Minified files

A client wondered what the difference between regular library files, and their minified counterparts.

Today, I'll share the answer with you.

File sizes are very important. Big files load slow and small files load fast. Every line of code slows down your site by a very small amount.

Library files are very big. They can slow down a site in a noticeable way.

Just like how someone can squeeze a lot of items into a small bag, you can do the same with code.

I'll give you a hint with the following images:

exhibit A:

```
function( global, factory ) {

	"use strict";

	if ( typeof module === "object" && typeof module.exports === "obje

		// For CommonJS and CommonJS-like environments where a pro
		// is present, execute the factory and get jQuery.
		// For environments that do not have a `window` with a `do
		// (such as Node.js), expose a factory as module.exports.
		// This accentuates the need for the creation of a real `w
		// e.g. var jQuery = require("jquery")(window);
		// See ticket #14549 for more info.
		module.exports = global.document ?
			factory( global, true ) :
			function( w ) {
				if ( !w.document ) {
					throw new Error( "jQuery requires
				}
				return factory( w );
			};
	} else {
		factory( global );
	}

// Pass this if window is not defined yet
} )( typeof window !== "undefined" ? window : this, function( window, noG

// Edge <= 12 - 13+, Firefox <=18 - 45+, IE 10 - 11, Safari 5.1 - 9+, iOS
// throw exceptions when non-strict code (e.g., ASP.NET 4.5) accesses str
```

Exhibit B: (minified)

```
function(e,t){"use strict";"object"==typeof module&&"object"==typeof
rror("jQuery requires a window with a document");return t(e)}:t(e)}("
],r=Object.getPrototypeOf,s=t.slice,g=t.flat?function(e){return t.fla
},o=n.toString,v=n.hasOwnProperty,a=v.toString,l=a.call(Object),y={},
.item},x=function(e){return null!=e&&e===e.window},E=C.document,c={ty
n=n||E).createElement("script");if(o.text=e,t)for(r in c)
i=t[r]||t.getAttribute&&t.getAttribute(r))&&o.setAttribute(r,i);n.hea
||"function"==typeof e?n[o.call(e)]||"object":typeof e}var f="3.6.0",
&&e.length,n=w(e);return!m(e)&&!x(e)&&("array"===n||0===t||"number"==
return s.call(this)},get:function(e){return null==e?s.call(this):e<0?
.prevObject=this,t},each:function(e){return S.each(this,e)},map:funct
return this.pushStack(s.apply(this,arguments))},first:function(){retu
his.pushStack(S.grep(this,function(e,t){return(t+1)%2}))},odd:functic
=this.length,n=+e+(e<0?t:0);return this.pushStack(0<=n&&n<t?[this[n]]
his.prevObject||this.constructor()},push:u,sort:t.sort,splice:t.splic
},s=1,u=arguments.length,l=!1;for("boolean"==typeof a&&(l=a,a=argumer
e=arguments[s]))for(t in e)r=e[t],"__proto__"!==t&&a!==r&&(l&&r&&(S.i
:{},i=!1,a[t]=S.extend(l,o,r)):void 0!==r&&(a[t]=r));return a},S.exte
rror(e)},noop:function(){},isPlainObject:function(e){var t,n;return!(
t=r(e))||"function"==typeof(n=v.call(t,"constructor")&&t.constructor)
)return!1;return!0},globalEval:function(e,t,n){b(e,{nonce:t&&t.nonce}
for(n=e.length;r<n;r++)if(!1===t.call(e[r],r,e[r]))break}else for(r i
ull!=e&&(p(Object(e))?S.merge(n,"string"==typeof e?[e]:e):u.call(n,e)
=+t.length,r=0,i=e.length;r<n;r++)e[i++]=t[r];return e.length=i,e},gr
},map:function(e,t,n){var r,i,o=0,a=[];if(p(e))for(r=e.length;o<r;o++
```

These two sections of code take the same space.

What could be different? The empty space doesn't do anything. Except take space.

Minifying removes the empty space. That's really the magic of minification.

This leads us back to our first concern. Speed. The smaller the file size, the faster it loads, the faster your site.

Yes, this is great information. But do I expect you to go and manually remove all the extra space yourself?
No.
We have tools for that. As a developer it's best to have a minification tool locally on your computer. You can have an online tool minify your code, but the risk of giving someone proprietary code is there.

This is an example an online one → https://www.minifier.org/