# COLLABORER À GRANDE ÉCHELLE DANS DES COMMUNAUTÉS DE PRATIQUE EN DESIGN GRAPHIQUE

AUTEURICE
Julie Blanc

LICENCE
Creative Commons Attribution NonCommercial 4.0 Inter

DATE DE PUBLICATION
02 juin 2024

Dans le domaine du design graphique, la collaboration est souvent perçue comme une participation directe, circonscrite dans un temps et un espace donné (par exemple, par la contribution à un workshop ou à un projet spécifique impliquant la collaboration du « public » pour produire une forme). Pourtant il existe de nombreuses formes de collaboration, notamment à grande échelle au sein de communautés de pratiques partageant une même culture. Cet article se penche ainsi sur l'intégration de la culture du libre et de ses aspects collaboratifs dans certaines pratiques de designers graphiques, en particulier dans le design pour et avec le web. Il s'inscrit dans le contexte d'une adoption progressive du code par une proportion croissante de designers graphiques [1], principalement dans les pratiques de publication [2].

La culture du libre est basée sur l'utilisation de programmes et de logiciels libres, dont les caractéristiques principales sont les possibilités d'accéder, de modifier et de redistribuer leur code source. La Free Software Foundation considère ainsi un logiciel comme libre s'il répond à quatre libertés accordées aux utilisateur·rice·s : celles d'utiliser (liberté 0), de copier (liberté 1), d'étudier (liberté 2) et modifier le programme (liberté 3). Outre ces considérations associées aux logiciels, la culture du libre est également un mouvement social et philosophique qui prône un rapport créatif et éthique aux technologies et aux manières de travailler. Les militant·e·s de la culture du libre s'engagent notamment pour que les utilisateur·ice·s puissent maîtriser leur informatique et pour la production de technologies socialement utiles.

Ces idées sont reprises et défendues par de nombreux·ses designers graphiques, qui prônent l'utilisation du code dans leurs pratiques [3]. Ils·elles souhaitent ainsi construire une culture technique du design graphique, fondée sur la transmission de savoirs techniques permettant d'intégrer ceux-ci dans un processus de valorisation [4]. L'insertion de la programmation dans le design graphique est aussi mise en valeur comme une pratique expérimentale et créative des technologies ; le processus de

création est orienté vers le faire artistique, ses dimensions formelles et esthétiques et la découverte d'outils inhabituels [5]. Plus particulièrement, le logiciel libre est par ailleurs abordé comme un exemple réussi d' une conception du travail radicalement différente dans un contexte de « post-production », terme utilisé par Andrew Blauvelt pour désigner la disparition progressive de la distinction traditionnelle entre production et consommation, ou création et copie[6].

Bien que les aspects collaboratifs de la culture du libre soient aussi souvent mis en avant, ils sont assez peu commentés depuis le point de vue spécifique du design graphique, où les participant·e·s à cette culture ne sont pas avant tout des programmeur·euse·s. Or le principe au fondement des logiciels libres – à savoir l'obligation de la mise à disposition du code source avec autorisation de le modifier et le distribuer librement – se traduit par la coopération de développeur·euse·s qui participent ainsi à la conception d'une œuvre collective. Le travail collaboratif y incarne des valeurs d'entraide et de mutualisation des efforts individuels.

Ce texte vise donc à interroger comment la culture du logiciel libre, marquée par la collaboration, le partage et l'apprentissage collectif, s'incarne concrètement dans les pratiques de programmation au sein du champ du design web et du design graphique. Pour cela, je propose d'aborder dans ce texte plusieurs notions thématiques développées plus longuement dans mon travail de thèse [7]. Je m'exprime depuis un point de vue situé, à travers mon implication à la fois dans le collectif informel PrePostPrint [8] et dans le développement de l'outil Paged.js. De ce fait, il est majoritairement question dans ce texte de pratiques en lien avec les technologies du web, utilisées indifféremment pour de la publication web ou de la publication imprimée.

Au-delà d'un travail critique, cette proposition est l'occasion de discuter avec l'appui d'exemples concrets des pratiques de la culture du libre dans le design graphique. Ces exemples permettent par la même occasion d'éclairer l'utilisation du code non comme une pratique du bricolage et de « hacking » à laquelle elle est souvent réduite mais comme une pratique culturelle et sociale centrée sur la collaboration.

## Ouvrir Et Modifier Le Code : Balisage Et Mode Texte

Une grande majorité de l'utilisation du code par les designers graphiques se concentre sur les technologies du web, dont les deux principaux langages sont HTML et CSS. En effet, la possibilité d'intégrer et de publier divers médias dans un espace unifié (vidéos, processus génératifs, animations, etc.) fait du Web un moyen de publication incourtournable et donc d'un grand intérêt pour les designers graphiques.

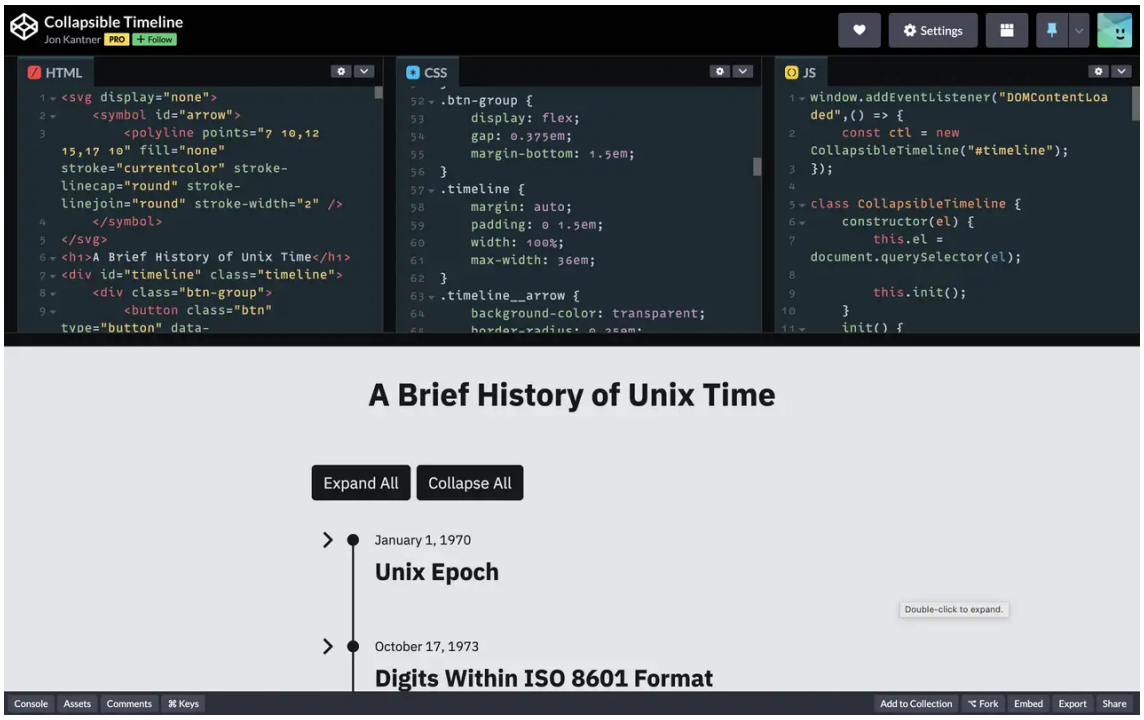
HTML, acronyme de HyperText Markup Language (langage de balisage hypertexte), est utilisé pour la structure d'une page web. Par exemple, la balise `<body>` indique où le contenu commence, tandis que `<h1>` indique un titre de niveau 1. CSS, abréviation de Cascading Style Sheets (feuilles de style en cascade), est utilisé quant à lui pour décrire la présentation d'un document, son style. C'est grâce à CSS que l'on peut définir la police de caractère d'un titre ou la couleur d'un encart. Ainsi, un site web est constitué d'un ensemble de fichiers HTML reliés entre eux par des hyperliens et mis en forme avec CSS [9].

HTML et CSS sont des langages standards et descriptifs, relativement simples à apprendre et à utiliser. Ils ne nécessitent pas de maîtrise approfondie de l'algorithmie ou de processus de compilation complexes. Il suffit d'un navigateur web pour les interpréter, ce qui rend leur usage accessibles à tous. Pour Alexendra De Visscher, l'utilisation des technologies du web pour la conception de publications hybrides, permet aux designers de « s'affranchir des interfaces WYSIWYG par le biais de syntaxes explicites, formelles, qui révèlent la structure sémantique du contenu » [10].

La spécificité de la pratique du code avec le web est ainsi sa relation directe au mode texte, sans besoin d'un logiciel particulier, ce qui facilite la reprise et le partage. Les designers peuvent facilement copier et coller des extraits de code écrits par d'autres, quel que soit leur niveau de compétence en

programmation. Cette facilité de reprise a permis de constuire toute une écologie du partage (ecology of sharing), basée sur la facilité de la mise à disposition de ressources. Cette approche, bien que moins ambitieuse que le « fork » [11], est la première forme de collaboration entre designers-développeur·euse·s. (À partir d'ici, nous utiliserons indistinctement les deux termes.)

Publier du code sur une plateforme dédiée, comme CodePen, permet de montrer clairement sa volonté de le voir repris et modifié par d'autres. Les designers graphiques peuvent ainsi puiser dans un grand nombre de ressources en ligne alimentées par une communauté de développeur·euse·s web expert·e·s ou amateur·rice·s.




Capture d'écran d'un projet sur CodePen, un site communautaire de partage de code HTML, CSS et JavaScript fonctionnant comme un éditeur de code source interactif.

Cette culture de l'appropriation et de la réutilisation soulève des débats sur le droit d'auteur et l'économie du partage. La notion d'autorité singulière associée communément au design graphique est en effet remise en question par le partage d'outils et l'utilisation du code. Bien que l'idée romantique d'un génie créateur ait été depuis longtemps réfutée [12], elle continue de hanter le domaine du design graphique. Cependant, le recyclage de morceaux de code illustre concrètement l'idée que les formes sont issues de diverses sources agrégées et réagencées. Comme l'a rappelé le psychologue Lev Vygotski et d'autres chercheur·euse·s après lui [13], toute création, même individuelle, inclut toujours un coefficient social et porte en elle la collaboration anonyme des autres. L'utilisation des technologies du web met en lumière cette dimension sociale de la création.

Dans un autre registre, l'utilisation du code permet de profiter concrètement de l'écosystème d'outils et de méthodes collaboratives développés depuis des décennies pour faciliter le travail de programmation. L'un des piliers de cette collaboration est l'utilisation de systèmes de gestion de versions décentralisés, dont Git est plus célèbre. Il permet aux développeur·euse·s de gérer les modifications du code source (fusionner les modifications, gérer les conflits) et facilite ainsi le travail collaboratif d'écriture du code. Les plateformes de gestion de projets comme GitHub, GitLab, Bitbucket ou encore Codeberg permettent de coordonner les efforts de développement, de suivre les problèmes et de faciliter la contribution des membres de la communauté. Tout un vocabulaire spécifique à la

programmation entre ainsi peu à peu dans le design graphique par l'usage de ces outils : fork, commit, version, issue, etc.

Ces outils et pratiques de collaboration sont par ailleurs au cœur des controverses de la culture du libre. En effet, depuis les années 1990, deux conceptions du logiciel libre s'affrontent entre free software et open source [14]. Portée notamment par Linus Torvalds, fondateur de Linux, l'open source est une méthode de développement reposant sur l'ouverture du code, sans que les valeurs de la culture du logiciel libre y soient nécessairement associées [15]. Ce mouvement s'appuie sur un discours visant à défendre une meilleure adéquation des principes d'ouverture avec les enjeux économiques et commerciaux du capitalisme. L'idée étant de dire que l'open source peut être une alternative viable, en termes commerciaux et techniques, au logiciel propriétaire. L'open source se focalise ainsi plutôt sur des considérations techniques de développement logiciel et l'organisation du travail communautaire qui rendrait les logiciels plus efficaces et fiables.

Ainsi, pour le mouvement de l'open source, l'accès au code source est une question pratique et non éthique. Le mouvement du logiciel libre, est lui une question sociale, profondément attaché à la notion de liberté. Toutefois, ces deux mouvements ont en commun d'avoir permit l'émergence de nouvelles formes de coordination, irréductibles aux organisations formelles, où la collaboration est encouragée et où les connaissances sont largement accessibles.

Les méthodes de travail collaboratives et horizontales propres à la culture du libre transforment la pratique des designers graphiques qui s'en réclament. Ces derniers passent ainsi d'une approche centrée sur une autorité singulière à une contribution collective et ouverte.

## Organiser Son Code : Pratiques D'écritures Codifiées

Écrire du code n'est pas seulement une compétence technique, mais aussi une pratique socialisée, qui prend en compte la communauté dans laquelle les designers graphiques évoluent. La possibilité de partager du code a une influence sur la façon dont il est écrit. Ainsi, les designers graphiques ne se contentent pas de mettre en forme des objets graphiques en codant, mais travaillent également la mise en forme du code lui-même pour qu'il soit intelligible et lisible – condition indispensable pour permettre le partage de celui-ci.

Dans le domaine de la programmation, la nécessité d'écrire un code source clair, rigoureux et bien structuré est une règle implicite. Cette clarté est essentielle non seulement pour que le code soit compréhensible par les autres, mais aussi pour que les développeur·euse·s puissent eux·elles-mêmes naviguer efficacement à travers leur propre travail. Différentes pratiques d'écriture codifiées sont ainsi mises en œuvre pour rendre le code plus lisible et plus accessible.

La sémantique du code est particulièrement travaillée. Par exemple, la règle de pertinence est primordiale lorsqu'il s'agit de nommer des fichiers, des classes HTML, des fonctions JavaScript ou certaines propriétés personnalisées. Ainsi, une variable CSS nommée « baseline » dans un code fournit un indice sémantique quant à son utilisation pour aligner les éléments en fonction d'une ligne de base. Inversement, un système de dénomination trop abstrait ou trop personnel peut entraver la possibilité d'une lecture collective du code et donc sa réappropriation.

La structure visuelle et graphique du code est également importante pour refléter son organisation. L'utilisation de retours à la ligne, d'indentations, d'imbrications et d'espacements permet de créer des métaphores visuelles ou des représentations littéraires des relations entre les différents éléments du code, afin de "laisser en vue" sa structure. Il est également fréquent de décomposer le code en unités appréhendables en le divisant dans des fichiers dédiés à une fonctionnalité précise.

En ce sens, écrire du code est un acte de composition [16]. Il est intéressant de rapprocher cette activité de mise en lisibilité du code, en particulier par sa structuration, à l'un des buts affichés du design graphique de mise en lisibilité des savoirs, notamment par l'architecture des informations et la mise en place de systèmes graphiques. Ainsi, structurer, organiser, agencer des éléments au sein d'un

format pour les rendre visibles et lisibles sont au cœur des pratiques des designers graphiques de manière générale. Cette similitude dans les pratiques peut expliquer l'appétence de certains designers graphiques pour le code.

Dans les pratiques de programmation en design graphique, l'acte de composition est donc double : dans les formes produites, et dans le code produit. En travaillant l'écriture du code, les designers graphiques s'adressent à eux·elles-mêmes et à leurs pairs. Il ne s'agit plus seulement d'écrire pour produire mais aussi d'écrire pour les autres et avec les autres en produisant un « beau code », « lisible et compréhensible » qui puisse être partagé avec la communauté.

Cette idée porte en elle une réciprocité certaine. C'est en partant du principe que leurs pairs partagent la même pratique de mise en lisibilité du code que les designers graphiques sont capable de s'approprier un code écrit par autrui. En s'adressant mutuellement leur code à travers des « bonnes pratiques » partagées – ce que nous avons appelé des pratiques d'écritures codifiées – les développeur·euse·s s'assurent ainsi une réciprocité dans la constitution de leurs ressources. Ainsi, dans un entretien réalisé dans le cadre de ma thèse, le designer Benjamin G. explique : « Il faut que les autres soient en mesure de comprendre mon code. (…) Regarde, j'ai pris un bout de code chez quelqu'un et je suis très content d'avoir pu comprendre. (…) Ça m'oblige à prendre des bonnes pratiques moi aussi, que mon code soit en accès libre ou non.[17] »

```
7   #page-article {
8       color: var(--color-article);
9
10      header {
11          padding-top: calc(var(--baseline)*0.75);
12          padding-bottom: calc(var(--baseline)*0.5);
13          border-bottom: 1px solid var(--color);
14          font-family: var(--font-title);
15          display: flex;
16          justify-content: space-between;
17
18          button {
19              color: var(--color);
20          }
21      }
```

Extrait de code (S)CSS où on voit l'utilisation de variable (`var(--color-article)`) et une structuration du code par indentation

Ces « bonnes pratiques » peuvent être vues comme des règles partagées par la communauté de manière formelle ou informelle et acquises au fil de l'expérience, par le biais de diverses documentations ou en se confrontant directement au travail collaboratif. C'est aussi ce qu'explique la designer graphique Amélie Dumont : « À partir du moment où j'ai commencé à me confronter à partager mon code avec d'autres personnes, ça m'a aidé à progresser et à comprendre… [Par exemple de comprendre que] (…) la structure de base de HTML et de CSS, ce n'est pas du tout inutile. Avoir des fichiers les plus clairs et le plus simple possibles, ça permet de mieux travailler avec d'autres personnes. [18] »

Cette activité de mise en lisibilité du code est profondément liée à la culture du logiciel libre et open source, où le partage du code et sa compréhension par la communauté sont des valeurs fondamentales. En rendant leur code lisible grâce à un ensemble de bonnes pratiques partagées, ils·elles assurent de ce fait une compréhension mutuelle au sein de la communauté.

Dans un article où ils étudient plus en profondeur l'activité d'écriture des programmes informatiques, Graham Button et Wes Sharrock mettent en évidence que l'un des critères d'identification de l'écriture professionnelle d'un programme informatique se trouve dans la manière dont son écriture est orientée vers le fait que les programmeur·euse·s travaillent dans le cadre d'une communauté de praticien·ne·s et sont, en tant que professionnel·le·s, dans l'obligation de prendre en compte les intérêts de ces autres personnes. À cet égard, « apprendre à écrire un programme informatique, c'est apprendre à l'écrire de manière à ce qu'il soit intelligible pour une communauté de praticiens. » [19] Ainsi se développe la collaboration.

## Documenter : Des Formes Multiples De Partage

Alors que la participation à la création de codes sources semble une évidence, le développement collaboratif d'un outil libre exige aussi « des échanges sur des questions réputées peu cruciales mais pourtant chronophages : répondre à des demandes d'aide, donner des conseils, donner des avis sur une contribution, évaluer une traduction, proposer une documentation, etc. [20]». Ce travail, souvent invisibilisé, voir dévalorisé, est pourtant crucial pour l'appropriation d'un outil ou un logiciel. Je prendrai ici l'exemple de la documentation et ses différentes formes, autant dans la conception de logiciels que dans le partage de ressources.

La **documentation technique et les manuels d'utilisation** sont les formes les plus répandues. Elles contiennent des informations détaillées sur le fonctionnement interne du programme, son utilisation, sa configuration et son développement. Elle comprend aussi généralement des descriptions des fonctionnalités du logiciel, des exemples d'utilisation, des guides d'installation, des références de code, des spécifications techniques, et parfois même des tutoriels pour aider les utilisateur·ice·s et les développeur·euse·s à comprendre et à utiliser le logiciel.

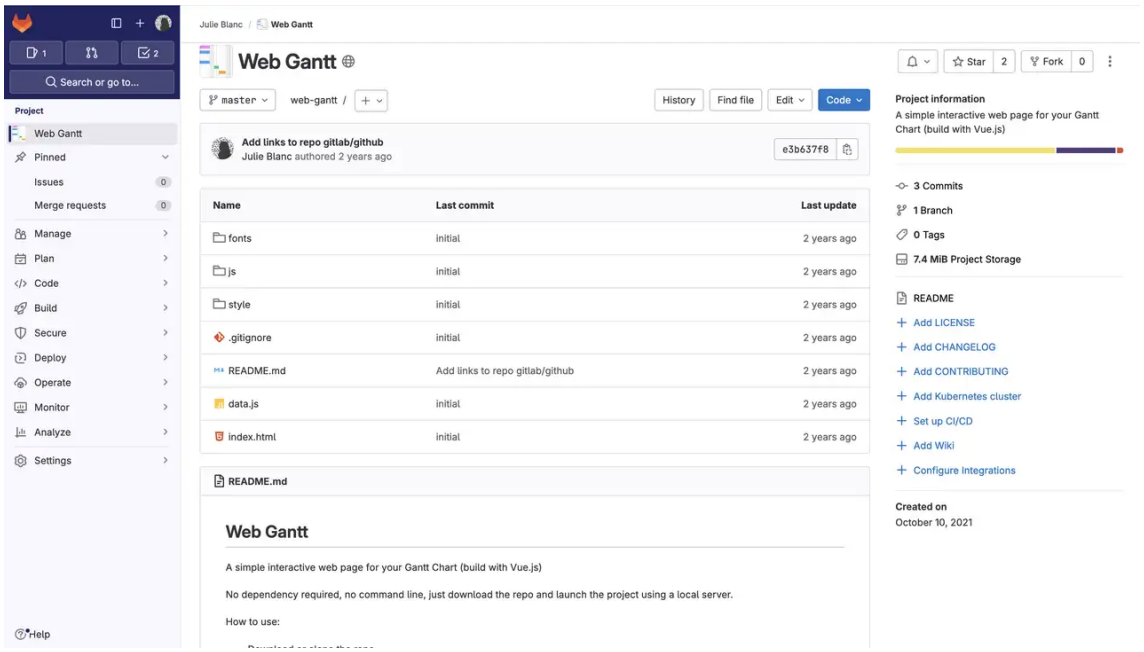
Si les logiciels à interfaces graphiques ont habitué les designers graphiques à se passer de ces types de documentations, elles sont des ressources essentielles en programmation. Par ailleurs, le célèbre acronyme RTFM (« Read The Fucking Manual ») est une expression courante utilisée dans le domaine pour suggérer de lire la documentation comme première étape pour résoudre tout problème technique.

Pour PageTypeToPrint, un outil de mise en page et de publication web + print conçu originellement comme un gabarit de mise en forme des mémoires des étudiant·e·s de l'École supérieure d'art et de design des Pyrénées, Julien Bidoret a ainsi fournit une documentation riche et exhaustive [21]. Cette dernière a été rédigée de manière à être lisible pour les étudiant·e·s qui auraient peu de compétences en code. Elle a ainsi facilité l'adoption de cet outil bien au-delà de l'école pour lequel il a été développé.

Les **Wiki ou bases de connaissances** sont une autre forme de documentation intéressante. Ces documents sont souvent collaboratifs et permettent aux utilisateur·ice·s de contribuer à la création et à la mise à jour d'informations sur un sujet ou un outil spécifique. Ils peuvent contenir des articles, des tutoriels et des guides. Wikipédia est la forme la plus célèbre de ce genre d'approche, mais elle peut aussi être déployée à une échelle plus réduite et située. L'école de recherche graphique de Bruxelles (erg) possède ainsi son propre wiki où sont répertoriées les informations de l'école et diverses ressources des ateliers et cours.

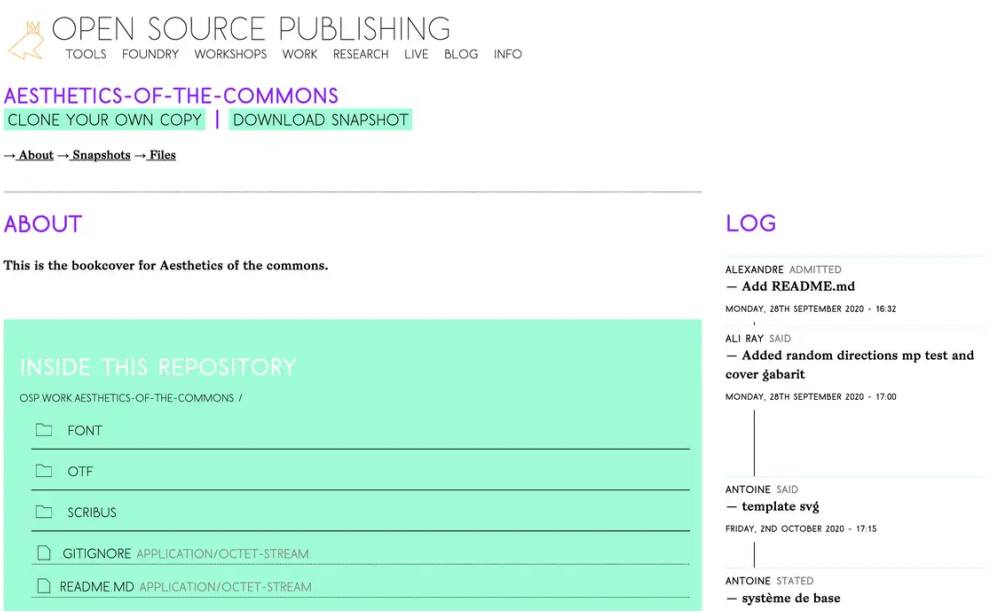
Plus spécifiques au monde de la programmation, les codes source des programmes libres et open-source sont souvent distribués via des services comme **Github ou Gitlab**. Ces services sont basés sur le logiciel de versionnement Git qui permet d'accéder à différentes versions d'un code source. Par convention, un dossier de code source lié à un projet (un « répertoire ») est toujours accompagné d'un

fichier de texte markdown nommé « README.md » qui sert de guide introductif à ce projet et y résume les fonctionnalités disponibles. Ce fichier est ainsi une documentation succinte du projet et sert souvent de porte d'entrée à son code source, raison pour laquelle il y est souvent accordé une grande importance.



Capture d'écran d'un répertoire de code sur GitLab

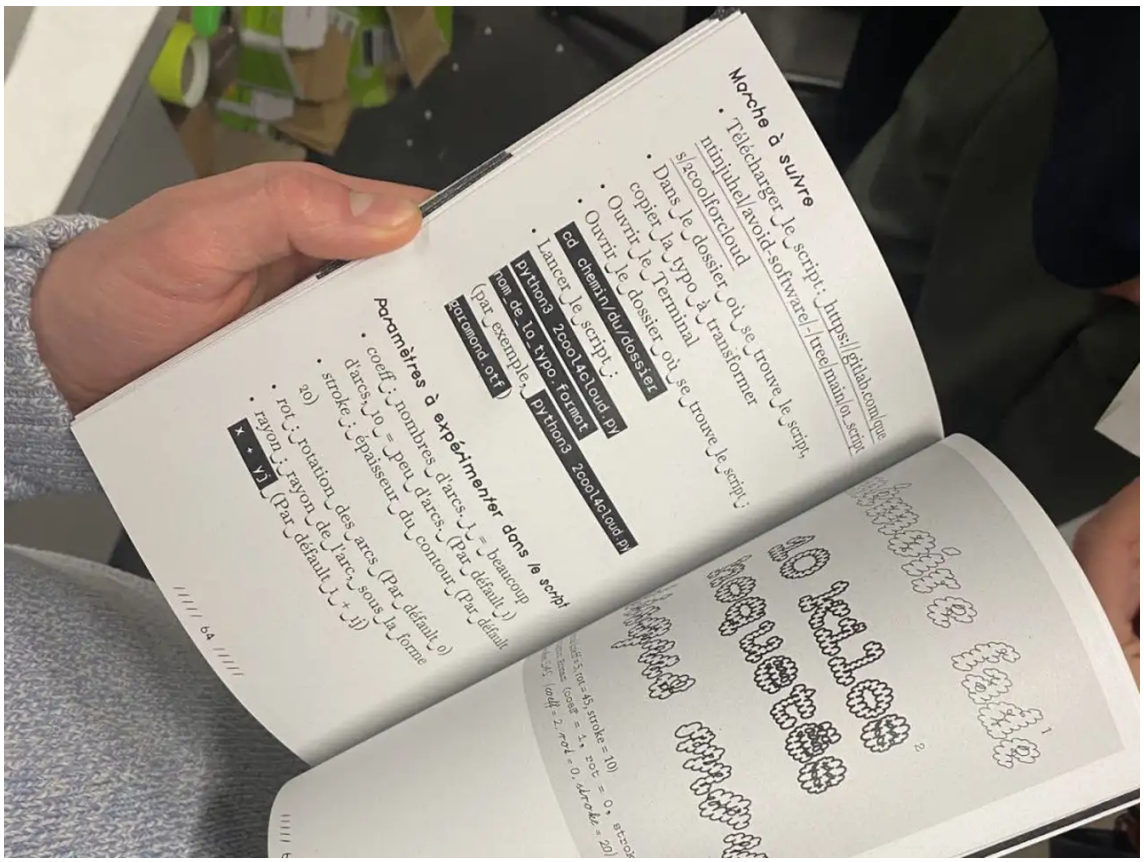
La plupart des designers graphiques et collectifs utilisant la programmation possèdent des comptes sur ces services (voir hébergent leur propre instance) qui servent alors de documentation de leur pratique. Open Source Publishing a par ailleurs adopté cette approche directement sur son site web où chaque projet est présenté avec une section « about » (présentation succincte du projet) , une section « inside the repository » (code source du projet) et une section « log » (historique des modifications commentées) rendant visibles les différentes évolutions du projet.



Capture d'écran de la vue d'un projet sur le site d'Open Source Publishing (osp.kitchen)

En dehors des pratiques de ces designers graphiques qui programment, la documentation s'inscrit dans des modalités d'écriture auxquelles les designers graphiques sont peu habitué·e·s, et peut parfois sembler intimidante. Elle est pourtant la condition pour espérer construire des pratiques collectives partageant des ressources communes.

Une approche intéressante serait alors de s'emparer de formes plus accessibles et familières aux designers graphiques qui ne codent pas. En ce sens, Sarah Garcin, Quentin Juhel et Emma Sizun ont proposé Avoid Software. Everyone has it [22], un **fanzine** web et imprimé, conçu et fabriqué spécialement pour l'évènement Open-Open qui s'est déroulé le 11-12 mai 2023 à l'ESAC Cambrai. La publication, dont le code source est accessible librement, propose « des méthodes, scripts, hacks libres et open-source utiles à tout artiste, designer ou étudiant·e en école d'art et de design ».



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

Les **récits d'expériences**, tels que les billets de blog, les carnets de projets et les articles, sont également une forme de documentation qui participe à la construction collective d'une culture de la programmation dans le design graphique. Benjamin G. et moi-même avons ainsi publié une série de trois articles de blog sur le projet de refonte du site web du médialab de Sciences Po. [23] Nous y détaillons les principes de sobriété qui ont été choisis pour la conception du site et comment nous les avons mis en application avec des exemples de code CSS. Dans la même idée, Nicolas Taffin, designer graphique et co-fondateur des éditions C&F, partage régulièrement sur son blog des « making-of » de la conception de ses livres [24].

Ce genre de publications peut aussi être consacré à un partage plus réflexif sur ces pratiques. Manetta Berens a ainsi profité d'une résidence chez Open Source Publishing pour revenir sur « dix ans de pratique web-to-print » au sein du collectif à travers des billets de blogs sous forme d'entretiens et compilant une liste de dépôts annotés et contextualisés [25]. Notons enfin, qu'il n'y a pas besoin d'être « expert confirmé » pour partager son expérience. Assez récemment, le designer graphique Delyo Dobrev a partagé un journal (logbook) consacré à sa tentative d'adapter la revue Curseurs de Tactic à une mise en page programmée avec Paged.js. [26]

Dans le design graphique tourné vers la programmation, les pratiques de documentation sont ainsi une part essentielle de l'activité. La documentation participe alors à l'apprentissage et à l'émergence de communautés de pratiques.

## Ajouter Des Fonctionnalités : La Conception Par Plugins

L'intégration de la programmation dans les pratiques du design graphique porte un imaginaire de création de ses propres outils. Pour le designer graphique Kévin Donnot, « un graphiste-hacker pourrait créer ses programmes à sa main, pour répondre à ses exigences propres, lesquelles

participent de son statut d'auteur [27] ». En ce sens, la majorité des outils créés par les designers graphiques sont conçus pour des pratiques personnelles ou pour des projets spécifiques.

Le collectif Luuse a par exemple développé H.I.R.P, Honey, I Resized the Poster, un outil web permettant de générer différents formats d'impression à partir d'un même contenu. Cet outil à été spécialement utilisé pour la conception de posters pour l'European Digital Rights (EDRi) [28]. Le collectif Bonjour Monde a quant à lui imaginé dataFace [29], un outil expérimental de manipulation typographique intégrant des propriétés de police variables et utilisé lors de plusieurs ateliers pour explorer de nouvelles formes typographiques. Ces projets illustrent l'idée d'une création sur mesure, où l'outil lui-même devient la production des designers graphiques. Cependant, en raison de leur nature de conception spécifique, il est assez rare que ces programmes soient repris en dehors du contexte pour lesquels ils ont été développés.

Pour envisager une conception collaborative d'outils de plus grande envergure et une autonomie de production dans le design graphique, il est nécessaire de se détacher de cette vision d'autorat (qui est intéressante par ailleurs, mais n'est pas le propos de ce texte). Nous nous intéresserons ici à la conception d'outils génériques pouvant être adaptés à plusieurs situations et utilisables par une plus grande communauté de praticien·ne·s. Plus particulièrement, nous nous intéressons aux possibilités d'ajouts de fonctionnalités dans des logiciels libres déjà existants.

Prenons pour exemple Paged.js, un outil libre et open source [30] permettant d'utiliser les technologies du web pour l'impression en implémentant dans les navigateurs web des spécifications CSS du W3C qui n'y sont pas disponibles. L'outil se présente sous la forme d'un fichier de code écrit en langage JavaScript [31]. Le fichier est à ajouter à n'importe quelle page HTML associée à ses feuilles de styles CSS. Lors du rendu de la page web dans un navigateur web, le contenu sera alors affiché sous la forme d'une prévisualisation paginée et les CSS spécifiques à l'impression seront appliqués. Une fois la mise en forme finie, le PDF peut être généré depuis les options d'impressions du navigateur web.

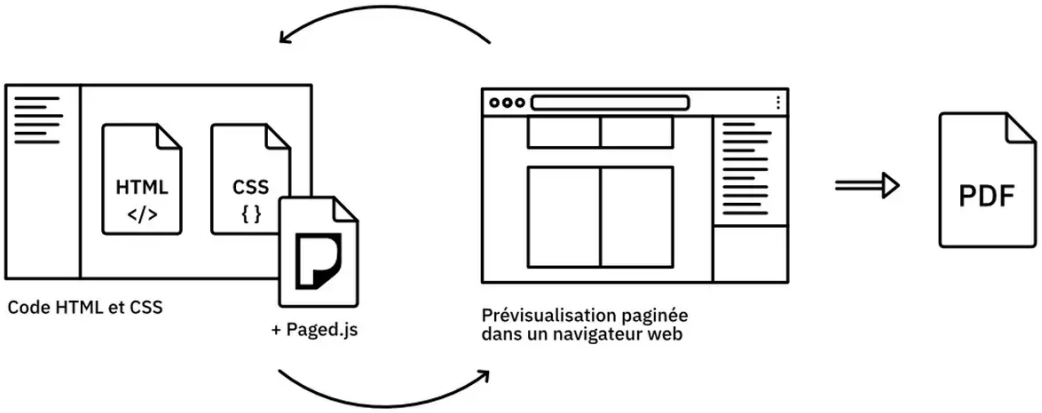



Schéma du fonctionnement de Paged.js

En cherchant à se conformer strictement aux standards CSS [32], certaines fonctionnalités de mise en page n'ont pas été incluses dans Paged.js parce qu'elles n'ont pas encore été prises en compte dans le langage CSS (par exemple, les images en pleine page ou les notes en marge). Les designers graphiques ont alors la possibilité de développer des plugins pour l'outil, c'est-à-dire des programmes sous forme de scripts conçus pour être intégrés à un autre logiciel via une interface dédiée, afin d'ajouter de nouvelles fonctionnalités sans modifier le code source du logiciel principal.

Cela implique le développement d'un code cohérent avec le logiciel principal, suffisamment générique pour s'adapter à une variété de situations, tout en étant assez spécifique pour répondre aux besoins

particuliers de chaque projet. Julien Bidoret, enseignant à l'École supérieure d'art et de design des Pyrénées, a par exemple travaillé un script permettant d'imposer les pages en cahier pour l'impression. Julien Taquet, designer pour la Collaborative Knowledge Foundation, a lui développé un script permettant de positionner les images en fonction de la page. Certains de ces plugins vont au-delà des besoins de mise en page, comme le script développé par Nicolas Taffin, designer graphique et co-fondateur des éditions C&F, qui permet de recharger la page web à la hauteur à laquelle elle se trouvait avant le rechargement, facilitant ainsi les allers-retours fréquents des designers graphiques entre l'écriture de leur code et la visualisation dans le navigateur.

D'autres scripts peuvent être plus sécifiques comme celui développé par Gijs De Heij, designer graphique d'Open Source Publishing, qui permet d'afficher du texte sur la tranche d'un livre. Pour que ces scripts deviennent des « plugins », ils doivent souvent être retravaillés pour être rendus plus génériques, c'est à dire ne pas être utilisables uniquement dans un projet spécifique, mais dans une diversité de projets de même type.




The Kin Arts Almanac, State of the arts (2024), mis en page par Open Source Publishing. À droite: capture d'écran de la mise en page dans Paged.js

En étant écrits de manière suffisament générale et adaptable, ces plugins peuvent être facilement repris par d'autres designers graphiques. Pour la mise en page du livre Controverses. Mode d'emploi pour le programme Forccast et les Presses de Sciences Po, Sarah Garcin a disposé les notes en marge, à hauteur de l'appel de note. Pour cela, elle s'est servi d'un plugin nommé « margin-note » que j'avais moi-même développé quelques semaines auparavant pour un autre projet.




Clémence Seurat, Thomas Tari, Controverses. Mode d'emploi, Presses universitaires de Sciences Po, mis en page Sarah Garcin.

La plupart des scripts additionnels à Paged.js ont ainsi été créés pour répondre à des besoins spécifiques rencontrés dans des projets par les designers graphiques. Un important travail de recensement est à faire, car les designers graphiques n'ont pas toujours le temps ou la possibilité de partager, de rendre générique et de documenter les scripts qu'ils·elles ont développés dans le cadre de projets de commande ou de projets personnels [33]. Cependant, c'est bien l'accumulation de ces scripts et leur articulation qui contribuent à la conception de l'outil et qui inscrivent cette conception dans les pratiques collectives de la communauté. En somme, la conception par plugin permet une mise en commun avec la communauté et une reprise de ces productions dans l'activité singulière des sujets lorsque des situations similaires à celles qu'ils ont contribué à résoudre se présentent.

## Penser La Transparence Opérative : Prises Temporelles Et DOM

Les logiciels libres et les pratiques de programmation en open source sont souvent mis en opposition aux « boîtes noires » que représenteraient les logiciels propriétaires. Consécutivement, les outils produits sur un mode « ouvert » seraient considérés comme des boîtes de verre, voir des « boîtes magiques [34] » permettant d'explorer toutes les possibilités des ordinateurs par le biais du code.

Seulement, il est assez facile de se rendre compte que la nature ouverte du code n'est pas suffisante pour considérer les outils comme accessibles et réappropriables [35]. Par exemple, la modification du code source de Paged.js peut s'avérer complexe pour les designers graphiques dont l'activité de programmation n'est pas première. Comprendre l'organisation des dizaines de milliers de lignes de code de l'outil et les concepts de programmation intégrés à sa conception nécessite en effet une certaine maîtrise du langage JavaScript. Pourtant, nous avons vu précédemment que certain·e·s designers graphiques sont capables de développer des plugins afin d'améliorer les fonctionnalités de l'outil.

Ceci est rendu possible grâce à la manière dont l'outil à été conçu. Ainsi, pour interagir avec le code source de Paged.js, les designers graphiques s'appuient sur l'utilisation de deux ressources qui permettent une certaine interface avec le code source de l'outil. La première est l'utilisation de différentes « prises temporelles » présentes dans le script de l'outil (les hooks) et la deuxième est le code HTML et CSS transformé par le script (le DOM, Document Object Model).

Pour comprendre, expliquons d'abord brièvement une partie du fonctionnement technique de Paged.js. Son but essentiel est de fractionner un flux de contenu HTML en différentes pages tout en y appliquant les styles CSS prévus par les designers graphiques. Comme tout script, il est constitué d'instructions s'exécutant automatiquement selon un déroulé temporel déterminé. Simplifions quelques étapes : création d'une page selon le format défini, ajout des éléments HTML dans l'ordre où ils apparaissent dans le code, création de nouvelles pages si nécessaire, et répétition jusqu'à ce que tout le contenu soit placé. D'autres instructions sont encodées et liées aux différents mécanismes de mises en page (notes en bas de page, placement des numéros de pages et des titres courants, création de gabarits de page, sauts forcés, etc.).

Le code utilise des « prises temporelles » prédéfinies nommées hooks [36] pour déterminer quand certaines actions doivent se produire. Par exemple, la fonction `beforePageLayout()` indique que les instructions qui y sont liées doivent s'effectuer à chaque fois qu'une nouvelle page est créée. Une vingtaine de hooks sont ainsi disponibles (et documentés sur le site de Paged.js) et peuvent être ré-utilisés autant de fois que nécessaire. Ils permettent alors de « brancher » au code source de Paged.js des scripts complémentaires pour améliorer les fonctionnalités de mise en page, sans que les designers graphiques n'aient à comprendre tous les détails du code source, souvent complexe à appréhender.

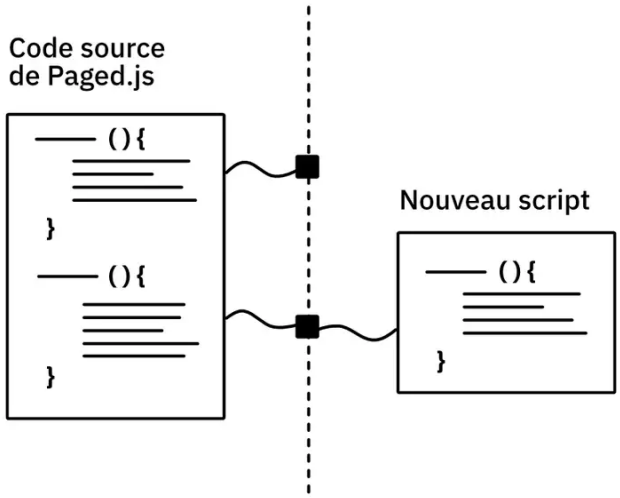
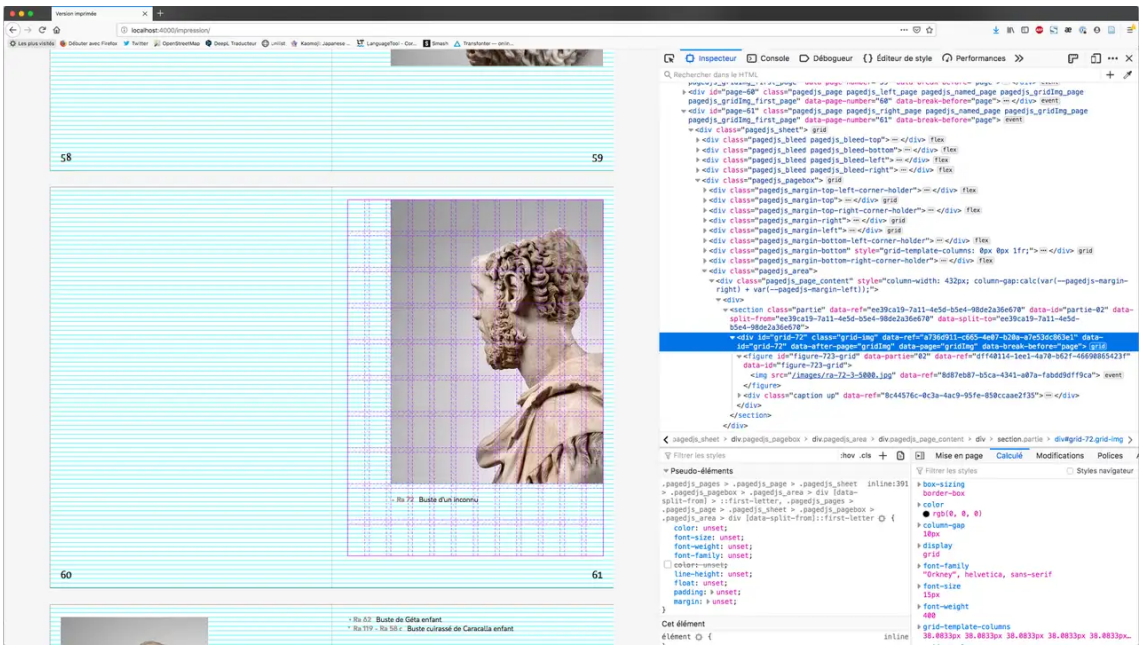



Schéma montrant l'ajout d'un nouveau script via les « prises temporelles » (hooks) proposées par Paged.js

La deuxième ressource sur laquelle les designers graphiques peuvent s'appuyer pour améliorer les fonctionnalités de mise en page de Paged.js, se situe directement dans le résultat produit par le script. La particularité de Paged.js est de ne pas produire directement un PDF, mais d'afficher un contenu paginé dans le navigateur web (une sorte d'aperçu du PDF). Pour contruire les pages de l'ouvrage, Paged.js ajoute un grand nombre d'éléments HTML et de styles CSS au code produit par les designers graphiques. Ces ajouts sont visibles depuis les outils d'inspection des navigateurs web, qui montrent le HTML et le CSS tel que transformé par Paged.js lors de l'affichage (on parle ici du DOM, le Document Object Model qui définit la hiérarchie des éléments sur la page et peut être manipulé pour changer

dynamiquement le contenu de la page). Les designers peuvent alors se servir de ces informations pour intervenir à leur tour sur le HTML transformé ou récupérer dans le CSS des éléments utiles pour leur mise en page (comme les noms des classes des éléments ajoutés ou les variables utilisées dans la construction des pages).



Capture d'écran d'un projet dans un navigateur web où l'on peut voir les éléments HTML et ajoutés par Paged.js affichés depuis les outils d'inspection.

La figure ci-dessus permet d'apprécier le code HTML transformé par Paged.js affiché depuis l'outil d'inspection du navigateur web (sous-fenêtre supérieure) ainsi que le code CSS ajouté (sous-fenêtre inférieure). Les nouveaux éléments HTML sont nommés selon une nomenclature qui rend leur fonction assez claire pour les personnes ayant des connaissances en HTML. Pour un·e designer graphique habitué·e à l'utilisation des technologies du web, il est ainsi facile de comprendre que la balise `<div class="pagedjs_margin-top">` contenu dans la balise `<div class="pagedjs_pagebox">` correspond à un élément indiquant la marge supérieure de la page. Il·elle peut alors utiliser cette information pour ajouter dans son code CSS des styles qui s'appliqueront spécifiquement à cet élément.

L'addition de ces deux ressources – les hooks et le DOM – offre donc des ressources aux designers graphiques pour concevoir de nouvelles fonctionnalités, et éventuellement des plugins génériques partageables avec la communauté.

Ainsi, Paged.js est conçu de manière à rendre visible une partie de son fonctionnement sans avoir besoin d'accéder à son code source. Cette particularité permet aux designers graphiques d'appréhender l'outil à partir de représentations laconiques de son fonctionnement construites sur la base de « prises » encodées dans Paged.js et de la mise en visibilité des effets de l'outil (le HTML et le CSS transformés). Pierre Rabardel utilise le concept de « transparence opérative » pour expliquer cette idée.

Depuis le point de vue de l'approche instrumentale, le concept de transparence opérative désigne « les propriétés caractéristiques de l'instrument, pertinentes pour l'action de l'utilisateur, ainsi que la manière dont l'instrument les rend accessibles, compréhensibles, voire perceptibles pour l'utilisateur[37] ». La transparence d'un artefact n'est donc pas à considérer comme une propriété interne stable d'un outil, elle doit être appréhendée comme un concept relationnel entre un outil et la personne qui utilise cet outil. Les besoins en information sur le fonctionnement ou l'état de l'outil

varient selon les situations et il peut être plus pertinent pour les designers graphiques que le fonctionnement interne du code soit caché ou révélé (en partie ou en totalité).

Intégrer à l'outil des ressources intermédiaires telles que nous les avons décrites, permet ainsi de comprendre la logique du code sans avoir à en maîtriser la complexité. Une partie du code est cachée pour mieux révéler son fonctionnement : c'est parce qu'il y a transparence opérative qu'une appropriation est possible et que les designers graphiques peuvent développer collectivement des outils.

## « Écrire Le Code Du **Code** » : Les Standards

Les standards sont un référentiel commun et documenté destiné à harmoniser un secteur d'activité. Ils se basent sur des recommandations et apparaissent notamment lorsqu'une masse critique de membres d'une communauté de pratique les apprécie suffisamment pour les utiliser collectivement. [38] Dans le champ de l'informatique, les standards permettent d'obtenir une compatibilité accrue entre plusieurs programmes ou matériels. Cette interopérabilité est une condition essentielle pour le partage de ressources, et donc la collaboration. Les standards apportent ainsi stabilité et viabilité aux outils, objets et publications qu'ils permettent de produire.

Par exemple, le web repose sur la rétrocompatibilité de ses langages, c'est-à-dire, l'idée que rien ne doit être cassé et qu'un site web codé vingt ans plus tôt doit toujours être lisible aujourd'hui [39] (s'il n'est pas retiré d'un serveur, ce qui est une autre question). Il est ainsi toujours possible d'apprécier la première page web publié en 1991 et accéder à son code source [40]. Cela est rendu possible car HTML et CSS sont des standards qui fonctionnent sur l'idée de leur amélioration progressive par cumulation. Leurs spécifications sont publiées publiquement et peuvent être utilisées par n'importe qui de la même manière, et notamment par les fabricants de navigateurs web.

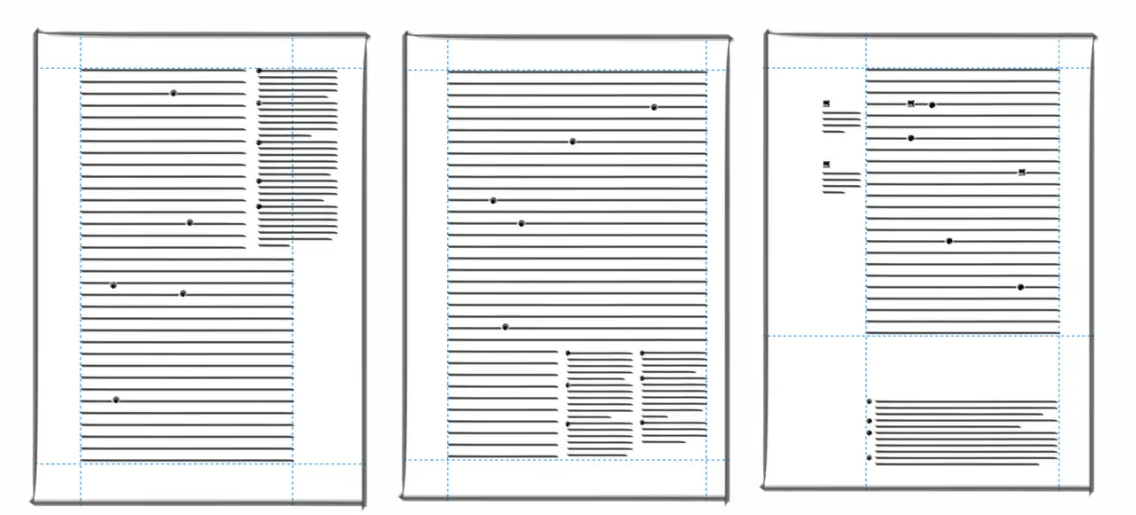
À contratio, les créations au format Flash, plate-forme qui a longtemps dominé le secteur de la conception de contenu multimédia sur le web, sont aujourd'hui illisibles sur la plupart des navigateurs web. En effet, la technologie utilisée n'était pas standard et nécessitait un interpréteur particulier maintenu par Adobe. Le logiciel Flash ayant cessé d'être maintenu, il n'est plus possible d'apprécier les milliers de créations qui ont été développées avec.

Revenons aux technologies du web. Leur évolution repose sur un processus de standardisation ouvert et communautaire piloté par le World Wide Web Consortium (W3C). Le CSS Working Group (CSSWG) travaille par exemple à l'amélioration du langage CSS de manière active depuis 1997 par le biais de recommandations qui décrivent sa syntaxe et spécifie le comportement attendu de chaque propriété lorsqu'elles sont utilisées dans les navigateurs web. Le groupe est particulièrement actif, et compte, en décembre 2022, 147 membres composés de diverses organisations et d'expert·e·s invité·e·s [41]. En réponse à des demandes de fonctionnalités ou à un besoin perçu, chaque proposition est ainsi débattue, adoptée ou rejetée par une communauté constituée de représentant·e·s d'entreprises, de développeur·euse·s de navigateurs, de développeur·euse·s web, de chercheur·euse·s, etc.[42]

C'est donc bien la collaboration entre différents individus et différentes organisations qui permet l'évolution du langage CSS. Si nous nous intéressons à cet aspect de la conception des standards, c'est parce qu'elle déplace encore d'un cran la question du rôle que les designers graphiques peuvent avoir dans la construction de leurs propres outils et fait entrer la collaboration dans un plan méta de la conception.

Reprenons l'exemple de Paged.js. Dans les sections précédentes de ce texte, nous avons vu que de nombreuses fonctionnalités, indispensables au travail de composition des designers graphiques n'ont pas été pensées dans l'écriture des spécifications CSS du W3C. Même si des plugins sont proposés pour pallier certains manques, ils ne représentent pas une solution suffisamment pérenne et sont étroitement associés à l'outil. Julien Taquet et moi-même avons donc décidé de travailler sur la rédaction de nouvelles spécifications, en nous concentrant dans un premier temps sur la disposition

des notes dans la page. Les notes en bas de page (footnote) ont déjà une syntaxe dédiée dans le module CSS Generated Content for Paged Media Module mais rien n'existe pour d'autres types de notes (notes groupées sur le côté, note en marge, notes en fin de colonnes, zones de notes multiples, etc.). Le 13 mai 2020, nous proposons un brouillon de spécifications présentant de nouvelles syntaxes CSS pour ces types de notes. Le brouillon est publié dans une issue du répertoire dédié au CSS Print Community Group sur le Github du W3C [43].



Quelques dispositions de notes détaillées dans les spécifications proposées.

Il ne s'agit pas ici de concevoir pour un projet spécifique ou pour un outil, mais de penser les standards à la base de l'interopérabilité entre plusieurs outils. Je souhaite inviter aujourd'hui les designers graphiques à participer activement aux discussions autour de CSS. Ce dernier est encore aujourd'hui le seul langage entièrement dédié à la présentation visuelle de documents et au rendu graphique des pages web. C'est une invention unique, « radicale » [44], dans le domaine informatique pour répondre à un enjeu jusqu'alors jamais imaginé dans l'histoire des médias : pouvoir, techniquement, exprimer à quel type de périphérique et quelle taille d'écran une règle stylistique particulière doit s'appliquer. Il est de ce fait, un langage profondément graphique, parfaitement adapté à la mise en page de tous types documents. Certes, participer à sa conception implique un certain savoir technique et avoir la capacité à se conformer à des règles extrêmement codifiées. Cependant, cela en vaut la peine puisque participer à la conception du langage CSS (« écrire le code du code [45] »), c'est participer profondément à la construction d'outils communs et durables.

Enfin, notons que la question de la standardisation dans le design graphique soulève des préoccupations quant à une possible uniformisation. Toutefois, il est crucial de faire la distinction entre standards reposant sur le bien commun et standards imposés par une industrie dominant un marché. Comprendre les enjeux de leur création et y participer permet de rester vigilant·e·s contre les appropriations privatives et d'interroger les implications futures des réutilisations qui se tourneraient vers des logiques extractivistes et hégémoniques.

Les standards, comme ceux issus de la culture du libre, peuvent créer une base commune, sans nécessairement restreindre la créativité. Tout comme l'utilisation des grilles dans le design graphique, qui offrent une structure tout en permettant des variations créatives. Ce socle partagé favorise l'apparition de communautés de pratiques, dont les membres s'attachent à une culture commune appréciable à travers ses références esthétiques, ses règles implicites et ses codes [46]. Beaucoup de sites les plus créatifs se jouent ainsi de références explicites au fonctionnement des technologies du

web, que cela soit leur façon de structurer l'information [47], leurs styles par défaut [48] ou encore leur nature profondément déclarative [49].

Cette dynamique n'est possible qu'en ayant connaissance des technologies et de leur dimensions éminemment politiques et sociales. Ainsi, avoir la possibilité de discuter collaborativement des standards, c'est avoir la possibilité d'y intégrer les perspectives de ses réutilisations futures, possibles et désirables. D'où la nécessité de disposer d'espaces informels tels que PrePostPrint, qui encouragent des discours politiques et non marchands sur la construction de communs. Le développement d'outils partagés nécessite l'établissement d'un langage commun (de pratiques et de standards) compris par une communauté élargie d'acteur·ice·s (W3C, organismes de constructions des navigateurs web, développeur·euse·s web, etc.) mais il est tout aussi important de préserver les possibilités de frictions et de controverses.[50]

## Conclusion : Construire Des Communautés De Pratiques

En conclusion, les pratiques de design graphique décrites dans cet article mettent en lumière l'importance de l'organisation, de la conception et de la programmation collaboratives à différents niveaux, allant du simple copier-coller à la création de standards. Cette diversité de pratiques participe à la création de communautés de pratique qui, en partageant leur travail et leurs intérêts de manière collaborative, développent non seulement un vocabulaire et des méthodes propres, mais aussi modifient les dynamiques socio-économiques du travail graphique.

Particulièrement, la culture du logiciel libre, marquée par la collaboration, le partage et l'apprentissage collectif, invite à repenser la relation « utilisateur·ice·s/concepteur·ice·s ». Elle montre une voie par laquelle les designers graphiques ont la possibilité de penser la conception de leurs propres outils. Ce positionnement marque une situation inédite pour la profession. L'histoire du design graphique a en effet montré des changements provenant souvent de l'extérieur du métier par la production de nouvelles technologies, machines et logiciels imposés par des entreprises tierces. Or, ici, comme nous l'avons montré avec Paged.js, le développement des outils résulte en partie du travail des designers graphiques qui les utilisent. Cela est rendu possible par la construction d'une communauté de pratique dont les actions sont bien plus larges que la seule participation au code source.

Ces pratiques trouvent un écho de plus en plus fort chez les designers graphiques concernés par les enjeux politiques, écologiques et économiques de leur métier. Utiliser des outils libres est ainsi considéré comme une façon de résister aux grandes entreprises de l'informatique, tout en obtenant une valorisation personnelle et l'estime des autres. De même, la figure du « hacker » est particulièrement valorisée et serait une base pour aborder la technologie de manière critique et créative.

Cependant, ces discours tendent à négliger plusieurs aspects cruciaux de la culture du logiciel libre et notamment l'exigence de participation à l'activité collective et la production effective de code partageable. À travers une conversation imaginaire entre un designer et un hacker, Anja Groten, membre du collectif néerlandais Hackers & designers, exprime ainsi sa méfiance à l'égard des approches du hacking par les designers qui oublient bien souvent la sociabilité inhérente à ce type de pratiques et s'en tiennent à utiliser son jargon pour repenser les méthodes de design.

> 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.[51]

La position critique des designers ne s'accompagne ainsi pas toujours d'une réelle participation à la communauté, au risque de rester dans la superficialité. Dans les faits, peu de designers graphiques qui utilisent des outils et logiciels libres et open source participent à leur développement ou au partage collectif (notons que cela est aussi vrai à une échelle plus large : peu d'utilisateur·ice·s des logiciels libres participent à leur développement). Nous observons donc une certaine dissonance entre de forts discours politiques et des pratiques concrètes où les logiciels libres et les technologies du web sont utilisés dans un simple rapport de consommation.

> L'ouverture des logiciels n'est pas un acte déclaratif. Dire qu'un logiciel est libre ne préserve pas la liberté des utilisateurs. La charge de liberté se trouve dans la distribution du code source ainsi que dans sa documentation. L'accès au code source du logiciel ainsi que la détermination pédagogique qui l'accompagne sont nécessaires à la transindividuation des communautés du logiciel libre.[52]

Les aspects de socialité et de communauté créés par ces pratiques deviennent des ingrédients cruciaux dans la production matérielle d'objets et d'artefacts. L'exigence du « contre don technologique [53] », du renvoi à l'autre, est un élément essentiel dans la construction des outils libres et open source. Le modèle promu par la culture du libre favorise ainsi la création d'une communauté de partage où chacun·e peut mettre les fruits de son travail au service des autres. Chaque personne y travaillant est sûre que son travail profitera à tou·te·s, de la même manière qu'elle-même pourra profiter du travail de tou·te·s. Il est donc important de penser la participation à ces dynamiques.

L'exigence d'une utilisation non-passive des technologies et d'engagement ne va pas sans certaines difficultés. Les designers graphiques qui s'y risquent sont ainsi continuellement exposés à toutes sortes de conflits et à des choses qui ne fonctionnent pas. Cependant, Anja Groten rappelle que la frustration (face à ces quelques lignes de code qui ne marchent pas), la rencontre avec la résistance et les dilemmes font partie intégrante de la culture hacker, parce qu'elle est justement ancrée dans le faire (« making ») et que c'est là son travail politique [54].

Il y a urgence à penser un modèle économique et social afin de rendre ces pratiques soutenables dans le champ du design graphique. La question est de savoir comment transformer cette communauté en un groupe plus large et plus diversifié d'acteur·rice·s participant de manière concrète au développement des outils, à leur documentation et à leur valorisation, ainsi qu'à des démarches pédagogiques ouvertes. L'engagement dans des projets libre et/ou open source, bien que valorisé pour ses dimensions politiques et sociales, impose une charge de travail significative à un nombre souvent restreint de contributeur·ice·s, mettant en lumière les risques d'épuisement et la nécessité d'un soutien institutionnel.

À plus grande échelle, il est nécessaire d'aborder les aspects du travail en relation avec les communautés de pratique libres. En ce sens, la figure qui est invoquée n'est pas celle du militant ou du bénévole qui soutient gratuitement une cause, ou encore celle du « hacker / bidouilleur / bricoleur » qui détourne les technologies pour son usage personnel, c'est aussi celle du travailleur ou de la travailleuse qui apporte sa pierre à une production collective et collaborative.

La culture du logiciel libre a montré que des alternatives sont possibles, par l'articulation de technologies et changement social à travers la création de communautés sociales. Cette culture est ainsi digne d'intérêt « en ce qu'(elle) constitue une manière de conduire l'action collective en adéquation avec les objectifs qu'elle se donne [55] ». En ce sens, ces pratiques sont politiques parce qu'elles sont occupées par leur propre politique organisationnelle, ainsi que par les discours portés par les artefacts qu'elles conçoivent et font circuler. Déplacer l'idée qu'il faut « fabriquer ses outils » vers la construction de communautés de pratiques et leurs ressources, permettrait de contribuer à créer une société dans laquelle les individus sont acteur·rice·s d'un système qu'ils·elles modèlent en prenant part aux pratiques collectives. Cette proposition ouvre à un discours potentiel sur d'autres relations

possibles à la conception et l'utilisation d'outils dans le design graphique, à travers une approche collaborative et distribuée de la conception.

Merci à Antoine Fauchié et Yann Trividic pour leurs relectures et les discussions qui ont suivi.

1. Pour un panorama plus global de la diversité des pratiques de programmation dans le champs du design graphique, voir Julie Blanc et Nolwenn Maudet, « Code <-> Design graphique, Dix ans de relations », Graphisme en France, n° 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. 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. ↩
4. « La post-production est une culture du re- : remixer, reformater, remanier, réinterpréter, reprogrammer, replanifier, redémarrer, réafficher, recycler ». Andrew Blauvelt, « Outil (ou le designer graphique face à la post-production) », Azimut, n°47, 2017, pp. 88-103 . Traduit de l'américain par Véronique Rancurel (révision Samuel Vermeil). ↩
5. Stéphanie Vilayphiou et 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. ↩
6. Étienne Ozeray, Pour un design gaphique libre, mémoire de master, École nationale supérieure des Arts Décoratifs, 2014. ↩
7. Julie Blanc, Composer avec les technologies du web : Genèses instrumentales collectives pour le développement d'une communauté de pratique de designers graphiques, thèse de doctorat en ergonomie, Université Paris 8, Vincennes – Saint Denis, 2023. ↩
8. PrePostPrint est un groupe informel se questionnant sur les publications expérimentales conçues avec des outils libres. Cet article a été rédigé peu après la participation de PrePostPrint au Libre Graphic Meetings qui s'est déroulé du 9 au 12 mai 2024 à Rennes (France). Il a donc fortement bénéficié des discussions qui s'y sont déroulées. ↩
9. La complexité du développement web a considérablement augmenté au fil des années. Aujourd'hui, il est courant d'utiliser des code snippets, des plugins JavaScript, et des systèmes de gestion de contenu (CMS) pour créer des sites web plus interactifs et dynamiques. Cependant, il est toujours possible de créer un site web simple et fonctionnel en utilisant seulement le HTML, le CSS et un peu de JavaScript. ↩
10. Alexia de Visscher, « Du design de la page à la pédagogie du flux : le cas belge », op.cit. ↩
11. Un « fork » est une copie d'un projet de code source, souvent utilisé dans le développement libre et/ou open source. Il permet aux développeur·euse·s de travailler sur leur propre version du code sans affecter le projet original. ↩
12. Eric Schrijver, « No-one Starts From Scratch: Type Design and the Logic of the Fork », i.liketightpants.net, 9 octobre 2013 ↩
13. Françoise Decortis, Anne Bationo-Tillon, et 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, octobre 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. Cette idée a très largement été développée dans ma thèse. Notons, que dans le domaine du design graphique, la « composition » désigne, avant l'apparition de l'informatique, le travail qui consiste à assembler les caractères (de plomb) pour former des lignes de texte. Il nous paraît donc ici particulièrement opportun à adopter. ↩
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, éd. par 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, septembre 2007, pp. 101-25. ↩
21. Voir https://esadpyrenees.github.io/PageTypeToPrint/ ↩
22. Version web du fanzine disponible à cette adresse : https://avoidsoftware.sarahgarcin.com/index.html ↩
23. Julie Blanc et Benjamin G., «Médialab (1/3) : Une décroissance heureuse », julie-blanc.fr, 27 mars 2020 ↩
24. Manetta Berens, « html2print as a practice of boilerplates », osp blog, 23 juin 2023 ↩
25. Notons par exemple ses billets dédiés à son utilisation de Paged.js : Making-of d'une collection libérée : Addictions sur ordonnance, publié le 10 février 2019, et Dans les recoins de la double page (Paged.js à la maison, saison 2) publié le 1er mars 2021 ↩

26. Curseurs sans sou(r/c)is, à retrouver à cette adresse: 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. Voir https://www.luuse.io/projects/edri/ ↩

29. Voir https://gitlab.com/bonjour-monde/tools/dataface ↩

30. Paged.js est sous la licence MIT, une licence très permissive provenant de l'Institut de technologie du Massachusetts (MIT) qui implique très peu de limitations sur la réutilisation du code. La licence donne à toute personne téléchargeant le code source le droit illimité de l'utiliser, le copier, le modifier, le fusionner, le publier, le distribuer, le vendre et le « sous-licencier » (l'incorporer dans une autre licence). ↩

31. « En informatique, un script désigne un programme (ou un bout de programme) chargé d'exécuter une action pré-définie quand un·e utilisateur·ice réalise une action ou qu'une page web est en cours d'affichage sur un écran. Il s'agit d'une suite de commandes (...) qui permettent l'automatisation de certaines tâches successives dans un ordre donné. » Définition du Dictionnaire du webmastering par le Journal du net ↩

32. Paged.js est un « polyfill », c'est-à-dire un palliatif logiciel implémentant des fonctionnalités non encore nativement disponible dans ceux-ci (ici, les navigateurs web). ↩

33. Ce travail de recensement et de documentation a débuté sur un répertoire dédié nommé « Plugins » sur le le site Gitlab de la Coko Fondation ↩

34. David Crow, « Magic Box: Craft and the Computer », Eye 18, n° 70, 2008. ↩

35. Nolwenn Maudet, « Appropriating to share better », Our collaborative tools, 2023 ↩

36. En informatique, un hook est un mécanisme qui permet à un programme d'intercepter et de modifier le comportement d'un autre programme ou d'un système d'exploitation en insérant du code supplémentaire dans le flux d'exécution. ↩

37. Pierre Rabardel, Les Hommes et Les Technologies : Approche Cognitive Des Instruments Contemporains, Armand Colin, 1995, p. 151. ↩

38. 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" ) ↩

39. Page web disponible à cette adresse : http://info.cern.ch/hypertext/WWW/TheProject.html ↩

40. En ce sens, Jarrett Fuller décrit leur utilisation comme un acte radical : « To build a website with just these tools is a small protest against platform capitalism: a way to assert sustainability, independence, longevity. » ↩

41. Parmi ces organisations, nous retrouvons les trois grandes multinationales américaines développant des navigateurs web – Apple (Safari), Google (Chrome) et Microsoft (Edge) –, deux organisations développant des logiciels libres ou open source – Mozilla (Firefox) et Igalia –, et, ironiquement, la multinationale Adobe Inc. ↩

42. Sur le fonctionnement du CSSWG voir Fantasai, « about:csswg », novembre 2011 , consulté le 12/02/2022. ↩

43. Le jeudi 13 février 2020, dans le cadre de la conférence XML Prague, à l'Université d'économie de Prague (République tchèque)**, le** <!-- correction typo (espace) --> groupe CSS Print Community Group est formé avec pour objectif de travailler sur les spécifications CSS pour l'impression, présenter des cas d'utilisation et plaider pour de meilleurs implémentations dans les navigateurs. Ce sous-groupe du W3C renforcerait ainsi les travaux du groupe de travail CSS en se concentrant sur le CSS pour les médias imprimés et paginés. Ma présence lors de cette conférence m'a **poussée** <!-- correction orthographique (féminin) --> à proposer les spécifications mentionnées dans le texte. ↩

44. Julien Rossi, « Écrire le code du code », RESET. Recherches en sciences sociales sur Internet, n° 11, mars 2022. ↩

45. Miriam Eric Suzanne, « CSS Is Rad », Miriam Eric Suzanne, 2020. ↩

46. Merci à Raphël Bastide de m'avoir soufflé cette idée. ↩

47. De nombreux manifestes mettent en avant l'importance que la structuration HTML a dans les pratiques du web : «Web design as architecture», « HTML energy ». ↩

48. Étienne Cliquet, « Esthétique par défaut. La beauté parfum vanille », Téléférique, serveur de téléchargement collectif & indépendant en ligne, août 2002. URL : https://web.archive.org/web/20110216053131/http://www.teleferique.org/stations/Cliquet/Default/ ↩

49. Voir à ce propos l'intitive Declarations, initiée et coordonnée par Doriane Timmermans. «Declarations is an 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 articulations. » ↩

50. Voir les réflexions de Julien Bidoret à ce propos : « Flossflop », publié le 15 avril 2024 sur son carnet en ligne ↩

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

52. 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. ↩

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

54. Groten, « Hacking & Designing », op. cit. ↩

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

NOTIONS

**Communauté**
Comme l'a si clairement dit Édouard Glissant, nous sommes humains, dans la mesure où il y a une diversité de cultures, de communautés, pourrait-on dire, d'« humanités », comme il le dit lui-même. Une communauté mondiale où tous les humains seraient « comme-un », où on aurait tous les mêmes modes de calcul, les mêmes modes de production, la même langue (à travers ce fantasme bien intentionné mais monstrueux que fut l'espéranto), ce serait un cauchemar bien davantage qu'une utopie. Le vrai défi, ce n'est pas du tout de créer de la communauté en aval où l'on soit tous d'accord, mais au contraire de permettre la coexistence de communautés diverses, qui restent diverses (par leurs langues, cultures, pratiques, significations), qui ne fusionnent pas du tout, parce que c'est de l'échange entre ces diversités de significations, de sensibilités, de constructions d'attention, que le développement de l'humain peut se faire.
L'enjeu de cette construction de commun global, qui est nécessaire, parce que nous n'avons qu'une seule planète, c'est de fixer ce qui doit être « communautisé » et mis en commun parce que ce sont des conditions de survie – et ce qui doit être protégé de l'homogénéisation, pour garder, de même qu'il y a la biodiversité de la vie animale, la biodiversité culturelle. Cela, ce n'est justement pas viser une communauté, c'est comprendre comment est possible la coexistence pacifique et inter-stimulante de communautés différentes… avec assez de frictions pour faire des étincelles éclairantes, mais pas trop pour ne pas risquer de prendre feu. (Astruc & Citton, 2016)

**Engagement**
Chaque fois que je me suis engagé (pratiquement ou théoriquement) dans un certain domaine d'activité, je me suis toujours demandé quelles étaient (ou pourraient être) les convergences possibles de ce domaine avec d'autres voisins ou des pays lointains. Ce qui m'a souvent amené,

**Empowerment**
**Le discours sur l'empowerment radical :** Né dans les années 1970, ce discours a été défendu par les féministes, les militants noirs et les activistes de l'éducation populaire, et il s'est concentré sur la libération collective en termes auto-émancipateurs et explicitement politiques. Il mettait l'accent sur les aspects égalitaires, participatifs et locaux et, fondamentalement, sur la capacité des organisations de terrain à prendre le pouvoir sur leur propre vie et à démanteler les systèmes oppressifs comme condition nécessaire au changement social.
**Le discours réformiste et libéral sur l'empowerment :** Puis, dans les années 1980, une première appropriation du sens par l'élite s'est produite, et « à partir de ses racines radicales, nous commençons à voir son incorporation et son institutionnalisation dans les systèmes de gouvernance de l'État » (McLaughlin, 2016, p.5). Le discours sur l'empowerment a évolué vers un cadre plus réformiste et libéral, avec des méthodes pour réformer sans menacer les systèmes en place. L'accent mis précédemment sur la conscientização (Freire, 2017) a été remplacé par des notions axées davantage sur des notions empruntées au champs de la thérapie et du développement personnel. Les figures de proue du changement sont devenues des professionnels et des experts. L'empowerment a ainsi privilégié une approche étatique, plutôt que communautaire, de la gouvernance hybride et de l'autogestion légère, en particulier dans le domaine du développement communautaire, comme alternative aux anciens types de gouvernement institutionnalisé, de type descendant, considérés comme paternalistes et obsolètes.
**L'évolution néolibérale de l'empowerment :** Enfin, le saut le plus important dans l'évolution du discours sur l'empowerment a peut-être été le virage néolibéral complet à partir des années 1990, lorsque les intérêts financiers sont entrés dans la conversation. Depuis lors, l'empowerment a été dépouillé de ses significations politiques et militantes et reformulé comme capacité entrepreneuriale des individus à s'insérer dans la société de marchande. Il s'est définitivement intégré dans les discours mondiaux sur l'expertise managériale et financière par le biais de programmes et de politiques menés par des institutions internationales telles que les Nations unies et la Banque mondiale et, par le biais de la déréglementation et de la privatisation, les gouvernements centraux ont abandonné leurs attributions de pouvoir, de responsabilité et de ressources au profit du marché libre. » (Volpi et al., 2024)

dans ma réflexion, à aller au-delà de l'activité que je faisais. Bien que ma formation ait été principalement artistique, je dois dire que depuis le début, mes intérêts vont bien au-delà du domaine spécifique de l'art. Plus mon engagement envers la pratique artistique était fort, plus mes intérêts étaient élargis à des sujets ayant des implications philosophiques, scientifiques et sociologiques. Et aussi et surtout politiques. En fait, déjà très jeune, mon souci (je dirai presque mon obsession) était de pouvoir contribuer à une vision globale de la culture. (Maldonado, cité et traduit dans Geel, 2019)