HTML-CSS
MISE EN PAGE DU SITE

SOMMAIRE

Version décembre 2019

MISE EN PAGE DU SITE

Le cours s'inspire du cours public Open Class Room de 2016.
https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3
Et du tutorial w3schools :
https://www.w3schools.com/html/default.asp

- ***Webographie :***

  http://www.w3schools.com : site de référence HTML, CSS et d'autres
  http://www.w3schools.com/tags/default.asp : toutes les balises
  https://www.w3.org : le site de référence du W3C
  http://caniuse.com: pour savoir quel navigateur gère quoi

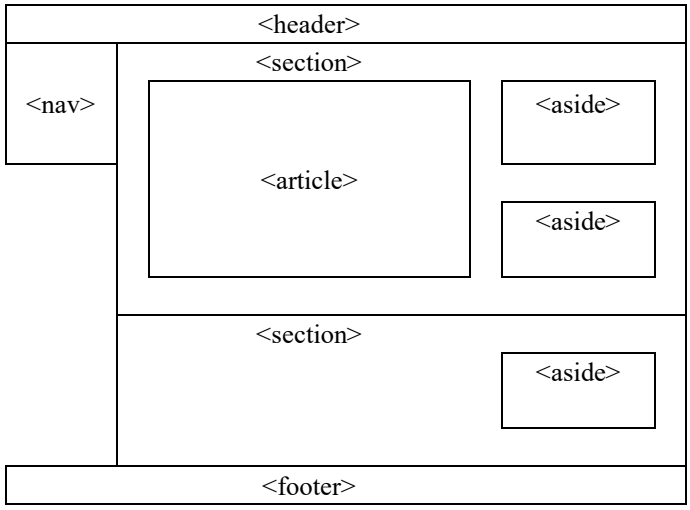
- ***Formation :***

  HTML et CSS chez CodeAcademy
  Make a Web site chez CodeAcademy
  http://pierre-giraud.com
  Tuto basique
  Tuto et astuces Alsacréations - HTML
  Tuto et astuces Alsacréations - CSS

1 : Structurer sa page

Diviser sa page en plusieurs morceaux

Un page HTML peut ressembler à cela :

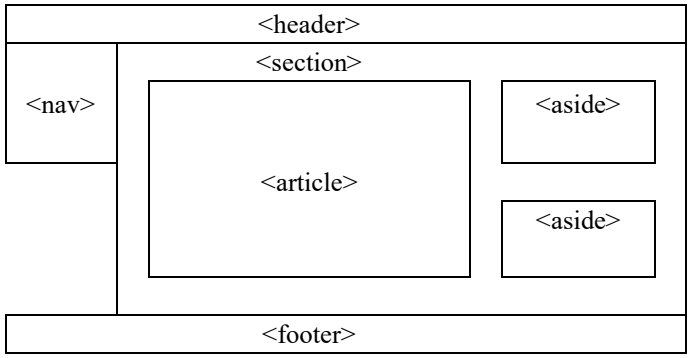


La page est divisée en plusieurs parties :
Le « **header** » : l'en-tête qui se répète sur chaque page.
Le « **nav** » : le menu de navigation
La « **section** » qui contient le contenu de la page. Il peut y en avoir plusieurs.
Les « **article** » à l'intérieur d'une section.
Les « **aside**» (ou « à côté ») qui sont à côté des articles dans la section.
Le « **footer** » : le pied de page qui se répète sur chaque page.

En HTML 4, chacune de ses parties étaient regroupées dans des « avec un id ou une class.
Le HTML5 introduit des balises spécifiquement pour ces parties.

Exemple de code

On part d'une maquette :



On arrive au code HTML, sans mise en page :

```
<header>
      bla bla
</header>

<nav>
      ul et li
</nav>

<section>
      <article>
            bla bla
      </article>

      <aside>
            bla bla
      </aside>

      <aside>
            bla bla
      </aside>
</section>

<footer>
      bla
</footer>
```

A partir de ce HTML, on réfléchit au positionnement.

Exemples visuels

Exemple simple :

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/02_flex/index.html

Exemples plus complexe :

https://www.paris.fr/services-et-infos-pratiques

https://opendata.paris.fr/pages/home/

Les balises de structuration de HTML 5

Principes

Pour structurer la page, on utilise des balises de structuration.

On connaît déjà

-
-

HTML5 introduit de nouvelles balises :

- **<header>**
- **<nav>**
- **<section>**
- **<article>**
- **<aside>**
- **<footer>**

Voir par exemple : https://www.alsacreations.com/article/lire/1376-html5-section-article-nav-header-footer-aside.html

La balise de structuration universelle : le div

http://www.w3schools.com/tags/tag_div.asp

La balise div est une **balise bloc** (cf. paragraphe un peu plus bas). Elle permet de créer les regroupements de balises qu'on souhaite pour pouvoir appliquer du CSS à tout le groupe.

On associera une class ou un id aux pour pouvoir y accéder avec le CSS.

Au bout du compte, un fichier HTML contient de nombreuses balises pour gérer le CSS. Mais attention à la « **divite** » : tendance qui consiste à mettre des div partout ! Il faut essayer d'en mettre le moins possible. Les balises de structuration évite la divite !

La balise de structuration inline : le span

https://www.w3schools.com/tags/tag_span.asp

La balise div est une **balise inline** (cf. paragraphe un peu plus bas). Elle permet de regrouper un ou plusieurs mots pour leur appliquer le CSS qu'on veut.

On associera une class ou un id aux pour pouvoir y accéder avec le CSS.

3 balises sémantiques plutôt uniques : header, footer et nav

Ces trois balises correspondent en général à des éléments qu'on retrouvera sur toutes les pages du site.

- ***<header> : bandeau d'en-tête***

  http://www.w3schools.com/tags/tag_header.asp

  On peut aussi avoir un header dans une section ou dans un article.

- ***<footer> : bandeau du bas (des liens de contacts, de réseaux sociaux, etc.)***

  http://www.w3schools.com/tags/tag_footer.asp

- ***<nav> : navigation à gauche ou sous l'en-tête (avec ul, li, a href…)***

  http://www.w3schools.com/tags/tag_nav.asp

3 balises sémantiques plutôt multiples : section, article, aside

- ***<section> : une section de la page avec un ou des articles***

  un rectangle entre le header, le nav et le footer. Sert à regrouper une thématique. On peut en avoir plusieurs. La section va plutôt contenir les articles et les asides.

  http://www.w3schools.com/tags/tag_section.asp

- ***<article> : un composant de la section***

  un rectangle dans la section, plutôt central, pour un article qui pourrait être repris tel quel sur un autre site. Il y a plutôt un article par section.

  http://www.w3schools.com/tags/tag_article.asp

- ***<aside> : l'à-côté des articles ou des sections***

  une rectangle, plutôt à droite dans la section, pour donner des informations complémentaires à celles de la section. On peut en avoir plusieurs dans la section.

  http://www.w3schools.com/tags/tag_aside.asp

La balise main

https://www.w3.org/TR/html-main-element/#the-main-element :

« La balise <main> représente la **section de contenu principale** du corps d'un document ou d'une application. La section principale du contenu consiste en un contenu directement lié au contenu central d'un document ou d'une fonctionnalité centrale d'une application ou en développant ce sujet. »

En clair, le main, c'est la raison d'être de la page, sa section principale.

Cette balise permet de retrouver facilement le contenu principal de la page, mais **n'est pas utile obligatoirement pour la mise en page**.

Méthodologie : 4 étapes pour une page : HTML, balises de structuration, style, mise en page

1 : contenu HTML simple, sans structure

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/index_sans_semantique.html

On a le simple code HTML.

Dans ce code, on gère de l'indexation syntaxique ET sémantique pour y voir plus clair : le h2 est décalé par rapport au h1 ; le p est décalé par rapport au h1 ; etc.

2 : on ajoute des balises de structuration

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/index_sans_CSS.html

Les balises sémantiques permettent de ne plus faire d'indexation sémantique : le code est plus clair.

Sans le CSS, ça ne change rien à l'apparence de la page.

Il faut prendre l'habitude d'utiliser ces balises de structuration en plus et à la place de la balise div.

Ca facilite le travail de recherche des robots et ça rend le code plus lisible.

3 : on ajoute du CSS pour le style

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/index_avec_CSS.html

On ajoute du CSS pour le style : coloration, marges, interligne, etc.

Les sélecteurs peuvent partir des balises de structuration : header, nav, nav ul li, section aside, etc.

On n'a pas forcément d'id et de classe.

4 : on ajoute du CSS pour la mise en page

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/5-flex/index_avec_CSS_flex.html

On ajoute du CSS la mise en page : ici des flex pour passer le menu à l'horizontal et aside et section à l'horizontal aussi.

Les détails de l'utilisation du flex sont présentés dans ce cours.

2 : Balises block et inline

Principes

Les deux principales catégories de balises sont :

- les balises block (la balise universelle ou générique div)
- les balises inline (balise universelle ou générique span)

Il y en a d'autres : les tableaux, les listes, …
Chaque type correspond à un affichage par défaut.

Les balises block

Quelles balises ?

h1, p, div, header, article, section, etc.

- ***Toutes les balises block :***

  https://www.w3schools.com/html/html_blocks.asp

Caractéristiques

- ***Toute la largeur de la page***

  Un bloc prend toute la largeur de la page

- ***Passage à la ligne***

  Un texte après un <bloc> ou un </bloc> passe à la ligne.

  2 <bloc> ou </bloc> qui se suivent ne font qu'un passage à la ligne

- ***Imbrication de blocs***

  Un bloc peut contenir d'autres blocs

Les paramétrages de structuration

On peut modifier :

- ***la taille des blocs :***

  width (largeur), **height** (hauteur)

- ***la marge des blocs :***

  margin (marge extérieure), **padding** (marge intérieure)

Les balises inline

Quelles balises ?

span, a, img, em, strong, mark, etc.

- ***Toutes les balises inline:***
  https://www.w3schools.com/html/html_blocks.asp

Caractéristiques

- ***Pas de passage à la ligne***
  Un texte après un <inline> ou un </inline> ne passe pas à la ligne.
- ***Que la place nécessaire***
  Une balise inline prend juste la place dont elle a besoin

L’attribut display : block, inline et inline-block

Attribut display

Inline et block sont les valeurs d’un attribut : **display**.

On peut **changer une balise block en balise inline et réciproquement** ou définir une balise **inline-block** qui aura les propriétés combinées d’un inline et d’un block.

https://www.w3schools.com/cssref/tryit.asp?filename=trycss_display

Usages courants du display :

- **display : none** -> pour ne pas afficher une balise
- **display : inline-block** -> pour avoir à la fois un inline et un block
- **display : flex** -> pour gérer la mise en page flex

Tous les usages du display :

https://www.w3schools.com/cssref/pr_class_display.asp

3 : Attributs pour un bloc

width, height, min-width, max-width

Principes

Un bloc s'étale par défaut sur toute la largeur de l'écran et sur toute la hauteur de son texte.

On peut contrôler la largeur et la hauteur.

width, height

Pour définir la largeur et la hauteur.

Les valeurs se définissent en **pixels**, en **em** ou en **pourcentage**.

http://www.w3schools.com/cssref/pr_dim_width.asp

http://www.w3schools.com/cssref/pr_dim_height.asp

- ***Exemples d'utilisation du width et du height***

  https://www.w3schools.com/css/css_dimension.asp

min-width, max-width, min-height, max-height

On peut fixer une taille minimum ou un taille maximum, selon la taille et/ou la résolution de l'écran.

Les valeurs se définissent en **pixels**, en **em** ou en **pourcentage**.

- ***Exemples de min-width***

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/index_avec_CSS.html

  L'encadrement du nav est à 50% et limité à 400px.

```
nav{
    width : 50%;
    min-width : 400px;
}
```

- ***Exemple de min-width à une image :***

  http://www.w3schools.com/tags/att_img_width.asp

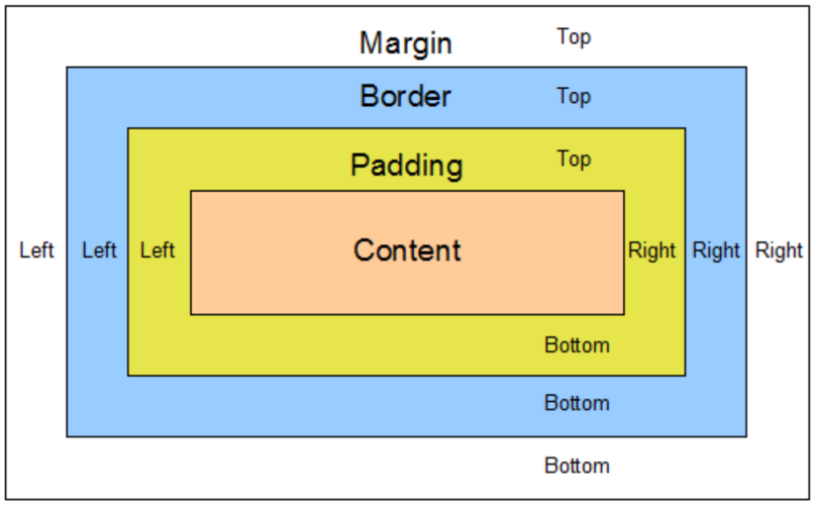
  Dans cet exemple, remplacez le code par :

```
<!DOCTYPE html>
<head><style>
img{
      width:50%;
      min-width:200px;
      border:1px solid black;
}
</style></head>
<html><body>
      <img src="smiley.gif" alt="Smiley face">
</body></html>
```

margin, padding, border

Principes

- **padding** : marges à l'intérieur des bordures.
- **border** : les bordures
- **margin** : marges à l'extérieur des bordures.



Valeurs

- **pixels**, **em**, **pourcentage**.
- **margin : auto** ; les marges sont définies automatiquement.

Width et margin auto : centrer

- width et margin : auto : ça centre le bloc.

Usage détaillée

padding-bottom, -top, -left, -right

margin-bottom, -top, -left, -right

margin : auto ; /* marge extérieure automatique en fonction du contexte */

margin : 3px 0px 1px 4px ; /* top right bottom left : sens des aiguilles d'une montre */

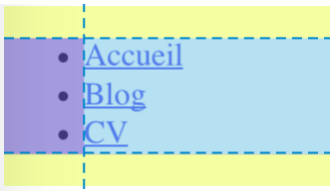
margin : 3px 0px ; /* haut-bas et droite-gauche*/

http://www.w3schools.com/css/css_padding.asp

http://www.w3schools.com/css/css_margin.asp

Exemples

- ***Visuel Firefox : margin en jaune, padding en violet, corps du texte en bleu***



Avec un <ul>, sur Firefox, on peut voir que :

Les margin top et bottom sont présents : en jaune

Il n'y a pas de margin left

Il y a un padding left : en violet.

Si on passe le padding left à 0 : ça va cacher les points de la liste à points.

- ***Exemple en ligne***

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/index_avec_CSS.html

Le texte des différents blocs est contrôlé avec margin et padding.

text-align, line-height, overflow, word-wrap

Justifier, centrer, etc.

text-align : left , right , center , justify

http://www.w3schools.com/cssref/pr_text_text-align.asp

Largeur des interlignes

line-height

http://www.w3schools.com/cssref/pr_dim_line-height.asp

line-height: normal | number | length | initial | inherit;

A noter que le comportement est différent pour un line-height et bloc et pour un line_height d'inline (cas de la première lettre en gros).

Gérer texte qui sort du bloc

Un texte peut être trop grand pour la hauteur du bloc : il peut déborder en bas.

Par défaut un texte trop long déborde de son bloc.

overflow : visible ; pour laisser le texte sortir de son bloc : c'est la situation par défaut.

overflow : scroll ou **auto** ; pour avoir un scroll vertical.

overflow : hidden; pour cacher la fin du texte.

L'attribut **auto** permet de gérer au mieux à l'horizontal et la verticale.

http://www.w3schools.com/cssref/pr_pos_overflow.asp

Gérer un mot qui sort de son bloc : word-wrap

Un mot peut être trop large pour la largeur du bloc (une URL par exemple) : il peut déborder à droite.

Par défaut, le mot trop long déborde de son bloc.

overflow : visible ; pour laisser le mot sortir de son bloc : c'est la situation par défaut.

overflow : scroll ou **auto** ; pour avoir un scroll horizontal.

overflow : hidden; pour cacher la fin du mot.

L'attribut **auto** permet de gérer au mieux à l'horizontal et la verticale.

word-wrap: **break-word** ; pour couper le mot et passer à la ligne.

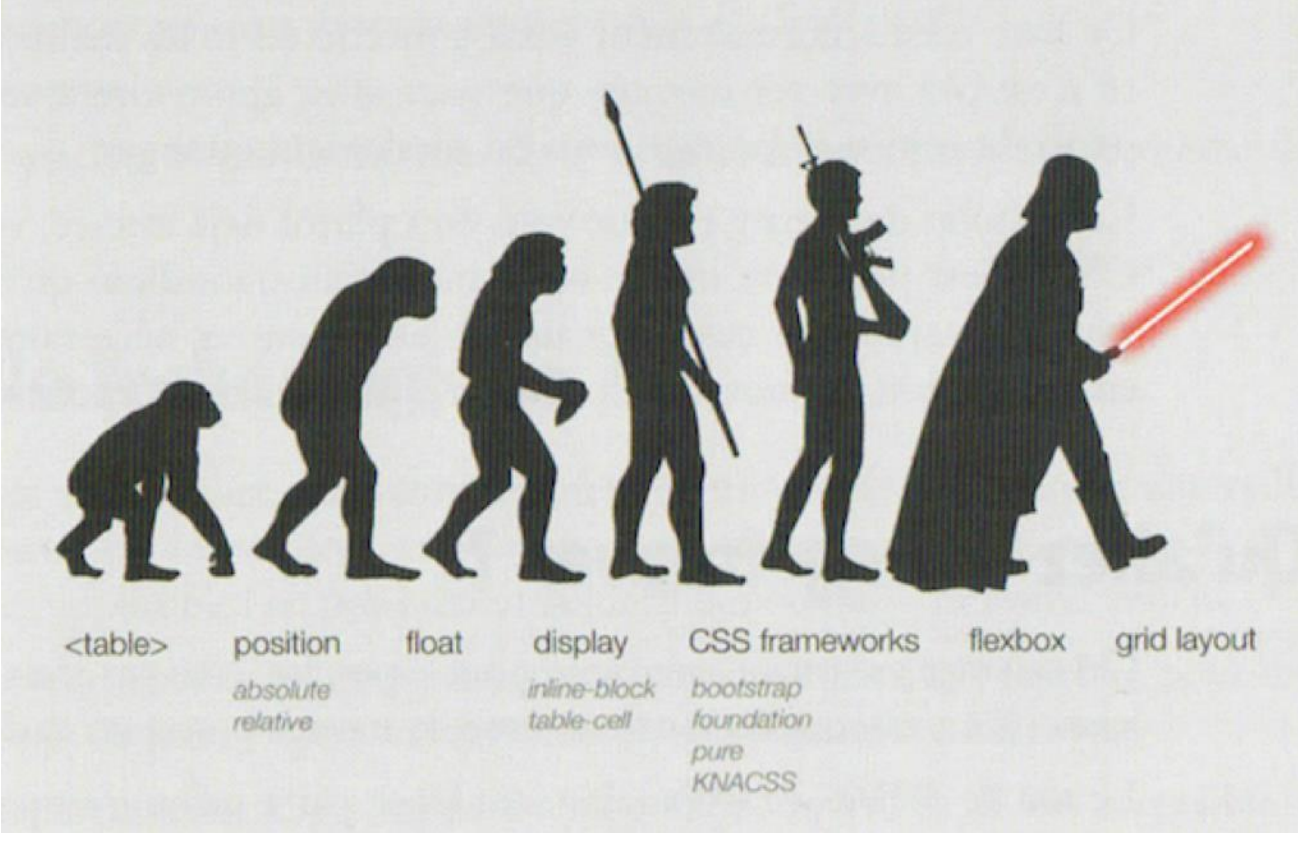
http://www.w3schools.com/cssref/css3_pr_word-wrap.asp

4 : Les 6 grands types de positionnement CSS

Historique

Ce chapitre montre le même exemple mis en page avec les différentes techniques (sauf le Grid Layout)

Le dessin ci-dessous montre l'évolution des techniques de mise en page en HTML.



A noter dans ce schéma (Raphaël Goetter, CSS3 Flexbox, Eyrolles 2017, pVII), que les frameworks CSS comme bootstrap apparaissent avant flexbox et grid layout. Cela montre que bootstrap a été une réponse face aux difficultés de la mise en page en « display-float-position ». Mais qu'avec flexbox et grid layout, mieux vaut revenir au HTML 5 natif.

Introduction à la mise en page en CSS

https://developer.mozilla.org/fr/docs/Apprendre/CSS/CSS_layout/Introduction

1-Grid Layout

- **Généralités sur la mise en page :**
  https://developer.mozilla.org/fr/docs/Apprendre/CSS/CSS_layout/Introduction

- **Propriété CSS display : grid**
- **Référence W3C :** https://drafts.csswg.org/css-grid/
- **Référence Mozilla :** https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grids
- **Principes :** On découpe la page ou une partie en lignes et en colonne : disposition en « trame ».

Grid Layout ou Flexbox :

Le découpage en trame est rigide.

https://www.alsacreations.com/article/lire/1794-flexbox-ou-grid-layout.html

Figure 1 *Exemplary Flex Layout Example*

Figure 2 *Exemplary Grid Layout Example*

Exemples :

Exemple1 et exemple2. : regarder le code HTML qui intègre le style.

On peut modifier les valeurs des grid directement dans l'inspecteur du navigateur pour voir l'effet.

2-flexbox

- **Généralités sur la mise en page :**
  https://developer.mozilla.org/fr/docs/Apprendre/CSS/CSS_layout/Introduction

- **Propriété CSS display : flex**
- **Référence W3C :** https://drafts.csswg.org/css-flexbox/
- **Référence Mozilla :** https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox
- Divise un bloc en sous-blocs à l'horizontale ou à la verticale, sans la rigidité du grid.
- Nouveauté CSS3. Le plus pratique et puissante aujourd'hui. **A utiliser massivement !**
- Les autres méthodes sont devenues moins utiles, mais utiles quand même et toujours utilisées.
  http://www.w3schools.com/css/css3_flexbox.asp

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/5-flex/index_avec_CSS_flex.html

  On ajoute du CSS la mise en page : ici des flex pour passer le menu à l'horizontal et aside et section à l'horizontal aussi.

  Les détails de l'utilisation du flex sont présentés dans ce cours.

3-inline-bloc

- **Propriété CSS display : inline-block**
- Permet de faire d'un bloc un inline ou un inline-bloc. Ainsi, on peut aligner plusieurs blocs à la suite.
- Très utilisé avant flexbox. Moins utile aujourd'hui.

  http://www.w3schools.com/cssref/pr_class_display.asp

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/4-inline-block/index_avec_CSS_inline_block.html

  On ajoute du CSS la mise en page : ici des inline-block pour passer le menu à l'horizontal et aside et section à l'horizontal aussi. Il faut préciser la largeur des colonnes si on veut qu'elles prennent toute la place. Il faut aussi faire « pendre les blocs » avec un vertical-align.

  Les détails de l'utilisation des inline-blocks sont présentés dans ce cours.

4-flottant

- **Propriété CSS float : left**
- Permet qu'un bloc soit entouré du bloc suivant. Du coup, ça met les deux blocs en ligne.
- On peut mettre plusieurs float à la suite et avoir ainsi plusieurs blocs en ligne.
- Très utilisé avant flexbox. Moins utile aujourd'hui. Toujours utile pour qu'un texte entoure une image.

  http://www.w3schools.com/cssref/pr_class_float.asp

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/3-float/index_avec_CSS_float.html

  On ajoute du CSS la mise en page : ici des float sur les li et les articles et aside de la section pour les passer en ligne. Il faut ajouter une balise vide derrière le conteneur de float pour stopper l'effet du float. Il faut préciser la largeur des colonnes si on veut qu'elles prennent toute la place (ce qui manque de précision).

  Les détails de l'utilisation des float sont présentés dans ce cours.

5-Positionnements relatif, absolu et fixe

- **Propriété CSS position : absolute**
- Le positionnement relatif définit un décalage par rapport à une position initiale
- Le positionnement absolu définit des coordonnées x, y dans la page web.
- Avec le positionnement fixe, la page bouge sous l'élément fixé.
- Ce positionnement n'est pas adapté à des tailles d'écran variables.

  http://www.w3schools.com/cssref/pr_class_position.asp

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/2-absolute/index_avec_CSS_absolute.html

  On ajoute du CSS la mise en page : ici des position relative sur les conteneurs, nav et section, et des position relatives sur les contenu à passer en colonnes : li et aside/article. On positionne ensuite absolument les contenus par rapport au haut à gauche du conteneur. Le footer est positionné absolument par rapport au body en bas à gauche.

  Les détails de l'utilisation des position absolute et relative sont présentés dans ce cours.

6 – Positionnement avec une table

- **Balise <table>**
- La balise table permet de mettre des colonnes (td) dans les lignes (tr).
- On peut donc organiser le positionnement en colonne en utilisant une balise <table>.
- C'est la solution la plus contraignante : ça oblige à écrire un code HTML compliqué et ça rend compliqué les évolutions de mise en page.
- C'est **à éviter totalement** sauf dans des situations très spécifiques.

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/1-table/index_avec_CSS_table.html

On crée des tables pour chaque mise en colonne. Une table pour le nav : chaque li est une td. L'ul est le tr conteneur. Une table pour la section : aside et article seront 2 td d'un même tr.

5 : Positionnement flexbox (CSS 3)

https://www.w3.org/TR/css-flexbox-1/

Division d'un bloc en colonnes : { display : flex ; }

Principes

Par défaut, quand un bloc contient des sous-blocs (un <ul> contient des <li>, une contient des <p>), **les sous-blocs se positionnent les uns en dessous des autres.**

Un simple **display : flex ; sur le bloc principal** (le parent) permet de **répartir les sous-blocs en colonnes** (les enfants).

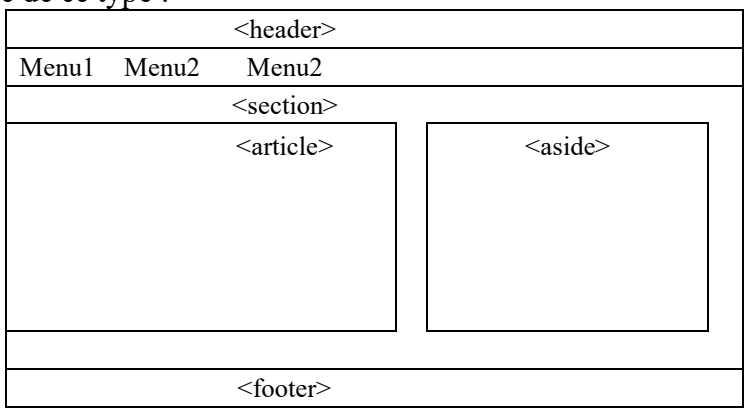
Hauteur et largeur par défaut pour les enfants sont celles nécessaire au remplissage **: les colonnes ne remplissent pas forcément toute la largeur**.

Un **simple flex : 1 ; sur une colonne** fera qu'elle remplira automatiquement toute la largeur restante.

Exemples 1 : page structurée

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/01_flex/index.html

On veut une page de ce type :



- ***Pour le menu***

On met un display flex dans l'ul (nav ul pour cibler).

Ensuite, on fixe la largeur de chaque menu et on supprime le point et le souligné des menus.

```
nav ul{
	display:flex; /* les li sont concernés */
}
li{
	width:15%;
	list-style-type: none; /* pour éviter le point */
}
a{
	text-decoration: none; /* pour éviter le souligné */
}
```

- ***Pour article et aside***

On met uniquement l'aside et l'article en colonne : il suffit d'un display flex dans la section.

Avec « flex : 3 ; » et « flex : 1 ; » on précise le rapport de taille entre article et aside

```
section{
	display:flex; /* aside et article concernés */
}
article{
	flex:3;
}
aside{
	flex:1;
}
```

Exemples 2 : w3school

- ***http://www.w3schools.com/css/css3_flexbox.asp***
    - Pour bien voir l'effet du display : **supprimez le display.**
    - Pour voir l'effet du flex-wrap : **mettez wrap à la place de nowrap.**

Largeur des sous-blocs. Division d'un bloc en colonnes : { flex : 1 ; }

Principes

Pour chaque sous-bloc, la largeur peut être gérée par l'attribut flex : **{ flex : nombre ; }**

Le nombre donne le rapport de largeur (ou de hauteur) entre les sous-blocs.

En mettant 1 à tous les sous-blocs, ils seront tous de la même taille.

Pour chaque sous-blocs, largeur et hauteur peuvent aussi être contrôlées par width et height.

On peut aussi utiliser les margin, margin-top, padding, etc.

Exemple 1 : w3school

http://www.w3schools.com/cssref/css3_pr_flex.asp

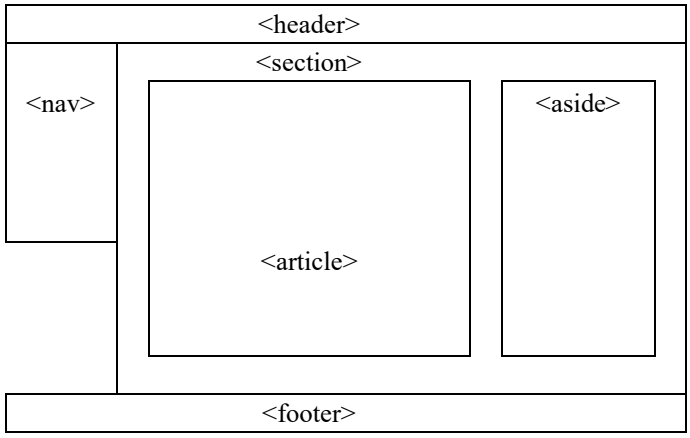
La balise du display: flex; est définie en 300px X 300px.

Chaque balise enfant sera en flex 1, donc en 100px X 300px.

Exemple 2 : page structurée avec menu vertical

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/02_flex/index.html

On reprend l'exemple précédent, mais on met un <nav> vertical :



Il faut créer un div id sur nav et section dans le html :

```
<div id="flex-nav-section">
      <nav>
      …
      <section>
      …
</div>
```

On a alors deux blocs à diviser : <flex-nav-section> et <section>

Ensuite, on précise le flex des sous blocs :

- un flex dans nav et section (rapport de 1 à 5)
- un flex dans article et aside (rapport de 1 à 3)

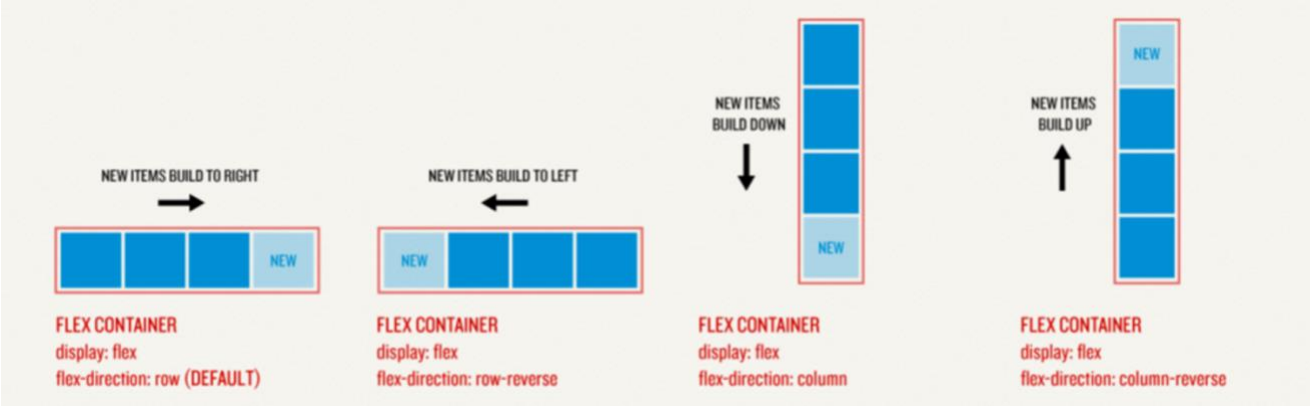
```
#flex-nav-section{
      display:flex; /* nav et section concernés */
}
nav{
      flex: 1;
      height:200px; /* si on veut réduire la taile du nav */
}
section{
      flex:5;
}

section{
      display:flex; /* aside et article concernés */
}
article{
      flex: 3;
}
aside{
      flex: 1;
}
```

Paramétrages du bloc principal : à regarder avec du réseau !

Présentation : 5 propriétés

- **flex-direction** : alignement des sous blocs
- flex-wrap
- justify-content
- align-items
- align-content

flex-direction : alignement des sous-blocs



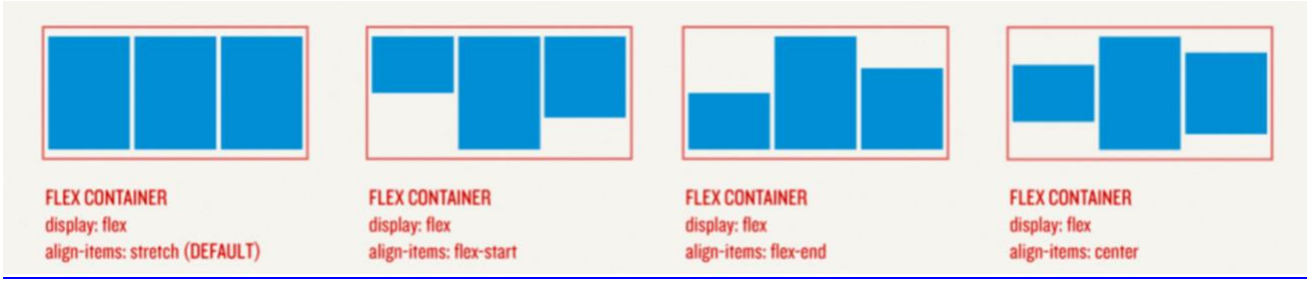
- flex-direction : row : 1 ligne en partant de la gauche. Situation par défaut.
- row-reverse : une ligne en partant de la droite.
- column : 1 colonne à gauche en partant du haut
- column-reverse : 1 colonne à gauche en partant du bas.
  https://www.w3schools.com/cssref/css3_pr_flex-direction.asp

flex-wrap : passer les blocs à la ligne

- **flex-wrap : nowrap :** les sous-blocs s'installent dans une ligne sans passer à ligne. La largeur s'adapte en conséquence. **Situation par défaut.**
- **wrap :** les sous-blocs vont **à la ligne** quand ils n'ont plus de place.
  http://www.w3schools.com/cssref/css3_pr_flex-wrap.asp

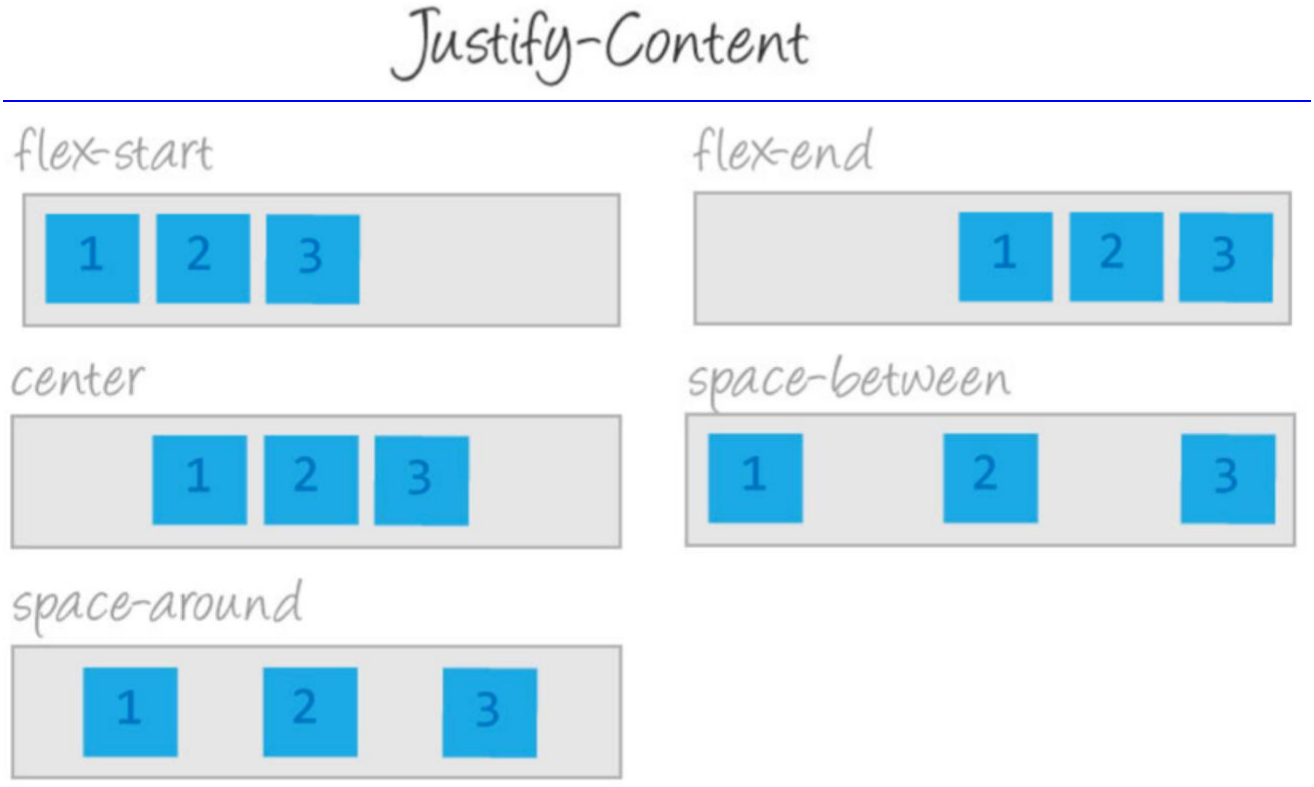
align-items: pour déplacer l'axe des blocs

- **align-items** : **flex-end** ; pour **déplacer l'axe des blocs** en bas (ou à droite) ; **flex-start** : en haut (ou à gauche, par défaut) ; **center** : centré ; **baseline** : ligne de base.
  http://www.w3schools.com/cssref/css3_pr_align-items.asp



justify-content : pour justifier, centrer, étaler

- **justify-content : flex-start** : justifié à gauche ou en haut. **Situation par défaut.**
- **flex-end** : justifier à droite.
- **center** : centré sans espace
- **space-between** : justifié en prenant toute la place de la ligne.
- **space-around** : justifié en prenant toute la place de la ligne avec de l'espace aux extrémités.
  http://www.w3schools.com/cssref/css3_pr_justify-content.asp



align-content : pour gérer les écarts quand les blocs vont à la ligne

- **Align-content :** permet de **gérer les écart entre les lignes quand les blocs vont à la ligne** (ou entre les colonnes si on est en column). On peut donc répartir très facilement 9 blocs de taille identique en 3 fois 3 blocs.
- On peut utiliser toutes les valeurs déjà vues.

➢ ***Exemple des valeurs***

- https://www.w3schools.com/cssref/playit.asp?filename=playcss_align-content&preval=flex-start

➢ ***Exemple des valeurs***

http://www.w3schools.com/cssref/css3_pr_align-content.asp

On peut facilement mettre 9 blocs répartis en 3 fois 3 :

#main div { width: 80px ; height: 80px; }

#main {width: 300px;height: 300px;justify-content:space-between;align-content: space-between;}

Paramétrages des enfants : à regarder avec du réseau !

flex : 1 ;

L'attribut flex impose que le bloc s'étale sur toute la largeur. Le chiffre permet de définir des largeurs relatives entre enfants.

Si on a « enfant1{flex :1 ;} enfants2{flex :2 ;} » alors l'enfant 2 est de fois plus large que l'enfant 1.

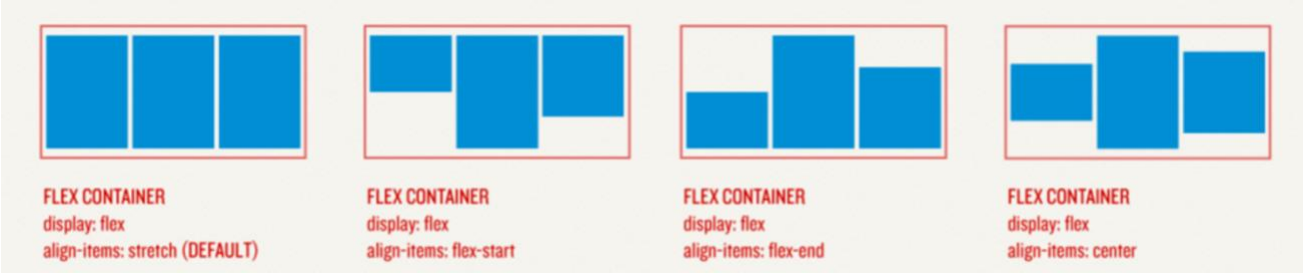
align-self

Equivaut à align-item, mais permet à un item (enfant) particulier de définir son alignement.

➢ ***Rappel align-item :***

- **align-items** : **flex-end** ; pour **déplacer l'axe des blocs** en bas (ou à droite) ; **flex-start** : en haut (ou à gauche, par défaut) ; **center** : centré ; **baseline** : ligne de base.

  http://www.w3schools.com/cssref/css3_pr_align-items.asp



Accès direct au sous-blocs : order :1 ;

Permet de choisir l'ordre d'affichage des blocs.

Si on a « enfant1{order:2 ;} enfants2{order:1 ;} » alors l'enfant 2 sera affiché avant l'enfant 1, quel que soit l'ordre dans le HTML.

Comme ça on peut tout bouger !!!

- ***Exemple***

  http://www.w3schools.com/cssref/css3_pr_order.asp

  Pour bien comprendre l'exemple : ajouter 1, 2, 3, 4 dans les div de l'id main

  Ajouter un « e » à style pour bloquer le style.

  Remettez le style.

Rappel sur la numérotation : p:nth-child(2)

nth-child permet d'accéder au n-ième enfant, ici le 2ème.

Nth-child s'applique aux enfants, on utilisera donc plutôt le sélecteur :

parent enfant:nth-child(2){

}

- ***Exemple***

  http://www.w3schools.com/cssref/sel_nth-child.asp

  Dans cet exemple : mettez les 2 premiers p dans un div et regardez le résultat. Mettez div p p:nth-child(2) et regardez le résultat.

grow, shrink, basis

- ***Présentation***

  La propriété flex peut avoir 3 paramètres : grow, shrink et basis.

  flex : 0 1 auto ; paramètre grow, shrink et basis.

  flex :1 ; veut dire que flex-grow :1 (donc l'enfant prend tout l'espace) et que shrink vaut 1 (valeur par défaut, rétrécissement possible) et que le basis vaut auto (valeur par défaut, la taille est définie par le contenu).

- ***Le bon usage : éviter grow, shrink et basis***

  Le W3C encourage à éviter l'usage des attributs flex-grox, flex-shrink, flex-basis.

- ***flex-grow***

  permet à un enfant d'occuper tout l'espace restant : flex-grow :1 (la valeur par défaut est 0 : pas d'élargissement). Si on a plusieurs flex-grow, la proportion d'occupation sera définie par la valeur donnée (même principe que flex :1 et flex :2).

- ***flex-shrink***

  permet d'empêcher un rétrécicement : flex-shrink :0 (la valeur par défaut est 1, rétrécissement possible). C'est l'inverse de flex-grow.

- ***flex-basis***

  permet de définir la taille selon le couple (grow, shrink).

  On écrit par exemple : flex-basis=50%.

  Pour (0,0), 50% sera la mesure exacte. Pour (1,0), 50% sera la mesure minimum. Pour (1,1), on ne sait pas ce que sera l'effet du flex-basis.

  A noter que les min-width et max-width restent prioritaires.

  Par défaut, la valeur est « auto », ce qui veut dire que la taille de l'élément est définie par son contenu.

Exemple : page complète

http://bliaudet.free.fr/Cours/HTML/33_CSS_TP/TP_superman_flex/superman.html

Dans Firefox, ctrl U pour afficher le HTML, puis on clique sur le fichier CSS. On peut chercher « **flex** » dans le CSS ou « **justify-content** » ou « **align-items** » ou « **align-content** » pour voir tous les usages de flex dans la page.

Exercice

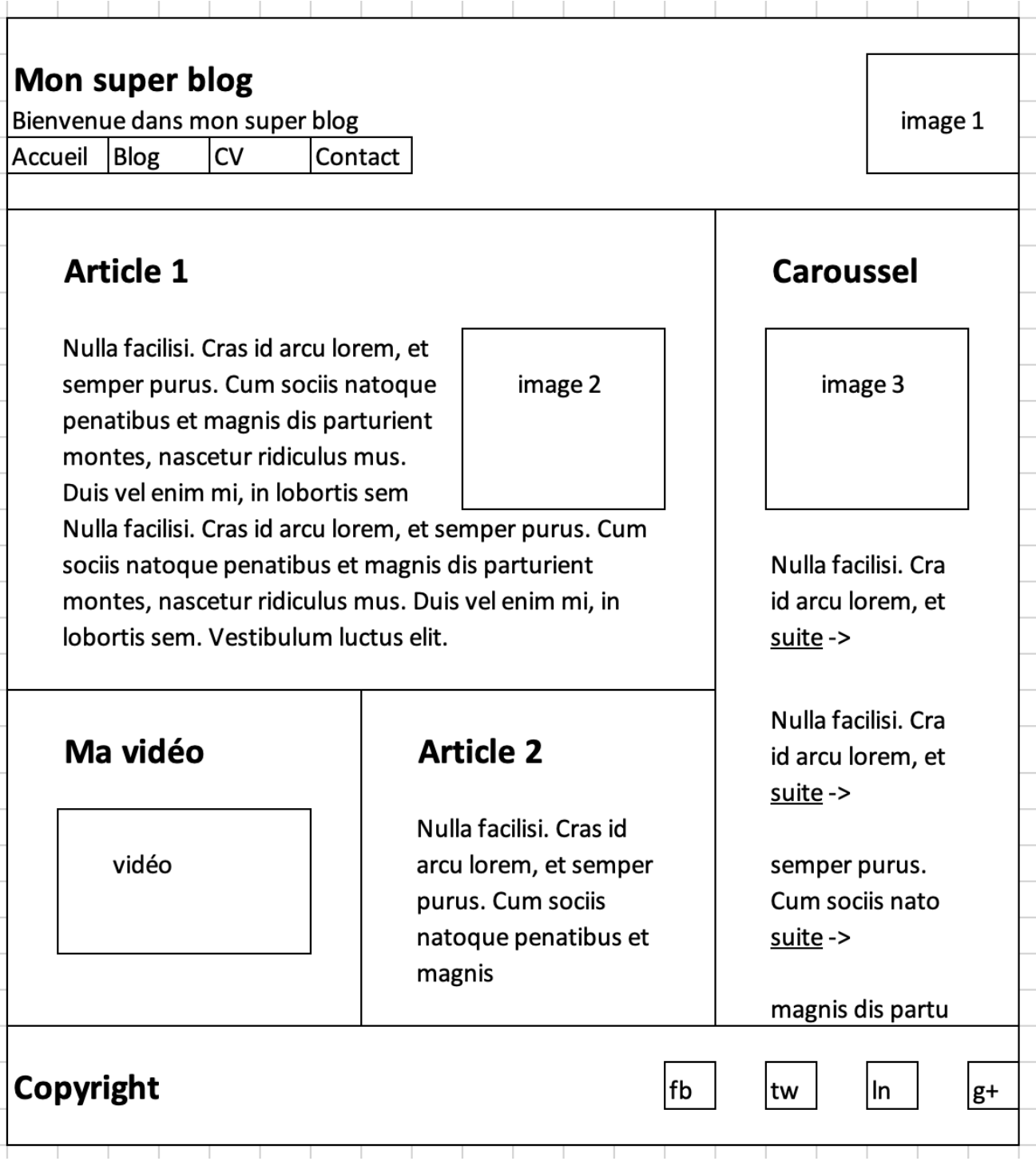
Coder la page ci-dessous :

Dans la colonne Caroussel, il y a 6 petits paragraphes. Faites-en sorte qu'au moins un paragraphe ne soit pas visible et accessible avec un scroll d'ascenseur.

Les « suite » sont des liens hypertexte. Ils amèneront sur une page avec le même en-tête et pied de page, un article complet et un lien pour revenir à la page. Vous pouvez n'avoir qu'un seul article complet.

Le menu (Accueil, Blog, etc.) est constitué de liens hypertexte.

fb, tw, ln et g+ sont des liens hypertexte. Dans l'idéal, ce sont des images.



6 : Positionnement inline-block - display

Principes

Attribut display

On connaît les éléments inline (span, etc.) et les éléments block (p, etc.).

Il existe d'autre type d'élément : list-item (li, etc.), table (pour les tableaux : ils n'ont pas de marges), **inline-block** (à la fois inline et block), inline-table, etc.

On peut **changer le type d'un élément** avec display.

Avec **display : none**, l'élément n'est plus affiché : c'est utile pour l'affichage responsive.

http://www.w3schools.com/cssref/pr_class_display.asp

inline-block

```
nav{
      display : inline-block ;
}
```

Les éléments inline-block s'affichent comme des inlines (l'un à la suite de l'autre) mais peuvent avoir des dimensions comme des blocks (des width particulièrement).

- **inline** : ils s'affichent les uns derrière les autres sur la **ligne de base (baseline).**
- **block** : on peut paramétrer leur dimensions, bordures, marges, etc.
- **Attention**, si le contenu est trop grand, pour avoir un inline, **il faut une largeur (un width)**. Sinon, c'est un bloc qui prend toute la largeur.

vertical-align

```
nav{
      display : inline-block ;
      vertical-align : top ;
}
```

Le vertical-align contrôle l'alignement vertical.

Un bloc peut être posé sur sa ligne du texte (**baseline**, valeur par défaut ou **bottom** ou **text-bottom**) ou comme « pendu au fil du texte » (top ou text-top)

http://www.w3schools.com/cssref/pr_pos_vertical-align.asp

http://www.w3schools.com/css/css_align.asp

display : none

permet de ne pas afficher une balise.

C'est utile associé par exemple à un :hover qui fait l'affichage. Et surtout pour l'affichage responsive.

```
.madiv{
      display : none;
}
```

Exemple 1 : menu horizontal

Objectifs

Solution propre :

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/31_menu_inline/index.html

On met un display:inline-block; sur le li{}

On style le a{} correctement

Et c'est bon ! C'est une solution propre et simple !

CSS complet :

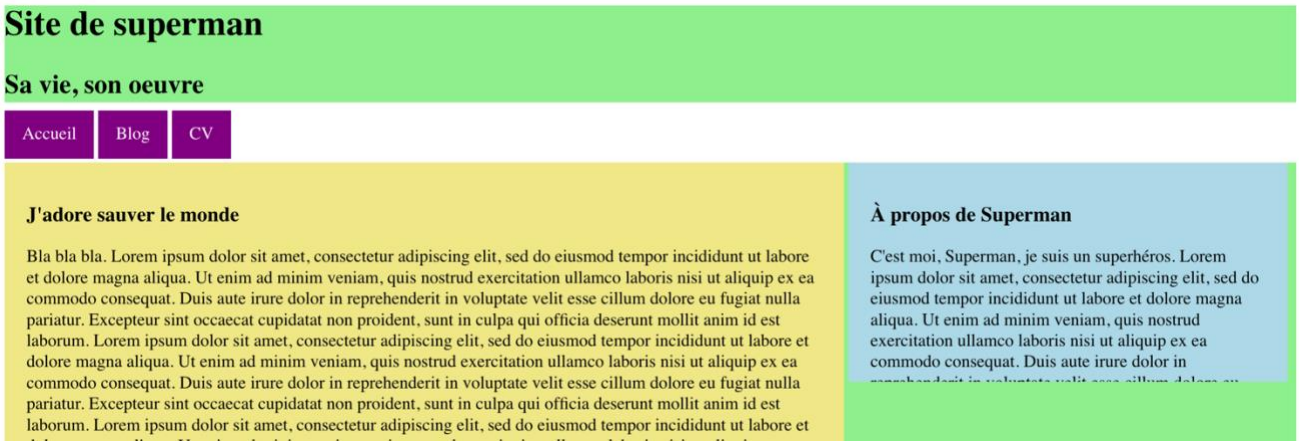
```
ul{
    padding-left: 0px;
    list-style-type: none;
}
li{
    display:inline-block;
}
a{ /* tout dans le a c'est mieux que dans le li */
    text-decoration: none;
    background-color: purple;
    color: white;
    padding: 14px 16px;
}
a:hover{
  background-color: orchid;
}
```

Solution moins propre : inline-block + float :

https://www.w3schools.com/cssref/tryit.asp?filename=trycss_float5

Exemple 2 : structure

Objectifs



Solution propre

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/32_inline_structure/index.html

On met un display: inline-block;et le vertical-align: top;pour tous les éléments de structuration.

On met des width à tous les éléments de structuration sinon il s'étale sur la largeur dont ils ont besoin.

En pourcentage : nav + section <100%.

En pourcentage : article + aside <100%.

On supprime les margin et les padding pour que les %age marche bien.

Quand on rétrécie la fenêtre d'affichage, ça suit jusqu'à passer dessous.

CSS du nav

```
nav ul{
      list-style-type: none;
      padding-left:0px;
}
nav li{
    display:inline-block;
}
nav a{
    text-decoration:none;
    background-color: purple;
    color: white;
    padding: 14px 16px;

}
a:hover{
  background-color: orchid;
}
```

CSS de la section

```
article, aside{
      padding : 20px;
      display:inline-block;
      vertical-align:top;
      box-sizing: border-box; /* pour avoir 64 + 35 le plus près
de 100 */
}

article{
      width : 65%; /* sans width, le bloc s'étale en largeur */
}

aside{
      width : 34%;
      height:200px;
      overflow : auto; /* ascenceur vertical */
}
```

Bilan

La gestion des marges rend les %age compliqués !

7 : Positionnement flottant : float

Principes

usage

Cette technique était et reste couramment utilisée. Elle est cependant plutôt à éviter et à remplacer par flex.

float

Déclarer une balise float fait que les balises suivantes viennent entourer la balise déclarée float.

Le float sert à entourer une image par du texte.

Usage normal

- ***Entourer une image par du texte***

  http://www.w3schools.com/cssref/pr_class_float.asp

  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa.

  Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut

- ***Mettre une première lettre en gros et l'entourer par la suite du texte***

  https://www.w3schools.com/cssref/tryit.asp?filename=trycss_float4

  This is some text.

  On peut aussi gérer la majuscule avec un « **::first-letter** » : c'est plus simple !

  https://www.w3schools.com/cssref/sel_firstletter.asp

- ***L'attribut clear : pour stopper l'effet du float***

  https://www.w3schools.com/cssref/tryit.asp?filename=trycss_class-clear

  This is some text. This is some text. This is some text. This is some text. This is some text. This is some text.

  This is also some text. This is also some text. This is also some text. This is also some text. This is also some text. This is also some text.

Usage de mise en page : blocs à l'horizontal

- ***Principes***

  On peut utiliser le float pour aligner des blocs successifs, par exemple les <li> d'un <ul>, mais aussi n'importe quels blocs.

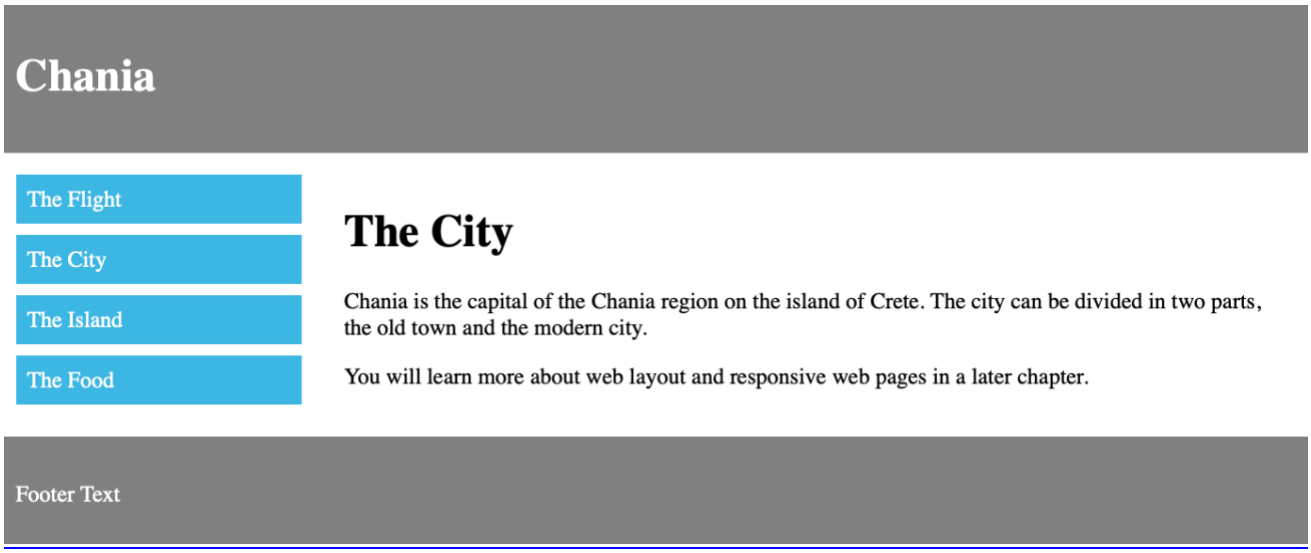
  Dans ce cas des <li> seront tous en « float », donc l'un après l'autre en ligne.

  C'est pratique pour les menus horizontaux ou pour des blocs à l'horizontal.

  Le problème est de placer proprement un clear :both.

- ***Exemple de blocs à l'horizontal***

  https://la-cascade.io/box-sizing-pour-les-nuls/



Ici, l'article est placé en colonne à droite du menu.

- ***Les trois outils pour régler le problème***
  - **float** : pour chacune des colonnes à mettre à l'horizontal
  - **::after** : pour ajouter une balise qui portera le **clear :both.** On le met sur le bloc des colonnes.
  - **box-sizing** : pour faciliter les %age.

- ***div::after***

  permet de créer une balise à la fin de la div. Cette balise sera une table -> display :table (à préférer à block car sans marge). Elle ne contient rien -> content :« » (pour mettre un contenu vide) et enfin on applique à cette nouvelle balise qui est la dernière de la div un clear :both qui annule les float pour les éléments suivants.

```
div::after {
    display: table;
    content: "";
    clear: both;
}
```

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

  Permet que les répartitions de blocs en %age se fassent proprement et simplement. Ici on a 75%

Exemple 1 : menu horizontal

Version 1 : solution propre -> ::after

Même principe que l'exemple précédent.

menuFloatAvecStyle.html

```
<!DOCTYPE html>
<html>
<html>
  <head>
    <meta charset="utf-8">
    <style>
      ul{
        background-color: pink;
        list-style-type: none;     /* supprimer les points */
        padding: 14px 0px;       /* 14 de hauteur, 0 à gauche */
      }
      ul::after {                  /* la balise ::after */
        display: table;
        content: "";
        clear: both;
      }
      li {                         /* les balises left */
        float: left;
      }
      li a {
        text-decoration:none;     /* pour supprimer le souligné */
        color: white;             /* pour le voir dans le pink */
        text-align: center;
        padding: 14px 16px;       /* 14 de haut, chaque li avec 16
à droite et gauche */

      }
      li a:hover {
        background-color: red;
      }
    </style>
  </head>

  <body>
    <ul>
      <li><a href="#home">Home</a></li>
      <li><a href="#news">News</a></li>
      <li><a href="#contact">Contact</a></li>
      <li><a href="#about">About</a></li>
    </ul>
  </body>

</html>
```

Version 2 : solution plus ou moins sale → on place un width 100%

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/04_menu_float/index.html

Au lieu du clear both, on met un width à 100%.

On travaille ensuite sur les a.

```
ul{
    float: left;
    width: 100%; /* ou alors un clear both dans le premier p */
    list-style-type: none;
}
a{ /* tout dans le a c'est mieux que le li */
    float: left;
    text-decoration: none;
}
a:hover{
  background-color: orchid;
}
```

Exemple 2 : version basique de page structurée

Objectif

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/05_float_page_structure/index.html

On veut obtenir ce résultat :

Il va falloir gérer les colonnes sur le nav et sur la section.

Solution propre

- ***Nav***

```
nav ul{
      list-style-type: none;
      padding-left:0px;
}
nav li{
      float:left;
}
nav a{
    text-decoration:none;
        padding-right : 16px;

}
nav ul::after {
    display: table;
    content: "";
    clear: both;
}
```

- ***Section***

```
article, aside{
       float: left;
}

* {
      box-sizing: border-box; /* pour avoir 65 + 35 = 100 */
}

article{
      width : 65%; /* sans width, le bloc s'étale en largeur */
}

aside{
      width : 35%;
      height:200px;
      overflow : auto; /* ascenceur vertical */
}
section::after{
    display: table;
    content: "";
    clear: both;
}
```

Autres exemples, plus ou moins sales !

Avec menu horizontal – exemple basique

On reprend le code précédent.

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/03-menu_float_de_base/index.html

On fait flotter toute la structure et on la passe à 100 % pour le paragraphe d'après.

On fait flotter tous les li et on les passe à 15% pour qu'ils ne prennent pas toutes la place.

```
nav{
    float: left;
    width: 100%;
}
li{
    float: left;
    width : 15%;
}
h1{
    clear:both
}
```

menu à gauche et plusieurs articles

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/10_page_structure_2/index.html

- ***A noter :***

Un simple float.

```
nav{
    float: left; /* le nav flotte dans le bloc suivant */
    width:8.7em; /* le nav de 155 px */
}

section{
    margin-left:9em; /* les sections à 160 px */
```

Les dimensions sont en em : c'est plus pratique.

menu à gauche et aside à droite

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/11_page_structure_3/index.html

- ***A noter :***

  Deux simples float à la suite ET un ::after avec un display :block et un clear :both.

```
nav{
    float: left; /* le nav flotte dans le bloc suivant */
    width:8.5em; /* le nav de 155 px */
}
section{
    margin-left:9em; /* les sections à 160 px */
}
article{
    float:left;
    width:49.5em;
}
aside{
    margin-left:50em;
}
section::after{
    display: block; /* on affiche un bloc */
    content: "";  /* sans texte */
    clear: both;  /* et on passe sous l'image */
}
```

page structurée un peu stylée mais avec largeur absolue : à éviter !

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/09_page_structure_1/index.html

L'aside (la colonne de gauche) est de 160px. Ca permet que son border right soit au même endroit que le boder-left de l'article (la colonne de droite). Pour ça, le margin-left est à 192 px.

```
aside{
    float:left;
    width:160px;
    padding:1em;
    border-right:1px solid gray;
}

article{ /* la même chose que article.content pour le 1er
exemple*/
    margin-left:192px;
    border-left:1px solid gray;
    padding:1em;
}
```

8 : Positionnements relatif et absolu

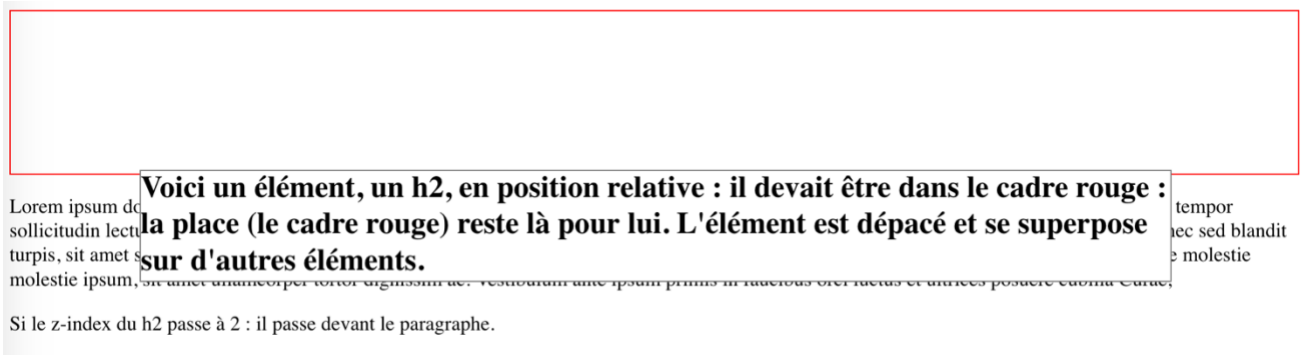
http://www.w3schools.com/cssref/pr_class_position.asp

https://www.alsacreations.com/tuto/lire/608-initiation-positionnement-css.html#positions

position : relative ;

Principe : toujours dans le flux

- L'élément en position relative est **toujours dans le flux des éléments affichés : la place qu'il devait prendre est toujours là,** potentiellement vide si le déplacement est important.
  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/41_relative_et_absolute/relative.html



Déplacement

- Le positionnement est fait **relativement à ce que serait sa position normale**.
- Le positionnement se fait **par rapport aux bordures du parent positionné** : top, bottom, left ou right.
- **Les éléments qui le suivent ne sont pas influencés par ce décalage.**

Superposition : z-index

- Le positionnement absolu peut créer des superpositions.
- L'attribut **z-index** permet de classer l'ordre de superposition. Le z-index le plus grand sera celui qui sera sur le dessus.

Eléments concernés

Ca s'applique à tous les types d'élément : inline, block, inline-block, etc.

Par exemple, je peux décaler les strong. Ou décaler un bloc.

Utilité

1) Pour générer des **effets particuliers** et pas vraiment de la mise en page.
2) Pour permettre de **gérer un z-index** c'est-à-dire une superposition pour l'élément.
3) Pour la position absolute : cf § suivant.

Exemples

- ***Position : relative ;***

```
Section{
      position: relative;
      bottom:15px; /* on monte de 15 px */
      left:25px;   /* on se déplace à gauche de 25  px */
}
```

- ***z-index : 1 ;***

```
#monImage{
      position: absolute;
      top: 100px;
      left:100px;
      z-index: 1;
}
```

http://www.w3schools.com/cssref/pr_pos_z-index.asp

position : absolute ;

principes

- L'élément en position : absolute n'est **plus dans le flux des éléments affichés**.
- Sa largeur par défaut devient celle du texte qu'il contient.
- Sa **position est fixée dans la page par rapport à son premier parent positionné** : le body par défaut**, le premier parent avec une position relative sinon**. Si la position est fixée par rapport à un autre élément, ce n'est plus vraiment une position absolue !
- Le positionnement se fait par rapport aux bordures du parent positionné : top, bottom, left ou right.

exemple

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/42_position_absolute/absolute.html

méthode de déplacement : bottom, right, left, top

On fixe une distance par rapport à un coin : top et right, top et left, bottom et right ou botom et left.

Ce faisant, on peut déplacer un bloc dans un autre, relativement à ses coins.

éléments concernés

Ca s'applique à tous les types d'élément : inline, block, inline-block, etc.

Par exemple, je peux décaler les strong. Ou décaler un bloc.

Utilité

Ca permet de faire de la mise en page.

Mais la position :absolute ne permet pas les redimensionnements automatiques en cas de changement d'écran ou de texte plus long.

position : fixed ;

principes

la position :fixed est une position :absolute. En plus la page peut circuler sous l'élément en position : fixed. C'est utilisé avec des images

exemple

On reprend le même exemple qu'avec le position :absolute, mais on le passe en fixed :

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/42_position_absolute/fixed.html

autre exemple : déplacements relatif et absolu d'éléments

Cet exemple permet de décaler un bloc de texte en bas à droite en le sortant du flux.

Du coup, le bloc suivant remonte (c'est le bloc bleu : test d'écriture 2). Et un texte est décalé comme le premier mais dans ce bloc bleu, en bas à droite.

La première image est placée en position absolue par rapport au body : en bas à droite.

La deuxième pareil mais par rapport au bloc bleu.

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/41_relative_et_absolute/index.html

Utilité pour la mise en page

La position absolue relative à un bloc peut être pratique pour placer un bouton dans un coin.

9 : Media Queries : responsive web design

Présentation

Problématique

Avec la multiplication des écrans (ordinateurs fixes, portables, tablettes, smartphones, télévision, etc.) et la généralisation du web, il devient nécessaire de pouvoir adapter facilement son site à différents écrans.

C'est la notion de « **responsive web design** » et de « **site web responsive** ».

Sur le site http://mediaqueri.es, on voit 4 formats « standards » : smartphone, tablette, portable, fixe.

- Smartphone : <768 px
- Tablette : 768 à <992 px
- Ordinateur portable : 992 à <1200 px
- Ecran fixe : >=1200 px

Solution

Une première solution consiste à faire un site par écran ! C'est long à faire et encore plus long à maintenir (répercuter les changements sur tous les sites !)

Les **medias queries de CSS-3** sont des **règles qui permettent de choisir les propriétés CSS à appliquer en fonction des caractéristiques de l'écran** ou de la fenêtre.

Principes de fonctionnement

Un seul fichier HTML

Dans tous les cas, on a un seul fichier HTML

Un fichier CSS

- Un fichier CSS unique qui utilise telle règle plutôt que telle autre selon l'écran : sélecteur @media.
- Sélecteur @media pour définir des règles sous conditions.

Plusieurs fichiers CSS

- On peut aussi définir un fichier CSS par écran.
- Dans ce cas, on utilise un attribut media dans la balise <link> pour dire quand choisir tel ou tel fichier CSS.

Exemples

- ***Page de l'exemple FLEX du chapitre 4***
  - Dans cette page, le but est de :
    1. Centrer le menu à la verticale
    2. Supprimer l'aside
    3. Centrer le footer qui passe sur fond bleu.
  - On fait ça quand la page fait moins de 1200 pixels.

    http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/5-flex-media/index_avec_CSS_flex_media.html
  - Techniquement :

```
@media only screen and (max-width: 1200px) {
    nav ul{
        flex-direction: column;
    }
    section aside {
        display:none;
    }
```

    Noter le fonctionnement de l'affichage du format avec du JavaScript.

- ***W3scholl - 1***

  http://www.w3schools.com/css/css_rwd_mediaqueries.asp

- ***W3scholl – version arrangée***

  http://bliaudet.free.fr/Cours/HTML/32_CSS_media_queries/0_Exemple_Media_Tout_HTML.html

1 seul fichier CSS : écrire des règles CSS sous conditions : @ media

Directive @media

Les règles s'écrivent dans la directive @media qui fixe la condition d'application de la règle.

```
@media screen and (max-width: 1280px){
      /* les règles sont ici */
}
```

screen veut dire que la règle s'applique sur des écrans d'ordinateurs (ordinateur, portable, tablette, etc.)

max-width précise que la règle s'applique quand la fenêtre est < 1280px. Donc si je rapetisse la fenêtre, je peux faire changer l'affichage.

Syntaxe

http://www.w3schools.com/css/css3_mediaqueries.asp

http://www.w3schools.com/cssref/css3_pr_mediaquery.asp

```
@media not|only typeDeMedia and (caractéristiquesDuMedia) {
    selecteur {
       attribute : valeur;
    }
}
```

Avec « typeDeMedia » on précise le type de média on peut préciser : pas celui-là (not) ou seulement celui-là (only).

Avec caractéristiqueDuMedia, on précise les caractéristiques pour que la règle se déclenche.

Il existe de nombreuses règles permettant de construire des media queries.

Type de media

Le plus souvent : screen. Mais aussi : tous (all), projection, tv, etc.
https://www.alsacreations.com/article/lire/930-css3-media-queries.html

Caractéristiques (features) du média

http://www.w3schools.com/cssref/css3_pr_mediaquery.asp
Il y en a 36 !

- ***A noter :***

  width, min-width, heigt, min-height : la fenêtre
  device-height, device-width : l'écran
  orientation : orientation de l'écran (portrait ou paysage)

Eléments techniques

- **Les nouvelles règlent prennent le dessus sur les anciennes.** On redéfinit un nouveau comportement dans un cas particulier : il faut donc écrire les règles en fin de fichier CSS.

- Pour faire disparaître une balise, on utilise : **display :none** ;

- Pour passer un menu à l'horizontal, si c'est un flex, on utilise : **flex-direction: column;**

Exemple

http://bliaudet.free.fr/Cours/HTML/32_CSS_media_queries/3_CSS_selecteur_media.html

```
p{ /* règle par défaut */
{
    color: blue;
    background: yellow;
}

/* Nouvelles règles si la fenêtre fait au plus 800 px de large */
@media all and (max-width: 800px) {
    p
    {
        color: red;
        background-color: aqua;
        font-size: 1.5em;
    }
}
```

Exemple flex : page complète

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/00_structurer_sa_page/5-flex-media/index_avec_CSS_flex_media.html

Exemple : page complète

http://bliaudet.free.fr/Cours/HTML/33_CSS_TP/TP_superman_flex/superman.html

Dans Firefox, ctrl U pour afficher le HTML, puis on clique sur le fichier CSS. On peut chercher « **flex** » dans le CSS ou « **justify-content** » ou « **align-items** » ou « **align-content** » pour voir tous les usages de flex dans la page.

Utiliser un fichier CSS plutôt qu'un autre : link media

Principes : attribut media

On utilise **l'attribut media dans le link** du fichier CSS pour définir une condition (une règle) à l'utilisation du fichier CSS.

http://www.w3schools.com/tags/att_link_media.asp

Exemple

```
<link rel="stylesheet" href="style.css" />

<link rel="stylesheet" href="style_petite_resolution.css"
     media="screen and (max-width: 1000px)" />
```

http://bliaudet.free.fr/Cours/HTML/32_CSS_media_queries/2_CSS_link_media.html

Explications

Le premier link dit que le fichier style.css est chargé dans tous les cas.

Le deuxième link conditionne le chargement du fichier style_petite_resolution.css au fait que le « media » soit un écran d'ordinateur (screen) de largeur inférieur à 1000px (max-width).

Le screen c'est la fenêtre dans laquelle se trouve le navigateur (et pas l'écran). Donc si je rapetisse la fenêtre, je peux faire changer l'affichage.

TP

On reprend le TP de la page 50.

On veut que sur un format smartphone, on obtienne le résultat suivant :

L'image 1 disparaît

Tout le bloc du carrousel disparaît

Le copyright n'est plus affiché

Les images du footer sont centrées.

Tout le reste est mis l'un en dessous de l'autre.

10 - Exemples de menus plus ou moins propres

Menu à l'horizontal

- ***Version minimaliste***

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/06_menus/01_menu_base.html

- ***Version stylée***

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/06_menus/01_menu.html

Menu déroulant

- ***Version minimaliste***

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/06_menus/02_menu_deroulant_base.html

- ***Version stylée flex***

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/06_menus/02_menu_deroulant_flex.html

- ***Version stylée flex sans id ni class***

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/06_menus/02_menu_deroulant_flex_sans_id_ni_class.html

- ***Version stylée***

  http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/06_menus/02_menu_deroulant.html

Menu déroulant progressif

http://bliaudet.free.fr/Cours/HTML/31_CSS_structure/06_menus/03_menu_deroulant_progressif.html

11 : Les formulaires <form> et <input>

Présentation

Première approche

Un formulaire c'est un ou plusieurs champs de saisie avec un bouton de validation.

Un formulaire est défini par une balise <form> qui contient les balises <input> des champs de saisie.

Exemple : http://www.w3schools.com/html/html_forms.asp

Utilisation avec un langage serveur

Les formulaires permettent de saisir de l'information, ou de cocher des boutons, ou de sélectionner un élément dans un menu déroulant, ou de sélectionner une couleur, etc.

L'information saisie ou sélectionnée pourra être récupérée pour des traitements dans le langage serveur (le PHP par exemple) et pour afficher une page par exemple.

Usage HTML sans langage serveur : préparer le travail

Pour bien comprendre les formulaires, il faut travailler sur un langage serveur ou avec du JavaScript.

Toutefois, on peut faire un usage simplifié dans un premier temps.

Premier usage : formulaire avec un seul champ

```
<body>
    <form action="url.html">
        <p>
            <label for="pseudo"> pseudo :  </label>
            <input type="text" name="pseudo" id="pseudo"
maxlength=8 placeholder="pseudo"/>
            <input type="submit" value="Envoyer">
        </p>
    </form>
</body>

<!-- label for + input id : si on clique sur le label, le curseur
va dans le champ de saisie -->
<!-- placeholder : exemple de valeur en grisé -->
<!-- maxlength : taille max du champ --></form>
```

3 balises importantes : form, label et input.

http://bliaudet.free.fr/Cours/HTML/41_HTML_fonctionnalites_avancees/04_test_formulaire/formulaire_champ_de_base.html

La balise form

http://www.w3schools.com/tags/tag_form.asp

L'attribut action

La balise form a un attribut :

- action=url : pour dire quelle page sera appelée quand on valide le formulaire.

La balise label

Elle permet de mettre un texte à côté de la balise input.

La balise input

Présentation

http://www.w3schools.com/tags/tag_input.asp

La balise input a trois attributs essentiels :

- **type**="nom du type" : champs de saisie (input), bouton à cocher, bouton à valider (submit), etc. Pour préciser le genre de saisie.
- **name**="nom de la variable" : pour définir le nom de la variable qui contiendra l'information saisie. Utile pour le langage serveur.
- **value**="valeur" : pour donner une valeur à la variable (valeur par défaut avant la saisie).

L'attribut value

C'est la valeur de la variable. On s'en sert si on veut donner une valeur par défaut ou dans le cas de bouton à cocher avec des valeurs ou de menu déroulant. On lui donne directement une valeur dans le cas d'un input submit.

L'attribut type

http://www.w3schools.com/html/html_form_input_types.asp

type est l'attribut qui permet de définir le type de formulaire : champ de saisie, bouton, etc.

- type text : pour saisir une information dans un champ
- type submit : bouton : c'est le bouton qui amène vers l'action.
- type password : caractères cachés
- type email
- type url
- type tel
- type number (+min, max, step)
- type range (+min, max, step avec curseur)
- type color (firefox propose une palette colorée)
- type date
- type search
- type chechobox (+checked) : boutons à cocher
- type radio : radio bouton (1 seul parmi plusieurs). On en met plusieurs avec le même name et value et id spécifiques. On ajoute un label avec for spécifique par input
- type hidden : champs caché. Les utilisateurs ne les voient pas. Ils permettent de faire circuler des informations de page en page sans que cela concerne l'utilisateur. Par exemple, avec un bouton, on peut vouloir faire passer une information pour la page appelée.

Autre balise de saisie : <textarea> pour un texte à saisir ou à afficher

Présentation

Textarea peut servir à : afficher ou saisir du texte (pour la saisie, il faut ajouter l'attribut name pour récupérer ce qu'on a saisi).

Taille : rows et cols en html ou widht et height en CSS.

http://www.w3schools.com/tags/tag_textarea.asp

Affichage

```
<textarea id="texte" rows="4" cols="50"><br/>
     On peut mettre du texte dans le textarea
</textarea>
```

Saisie

```
<label for="texte">nom</label>
<textarea name="texte" id="texte" rows="4" cols="50" autofocus>
     On peut mettre un pré-remplissage dans le textarea = value
</textarea>
```

rows et cols peuvent se paramétrer en CSS avec width et height sur l'id texte.

Autre balise de saisie : <select> et <option> pour un menu déroulant

Principe : balises <select> et <option>

http://www.w3schools.com/tags/tag_select.asp

Autre exemple

```
<label for"voitures">Voiture</label><br/>
<select name="voitures" id="voitures">
  <option value="volvo">Volvo</option>
  <option value="bmw">BMW</option>
  <option value="renault" selected>Renault</option>
  <option value="ford">Ford</option>
</select>
```

+ selected pour la valeur par défaut

http://bliaudet.free.fr/Cours/HTML/41_HTML_fonctionnalites_avancees/04_test_formulaire/formulaire_menu_de_base.html

Formulaire de saisie avec plusieurs champs

Présentation

Le code suivant :

http://bliaudet.free.fr/Cours/HTML/41_HTML_fonctionnalites_avancees/04_test_formulaire/formulaire/formulaire.html

permet de produire le formulaire complet suivant.

A noter que ce formulaire contient du JavaScript et fait des vérifications de saisie

Les balises < fieldset > et < legend >

< fieldset > et <legend> pour regrouper des input dans un cadre avec un titre.

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_fieldset

La balise <fieldset> permet de mettre un cadre autour d'un ensemble de balise input.

On peut aussi ajouter une balise <legend> pour mettre un titre au cadre

12 : Les boutons JavaScript : balise <button>

Présentation

La balise <button> sert à faire des boutons cliquables qui seront utilisés par le JavaScript.

On n'est pas obligé de les utiliser pour ça et on peut créer des boutons cliquables qui appelle du JavaScript qui ne soit pas des <button », mais mieux vaut garder l'usage normal.

Le code suivant propose un bouton qui affiche un petit message :

```
<!DOCTYPE html>
<html>
<body>
     <button type="button" onclick="alert('Hello world!')">Click
Me!</button>
</body>
</html>
```

http://bliaudet.free.fr/Cours/HTML/41_HTML_fonctionnalites_avancees/05_button_javascript/button_hello_world.html

href – submit – button

De façon standard, on trouve donc trois façons de coder des boutons en HTML :

Avec un href

Avec un submit de formulaire

Avec un button de JavaScript

13 : Validité du code

Déboguer son code HTML

Comment trouver les erreurs ?

Editeur à coloration syntaxique

La coloration syntaxique permet de repérer les fautes de syntaxe

Mise en page du code : indentation

Il faut indenter correctement son code pour y voir clair et bien fermer toutes les balises ouvertes.

Imbrication des balises

Si la solution n'apparaît pas immédiatement, il est toujours utile de vérifier la cohérence de l'imbrication des balises (la hiérarchie des balises).

Outils des navigateurs

Les navigateurs fournissent des outils (de développement, de débogage) qu'il faut savoir utiliser un minimum. Sur Firefox, il faut installer Firebug.

Les navigateurs permettent de tester le code « en direct ».

Contour et couleur de fond

On peut toujours ajouter une couleur de fond sur un élément qui pose un problème pour y voir plus clair, et/ou ajouter des bordures.

Validité W3C

Principe

Le W3C fournit un outil en ligne de vérification du code des pages HTML.

https://validator.w3.org

On fournit le fichier HTML à vérifier et l'outil renvoie le résultat.

Type de commentaires : info, warning, error

Les info, en vert, précisent ce que fait le validateur

Les warning, en jaune, peuvent mettre sur la piste de problème. Mais parfois ce qu'il signale n'est pas pertinent, particulièrement en ce qui concerne les éléments de structuration.

Les erreurs : en rouge sont à traiter. L'objectif est de n'avoir aucune erreur.

Objectif : pas d'erreur !

Même la page semble s'afficher correctement, il faut corriger son fichier HTML jusqu'à n'avoir aucune erreur.

Cela permet d'éviter des comportements étranges de la page sur différents navigateurs et cela facilitera la maintenance du code.

Quelques règles de syntaxe – les bonnes pratiques

- ***Quelques règles***
  - Les balises doivent être fermées et dans l'ordre
  - Un guillemet doit être fermé
  - La structure head body doit être respectée
  - Une image doit avoir un alt
  - Un br doit être dans un paragraphe
  - Eviter les balises obsolètes (comme les frames)
  - Etc !!!

- ***Voir ici les bons usages :***
  http://www.w3schools.com/html/html5_syntax.asp

Compatibilité avec les navigateurs

Principes

Le site doit fonctionner sur tous les navigateurs et sur toutes les versions en usage. Pour les version le problème se pose surtout avec IE et les version 6, 7 et 8.

L'objectif numéro 1 est d'atteindre un résultat satisfaisant (pas forcément parfait) sur toutes les versions d'usage courant des navigateurs.

Sur les anciennes versions, il faut adapter les solutions à apporter au temps qu'on est prêt à y consacrer. Au minimum, mieux vaut ne pas afficher de page illisible et mettre un message demandant à l'utilisateur de mettre à jour son navigateur.

Aujourd'hui, avec HTML 5 et CSS 3, le problème se pose moins car tous les utilisateurs ont intérêt à avoir des navigateurs à jour.

Solution 1 : code JavaScript

IE9 ne prend pas en compte les balises structurantes.

Un code JavaScript à placer dans le head permet de régler le problème :

Ce code va chercher un script sur le web : le fichier html5.js

```
<link rel="stylesheet" href="superman.css" />
<!--[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

Solution 2 : cas spécifique aux vieilles version d'IE, 6 à 8 : <!--[if lte IE]>

- ***On peut mettre des commentaires HTML spécifiques à IE :***

```
<!--[if lte IE]>
      code pris en compte par IE 7
<![endif]-->
```

Ces commentaires sont considérés comme tels par tous les navigateurs sauf IE qui va traiter le code dans le commentaire.

On peut préciser IE6, IE7 ou IE8.

On peut aussi écrire [if lte IE]

- lte : inférieur ou égal,
- lt : inférieur ;
- gt : supérieur ;
- gte : supérieur ou égal ;
- !: négation

- ***On peut charger un fichier CSS sur ces conditions :***

```
<!--[if IE7]>
    <link rel="stylesheet" href="ie7.css" />
<![endif]-->
```

Pour gérer toutes les anciennes versions de IE, on pourrait définir autant de fichier CSS que de versions. Mais c'est compliqué à gérer.

- ***On peut cibler les modifications de style par navigateur***

Mieux vaut ne garder qu'un fichier CSS mais et préciser dans la balise body à quelle version on a affaire. On peut ainsi cibler les modifications de style par navigateur.

HTML : à la place du <body>

```
<!--[if IE6]><body class="ie6 old_ie"><![endif]-->
<!--[if IE7]><body class="ie7 old_ie"><![endif]-->
<!--[if IE8]><body class="ie8"><![endif]-->
<!--[if IE9]><body class="ie9"><![endif]-->
<!--[if !IE]><!--><body><!--<![endif]-->
```

CSS : pour un nav de IE8

```
.ie8 nav
{
      code pour IE8
}
```

Balises inconnues chez IE 8 et inf

Certaines balises ne sont pas reconnues par IE8 et inférieurs.

Il faut rajouter du javascript dans le <head> pour régler le problème, entre <meta> et <title>:

```
<!--[if lt IE 9]>
  <script
     src="http://html5shim.googlecode.com/svn/trunk/html5.js">
  </script>
<![endif]-->
```

http://www.htmlgoodies.com/html5/tutorials/three-ways-you-can-use-html5-on-your-website-today.html#fbid=Uc1UaoqdbQR

14. Méthode de réalisation

La méthode

Conception

Avant de réaliser, il faut commencer par réfléchir à ce qu'on va faire : c'est la conception.

4. **De quoi va parler le site** : un blog qui présente un sujet. Il faut définir les différents éléments dont on va parler et **définir les différentes pages sur lesquels on va arriver**.
5. Quelle mise en page : il faut **faire une maquette**. On peut demander à un graphiste de faire la maquette. Le **graphiste** fournit des images, des couleurs, des polices, ensuite on va les programmer.
6. Ensuite on fera le **squelette HTML** de la page web : header, nav, section, etc.

Réalisation

7. Il faut récupérer les images et les polices.
8. Ensuite on écrit les pages HTML : c'est assez facile.
9. Enfin on fait le **CSS**. C'est là qu'il y a le plus de difficultés.

Réalisation HTML

- Dans le body, on trouve souvent une balise div qui contient tout avec un id bloc_page (main_wrapper en anglais, wrapper = emballage). Ca permet de contrôler les dimensions et caractéristiques générales de la page. A noter que c'est inutile et qu'on peut passer directement par le body.
- On met des balises div pour tous les blocs qui ne sont pas structurés (section, article, etc.).
- On peut faire des regroupements de balises de structuration (mettre un div autour d'une section et d'un nav, ou autour d'une section et d'un div, etc.)
- On peut mettre des id ou des class à chaque div pour préparer le CSS
- Quand on passe au CSS, il se peut qu'on rajoute des balises pour permettre de finaliser le design.

Réalisation CSS

Principe

Ne pas chercher la perfection d'autant que le rendu varie d'un navigateur à l'autre.
Ca doit juste rendre « bien » et être fonctionnel (user friendly).

Méthode

Plusieurs méthodes sont possibles.
Les éléments à prendre en compte sont :

- Les éléments structurant de la page : body général, header, nav, corps, sections, footer
- Le positionnement qui inclus l'ajout des images
- Les polices
- Le graphisme

Choix d'une méthode

1. Commencer par la mise en page par élément structurant.
2. Ensuite, choisir les polices et les images
3. Finaliser le graphisme : couleur, taille et style de typo, calage divers.
4. Gérer le web responsive.

Dans tous les cas, il faut finir par :

5. Assurer la compatibilité avec les navigateurs (surtout IE)
6. Vérifier la validité W3C

15 : TP

1 - Les merveilles

Structurer la page principale en utilisant des balises sémantiques.

On se dote de :

- Une en-tête à l'horizontal qui va contenir un menu horizontal centré à 4 choix. On met choix 1, choix2, choix 3, choix 4.
- Sous l'en-tête, les deux listes l'une à côté de l'autre. Sous les deux listes, le tableau centré.
- A droite des deux listes et du tableau centré, une colonne qui contiendra un sommaire avec la liste des merveilles par ordre alphabétique. Chaque intitulé permettra d'accéder à la merveille correspondante dans la liste. Sous le sommaire, on trouvera, une image et une ou deux vidéos avec un titre à chaque fois.
- En pied de page, on mettra 4 images à l'horizontal et centrées permettant d'accéder aux références : Wikipedia, Universalis et Unesco.

Structurer les pages de détails.

Dans chaque page de détail, on retrouve l'en-tête et le pied de page. On retrouve aussi la colonne de droite avec uniquement des vidéos (pas d'image et bien sûr pas de sommaire).

On mettra un bouton de retour à la page principale en haut de la colonne de droite.

Media queries

Reprendre l'exercice sur les merveilles mis en page.

Quand vous atteignez un format smartphone, faites disparaître le tableau et remettez les anciennes merveilles et les nouvelles merveilles l'une en dessous de l'autres.

Centrez tout le contenu de l'en-tête (menu des choix l'un en dessous de l'autre).

Faites passer le sommaire directement sous l'en-tête, avant les merveilles. Effacer les images et vidéos du jour.

Dans le pied de page : laissez les icones tel quel mais centrer le texte : « référence ».

Visuel

Exemple de visuel possible. Vous pouvez, tout en suivant le cahier des charges, changer pas mal de choses pour styler à votre convenance.

suite :

Page appelée

2 - Coder une page « classique »

Regardez cette page.

https://opendata.paris.fr/pages/home/

Construisez une structure équivalente.

Dans la colonne de gauche, on pourra mettre moins de liens.

- ***Alternative :***

  Idem avec cette page :

  https://www.paris.fr/services-et-infos-pratiques

Media queries

Regardez comment évolue la page précédente quand passe à une résolution de type smartphone (ou regardez la page sur un smartphone). Constater l'évolution de la présentation. Coder cette évolution.

3 – Structuration in-line block et structuration float

1) Reprenez l'exercice Merveilles :
   - Remplacez la structuration flex par une structuration in-line block.
   - Remplacez la structuration flex par une structuration float.

2) Reprenez l'exemple « superman » :

http://bliaudet.free.fr/Cours/HTML/33_CSS_TP/TP_superman_flex/superman.html

   - Remplacez la structuration flex par une structuration in-line block.
   - Remplacez la structuration flex par une structuration float.