Le Guide
(très modeste)
Ultime
de l'intégration réussie

Vanessa Sant'André - vanessa@14bis.fr

Tout d’abord, comme tout dans le monde de l’informatique (et pas seulement dans le métier d’intégrateur/trice), la première règle d’or est :

chaque frappe de trop dans ton code
est une frappe de trop dans ton code

(et vice versa)*

** y compris pour nommer vos dossiers.*

Vous allez voir que toutes les bonnes pratiques qui suivront vont dans ce sens.

On organise correctement nos dossiers

La façon de nommer les fichiers et dossiers est très importante car nous, les intégrateur.trice.s, nous parlons en **jargon.** Vu que nous sommes au milieu de la chaîne de création d'un site internet, le jargon nous permet de communiquer correctement avec le.la graphiste, le.a dév, le.a chef.fe de projet, etc. Donc, voici la structure alpha d'organisation de nos fichiers et dossiers basée dans ce jargon :

Fichiers :	
index.html (ou .aspx / .php, etc) contact / panier, etc	On nomme notre page d'accueil **TOUJOURS** index et on la place à la racine. Le.a dév s'occupera de la renommer et changer son emplacement, si nécessaire
Dossiers :	
img	Ici on place les images. Attention : img et pas image ou images. Pourquoi ? Économie de frappe always
js	Ici on place notre « main.js » / « app.js » / etc
css	Ici on place notre css personnelle nommée « **style.css** » (et pas autrement) et nos css vendors (normalise ou reset, etc)
fonts ou webfonts	Et pas « polices » ou « fts » etc. Pourquoi ? Parce que nous suivons la terminologie des grands vendors, comme Fontawesome

Et surtout : **PAS DE MAJUSCULE, pas d'espace (si tu tiens à séparer les mots : _ ou -), pas de caractères spéciaux dans le nom de tes dossiers et fichiers**

Dans style.css

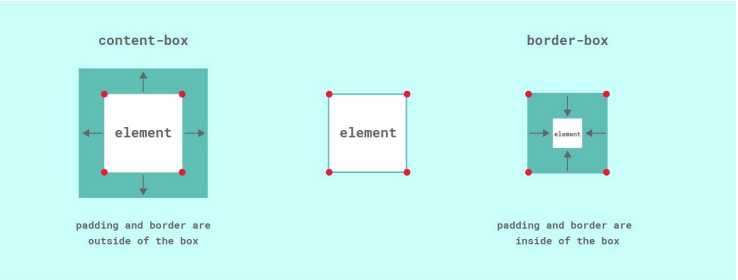
L'ordre d'écriture est :

1) On fait nos @import et on déclare nos @font-face -> **les sources externes sont les premières à être chargées et affichées**

2) On déclare les styles assignés à toutes les balises. Sans que cela soit une règle gravée dans le marbre, en général on ajoute dans les * seulement :

```
* {box-sizing : border-box} // par défaut c'est content-box. En changeant à border-box, les paddings seront désormais ajoutés à l'intérieur de l'élément. Beaucoup plus simple pour calculer leur taille
```

On peut également ajouter une transition à ce niveau, mais attention, selon la complexité et taille du site, ça peut ralentir son chargement (trop de balises concernées)



Dans style.css - le /*GLOBAL*/

3) Nous entrons dans le territoire des éléments dits globaux. A partir d'ici, vous pouvez même commenter **GLOBAL** dans ton code (jargon)

Et on s'occupe de la balise html. Dans cette balise, **une seule et unique ligne de code** :

```
/**GLOBAL**/

html {font-size : 62.5 %} // vu qu'on travaillera en priorité avec des unités rem, c'est beaucoup plus simple de les calculer à partir de 10px que de 16px (16px est la taille par défaut du paragraphe dans les principaux navigateurs). Comme vous pouvez imaginer, 62.5 % de 16px = 10px. Pourquoi on met 62.5 % et pas directement 10px ici ? Car nous sommes gentil.le.s et nous respectons les personnes malvoyantes qui augmentent la taille de la police dans les paramètres de leurs navigateurs.
```

Dans style.css - le body

4) Ensuite on déclare le body. Dans cette ligne (qu'on charge le maximum qu'on peut), on ajoute les mises en formes qui concernent la balise <p>. Pourquoi ? On profite d'un puissant principe d'héritage (le body est un super-parent) et on règle, déjà à ce niveau, toute la mise en forme d'une grande majorité du contenu du site.

```
body {
font-family: 'Open Sans', sans-serif;  // font-family du <p>
font-size: 1.8rem;  // on redémarre la taille du <p> à 16px (ou la
taille désigné par le graphiste)
line-height: 2;  // l'interligne sans unité de valeur (pas de rem, px ou
em) l'idée est de relativiser l'interligne selon la taille de la
police. Ici la balise enfant du body (hs, ps, etc) prendra une
interligne de 2x la taille de son font-size
margin: 0;  // au cas où on n'utilise pas normalise ou reset (attention,
bootstrap est normalise). Car le body a toujours 8px de margin par
défaut..
color: #6a5c6b;  // basée sur la color du <p> toujours
}
// + background, etc, au cas où
```

Dans style.css - les balises correctives

5) Toujours dans le territoire */**GLOBAL**/*, on ajoute les balises correctives (au cas, vous n'utilisez pas normalize.css) et les classes utiles :

```
img {max-width:100%} // les images ne dépasseront pas la taille de leur parent ni leur propre taille
.container {
max-width: 1100px; //taille toujours en pixel
margin: 0 auto;
padding : 0 15px
} // et c'est tout (pas de flex, rien d'autre). Attention, container est une class jargon, donc, ne vous amusez pas à l'appeler contenant, toto, etc
.clearfix, etc
```

Dans le territoire global, on règle également toutes les balises génériques : tous les a, les hs, etc. Pensez toujours à fusionner le max vos mises en formes :

```
h1, h2, h3 {} .. etc
```

Dès qu'on démarre des balises spécifiques (<header> / les class / etc) on quitte le GLOBAL (et on commente ça dans le code). Exemple : `/*End GLOBAL*/`

CSS & HTML - astuces en vrac

1) On ajoute toujours un id dans nos sections. On utilisera cet id pour mettre en forme les enfants. Exemple :

`<section id="rappeurs">`
`<h2>Mes rappeurs préférés</h2>`
`<h3>Le rap, LA musique du XXIème siècle`
`</h3>`
`</section>`
	`//Le html est plus propre et plus malléable`
	`#rappeurs h2`
	`#rappeurs h3`

`<section>`
`<h2 id="rap_prefere">Mes rappeurs préférés</h2>`
`<h3 id="rap">Le rap, LA musique du XXIème siècle`
`</h3>`
`</section>`
	`//Je suis, désormais obligé.e à ajouter un id ou class par nouvelle balise enfant de ma section, bof`
	`#rap_prefere`
	`#rap`

Ça que nous revient à connaître la **deuxième règle d'or** :

L'info, on la monte le plus haut possible.
CSS est une affaire de PARENT (et d'héritage)

Plus haut vous montez l'info (en donnant des ids aux sections, par exemple) plus propre et malléable sera votre code. On garde les class pour les généralisations (classes utiles), comme par exemple, la mise en forme d'un bouton (à savoir, une mise en forme très agressive ou marquée) qu'on utilisera mille fois dans notre code. Autrement dit, on attrape les enfants en utilisant l'id de leur parent.

2) Les labels doivent être branchés à leurs inputs.

Option 1	Option 2

`<label for="prenom">Ton prénom</label>`
`<input type="text" id="prenom">`
	`<label>Ton prénom`
	`<input type="text">`
	`</label>`

Vous ne voulez pas que le label s'affiche dans l'écran (mais pourtant il est un élément obligatoire), vous faites :

```
label {position:absolute ; left :-999px} // vous le enlevez de l'écran visuellement, mais vous gardez sa fonction, qui est d'indiquer à des personnes non-voyantes qu'est-ce que l'input attend d'elles (leur prénom, par exemple). Pas de display:none (car un élément en display:none est muet pour les logiciels lecteurs d'écran comme NVDA)
```

2) On intègre en Mobile First.
Pourquoi ? Parce que depuis 2014, l'internet est mobile et pas desktop. Donc, soyons dans l'air des temps

3) On fusionne au max les informations (margins / paddings / backgrounds) et on écrit en une seule ligne. Un schéma :

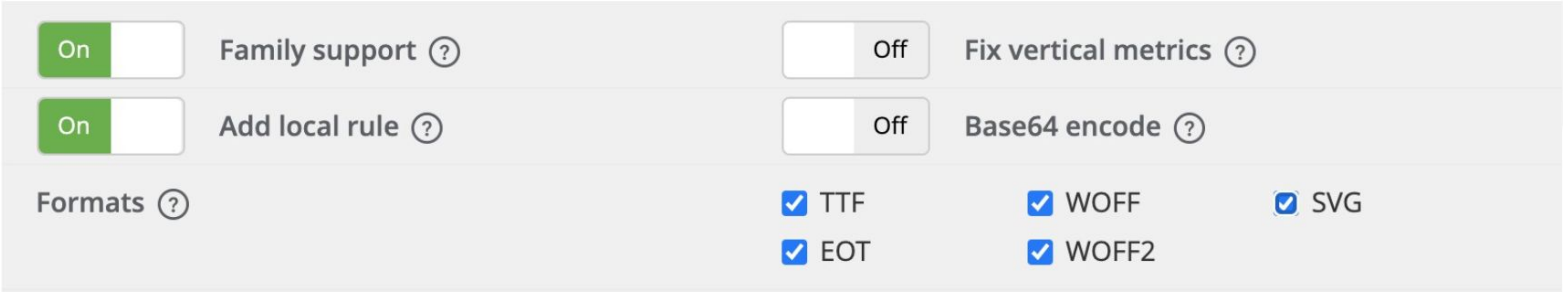
Margins ou paddings :
Y X Y X (top / right / bottom / left) - ex : margin : 10px 4px 5px 6px
Y X (top ET bottom / right ET left) - ex : margin : 10px 5px
~ (toutes les valeurs sont égales) - ex : margin : 10px
Y X Y (top / right ET left / bottom) - ex : margin : 10px 5px 6px

Backgrounds :
background : url(../img/chaton.png) repeat-x center bottom #fff;
valeurs repeat : repeat-x / repeat-y / no-repeat / repeat
valeurs X : left / right / center ou (px/rem/em/%)
valeurs Y : top / bottom / middle ou (px/rem/em/%)
valeurs color : #fff, rgb/rgba, white/etc ou transparent

En plus : on centre nos background (pour la responsive) et on soigne nous backgrounds-size (cover plutôt que contain)

4) On privilégie les rem pour les petits éléments (taille maximum d'un élément en pixel : 300px) / **on fait des max-width en pixel pour les grands parents** (main / section) **et pourcentage pour les enfants**

5) Pour les font-faces, on fait comme il faut, à savoir, avec toutes les extensions (ttf / woff, etc). Pour ça, il y a des générateurs. Perso, j'aime bien « Transfonter » - https://transfonter.org



6) SUR INTERNET, ON NE CRIE PAS. Si vous avez besoin d'écrire une phrase en majuscule, vous écrivez normal, détendu.e... et ensuite vous faites un text-transform dans votre feuille de style. Pourquoi ?

- Déjà parce que ce n'est pas vous qui allez entrer le contenu et sans text-transform, vous obligez l'éditeur.trice du site à écrire toujours en criant (bof et en plus il/elle ne le fera pas et votre jolie mise en forme ne va pas tenir dans la durée)
- Ensuite car Google est case-sensible. Si vous criez, il vous référence en criant et vous allez attirer que des visiteurs bizarres :-)

7) Vous allez ajouter des classes à vos boutons (classes utiles) ou faux boutons (un <a> qui ressemble à un bouton, par exemple). Et ces classes s'appelleront soit **.cta** (call to action) soit **.btn**. **C'est comme ça, c'est du jargon.**

Un tour sur les balises sémantiques

<article> - c'est une balise plutôt bizarre. La documentation de W3 ne nous avance pas trop sur sa bonne utilisation. L'unique vraie règle est : tous les articles doivent avoir un enfant H. Après, c'est l'inconnu... On trouve des blocs météo dans des balises article...

Bizarrement, ça peut être un article si 10 et/ou 30 sont des titres ...

Donc, officieusement, c'est convenu qu'**article est tout ce qui est passible d'entrer dans un flux RSS**. Utilisez cette règle si vous voulez coder proprement…Exemple d'un potentiel article, récupéré du flux du journal Le Monde

```
▼<item>
  ▼<title>
     <![CDATA[ Entreprises : le Covid-19 entraîne une vague de consolidations ]]>
   </title>
   <pubDate>Fri, 21 May 2021 12:00:24 +0200</pubDate>
  ▼<description>
     <![CDATA[ Depuis le début de l'année, le marché des fusions-acquisitions dans le monde
     atteint des niveaux records. ]]>
   </description>
   <guid isPermaLink="true">https://www.lemonde.fr/economie/article/2021/05/21/entreprises-
   le-covid-19-entraine-une-vague-de-consolidations_6081009_3234.html</guid>
   <link>https://www.lemonde.fr/economie/article/2021/05/21/entreprises-le-covid-19-
   entraine-une-vague-de-consolidations_6081009_3234.html</link>
  ▼<media:content
   url="https://img.lemde.fr/2021/05/21/223/0/2734/1367/644/322/60/0/1b71c47_21190428-covid-
   banques-affaires.jpg" width="644" height="322">
     <media:credit scheme="urn:ebu">AUREL</media:credit>
   </media:content>
 </item>
```

<section> - il faut un H à l'intérieur, obligatoire. Par définition, une section est un lot d'information différent d'un autre lot d'information. Exemple : Témoignages client / Nos services, etc

<aside> - complément d'information de son parent (et pas uniquement sidebar, comme c'est normalement utilisé). Un sidebar peut être un aside. Mais également ce « Lire aussi » ci-dessous :

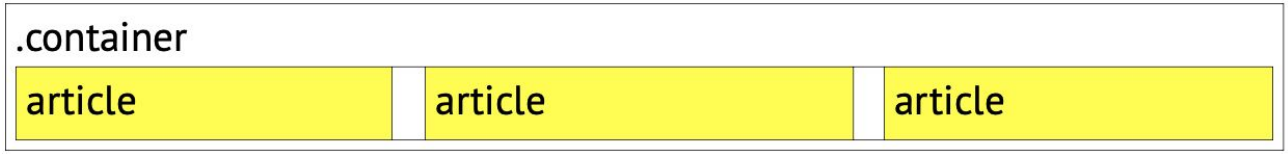
septentrionale du royaume chérifien. Si la pression semblait s'être dissipée, jeudi 20 mai, après l'expulsion de 5 600 de ces migrants vers le Maroc, cet épisode va marquer durablement les relations entre Rabat et Madrid et, au-delà, Bruxelles.

Lire aussi | **« Tout ça pour ça » : le dépit des migrants arrivés à la nage ou à pied à Ceuta et refoulés vers le Maroc**

Car cette crise a été mûrie et mise en scène par les autorités marocaines, dont la police a quasiment montré la voie de Ceuta à une jeunesse en pleine détresse sociale. A Madrid, où le ministre de la défense a accusé le Maroc d'*« agression »* et de *« chantage »*,

Un sidebar peut être un aside qui complémente son parent <main> et notre « Lire aussi » est un aside qui complement son parent <article> (donc, son contenu doit être forcément lié au contenu de l'article)

Petite astuce finale pour la route vers l'intégration parfaite. Imaginez que vous avez cette structure




```
article + article {margin-left:10px} // saute le premier article et mettez une margin-left dans les suivants
```

Cela équivaut à :

```
article {margin-left:10px}
article:first-child {margin-left:0px}
```

mais en une ligne:)

Et la troisième et plus importante des règles d'or :

Questionnez-vous toujours avant de démarrer votre intégration: Est-ce que cet élément dans la maquette concerne la mise en forme du site (donc ça vous concerne) ou son contenu (donc, ça ne vous concerne pas)

Et si c'est du contenu (à savoir, si ça ne vous concerne pas), c'est votre boulot faire de façon que son entrée et édition soit malléable, évolutive et robuste

La force soit avec vous !
Vanessa