

MINI-CONVENTION CSS

bonnes pratiques au quotidien



GÉNÉRALITÉS

- La feuille de style CSS est de préférence **unique** et **minifiée** et appelée à l'aide d'un élément `<link>` dans la section `<head>`
- Les versions pour médias alternatifs (print, mobile) sont situées dans la même feuille de styles, en fin de document avec des règles `@media`
- Utiliser un seul mode d'indentation et s'y tenir (espaces, tabulations). Utiliser EditorConfig
- Utiliser toujours le même type de guillemets. De préférence des doubles guillemets, exemple : `content: ""`;
- Utiliser toujours des guillemets pour les valeurs dans les sélecteurs, exemple : `input[type="checkbox"]`
- Toujours terminer les déclarations par un `;`
- Éviter d'utiliser `!important`
- Valider le code avec **CSSLint** (disponible en plugins d'éditeur de code ou gulp)

BONNES PRATIQUES

qui facilitent la vie



MODÈLE DE BOÎTE

Opter pour le modèle de boîte CSS3 (**box-sizing: border-box**) en début de la feuille de style

```
* {  
  box-sizing: border-box;  
}
```



(ou bien)

```
html {  
  box-sizing: border-box;  
}  
  
* {  
  box-sizing: inherit;  
}
```



info : <https://blog.goetter.fr/2012/07/27/box-sizing-et-pourquoi-pas/>

TAILLES DE POLICES

Opter pour des tailles de polices **fluides** (de préférence en **rem**).

```
body {  
  font-size: 14px;  
}
```



```
html {  
  font-size: 62.5%;  
}  
body {  
  font-size: 1.4rem;  
}
```



FLUX

Éviter de sortir les éléments du flux (float, position) sans nécessité.

```
div {  
  position: absolute;  
  right: 0;  
}
```



```
div {  
  margin-left: auto;  
}
```



```
div {  
  float: left;  
  clear: both;  
  width: 100%;  
}
```



```
div {  
  float: none;  
}
```



POSITIONNEMENT

Positionner les éléments en choisissant de préférence parmi ces méthodes, dans l'ordre :

1

```
display: block | inline;
```

2

```
display: flex | inline-flex; ❤️
```

3

```
display: inline-block | table-cell;
```

4

```
float: left | right;
```

5

```
position: relative | absolute | sticky | fixed;
```


LECTURE

Écrire des syntaxes **compréhensibles** par des êtres humains et des collègues.

```
li + li {  
  visibility: hidden;  
}
```



```
li:not(:first-child) {  
  visibility: hidden;  
}
```



(un peu plus lisible)

NAMESPACES

Préfixer les classes par « namespace » pour les regrouper et les distinguer aisément.

```
.o-container, .o-mod, -o-grid-container {  
  /* objects : éléments génériques multitâches */  
}
```

```
.c-button, .c-nav, -c-lightbox {  
  /* components : éléments concrets */  
}
```

```
.is-opened, .is-hidden, .has-* {  
  /* state : désigne un état ou une condition */  
}
```

```
.js-menu, .js-is-hidden {  
  /* comportement : éléments liés à JavaScript */  
}
```

MAINTENANCE ET LISIBILITÉ

produire du code sur le long terme



POIDS DES SÉLECTEURS

Éviter de surcharger un sélecteur, car cela lui ajoute du poids inutilement.

```
ul.nav li a.navlink {  
  ...  
}
```



```
.navlink {  
  ...  
}
```



```
input[type="submit"] {  
  ...  
}
```



```
[type="submit"] {  
  ...  
}
```



SÉLECTEUR #ID

Éviter d'utiliser le sélecteur d'id, son poids est trop important et difficile à maintenir, éviter également le bazooka !important

```
#nav a {  
  ...  
}
```



```
[id="nav"] a {  
  ...  
}
```



(ou bien)

```
.nav a {  
  ...  
}
```



SÉLECTEURS DE STRUCTURE

Éviter les sélecteurs associés à la structure HTML, **un élément doit pouvoir être ciblé quel que soit son conteneur** ou son emplacement dans le DOM.

```
div > h1 + p {  
  ...  
}
```



```
.intro {  
  ...  
}
```



```
#navigation h2, #sidebar h2 {  
  ...  
}
```



```
.h2-like {  
  ...  
}
```



```
.sidebar .button {  
  ...  
}
```




```
.button-primary {  
  ...  
}
```




STRUCTURE ET APPARENCE

Séparer la structure de l'apparence dans les sélecteurs pour faciliter la **factorisation**.

```
.button {  
  display: inline-block;  
  padding: 1em;  
  background: blue;  
  color: white;  
}
```



```
.button {  
  display: inline-block;  
}  
.button-large {  
  padding: 1em;  
}  
.button-primary {  
  background: blue;  
  color: white;  
}
```



FACTORISER LES PROPRIÉTÉS

Toujours tenter de **rassembler** les propriétés identiques.

```
body::before {  
  content: "";  
  position: absolute;  
  top: 40%;  
  background: #fff;  
}
```

```
body::after {  
  content: "";  
  position: absolute;  
  top: 20%;  
  background: #fff;  
}
```



```
body::before,  
body::after {  
  content: "";  
  position: absolute;  
  top: 40%;  
  background: #fff;  
}
```


```
body::after {  
  top: 20%;  
}
```



SURCHARGE

Éviter d'écraser une règle par une autre.

```
li {  
  visibility: hidden;  
}  
li:first-child {  
  visibility: visible;  
}
```




```
li + li {  
  visibility: hidden;  
}
```



(ou bien)

```
li:not(:first-child) {  
  visibility: hidden;  
}
```



REDONDANCES

Utiliser des **pré-processeurs** (Sass, LESS, Stylus) pour éviter les répétitions de code.

```
li {  
  color: red;  
}  
div {  
  color: #F00;  
}  
p {  
  color: #FF0000;  
}
```



(Sass)

```
$color: #F00;  
  
li {  
  color: $color;  
}  
div {  
  color: $color;  
}  
p {  
  color: $color;  
}
```




RÉUTILISER LES BLOCS

Grouper les éléments par **composants réutilisables**.


```
<div class="module-left"></div>  
<div class="module-right"></div>
```

```
.module-left {  
  overflow: hidden;  
  float: left;  
  margin: 0;  
  background: #fff;  
}  
.module-right {  
  overflow: hidden;  
  float: right;  
  margin: 0;  
  background: #fff;  
}
```



```
<div class="module fl"></div>  
<div class="module fr"></div>
```

```
.module {  
  overflow: hidden;  
  margin: 0;  
  background: #fff;  
}  
.fl {  
  float: left;  
}  
.fr {  
  float: right;  
}
```



NE PAS TROP RÉUTILISER

Se limiter à 4 noms de classes au maximum par élément HTML.

Si l'on pense qu'il en faut davantage, il est temps d'envisager une classe personnalisée.

```
<div class="mod clearfix left inbl w200p pas mb1 lg-mb2 sm-mbn"></div>
```

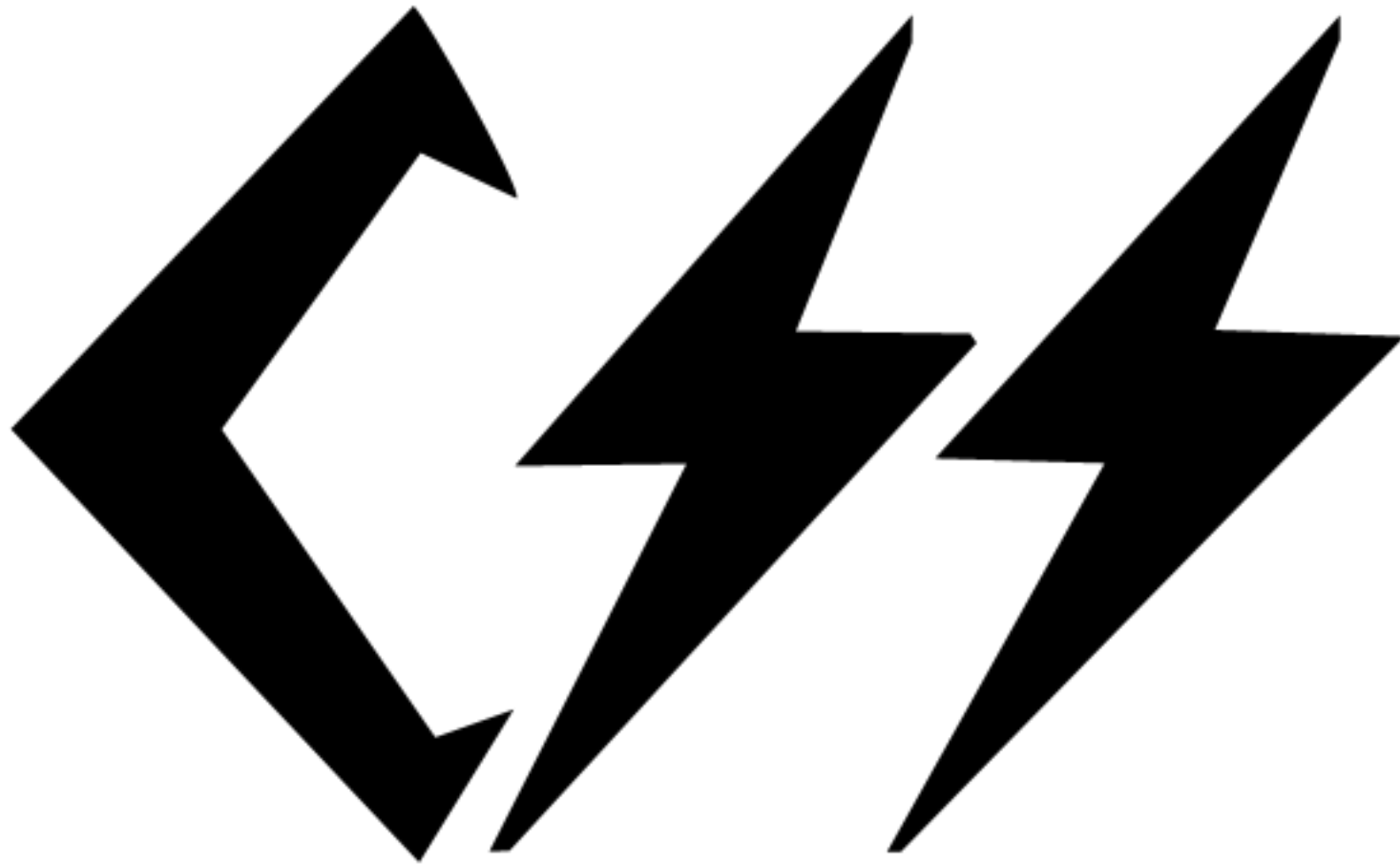


```
<div class="nom-de-classe-spécifique"></div>
```



PERFORMANCES

accélérez vos productions



ANIMATIONS GOURMANDES

Éviter d'animer des propriétés autres que `transform` ou `opacity` , ou alors ajouter la propriété `will-change` et/ou le hack de `translateZ()`.

```
div:hover {  
  margin-left: 100px;  
  transition: .5s;  
}
```



```
div:hover {  
  transform: translateX(100px);  
  transition: .5s;  
}
```



```
div:hover {  
  margin-left: 100px;  
  will-change: margin-left;  
  transition: .5s;  
}
```




@FONT-FACE PERFORMANT

N'imposez pas de chargements aux anciens navigateurs (IE8). **Privilégiez .woff2.**
Conservez l'astuce du #iefix

```
@font-face {  
  font-family: kiwi;  
  src: url("/fonts/kiwi.eot?#iefix") format("embedded  
opentype");  
  src: url("/fonts/kiwi.eot?#iefix") format("embedded  
opentype"),  
    url("/fonts/kiwi.woff2") format("woff2"),  
    url("/fonts/kiwi.woff") format("woff"),  
    url("/fonts/kiwi.ttf") format("truetype"),  
    url("/fonts/kiwi.svg#svgFont") format("svg");  
}
```



```
@font-face {  
  font-family: 'kiwi';  
  src: url('kiwi.woff2?#iefix') format('woff2'),  
    url('kiwi.woff') format('woff');  
}
```




info : <https://twitter.com/kaelig/status/609362210759012353>


PROPRIÉTÉS RACCOURCIES

Préférer les propriétés **raccourcies**.

```
div {  
  top: 50%;  
  margin-top: -10px;  
  flex-grow: 1;  
  flex-basis: 0;  
  padding-top: 5px;  
  padding-right: 10px;  
  padding-bottom: 20px;  
  padding-left: 10px;  
}
```



```
div {  
  top: calc(50% - 10px);  
  flex: 1;  
  padding: 5px 10px 20px;  
}
```



UNITÉS

L'unité est inutile si la valeur est nulle. Ne pas donner d'unité à `line-height`.

```
div {  
  margin: 0px;  
  font-size: 0.9rem;  
  line-height: 2em;  
  border: none;  
}
```



```
div {  
  margin: 0;  
  font-size: .9rem;  
  line-height: 2;  
  border: 0;  
}
```



PRÉFIXES VENDEURS

Automatiser la gestion des préfixes à l'aide de **Autoprefixer**, ne pas le faire à la main et ne pas utiliser un mixin Sass/LESS pour cela.

```
div {  
  transform: scale(2);  
  -webkit-transform: scale(2);  
  -moz-transform: scale(2);  
  -ms-transform: scale(2);  
  transition: 1s;  
  -webkit-transition: 1s;  
  -moz-transition: 1s;  
  -ms-transition: 1s;  
}
```



```
div {  
  transform: scale(2);  
  transition: 1s;  
}
```

(Autoprefixer)



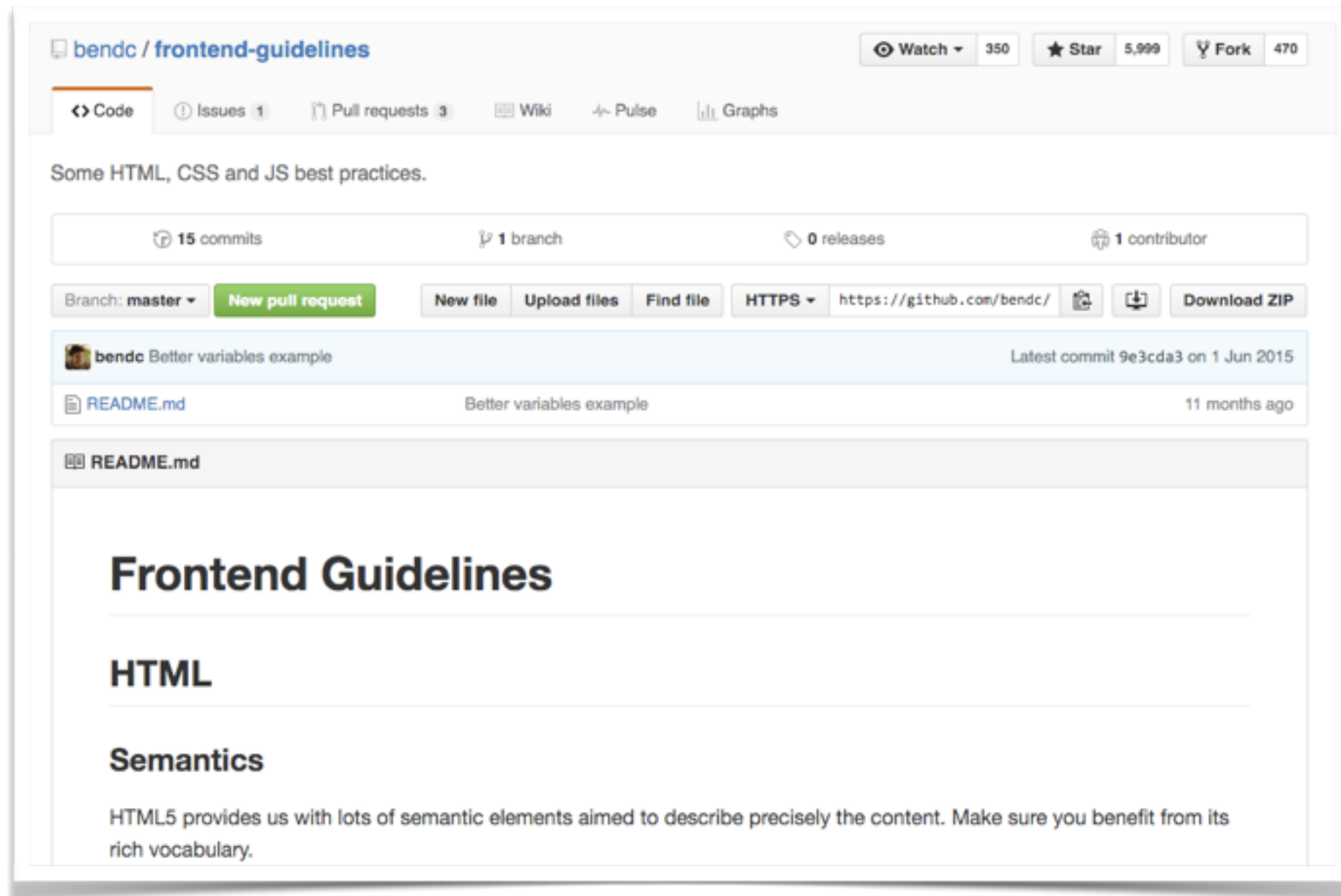
```
div {  
  -webkit-transform: scale(2);  
  transform: scale(2);  
  transition: 1s;  
}
```



Autoprefixer : <https://autoprefixer.github.io/>

FRONT-END GUIDELINES

plein de bonnes pratiques



source : <https://github.com/bendc/frontend-guidelines>

À VOUS DE JOUER

créez (et maintenez) VOTRE convention !

(1)

Charte Alsacréations

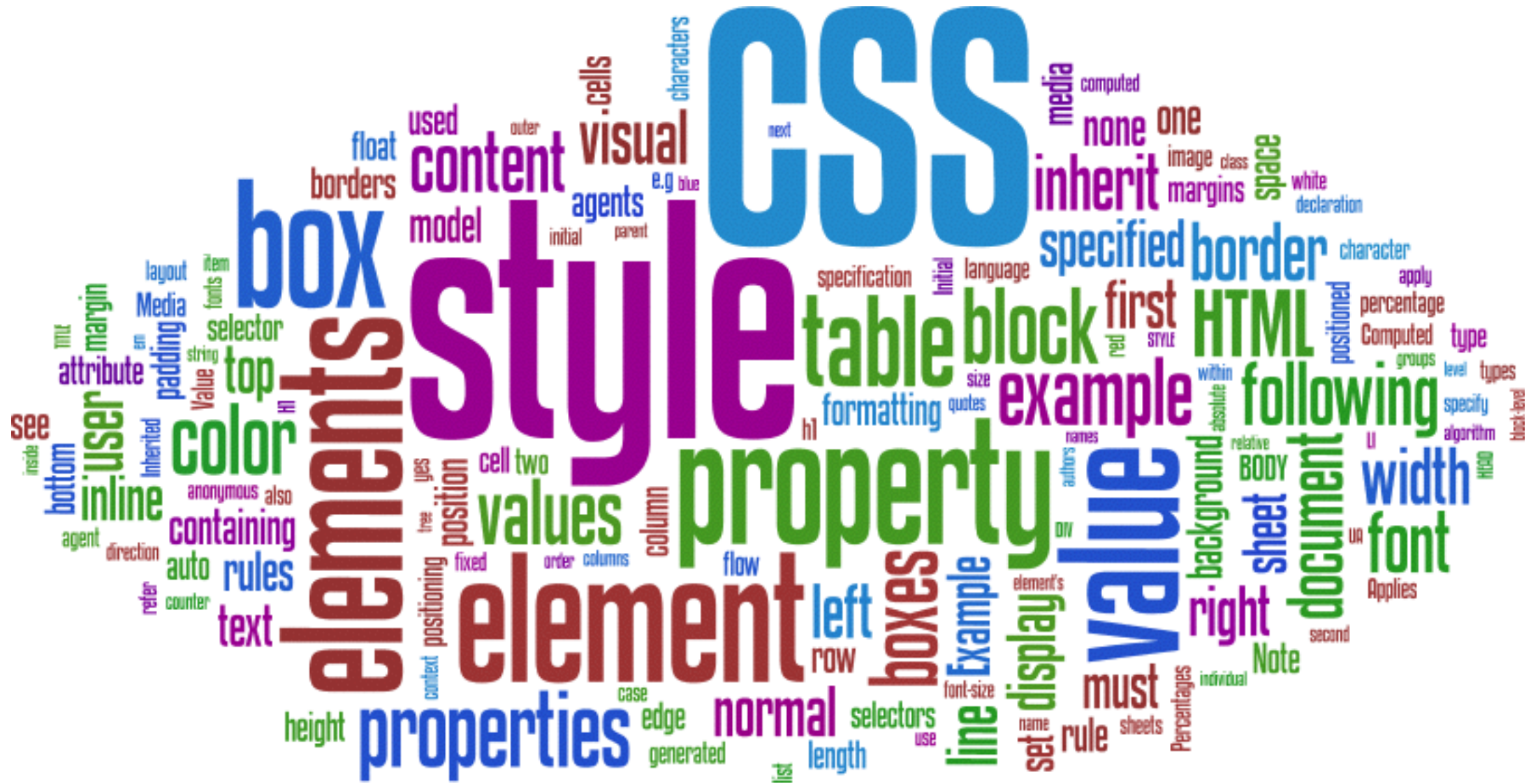
Cette présente charte a pour objectif d'uniformiser et rationaliser les processus, de faciliter la production de documents au sein d'une équipe et de disposer d'outils réutilisables.

Généralités

- "Alsacréations" a toujours un "s" à la fin. Même quand... non. Même pas.
- L'encodage des fichiers et des bases de données doit se faire en UTF-8 (sans BOM)
- Les indentations se font à l'aide de la touche Tab et non avec des espaces
- Les liens absolus ne doivent pas faire apparaître le protocole (par exemple href="//www.alsacreations.fr/" et non href="<http://www.alsacreations.fr/>")
- Chaque étape dans la tâche d'intégration doit faire l'objet de tests navigateurs sur l'ensemble des navigateurs récents, ainsi que leurs versions majeures précédentes (n-1)
- Choisir des noms en anglais prioritairement (classes, fichiers, images, etc.)
- Opter pour un nommage en caractères minuscules (lowercase) uniquement
- Séparer les noms des fichiers, des images des classes et id CSS par des tirets (.slide-info, styles-ie.css, jquery-1.8.min.css)

ALLER PLUS LOIN

quelques bonnes ressources



LES CONVENTIONS DES AUTRES

- [Google HTML / CSS style guide](#)
- [GitHub CSS styleguide](#)
- [WordPress CSS coding standards](#)
- [WordPress HTML coding standards](#)
- [Idiomatic CSS](#)
- [Trello CSS styleguide](#)



compil : css-tricks.com/css-style-guides/

LES MÉTHODOLOGIES

- OOCSS
- BEM
- SMACSS
- Atomic CSS
- ITCSS



OOCSS

à la recherche de « patterns visuels »

Le concept de OOCSS est de repérer des « objets CSS », c'est-à-dire des « patterns visuels » qui se répètent, et de définir ainsi des classes réutilisables.



Nicole Sullivan

OOCSS met en avant deux principes :

1. Le principe de séparation de la structure et de l'apparence ;
2. Le principe de séparation du conteneur et du contenu.

OOCSS

1) séparation entre structure et apparence

```
.button {  
  display: inline-block;  
  padding: 1em;  
  background: blue;  
  color: white;  
}
```



```
.button {  
  display: inline-block;  
}  
.button-large {  
  padding: 1em;  
}  
.button-primary {  
  background: blue;  
  color: white;  
}
```



OOCSS

2) séparation entre conteneur et contenu

```
.sidebar .button {  
  ...  
}
```



```
.button-primary {  
  ...  
}
```



BEM

« block - element - modifier »

1. **Block**
entité indépendante et autonome
2. **Element**
partie d'un Block
3. **Modifier**
variante d'un Block ou Element

(en théorie)

```
.block { }  
.block__element { }  
.block__element--modifier { }
```

(en pratique)

```
.nav { }  
.nav__link { }  
.nav__link--active { }
```



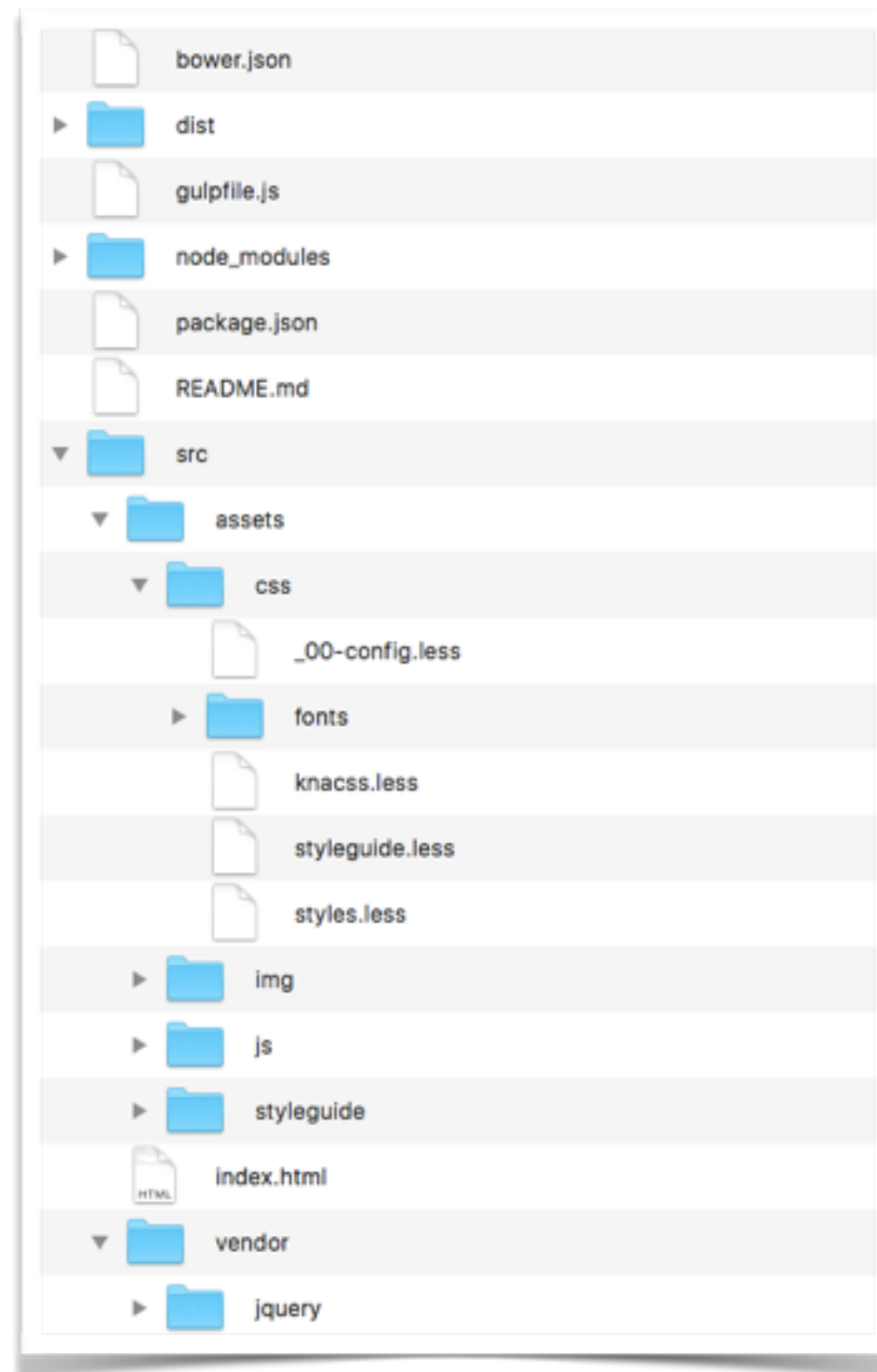
BEM

principes généraux

- 1 Les blocs et les éléments doivent chacun avoir un nom unique, lequel sera utilisé comme classe CSS
- 2 Les sélecteurs CSS ne doivent pas utiliser les noms des éléments HTML (pas de `nav`)
- 3 Les cascades dans les sélecteurs CSS devraient être évitées (pas de `.menu .list`)

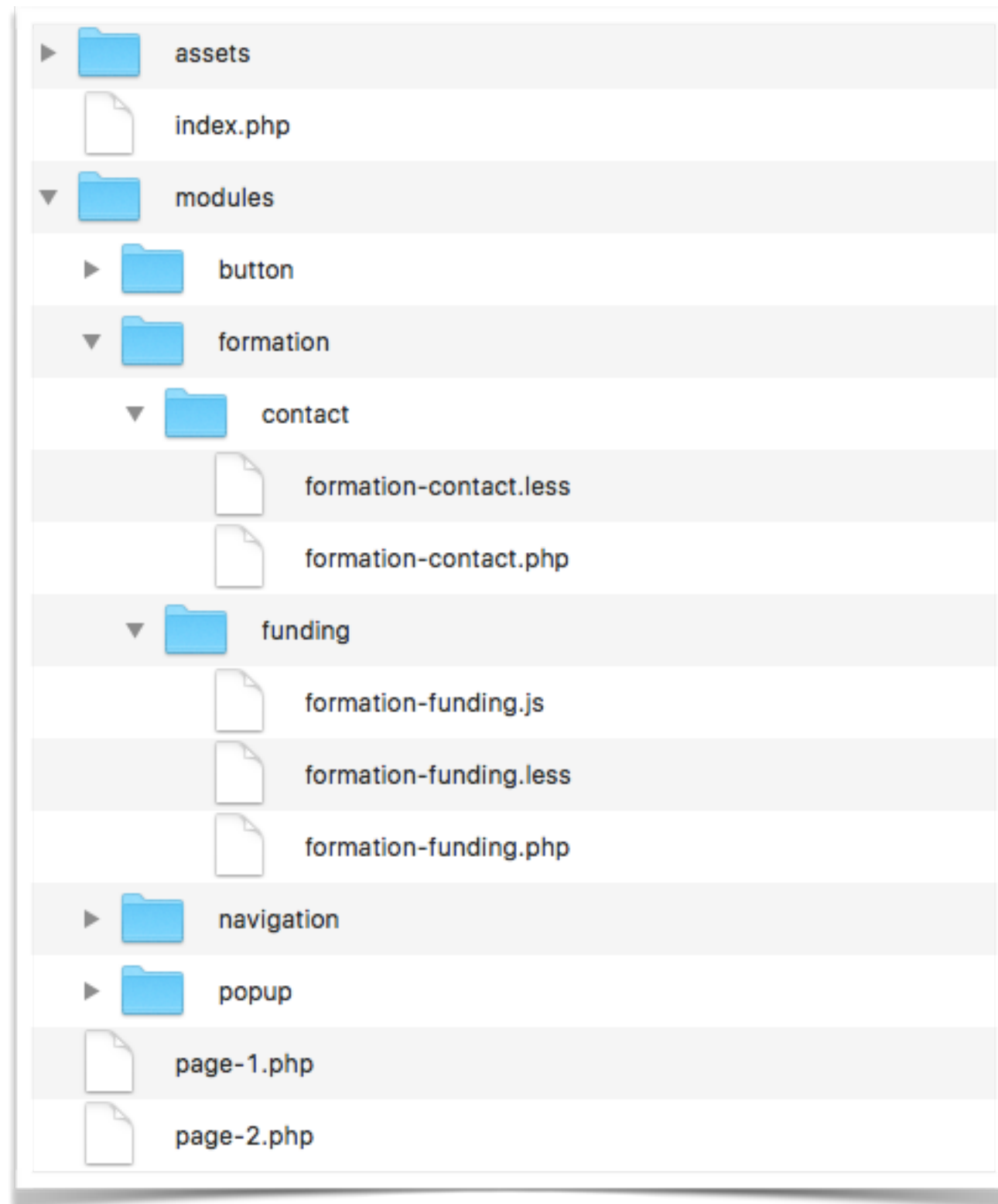
ARCHITECTURE CLASSIQUE

dite « traditionnelle »



ARCHITECTURE « À LA BEM »

complètement modulaire

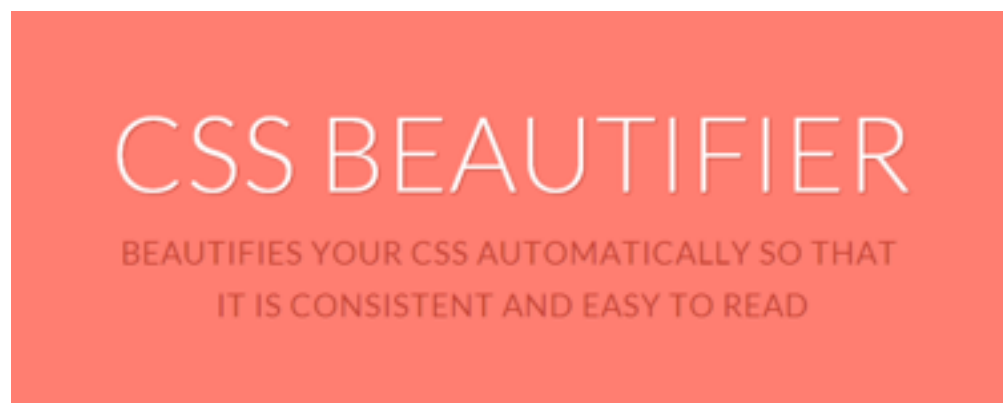


LES OUTILS

- [CSSLint](#)
- [CSScomb](#)
- [Beautify](#)
- [Autoprefixer](#)

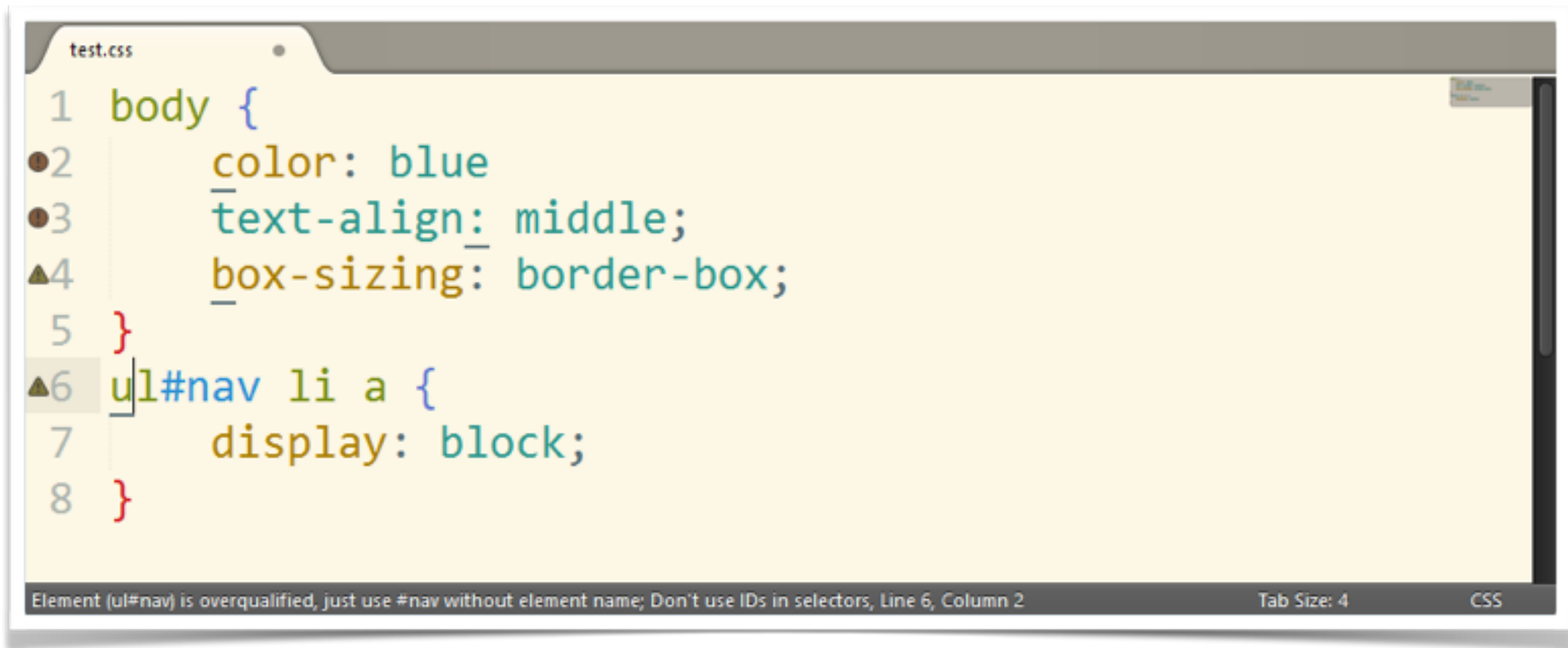


CSS LINT



CSSLINT

validation ***et*** bons conseils



The screenshot shows a code editor window titled 'test.css' with the following CSS code:

```
1 body {  
2     color: blue  
3     text-align: middle;  
4     box-sizing: border-box;  
5 }  
6 ul#nav li a {  
7     display: block;  
8 }
```

On line 6, the selector 'ul#nav' is highlighted with a green squiggly line, indicating a validation error. The error message at the bottom of the editor reads: 'Element (ul#nav) is overqualified, just use #nav without element name; Don't use IDs in selectors, Line 6, Column 2'. The status bar at the bottom also shows 'Tab Size: 4' and 'CSS'.

plugins éditeurs : <https://github.com/CSSLint/csslint/wiki/IDE-integration>

CSSCOMB

réordonner son CSS

```
1 ▼ body {  
2   background: hotpink;  
3   display: flex;  
4   color: maroon;  
5   margin: 0;  
6 }  
7  
8 ▼ div {  
9   border: 5px solid;  
10  position: absolute;  
11  color: green;  
12  z-index: 1337;  
13 }  
14  
15 ▼ @media (min-width: 640px) {  
16 ▼ p {  
17   box-shadow: 0 0 0 pink;  
18   float: left;  
19   color: black;  
20 }  
21 }
```

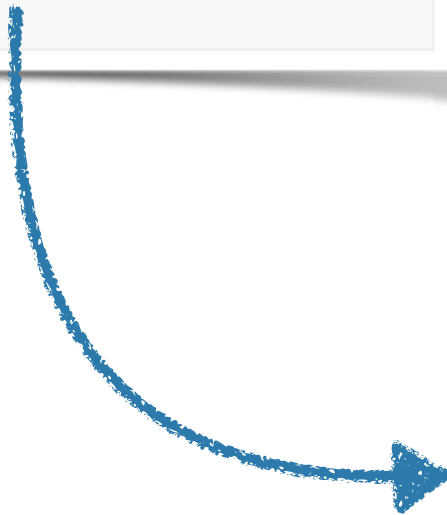


```
1 ▼ body {  
2   display: flex;  
3   margin: 0;  
4   background: hotpink;  
5   color: maroon;  
6 }  
7  
8 ▼ div {  
9   position: absolute;  
10  z-index: 1337;  
11  border: 5px solid;  
12  color: green;  
13 }  
14  
15 ▼ @media (min-width: 640px) {  
16 ▼ p {  
17   float: left;  
18   box-shadow: 0 0 0 pink;  
19   color: black;  
20 }  
21 }
```


CSS BEAUTIFY

ré-indenter et rendre lisible CSS

```
1 ▼ body {  
2   | background: hotpink; display: flex;  
3   | | | color: maroon; margin: 0;  
4   | }  
5 ▼ div {  
6   | border: 5px solid;  
7   | position: absolute; color: green; z-index: 1337;  
8 ▼ @media (min-width: 640px) {  
9   | p { box-shadow: 0 0 0 pink; float: left;  
10  | | | color: black; }  
11  | }
```



```
1 ▼ body {  
2   | background: hotpink;  
3   | display: flex;  
4   | color: maroon;  
5   | margin: 0;  
6   | }  
7  
8 ▼ div {  
9   | border: 5px solid;  
10  | position: absolute;  
11  | color: green;  
12  | z-index: 1337;  
13  | }  
14  
15 ▼ @media (min-width: 640px) {  
16 ▼ | p {  
17   | | box-shadow: 0 0 0 pink;  
18   | | float: left;  
19   | | color: black;  
20   | }  
21  }
```

plugin Atom : <https://atom.io/packages/atom-beautify>

AUTOPREFIXER

ajout automatique des préfixes CSS3

```
1 ▼ body {  
2   display: flex;  
3   order: 1;  
4   transition: .5s;  
5   hyphens: auto;  
6 }
```



```
1 ▼ body {  
2   display: -webkit-box;  
3   display: -webkit-flex;  
4   display: -ms-flexbox;  
5   display: flex;  
6   -webkit-box-ordinal-group: 2;  
7   -webkit-order: 1;  
8   -ms-flex-order: 1;  
9   order: 1;  
10  -webkit-transition: .5s;  
11  transition: .5s;  
12  -webkit-hyphens: auto;  
13  -moz-hyphens: auto;  
14  -ms-hyphens: auto;  
15  hyphens: auto;  
16 }
```

LES PLUGINS INDISPENSABLES

- **EMMET** : raccourcis clavier ++ (Emmet, c'est la vie)
- **HTMLhint** : affiche les erreurs de validation
- **CSSlint** : affiche les erreurs de CSS et les conseils "OOCSS"
- **JShint / JSLint** : vérification de syntaxe JavaScript
- **Beautify** : ré-indentation, ré-agencement des fichiers JS, HTML et CSS
- **Autoprefixer** : ajout de préfixes automatiques
- **Minifier** : Minifie CSS (et JavaScript) dans un fichier *.min.css
- **CSScomb** : réordonne les propriétés CSS dans leur ordre d'importance
- etc.