textual data? You can append binary `Blob`s and `ArrayBuffer`s to a `BlobBuilder` too! The following is an example of setting generating some binary data and saving it.

```
var bb = new BlobBuilder();
var buffer = new ArrayBuffer(8); // allocates 8 bytes
var data = new DataView(buffer);

// You can write (u)int8/16/32s and float32/64s to dataviews
data.setUint8 (0, 0x01);
data.setUint16(1, 0x2345);
data.setUint32(3, 0x6789ABCD);
data.setUint8 (7, 0xEF);

bb.append(buffer);
var blob = bb.getBlob("example/binary");
saveAs(blob, "data.dat");
// The contents of data.dat are &lt;01 23 45 67 89 AB CD EF&gt;
```

If you're generating large files, you can implement an abort button that aborts the `FileSaver`.

```
var filesaver = new FileSaver(blob, "video.webm");
abort_button.addEventListener("click", function() {
  filesaver.abort();
}, false);
```

Last updated July 2, 2018.