Beautiful CSS

Structurer, documenter, maintenir.

Petit rappel CSS :

- Décrit la présentation de documents HTML, XML, XUL, ...

Bloc de règles

- sélecteur {
  propriété: valeur;
  }

Structure (ou pas)

```
#TB_window a:hover {color: #000;}

#TB_window a:active {color: #666666;}

#TB_window a:focus{color: #666666;}

#TB_overlay { position:fixed; z-index:100; top:0px; left:0px; height:100%; width:100%; }

* html #TB_overlay { position:absolute; height:expression(document.body.scrollHeight > document.body.offsetHeight ? document.body.scrollHeight :
document.body.offsetHeight + 'px'); }

.TB_overlayMacFFBGHack {background: url(../img/skin/macFFBgHack.png) repeat;}

.TB_overlayBG { background-color:#000; filter:alpha(opacity=75); -moz-opacity: 0.75; opacity: 0.75; }

#TB_window { position:fixed; background:#ffffff; z-index:102; color:#000000; display:none; border:4px solid #525252; text-align:left; top:50%; left:50%; }

* html #TB_window { position:absolute; margin-top:expression(0 - parseInt(this.offsetHeight / 2) + (TBWindowMargin = document.documentElement &&
document.documentElement.scrollTop || document.body.scrollTop) + 'px'); }

#TB_window img#TB_Image { display:block; margin:15px 0 0 15px; border-right:1px solid #ccc; border:1px solid #666; }

#TB_caption { height:25px; padding:7px 30px 10px 25px; float:left; }

#TB_closeWindow { height:25px; padding:11px 25px 10px 0; float:right; }

#TB_closeAjaxWindow { padding:7px 10px 5px 0; margin-bottom:1px; text-align:right; float:right; }

#TB_ajaxWindowTitle { float:left; padding:7px 0 5px 10px; margin-bottom:1px; }

#TB_title { background-color:#e8e8e8; height:27px; }

#TB_ajaxContent { clear:both; padding:2px 15px 15px 15px; overflow:auto; text-align:left; line-height:1.4em; }

#TB_ajaxContent.TB_modal { padding:15px; }

#TB_ajaxContent p { padding:5px 0px 5px 0px; }

#TB_load { position:fixed; display:none; height:13px; width:208px; z-index:103; top:50%; left:50%; margin:-6px 0 0 -104px; }

* html #TB_load { position:absolute; margin-top:expression(0 - parseInt(this.offsetHeight / 2) + (TBWindowMargin = document.documentElement &&
document.documentElement.scrollTop || document.body.scrollTop) + 'px'); }
```

Structure (ou pas)

Stop au carnage !

Convention de dev

Formatage :

Le formatage du code consiste en son écriture aéré et homogène.

- Une ligne ne doit pas dépasser 80 colonnes ;
- Retour à la ligne :
  - A la fin de chaque bloc de règles ;
  - A la fin de chaque règle CSS, sauf si il en existe qu'une ;
  - Avant chaque nouveau groupe de bloc de règles ;
- Tabulation :
  - une tabulation correspondant à 4 espaces devant chaque règle CSS
- Espaces :
  - Peu être utilisé avant une valeur (ex : propriété: valeur)

Convention de dev

Nommage :

- Ecrivez votre code et vos commentaires toujours en anglais ;
- Utilisez slug plutôt que “lowerCamelCase” pour créer des ensembles de mots ;
  ex : ~~boxTypeI~~ devient box-type-I
- Les caractères recommandés par le W3C sont de type alphanumérique, ainsi que le trait d’union, l’underscore ne fait pas partie des recommandations.
- Nommez vos sélecteurs de manières sémantiques :
  ex : ~~texteRougeGras~~ devient warning-I
  ex : blocSimpleShadow devient box-type-I
  ex : blocRight devient box-contextual
- Utilisez des mots clefs génériques : cols, col, inner, box, content, list, menu, footer, header ... inspirez vous de HTML 5

Convention de dev

Commentez :

- Utilisez un en-tête de fichier qui décrit son contenu :

```
/* ------------------------------------------------------------------------
    CSS Document

    project:    XYZ Inc.
    created:    2008-10-17
    author:     Yves Van Goethem
    email:      yves.vangoethem@entreprise.com
    website:    http://www.entreprise.com

    summary:    GENERIC
                HEADER
                FOOTER
                BREADCRUMP
                CONTENT
                ...
------------------------------------------------------------------------ */
```

Convention de dev

Commentez :

- Utilisez des en-têtes pour chaque catégorie définie dans “summary” (précédé par un retour de ligne en plus du dernier bloc de règles CSS) :

```
/*   =HEADER
---------------------------------------------------------------------------- */
#header {
     border:1px solid #000;
}


/*   =FOOTER
---------------------------------------------------------------------------- */
#footer {
     border:2px solid red;
}
```

Convention de dev

Commentez :

- Expliquez l'utilisation de choses propriétaires, de hacks, de règles compliqués ...
  A l'aide de mots clefs et/ou de références : TRICKY, INFO, BUGGY ...

```
body {
	font-size:72.75%; /* :INFO: 1em = 11px */
}

.box-type-1 {
	float:left;
	margin:10px 20px;
	display:inline; /* :TRICKY: IE 6 double-margin bug */
}

.box-type-2 {
	/*
		:TRICKY: Firefox 3 transparent background
		http://developer.mozilla.org/En/CSS/Color
	*/
	 background:rgba(125, 125, 125, 0.5);
}
```

Structure de dev

```
/* ------------------------------------------------------------------------
     CSS Document

     project:    XYZ Inc.
     created:    2008-10-17
     author:     Yves Van Goethem
     email:      yves.vangoethem@entreprise.com
     website:    http://www.entreprise.com

     summary:    GENERIC
                 HEADER
------------------------------------------------------------------------ */


/*    =GENERIC
------------------------------------------------------------------------ */
body {
     font-size:68. 75%; /* :INFO: 1em = 11px */
}


/*    =HEADER
------------------------------------------------------------------------ */
#header {
     display:-moz-inline-box; /* :TRICKY: Gecko 1.8 inline-block */
     display:inline-block;
}
```

Simplifie toi la vie !

- Toutes ces conventions permettent de définir un vrai modèle de travail, elles sont inspiré d'autres conventions de développement tel que Java, PHP, C, JavaScript, ... tout développeur pourra s'y retrouver.

- Vous n'êtes pas seul, on bosse sur un marché international et notre métier est déjà assé incompris comme ça...
  Rappelez-vous ... CSS est destiné à faciliter la maintenance d'un site, d'où la nécessité d'avoir un code lisible, structuré, et donc accessible à tout développeur, peu importe son origine, son équipement, lieu de travail, ...

Bonnes pratiques

CSS reset

Appliquez toujours un reset CSS avant de faire quoi que ce sois :

```
* {
    margin:0;
    padding:0;
    vertical-align:baseline;
    line-height:1.35em;
}
```

Ciblez d'autres spécificités :

```
a { color: green; }

a img, fieldset, form {
    border:0;
}

ins, abbr, acronym {
    text-decoration:none;
    border:0;
    font-style:normal;
}
```

Bonnes pratiques

Class génériques intrusives :

- N’utilisez pas de class génériques intrusives de types :
  - clear, clearfix, spacer
  - floatLeft, floatRight
  - marginT10, paddingB10, border5px
  - bold, underline, leftArrow
  - ...

Bonnes pratiques

Imbriquation de fichiers (en prod) :

- Préférez l'utilisation de media-queries dans votre code CSS, plutôt que de faire plusieurs appels dans votre code HTML.
  Améliorez la maintenance en séparant d'avantage la CSS de l'HTML et les performances en faisant moins de requêtes.

```
<link rel="stylesheet" href="css/basic.css" type="text/css" charset="utf-8" />
<link rel="stylesheet" href="css/print.css" type="text/css" media="print" charset="utf-8" />
```

devient 1 seul fichier CSS avec :

```
@media print {
    ...
}
@media screen {
    ...
}
```

Bonnes pratiques

Remédiez à la “Classitis”

```
<h2 class=”ttlNews”>La fin des happy hours</h2>
<p class=”blocNews”>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
<span class=”blocNews”><a href=”#” class=”linkNews”>En savoir plus</a></span>
```

Devient :

```
<div class=”box-news-1”>
    <h2>La fin des happy hours</h2>
    <div>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.</div>
    <span><a href=”#”>En savoir plus</a></span>
</div>
```

Bonnes pratiques

Ne déstructurez par le code en pensant le structurer ...

- Définissez tout à un endroit, ça évitera aux développeurs d'ouvrir 3 fichiers pour modifier un même sélecteur.
  L'erreur la plus commune est de se baser sur plusieurs fichiers : layout.css, typography.css, color.css ... Ce qui mène souvent à des surchages de règles.
- Par ailleurs, écrivez vos hacks de manière à ce qu'ils sautent aux yeux, utilisez les commentaires d'aides qu'on a vu tout à l'heure ou écrivez les entièrement sur une ligne, ne les externalisez pas dans d'autres fichiers.
- Suivez les recommendations de développement "Zen", développez de manière à faciliter la dégradation sur les navigateurs d'anciennes générations et l'évolution sur les navigateurs modernes.

Bonnes pratiques

Utilisez les sprites CSS en sachant que :

- CSS est destiné à définir la présentation, évitez les techniques de remplacement CSS, le contenu n'est pas sensé se retrouver dans un code CSS, cela entraine des problèmes d'accessibilités :
  - Impossible de modifier les tailles de polices ;
  - Impossible d'imprimer le contenu ;
  - Illisible pour les lecteurs d'écrans ;
  - Tout cela est également le câs pour sIFR
- Si le contenu de votre bloc dépasse ses dimensions, il risque de faire apparaitre d'autres parties du fichier qui sert de Sprite

Bonnes pratiques

Typographie

- Concevez des pages accessibles, peu importe l'équipement et les méthodes de navigation des utilisateurs, pour ce faire :
  - Utilisez des unités de valeurs relatives Cadratin (em), Pourcentage (%) pour définir les tailles de textes ;
  - Évitez les unités fixes et non destinées au web Pixel (px,) Point (pt), .. ;
  - Assurez-vous de la lisibilité de votre texte : les couleurs, les tailles, l'interlignage, les polices ...

Bonnes pratiques

display:none; accessible

- Les lecteurs d'écrans autant que les navigateurs sont sensé respecter autant que possible les spécifications du W3C, donc évitez :
    - text-indent:-9999em; le texte sera autant tronqué que sur un navigateur
    - position:absolute; top:-9999em; le texte sera positionné de manière non naturelle par rapport au reste du document et entraine des bugs sur WindowsEyes
- Utilisez display:none, ou supprimer les du documents quand il s'avère vraiment nécessaire de cacher des choses.
- Sinon préférez position:absolute; left:-9999em;
  Vous assurerez ainsi une position naturelle dans le flux de la page tout en le cachant sans en tronquer le texte.
  Si vous développez des pages dans lesquels la langue se lit de droite à gauche préférez right:-9999em;

Bonnes pratiques

Filter et Expression MSIE

- Évitez autant que possibles les filtres et les expressions, en réalité ce sont des moulinettes infames qui surcharge la mémoire du client en ré-exécutant l'expression à chaque action de l'utilisateur ... un mouvement de souris suffit.

- Si vous ne pouvez pas les éviter, assurez-vous de cibler le bon navigateur, sachez qu'un star-hack : `* html #foobar {filter: progid:DXImageTransform.Microsoft.AlphaImageLoader(...); }`
  Sera insuffisant pour cibler IE 6, IE 7 en sera également affecté d'un point de vue performance même s'il n'est pas pris en compte visuellement.
  Pour évitez cela, combinez le star-hack à l'underscore-hack :
  `* html #foobar { _filter: progid:DXImageTransform.Microsoft.AlphaImageLoader(...); }`

Merci.

Sources d'inspiration :

- W3C
- Clearleft
- Alsacréations

yves.vangoethem@gmail.com