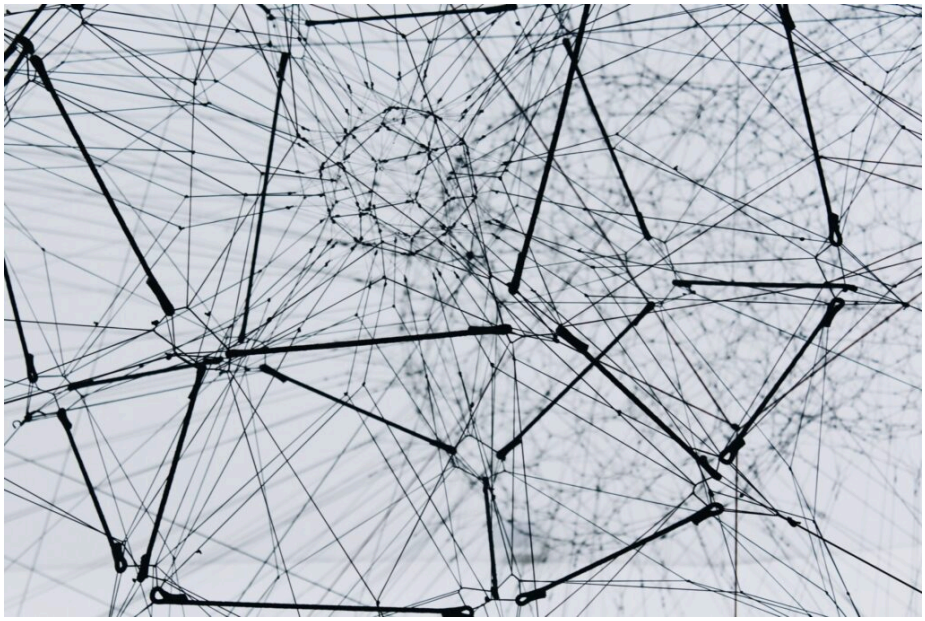
Issue 052: The World Wide Web

Adrian Kosmaczewski

January 2nd, 2023



Welcome to the fifty-second issue of *De Programmatica Ipsum*, about *The World Wide Web.*

In this edition:

- We analyze the evolution of websites, from SPAs to server-rendered Hypertext[1].
- In the Library section[2], we review "Transcending CSS" by Andy Clarke.[3]
- In our Vidéothèque section[4], we learn how to use a NeXT Computer thanks to Steve Jobs.[5]

We opened an account on Mastodon last year: follow us at @deprogrammaticaipsum@mas.to[6] to be notified of new releases!

We would also like to thank our patrons who generously contribute every month (or have contributed in the past) to our work and help us run this magazine. Thank you so much! In alphabetical order: Adam Guest, Adrian Tineo Cabello, Benjamin Sheldon, Christopher Nascone, Jean-Paul de Vooght, Patryk Matuszewski, Paul Hudson, Roger Turner, and Szymon Licau.

Enjoy this issue! Please subscribe to our free newsletter[7] to stay updated about new releases, share the articles on social media, or contribute[8] if you would like to support our work.

[1] https://deprogrammaticaipsum.com/from-hypertext-to-spas-to-hypertext/
[2] https://deprogrammaticaipsum.com/category/library/
[3] https://deprogrammaticaipsum.com/andy-clarke/
[4] https://deprogrammaticaipsum.com/category/videotheque/
[5] https://deprogrammaticaipsum.com/steve-jobs/
[6] https://mas.to/@deprogrammaticaipsum
[7] https://deprogrammaticaipsum.com/newsletter/
[8] https://deprogrammaticaipsum.com/contribute/

Cover photo by Alina Grubnyak[9] on Unsplash[10].

[9]https://unsplash.com/@alinnnaaaa?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[10]https://unsplash.com/photos/ZiQkhI7417A?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

From Hypertext To SPAs To Hypertext

Adrian Kosmaczewski

January 2nd, 2023

In March 1989, Tim-Berners Lee (aka TimBL), a scientist at CERN[1] in Geneva, Switzerland, proposed[2] a Hypertext-based system to organize the incredible amounts of information generated annually by the organization. He followed this proposal with prototypes, leading to the creation of the first web server and browser and the development of the first versions of the HTTP protocol and the HTML markup language.

CERN needed a way to cope with a massive turnover of scientists generating tons of written content, primarily papers, journal articles, and the occasional book draft. Words on those documents should easily be "linked" to other documents on the same network to enable references, discussion, and sharing.

Unbeknownst to most younger developers, "Hypertext" was one of the hottest ideas at the end of the eighties; it was right up there with Object-Oriented Programming[3], the Lambada[4] (no, not "Lambda,") and the first Batman[5] movie. It is no coincidence that TimBL used the H word in his proposal. Case in point: the October 1988 issue of Byte Magazine[6] (available on newsstands merely five months before the proposal) is titled "Hypertext." It contains[7] several pages of explanations and historical references to various Hypertext ideas, prototypes, and concepts, including Vannevar Bush[8]'s Memex[9], Ted Nelson[10]'s Project Xanadu[11] (noth-

[1] https://home.cern/
[2] https://www.w3.org/History/1989/proposal.html
[3] https://deprogrammaticaipsum.com/issue-49-object-oriented-programming/
[4] https://en.wikipedia.org/wiki/Lambada_(song)
[5] https://en.wikipedia.org/wiki/Batman_(1989_film)
[6] https://archive.org/details/byte-magazine-1988-10
[7] https://archive.org/details/byte-magazine-1988-10/page/n263/mode/2up
[8] https://en.wikipedia.org/wiki/Vannevar_Bush
[9] https://en.wikipedia.org/wiki/Memex
[10] https://en.wikipedia.org/wiki/Ted_Nelson
[11] https://en.wikipedia.org/wiki/Project_Xanadu

ing to do with Olivia Newton-John[12], pretty promise), and Apple's HyperCard[13] software product.

We will never know, but let us pretend that TimBL had a copy of this magazine on his desk while drafting the proposal. It is said that the very moniker of "Home Page" comes from Apple's HyperCard and its "Home Card," where all interaction would begin. We also know that he attended[14] the HyperText conference in San Antonio, Texas, in December 1991.

It means that the Web (with an uppercase W) we have been using for the past 25 years was born as a Hypertext (with an uppercase H) document management system. Its most important feature, by all means, was the `<A HREF="…">` tag, enabling words to appear underlined in blue and making the "Hypertext" concept a reality. Couple that with a "Back" button driving a FIFO data structure in memory; boom, you have got yourself a browser.

Various other standards have joined HTML ever since, most notably JavaScript in 1995 and CSS in 1996, forming the saint trinity of web development technologies: HTML, CSS, and JavaScript, each providing structure, style, and behavior to everything we see online. Each of these languages became standardized, and many others came after, all managed by a single group of people collectively called the W3C[15], short for "World Wide Web Consortium."

Despite such conceptual origin, however, the Web became a construction of historical proportions; the computer industry equivalent of Gutenberg's press, if you are into corny comparisons. Leaving aside the ripples generated in the social and economic tissue of our modern world (which are many, including a global stock market crash[16] in 2000,) software developers have embraced the Web and transformed the aforementioned trinity of technologies (HTML, CSS, and JavaScript) into a fully-fledged "platform"[17] (whatever that is) upon which to deploy applications of all kind.

The problem is that the Web was never meant to be a programming platform. Remember? It was a document-sharing-and-linking system. It was *coerced* into a programming platform.

Starting with Paul Graham and his Viaweb[18] project in 1995, followed by NeXT's WebObjects[19] the same year, companies and individuals have created again and again a myriad of systems used to generate HTML, CSS, and JavaScript in one way or another. A lot of "innovation" and billions of dollars have been poured into projects that are just essentially glorified text generators. Thus, "web applications" were born.

One of web applications' most significant selling points was the "single deployment" idea: copy your app to a server, give customers a URL, and profit. Time for an upgrade? No need for CD-ROMs or downloading binaries; update the app, and let your users enjoy new features and bug fixes immediately. Coupled with the emergence of the Agile movement, a whole new category of software developers emerged in 1997, crashed in 2000, and re-emerged in 2004 thanks to the "années folles" also called the Web 2.0 frenzy.

Since neither HTML, CSS, nor JavaScript was "developer-friendly" enough, we have created

[12] https://en.wikipedia.org/wiki/Xanadu_(film)
[13] https://en.wikipedia.org/wiki/HyperCard
[14] https://lists.w3.org/Archives/Public/www-talk/1991SepOct/0005.html
[15] https://w3c.social/@plehegar/109593223455061643
[16] https://en.wikipedia.org/wiki/Dot-com_bubble
[17] https://deprogrammaticaipsum.com/geoffrey-g-parker-marshall-w-van-alstyne-sangeet-paul-choudary/
[18] http://paulgraham.com/vw.html
[19] https://en.wikipedia.org/wiki/WebObjects

other languages to generate them: Haml[20], Mustache[21], Emmet[22], Sass[23], LESS[24], CoffeeScript[25], TypeScript[26], and so many others are leaky abstractions built on top of the trinity so that developers could finally make the apps they needed within time and budget.

It is pointless to enumerate the gazillion forms these abstractions have taken shape. Still, it is worth mentioning one: a file format[27] (usually referred to as JSX or TSX, depending on whom you ask) merging the three elements of the trinity into a single file (something that Microsoft had tried 15 years before with those HTAs[28] for Internet Explorer 5.) Those JSX files are at the core of one of the hottest new trends of the 2010s: Single-Page Applications, or SPAs, built by a new generation of "front-end engineers" dealing every day with `npm` and its various quirks (and security issues.)

SPAs effectively moved the generation of HTML from the server to the client; usability was dammed. For a long time, servers processed requests and returned HTML, but after Douglas Crockford invented JSON, it became apparent that we could also return pure data instead. Given the new "AJAX" capabilities of modern browsers (that is, browsers that were not Internet Explorer 6) and the fact that we finally understood JavaScript (thanks, Douglas[29]!), developers gradually moved the generation of HTML from the server to the client. REST begat GraphQL[30], and the rest is history.

The thing is, SPAs are a terrible idea.

Moving text processing from the server to the client meant downloading tons of megabytes of (hopefully minified) JavaScript to customers and letting them deal with broken "Back" buttons. Of course, this also meant cheaper hosting for companies, which makes stockholders happy. Some SPAs require server-side processing, particularly for user authentication and authorization, and some other intensive operations, such as generating PDFs or processing video; otherwise, they defer all logic processing to the client.

The result is a terrible degradation in user experience, particularly for resource-constrained devices such as mobile apps and smart TVs, and for users in locations with poor connectivity, which is a lot of places. Still, developers in first-world countries with lots of bandwidth could not care less. Most of these apps seem to be built around a single assumption: let us just hope the user does not hit the back button, and everything will be fine.

We also got "real time" technologies to enhance our experience. We mentioned AJAX previously: another thing initially invented by Microsoft as an ActiveX component to empower its Microsoft Outlook web application on Internet Explorer, later adopted by Firefox and Safari. Then we got WebSockets, and its less well-known sibling, SSE[31] (Server-Sent Events,) a much simpler yet often overlooked alternative to WebSockets.

In the meantime, such is the importance of HTTP (and HTTPS) as a protocol for our modern world that a whole new platform (Kubernetes) was built to serve content by default on

[20] https://haml.info/
[21] https://mustache.github.io/
[22] https://www.emmet.io/
[23] https://sass-lang.com/
[24] https://lesscss.org/
[25] https://coffeescript.org/
[26] https://www.typescriptlang.org/
[27] https://facebook.github.io/jsx/
[28] https://learn.microsoft.com/en-us/previous-versions/ms536496(v=vs.85)
[29] https://deprogrammaticaipsum.com/douglas-crockford/
[30] https://graphql.org/
[31] https://germano.dev/sse-websockets/

port 80 (or 443, respectively.)

JavaScript became minified JavaScript which became WebAssembly which became WASM, and now we can compile Rust apps into WASM and run them into Kubernetes to render complete user interfaces... as HTML. A full SPA generated from Rust, hell yeah! Hey, Rust can even help you if you missed Flash movies[32] on your website.

Knock knock, who is that?

Hold my beer: Hotwire[33] proposes the revolutionary idea of... sending HTML over the wire.

Crickets.

Meanwhile, Ruby on Rails, PHP, ASP.NET, and Java developers are shaking their heads in dismay while they deliver an umpteenth server-rendered web application to their customers. Who woulda thunk? Servers are cool again[34]. But running monolithic apps in Kubernetes remains challenging; good luck deploying that Mastodon instance on your cluster.

But things are not going well. TimBL has regrets[35]. Jake Archibald is swearing[36]. Brian LeRoux is sad[37]. PPK does not know[38] whether to teach Flexbox or Grid CSS first. MIT is withdrawing[39] from the W3C; Safari (or Chrome?) risks[40] becoming the new Internet Explorer; the Google Gruyere[41] app shows that security was always an afterthought. In the meantime, ChatGPT is creating[42] a new HTML standard element for music, and developers are still unsure[43] which HTTP headers are case-sensitive.

One of the original goals of Ted Nelson's Xanadu project was to avoid broken links; on the Web, the best we can aspire to is to have our web page crawled by the Internet Archive's Wayback Machine[44] and to be able to point to it instead in the future. So please ensure your 404 page is user-friendly; it will be shown very often.

Maybe Web browsers as we know them are just a first step in the good direction. Maybe Kubernetes is another step in that direction. Maybe there will be an upcoming technology (open and standardized) that will make real web applications possible, with a single programming language, a unified debugging experience, and a mandatory plugin for Visual Studio Code. With it, developers will *finally* deliver apps on time and within budget, using a single, modern programming language running in a distributed platform that serves users with structure, style, and behavior.

Or maybe, just maybe, the paragraph above is the reason why so many billion dollars have been invested in innovation around web technologies. As a community effort, the Web (HTML, CSS, JavaScript) is already the best platform it could ever be, and all those leaky abstractions built on top (Haml, Sass, TypeScript) are not only necessary but also unavoidable.

[32] https://ruffle.rs/
[33] https://hotwired.dev/
[34] https://css-tricks.com/servers-cool-once-again/
[35] https://www.vanityfair.com/news/2018/07/the-man-who-created-the-world-wide-web-has-some-regrets
[36] https://alistapart.com/article/application-cache-is-a-douchebag/
[37] https://indieweb.social/@brianleroux/109469337914637199
[38] https://front-end.social/@ppk/109444203203546926
[39] https://mastodon.social/@robin/109524929231432913
[40] https://www.theregister.com/2021/10/22/safari_risks_becoming_the_new_ie/
[41] https://google-gruyere.appspot.com/
[42] https://twitter.com/matvelloso/status/1600169072586391556
[43] https://misfra.me/2022/http-request-case-sensitivity/
[44] https://web.archive.org/

This author is praying for the return of server-side rendering, with good old Mustache templates, short and sweet vanilla JavaScript[45] (seriously, you might not need jQuery[46] anymore these days,) and small and lean HTML pages with just enough CSS to get things done, one page at a time. If you are a front-end developer, consider stopping using JavaScript for something it was never meant to do. Correction: consider using the Web as it was meant to be used: as a set of linked pages. Stop manipulating the Back button with JavaScript; browsers can do that job perfectly well; thank you so much.

The Web was conceived as a document browsing environment; stretching it in other directions, particularly for software applications, was a mistake.

(Just like misusing its good name in this whole "web3" nonsense[47] which has nothing to do with the Web.)

Cover photo by Denny Müller[48] on Unsplash[49].

[45] http://vanilla-js.com/

[46] https://youmightnotneedjquery.com/

[47] https://web3isgoinggreat.com/

[48] https://unsplash.com/@redaquamedia?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[49] https://unsplash.com/photos/JySoEnr-eOg?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Steve Jobs

Adrian Kosmaczewski

January 2nd, 2023

In 2019 an extraordinary team of web development and design celebrities gathered in the offices of CERN in Geneva, Switzerland, to build an emulator of the original web browser created by Tim Berners-Lee, known as WorldWideWeb. This emulator is available online[1] and, fittingly enough, runs on a modern[2] web browser.

The team included John Allsop[3], Kimberly Blessing[4], Mark Boulton[5], Martin Chiteri[6], Jeremy Keith[7], Craig Mod[8], Angela Ricci[9], Brian Suda[10], and Remy Sharp[11], who kept a diary[12] of the project in his blog.

The original WorldWideWeb browser, however, was written in Objective-C[13] and ran on a NeXT Computer. The source code of version 0.15 is nowadays available on GitHub[14], but building and running it is not a straightforward task, even if a NeXT emulator such as Previous[15] could be used for that matter.

The WorldWideWeb emulator gives a good overview of the NeXT UI. But what about the rest of the system? What was it like to use a NeXT Computer 30 years ago? How can we get

[1] https://worldwideweb.cern.ch/browser/
[2] https://deprogrammaticaipsum.com/issue-32-modernism/
[3] http://johnfallsopp.com/
[4] https://www.kimberlyblessing.com/
[5] http://www.markboulton.co.uk/
[6] http://geek.co.ke/about/
[7] https://adactio.com
[8] https://craigmod.com
[9] https://gericci.me
[10] https://suda.co.uk
[11] https://remysharp.com/
[12] https://remysharp.com/2019/02/18/cern-day-5
[13] https://deprogrammaticaipsum.com/brad-cox/
[14] https://github.com/cynthia/WorldWideWeb
[15] http://previous.alternative-system.com/

an idea of how revolutionary those machines were? For that, it is much simpler to go back to 1992 and watch a demo of NeXTSTEP Release 3[16] by Steve Jobs himself.

This month's video[17] is roughly divided into three sections: the first is about the NeXT productivity environment, the second one is about networking and collaboration, and finally, the third one showcases a demo of how to build a NeXT application from scratch.

The first two sections are anecdotal; apart from seeing a picture of his son Reed Paul Jobs, or learning about the multi-protocol capabilities of the NeXT networking stack (including fax support!) there is not much to learn. If anything, the live document update feature brings back memories of Windows' own DDE and OLE capabilities, which later begat COM component technology, the ancestor of .NET. But I digress.

Current macOS (or as it was previously known, Mac OS X) users will recognize lots of UI elements popping all over the place: the font and color inspector, the services menu, and so on. Let us try to forget Steve's choice of the "Stencil" font; fashion trends have certainly changed since the early nineties.

Modern users of computer systems might not realize how revolutionary it was to see window contents redrawn while moving them around, or how much WYSIWYG was pervasive throughout the interface. We take those things for granted these days, but by all means, they were not common at all.

From a software development standpoint, the exciting stuff begins at the presentation[18] of Interface Builder, where we learn that

> And I'm going to run an application called Interface Builder. Now, think of NeXTSTEP, which is NeXT's object-oriented environment, as an object-oriented cake. I'm about to show you just the frosting on the cake. Many people have tried to copy this frosting, but what they found out is without the object-oriented cake underneath, it just doesn't work.

Using Interface Builder, Steve drags and drops elements on the UI and wires them automatically to the tables and fields of a Sybase database. This is done thanks to a framework he calls "DatabaseKit," a name later shared by at least two abandoned[19] projects[20] for macOS and iOS. As expected (in 2022), DatabaseKit abstracts developers from the intricacies of the underlying database engine and provides a path to migrate applications to other RDBMS systems, such as Oracle or DB/2.

Again, nothing to phone home about in 2022, but revolutionary 30 years ago.

Steve finally uses SoftPC by Insignia[21] to run DOS apps, and one should not miss his face while opening Lotus 123 at minute 31:25[22] of the video. Of course, being compatible with DOS was very important in 1992; after all, even an operating system such as IBM's OS/2 included DOS compatibility.

The Object-Oriented hype train[23] of the era brings the end words[24] of the video:

[16] https://www.youtube.com/watch?v=rf5o5liZxnA
[17] https://www.youtube.com/watch?v=rf5o5liZxnA
[18] https://youtu.be/rf5o5liZxnA?t=1397
[19] https://github.com/vapor/database-kit
[20] https://github.com/fjolnir/DatabaseKit
[21] https://en.wikipedia.org/wiki/SoftPC
[22] https://youtu.be/rf5o5liZxnA?t=1879
[23] https://deprogrammaticaipsum.com/the-hype-cycle-of-oop/
[24] https://youtu.be/rf5o5liZxnA?t=2096

The Object is the Advantage.

Couple this video with Steve Jobs talking about the app store[25] in 1983, and let's reflect on the fact that somebody, somewhere, somehow, is building the next big thing as you read these words.

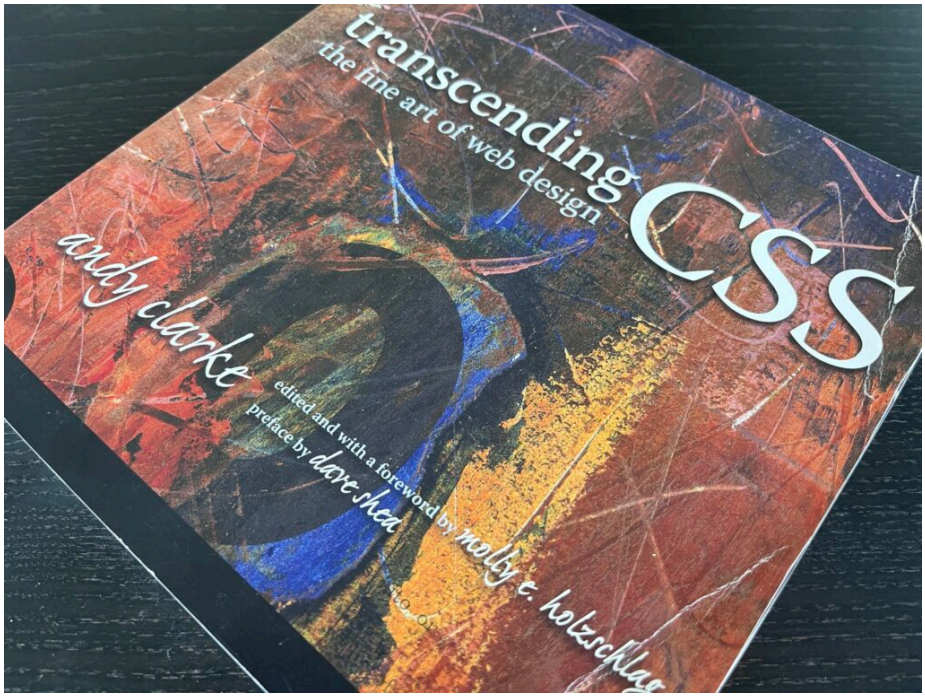
Cover snapshot by the author.

[25] https://www.youtube.com/watch?v=jmody6wQrLA

Andy Clarke

Adrian Kosmaczewski

January 2nd, 2023



We have often said in the pages of this magazine that some books carry with them the Zeitgeist of their era. Examples are Bruce Tate's "Beyond Java," Joe Armstrong's "Programming Erlang," and Toby Segaran's "Programming Collective Intelligence." Such books have a tremendous impact upon publication, freezing in words not only a valuable body of knowledge, but also the spirit and promise of a new direction for the industry. Even if the APIs they describe become obsolete over time (which is mainly unavoidable), they remain as hallmarks of an era, valuable witnesses of the preoccupations and needs of practitioners at the time of their publication.

This month we will review another of those books, "Transcending CSS: The Fine Art of Web Design"[1] by Andy Clarke[2], published in 2006 by New Riders. Those were the times of Web 2.0, a rebirth from the ashes of the excesses of the dot-com boom of the previous decade. A moment to start anew, to apply the lessons learned, and to rebuild a new Web.

In those days, the state of the Web was not pretty; Internet Explorer 6 was already showing signs of decadence, yet it was far from being a priority for Ballmer's Microsoft[3]. Its support for standards was mediocre at best, and upcoming products such as Apple Safari (whose WebKit rendering engine was a fork of KDE's Konqueror[4]) and Mozilla Firefox (the most visi-

[1] https://www.peachpit.com/store/transcending-css-the-fine-art-of-web-design-9780321410979
[2] https://stuffandnonsense.co.uk/
[3] https://deprogrammaticaipsum.com/where-does-microsoft-want-to-go-today/
[4] https://apps.kde.org/konqueror/

ble result of the "open-sourcing"[5] of Netscape's code in 1998) were showing that alternatives were not only possible but also desirable.

Those were the times of Prototype.js[6], script.aculo.us[7], and jQuery. Jesse James Garrett had just given the Ajax name to "a new approach to web applications."[8] Peter-Paul Koch (also known as "PPK") was busy explaining how Quirks Mode[9] worked. New cross-platform JavaScript frameworks were helping developers work around the quirks of IE 6, unifying experiences and reducing development times. Frameworks like Ruby on Rails or Django were breaking waves, enabling unprecedented productivity and making the Web a fantastic place to make things again.

But in terms of design, who am I kidding; many of us still used the `<TABLE>` tag for the layout of our pages. I know it very well; I was one of those wannabe designers stuck in the past. In our defense, Your Honor, I will argue that those who started designing web experiences in 1997, like me, well, we did not have many tools at our disposal back then when Netscape 3.0 was still around. So, well, tables it was. *Mea culpa:* for a long time, CSS was, for me, just a fancy way to remove `<FONT>` tags and `ALIGN=` attributes on paragraphs. Ouch.

Fast-forward 10 years, and in 2007 the landscape was different and much more promising. We were at the point where CSS 3.0 was starting to become widely available on browsers, which meant we could leave behind tables for layout altogether. And Andy Clarke's book was precisely the book that showed me how.

Andy was part of a larger group of web designers who started understanding the power of standards to make better websites. One of the most striking examples of that era was, without any doubt, Dave Shea's CSS Zen Garden. Maybe some of the readers of this piece will remember; it was a website whose premise was to showcase what CSS was capable of. Every link would load a different CSS stylesheet, always applied to the same standards-compliant HTML5 website. CSS Zen Garden is still online[10] at the time of this writing, and you can visit it just like fifteen years ago. Viewing the HTML source of the site shows a clean construction of intertwined `<DIV>`, `<UL>`, and `<SECTION>` tags, neatly woven into one another, describing the hierarchy and structure of a web page.

Not its look and feel; its structure. The core seed of progressive enhancement[11] as a design principle was laid out for the first time.

This is a fundamental concept: it felt like we were beginning to understand what this HTML thing was helpful for, *finally.* CSS was not only able to provide `font-family` or `align` information, but also to lay out those elements on a page, next to one another, on top of another, fixed on a specific position, or flowing as the user scrolled. The possibilities were endless.

Andy Clarke identified a significant problem with CSS, however. Designers are not developers, and CSS as *a language* was not a designer-friendly technology. He takes time to explain the meaning, the inner working, and the logic of CSS to an audience he knows very well; the book seems to be written for him, primarily, or at least for a younger version of himself. Such is the working progress that leads to many masterpieces, and this is no exception.

[5] https://deprogrammaticaipsum.com/open-always-wins/
[6] http://prototypejs.org/
[7] http://script.aculo.us/
[8] https://web.archive.org/web/20150910072359/http://adaptivepath.org/ideas/ajax-new-approach-web-applications/
[9] https://quirksmode.org/
[10] https://www.csszengarden.com/
[11] https://en.wikipedia.org/wiki/Progressive_enhancement

"Transcending CSS" is another work of art comparable to "Designing Interfaces" by Jenifer Tidwell, which we reviewed[12] precisely one year ago. The book is profusely illustrated and delightfully printed.

Dave Shea's work led to the 2005 book "The Zen of CSS Design"[13] written together with Molly Holzschlag... Molly edited and wrote the foreword to Andy's book, and Dave wrote the preface. We are all in good company here.

The evolution of CSS did not stop after the book's release; far from that. "Transcending CSS" was released the same year as the iPhone, an invention that would put a fully-working web browser in people's pockets for the first time. In 2010, Ethan Marcotte[14] coined the term "Responsive Web Design"[15] to describe a technique consisting of using media queries and rules to select different stylesheets depending on the viewport size. The stage was set to create beautiful, flexible, and accessible user interfaces for any device, from the largest of TVs to the smallest of smartphones.

Andy Clarke revisited his book in 2019, and this new edition is available for everyone to read online[16]. If your work involves generating HTML content of any kind in any way, you owe it to yourself to take the time and dive into the mind of a man who can fight against King Kong[17] and save the world. And then read everything else[18] he has written; you will thank me later.

(And I promise I have not used a single `<TABLE>` to lay out any HTML page since I read this book.)

Cover photo by the author.

[12] https://deprogrammaticaipsum.com/jenifer-tidwell/
[13] https://www.amazon.com/exec/obidos/ASIN/0321303474/
[14] https://ethanmarcotte.com/
[15] https://alistapart.com/article/responsive-web-design/
[16] https://stuffandnonsense.co.uk/transcending-css-revisited/index.html
[17] https://mastodon.social/@malarkey/109466581363187302
[18] https://stuffandnonsense.co.uk/books