The Unbearable Lightness of Websites

Joel Dueck

Tuesday, January 12th, 2016

My brother in law is teaching himself how to make websites, typing HTML and CSS by hand into a text editor. Like me, he knows a thing or two about programming, and he asked me "once you've spent all this time designing and composing all this HTML, is there some way to automate its reuse, as one does with code?"

What an interesting question. Because of course when you're writing a program, and you write a function that saves you a lot of work, you can take that code and reuse it in other programs. But HTML isn't code, and writing it isn't programming. So no, you can't reuse HTML the same way[1] you would reuse code—you have to find some way to bring actual code into the picture. But of course he's on the right track, because this is The Question faced by web publishers, going right back to the 1990s. The day the first website went live people started looking for ways to automate web pages. No one wants to type that stuff by hand over and over and over.

What we usually do is install and configure a server-side programming language like PHP or Ruby. Then we install and configure a bunch of scripts written in that language, and we spin up and configure a database engine like MySQL, and connect that with the scripts. The scripts let us store our HTML templates in these database tables over here, and the content in those tables over there. The scripts often define their own mini-programming language for adding some smarts to the templates, so we write more code to get those working the way we want.

And now, instead of having a tidy little stack of web pages, what you have is a Content Management System, or CMS: the software equivalent of a large multi-function appliance, like the printer shown here. Once it's all assembled, configured, tuned and warmed up, it will sort and recombine your text into web pages and spit those out all day long. You don't even have to touch HTML any more. But you need someone who knows how to realign the print heads, clear jams and replace the fuser, and every few years you'll need to replace the whole thing.

This is how we publish websites now. It works really well for just about every kind of web site: company websites, web apps, online shopping, social media, and even personal blogs and journals.

My introduction to the multi-function appliance approach came in 2004, when Textpattern was released. Until then I had been writing every web page by hand in an HTML editor. Getting a CMS running meant I needed to get familiar with a database, two new programming languages, and a web server. For someone who enjoys tinkering with computers, this was not actually that hard, and it's gotten much easier. These days you can get your own virtual

[1] By which I mean including code by reference that is defined elsewhere—cutting and pasting doesn't count.

Figure 1: An on-demand book printer. Photo from Wikipedia (*https://en.wikipedia.org/wiki/Print_on_demand*), GNU Free Documentation License.

server for $5 a month, and with a bit of searching and reading[2] you can have everything up and running in a couple of hours, or even less than an hour if you've done it a couple of times before.

[2] *https://thelocalyarn.com/files/migration-quickref.html*

But although my appliance still works well, it's definitely starting to show its age. One of the pieces might be updated so that it's no longer compatible with the others, or a piece might be abandoned altogether, leaving you to find a replacement and rework everything to fit around it. (Why replace the pieces that no one is updating any more? Because all the other pieces *are* being updated: you need them all to keep pace with each other.) And security holes are always a risk: someone finding a way inside your server through a vulnerability in one of the layers, or in the way they interact with each other.

I've managed to keep my web publications alive across a few of these transitions. When the machine works, it works really well. But maintaining the machine soon begins to feel like drudgery. And besides the actual maintenance I have to keep current on four or five loosely-related technologies, as well as regularly trawl the web to make sure no bad news has surfaced about any of them.

Any sane person in this kind of situation starts to step back a bit and ask a few questions, starting with *how many times am I going to have to do this?* and perhaps ending with *I going to have anything to show for it in ten, twenty, or thirty years?*

The answer to that last question is *probably not, unfortunately*—not if I keep doing things the same way. Web pages—web publishing—isn't durable. If the machine breaks or is turned off, everything you made with it goes away.

If the web server were more like our last major humanist invention, the printing press, the durability of my web publishing work would be less of a concern. We can still read books and pamphlets printed five hundred years ago, even though the presses that made them have long since been destroyed. Not so a web page: when the appliance goes dark, the page disappears. If the outage is permanent, the disappearance is too. This is happening all the time, as servers fail, or companies are acquired and shut down.[3] Looking out even further, suppose some black swan event destroyed all or most of the internet—what would the human race have to show for it? Even only as a shared memory, "the internet" and all it once contained would continue to affect us well into the future; but under that circumstance, its effect on us would come to seem less like an awakening and more like that of a repressed trauma.

[3] "When Yahoo! switched off the servers for GeoCities, the Web posting service, on Oct. 27, some 7 million of the Internet's first websites went dark forever." Time Magazine, 9 Nov 2009 (*http://content.time.com/time/business/article/0,8599,1936645,00.html*)

So this is the Two-Fold Problem of web publishing: the appliances we use to publish web pages are heavy (powerful but complicated and high-maintenance) and the web pages they produce are unbearably light (they just won't endure over time). This isn't really a problem for sites like shopping catalogs and web apps, because they need high performance and flexibility, and don't cost us anything culturally when they disappear. But for others it is

a huge problem. I subscribe to several blogs with great writing: I doubt if any of the content on those sites will still exist in thirty years. This is sad for the authors and sad for me.

The most common single response to the Two-Fold Problem, for the few web publishers that grapple with it, is to use a *static site generator* instead of a database-backed CMS. I won't go into the details here—if you're still reading, you probably already know them—but the essential move is to get the *source* of your web pages out of a database and into a bunch of plain old text files. This *partially* addresses with the Two-Fold Problem, because it removes the need for a database, which makes the appliance (the first part of the problem) a lot smaller, faster and simpler. But it turns out that putting your documents in a text file instead of a database doesn't make them any more durable. True, it makes those documents simpler to work with in some ways. But anything digital, anything that needs to go through a circuit board before your eyes can look at it, is still ultimately invisible junk on a fifty-year time scale—no matter what format it's in.

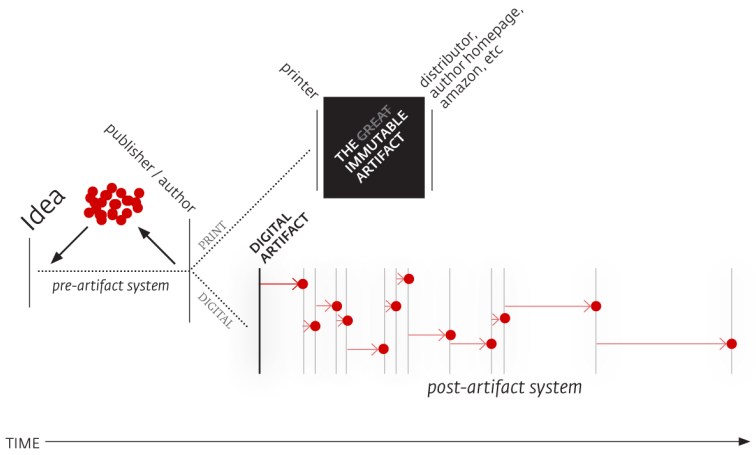

Figure 2: From Post-Artifact Books and Publishing (*http://craigmod.com/journal/post_artifact/*) by Craig Mod (cc License)

Craig Mod drew this diagram to illustrate how traditional books have been changed—or perhaps how they *should* be changed—by digital publishing. But it could also a two way street: Digital publishing should be changed by books. Digital publishing could use more Immutable Artifacts, because artifacts are durable.