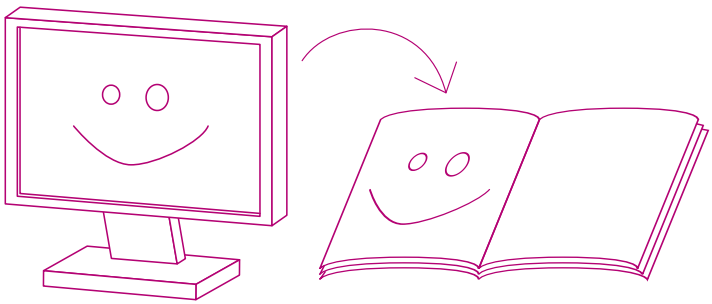
# *Buffet à volonté*
# Web2Print

Ce carnet regroupe toutes les expérimentations et apprentissages que j'ai pu faire avec le concept de web2print. Il relève d'une expérience personnelle qui ne prend en compte que certaines fonctionnalités et n'est pas un guide exhaustif du html2print. Personnellement, je n'ai pas beaucoup d'expérience en la matière, j'ai donc proposé à des personnes un peu plus calées de m'aider à faire ce cahier.





# Sommaire

# Avant de commencer

## Qu'est-ce que c'est ?

Le but du web2print ou web to print est avant tout d'avoir un fichier commun entre ce qui est créé sur une page web et ce qui est imprimé/produit en pdf. Tel que je vais en parler ici, et de manière extrêmement simplifiée c'est une version améliorée de l'action **Fichier ↠ Imprimer** sur un fichier web. Cela permet de créer des fichiers pdfs qui ressemblent à ce qu'on peut produire via des logiciels de mise en page, avec des pages gauches et droites, des marges, des numéros de page, des en-têtes etc.... On peut ensuite les imprimer, en faire des livres, des cahiers ou des affiches, des cartes de visite, n'importe quel support imprimable dans n'importe quel format.

Cécile, la pro du web to print du Master Typographie de la Cambre, le définit comme ceci :

> « Le web to print, c'est simplement concevoir de l'éditorial avec les outils du web. Le code va produire des publications hybrides dont les supports ou les processus de conception impliquent un mélange de technologies physiques et numériques. Le même contenu a deux formes différentes mais complémentaires, produites par le même outil : un navigateur web, connecté à internet d'un côté, et capable de générer un pdf de l'autre. Dans la conception,

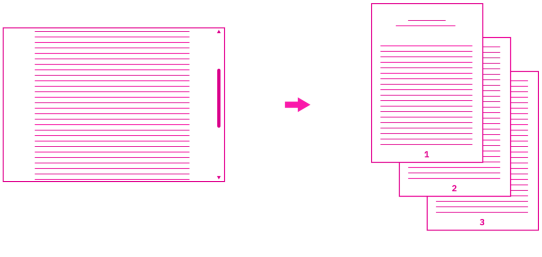
cela se traduit par la séparation sémantique propre au web : il y a d'une part le contenu textuel et iconographique, le squelette ou html et d'autre part la présentation de cette page, son enveloppe corporelle ou css. Ces deux langages informatiques seuls permettent d'offrir une visualisation des objets éditoriaux. »

Comme OSP (Open Source Publishing) le dit très bien :

*« La raison la plus excitante d'utiliser html/css est le fait qu'on peut naviguer entre le code et la manipulation visuelle grâce à l'inspecteur d'éléments des navigateurs. [...] Ce va-et-vient entre les manipulations manuelles et le code est nouveau pour la production imprimée. La deuxième raison pour laquelle nous avons mis en place ce système est que la conception est faite avec du code/texte, ce qui signifie que nous pouvons utiliser des éditeurs de texte collaboratifs tels que Etherpad pour concevoir avec plusieurs personnes en même temps. »*

En résumé, le web to print, c'est la possibilité de faire le lien entre une page web et une page imprimée, l'ordinateur et le livre au travers d'un même fichier source. On appelle cela le single source publishing.






pagedjs.org « How Paged.js works »

# Débuter

Pour faire du html to print, il faut quand même posséder quelques notions de code, et comprendre comment ça marche, ce qui agit, sur quoi...
La plupart des ressources disponibles restent très complètes et permettent d'arriver à un bon résultat en n'entrant pas dans de la programmation javascript. En général, être familiɛrɛ avec les balises html et css est suffisant.

Personnellement, je ne touche qu'aux codes html et css car je n'y connais pas grand chose en javascript (qui permet d'obtenir du contenu interactif) ou en php (principalement utilisé pour produire des pages Web dynamiques via un serveur web). Ça peut suffir si ce que l'on cherche à faire n'est pas trop complexe et qu'on va chercher les fonctionnalités qui nous manquent sur internet ou auprès de personnes plus calées, et qu'on sait s'adapter lorsque ce qu'on voulait faire ne fonctionne pas.

Pour commencer à apprendre à coder en html et en css, il existe des cours en ligne de type Open Classroom **openclassrooms.com**, W3schools **w3schools.com**

Quand on commence à maîtriser le sujet, on peut rapidement créer des publications assez complexes en web to print.

# Quelques possibilités

Le web to print n'est pas un logiciel. C'est une manière de faire des éditions. Il existe donc de nombreuses façons d'expérimenter le web to print.
En voici quelques unes.

## PagedJs

> *PagedJS est une bibliothèque JavaScript gratuite et open source qui pagine le contenu dans le navigateur pour créer une sortie pdf à partir de n'importe quel contenu html. Cela signifie que vous pouvez concevoir des ouvrages destinés à l'impression (par exemple des livres) à l'aide de html et de css.*

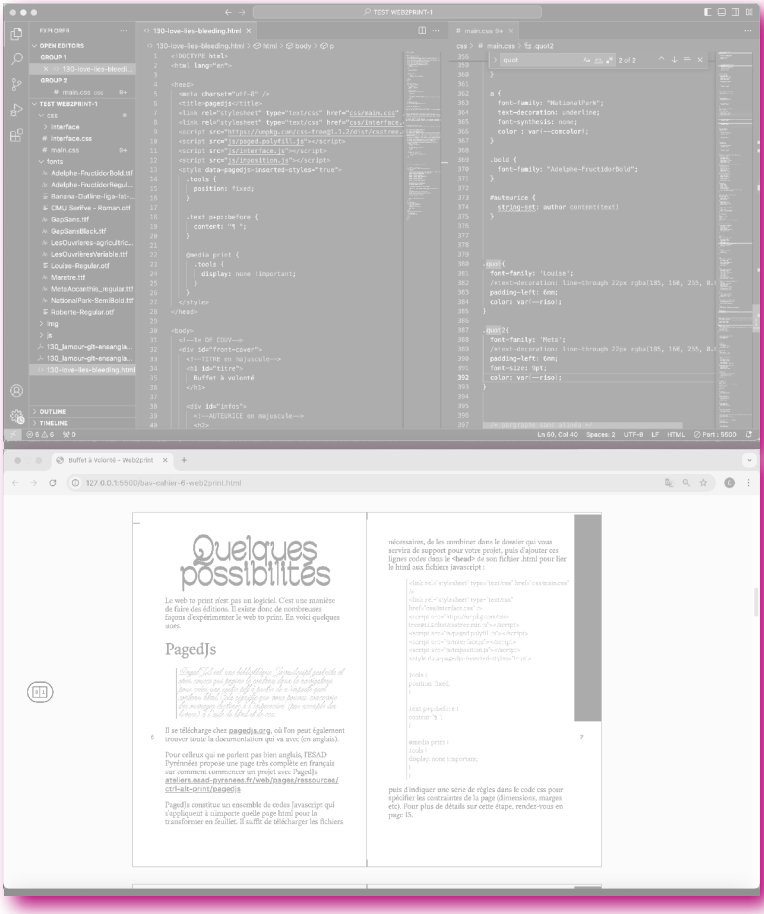
Il se télécharge chez **pagedjs.org**, où l'on peut également trouver toute la documentation qui va avec (en anglais).



Pour celleux qui ne parlent pas bien anglais, l'ESAD Pyrénées propose une page très complète en français sur comment commencer un projet avec PagedJs **ateliers.esad-pyrenees.fr/web/pages/ressources/ctrl-alt-print/pagedjs**

PagedJs constitue un ensemble de codes Javascript et css qui s'appliquent à n'importe quelle page html pour la transformer en page imprimable, notamment en feuillet.

PagedJs permet même l'imposition des pages pour une impression en cahier.




Les contenus html et css dans l'éditeur de code Visual Studio Code, puis le rendu Go Live dans Chrome

# Html to print

Le collectif bruxellois Open Source Publishing, centré sur la production de contenus graphiques imprimés et numériques en open source, propose un outil de html2print pour débuter avec le principe : **https://osp.kitchen/tools/html2print/**
Le fichier permet de régler le contenu dans une page, de décider des dimensions, marges de la page etc, depuis une page web. La visualisation ne se fait pas par la fonctionnalité Go Live de VS code mais en effectuant la commande **cmd + P** sur la page web. En parallèle, iɛls ont également créé un navigateur, OSPkit, destiné à être utilisé avec leur outil html2print.

> L'objectif est de mettre en page des documents imprimés dans un navigateur web. Nous avons construit notre propre navigateur webkit afin d'avoir un navigateur plus rapide et de bons réglages typographiques (bizarrement, les espacements et les crénages peuvent être bizarres dans certains navigateurs webkit).

ClaraB: 08/05/2024 15:43:58

# ChattyPub

ClaraB: 09/05/2024 09:06:38

Chatty pub est un outil développé par le collectif néerlandais Hackers&Designers hackersanddesigners.nl. C'est un outil en ligne qui fonctionne avec une messagerie instantanée, Zulip, et un outil de conversion web. Les messages envoyés dans le canal de discussion constituent le contenu texte (équivalent du contenu html).
On applique ensuite un style (css) aux emojis de réaction aux messages. Ainsi, si on applique une caractéristique **bold, 11pt, rouge** à l'emoji fraise, tous les messages auxquels on aura réagi par l'émoji fraise auront les caractéristiques bold, 11pt et rouge.
On peut appliquer plusieurs emojis à un même message, du moment que les styles qu'ils appliquent ne s'annulent pas entre eux.
Chatty pub fonctionne à peu près de la même manière que paged.js (sauf quelques fonctionnalités qui posent encore problème) et il suffit d'ajouter une « rule » **@page** avec les caractéristiques de la page pour obtenir n'importe quel format de page.

ClaraB: 09/05/2024 09:07:01

Pour tester, télécharger le fichier git github.com/hackersanddesigners/chatty-pub et suivre les instructions qui figurent dans le fichier **»→ docs »→ ChattyPub.md**.

ClaraB: 02/06/2024 10:18:59

Une mise en page par Chatty Pub se fait en ligne : le fichier à télécharger comporte juste les instructions pour débuter un projet et la documentation sur Chatty Pub si quelqu'unɛ souhaite s'en inspirer.

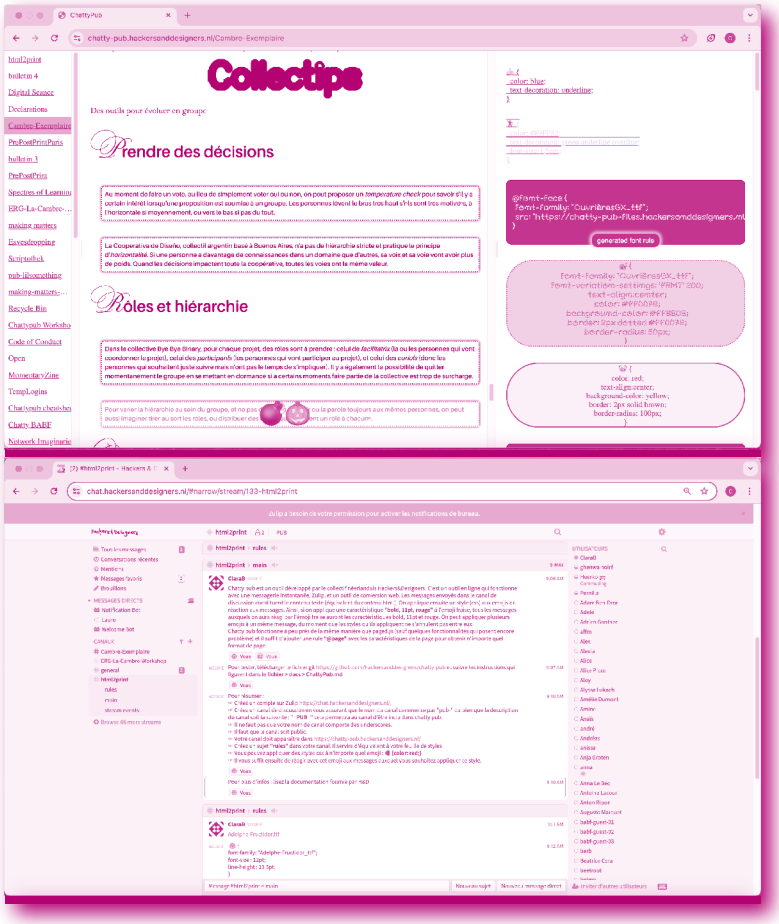
ClaraB: 02/06/2024 10:19:20

Pour résumer :
☞ Créez un compte sur Zulip chat.hackersanddesigners.nl
☞ Créez un canal de discussion en vous assurant que le nom du canal commence par **« pub-»** ou bien que la description du canal soit la suivante : **«_PUB_»** cela permettra au canal d'être inclu dans chatty pub.
☞ Il ne faut pas que votre nom de canal comporte des underscores « _ ».
☞ Il faut que le canal soit public.
☞ Votre canal doit apparaître dans chatty-pub.hackersanddesigners.nl
☞ Créez un sujet **« rules »** dans votre canal. Il servira d'équivalent à votre feuille de styles
☞ Vous pouvez appliquer des styles css à n'importe quel emoji : 🍓 **{color: red;}**
☞ Il vous suffit ensuite de réagir avec cet emoji aux messages auxquel vous souhaitez appliquer ce style.



ClaraB: 02/06/2024 10:19:47

Pour plus d'informations, lire la documentation fournie par H&D sur le fichier Git : github.com/hackersanddesigners/chatty-pub




L'interface Chatty pub et la messagerie Zulip

# Pad2print

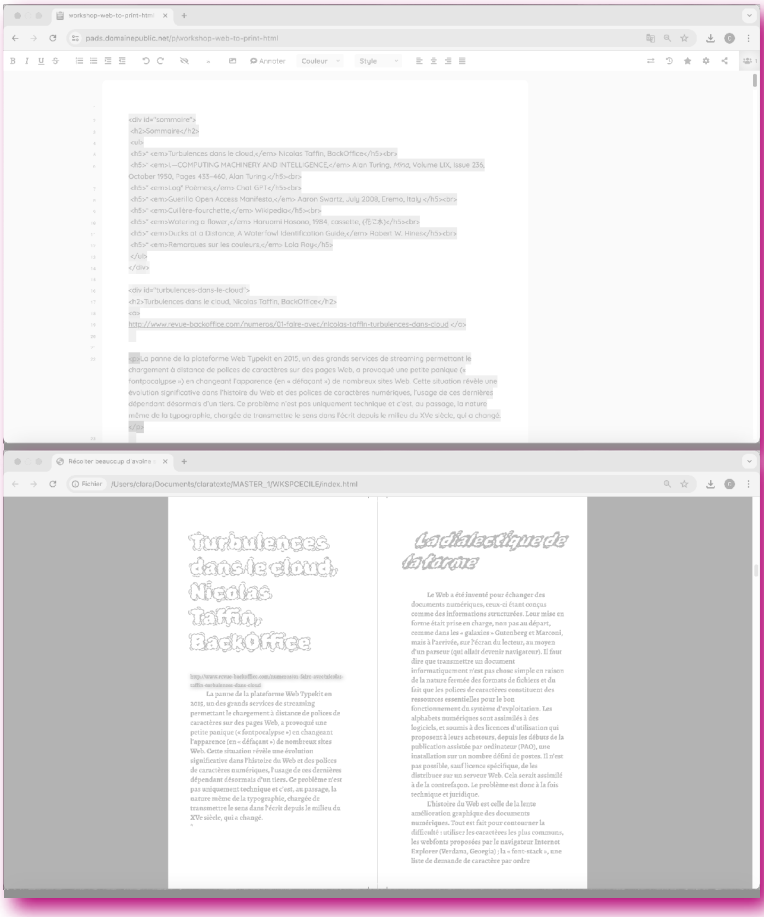
Un pad, c'est un système de bloc note collaboratif en ligne, une alternative open source et libre à Google Docs.

On écrit sur les pads comme on écrirait sur deux documents html / css dans un éditeur de code, sauf qu'ici, ils sont en ligne et collaboratifs, et peuvent donc être changés en temps réel par tous·tes celleux qui ont accès à ces pads.

Le Pad2print est une variante du html2print, développé par le collectif Luuse (Bruxelles) : **gitlab.com/Luuse/pad2print**

OSP ont elleux aussi développé un outil similaire, appelé Ether2Html, **osp.kitchen/tools/ether2html**. Il fonctionne avec les Etherpad (un type de pad) desquels il est possible d'extraire un fichier texte.

Dans le cas du Ether2html, on crée 2 pads dinstincts, un pour le contenu (html) et un pour les feuilles de style, qui définissent les caractéristiques visuelles (css). On écrit dans ces pads comme dans un fichier de code. On crée ensuite un autre document html, dans lequel on écrit un code qui indique qu'il faut aller extraire le contenu d'un des pads pour en faire le contenu html et lui appliquer le contenu de l'autre pad en tant que feuille de style css.




Le contenu html du pad puis le résultat à imprimer

Néanmoins, dans le système pad2print, l'hébergement reste local. Cela signifie que le fichier qui transforme le pad en html / css puis en objet imprimable doit être sur l'ordinateur de toustes celleux qui veulent visualiser le résultat.

**Voici le tutoriel proposé par OSP sur comment installer le fichier. Attention, le pad2print ne fonctionne que sur Chrome ou Chromium.**

How to use it

☞ Create an etherpad somewhere for the CSS (e.g. https://framapad.org).

☞ Create an etherpad somewhere for the content (e.g. https://framapad.org).

☞ Click on the link ether2html.html then download the file by right-click "Download raw" and choose "Save link as".

☞ Edit that ether2html.html file by replacing the URL under the comment < !-- CHANGE THE URL OF YOUR CSS PAD BELOW --> by the export URL of the pad CSS you created in step 1, copy the link location of the plain text export of the pad - see below the screen capture.

☞ Edit that ether2html.html file by replacing the URL under the comment < !-- CHANGE THE URL OF YOUR MARKDOWN CONTENT PAD BELOW --> by the export URL of the pad for the content you created in step 2, copy the link location of the plain text export of the pad - see below the screen capture.



☞ Open the file ether2html.html in your web browser (Firefox or Chrome).

☞ Edit your pad with content (markdown or html) and your pad with CSS styles.

☞ Reload the file ether2html.html opened in your web browser.

☞ Use the print function of your browser and choose "Save as file". Here is your pdf ready to print!

# Journal de Bord

C'est un peu compliqué de faire un résumé complet sur comment on utilise le html2print dans un cahier comme celui-ci. Il faudrait déjà faire un cours complet d'HTML et de CSS. Voici donc, en attendant, toutes les expérimentations que j'ai pu faire pendant la création de ce cahier en PagedJs, sous forme de journal de bord.

## Créer un document

Il n'est pas possible d'ouvrir le code PagedJs directement dans le navigateur web. L'affichage PagedJs fonctionne avec un serveur web, c'est à dire un éditeur de code ayant une fonctionnalité Go live (le plus connu étant VS Code). Pour les utilisateurices de OSX ou bien Linux, il faut indiquer ces lignes de code dans le terminal :

```
cd votre/dossier/de/travail
php -S localhost:8888
# ou bien
python -m SimpleHTTPServer 8888
# ou bien
python3 -m http.server 8888
# ces 3 commandes permettent d'accéder à la page web sur http://localhost:8888
```

(Esad Pyrénées)

Pour opérer PagedJs, il faut utiliser un navigateur parmis ceux-ci : Ungoogled Chromium, Chromium (Open source), Edge ou Chrome (propriétaires). Les autres ne supportent pas bien l'outil et n'affichent pas le résultat exact de ce qui est codé.

Pour faire du html2print, il faut 3 fichiers : un fichier HTML, soit le contenu, balisé selon le langage web, un fichier CSS, c'est à dire une feuille de style, qui permet de mettre en forme le contenu, appliquer des couleurs, des styles de paragraphe etc... Il faut aussi un fichier javascript, qui ici permet de transmuter la page web en pages imprimables (avec les dimensions en mm et en points au lieu de px ou vw, avec des format A4, A3 au lieu de formats pixels, des marges, des pages gauches et droites etc). Ce dernier fichier, c'est un combo de 3 scripts écrits en langage javascript qui le permettent.

Ils se téléchargent ici :
https://unpkg.com/browse/pagedjs/dist/

Il suffit de télécharger les fichiers nécessaires, de les déplacer dans le dossier qui vous servira de support pour votre projet, puis d'ajouter ces lignes codes dans le **<head>** de son fichier .html pour lier le html aux fichiers javascript et css :



```
<link rel="stylesheet" type="text/css" href="css/main.css"
/>
<link rel="stylesheet" type="text/css"
href="css/interface.css" />
<script src="https://unpkg.com/css-
tree@1.1.2/dist/csstree.min.js"></script>
<script src="js/paged.polyfill.js"></script>
<script src="js/interface.js"></script>
```

```
<script src="js/imposition.js"></script>
<style data-pagedjs-inserted-styles="true">

.tools {
position: fixed;
}

.text p+p::before {
content: "¶ ";
}

@media print {
.tools {
display: none !important;
}
}
```

# Paramètrer l'impression

Après avoir paramétré le fichier html, on paramètre le format de page dans le CSS, avec la règle **@media print {@page{vos règles ici}}**. Il faut d'abord indiquer la disposition des pages (ici, en cahier, donc **booklet**)



```
@media print
{
body
{
--paged-layout: booklet;
}

/* toutes les pages */
@page
{
```

```
size: 125mm 185mm; /*largeur / hauteur*/
marks: crop;
margin-top :8mm;
margin-bottom : 15mm;
margin-left : 12mm;
margin-right:12mm;
}
}
```

Pour la taille de la page, les formats DIN fonctionnent également : on peut indiquer **A5**, **A3**, **letter**, **ledger** etc, et préciser **landscape** ou **portrait**. Cela donnerait :

```
@page {
size: A4 landscape
}
```

Pour ajouter des numéros de page, on peut suivre le tutoriel proposé par l'ESAD Pyrénées :

> *Ajoutez des numéros de page et des titres courants. Les margin boxes créées par Paged.js peuvent accueillir du contenu: généré automatiquement, tels des numéros de page, ou présents dans le HTML (section courante, titre du document, etc.).*



```
@page : left{
@bottom-left {
content: counter(page);}
}
@page :right{
@bottom-right {
content: counter(page);}
}
/* pas de numéro de page pour la couverture */
```

```
@page macouverture {
@bottom-right { content: none; }
@bottom-left { content: none; }
}
```

Pour des spécification sur les pages gauches et droites dans le cadre d'un livret, indiquer (toujours dans **@media print**) les règles :

```
@page:left {
}
@page:right{
}
```

Pour tourner du contenu dans les marges et mettre le texte ou le numéro de page à la verticale :

```
@page {
@left-top {
transform: rotate(-90deg);
transform-origin: top left;
position: relative;
}
}
```



Pour les fonds perdus et les traits de coupe :

```
@page {
bleed : 5mm;
marks : crop;
}
```

# Quelques exemples CSS

## Modifier une image

Filtre de couleur sur une image :
Créer une div et lui attribuer une class, ici **.filter-pink**
Dans cette div, insérer l'image et attribuer à l'image une autre class, ici **.img-filter**

```
.filter-pink{
background-color: var(--riso);
background-repeat: no-repeat;
}

.img-filter{
mix-blend-mode : screen;
z-index: 0;
filter: grayscale(100%);
}
```



(Solution proposée par Anna Le Bec)

Pour afficher l'ombre d'un objet ou d'une div :

```
box-shadow: 5px 5px 10px red /*(couleur de votre
choix)*/;
```

# Couleurs

Faire un nuancier de couleurs interchangeables :

Comme je ne suis pas sûre de mes couleurs, mais que je sais où chacune va aller, au lieu de donner une valeur colorimétrique absolue à mes objets, je leur donne une valeur variable. J'associe un nom à la couleur, puis je donne comme couleur à mes div ce nom en particulier. Comme ça, si je change la valeur de la couleur dans le "root", elle se change dans toutes les div concernées par ce nom.

```
:root {
--comcolor: #ff00bb;
--riso:#003d1b;
}
```

puis quand je veux attribuer une couleur à un objet, je marque dans ma balise css :

```
p {
color : var(--comcolor);
}
```

(Un tutoriel de **w3schools)**

# Typographie

Pour ajouter une fonte depuis notre ordinateur, on commence par placer le fichier .otf ou .ttf dans un dossier "fonts" dans le dossier de codes qu'on a créé. Puis on code dans le css :

```
@font-face {
font-family: "Nom-de-la-fonte";
src: url("/fonts/Nom-de-la-fonte.ttf"); }
```

Dans la balise à laquelle on veut appliquer la fonte (par exemple tout les <h3>), on indique

```
h3 {font-family : "Nom-de-la-fonte";}
```

Pour créer une lettrine (n'appliquer une propriété qu'à la première lettre de chaque paragraphe)

```
p ::first-letter{
font-family : "Maratre_otf";
font-size : 70px;
}
```

Il existe également un script qui permet de faire des corrections microtypo en français en anglais ou en anglais canadien, JoliTypo :
**github.com/jolicode/JoliTypo**

# Imprimer



Pour imprimer, il faut faire simplement **cmd + 'P'**. ou bien **Fichier ⇢ Imprimer**. Vous pouvez alors enregistrer le pdf pour ensuite l'imprimer
Sur Chrome, il faut cocher la case **Graphiques d'arrière plan** pour que les objets avec une propriété **background-color** s'affichent à l'export et s'impriment.

Si vous avez déterminé le fichier en tant que **booklet**, appuyer sur le petit bouton  pour préparer les fichiers à s'imprimer en cahier.

# Ressources

☞ Le site de l'ESAD Pyrénées propose un grand nombre de ressources html2print et open source en général, des bibliothèques de code, des listes de ressources : **ateliers.esad-pyrenees.fr/web**

☞ PrePostPrint est une plateforme qui regroupe de nombreuses ressources en html2print, et plus globalement de la documentation sur les outils d'impression web open source, experimentaux et/ou gratuits : **prepostprint.org**

☞ Iɛls possèdent également une page wiki. **wiki.prepostprint.org**

☞ W3schools est une site web de tutoriels pour apprendre à coder dans de nombreux langages web. **www.w3schools.com**

☞ Codepen est une plateforme de code en ligne, où les developpeureuses postent leurs tests de code html / ccs / js, qu'il est possible de copier et d'adapter en live. **codepen.io**

Voilà, vous savez tout ce que je sais sur le HTML2print, ou plutôt tout ce que l'on m'a appris dessus.
Les contributions textuelles des personnes extérieures sont indiquées avec une fonte différente. Dans la mesure du possible, je cite toutes les sources de mes trouvailles. Pour le reste, soyez assurée que je n'ai rien appris seule.

Ce cahier fait partie
de la collection *Buffet à Volonté*
(petit manuel de transition au
libre pour celleux qui se posent
beaucoup de questions)
aux côtés de
*Buffet à Volonté* Ouvrir l'appétit
*Buffet à Volonté* Licences
*Buffet à Volonté* Ustensiles
*Buffet à Volonté* Scribus
*Buffet à Volonté* Gimp & Inkscape
*Buffet à Volonté* Web2print
*Buffet à Volonté* Libre Office

Il a été imprimé en Comcolor au PrintLab de l'ENSAV La Cambre sur Muken Print White 18 90g en mai 2024

Graphisme : Clara Bougon, mise en page avec PagedJs et images modifiées sur Inkscape et Gimp



Fontes
Adelphe Fructidor
par Eugénie Bidaut
*Louize* par Luna Delabre
et Camille Depalle
Banana Fatt par Clara Bougon
**National Park** par DO STUDIO
Meta Accanthis
par Amélie Dumont
*Mésanie* par Claude Pelletier

Merci à Domitille Debret
et Quentin Creuzet de F451,
Vinciane Dahéron d'OSP,
et à Cécile Faure pour leur temps
et le partage de leur vision du libre.

Antoine Gelgon, Enz@ le Garrec,
David Le Simple pour leurs super
retours

Anaïs, Enora, Fredo et Matalie pour
leurs lectures, relectures, soutien et
conseils

Femke Snelting et Eva Weinmayr
pour m'avoir permis de participer à la
réécriture de la CC4R et avoir initié
mon intérêt pour les licences

Ajouts textuels et conseils :
Cecile Faure, Anna Le Bec, Open
Source Publishing, w3school.com,
Delyo Dobrev, ESAD Pyrénées,
Wikipedia, Hackers&Designers

Merci à Anna Barres pour la
comcolor, Pierre Huyghebaert,
Laure Giletti, les typotes et les
copaines pour une super scolarité.