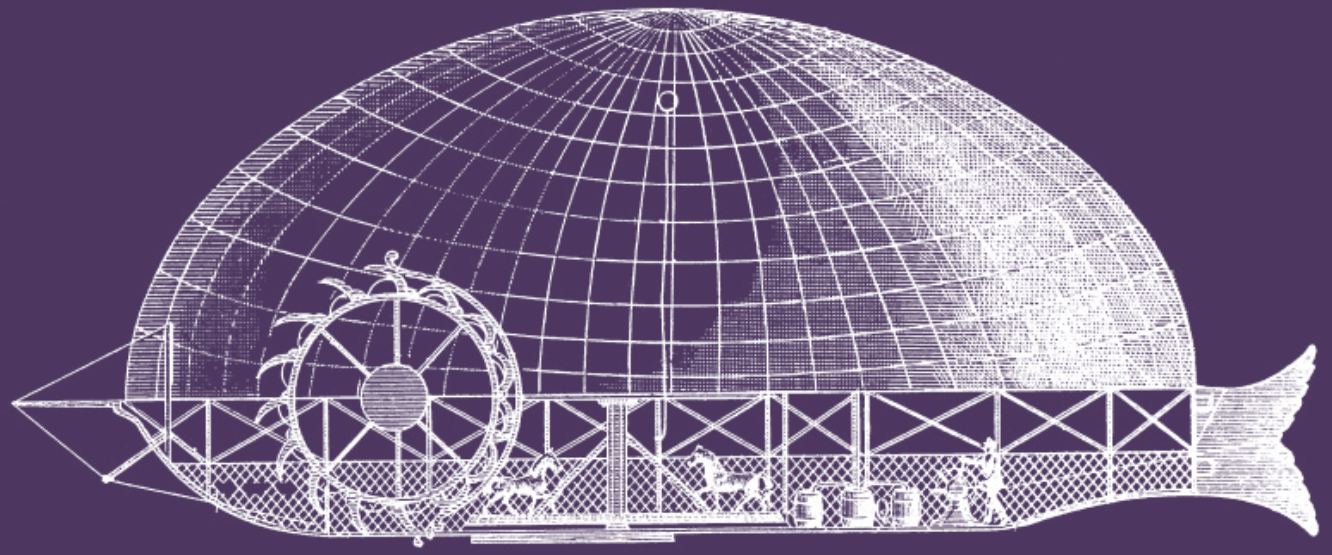
Méthodologie pour l'écriture du code (X)HTML



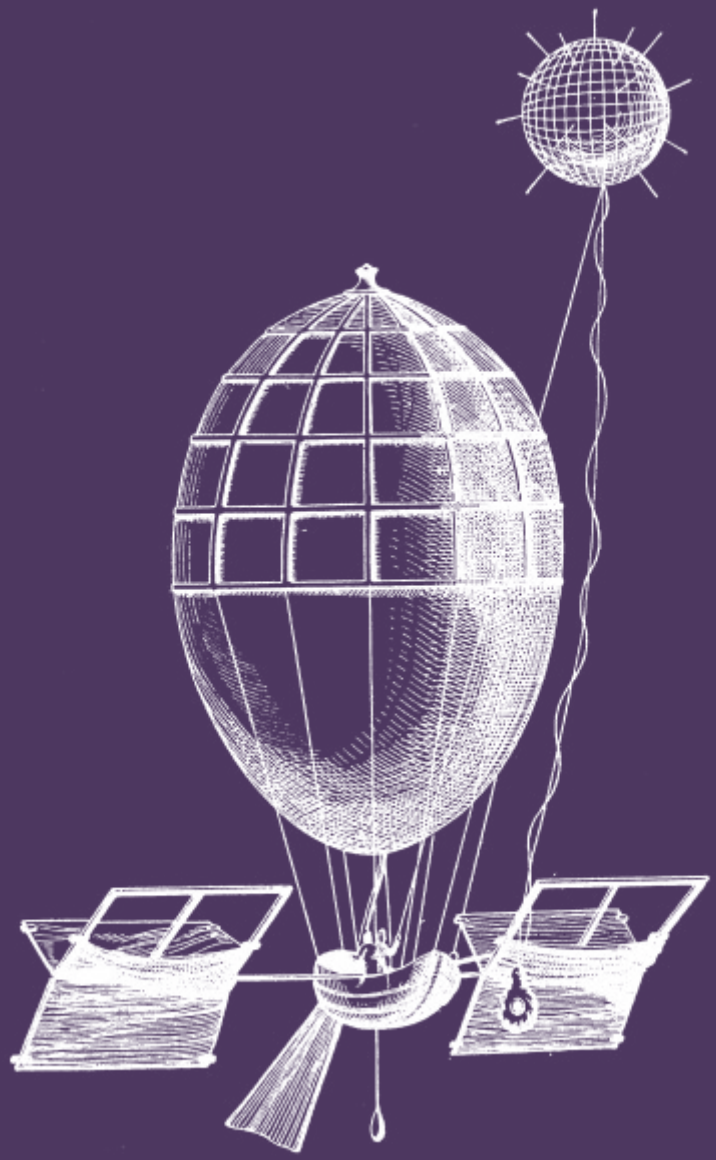
Philosophie *de* travail

Quand nous parlons de qualité, nous visons surtout l'ensemble des caractéristiques inhérentes à un site qui rendent possible, **de façon efficace, économique et contrôlée,** son **évolution**, sa **maintenance** et sa **portabilité** vers d'autres médias**.**

- Evolution et compatibilité future
- Maintenance
- Portabilité

La façon dont on écrit le code est **le point le plus sensible** *de toute la philosophie de la séparation contenu/forme.*

Pensez à votre code (X)HTML comme aux fondations de votre maison

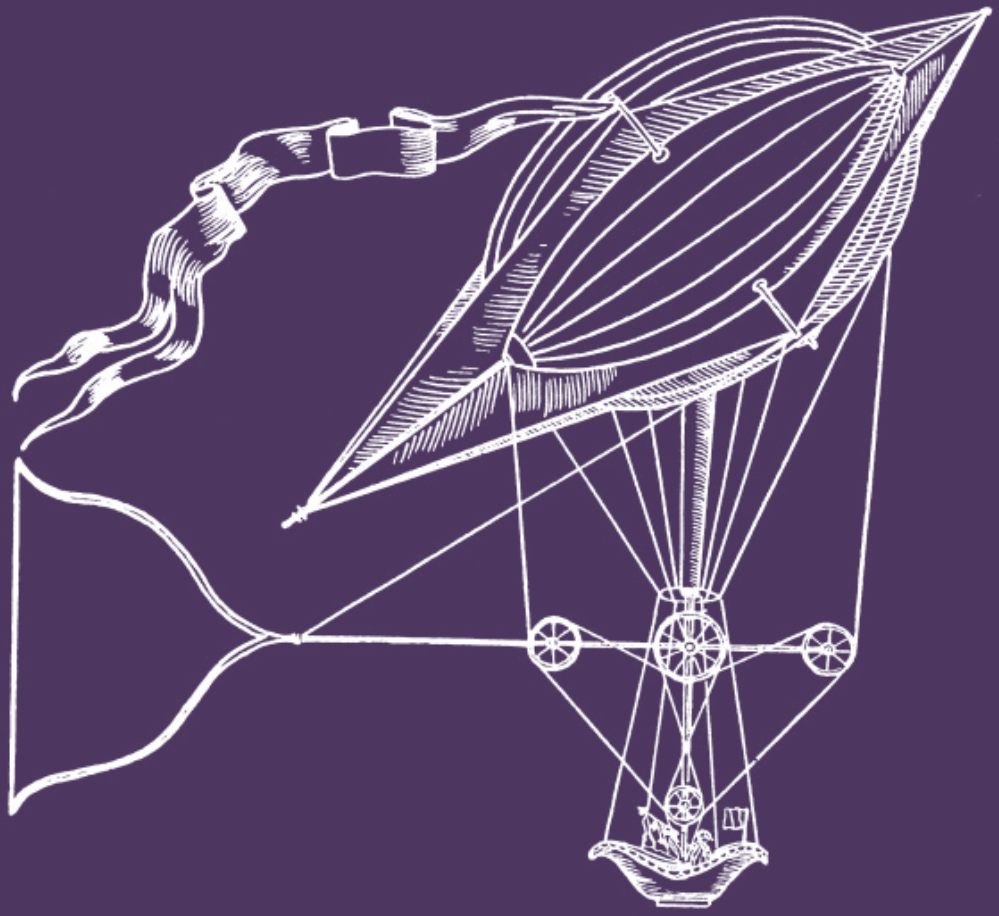


Les Règles

A respecter *toujours* …

3.1 La déclaration du DOCTYPE
3.2 L'appel aux feuilles de style
3.3 Déclaration du "*namespace*"
3.4 Déclaration de la langue principale
3.5 Définition du "*content-type*"
3.6 Titre du document
3.7 Les règles de compatibilité XHTML
3.8 Respecter la syntaxe du langage et ses éléments
3.9 Utiliser les tabulations et les retours à la ligne pour un code clair
3.10 Ne pas "forcer" l'ajout d'espaces ou les retours à la ligne
3.11 Valider votre code

Méthodes *et* Conventions



Editeur :

Choisissez-le et sachez **le pourquoi**

- Contrôlez le code
- Contrôlez le *charset*

Appeler toujours la CSS de **développement**

- Garder l'intégrité de la feuille de styles originale Studio
- Mieux suivre les mises à jour
- Mieux contrôler les bugs

Le code **(X)HTML** d'abord, le design **CSS** après …

…mais l'un n'exclut pas l'autre.

- Codage réfléchie, cohérente et précise
- « Voir » la page sans la CSS

Respecter les règles de nommage

- Noms sémantiques
- Sans caractères spéciaux ou espaces
- Qui ne commence pas par une chiffres
- Qui ne sont pas trop longs
- En anglais
- Nommage Camel
- *Quid* des microformats ? Nouvelles balises HTML5 ?

Conventions *des* commentaires

- Ajoutez toujours un commentaire à l'ouverture d'un **bloc composant** et un commentaire de fermeture

- Pensez à ajouter un commentaire simple à coté de la fermeture des principaux containers pour indiquer à quel bloc cette fermeture fait référence

Le code doit être **flexible** et **évolutif**

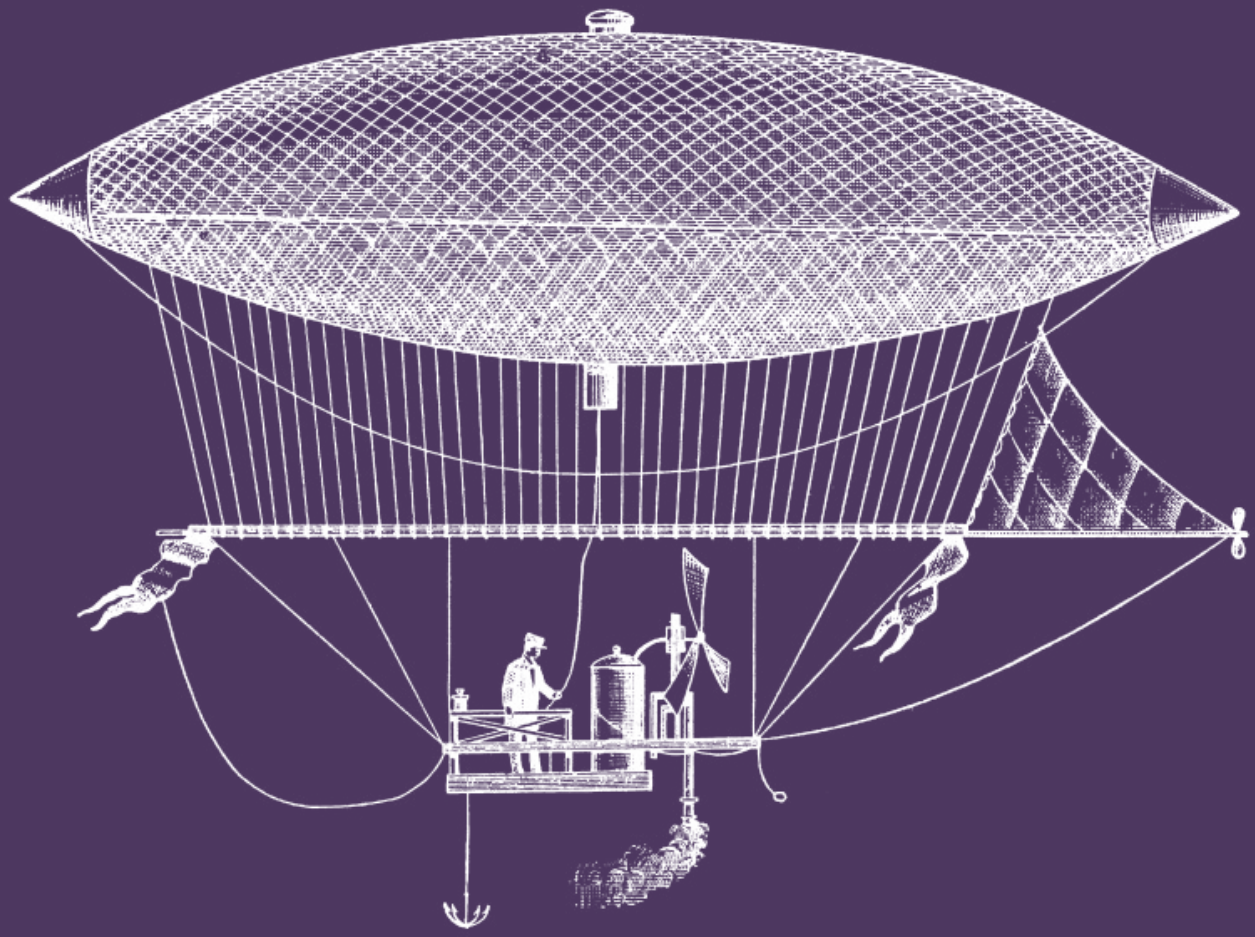
Pensez à tester les pages avec des contenus « extrêmes » et avec la plupart des balises

Eviter de commencer l'édition (X)HTML par la ***page d'accueil***

Avoir une très bonne visibilité des différents gabarits et éléments des pages internes et de commencer par elles

Penser en termes de « *composants* »

- Etudier la maquette graphique et/ou le wireframe pour identifier les différents blocs composants
- Grouper des blocs composants en fonction de leurs différences et/ou de leurs similitudes
- Séparer les composants dans un fichier XHTML à part, commenté



Les bonnes pratiques

Eviter les " *div-ities* "

Chaque balise a sa fonction,
chaque contenu a un rôle

Bien considérer l'attribution de classes et ou d'Ids

- Les classes sont utilisées **pour les exceptions**
- Les classes peuvent être utilisées en **combinaison avec d'autres classes**
- Donner un Id aux éléments **uniques (structurants et/ou dynamiques)** ou qui doivent être **manipulés par javascript**

Il ne doit pas exister de contenu "anonyme"

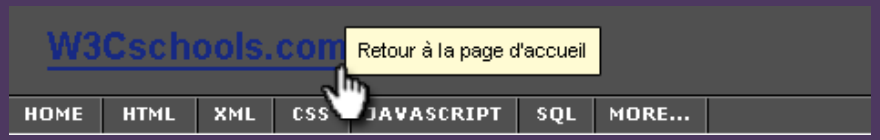
```
<body>
	<h1>Mon site</h1>
	<p><img src= "imgs/logo.gif" alt= " " /></p>
	<p>Lorem ipsum sit amet.</p>
	<p><a href= "doc2.html ">page suivante</a></p>
</body>
</html>
```

Ne **pas ajouter** de propriétés directement dans la page avec l'attribut "***style***"

Préférez **Firebug** pour faire des tests avec des différentes valeurs

Utiliser l'attribut " *title* " avec discernement

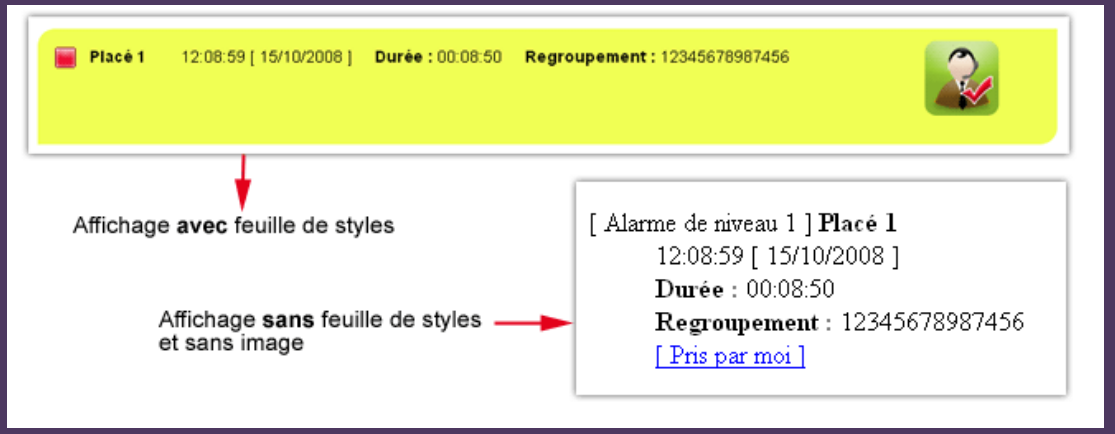
Images on



Images off

- « *title* » explique un lien, un contexte ou un contenu
- Préférez l'attribuer à la balise a

Sachez utiliser les listes de définition




```
<dl>
  <dt>
    <img src="imgs/alert1.gif" alt="[ Alarme de niveau 1 ]" />
    <strong>Placé 1</strong>
  </dt>
  <dd>12:08:59 [ 15/10/2008 ]</dd>
  <dd><strong>Durée :</strong> 00:08:50</dd>
  <dd><strong>Regroupement :</strong> 12345678987456</dd>
  <dd class="alertAc">
    <a href="#" title="Libérer main">
        <img src="imgs/suivi_pris.gif" alt="[ Pris par moi ]"/></a>
  </dd>
</dl>
```

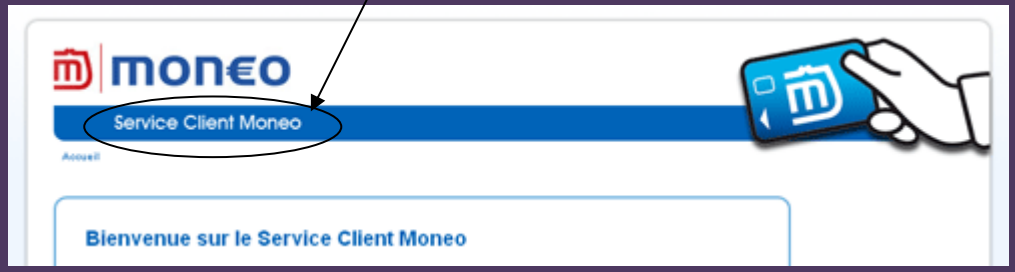
<*strong*> et **<*em*>** sont des informations sémantiques :

définissez-les dans le code

Utiliser des lignes (<hr />) pour séparer les principaux *groupes de contenu*

Consultez la page sans sa feuille de styles et insérez une balise <hr /> pour séparer les contenus que vous estimez nécessaires pour une meilleure lecture de la page.

Toujours utiliser la balise <h1> pour le *titre du site*



Le titre du site doit envoyer l'utilisateur à la page d'accueil

Toujours ajouter un **lien caché** pour sauter les menus

```
<p class="skip"><a href="#wrapper">Sauter le menu</a></p>
<ul id="ulMainNav">
   <li><a href="#">Rubrique 1</a></li>
   <li><a href="#">Rubrique 2</a></li>
   <li><a href="#">Rubrique 3</a></li>
</ul>
```

Créer vos **menus** avec ***soin***

- Un menu de rubriques doit toujours être structuré par des balises de liste ***(<ul>)***
- Traiter et identifier la rubrique courante (dans tous ses états)
- Une ligne de navigation n'est pas une liste

Les Images

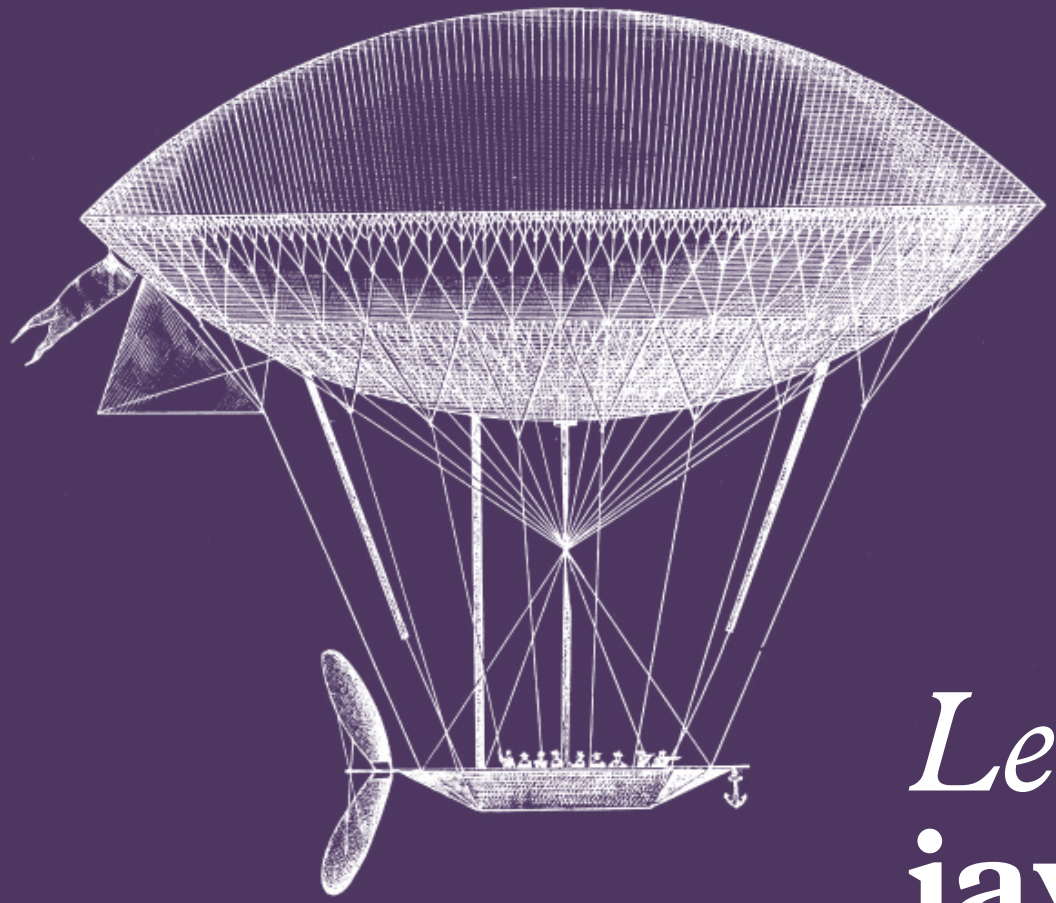
- Donner toujours une alternative textuelle aux images
- Donner ou pas une dimension aux images ?
- Pour les images décoratives, préférez l'utilisation d'images de fond (sauf pour celles qui « occupent de la place »)

Les formulaires

- Donner toujours un ID à un formulaire
- Bien considérer les attributs de la balise <*form*>
- Grouper les champs de même nature avec la balise <*fieldset*>
- Si possible, préférer la balise <*legend*> pour le titre d'un fieldset
- Utiliser les labels pour associer un intitulé à son champ
- Toujours grouper le label et son champ dans une balise <p>
- Attribuer toujours un ID et un "*name*" à un champ de formulaire
- Prévoir la construction de boutons liens et des boutons input
- Créer une structure pour les boutons des formulaires
- Coder correctement les *Checkboxes* et/ou *Radios*
- Prévoir les éléments d'information fréquents dans un formulaire

Les Tableaux

- Utiliser la bonne balise pour le titre d'un tableau
- Ajouter un résumé des données du tableau
- Utiliser la balise correcte pour les entêtes de colonne ou de ligne
- Renseigner la logique de lecture du tableau
- Considérer les cas de figure possible (alignement cellules, lignes alternées, lien pour les entêtes, flèche de tri, etc.)



Le codage **javascript**

Répétez après moi :
Je suis non-intrusif

- Ne pas définir une fonction directement dans la page
- Séparer le comportement du contenu
- Donner toujours une alternative au javascript
- Penser les fonctions pour qu'elles ne soient pas dépendantes d'un seul type d'input ou de navigateur
- Utiliser la syntaxe correcte pour le passage des variables par une URL