# Développement web

Des guides sur le développement web (design, HTML, CSS, PHP et SQL) ainsi que le projet.

- Comment faire le design d'un site
- Ma procédure CSS pour le projet
- Petit guide PHP pour le projet web

# Comment faire le design d'un site

> Ce qui suit est un résumé hyper condensé du livre Refactoring UI

**Avant toute chose**, je trouve important de noter qu'il est important de ne pas se perdre dans le design de son site, ce n'est vraiment pas la partie la plus importante pour le projet. Personnellement je trouve que le développement web est particulièrement chronophage et épuisant et c'est pour cela que je vous conseille vivement de faire le design avec l'aide de quelqu'un qui peut vous donner des idées, des feedbacks, vous soutenir, etc.

# La base (fonctionalités, fonts, personalité, layout global, etc)

Pour commencer on peut d'abord lister les fonctionalités de la page que l'on veut designer par ordre d'importance.

Ensuite on peut prendre une feuille et un gros marqueur et faire le layout global de la page en mettant en avant les fonctionalités importantes. Les couleurs viendront bien après.

On peut ensuite considérer la personalité générale du site, ce qui mène déjà à certaines décisions au niveau du design :

- Les couleurs (i.e bleu est plus tot neutre, rose est plus tot casuel et or est plus tot sophistiqué)
- La rondeur (border-radius) des éléments (en général plus = plus casuel et moins = plus formel)
- Les polices d'écritures (serif = pro, rond et sans serif = casuel, monoespacé = techy, sans = neutre)
- Le ton de language utilisé dans les messages

# Les couleurs

Ensuite on peut utiliser des sites comme Coolors pour avoir des idées de couleurs, mais il ne faut pas trop compter sur la palette générée car beaucoup plus de couleurs sont nécessaires :

- Une échelle de gris (mais ça peut ne pas être du gris et être du brun très foncé par exemple) pour le texte, formulaires, lignes, certaines bordures, etc
- Une échelle d'une autre couleur comme couleur principale qui va vraiment donner la personalité du site
- Plusieurs autres échelles comme couleurs d'accents de certaines choses (exemple: jaune pour un avertissement, rouge pour un danger, vert pour quelque chose de positif, etc)

# Mettre en place une hierarchie des éléments

Il ne faut pas confondre la hierarchie du code (body, main, h1, etc) et la hierarchie des éléments. Ce n'est pas par ce qu'un élément est un `<h1>` qu'il doit forcément être plus grand par exemple.

On peut donc reprendre la liste faite précédemment et définir quels éléments de la page doivent être le plus mis en avant. Pour mettre en avant un élément on peut:

- Mettre une couleur de fond, un gradient, ou autre
- Augmenter la taille (mais attention car cela peut vite devenir excessif)
- Augmenter le contraste de l'élément ou bien le "poid" de la police (font weight)
- Rendre les autres éléments autour moins important (réduire le contraste, ou le poid de la police par exemple)

Une autre chose importante est d'éviter les labels car la hierarchie des éléments peut rendre la chose à la fois plus claire, plus intuitive et plus belle. On peut donc remplacer un label tel que "nom: John Doe, Occupation: freelance" par

- Utiliser des tailles et des importances différentes (par exemple remplacer "name: John Doe, Occupation: Freelance" par "John Doe" en titre et "Freelance" en sous titre)
- Une phrase (par exemple remplacer "En stock: 12" par "12 en stock")
- Quand le label ne peut pas être remplacé, on peut alors essayer de le rendre plus petit et mettre la valeur plus en avant.
- Sinon on peut toujours utiliser un label si c'est vraiment indispensable (par exemple un tableau de spécifications d'un produit)

Une autre chose est que la sémantique des éléments (tel qu'associer une couleur à une action) vient après la hierarchie des éléments donc à la place de mettre un bouton "Supprimer" en rouge fluo, il est préférable de garder la couleur principale et d'après envoyer une popup de confirmation avec un bouton rouge pour confirmer l'action.

# Gestion de l'espace sur la page

- Designer et coder en "mobile first" pour s'assurer de d'abord faire fonctionner les choses sur mobile
- Ajouter beaucoup de padding et de margin puis de suprimer l'espace en trop par après
- Le contenu n'a pas besoin de prendre toute la place, on peut donc diminuer la largeur de la page et la centrer
- On peut aussi diviser la page en colonnes pour mieux rentabiliser l'espace
- Les tailles relatives ne fonctionnes pas toujours donc il faut faire attention et tester un peu plus
- S'assurer que les éléments qui vont ensemble ont moins d'espace les séparant que les autres

# Améliorer le texte sur la page

- Utiliser des vraimnet bonne polices (voir au début de cette page dans la section sur les bases)
- Eviter de faire des lignes trop longues (pas plus de 75 caractères de préférence)
- Pour rendre le texte moins "dense" on peut changer le "line height" et le "letter spacing"
- Ne pas centrer des textes trop long car cela nuit à la lisibilité
- Eviter de mettre des couleurs sur tous les liens. On peut simplement mettre un peu plus de poid sur le texte et changer la couleur quand on passe la souris dessus par exemple
- Utiliser des tirets pour diviser les lignes des textes "jusifiés" pour éviter d'avoir des gros trous dégeulasses dedans

# Ajouter plus de profondeur à la page

- On peut utiliser des gradient sur le fond et des ombres pour simuler l'action de la lumière et élever ou dessendre des éléments
- On peut superposer des éléments pour ajouter une dimension supplémentaires (par exemple superposer un champ de texte sur une première et une deuxième section qui ont

tous deux des couleurs différentes)

# Gérer les images

- Avoir des images de la meilleur qualité possible
- Faire attention à ce que les images envoyées par les utilisateurs aient toute la même taille
- Mettre une bordure ou une ombre derrières les images envoyées par des utilisateurs pour éviter qu'elles se fondent avec la couleur de fond de la page
- Changer la luminosité d'une image quand du texte est écrit dessus

# Les fonds de page

- Utiliser des fonds différents pour différentes sections
- Utiliser des fonds différents pour mettre des élémnets en avant plus tot que d'autres

Plusieurs solutions existe pour changer le fond

- Utiliser une couleur unie
- Utiliser une couleur unie avec du grain
- Utiliser une image de fond
- Utiliser un gradient
- Utiliser un pattern qui se répète en fond
- Utiliser quelques formes en fond

# Quelques touches en plus

- Eviter de mettre trop de bordures pour séparer les éléments et préférer ajouter de l'espace et changer la couleur quand on passe la souris dessus
- Ajouter une bordure sur certains éléments pour ajouter de la couleur (accent border)
- Les tableaux et les menus n'ont pas besoin d'être chiant. On peut donc ajouter plus de couleurs, de layouts, et d'autres éléments plus "riches" à l'intérieur
- Un état vide ne doit pas vraiment être vide

# Ma procédure CSS pour le projet

Pour la première eval du projet j'étais grave à la bourre j'ai vraiment commencé mon projet 5 jours avant la deadline et j'ai changé de design en cours de route et tout recommencé de zéro plusieurs fois.

Mais la méthode qui marchait le mieux pour être organisé (en tout cas pour moi) c'était d'avoir une procédure à suivre rigoureusement pour chaque élément du site. Et donc faire les choses au fur et à mesure pour ne pas être submergé. Si je ne faisais pas ça, rien n'étais organisé et tout s'éffondrait à la première modification venue, donc pas top.

Avec cette procédure, ça m'a permit de faire un site en utilisant flexbox et qui est aussi totalement responsive (à quelques détails près).

Je n'ai pas utilisé SASS pour ce projet mais à bien y réfléchir ça n'aurait peut'être pas été une mauvaise idée car la syntaxe est bien plus simple.

# Préparer le CSS

1. Ajouter un nouveau fichier `style.css` dans un dossier `css` et y ajouter le contenu suivant pour faire en sorte de se simplifier la vie par après

```
* {
    box-sizing: border-box;
    margin: 0px;
    padding: 0px;
    font-family: "nom de la police"; /* Ajout de(s) la police(s) par défault */
}
```

2. Ajouter la ligne suivante dans tous les `<head>` de toutes les pages pour un bien meilleur rendu sur mobile :

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

3. Ajouter la ligne suivante dans tous les `<head>` de toutes les pages pour lier la feuille de style

```
<link rel="stylesheet" href="css/style.css">
```

# Design de l'élément pour mobile (mobile first pour être responsive)

1. Faire le style élément par élément pour ne pas devenir fou. Et utiliser l'éditeur de style du navigateur pour s'aider et avoir les changements en temps réel. **Mettre le navigateur en mode "téléphone"**

```
.classeDeLelementOuSelecteur {
  /* Code suivant ici */
}
```

2. Si l'élément contient d'autres éléments, ajouter un flex

```
display: flex;
flex-direction: column; /* Définir la direction des éléments "row" pour horizontal et "column" pour vertical */
```

3. Définir la taille (en % pour que ce soit relatif) si nécessaire

```
width: 50%;
```

4. Ajouter des marges, pour centrer un élément, utiliser `auto`

```
margin: auto; /* Centrer un élément */
margin: 10px auto; /* Centrer un élément horizontalement et mettre une marge de 10px en haut et en bas */
margin: 0px auto 0px 0px; /* Aligner un élément à gauche en mettant une marge auto à droite */
margin: 0px 0px 0px auto; /* Aligner un élément à droite en mettant une marge auto à gauche */
```

5. Ajouter du padding pour "grossir" l'intérieur d'un élément (le padding est une marge interne entre le contenu et la bordure de l'élément)

```
padding: 10px;
```

6. Ajouter une bordure et des bords arondis

```
border: solid black 3px;
border: dashed black 3px;
```

```
border-radius: 10px;
```

7. Annuler le style automatique des liens (`<a>`) et des listes (`ol` et `ul`) :

```
text-decoration: none;
list-style: none;
```

8. Définir une couleur du texte et du fond

```
color: white; /* Couleur du texte */
background-color: black; /* Couleur du fond */
```

9. Définir la police, le niveau de "gras", l'alignement du texte et la taille de la police (utiliser des rem pour cela pour avoir une unité relative)

```
font-family: Roboto;
font-weight: 700;
text-align: center;
font-size: 1.5rem;
```

10. Changer le curseur pour que cela soit un curseur pointeur ("la main")

```
cursor: pointer;
```

# Design de l'élément quand on passe la souris dessus (:hover)

1. Ajouter le même sélecteur avec ":hover"

```
.classeDeLelementOuSelecteur:hover {
  /* Code suivant ici */
}
```

2. Changer la couleur si nécessaire

```
color: black;
background-color: lightgrey;
```

3. Ajouter une transition pour que l'élément bouge quand on met la souris dessus

```
transition: ease-in-out;
transition-duration: 300ms;
transform: translateY(-5px);
```

# Design de l'élément sur PC

1. Quitter le mode "téléphone" mis précédemment, mais ne pas oublier de le réactiver après. Ajouter dans le sélecteur `@media` si il existe, sinon le créer :

```
@media (min-width: 1024px) {
  .classeDeLelementOuSelecteur {
    /* Code suivant ici */
  }

  /* Autres classes et sélecteurs peuvent s'imbriquer ici aussi */
}
```

2. Changer la direction du flex si nécessaire

```
flex-direction: row;
```

3. Définir une taille limite si besoin

```
width: 50%;
```

4. Ajouter une marge / centrer :

```
margin: auto;
margin: 20px;
/* ... comme vu précédemment */
```

5. Ajouter d'autres propriétés, si besoin voir la première procédure.

# Enregistrer les changements

1. Copier le code CSS dans le fichier CSS pour le sauvegarder

2. Répèter la procédure pour chaque élément de chaque page. Toujours le faire au fur et a mesure pour ne pas devenir fou. Et ne pas oublier le *mobile-first*

# Petit guide PHP pour le projet web

## Mettre en place des templates avec les fichiers .inc.php

Pour commencer on peut d'abord essayer de se consacrer à diviser les différentes parties du site en petites (séparer le header du reste par exemple).

Cela permet de minimiser la recondance du code dans le projet. Le projet oblige la création d'un dossier `inc/` dans le dossier principal du projet. Dans celui ci on peut y placer des fichiers tel que `header.inc.php` dans lequel on va simplement mettre notre header.

Ainsi ce code :

```
<html>
  <head>
    <title>Ewins</title>
  </head>
  <body>
    <header>
      <h1>Ewins</h1>
      <nav></nav>
    </header>
    <!-- Ici viens tout le reste du code mais tout ce qui est précédemment est répétitif sur toutes les pages du site
-->
  </body>
</html>
```

Peut devenir ceci :

```
<?php require_once "inc/header.inc.php"; ?>
<!-- Ici viens tout le reste du code mais tout ce qui est précédemment est répétitif sur toutes les pages du site --
```

```
>
```

# Mettre en place les classes modèles

Le projet web s'approche d'un modèle appellé le MVC (Modèle-Vue-Controlleur) c'est un modèle conçu pour éviter de mélanger tous les codes différents dans le projet. Il est séparé en 3 parties

- Les **modèles** qui seront dans le dossier `php/` du projet et qui sont des classes qui représentent les différents concepts du site (Utilisateur, Tournois, etc), en somme les différentes entités du MCD ou *presque* chaque table. Dans ce même dossier doit aussi se situer des autres classes qui permettent de créer les modèles et qui vont se connecter à la base de donnée (ce sont les seules classes qui vont contenir du code SQL)
- Le **controlleur**, c'est la logique du programme, ici il y a de grande chances que les controlleur soit au même endroits que les vues, ce n'est pas la meilleure idée mais c'est pas grave pour le projet.
- Les **vues**, c'est le code HTML et CSS du site, tout ce qui est relatif à l'affichage en tant que tel.

Je vais montrer un exemple plus concret plus tard dans cette synthèse

# Utiliser les sessions dans le projet pour gérer les utilisateur·ice·s

On peut se faire chier à créer les cookies soit même, ou bien on peut demander à PHP de tout gérer pour nous. Une **session** dans PHP va créer un cookie avec un identifiant unique (qui va être stoqué chez l'utilisateur), à cet identifiant, PHP va associer des informations (celles que vous définissez dans votre code). Ainsi cela permet de garder des informations sur les gens qui visitent le site et nottament de les authentifier ou de stoquer des données "en cache" sans avoir besoin de faire tout le temps des requètes vers la base de donnée.

Encore une fois je vais en parler de manière plus concrete un peu plus tard.