
By Scott Robinson

Takeaway

There are common, obvious mistakes anybody can make when creating an AJAX application; then there are the mistakes that can obliterate function and performance, stripping away all of AJAX's benefits. Scott Robinson shows you how to avoid the Seven Deadly Sins of AJAX application development.

AJAX

The AJAX bandwagon is a good place to be. It takes you to faster, more efficient, more dynamic apps. But it also has pitfalls all its own.

At first blush, those pitfalls may seem avoidable through sheer common sense, and to a degree, that's true. But [AJAX apps](#) bristle with architectural differences from its DHTML forebears. No matter how much common sense you bring to the [application development](#) task, it is still good to learn from the mistakes of those who have preceded you. We'll call these mistakes our Seven Deadly Sins, but they are by no means the whole story.

In fact, before you commit any of the deadly sins, you're much more likely to commit a few lesser sins. So we'll start with those. These are the ones that make everybody's list. These mistakes are so common that you can find most of them in a simple Google search.

The Seven Lesser Sins

- 1. Misuse of the Back Button** -- This is the one mistake everyone's going to have on their list. The Back Button has become a user expectation in many kinds of Web apps. Many new AJAX developers drop a Back Button into their AJAX app and promptly break it, for several reasons. First, JavaScript isn't the friendliest language for it; and second, AJAX design requires a new way of thinking. It's not always readily apparent to a new AJAX developer that "Back" isn't best. "Back" is a feature you'll need at a page-update point or more often, when you want an event-specific "Undo" feature. Think this through before you code, or you'll end up doing it twice.
- 2. Not telling the user what's happening** -- Part of how AJAX works is that it doesn't use conventional Web UI page loading. So you need to explicitly design some visual cue to let the user know what's happening.
- 3. Neglecting Links** -- Here's another AJAX standard blunder: neglecting URLs that can be cut-and-pasted for uses outside the app. How many times have we grabbed a URL and e-mailed it to someone? When you're using AJAX, the only way to provide your user with useful cut-and-pasteable URLs is to manually provide them. Why? Because in an AJAX app, *the server is not providing the page*: JavaScript is dynamically generating it! Don't neglect your users' potential interest in this all-too-common feature of Web apps. Take pains to provide URLs, since the server won't.

4. **Trading content control for page control** -- Breaking with the traditional client-server interaction is a wonderful gift, if you're seeking dynamic content control. But it's also a curse: rewriting highly-specific areas of a page to fine-tune a user's interactive experience does indeed give you fine control, but this pulls you away from the big picture.
All too often, we focus on processing some sector of the page and forget that the page isn't going to be refreshed by the server. This can lead to a fragmented page, a confusing user experience, wherein the screen they're looking at can be out-of-date with itself! Take care to keep your attention on the entire page; make sure that any dynamic content event evokes changes everywhere the user will need to see them.
5. **Killing spiders** -- The upside of AJAX is the truckloads of text you can feed a page without a reload. And the downside of AJAX is the truckloads of text you can feed a page without a reload. If the app is intended to be search-engine-friendly -- well, you can imagine. Whatever is happening in the guts of the page, be sure to plant plenty of stable text up top, for the spiders to play with.
6. **Producing unspeakable text** -- AJAX doesn't support many character sets. This isn't a life-or-death limitation, but forgetting it can create real problems. Your basic character set is UTF-8. Don't forget to properly encode whatever JavaScript is sending, and remember to set the server-side character set for content.
7. **Not letting the JavaScript-challenged user know what's going on.** There are browsers out there that don't have JavaScript enabled, and users out there who don't understand immediately what that means. Be sure to clue them in.

Truth to tell, most of those are common-sense issues. The real killers are less obvious.

The Seven Deadly AJAX Sins

1. **Letting memory leak** -- Anyone who has worked in development for very long understands cyclic references and the danger they pose to memory management. JavaScript, the workhorse of AJAX, is a memory-managed language. What that means is that JavaScript has built-in garbage collection, so that it sniffs out variables no longer accessible by any reference path and de-allocates the memory they're using.
That's fine as a general principle; but you can't count on it to keep your memory usage optimized, because of those cyclic references, when a Model-layer object is referencing a View element and vice versa. Nulling the object nulls the element, in principle, but if there's a backward reference from element to object, then the garbage collector won't touch the object.
Now, here's where it gets tricky: in the Document Object Model, any DOM node that's part of the document tree may be referenced by virtue of its presence in the tree, whether it's explicitly referenced by another object or not! So any object you tag for garbage collection that is back-referenced by a DOM node must be explicitly nulled in that direction, or its memory remains allocated!
2. **Not knowing what "asynchronous" means** -- Asynchrony can be really unsettling to a user unaccustomed to it, and this can be ironic, since the Web apps you'll design and build for those users wouldn't find them at all unsettling *if they were desktop apps*. This is a crucial design point! Most Web apps are functionally very similar to their desktop counterparts. But in a Web app, there's a looming characteristic that sets it apart: *user expectations*.
Users carry a very different set of biases and anticipations into a session with a Web browser that they don't tote into a desktop app exchange. So while it may be incredibly cool and efficient to have myriad responses trotting back to the page from the server, and the page revising itself in the meantime, it can be deeply disorienting to the user to see this constant barrage of change. For this reason, you need to apply two rules, and consider them with *every* change that comes into the user's field-of-vision: If the update is not immediately essential to the user, make the update unobtrusive, so that it does not distract; and if the update is vital to the user's interaction with the app, make its presence on the page clear and prominent.

- 3. Keeping the server in the dark** -- The decoupling of client and server has a huge downside that seldom occurs to us up-front. Server-side apps know all and see all: every exception, every reload, every event can be watched and recorded, and of course the server knows exactly what the client looks like, because the server basically wrote whatever's on the screen.
In an AJAX app, this isn't the case. Stuff happens, and it happens independently of the server, and this means that there can be problems on the client side that the server won't immediately be privy to. Capture and log events and exceptions on the client side, in a place and manner that enables the server to home in on problems that require its intervention as soon as possible.
- 4. Getting lazy with GET** -- GET is for retrieving data; POST is for setting it. Don't use GET when you know you shouldn't, even if you think it will do no harm. GET operations change state, and links that change state are confusing to users; most are accustomed to links as guides to navigation, not function.
- 5. Not accounting for data type** -- [JavaScript](#), isn't part of the .NET Framework. While this may draw tears of sorrow, it presents us with an example of a problem you'll eventually run into: making certain that JavaScript understands the data types of the platform it's running on, and vice versa, be it .NET or otherwise. There are various converters available to you, and you should seek them out. The Ajax.NET Pro library, for instance, offers converters that can convert objects between .NET and JavaScript Object Notation.
- 6. Some apps just don't know when to shut up** -- Dynamic generation of content, with no dependency on page-reload, can be disastrous if left open-ended. How many Web pages have you seen that were longer than the Congressional Record? If Web pages that go on forever are a confusing nightmare for users, just imagine how they'll feel about an *application* that goes on and on and on. Make your Web app pages dynamic, but build in some practical limits.
- 7. Keep your JavaScript out of your DOM** -- Remember that AJAX is built on the [Model-View-Controller](#) structure. Take this seriously. JavaScript belongs in the Model layer, DOM in the View layer, and the Controller is their marriage counselor. Make sure your Web document has the same content independent of JavaScript (this helps with the JavaScript-disabled user) -- *except when the content itself is only meaningful or practical if the user can use JavaScript*. In that instance, *create that content with JavaScript*.

Additional resources

- TechRepublic's [Downloads RSS Feed](#) **XML**
- Sign up for TechRepublic's [Downloads Weekly Update](#) newsletter
- Sign up for TechRepublic's [Web Development Zone](#) newsletter
- Check out all of TechRepublic's [free newsletters](#)

Version history

Version: 1.0

Published: February 27, 2007

Tell us what you think

TechRepublic downloads are designed to help you get your job done as painlessly and effectively as possible. Because we're continually looking for ways to improve the usefulness of these tools, we need your feedback. Please take a minute to [drop us a line](#) and tell us how well this download worked for you and offer your suggestions for improvement.

Thanks!

—The TechRepublic Downloads Team