

# Mettre le web en page(s)

Lucie Anglade — Guillaume Ayoub



# Lucie Anglade

- Ingénieure en informatique
- Développeuse de logiciels libres
- Présidente de l'AFPy, organisatrice des Meetups Python à Lyon

# Guillaume Ayoub

- Ingénieur en informatique
- Développeur de logiciels libres
- Ancien membre du W3C et lecteur assidu de spécifications

# Que va-t-il se passer pendant cet atelier ?

- Pourquoi ne pas utiliser LaTeX, Word, InDesign... ?
- Non, mais du HTML et du CSS pour faire des PDF, c'est pas prévu pour ça, voyons, c'est pour faire du web !
- Et si on mettait en page un livre ?

# Il était une fois...

Une petite entreprise lyonnaise qui générait divers documents : des factures, des rapports, des livrets de formation...

Pour générer ces documents, iels ont utilisé différentes solutions : LibreOffice, LaTeX...

# Mais...

Même si ces logiciels sont très bien, ils ne sont pas adaptés à la génération automatique de documents.

Plein de petites choses peuvent changer entre deux documents globalement identiques, et ce n'est pas très pratique à gérer avec ces outils.

Et puis, n'essayez pas de demander à un·e graphiste de faire du design sur ces outils. En général, iels refuseront poliment.

## C'est alors que...

L'idée d'utiliser HTML et CSS pour créer des PDF émergea.

Une preuve de concept plus tard, voilà un stagiaire lancé sur le projet.

# C'est pas un peu bizarre quand même ?

On connaît bien HTML et CSS pour faire du web, mais en réalité, c'est prévu.

Dès la première version de la spécification CSS en 1996, on trouve tout un chapitre dédié aux documents paginés.



# Bon, c'est quoi WeasyPrint ?

WeasyPrint est une bibliothèque en Python qui permet de transformer du HTML et CSS en PDF.

Et c'est nous qui la développons ! 💜

# Ça s'installe comment ?

WeasyPrint est packagé dans de nombreuses distributions Linux.

N'hésitez pas à jeter un œil à [la documentation](#) si jamais votre distribution favorite ne dispose pas (encore) de WeasyPrint.

# Et si on imprimait un truc ?

Vous pouvez récupérer le dépôt des exemples sur [GitHub](#).

```
weasyprint book.html book.pdf
```

C'est pas très beau, il n'y a que le style par défaut de WeasyPrint. Pas de panique, on va changer ça !

# Du CSS pour des documents ou pour le web, c'est (presque) pareil !

On va pouvoir utiliser les mêmes sélecteurs et les mêmes propriétés.

Note : tout CSS n'est pas (encore) supporté dans WeasyPrint. Vous pouvez oublier `grid` pour l'instant.

# Et si on cachait les images ?

Dans un livre de poche, il n'y a généralement pas d'images. On va pouvoir les cacher en utilisant `display: none`.

Vous pouvez créer une feuille de style qui va servir à définir la mise en page de votre livre.

Pour que WeasyPrint prenne en compte cette feuille de style, vous pouvez ajouter un `link` dans le HTML ou utiliser l'option `-s` en ligne de commande.

```
weasyprint book.html -s style.css book.pdf
```

# Mais avec des trucs en plus !

Certaines fonctionnalités sont spécifiques aux documents paginés, comme par exemple... les numéros de page.

Et plein d'autres :

- les coupures de page,
- les en-têtes et pieds de page,
- les marges de page,
- la table des matières...

# À vous de jouer !

Mettez en page ce livre au format poche.

N'hésitez pas à faire un petit dessin pour savoir où vous allez ou à vous inspirer du résultat disponible dans le dépôt.



# Le livre serait bien mieux avec des numéros de page

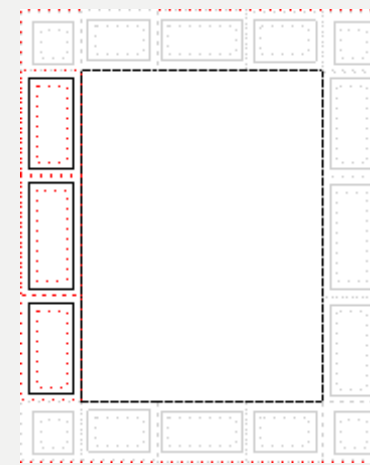
Pour rajouter les numéros de page (autrement appelés « folios »), on va se servir des règles `@page`.

Elles vont nous permettre déterminer les propriétés des pages, comme les marges et le contenu de ces marges (c'est là que l'on va mettre les numéros).

Nous voulons aussi des compteurs. Les compteurs pour les pages existent de base. Ils s'appellent `page` (pour la page en cours) et `pages` (pour le nombre total).



# Les marges de page



```
@page {  
  @bottom-center {  
    content: counter(page) " / " counter(pages);  
  }  
}
```

# Le livre serait bien mieux si l'on passait à la page suivante à chaque chapitre

À certains moments, on voudrait bien que le contenu passe à la page d'après, et à d'autres moments que l'on évite de couper. Pour cela, on utilise les propriétés `break-before`, `break-inside` et `break-after`.

La valeur `avoid` permet d'empêcher la coupure, `page` permet de la forcer. On peut même choisir si l'on coupe quand on est sur une page de droite ou de gauche !

Chaque chapitre est dans une section, on peut donc dire que l'on veut passer à la page suivante avant chacune d'elles.

# Le livre serait bien mieux avec le titre du chapitre courant en haut de la page

Il arrive souvent que l'on veuille avoir un morceau du contenu qui se répète sur chaque page. Dans le cas d'un livre, on aimerait avoir le titre du chapitre en cours.

Pour cela, on va utiliser les chaînes de caractères nommées qui permettent de définir une sorte de variable et son contenu `string-set: titre content();`.

La valeur se récupère en utilisant `string(titre)`.

Et on utilise aussi le placement dans les marges pour mettre le titre où l'on veut !

# Le livre serait bien mieux avec une table des matières

Vous avez peut-être remarqué cette liste plutôt vide au début du HTML. C'est grâce à elle que l'on va pouvoir faire la table des matières.

Chaque élément de la liste comporte un lien vers un chapitre. Grâce à ce lien, on peut récupérer le titre du chapitre et sa page.

Pour le titre, on utilise `target-text(attr(href))`. Pour le numéro de la page, `target-counter(attr(href), page)`.

Oh, et les petits points que l'on trouve dans les tables des matières ? Il existe une fonction en CSS pour faire cela : `leader()`.

Alors, il ressemble à quoi votre livre ?

# N'hésitez pas à suivre CourtBouillon !

- Le site de [CourtBouillon](#) et [WeasyPrint](#)
- [Twitter](#)
- [Mastodon](#)
- [LinkedIn](#)