LARGE-SCALE COLLABORATION IN GRAPHIC DESIGN COMMUNITIES OF PRACTICE

AUTHOR
Julie Blanc

PUBLICATION DATE
02 June 2024

In graphic design, collaboration is often understood as direct participation, typically limited to specific time frames and spaces, such as workshops or projects involving "public" contributions to produce a design. Nonetheless, different forms of collaboration exist, especially at a larger scale within communities of practice that share a common culture. This article explores how open-source culture and its collaborative aspects are being integrated into various graphic design practices, especially in web design. It reflects the progressive adoption of coding, along with the growing number of graphic designers [1], mainly in publishing practices [2].

Free software culture revolves around the use of open-source software and programs, characterized by the freedom to access, modify, and redistribute their source code. According to the Free Software Foundation, software is considered 'free' if it grants users four essential freedoms: to use (freedom 0), to copy (freedom 1), to study (freedom 2), and to modify the program (freedom 3). Beyond software itself, Open-Source culture represents a social and philosophical movement advocating for a creative and ethical approach to technology and work. Free software activists are particularly committed to empowering users to control their own computing and to fostering technologies that benefit society.

These ideas are reclaimed and supported by many graphic designers who favor the use of code in their work [3]. Their goal is to establish a technical culture within graphic design, based on the transmission of technical knowledge that allows it to be integrated into a valorisation process [4]. The integration of programming into graphic design is also significant as an experimental and creative practice of technology. This creation process focuses on artistic know-how, exploring formal and aesthetic aspects, and discovering of unconventional tools [5]. More specifically, free software is a successful example of a work approach that differs significantly in the context of 'post-production,' a term coined by Andrew Blauvelt to describe the diminishing distinction between production and consumption, or creation and reproduction [6].

While the collaborative aspects of open-source culture are often emphasized, they are rarely discussed specifically in the context of graphic design, where participants are not primarily programmers. Nonetheless, the core principle of free software – the obligation to make source code

available and allow modifications and redistribution – promotes cooperation among developers, enabling them to contribute to a collective work. This collaborative effort reflects the values of mutual aid and the pooling of individual contributions.

The aim of this text is to examine how the culture of free software – characterized by collaboration, sharing, and collective learning – is embodied in programming practices within web and graphic design. To achieve this, I will explore several thematic concepts that are discussed in greater detail in my thesis [7]. My perspective is informed by my involvement in the informal collective PrePostPrint [8] and in the development of the Paged.js tool. Consequently, this text focuses primarily on practices related to web technologies, which are used interchangeably for web and print publishing.

This proposal offers a critical analysis and an opportunity to discuss the practices of free culture in graphic design through concrete examples. These examples highlight the use of code not merely as a DIY or "hacking" practice, but as a cultural and social one focused on collaboration.

Opening And Modifying A Code: Markup And Text Formats

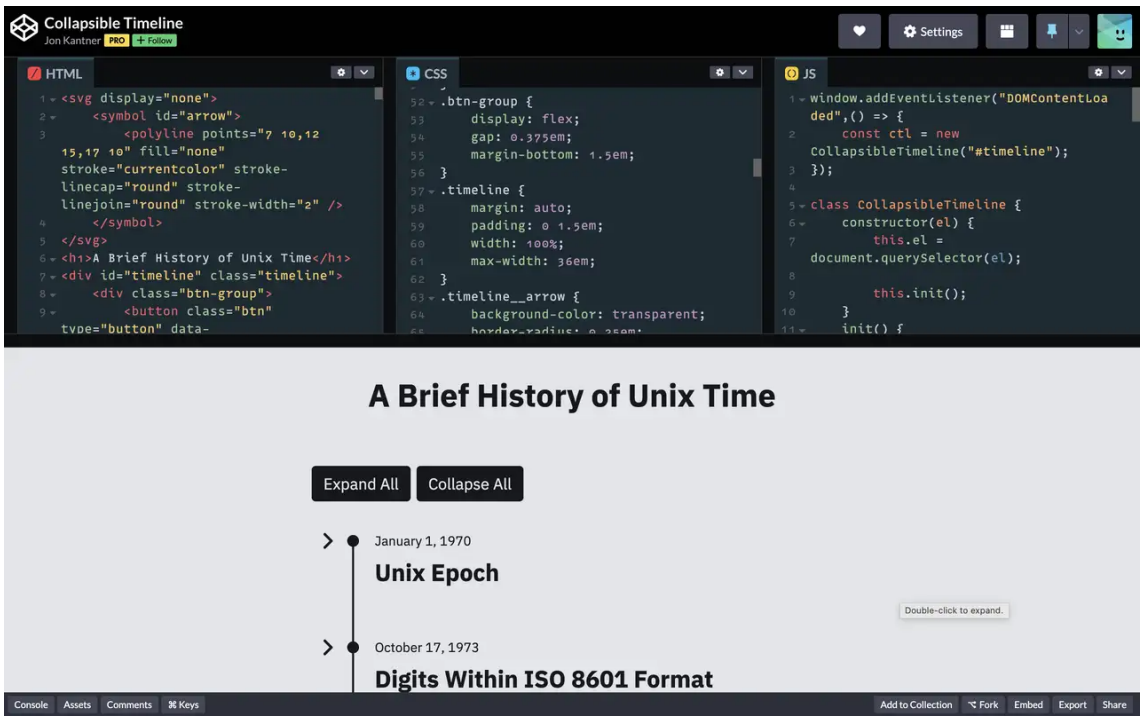
The vast majority of coding used by graphic designers focuses on web technologies, primarily HTML and CSS. The ability to integrate and publish various media – such as videos, generative processes, and animations – in a unified space makes the web an essential platform for publishing, which is of great interest to graphic designers.

HTML, which stands for HyperText Markup Language, is used to structure a web page. For example, the `<body>` tag indicates where the content begins, while `<h1>` denotes a level 1 title. CSS, or Cascading Style Sheets, is used to define a document's presentation and style, such as setting the font for a title or the color of an element. Thus, a website consists of a set of HTML files linked together by hyperlinks and formatted using CSS [9].

HTML and CSS are standard and descriptive languages that are relatively simple to learn and use. They don't require in-depth knowledge of algorithms or complex compilation processes; all you need is a web browser to interpret them, making them accessible to everyone. According to Alexendra De Visscher, using web technologies to design hybrid publications allows designers to 'free themselves from WYSIWYG interfaces through explicit, formal syntaxes that reveal the semantic structure of content [10]'.

The unique relationship between coding and the web arises from its direct connection to text mode, eliminating the need for specialized software and facilitating easy copying and sharing. Designers can readily copy and paste code snippets created by others, regardless of their programming skill level. This ease of reuse has cultivated a whole 'sharing ecology' based on how easily resources can be made available. Although this approach is less ambitious than a 'fork'[11], it represents the initial form of collaboration between designers and developers. (From now on, we will use these two terms interchangeably.)

Publishing code on dedicated platforms like CodePen signals a desire for it to be used and modified by others. Graphic designers can leverage a wealth of online resources offered by a community of both expert and amateur web developers.



Screenshot of a project on CodePen, a community site for sharing HTML, CSS, and JavaScript code that functions as an interactive code editor.

This culture of appropriation and reuse sparks debates about copyright and the sharing economy. The notion of singular authority traditionally associated with graphic design is challenged by the sharing of tools and the use of code.

Although the romantic notion of the genius creator has long been dispelled [12], it still lingers in the field of graphic design. However, recycling pieces of code is a concrete illustration of the idea that forms are derived from various sources, aggregated, and rearranged. As the psychologist Lev Vygotsky and later researchers have noted [13], any creation, no matter how individual, always includes a social component and reflects the anonymous collaboration of others. The use of web technologies emphasizes this social dimension of creation.

On a different note, the use of code allows for concrete benefits from the ecosystem of collaborative tools and methods developed over decades to facilitate programming. A key component of this collaboration is the use of distributed version control systems, with Git being the most well-known. Git enables developers to manage changes to source code – merging changes, resolving conflicts – and thus facilitates collaborative coding. Project management platforms like GitHub, GitLab, Bitbucket, and Codeberg help coordinate development efforts, track issues, and facilitate contributions from community members. As a result, a specific vocabulary related to programming is gradually entering the field of graphic design, including terms like fork, commit, version, and issue.

These collaborative tools and practices are central to the controversies surrounding free software culture. Since the 1990s, there have been two opposing conceptions: free software and open-source [14]. Conveyed by Linus Torvalds, the founder of Linux, open-source is a development method that is based on code-openness, without necessarily aligning with the values of free software culture [15]. This movement exploits a discourse aimed at reconciling the principles of openness with the economic and commercial stakes of capitalism. The idea is that open-source can serve as a viable alternative, both commercially and technically, to proprietary software. Hence, open source focuses more on the technical aspects of software development and the organization of community work to enhance software efficiency and reliability.

Thus, accessing to source code is a rather practical matter than an ethical one for the open-source movement. The free software movement is fundamentally a social matter, deeply tied to the notion of liberty. However, these two movements have one thing in common: the emergence of new forms of coordination that cannot be reduced to formal organizations, where collaboration is encouraged and knowledge is widely accessible.

The collaborative, horizontal working methods inherent in open-source culture are transforming the practice of graphic designers, shifting their focus from a model centered on singular authority to one based on collective, open contributions.

Organize Your Code: Practices For Coded Writing

Writing code is not just a technical skill; it is also a social practice that considers the community in which graphic designers operate. The ability to share code influences how it is written. In this way, graphic designers not only create graphic objects through coding but also work to shape the code itself, ensuring it is intelligible and readable – an essential condition for sharing.

In programming, the need to write clear, rigorous, and well-structured source code is an implicit rule. The clarity is vital not only for others to understand the code but also for developers to navigate their own work effectively. As a result, various coded writing practices are implemented to enhance the readability and accessibility of code.

Particular attention is given to the semantics of the code. For instance, the rule of relevance is crucial when naming files, HTML classes, JavaScript functions, or custom properties. A CSS variable named 'baseline' serves as a semantic clue for its use in aligning elements according to a baseline. Reversely, a naming system that is overly abstract or personal can hinder collective understanding and re-appropriation of the code.

The visual and graphic structure of the code is equally important in reflecting its organization. Using line breaks, indentations, nesting, and spacing can create visual metaphors or representations of the relationships between various elements, making the structure more apparent. It is also common to break down code into manageable units by organizing it into files dedicated to specific functionalities.

In this sense, writing code is an act of composition [16]. It is particularly noteworthy to link the readability of code – especially its structure – with one of the primary objectives of graphic design: the readability of information, particularly within the architecture of information and graphic system setups. Therefore, the core of graphic design practices generally involves structuring, organizing, and arranging elements within a format to enhance their visibility and readability.

In graphic design programming practices, the act of composition is twofold: it occurs both in the forms produced and in the code itself. By writing code, graphic designers engage with their peers, moving beyond simply writing to produce. Instead, they write for and with others, aiming to create 'beautiful code' that is 'readable and understandable' and can be shared with the community.

This idea embodies a certain reciprocity. By assuming that their peers share the same commitment to making code legible, graphic designers can appropriate code written by others. By addressing their code through shared 'best practices' – what we refer to as codified writing practices – developers ensure reciprocity in the development of their resources.

In an interview conducted as a part of my thesis, designer Benjamin G. explains: "Other people have to be able to understand my code. (...) Look, I just took a part of the code that someone else wrote and I am happy to be able to understand it. (...) It forces me to adapt good practices as well, whether my code is open access or not [17]".

```
#page-article {
    color: var(--color-article);

    header {
        padding-top: calc(var(--baseline)*0.75);
        padding-bottom: calc(var(--baseline)*0.5);
        border-bottom: 1px solid var(--color);
        font-family: var(--font-title);
        display: flex;
        justify-content: space-between;

        button {
            color: var(--color);
        }
    }
```

(S)CSS code excerpt showing the use of variables (var(--color-article)) and indented code structuring

These "good practices" can be seen as community-shared rules, whether formally or informally, learned through experience, documentation, or collaboration. That's also what graphic designer Amélie Dumont expresses: "Confronting the idea of sharing my code with other people, helped me progress and understand... [For instance, to realize that] (...) the basic structure of HTML and CSS is not at all useless. Having files that are as clear and simple as possible makes it easier to work with other people [18]."

Making code readable is closely tied to the culture of free and open-source software, where sharing and clarity are fundamental values. By adhering to a set of shared and good practices, it is possible to facilitate mutual understanding within the community.

Graham Button and Wes Sharrock, in their in-depth study of computer programming, they emphasize that one key identification criterion for professional programming is its orientation toward community, arguing that practitioners as professionals must take into account the interests of the other people within the community. Thereon, 'Learning to write a computer program means learning to write it in a way that is intelligible to a community of practitioners [19].' This understanding is essential for fostering collaboration.

Documenting: Multiple Forms Of Sharing

While participating in source code creation may seem obvious, collaboratively developing an open-source tool involves other essential tasks that may seem minor but are often time-consuming: 'responding to help requests, offering advice, reviewing contributions, assessing translations, and creating documentation, among others [20]'. This work – often overlooked or undervalued – is vital for the effective adoption of any tool or software. I'll use documentation and its various forms in both software design and resource sharing as an example.

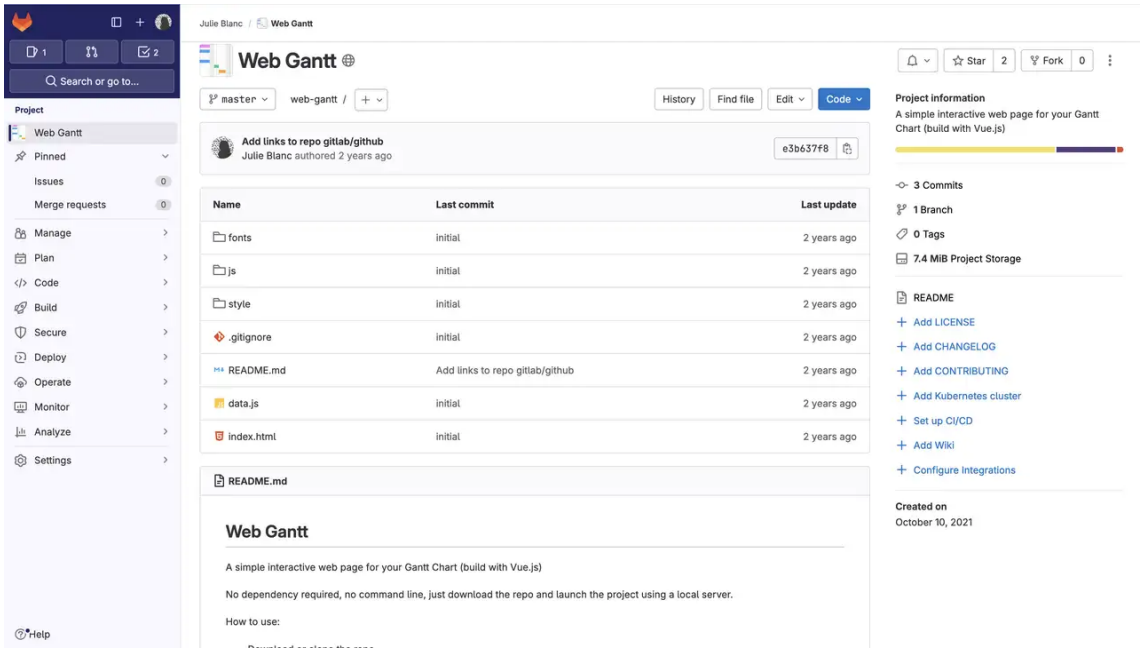
Technical documentation and instruction manuals are the most common forms of sharing. They contain detailed information about internal functioning of a program, its use, setting, and development. They also include descriptions of the software's features, examples of use, installation guides, code references, technical specifications, and sometimes even tutorials to help users and developers understand how to use the software. Although software with graphical interfaces has led many graphic designers to work without such documentation, it remains an essential resource in

programming. The well-known acronym RTFM ("Read The Fucking Manual") is often used in the field to emphasize reading documentation as the first step in tackling any technical issue.

PageTypeToPrint is a web and print layout tool originally designed as a template for formatting student theses at the École supérieure d'art et de design des Pyrénées, for which Julien Bidoret provided extensive and accessible documentation [21]. Written to be easily understood by students with limited coding skills, this documentation greatly facilitated the tool's adoption beyond the school it was initially developed for.

Wiki or knowledge bases represent another valuable form of documentation. These resources are often collaborative, allowing users to contribute and update information on a specific topic or tool. They may include articles, tutorials, and guides. Wikipedia is the most well-known example of this approach, but it can also be used on a smaller, more localized scale. For instance, the École de Recherche Graphique de Bruxelles (erg) has its own wiki, which provides information about the school and resources for workshops and courses.

In programming, source code for free and open-source projects is often shared via platforms like **GitHub or GitLab**. These platforms rely on Git version control software, which provides access to different versions of the source code. By convention, each project's source code folder (or 'directory') includes a markdown text file named 'README.md,' which serves as an introductory guide and outlines the project's features. This file acts as a concise documentation for the project and often serves as an entry point to the source code, which is why it's given significant importance.



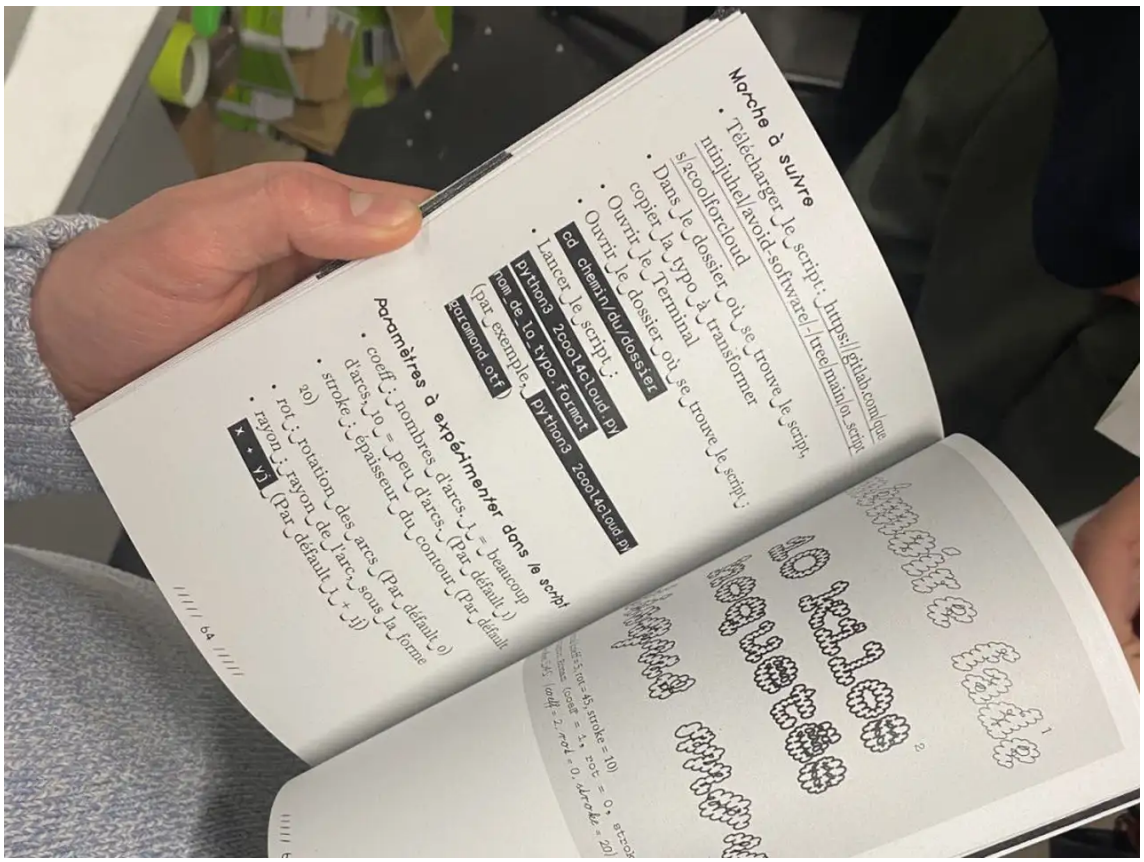
Screenshot of a code-directory on GitLab

Many graphic designers and programming collectives have accounts on these platforms (or even host their own instances), using them as documentation for their work. Open-Source Publishing has taken a similar approach directly on its website, where each project is displayed with an 'About' section (a brief project overview), an 'Inside the Repository' section (the project's source code), and a 'Log' section (a history of modifications with comments). This setup makes the project's various stages and developments visible.

Screenshot of a project view on the Open-Source Publishing website (osp.kitchen)

Apart from the practices of graphic designers who program, documentation can be an unfamiliar and sometimes intimidating aspect of the writing process. Yet it's essential for building collective practices that share common resources.

One promising approach is to use documentation formats that are more accessible and familiar to non-coding designers. With this in mind, Sarah Garcin, Quentin Juhel, and Emma Sizun created Avoid Software. Everyone has it [22], a web and print **fanzine** developed specifically for the Open-Open event held on May 11-12, 2023, at ESAC Cambrai. This publication, with freely accessible source code, offers 'free and open-source methods, scripts, and hacks useful to any artist, designer, or student in art and design schools.'



Fanzine Avoid Software par Sarah Garcin, Quentin Juhel and Emma Sizun

Experience-based stories, like blog posts, project diaries, and articles, also serve as a form of documentation that contributes to building a programming culture within graphic design. For example, Benjamin G. and I published a series of three blog posts on the Sciences Po médialab website redesign project [23], where we outline the design's chosen principles of sustainability and how we implemented them, including CSS code examples. Similarly, Nicolas Taffin, graphic designer and co-founder of C&F publishing, regularly shares 'making-of' posts on his blog, detailing the design process behind his books [24].

This type of publication can also focus on a more reflective sharing of practices. During a residency at Open-Source Publishing, Manetta Berens took the opportunity to reflect on \'ten years of web-to-print practice\' within the collective, using blog posts in the form of interviews and compiling a list of annotated, contextualized repositories [25].

Finally, one doesn't need to be a 'confirmed expert' to share their experiences. Recently, graphic designer Delyo Dobrev published a logbook documenting his attempt to adapt Tactic\'s Cursors magazine to a layout programmed with Paged.js [26].

In programming-oriented graphic design, documentation practices are essential – they support learning and foster communities of practice.

Adding Functionality: Plug-in Design

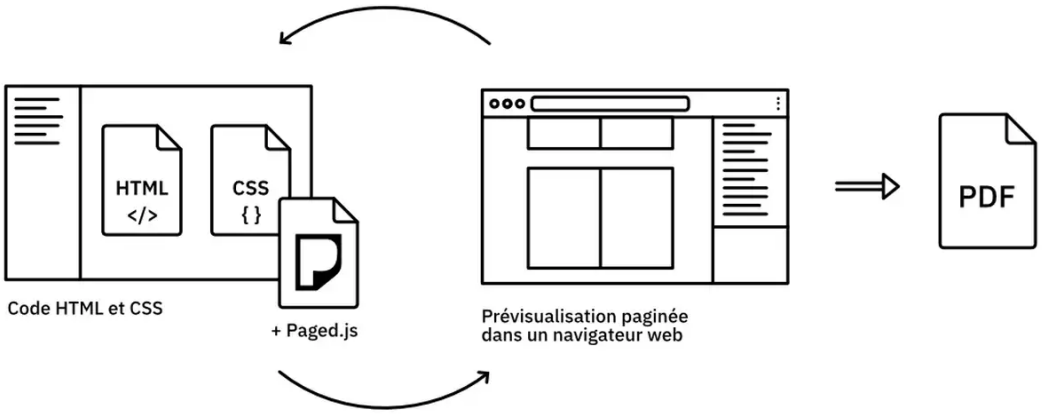
The integration of programming into graphic design has sparked the desire to create custom tools. Graphic designer Kévin Donnot states, 'A graphic designer-hacker can write programs by hand to meet their specific needs, reflecting their authorial identity [27].' As a result, most tools developed by graphic designers are intended for personal use or specific projects.

The Luuse collective developed H.I.R.P. (Honey, I Resized the Poster), a web-based tool for generating various print formats from the same content, specifically used for designing posters for European

Digital Rights (EDRi) [28]. Similarly, the Bonjour Monde collective created dataFace [29], an experimental tool for typographic manipulation with variable font properties, used in workshops to explore new typographic forms. These projects exemplify custom-made creation, where the tools themselves become part of the graphic designers' production. However, due to their specialized design, these programs are rarely utilized outside their intended contexts.

To envision the collaborative design of larger-scale tools and promote production autonomy in graphic design, we must move beyond the concept of auteur (which, though interesting, is not the focus of this text). Instead, we aim to design generic tools that can be adapted to various situations and utilized by a broader community of practitioners. Specifically, we are interested in exploring ways to enhance the functionality of existing open-source software.

For example, Paged.js is a free, open-source tool that allows web technologies to be used for printing by implementing W3C CSS specifications that are otherwise unavailable in web browsers. It is delivered as a code file written in JavaScript [30]. The file is added to any HTML page along with its CSS style sheets. When the web page is rendered in a browser, it displays a paged preview with the print-specific CSS applied. Once formatting is complete, the PDF can be generated using the browser's print options.



How Paged.js works

To comply strictly with CSS standards [31], certain page layout functionalities have been excluded from Paged.js, as they are not yet supported by the CSS language (for example, full-page images or margin notes). As a result, graphic designers can develop plug-ins – scripts designed to integrate with the tool via a dedicated interface – to add new functionalities without modifying the main software's source code.

This involves developing code that aligns with the main software: it should be generic enough to adapt to various situations, yet specific enough to address the unique needs of each project. For example, Julien Bidoret, a teacher at the École supérieure d'art et de design des Pyrénées, created a script for imposing pages in booklets for printing. Similarly, Julien Taquet, a designer with the Collaborative Knowledge Foundation, developed a script to position images on the page. Some of these plug-ins extend beyond page layout needs; for instance, Nicolas Taffin, a graphic designer and co-founder of C&F Publishing, created a script that reloads the web page to the same scroll position it had before reloading, making it easier for graphic designers to switch between writing code and previewing it in the browser.

Other scripts can be more specific, like the one developed by Gijs De Heij, a graphic designer at Open-Source Publishing, which allows text to be displayed on the spine of a book. To transform these scripts

into 'plug-ins,' they often need to be reworked to ensure they are more generic, making them usable across various projects of the same type.

Kin Arts Almanac, State of the arts (2024), layout by Open-Source Publishing. On the right: screenshot of the layout in Paged.js

By being written in a sufficiently general and adaptable manner, these plug-ins can be easily utilized by other graphic designers. Sarah Garcin positioned the margin notes aligned with the note callouts for the layout of the book Controverses, for an instruction manual for the Forccast program and for Presses de Sciences Po. To achieve this, she used a plug-in called 'margin-note' that I developed a few weeks earlier for another project.

Clémence Seurat, Thomas Tari, Controverses. Mode d'emploi, Presses universitaires de Sciences Po (figures/controverses-typo.jpg), designed by Sarah Garcin.

Most scripts additional to Paged.js are created to address specific needs encountered by graphic designers in their projects. However, there is a inventory work to do, since designers often lack the time or opportunity to share, generalize, and document the scripts they develop for commissioned or personal projects [32]. The accumulation and articulation of these scripts contribute to the tool's design and embed this design within the community's collective practices. In short, plug-in-based design allows these productions to be shared, enabling their reuse in individual projects when similar situations arise.

Standards For "Writing The Code Of Code"

Standards provide a common, documented framework designed to harmonize a sector of activity. They emerge from recommendations and gain traction when a critical mass of community members adopts them collectively [33]. In information technology, standards enhance compatibility between different programs and hardware, making interoperability essential for resource sharing and collaboration. By bringing stability and viability to the tools, objects, and publications they support, standards play a crucial role in fostering effective practices.

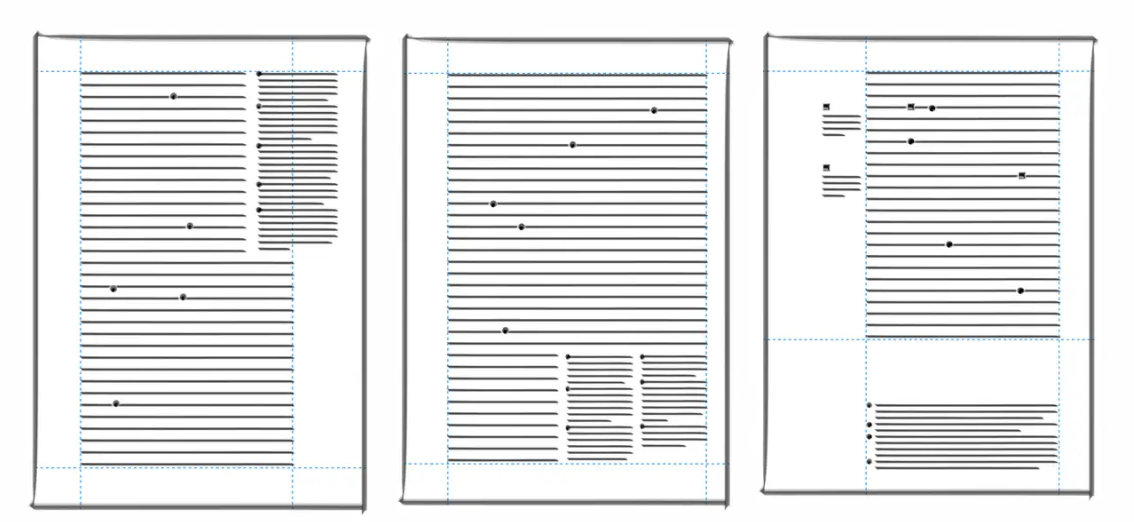
For example, the web relies on backward compatibility, meaning that nothing should break over time – a website coded twenty years ago should still display correctly today [34] (assuming it hasn't been removed from its server, which is another matter). This principle allows us to still access and enjoy the first web page published in 1991, along with its source code [35]. This longevity is possible because HTML and CSS are standards built around the concept of progressive enhancement through accumulation. Their specifications are published publicly, allowing them to be utilized by anyone in the same way, including web browser manufacturers.

In contrast, Flash-based creations – once dominant in web multimedia content design – are now inaccessible on most web browsers. This is because Flash was a non-standard technology that required a special interpreter maintained by Adobe. With the discontinuation of Flash software, it's no longer possible to access the thousands of creations developed in this format.

Let me return to web technologies: Their development relies on an open, community-driven standardization process led by the World Wide Web Consortium (W3C). For instance, the CSS Working Group (CSSWG) has been actively enhancing the CSS language since 1997, issuing recommendations that define its syntax, and outlining the expected behavior of each property used in web browsers. As of December 2022, the group had 147 members from various organizations and invited experts [36]. Each proposal, whether based on feature requests or identified needs, is debated and then either adopted or rejected by a community that includes company representatives, browser developers, web developers, researchers, and others [37].

This collaboration among diverse individuals and organizations is the source of the evolution of CSS. If we're interested in this aspect of standards design because it elevates the question of how graphic designers can play a role in building their own tools, bringing collaboration to a meta level within design.

Let's pick up where we left with the example of Paged.js. In previous sections, we discussed that many features essential to the compositional work of graphic designers were overlooked when the W3C CSS specifications were developed. Although plug-ins can address certain gaps, they do not provide a sufficiently permanent solution and are tightly linked to specific tools. Julien Taquet and I therefore decided to draft new specifications, focusing initially on page note layout. While footnotes already have dedicated syntax in the CSS Generated Content for Paged Media Module, there is no standard for other types of notes, such as side notes, margin notes, end-of-column notes, or multiple note zones. On May 13, 2020, we proposed a draft specification introducing new CSS syntaxes for these note types, which was published in an issue within the CSS Print Community Group directory on the W3C GitHub [38].



Some note layouts detailed in the proposed specifications.

The goal is not to design for a specific project or tool but to think in terms of standards that enable interoperability between different tools. Today, I'd like to encourage graphic designers to actively participate in discussions around CSS. CSS remains the only language dedicated entirely to the visual presentation of documents and the graphic rendering of web pages. It's a unique, even 'radical [39]' invention within the field of informatics, created to address an unprecedented challenge in media history: the ability to technically specify which stylistic rules apply to particular devices and screen sizes. CSS is, therefore, a profoundly graphic language, ideally suited to the layout of all kinds of documents. Certainly, participating in its development requires some technical know-how and a willingness to follow highly codified standards. However, it's well worth the effort – contributing to CSS design ('writing the code for the code [40]') is a significant way to help building shared and sustainable tools.

However, it's crucial to distinguish between standards based on the common good and those imposed by a market-dominating industry. Understanding the stakes involved in their creation and participating in the process enables us to remain vigilant against proprietary appropriations and to examine the future implications of reuses that may lean towards extractivist and hegemonic practices.

Standards, such as those arising from open-source culture, can establish a common foundation without necessarily limiting creativity. Similar to the use of grids in graphic design, which provide structure while allowing for creative variation, this shared base encourages the development of communities of practice. And, members of these communities become connected through a common culture, characterized by aesthetic references, implicit rules, and codes [41]. Many of the most creative websites make explicit references to web technologies in how they structure information [42], their default styles [43], and their inherently declarative nature [44].

This dynamic relies on a deep understanding of the technologies and their inherently political and social dimensions. Engaging in collaborative discussions about standards allows for the integration of visions for their future reuse. This underscores the need for informal spaces like PrePostPrint, which promote political and non-commercial discourse on the construction of the commons. Developing shared tools requires establishing a common language – of practices and standards – understood by a broader community of stakeholders (such as W3C, web browser developers, web developers etc.). What is equally important is that the preservation of potential friction and controversy [45].

Conclusion: Building Communities Of Practices

In conclusion, the graphic design practices described in this article highlight the importance of collaborative organization, design, and programming at various levels, ranging from simple copy-pasting to the creation of standards. This diversity of practices contributes to the formation of communities of practice that, by sharing their work and interests collaboratively, not only develop their own vocabulary and methods but also transform the socio-economic dynamics of graphic work.

Particularly, the culture of free software, characterized by collaboration, sharing, and collective learning, invites you to rethink of the relationship between 'users' and 'designers.' It illustrates a path through which graphic designers can conceptualize the creation of their own tools. This positioning represents an unprecedented situation for the profession. The history of graphic design has often shown changes arising from outside the field, driven by the production of new technologies, machines, and software imposed by third-party companies. However, as we have demonstrated with Paged.js, the development of tools is partly the result of the work of the graphic designers who use them. This is made possible by the construction of a community of practice whose actions extend far beyond mere participation in the source code.

These practices are increasingly resonating with graphic designers who are concerned about the political, ecological, and economic implications of their profession. Using open-source tools is viewed as a way to resist large IT companies while gaining personal recognition and the respect of peers. Similarly, the concept of the 'hacker' is highly valued and would be a base for approaching technology in a critical and creative manner.

However, these discussions tend to overlook several crucial aspects of free software culture, particularly the demand for participation in collective activities and the actual production of shareable code. Through an imaginary conversation between a designer and a hacker, Anja Groten, a member of the Dutch collective Hackers & Designers, expresses her skepticism about the ways designers approach hacking. She notes that they often forget the inherent sociability of this kind of practices and merely adopt the jargon to rethink design methods.

> Hacking seems enticing, holding out appealing modes of self-determined making. (...) We need to move beyond fetishizing the hacker mode of production, and instead investigate the convoluted social construct of hacking – including its frictions and dilemmas. (...) Hacking is not a method you can first learn and then apply. Neither can you conceptualize hacking by means of design. Designers need to learn how to write, read, and fix code. They need to get literate before they can call themselves hackers. (...) Hacking might be an attitude towards making. But this attitude is tightly connected to the practice of writing software, debugging, running and maintaining systems. [46]

The critical stance of designers does not always lead to an actual participation in the community, at the risk of getting caught up in superficiality. In practice, few graphic designers who use free and open-source tools and software engage in their development or collective sharing (notably, this is also true on a larger scale: few users of free software contribute to its development). Thus, we observe a certain dissonance between strong political discourse and concrete practices, where free software and web technologies are used in a purely consumer-oriented manner.

Using software is not inherently an act of demonstration. Just because software is labeled as 'free' does not mean that the user is truly free. The essence of freedom lies in the distribution of the source code and its accompanying documentation. Access to both the software's source code and the educational resources that come with it is essential for the trans-individuation of free software communities. [47].

The aspects of sociality and community created by these practices are crucial in the material production of objects and artifacts. The concept of 'technological counter-gifting [48]' – of referral to the other – is essential in building free and open-source tools. The model promoted by free software

culture encourages the development of a sharing community where everyone contributes the fruits of their labor for the benefit of others. In this environment, individuals can be confident that their work will benefit the community while also gaining from the efforts of others. Therefore, it's important to think in terms of participation within these dynamics.

The demand for a non-passive use of technologies and commitment, comes with its challenges. Graphic designers who take this risk are constantly exposed to various conflicts and technical failures. However, Anja Groten emphasizes that frustration – whether due to lines of code that don't work – encounters with resistance, and dilemmas are integral to hacker culture. This culture is rooted in 'making,' and it is precisely this aspect that constitutes its political work [49].

There is an urgent need to develop an economic and social model that makes these practices sustainable within the field of graphic design. The question is how to transform this community into a larger and more diverse group of participants who are actively involved in the development, documentation, and promotion of tools, as well as in open pedagogical approaches. While engagement in free and/or open-source projects is valued for its political and social dimensions, it often places a significant workload on a limited number of contributors, highlighting the risks of burnout and the necessity for institutional support.

On a larger scale, it is essential to address the aspects of work in relation to free practice communities. In this context, the figure invoked is not that of the activist or volunteer who supports a cause for free, nor that of the 'hacker', 'tinkerer' or 'do-it-yourselfer' who repurposes technology for personal use; it is the worker who contributes to a collective and collaborative production.

The culture of free software has demonstrated that alternatives are possible through the articulation of technology and social change, facilitated by the creation of social communities. This culture is worthy of interest 'in what it constitutes a way of conducting collective action in alignment with its goals [50].' In this sense, these practices are political because they are driven by their own organizational policies, as well as by the discourses embodied in the artifacts they design and circulate. Shifting the idea of \'creating one's own tools\' toward the construction of practice communities and their resources would contribute to building a society in which individuals are active participants in a system they shape through collective practices. This proposal opens the door to a potential conversation on alternative relationships with both design and use of tools within the field of graphic design, through a collaborative and distributed approach to design.

Many thanks to Antoine Fauchié and Yann Trividic for their proofreading and the discussions that we had afterwards.

1. For a more comprehensive overview of the diversity of programming practices in graphic design, see Julie Blanc and Nolwenn Maudet's Code <-> Design graphique: Dix ans de relations, in Graphisme en France, no. 28, 2022, pp. 3-30. ↩
2. Loraine Furter, Trouble dans le genre – pédagogie alternative de l'édition hybride, Design research, Publications hybrides, 2018. Alexia de Visscher, Du design de la page à la pédagogie du flux : le cas belge, Back Office, n°3, 2019, pp. 122-135. ↩
3. Stéphanie Vilayphiou and Alexandre Leray, Écrire le design : vers une culture du code, Back Cover, n° 4 (2011), pp. 37-44 ; Kévin Donnot, Code = design, Graphisme en France, n°18, 2012, pp. 4-12. ↩
4. Antoine Gelgon, Un dialogue à réaliser : design et technique, in .txt 3, Éditions B42 et École supérieure d'art et de design Grenoble-Valence, 2018, pp. 38-58. ↩
5. Étienne Ozeray, Pour un design graphique libre, Master Thesis, École nationale supérieure des Arts Décoratifs, 2014. ↩
6. "The post-production is a culture of re-: remixing, reformatting, reworking, reinterpreting, reprogramming, replanning, restarting, re-displaying, recycling": Andrew Blauvelt, Tool (or, post-production for the graphic designer), Walker Art Center; Distributed by D.A.P., Minneapolis, New York, cop. 2011. ↩
7. Julie Blanc, Composer avec les technologies du web : Genèses instrumentales collectives pour le développement d'une communauté de pratique de designers graphiques, PhD thesis in ergonomics, Université Paris 8, Vincennes - Saint Denis, 2023. ↩

8. PrePostPrint is an informal group focusing on experimental publications designed with free tools. This article was written shortly after PrePostPrint's participation in the Libre Graphic Meetings, which took place from May 9 to 12, 2024, in Rennes, France. It has greatly benefited from the discussions that occurred during the event. ↩
9. The complexity of web development has increased considerably over the years. Today, it's common to use code snippets, JavaScript plugins, and content management systems (CMS) to create more interactive and dynamic websites. However, it's still possible to create a simple, functional website using only HTML, CSS, and a little JavaScript. ↩
10. Alexia de Visscher, Du design de la page à la pédagogie du flux : le cas belge, op.cit. ↩
11. A fork is a copy of a source code project, often used in free and/or open-source development. It allows developers to work on their own version of the code without affecting the original project. ↩
12. Eric Schrijver, No-one Starts From Scratch: Type Design and the Logic of the Fork, i.liketightpants.net, 9 october 2013 ↩
13. Françoise Decortis, Anne Bationo-Tillon, and Lucie Cuvelier, Penser et concevoir pour le développement du sujet tout au long de la vie : de l'enfant dans sa vie quotidienne à l'adulte en situation de travail, Activités 13, n° 2, october 2016. ↩
14. Richard Stallman, Why Open-Source Misses the Point of Free Software, gnu.org ↩
15. Glyn Moody, Rebel Code: Linux and the Open-Source Revolution, Perseus Books Group, 2002. ↩
16. This idea was extensively developed in my thesis. It's important to note that in graphic design, 'composition' specifically refers to the process of assembling (lead) characters to form lines of text. Therefore, adopting this definition seems particularly appropriate. ↩
17. Julie Blanc, Composer avec les technologies du web, op. cit., p. 208 ↩
18. Idem ↩
19. Graham Button et Wes Sharrock, The Mundane Work of Writing and Reading Computer Programs, in Situated Order: Studies in the Social Organization of Talk and Embodied Activities, Paul ten Have et George Psathas, International Institute for Ethnomethodology and Conversation Analysis [u.a.], 1995, pp. 231-86. ↩
20. Didier Demazière, François Horn, et Marc Zune, Des relations de travail sans règles ? L'énigme de la production des logiciels libres, Sociétés contemporaines n 66, n° 2, september 2007, pp. 101-25. ↩
21. See https://esadpyrenees.github.io/PageTypeToPrint/ ↩
22. See web version of this fanzine on this link: https://avoidsoftware.sarahgarcin.com/index.html ↩
23. Julie Blanc et Benjamin G., Médialab (1/3) : Une décroissance heureuse, julie-blanc.fr, 27 March 2020, ↩
24. Here are his notes on Paged.js use: Making-of d'une collection libérée : Addictions sur ordonnance, published 10 February 2019, and, Dans les recoins de la double page (Paged.js à la maison, saison 2), published 1st March 2021 ↩
25. Manetta Berens, html2print as a practice of boilerplates, osp blog, 23 June 2023 ↩
26. Find Cursors without sou(r/c)is, on this address: https://web.archive.org/web/20240521122022/https://garden.delyo.be/journalbord/log.html ↩
27. Kévin Donnot, Code = design, Graphisme en France, n°18, 2012, pp. 4-12. ↩
28. See https://www.luuse.io/projects/edri/ ↩
29. See https://gitlab.com/bonjour-monde/tools/dataface ↩
30. "In computing, a script means a program (or part of a program) that is a responsible of executing a preset action when a user performs an action or a web page is being displayed. It is a series of commands (...) that automate certain successive tasks in a given order." Translated from French to English, definition of Webmastering Dictionary, available thanks to Journal du net ↩
31. Paged.js is a 'polyfill' – a palliative software solution that implements functionalities not yet natively available in web browsers. ↩
32. This inventory and documentation work began on a dedicated directory called "Plug-ins" on the Coko Foundation's Gitlab site; it can be viewed at the following address: https://gitlab.coko.foundation/pagedjs/pagedjs-plugins. ↩
33. We're talking about de facto standards here. They operate differently from de jure standards, or standards according to law, endorsed by a formal standards organization (like ISO, International Organization for Standardization). The organization ratifies each standard through its official procedures and gives the standard its stamp of approval. (In french, de facto standards are called "standards" and de jure standards are called "normes") ↩
34. In that respect, Jarrett Fuller described their use as a radical action: "To build a website with just these tools is a small protest against platform capitalism: a way to assert sustainability, independence, longevity. » ↩
35. Web page available at the following address: http://info.cern.ch/hypertext/WWW/TheProject.html ↩
36. These include three major US multi-national web browser developers - Apple (Safari), Google (Chrome) and Microsoft (Edge) -- and two organizations developing free or open-source software - Mozilla (Firefox) and Igalia - and, ironically, the multinational Adobe Inc. ↩

37. To see how CSSWG works: Fantasai, "about:csswg ", https://fantasai.inkedblade.net/, November 2011, viewed on 12/02/2022. ↩

38. On Thursday February 13, 2020, as part of the XML Prague conference, at the University of Economics in Prague (Czech Republic), the CSS Print Community Group is formed with the aim of working on CSS specifications for print, presenting the cases of use and advocating for better implementations in browsers. This W3C sub-group would reinforce the work of the CSS Working Group by focusing on CSS for print and paged media. My presence at this conference **encouraged** me to propose the specifications mentioned in the text. ↩

39. Miriam Eric Suzanne, CSS Is Rad, Miriam Eric Suzanne, https://www.miriamsuzanne.com/speaking/css-rad/, 2020. ↩

40. Julien Rossi, Écrire le code du code, RESET. Recherches en sciences sociales sur Internet, n° 11, march 2022. ↩

41. Many thanks to Raphaël Bastide for giving me the idea. ↩

42. Numerous manifestos highlight the importance of HTML structuring in web practices: "Web design as architecture", "HTML energy". ↩

43. Étienne Cliquet, "Esthétique par défaut. La beauté parfum vanille", Téléférique, collective & independent download server online, August 2002. ↩

44. See also the Declarations initiative, initiated and coordinated by Doriane Timmermans. "Declarations is ongoing artistic research into the poetic materiality of the CSS web-standard. Declarations is a love letter to the crafts of designing with language. (...) We research how, similarly to the choice of words we decide to use to tell a story, designing with declarations speak about our intentions, and encodes narrations into the things we make. By looking at the web through declarativeness, a curious materiality starts to glimmer. The web is both languages and materials. Web-designers become both author and architect. And websites become a work of articulation." ↩

45. See Julien Bidoret's thoughts on this subject: "Flossflop", published on April 15, 2024 on his online notebook ↩

46. Anja Groten, Hacking & Designing: Paradoxes of Collaborative Practice, in The Critical Makers Reader: (Un)Learning Technology, éd. par Læs Bogers et Letizia Chiappini, INC Reader 1, Institute of Network Cultures, 2019, pp. 238-239. ↩

47. Antoine Gelgon, Un dialogue à réaliser : design et technique, in.txt 3, Éditions B42; École supérieure d'art et de design Grenoble-Valence, 2018, 38-58. ↩

48. Nicolas Oliveri, Logiciel libre et open source : une culture du don technologique, Quaderni. Communication, technologies, pouvoir, n° 76, september 2011, pp. 111-119. ↩

49. Groten, Hacking & Designing, op. cit. ↩

50. Sébastien Broca, Utopie du logiciel libre. Du bricolage informatique à la réinvention sociale, Le passager clandestin, 2013, p. 266. ↩

NOTIONS

Community
En attente traduction EN

Empowerment
"The Radical Empowerment Discourse: Originating in the 1970s, this discourse was championed by feminists, black, and popular education activists, and focused on collective liberation in self-emancipatory and explicitly political terms. It emphasised egalitarian, participatory, and local aspects, and at its core, was framed as the self-capacity of grassroots groups to take power over their own lives and dismantle oppressive systems as a necessary condition for social change.
The Reformist and Liberal Empowerment Discourse: Then, in the 1980s, a first elite capture of the meaning occurred, and "from its radical roots, we begin to see its incorporation and institutionalisation within systems of state governance" (McLaughlin, 2016, p.5). The empowerment discourse evolved towards a more reformist and light-liberal framework of incremental ways to reform without posing a threat to existing systems. The previous focus on conscientização (Freire, 2017) embraced more therapeutic and isolated notions of individual self-awareness, and the leading figures for change became professionals and experts. Empowerment adopted a state-based, rather than community-based, approach to hybrid governance and light self-management, especially in the realm of community development, as an alternative to older, top-down types of institutionalized government considered paternalistic and obsolescent.
The Neoliberal Evolution of Empowerment: Finally, perhaps the most significant leap in the evolution of empowerment discourse was the full neoliberal shift from the 1990s onwards, when financial interests took over the conversation. Since then, empowerment has been ultimately stripped of its political and militant meanings and reframed as the entrepreneurial capacity of individuals to insert themselves into the market society. It established its definitive integration into global discourses of managerial and financial expertise through programs and policies led by international institutions such as the United Nations and the World Bank, and, through deregulation and privatisation, saw central governments abdicating allocations of power, responsibility, and resources to the free market." (Volpi et al., 2024)

Engagement