Julie Blanc
contact@julie-blanc.fr
Our collaborative tools, https://ourcollaborative.tools/
18/05/2024

Proposal

Table des matières

- Programmation web et collaboration dans le design graphique, glossaire de quelques pratiques
  - Ouvrir et modifier le code: balisage et mode texte
  - Organiser son code: pratiques d'écritures codifiées
  - Documenter: des formes multiples de partage
  - Concevoir par modularité: ajout de fonctionnalités
  - Penser la transparence opérative: prises temporelles et DOM
  - « Écrire le code du code »: les standards
  - Conclusion: construire des communautés de pratiques

Programmation web et collaboration dans le design graphique, glossaire de quelques pratiques

---

Dans le design graphique, la collaboration a parfois tendance à être entendue uniquement comme une participation directe, circonscrite dans un temps et un espace donnée alors que ces formes sont multiples.

Cet article explore comment la culture du libre, et plus particulièrement ses aspects collaboratifs, sont intégrés dans certaines pratiques des designers graphiques, notamment 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], et notamment dans les pratiques de publication [2].

La culture du libre est notamment basée sur l'utilisation de de programmes et logiciels libres dont les caractéristiques principales sont les possibilités d'accéder, modifier et dupliquer leurs code source. La Free Software Foundation considère ainsi un logiciel comme libre s'il réond à quatres libertés accordées aux utilisateur·rice·s: celles (0) d'utliser, (1) copier, (2) étudier et (3) modifier le programme. L'accès au code source est une condition d'exercice des libertés 1 et 3. La culture du libre est aussi un mouvement social caractérisé par des questions philosophiques, un

rapport créatifs aux technologies et aux manières de travailler ou encore des engagements militants (permettre aux utilisateur·rice·s de contrôler leur informatique, et produire des technologies socialement utiles).

De nombreux articles citent ainsi la culture logiciel libre comme un exemple à suivre pour le design graphique, notamment à travers des pratiques du code [3]. Il y est particulièrement défendu l'idée de construire une culture technique et un commun du design graphique, fondés sur la transmission des savoirs techniques permettant de faire rentrer la technique dans un processus de valorisation [4]. Le logiciel libre y est aussi abordé comme un exemple réussi d' une conception du travail radicalement différente dans un contexte de «post-production» abolissant la distinction traditionnelle entre production et consommation ou création et copie [5]. L'insertion de la programmation dans le design graphique est aussi valorisé 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 [6].

Bien que les aspects collaboratifs sont aussi souvent mis en avant dans ces textes, ils sont assez peu commentés depuis le point de vue spécifique du design graphique, où les participant·es à cette culture ne sont pas avant tout des programmeur·euses. Or le principe au fondement des logiciels libres –à savoir la mise à disposition du code source avec autorisation de le modifier et le distribuer librement- se traduit par la coopération de développeur·euses 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 champs du design web et du design graphique. Pour cela, nous proposons un lexique en six sections thématiques où nous aborderons plusieurs notions développées plus longuement dans notre travail de thèse [7]. Nous nous exprimons depuis un point de vue situé, à travers notre 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 indifférement pour de publication web ou de la publication imprimée.

Au-delà d'un travail critique, notre proposition est l'occasion de discuter et donner des exemples concrets des pratiques de la culture du libre. Nous montrons ainsi comment les principes de collaboration sont répartis dans différentes pratiques. Cela nous permet par la même occasion d'éclairer l'utilisation du code dans le design graphique non comme une pratique du bricolage et de « hacking » à laquel 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 situe dans l'utilisation des technologies du web, dont les deux principaux langages sont HTML et CSS. La possibilité d'intégrer et de publier divers médias (vidéos, processus génératifs, animations, etc.) dans un espace unifié 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), décrit la structure d'un site web. Par exemple, la balise `<body>` indique au site web où le contenu va commencer ; `<p>` indique qu'un nouveau paragraphe commence, `<h1>` un titre de niveau 1, etc. CSS, abréviation de *Cascading Style Sheets* (feuilles de style en cascade), est utilisé pour décrire la

présentation d'un document, son style. C'est en CSS que l'on déclare par exemple la police de caractère utilisé pour un titre ou la couleur d'un encart. 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 proposant des paradigmes de programmation relativement simples, plus accessibles que des langages nécessitant une maîtrise approfondie des concepts d'algorithmie ou un processus de compilation. Il suffit d'un navigateur web pour les interpréter, logiciel que nous possédons tous sur nos ordinateurs (et nos smartphones). 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, ce qui rend rend par ailleurs très facile la reprise directe, par un simple copier/coller puisque l'écriture ne dépend d'aucun logiciel particulier. 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. Ainsi, la reprise de bout de code écrit par autrui (de quelques syntaxes à plusieurs lignes) est une pratique hégémonique quel que soit le niveau de compétence en programmation. Cette approche, bien que moins ambitieuse que le « fork » [11], est la première forme de collaboration entre designers-développeur·euses. (À partir d'ici, nous utiliserons indistinctement les deux termes.)

Publier un code sur une plateforme dédiée permet d'afficher clairement une volonté de reprises et de modifications par d'autres. Par exemple, CodePen est un site communautaire de partage d'extraits de code HTML, CSS et JavaScript fonctionnant comme un éditeur de code source où les développeur·euse·s inscrit·e·s peuvent créer directement des extraits de code et les tester de manière interactive. Ainsi, les designers graphiques puisent dans un grand nombre de ressources en ligne alimentés par une communauté de développeur·euse·s web expert·e·s ou amateur·rice·s.

Figure 1. Capture d'écran d'un projet sur CodePen

Cette culture de l'appropriation et de la réutilisation soulève des débats sur le droit d'auteur et l'économie du partage chez les designers graphiques. La notion d'autorité singulière rattachée depuis longtemps au design graphique est redistribuée par le partage d'outils et l'utilisation du code. En effet, bien que l'idée romantique d'un génie créateur ait été depuis longtemps réfutée, elle continue de hanter le domaine du design graphique. Cependant, le recyclage de morceaux de code, créés par soi-même ou par d'autres, illustre concrètement l'idée que les formes sont issues de diverses sources agrégées et réagencées.Comme l'a rappelé Vygotski et d'autres chercheur·euse·s après lui [12], 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, notons que coder permet de profiter concrètement de l'écosystème d'outils et de méthodes collaboratives développés depuis des décennies pour faciliter le travail de programmation. Depuis les années 1990, deux conceptions du logiciel libre s'affrontent entre *free software* et *open source*. 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. 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.

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·euses de gérer les modifications du code source (fusionner les mofifications, gérer les conflits) et facilite ainsi le travail collaboratif d'écriture du code. Les plates-formes de gestion de projets comme GitHub, GitLab et Bitbucket 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 les pratiques graphiques par l'usage de ces outils *fork*, *commit*, *version*, *issue*, etc.

Au-delà des outils techniques, la réflexion sur les licences logicielles revêt par ailleurs une importance capitale dans le domaine de la programmation libre. Les licences, telles que la GNU General Public Licens , la MIT License ou la Creative Commons, définissent les conditions d'utilisation, de distribution et de modification du code source. Ces licences philosophiquement fondées sur le partage et la liberté ont permis de créer un écosystème où la collaboration est encouragée et où les connaissances sont largement accessibles.

Ainsi, le logiciel libre repose sur un socle à la fois technique, légal et culturel, induisant des méthodes de travail collaboratives et horizontales qui redessinent le rôle des designers graphiques qui se revendiquent de cette culture.

Organiser son code: pratiques d'écritures codifiées

Écrire du code n'est pas seulement une compétence technique, mais une pratique socialisée, qui prend en compte la culture de la communauté dans laquelle les designers graphiques évoluent. La possibilité de mise en partage du code, inévitablement liée à des valeurs portées par la culture du logiciel libre et open source a ainsi une influence sur la façon d'écrire ce code. Ainsi, les designers graphiques ne mettent pas seulement en forme des objets graphiques en codant, ils travaillent aussi la mise en forme du code lui-même afin que celui-ci soit intelligible et lisible – condition indispensable pour permettre le partage de celui-ci.

La nécessité d'écrire un code source clair, rigoureux et bien structuré est une règle implicite dans la communauté des praticien·nes. Cette clarté est essentielle non seulement pour que le code soit compréhensible par les autres, mais aussi pour que les développeur·uses puissent eux·elles-mêmes naviguer efficacement à travers leur propre travail. Ainsi, différentes pratiques d'écriture codifiées existent pour rendre le code plus lisible et plus accessible.

La sématique du code est par exemple partivulièrement travaillée. La règle de pertinence prédomine ainsi lorsqu'il s'agit de nommer des fichiers, des classes HTML, des fonctions JavaScript ou des variables CSSl. 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. Une nommenclature trop abstraite ou personnelle entrave la possibilitié d'une lecture collective de ce code, et donc sa réappropriation.

L'organisation visuelle et graphique de la structure du code (utilisation de retours à la ligne, d'indentations, d'imbrications et d'espacements) permet également de refléter son organisation. Le code est ainsi organisé en des métaphores visuelles ou des représentations littéraires des relations entre les différents éléments du code afin de « laisser en vue » la structure. Il est aussi fréquent de décomposer le code en unités appréhendables en le divisant dans des fichiers dédié à une fonctionnalité précise.

En ce sens, écrire du code est un acte de composition[13]. Il est par ailleurs intéressant de rapprocher cette activité de mise en lisibilité du code, en particulier par sa structuration, à l'un des but affiché 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 pour les rendre visibles et lisibles sont au cœur des pratiques des designers graphiques de manière générale. Cette similitude dans les pratiques, nous donne un indice de 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. Cependant, en travaillant l'écriture du code, c'est à eux-même et à leur pairs développeur·euses que les designers graphiques s'adressent. 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 codifiée -, les développeur·euses s'assurent ainsi une réciprocité dans la constitution de leurs ressources. Ainsi, dans un entretien, la designer graphique Amélie Dumont 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. [14]»

Figure 2. 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 le designer graphique Benjamin Gremillon: « 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 possible, ça permet de mieux travailler avec d'autres personnes. [15] »

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 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. » [16] Ainsi peut se développer la collaboration.

Documenter: des formes multiples de partage

Alors la participation à la création de code source semble une évidence, le développement 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.[17]». Ce travail, souvent invisibilisé, voir dévalorisé, est pourtant crucial pour l'appropriation d'un outil ou un logiciel.

Nous prendrons ici l'exemple de la documentation, un élément essentiel dans toute communauté de pratique, et ce, à différents niveaux et sous diverses formes. La documentation n'est pas toujours associée à un outil spécifique, mais peut aussi être relative à un projet ou à l'exploration d'une technologie particulière. Elle participe à des pratiques collaboratives 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·ices et les développeur·icess à comprendre et à utiliser le logiciel.

Si les logiciels à interfaces graphiques ont habitué les designers graphiques à se passer de ces 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·es de l'École supérieure d'art et de design des Pyrénées, Julien Bidoret a ainsi fournit une documentation riche et exhaustive[18]. Cette dernière a été rédigée de manière à être lisible pour les étudiant·es 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 utilisateurs 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 célébre de ce genre d'approche, mais elle peut aussi être déployé à une échelle plus réduite et située. L'école de recherche graphique de Bruxelles (erg) possède ainsi son propre wiki où sont répertoriés les infromations de l'école et divers ressources des ateliers et cours.

Plus spécifique au monde de la programmation, les codes sources des programmes libres et open-source sont souvent distribués via des services comme **Github ou Gitlab**. Ces services sont basé 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 ait souvent accordé une grande importance.

Figure 3. Capture d'écran d'un répertoire de code sur GitLab

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

Figure 4. Capture d'écran de la vue d'un projet sur le site d'Open Source Publishing (osp.kitchen)

En dehors de ces pratiques, la documentation s'inscrit dans des modalités d'écriture auxquelles les designers graphiques sont peu habitué·es, et peut parfois semblait 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* [19], 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 ».

Figure 5. 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 Gremillion 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. [20] Nous y détaillons les principes de sobriété qui ont été choisi 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 [21].

Ce genre de publications peuvent aussi être consacrées 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 annotée et contextualisé [22]. 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*) à sa tentative d'adapter la revue *Curseurs* de Tactic à une mise en page programmée avec Paged.js. [23]

Dans le design graphique touné 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.

Concevoir par modularité: ajout de fonctionnalités

L'usage de la programmation dans les pratiques du design graphique, porte un imaginaire de création de ses propres outils. Ainsi, 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. [24]» En ce sens, la grande majorité des outils programmés par les designers graphiques sont développés pour des pratiques personnelles ou pour des projets spécifiques. Le collectif Luuse, a ainsi développé *H.I.R.P, Honey, I Resized the Poster* un outil structuré basé sur le web permettant de produire différents formats imprimés à partir d'un même contenu et utilisé pour la production de posters pour European Digital Rights (EDRi) [25]. Le collectif Bonjour Monde a lui imaginé *dataFace* [26], un outil programmatique expérimental de manipulation typographique intégrant des propriétés de font variables. L'outil a ensuite été utilisé lors de plusieurs workshops pour expérimenter de nouvelles formes typographiques. Les projets de ce type impliquent l'idée

d'une création sur mesure où l'outil lui-même devient la production des designers graphiques. Et parce qu'ils sont sur-mesure et adaptagé à un projet spécifiques, il est très rare que ces programmes soient repris en dehors du contexte pour lesquels ils ont été développés.

Penser la conception collaborative d'outils de plus grande ampleur et une autonomonie de production dans le design graphique exige de s'éloigner de cet imaginaire d'autorat. Nous proposons de nous intéresser ici à la conception d'outils génériques, s'adaptant à plusieurs cas d'usage, et, notamment par l'ajout de fonctionnalités dans des logiciels libres et open source déjà existants. Un plugin est ainsi un programme conçu pour être greffé à un autre logiciel à travers une interface prévue à cet effet, et apporter à ce dernier de nouvelles fonctionnalités sans modifier son code source. Cela implique la production d'un code développé en cohérence avec un programme principal, suffisamment générique pour s'intégrer à une diversité de situations, tout en étant assez spécifique pour répondre aux besoins particuliers de chaque projet.

Prenons pour exemple Paged.js, un outil et open source [27] 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 Java-Scrip , un langage de script comme son nom l'indique [28]. 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. Aucune autre action n'est requise pour son fonctionnement.

Figure 6. Schéma du fonctionnement de Paged.js

En cherchant à respecter strictement les standards CSS [29], certaines fonctionnalités de mise en page n'ont pas été incluses dans l'outil parce qu'elles n'ont pas encore été pensée dans le langage CSS (images pleines pages, notes en marges, etc.). Cependant, les designers graphiques peuvent avoir besoin de ces fonctionnalités pour la mises en page d'ouvrages. Ils·elles peuvent alors développer ces fonctionnalités sous forme plugins et les partager avec la communauté, fidèle à l'esprit de la culture du Libre. 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 développé un script permettant de positionner les images en fonction de la page. Certaines de ces plugins vont parfois au-delà des besoins de mise en page : Nicolas Taffin, designer graphique et co-fondateur des éditions C&F, a ainsi produit un script très utile permettant de recharger la page web à la hauteur de laquelle elle était avant le rechargement, facilitant les nombreux allers-retours des designers graphiques entre l'écriture de leur code et la visualisation dans le navigateur.

D'autres scripts peuvent être plus sécifiques. Dernièrement, Gijs De Heij, designer graphique d'Open Source Publishing, a fait la démonstration d'un script permettant d'afficher du texte sur la tranche du livre. Pour que ces scripts deviennent des « Plugins », ils doivent souvent être retravaillé pour être rendu plus générique, c'est à dire être utilisable uniquement dans un projet spécifique mais dans une diversité de projet de même type.

Figure 7. *The (Fair) 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

Ainsi, en étant écrit de manière suffisament génériques et adaptables, 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.

Figure 8. Clémence Seurat, Thomas Tari, *Controverses. Mode d'emploi*, Presses universitaires de Sciences Po (2021), mis en page Sarah Garcin.

Ainsi, la plupart des scripts disponibles ont été créé pour répondre directement à des besoins rencontrés dans des projets spécifiques par les designers graphiques. Un important travail de ressencement est à faire, les designers graphiques n'ayant pas toujours le temps ou la possibilités de partager, rendre générique et documenter, certains scripts développés lors de projets de commandes ou de projets personnels.[30] Pour autant, c'est bien l'accumulation et l'articulation de ces scripts qui participent à la conception de l'outil et inscrivent cette conception dans les pratiques collectives de la communauté. En somme, la conception par plugin permet une mise en partage 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 viendraient se présenter.

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[31] » permettant d'explorer toutes le 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 suffisance pour considérer les outils comme accessibles et réappropriables[32]. 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·es 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, nous observons que 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 (hooks et handlers) et la deuxième est le code HTML et CSS transformé par le script.

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, ajout du contenu 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 intégrés dans Paged.js (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* [33] pour déterminer quand certaines actions doivent se produire. Par exemple, la fonction `beforePageLayout()` indique que les instructions qui y sont liées doivent s'effectuer avant qu'une nouvelle page soient créée. Une vingtaine de *hooks* sont ainsi disponibles (et documenté sur le site de Paged.js) et peuvent être ré-utilisée autant de fois que nécessaire. Ils permettent alors de « brancher » au code source de Paged.js des scripts complémentaire pour améliorer les fonctionnalités de mise en page, sans que les designers graphiques n'aiet à comprendre tous les détails du code source de Paged.js, souvent complexe à appréhender.

Figure 9. Schéma montrant l'ajout d'un nouveau script via les «prises temporelles» (*hooks*) proposées par Paged.js

La deuxième ressource sur lesquels 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 HTML et CSS produit par les designers graphiques. Ces ajouts sont visibles depuis les outils d'inspection des navigateurs web, qui rendent visible 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).

Figure 10. 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 rendant assez claire leur fonction pour les personnes possédant une connaissance d'HTML. Pour un·e designer graphique habitué·e à l'utilisation des technologies du web, il est ainsi facilement compréhensible que l'objet `<div class="pagedjs_margin-top">` contenu dans l'objet `<div class="pagedjs_pagebox">` correspond à un élément indiquant la marge supérieure de la page. Il·elle peut alors se servir de cette information pour ajouter dans son code CSS des styles qui s'appliqueront spécifiquement sur cet élément.

L'addition de ces deux ressources – les fonctions temporelles et le code HTML et CSS ajoutés par Paged.js – 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[34] ». 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 où il peut être plus pertinent pour les designers graphiques que le fonctionnement interne du code soit caché ou révélé (en partie ou en totalité).

Autrement dit, lors de la conception de plugins, les designers graphiques doivent travailler avec le code source de Paged.js qui peut être trop complexe vis-à-vis de leurs compétences en programmation. L'accès à l'intégralité du code ne leur est donc pas utile. Intégrer à l'outil des ressources intermédiares telles que nous les avons décrites, permet alors de comprendre la logique du code sans avoir à en maîtriser la complexité. Une partie du code est caché 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

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 l'idée que rien ne doit être cassé et qu'un site web codé vingt ans plus tôt doit toujours être lisible aujourd'hui (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. [35] Cela est rendu possible car HTML et CSS sont des standards qui fonctionne sur l'idée de leur amélioration progressive par cumulation sont utilisables de la même manière depuis plus de 20 ans [36]. 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 fichiers Flash, plate-forme qui a lontempts étédominante pour le contenu multimédia sur le web, sont aujourd'hui illisibles sur la plupart des navigateurs web car la technologie utilisé n'étaient pas standard et nécessitait un interpréteur particulier maintenu par Adobe. Le logiciel Flash eyant cessé d'être maintenu, il n'est aujourd'hui plus possible d'apprécier les milliers de créations qui ont été développées avec.

Les évolutions d'HTML et CSS reposent sur un processus de standardisation ouvert piloté par le World Wide Web Consortium (W3C). Plus particulièrement, le *CSS Working Group* (CSSWG) travaille à l'amélioration du langage CSS de manière active depuis 1997 en standardisant sa syntaxe et en spécifiant le comportement attendu de chaque proprété 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 [37]. 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 représentants d'entreprises, de développeur·euse·s de navigateurs, de développeur·euse·s web, de chercheur·euse·s, etc. [38]

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. Nous avons déjà évoqué 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. Par exemple, il est seulement prévu de disposer les notes

en bas de page alors que certaines mises en pages requièrent de disposer les notes dans les marges verticales des ouvrages, ou encore à hauteur de l'appel de note. Même si des plugins sont proposé pour pallier ce manque, ils ne représentent pas une solution suffisamment pérenne. Julien Taquet et moi-même avons donc décidé de travailler sur la rédaction de nouvelles spécifications. 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 [39].

Figure 11. Quelques dispositions de notes détaillées dans les pécifications proposées.

Il ne s'agit ici ni de concevoir pour un projet spécifique ou pour un outil mais de penser les standards à la base de l'interopérabilité entre plusieurs outils. Cela implique aussi une très grande réflexivité sur sa pratique et que les designers soient capables de relier les futurs usages possible à des règles extrêmement codifiées. Cependant, cela en vaut la peine puisque participer à la conception du langage CSS (« écrire le code du code [40] »), c'est participer profondément à la construction d'outils communs et durables.

CSS 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 » [41] , dans le domaine informatique pour répondre à un enjeu jusqu'alors jamais imaginée 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, en y incluant la composition imprimée. [← paragraphe à supprimer ?]

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 distinguer entre standards reposant sur le bien commun et normes imposées par une industrie dominant un marché. Comprendre les enjeux de leur création et y participer permet de rester vigilant contre les appropriations privatives et 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'appartition de communauté de pratiques, dont les membres s'attachent à une culture commune appréciable à travers ses références ethétiques, ses règles implites et ses codes. [42] Beaucoup de sites les plus créatifs se jouent ainsi de références explicites au fonctionnement des technologies du web, et notamment en se référent à sa façon de structurer l'information [43], ses styles par défaut [44] ou encore leur nature profondément déclarative [45].

Cette dynamique n'est possible qu'en ayant connaissances des technologies et de leur dimensions éminament politique et sociale. 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 communs nécessite l'établissement d'un langage commun (de pratiques et de normes) compris par une communauté élargie d'acteurs et d'actrices (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 [46].

Conclusion: construire des communautés de pratiques

En conclusion, les pratiques que nous avons décrites dans ce glossaire nous invitent à accorder une attention particulières aux manière d'organiser, de concevoir et de programmer dans le design graphique, dans des dimensions collaboratives qui se situent à plusieurs niveaux, d'un simple copier-coller à la création de standards, en passant par des pratiques d'écritures codifiées. Cette diversité de pratiques participent à la création de communautés de pratique qui, en partageant leur travail et leurs intéret de manière collaborative, développent non seulement des vocabulaires et des méthodes propres, mais aussi modifient les dynamiques socio-économiques du travail graphique.

Particulièment, la culture du logiciel libre, marquée par la collaboration, le partage et l'apprentissage collectif, invite à repenser la relation « utilisateur/concepteurs ». Elle montre une voie par lesquels les designers graphiques ont la possibilité de penser la conception de leur propre outil. 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, mais est seulement possible parce que cette conception s'inscrit dans une communauté de pratique dont les actions sont bien plus large que la seule participation au code source.

Les valeurs portées par l'open source et les logiciels libres 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 produits libres open source 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 néerlendais Hackers & designers, exprime ainsi sa méfiance à l'égard des approches du hacking par les designers qui oublient bien souvent la sociabilité inhérente aux pratiques du hacking 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. [47]

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. 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. [48]

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 [49] », 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 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 [5O].

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 open source, bien que valorisé pour ses dimensions politiques et sociales, impose une charge de travail significative à un nombre souvent restreint de contributeur·ices, 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 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 [51] ». 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. En déplaçant l'idée qu'il faut « fabriquer ses outils » vers la construction de communautés de pratiques et leurs ressources, notre volonté est 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.

1. Pour un panorama plus global de la diversités 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*, nº 28 (2022) , pp. 3-30.↩
2. Loraine Futer« Trouble dans le genre — pédagogie alternative de l'édition hybride », *Design research, Publications hybrides* [en ligne] (2018), URL: https://www.design-research.be/hybrid/publications.html; Alexia de Visscher, « Du design de la page à la pédagogie du flux: le cas belge », *Back Office*, nº3 (2019) , pp. 122-135.↩
3. Stéphanie Vilayphiou et Alexandre Leray, « Écrire le design : vers une culture du code », Back Cover, nº 4 (2011), pp. 37-44; Kévin Donnot, « Code = design », *Graphisme en France*, nº18 (2012), pp. 4-12.↩
4. Antoine Gelgon, « Un dialogue à réaliser : design et technique », in.*txt 3* (Éditions B42 ; École supérieure d'art et de design Grenoble-Valence, 2018), pp. 38-58.↩
5. Andrew Blauvelt, « Outil (ou le designer graphique face à la post-production) », *Azimut*, nº47 (2017), pp. 88-103 . Traduit de l'américain par Véronique Rancurel (révision Samuel Vermeil).↩
6. Étienne Ozeray, *Pour un design gaphique libre*, mémoire de master ,École nationale supérieure des Arts Décoratifs (EnsAD) 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. http://www.revue-backoffice.com/numeros/03-ecrire-lecran/10_devisscher↩
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 développeurs de travailler sur leur propre version du code sans affecter le projet original.↩
12. 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*, nº 2 (octobre 2016) .↩
13. Cette idée a très largement était développé dans ma thèse. Notons, que dans le vocabulaire du métier, 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.↩
14. Entretien réalisé dans le cadre de ma thèse.↩
15. *Idem*.↩
16. 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 (Washington, DC : International Institute for Ethnomethodology and Conversation Analysis [u.a.], 1995), pp. 231-86.↩
17. 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, nº 2 (septembre 2007) , pp. 101-25.↩
18. Voir https://esadpyrenees.github.io/PageTypeToPrint/↩
19. Version web du fanzine disponible à cette adresse: https://avoidsoftware.sarahgarcin.com/index.html↩
20. https://julie-blanc.fr/blog/2020-03-25_medialab-1/↩

21. Notons par exemple ses billets dédié à son utilisation de Paged.js: *Making-of d'une collection libérée : Addictions sur ordonnance*, publié le 10 février 2019 (https://polylogue.org/addictions-sur-ordonnance-making-of-dune-collection-liberee/), et *Dans les recoins de la double page* (*Paged.js à la maison, saison* 2) publié le 1er mars 2021 (https://polylogue.org/apres-la-page-la-double-page/)↩

22. http://blog.osp.kitchen/residency/a-practice-of-boilerplates.html↩

23. *Curseurs sans sou*(*r/c*)*is*, à retrouver à cette adresse: https://web.archive.org/web/20240521122022/https://garden.delyo.be/journalbord/log.html↩

24. Kévin Donnot, « Code = design », *Graphisme en France*, n°18 (2012), pp. 4-12.↩

25. Voir https://www.luuse.io/projects/edri/↩

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

27. 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).↩

28. « En informatique, un script désigne un programme (ou un bout de programme) chargé d'exécuter une action pré-définie quand un utilisateur·rice 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, URL: https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203599-script-definition/↩

29. 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).↩

30. Ce travail de ressencement et de documentation a débuté sur un répertoire dédié nommé « Plugins » sur le le site Gitlab de la Coko Fondation; visible à l'adresse suivante https://gitlab.coko.foundation/pagedjs/pagedjs-plugins)↩

31. David Crow, « Magic Box: Craft and the Computer », *Eye* 18, nº 70 (2008).↩

32. Nolwenn Maudet, « Appropriating to share better », *Our collaborative tools* [en ligne], 2023, URL: https://ourcollaborative.tools/article/appropriating-to-share-better↩

33. 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.↩

34. Pierre Rabardel, *Les Hommes et Les Technologies; Approche Cognitive Des Instruments Contemporains* (Armand Colin, 1995), p. 151.↩

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

36. 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. » https://www.untappedjournal.com/issues/issue-11/jarrett-fuller-building-with-simple-tools-longevity↩

37. 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. ↩

38. Sur le fonctionnement du CSSWG voir Fantasai, « about:csswg », *https://fantasai.inkedblade.net/*, novembre 2011 , consulté le 12/02/2022.↩

39. Le jeudi 13 février 2020, dans le cadre de la conférence *XML Prague*, à l'Université d'économie de Prague (République tchèque),le groupe *CSS Print Community Group* (https://www.w3.org/community/cssprint/) 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 poussé à proposer les spécifications mentionnées dans le texte.↩

40. Julien Rossi, « Écrire le code du code », *RESET. Recherches en sciences sociales sur Internet*, nº 11 (mars 2022).↩

41. Miriam Eric Suzanne, « CSS Is Rad », *Miriam Eric Suzanne* (https://www.miriamsuzanne.com/speaking/css-rad/, 2020.↩

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

43. De nombreux manifestes mettent en avant l'importance que la structuration HTML a dans les pratiques du web: «Web design as architecture» (http://www–arc.com/), « HTML energy » (https://html.energy/index.html).↩

44. 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/↩

45. 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. »↩

46. Voir les réflexions de Julien Bidoret à ce propos: «Flossflop», publié le 15 qvril 2024 sur son carnet en ligne, URL: https://accentgrave.net/log/2023-04-15–flossflop/↩

47. 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 12 (Amsterdam : Institute of Network Cultures, 2019), pp. 238-239.↩

48. 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.↩

49. Nicolas Oliveri, « Logiciel libre et open source : une culture du don technologique », *Quaderni. Communication, technologies, pouvoir*, nº 76 (septembre 2011) : 111-19.↩

50. Groten, « Hacking & Designing », *op. cit.*↩

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