MÉMO • HTML/CSS

La base

Le html utilise des balises pour délimiter ou créer des éléments

<balise>...</balise> (pour les balises par paire ex : ...)
ou <balise /> ou <balise> (pour les autres ex : <hr>)
<balise attribut="valeur">...</balise>
<balise attribut="valeur"> (ex : <meta charset="UTF-8">)
➔ pas de guillemets typographiques “” « »
➔ on n’invente pas ses propres balises

html5

Un document HTML5 se structure ainsi (!+tab avec Emmet):

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Le titre de la page *</title></head>
<body>
  <p>Bonne nouvelle...</p><!-- commentaire -->
</body>
</html>
```

* Lisible sur l’onglet et dans les moteurs de recherches !

les balises de titrage

h1,h2,h3,h4,h5,h6 (pas de h7, h8...)

les premières balises à connaitre

p : paragraphe
ul,ol,li : listes et item
strong : important (gras par défaut)
em : emphase (italique par défaut)
br : ¬ retour à la ligne forcé
div : bloc (regroupe/divise la page/structure)
span : conteneur de texte
nav : bloc pour liens
article : bloc pour article
header : entête de page ou article...
footer : pied de page ou article...

faire un lien

```
<a href="http://exemple.fr">texte du lien</a>
```

insérer une image

```
<img src="image.jpg" alt="texte alternatif">
```
➔ src pour source (attention à la coquille « scr »)
```
<a href="http://exemple.fr"><img src="image.jpg" alt="texte
alternatif"></a><!-- image dans un lien-->
```

insérer une feuille de style

```
<link rel="stylesheet" href="style.css">
```
(link:css+tab avec Emmet)

la class et l’id

L'id est **unique** dans la page HTML, la class est réutilisable.
Un seul id par élément, une ou plusieurs class par élément.

```
<div id="post-12" class="post sticky"> (1 id et 2 class)
  <p class="date">...</p>
</div>
```

le CSS

```
nom-de-La-balise, p, div, ...
.nom-de-la-class, .header, .header-title, .main,
#id-de-l-élement, #top, #menu,
sélecteur-père > enfants,
sélecteur-ancestre héritiers{
  propriété-1 : valeur;
  propriété-2 : valeur;
}
```

propriétés à connaître

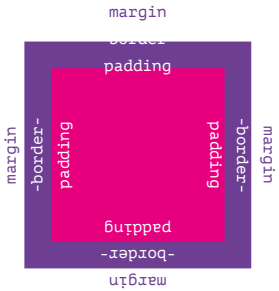
```
| = ou, ne pas taper ce signe
display : block | inline-block | grid...
width : 100% | 960px | 100vw | 100vh...
height : 100% | 250px | 100vh...

font-family : Arial, sans-serif; ...
font-size : 16px | 1em | 1rem | 2vw | clamp(16px,2vw,21px) ...
font-weight : 100 | 700 | light | bold...
line-height : 1.5 | 1 | 24px | 1.5em
text-transform : uppercase;
text-decoration : none;

color : #fff | white | rgba(255,255,255,0.5) ...
background : #f60 url(“a.jpg”) no-repeat;

grid-template-columns : repeat(3, 1fr) | 1fr 1fr | 20px 300px...
grid-template-rows : 200px 1fr 200px | ...
gap: 24px 24px | 1px ...
grid-column : 1/3 | span 2 | 1/-1 ...
grid-row : 1/3 | span 2 | 1/-1 ...
align-items : start | end | baseline | center...
align-self : start | end | baseline | center...
align-content : start | end | stretch | center...

justify-items : start | end | stretch | center...
justify-self : start | end | stretch | center...
justify-content : start | end | stretch | center÷
```




```
.exemple{
  margin: 10px 12px 14px 18px;
  border: 1px solid #f60;
  padding: 20px;
}
```

margin/padding
1 valeur : les 4 marges identiques
2 : haut = bas & droite = gauche
3 : haut, droite = gauche & bas
4 : haut, droite, bas et gauche

mise en page & display

(HTML avec Emmet : .parent>(.item*3)+tab)

```
.item{
  display:block;
}
```

item
item
item

```
.item{
  display:inline-block; /* (ou inline) */
}
```

item item item

Positionnement avec flex

```
.parent{
  display:flex;
  justify-content:space-between;
}
```

item item item

grille avec Grid

```
.parent{
  display: grid; /* LA grille */
  /* 3 colonnes */
  grid-template-columns: 1fr 1fr 1fr | repeat(3, 1fr);
  gap: 24px; /* goutière */
  justify-items : center;
}
```
➔ attention aux virgules et espaces

```
.item-a{
  grid-column: 1 / 3; /* place sur deux colonnes */
  grid-row: 2 / 5; /* de la 2e ligne jusqu'à la 5e */
  justify-self : end;
}
```
demo

pratiques courantes

box-sizing

Recommandation pour le calcul simplifié de la taille des blocs.

```
html {
  box-sizing: border-box;
}
*,*::before,*::after {
  box-sizing: inherit; min-width: 0; min-height: 0;
}
```

image et max-width

Empêche toutes les images de dépasser du navigateur

```
img{
  max-width: 100%; height: auto;
  /* et on n'ajoute rien sinon c'est pas gérable */
}
```

reset rapide

Pour un petit projet ou une démo.

```
*{margin: 0; padding: 0;}
```

➧ Pour un projet plus important « normalize.css ou reboot.css ».

la meta viewport (responsive)

Pour que le site s'adapte correctement aux écrans.

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

mobile-first et media queries

Pensez vos projets pour mobile dès le début, pour la maquette, la navigation, le tactile et codez-le d'abord dans sa version mobile !. demo

```
.article p { color: #333; /* régle pour tous les écrans */ }

@media (min-width:640px){
  .article p {
    columns: 2; /* pour exemple */
  }
}

@media (min-width:960px){
  .article p { columns: 3; /* pour exemple */}
}
```

margin : auto

Quand un élément a une taille définie (width|max-width), margin : auto permet de centrer celui-ci en horizontal et même en vertical si son père est en display: flex.| grid (forcer la hauteur si besoin) demo

```
.wrap{
  max-width: 480px;
  margin: auto;
}
body{
  min-height:100vh;
}
```

@font-face

Pour utiliser une fonte web dans vos pages à partir du fichier woff et/ou woff2 :s

```
@font-face {
  font-family: Faune;
  src: url(fonts/woff2/Faune_Thin.woff2) format("woff2"),
       url(fonts/woff/Faune_Thin.woff) format("woff");
  font-weight: 100; /* ultra-light */
  font-style: normal
}

@font-face {
  font-family: Faune; /* toujours le nom de la famille */
  src: url(fonts/woff2/Faune_Bold.woff2) format("woff2"),
       url(fonts/woff/Faune_Bold.woff) format("woff");
  font-weight: 700; /* bold */
  font-style: normal
}
```

article plus complet

grid responsive

Des combinaisons rapides permettent de faire des grilles responsives :

```
/* Si on peut on ajoute une colonne à droite de 320px */
grid-template-columns: repeat(auto-fill,320px);

/* Colonne fluide */
grid-template-columns: repeat(auto-fill,minmax(320px,1fr));

/* Colonne fluide et suppression de colonne vide */
grid-template-columns: repeat(auto-fit,minmax(320px,1fr));
```

Noms de class fréquentes

Même si les noms sont libres, bien nommer ses class apporte un gain de lisibilité et de temps, voici quelques exemples pour vous inspirer.

`.wrap (pour emballer la page), .header, .header-title, .navbar ou .header-nav, .pagination, .is-hidden, .home, .page, .post, .post-title, .footer, .footer-nav`

En privilégiant les class dans le fichier CSS, on verra souvent des class avec le nom de la balise comme <header class="header">.

la meta description (SEO)

Pour que le site ait un texte optimisé sur les liens des moteurs de recherches.

```
<meta name="descrition" content="Résume mon site/page en 150
signes">
```

Raccourcis clavier

```
Pomme ⌘ + Shift ⇧ + r (rafraîchir la page sans cache)
Pomme ⌘ + Alt ⌥ + i (ouvrir l'inspecteur)
ctrl + Tab ⇥ (bascule entre les onglets)
Pomme ⌘ + Tab ⇥ (bascule entre applications)
```

les outils

VS Code (et penser à ouvrir le dossier du projet !)
Extension Beautify, LiveServer...

Sites indispensables

https://developer.mozilla.org/fr/Apprendre
https://www.alsacreations.com/
https://www.grafikart.fr/

https://css-tricks.com/
https://stackoverflow.com/
https://codepen.io/

https://cssgrid.io/
https://labs.jensimmons.com/
https://www.youtube.com/layoutland
https://gridbyexample.com/