Comprendre le CSS

LEÇON 1 Le CSS pour donner un style à vos pages HTML

Utiliser la balise LINK pour lier une feuille de style CSS à une page HTML

1/26

Regardez le fichier `index.html`. Plutôt standard, non ? Vous connaissez tout ça : titres, paragraphes, images, listes et tableaux.

Regardez le fichier `stylesheet.css`. Il contient toutes les informations de style CSS pour mettre en page les éléments HTML (leur position, leur couleur, leur taille, etc.).

Nous avons commenté une ligne cruciale (**ligne 4**) dans l'onglet `index.html`. Si vous enlevez les balises `<!--` et `-->` au début et à la fin de cette ligne, vous verrez alors la magie du CSS opérer. **Attention ! Ne supprimez surtout pas la balise `<link>`**.

2. Qu'est-ce que le CSS : Cascading Style Sheets

2/26

Le CSS (qui signifie **C**ascading **S**tyle **S**heets ou feuilles de style en cascade) est un langage utilisé pour définir l'apparence et la forme d'un contenu en HTML.

Une **style sheet** décrit ce à quoi doit ressembler le HTML, tout simplement !

On dit que ces feuilles de style sont **en cascade** car si vous appliquez un style à un élément HTML, il va se répercuter "en cascade" à tous les autres éléments HTML qu'il contient.

3. Pourquoi utiliser les feuilles de style CSS

3/26

Il faut séparer la forme (CSS) de la structure (HTML) pour deux raisons principales :

- cela permet d'appliquer le même style à plusieurs éléments HTML sans devoir écrire le code (ex : `style="color:red"`) pour chacun des éléments.
- vous pouvez appliquer une apparence et un formatage identique à plusieurs pages HTML à partir d'un seul fichier CSS.

Jetez un œil au HTML dans l'onglet `index.html`. Il y a beaucoup de balises ‹span›‹/span› ! Tous ces mots ont une police basique mais nous voulons qu'ils soient super canons !

4. Lier une feuille de style CSS dans le HEAD ou les balises STYLE

4/26

Nous vous avons montré précédemment comment déclarer un style directement dans les balises HTML, de la manière suivante :

```
<p style="color: red">Texte en rouge !</p>
```

En fait, c'est une façon peu efficace de déclarer les styles utilisés par votre site internet pour les raisons que nous avons déjà mentionnées : vous devez écrire le même code à de multiples endroits et si vous voulez faire une modification importante de style sur plusieurs éléments, vous devrez modifier chacune des balises une par une. Avec un seul fichier CSS, il vous suffira de faire les modifications à un seul endroit !

Il y a deux manières de déclarer du CSS. La première est de mettre votre CSS entre des balises `<style></style>`, directement dans votre fichier HTML. Ces balises `<style>` s'insèrent entre les balises `<head></head>` de votre fichier HTML.

5. Lier CSS et HTML avec les attributs TYPE, REL et HREF

5/26

Vous avez vu qu'on pouvait mettre le CSS dans la page HTML en utilisant les balises `<style></style>`. Mais il y a un autre moyen, beaucoup plus pratique.

Vous savez que vous devez écrire votre CSS dans **un fichier à part**. Mais comment être sûr que le fichier HTML aura accès aux informations de style contenues dans le fichier CSS ?

Grâce à une balise ‹link› (comme vous l'avez vu dans les premiers exercices de ce cours) insérée entre les balises ‹head›‹/head› de votre page HTML. Votre balise `<link>` a besoin de trois attributs :

Un attribut `type` qui doit toujours être égal à `"text/css"`.

Un attribut `rel` qui doit toujours être égal à `"stylesheet"`

Un attribut `href` qui doit pointer vers l'adresse internet de votre fichier CSS.

Dans l'éditeur à droite, vous pouvez voir deux fichiers : `index.html` et `stylesheet.css`.

6. Les balises auto-fermantes pour lier CSS et HTML

6/26

Ceci nous amène à un concept HTML bien pratique : **la balise auto-fermante**.

Étant donné qu'il n'y a jamais aucun texte entre les balises `‹link› et ‹/link›`, il suffit d'utiliser une seule balise qui va servir de balise ouvrante **et** fermante. Comme ceci :

```
‹link type="text/css" rel="stylesheet" href="adresse du fichier CSS" /›
```

```
‹img src="adresse web" /›
```

La plupart des balises ne sont pas auto-fermantes mais nous vous signalerons celles qui le sont pour vous faire gagner du temps.

7. La syntaxe CSS pour définir le style des éléments HTML

7/26

La syntaxe du CSS est différente de celle du HTML mais ne vous inquiétez pas : c'est facile à apprendre ! Le format de base est le suivant :

```
sélecteur {
    propriété: valeur;
}
```

Un **sélecteur** peut être n'importe quel élément HTML tel que `<p>` `<div>` `<span>` `<i>` `<img>`. Il suffit de retirer les < >.

Par exemple, pour mettre le texte des paragraphes en rouge, il suffit d'ajouter dans le fichier CSS :

```
p {
    color: red;
}
```

Une **propriété** est une caractéristique d'un sélecteur. La famille de police (`font-family`), la couleur (`color`) et la taille de police (`font-size`) sont des caractéristiques qui peuvent être attribuées à un mot ou à des paragraphes dans vos pages web (il y en a beaucoup d'autres !).

Une **valeur** est - comme son nom l'indique - une valeur possible de la propriété. À la propriété `color`, on peut attribuer la valeur rouge (`red`), bleu (`blue`), noire (`black`) ou n'importe quelle autre couleur. La propriété `font-family` peut prendre pour valeur `Arial, Verdana, Impact` ou toute autre police de caractères. Et ainsi de suite...

Chaque couple **propriété/valeur** doit se terminer par un **point-virgule (;)**. C'est ce qui permet au CSS de savoir que votre paire est terminée et que vous êtes prêt pour la prochaine.

8. Un sélecteur, beaucoup de propriétés

8/26

L'un des autres avantages du CSS est qu'il vous permet de définir plusieurs propriétés pour un même sélecteur. Par exemple, si vous voulez que tous les paragraphes aient la même couleur, la même famille de police et la même taille, il vous suffit d'écrire :

```
p {
    font-family: Arial;
    color: blue;
    font-size: 24px;
}
```

N'oubliez pas : chaque couple `propriété:valeur` doit se terminer par un point-virgule !

Note : Si vous avez ajusté le zoom de votre navigateur, les exercices qui utilisent `font-size` (taille de police) et height (hauteur) ne fonctionneront pas correctement. Pour y remédier, tapez Cmd+0 ou Ctrl+0 pour remettre votre affichage par défaut.

9. Utiliser le CSS pour changer la forme d'une page HTML

9/26

Bon travail ! C'est en pratiquant que l'on progresse. Entraînez-vous donc encore un peu !

Astuces:

Rappelez-vous de la syntaxe :

```
   sélecteur {

        propriété: valeur;

    }
```

10. Utiliser les points-virgules et les accolades pour votre document CSS

10/26

Comme vous commencez à mettre de plus en plus de paires de propriété:valeur pour chaque sélecteur CSS, il est important de ne pas oublier les **points-virgules ;** à la fin de chaque ligne.

Le point-virgule indique au CSS que la paire propriété:valeur est terminée et qu'il est temps de passer à la suivante. Sans point-virgule, cela va devenir confus et votre page ne ressemblera pas à ce que vous voulez.

Rappelez-vous également que toutes les paires de propriété:valeur d'un sélecteur **doivent se trouver dans des accolades** `&#123; }`.

11. Mettre des commentaires sur une feuillet de style CSS

11/26

Bien ! Vous commencez vraiment à avoir la main.

Comme il est important d'avoir une bonne syntaxe, c'est un bon réflexe d'écrire **des commentaires**. De bons commentaires vous aideront à vous souvenir pourquoi vous avez fait certaines choses d'une certaine manière (ou pour aider quelqu'un qui souhaiterait comprendre votre code).

Comme vous l'avez vu, les commentaires HTML ressemblent à cela :

```
<!-- Commentaire HTML -->
```

Contrairement aux commentaires CSS qui s'écrivent comme ceci :

```
/* Commentaire CSS */
```

Rappelez-vous que l'ordinateur ne prend pas en compte les commentaires quand il interprète le HTML et le CSS. Écrire des commentaires est une très bonne habitude à prendre !

12. Coder une page HTML et une feuille de style CSS

12/26

Vous avez beaucoup appris en seulement quelques leçons.

Cet exercice est rappel pour être certain que vous maîtrisez bien vos nouvelles connaissances.

13. Les valeurs hexadécimales pour les couleurs en HTML

Depuis le début, pour attribuer une valeur à la propriété color avec CSS, vous écrivez `color:nomdelacouleur;`.

Vous vous êtes peut-être demandé comment faire pour utiliser du mauve ? Ou du bordeaux ? Ou encore du saumon ? Le CSS connaît-il toutes ces couleurs ?

Non, le CSS ne connaît pas le nom de toutes les couleurs ! Mais il peut comprendre des millions de couleurs exprimées en **valeur hexadécimale**.

Vous avez l'habitude des **valeurs décimales** avec lesquelles nous comptons tous les jours ! Prenons par exemple le nombre 1432. Chaque chiffre de ce nombre peut prendre des valeurs entre 0 et 9. Donc, chaque chiffre n'a que 10 valeurs possibles. C'est pourquoi on parle de **base 10** pour décrire cette forme classique de comptage.

Une valeur hexadécimale est composée **de chiffres et/ou de lettres**. Le comptage hexadécimal est en **base 16** : chaque signe qui compose la valeur est soit **un chiffre de 0 à 9** soit **une lettre de A à F**.

14. Les codes couleurs HTML avec les valeurs hexadécimales

Il y a beaucoup d'outils sur Internet pour trouver le code hexadécimal (ou hex) d'une couleur.

Il vous suffit, par exemple, de taper *palette couleur hex* ou *nuancier couleur hex* dans votre moteur de recherche.

Les codes hexadécimaux commencent tous par le symbole dièse `#`. Ils sont composés de **six signes** (chiffres ou lettres) et **ne sont pas sensibles à la casse** : les lettres peuvent être écrites en majuscules ou en minuscules, c'est pareil ! `#FFC125` et `#ffc125` représentent donc la même couleur.

15. Définir la taille d'une police en pixels ou en EMS

15/26

Dans les exercices précédents, vous avez utilisé l'unité `px` (pour "pixels") pour définir la taille d'une police. Par exemple :

```
p {
        font-size: 10px;
  }
```

Un pixel est un point sur votre écran d'ordinateur. Spécifier la taille de police en pixels, c'est très bien si vous voulez que les visiteurs de votre site web voient exactement la même chose que ce qui est affiché sur votre propre écran.

Mais que se passe-t-il si des utilisateurs consultent votre site sur un écran ayant des dimensions très différentes du vôtre, celui d'un smartphone ou d'une tablette par exemple ? C'est ici qu'entrent en scène les `em` (à ne pas confondre avec les balises `<em></em>` utilisées pour mettre du texte en italique dans le HTML).

L'unité de mesure `em` est une mesure relative : `1em` est égal à la taille de police par défaut de l'écran utilisé. C'est donc parfait pour les smartphones par exemple puisque la taille affichée des textes dépend de la taille de

l'écran du smartphone utilisé et non de la taille absolue de vos polices (celle que vous avez choisie quand vous avez codé votre site web).

Une taille de police en `em` dit seulement : "Hey, `1em`, c'est la taille de police utilisée normalement ; la valeur `2em` correspond à une taille de police deux fois plus grande ; la valeur `0.5em` correspond à une taille de police deux fois plus petite !".

Testons tout cela ! Nous avons créé trois paragraphes différents avec des tailles de police (`font-size`) égales à `1em`, `0.5em` et `2em`. Désormais, vous pouvez utiliser soit `px` soit `em` comme unités de mesure. Nous voulions seulement vous montrer ce qu'étaient les em pour que vous ne soyez pas surpris quand vous en rencontrerez plus tard.

16. 3 types de Font en CSS : Serif, Sans-serif, Cursive

16/26

Jusqu'à présent, vous avez utilisé les polices Arial, Verdana, Courier, Impact et Garamond. Mais combien de polices différentes CSS connaît-il ?

Cela dépend. La plupart des ordinateurs vont comprendre les polices de base comme Arial, Verdana, Courier et Garamond mais chacun peut avoir différentes polices installées dans son ordinateur personnel. Heureusement, CSS a des polices intégrées par défaut. Cela permet à n'importe quel utilisateur de voir ce que vous voulez lui montrer. Ces polices intégrées sont de **3 types :**

> **Serif :** dans ce type de police, les extrémités des lettres ont des extensions. La Garamond est une police serif. Faites une recherche sur "serif" pour trouver des exemples.
>
> **Sans-serif :** c'est une police sans extensions. L'Arial et la Verdana sont des polices sans-serif.
>
> **Cursive :** c'est une police qui ressemble à une écriture manuscrite.

Vous verrez ultérieurement comment importer votre propre police. Ainsi, vous serez sûr que les visiteurs de votre site verront vos pages avec toutes les polices que vous voulez afficher.

17. Utiliser des valeurs de secours pour les Fonts en CSS

17/26

Il n'est pas nécessaire d'utiliser directement des valeurs de police par défaut telles que `cursive` ou `sans-serif`. Vous pouvez dire à CSS d'essayer plusieurs polices : si la première n'est pas disponible, CSS passera à la suivante et ainsi de suite.

Par exemple, si vous écrivez :

```
p {
    font-family: Tahoma, Verdana, sans-serif;
  }
```

CSS va d'abord essayer d'appliquer Tahoma à vos paragraphes. Si l'ordinateur de l'utilisateur n'a pas cette police, alors CSS essaiera d'appliquer Verdana. En cas d'échec, le texte s'affichera finalement en sans-serif.

18. Résumé : Donner du style à votre page HTML avec le CSS

18/26

Dans les exercices précédents, vous avez vu :

- la nature et l'intérêt du CSS;
- l'intérêt de séparer la forme du contenu;
- la syntaxe CSS avec les (multiples) sélecteurs, les (multiples) paires de propriétés:valeurs et les commentaires;
- des détails sur le fonctionnement des polices, de leur taille et des couleurs.

19. Couleur d'arrière-plan, hauteur et largeur en HTML

19/26

La balise générique `<div>` permet de structurer une page web en blocs. Pour mieux les visualiser il est possible de leur appliquer les propriétés suivantes.

Pour cela, il faut choisir les valeurs de trois propriétés :

- La couleur d'arrière-plan du bloc (`background-color`), pour laquelle vous choisissez une valeur hexadécimale.
- La hauteur du bloc (`height`) que vous mesurez en pixels.
- La largeur du bloc (`width`) en pixels également.

Les exercices ci-après vous donneront un bref aperçu des différents éléments HTML que vous pouvez sélectionner ainsi que quelques exemples de paires propriété:valeur (comme les trois présentées ci-dessus). Nous reviendrons plus en détail sur la sélection d'éléments HTML dans le prochain cours !

20. Créer un style de bordure de tableau avec la propriété BORDER

20/26

De nombreux éléments HTML peuvent faire appel à la propriété `border`. Elle est tout particulièrement utile pour les tableaux.

La propriété `border` peut prendre plusieurs valeurs. Par exemple, pour afficher une bordure de 2 pixels, continue et rouge, il suffit d'écrire :

```
sélecteur {
    border:2px solid red;
}
```

21. Définir un style pour les liens avec le CSS

21/26

Les liens hypertexte (balises `‹a›‹/a›`) peuvent être mis en forme de la même façon que du texte normal, en utilisant les propriétés `color, font-family, font-size`, etc.

Par défaut, les liens sont soulignés mais il est possible de changer cela avec l'utilisation de la propriété `text-decoration`.

22. Rappel : Lier une feuillet de style CSS à une page HTML

22/26

Vous avez beaucoup appris depuis le début. À partir de maintenant, nous ferons des révisions plus fréquentes pour être sûrs que vous maîtrisez bien les notions de CSS déjà vues.

23. Rappel : La syntaxe à utiliser en CSS

23/26

Notre squelette HTML est désormais connecté à notre page CSS.

Cet exercice fait un rappel sur les sélecteurs, les propriétés et leurs valeurs. Rappelez-vous que la syntaxe est la suivante :

```
sélecteur {
    propriété: valeur;
}
```

24. Rappel : Utiliser des polices de secours pour les fonts en CSS

L'ordinateur d'un utilisateur ne possède pas toujours la police que vous souhaitez utiliser pour votre site web. Pour cette raison, il faut prévoir des solutions de secours.

25. Rappel : Dimensionner un élément HTML

Pour ce rappel, vous allez revoir les propriétés pour donner des dimensions et une bordure à une balise HTML.

astuces: Rappelez-vous que votre balise `‹ img / ›` doit comporter une propriété `src` :

```
‹ img src="URL du site web" / ›
```

26. Rappel : Utiliser la propriété BOX en CSS

Pour ce dernier rappel, vous allez revoir comment modifier l'apparence par défaut d'un lien.

- - - - - - - - - - - - - - - - - - -

LEÇON 2 Créer un bouton avec le CSS sur une page HTML

1. Introduction : Créer un bouton HTML à partir d'un lien

1/6

Sur un site web, un bouton est en lien `<a></a>` classique auquel on applique un ensemble de propriétés CSS pour le mettre en valeur au sein de la page HTML.

Le but est d'attirer l'oeil de l'utilisateur afin qu'il effectue une action spécifique (inscription, connexion, etc.).

2. Utiliser une balise de type bloc pour créer un bouton HTML

2/6

Un lien `<a></a>` est de type `inline`. Pour respecter la sémantique HTML, il faut l'insérer dans une balise de type `bloc`.

L'objectif de cet exercice est de créer la structure HTML du bouton.

Le principe du bouton est qu'il soit cliquable afin de diriger l'utilisateur là où vous le souhaitez. Il faut donc utiliser une paire de balises `<a></a>`.

Note : La balise `<a>` est de type `inline`, il faudrait donc l'insérer dans une balise de type `bloc` afin de respecter la sémantique HTML. Mais nous contournerons le problème grâce au CSS par la suite.

3. Mettre un forme un bouton avec le CSS

3/6

Dans cet exercice, vous allez appliquez une mise en forme CSS de base afin de convertir la balise `‹ a ›` en `block`.

De cette façon, vous pourrez lui appliquer des dimensions et des marges afin de lui donner un rendu de "bouton".

4. Positionner un bouton en HTML avec la propriété MARGIN

4/6

L'objectif de cet exercice est de **positionner** le bouton.

Nous approfondirons le positionnement dans les exercices à venir mais voici quelques explications pour la compréhension de cet exercice :

> Quand on applique la valeur `auto` aux **marges externes** (`margin`) gauche et droite, cela permet de centrer horizontalement un élément HTML.
> `text-align: center` est le code qui permet de centrer horizontalement du contenu textuel;

5. Mettre en forme le texte d'un bouton HTML

5/6

Le bouton est maintenant correctement formaté. Mais il reste à le mettre en forme afin qu'il soit plus attirant.

L'objectif de cet exercice est de **mettre en forme le texte** avec les propriétés que vous avez vu lors des exercices précédents.

- - - - - - - - - - - - - - - - - - -

CLASSES CSS ET IDS

LEÇON 1 Les sélecteurs CSS

1. Tous les éléments HTML sont des sélecteurs CSS

1/20

À ce stade, vous avez déjà utilisé un certain nombre d'éléments HTML comme sélecteurs CSS. Vous avez appliqué des styles sur les balises HTML `<h1></h1>` avec le sélecteur CSS `h1`. Vous avez procédé de la même façon pour les balises `<p></p>` avec le sélecteur `p` et ainsi de suite.

En fait, **tous les éléments HTML peuvent être utilisés comme sélecteurs CSS.** Vous pouvez donc modifier des balises `<ul>`, `<table>` et même l'intégralité de ce qui se trouve dans `<body>` en utilisant respectivement comme sélecteur `ul`, `table` ou `body`.

2. Les sélecteurs multiples en CSS

2/20

Vous savez qu'il est possible d'imbriquer des éléments HTML les uns à l'intérieur des autres, comme ceci :

```
    <div>
        <div>
            <p>J'aime les tacos !</p>
        </div>
```

```
    ‹/div›
```

Comment faire pour modifier uniquement les ‹p› qui sont à l'intérieur des deux ‹div› et non l'intégralité des ‹p› de la page HTML ?

Il suffit d'utiliser le sélecteur CSS suivant :

```
  div div p &#123;
      /*votre code CSS!*/
  }
```

3. Le sélecteur universel en CSS

3/20

Il y a aussi un sélecteur très spécial qui permet d'appliquer des styles CSS à tous les éléments de la page : le sélecteur `*`. Par exemple, si vous tapez :

```
  * {
   border: 2px solid black;
}
```

Vous allez créer une bordure noire continue de 2 pixels de largeur autour de **tous les éléments** de la page HTML.

4. Notion de spécificité des sélecteurs CSS

4/20

L'utilisation de sélecteurs peut paraître un peu complexe mais plus vous les utiliserez, plus vous vous sentirez à l'aise avec.

Dans cet exercice vous allez voir la notion de **spécificité** des sélecteurs CSS.

5. Représentation de la structure d'une page HTML

5/20

Considérez un document HTML comme un arbre avec des éléments partant du tronc principal délimité par `<html></html>`. Les deux premières grosses branches sont `<head>` et `<body>`, puis les branches se multiplient et s'affinent pour donner naissance à des éléments tels que des titres `<h1>` `<h2>` `<h3>`, des tableaux `<table>` ou des paragraphes `<p>`.

Vous trouverez ici un exemple de représentation de la structure d'un document HTML,structure appelée "Document Object Model".

Parents, enfants, frères

Si vous imaginez la balise `<html>` comme étant le tronc de l'arbre, vous pouvez vous représenter ses deux premières branches `<head>` et `<body>` comme ses **enfants**. Ces deux balises sont les enfants de `<html>` et `<html>` est leur élément **parent**. Puisque `<head>` et `<body>` sont tous les deux des enfants directs de `<html>` (il n'y a aucun autre élément entre eux et leur parent), ils sont **frères**.

Note :

En HTML, il n'y a pas de distinction entre l'enfant (A) d'un élément et les enfants (A1, A2, A3, etc.) de cet enfant A. Ainsi, A1, A2, A3 et A sont tous considérés comme les enfants du premier élément parent (il n'y a pas la notion de "petits-enfants").

6. Ajouter des styles à des sélecteurs CSS

6/20

Maintenant que vous avez une meilleure idée de la façon dont les documents HTML sont structurés, il est temps de voir comment faire pour parcourir les branches avec les sélecteurs CSS.

7. Coder un style CSS pour un élément qui est l'enfant d'un autre élément sur une page HTML

7/20

Essayons maintenant quelque chose de plus complexe.

Rappelez-vous que vous pouvez atteindre un élément qui est l'enfant d'un autre élément de cette façon :

```
div div p {
    /* votre CSS */
}
```

En écrivant `div div p &#123;` , nous sélectionnons tous les `‹ p ›` imbriqués quelque part dans la page HTML à l'intérieur d'une `‹ div ›`; cette `‹ div ›` est elle-même imbriquée quelque part à l'intérieur d'une autre `‹ div ›`.

Si vous voulez sélectionner un **enfant direct**, c'est-à-dire un élément qui est **directement** imbriqué à l'intérieur d'un autre élément et **sans aucun autre élément entre les deux**, vous devez utiliser le symbole `**>**`, comme ceci :

```
div > p {
    /* votre CSS*/
}
```

En procédant ainsi, seuls les `‹ p ›` qui sont imbriqués directement à l'intérieur d'une `‹ div ›` seront sélectionnés. Cela ne sélectionnera pas les paragraphes qui sont, par exemple, imbriqués dans des `‹ div ›` qui sont elles-mêmes imbriquées dans d'autres `‹ div ›`.

8. Utiliser les CLASS et les ID pour des styles spécifiques sur votre page HTML

8/20

Comme vous venez de le voir, certains sélecteurs en "annulent" d'autres quand ils ont un plus grand degré de **spécificité**.

Cependant, deux autres types de sélecteurs sont encore plus spécifiques que ceux déjà vu : les **class** et les **id**.

9. Fonctionnement du sélecteur CLASS en CSS

9/20

Le sélecteur `**class**` est utile quand vous souhaitez appliquer le même style à plusieurs éléments HTML différents. Plutôt que d'appliquer, en la répétant, la même règle aux différents sélecteurs, vous assignez **une seule et même class aux éléments HTML concernés**, puis définissez le style de cette class dans l'onglet CSS.

Les "class" sont assignées aux éléments HTML grâce à l'attribut `class` suivi du signe `=`, comme ceci :

```
<h1 class="NomDeLaClass"></h1>
<p class="NomDeLaClass"></p>
```

Dans le fichier .css, la class s'écrit comme ceci :

```
.NomDeLaClass {
    color: red;
}
```

N'oubliez pas le point "**.**" devant le nom de votre class.

Dans l'exemple ci-dessus, le titre h1 et le paragraphe p seront tous les deux en rouge.

10. Fonctionnement du sélecteur ID en CSS

10/20

Le sélecteur `**id**` est utile quand vous souhaitez appliquer un style à **un seul et unique** élément HTML.

Si le sélecteur `class` a vocation à être réutilisé plusieurs fois, chaque sélecteur `id` ne sera utilisé qu'une seule fois sur un site web.

La syntaxe dans le fichier HTML est la suivante :

```
<h1 id="NomIdUnique"></h1>
```

Dans le fichier .css, le sélecteur `id` s'écrit comme ceci :

```
#NomIdUnique {
    color: red;
}
```

N'oubliez pas le dièse "#" devant le nom de votre id.

11. Combiner CLASS et ID sur une feuille de style CSS

11/20

Dans cet exercice, vous allez rassembler vos nouvelles connaissances et utiliser les deux nouveaux sélecteurs `.class` et `#id`.

12. Utiliser un sélecteur de pseudo-classe en CSS

12/20

Vous venez de découvrir les sélecteurs de classe. Il est temps à présent d'aborder **les sélecteurs de pseudo-classe**.

Un sélecteur de pseudo-classe est un moyen d'accéder à un élément HTML qui se trouve dans un état particulier, par exemple les liens quand ils sont **survolés**, **cliqués** ou encore **visités**.

Il existe un grand nombre de pseudo-classes, vous allez voir ici les plus communs.

La syntaxe d'un sélecteur pseudo-classe dans un fichier CSS est la suivante :

```
sélecteur:pseudo-classe {
        propriété: valeur;
    }
```

N'oubliez pas les deux points `:` entre le sélecteur et la pseudo-classe

13. Différents styles pour un lien cliqué, survolé ou visité avec les attributs Link, Visited ou Hover

13/20

Voici 3 sélecteurs de pseudo-classe pour les liens (il y en a d'autres). Ils permettent de définir le style pour :

`a:link` : un lien non cliqué
`a:visited` : un lien qui a été cliqué
`a:hover` : un lien survolé par la souris et non cliqué

14. Sélecteur de pseudo-classe first-child en CSS

14/20

Un autre sélecteur de pseudo-classe utile est `:first-child`. Il est utilisé pour appliquer un style uniquement aux éléments qui sont le premier enfant de leur parent.

15. Sélecteur de pseudo-classe nth-child(x) en CSS

15/20

En fait, vous pouvez sélectionner n'importe quel enfant d'un élément parent en utilisant le sélecteur de pseudo-classe `:nth-child(x)` où "x" correspond à la place de l'enfant dans la descendance de l'élément parent.

Par exemple :

```
p:nth-child(2) {
    color: red;
}
```

Cela va sélectionner les paragraphes `p` qui sont le deuxième enfant de leur parent.

L'élément représentant **l'enfant que l'on veut sélectionner** est spécifié juste avant `:nth-child`. Dans l'exemple ci-dessus, l'élément ciblé est un `p`. Son élément parent est celui qui le contient.

16. Rappel : Les sélecteurs multiples en CSS

16/20

Cet exercice est un rappel sur les sélecteurs multiples.

Rappelez-vous comment utiliser les sélecteurs pour cibler des éléments imbriqués les uns dans les autres. Si vous avez un paragraphe `p` dans un `div`, ce `div` étant lui-même imbriqué dans un autre `div`, vous pouvez cibler le paragraphe `p` en question comme cela :

```
  div div p {
        /*Votre CSS*/
    }
```

Ce sélecteur va appliquer le style à tous les paragraphes imbriqués dans deux div, sans toucher aux paragraphes qui ne correspondent pas à ces critères

17. Rappel : le sélecteur "class" en CSS

17/20

Cet exercice est un rappel sur le **sélecteur "class"**. Il permet d'attribuer un même style à différents éléments (que ce soit à un titre, un paragraphe, un lien ou un tableau).

18. Rappel : le sélecteur "id" en CSS

18/20

Cet exercice est un rappel sur le **sélecteur "id"**. Il permet de cibler un élément HTML unique dans un site web.

19. Rappel : les pseudo-classes 1 en CSS

19/20

Cet exercice est un premier rappel sur les pseudo-classes.

20. Rappel : les pseudo-classes 2 en CSS

20/20

Cet exercice est un second rappel sur les sélecteurs de pseudo-classes.

- - - - - - - - - - - - - - - - - - -

CLASSES CSS ET IDS

LEÇON 1 Les sélecteurs CSS

1. Tous les éléments HTML sont des sélecteurs CSS

1/20

À ce stade, vous avez déjà utilisé un certain nombre d'éléments HTML comme sélecteurs CSS. Vous avez appliqué des styles sur les balises HTML `<h1></h1>` avec le sélecteur CSS `h1`. Vous avez procédé de la même façon pour les balises `<p></p>` avec le sélecteur `p` et ainsi de suite.

En fait, **tous les éléments HTML peuvent être utilisés comme sélecteurs CSS.** Vous pouvez donc modifier des balises `<ul>`, `<table>` et même l'intégralité de ce qui se trouve dans `<body>` en utilisant respectivement comme sélecteur `ul`, `table` ou `body`.

2. Les sélecteurs multiples en CSS

2/20

Vous savez qu'il est possible d'imbriquer des éléments HTML les uns à l'intérieur des autres, comme ceci :

```
    <div>
        <div>
            <p>J'aime les tacos !</p>
        </div>
```

```
    </div>
```

Comment faire pour modifier uniquement les ‹p› qui sont à l'intérieur des deux ‹div› et non l'intégralité des ‹p› de la page HTML ?

Il suffit d'utiliser le sélecteur CSS suivant :

```
  div div p &#123;
      /*votre code CSS!*/
  }
```

3. Le sélecteur universel en CSS

3/20

Il y a aussi un sélecteur très spécial qui permet d'appliquer des styles CSS à tous les éléments de la page : le sélecteur `*`. Par exemple, si vous tapez :

```
* {
  border: 2px solid black;
}
```

Vous allez créer une bordure noire continue de 2 pixels de largeur autour de **tous les éléments** de la page HTML.

4. Notion de spécificité des sélecteurs CSS

4/20

L'utilisation de sélecteurs peut paraître un peu complexe mais plus vous les utiliserez, plus vous vous sentirez à l'aise avec.

Dans cet exercice vous allez voir la notion de **spécificité** des sélecteurs CSS.

5. Représentation de la structure d'une page HTML

Considérez un document HTML comme un arbre avec des éléments partant du tronc principal délimité par `<html></html>`. Les deux premières grosses branches sont `<head>` et `<body>`, puis les branches se multiplient et s'affinent pour donner naissance à des éléments tels que des titres `<h1>` `<h2>` `<h3>`, des tableaux `<table>` ou des paragraphes `<p>`.

Vous trouverez ici un exemple de représentation de la structure d'un document HTML,structure appelée "Document Object Model".

Parents, enfants, frères

Si vous imaginez la balise `<html>` comme étant le tronc de l'arbre, vous pouvez vous représenter ses deux premières branches `<head>` et `<body>` comme ses **enfants**. Ces deux balises sont les enfants de `<html>` et `<html>` est leur élément **parent**. Puisque `<head>` et `<body>` sont tous les deux des enfants directs de `<html>` (il n'y a aucun autre élément entre eux et leur parent), ils sont **frères**.

Note :

En HTML, il n'y a pas de distinction entre l'enfant (A) d'un élément et les enfants (A1, A2, A3, etc.) de cet enfant A. Ainsi, A1, A2, A3 et A sont tous considérés comme les enfants du premier élément parent (il n'y a pas la notion de "petits-enfants").

6. Ajouter des styles à des sélecteurs CSS

Maintenant que vous avez une meilleure idée de la façon dont les documents HTML sont structurés, il est temps de voir comment faire pour parcourir les branches avec les sélecteurs CSS.

7. Coder un style CSS pour un élément qui est l'enfant d'un autre élément sur une page HTML

7/20

Essayons maintenant quelque chose de plus complexe.

Rappelez-vous que vous pouvez atteindre un élément qui est l'enfant d'un autre élément de cette façon :

```
div div p {
    /* votre CSS */
}
```

En écrivant `div div p &#123;` , nous sélectionnons tous les `‹p›` imbriqués quelque part dans la page HTML à l'intérieur d'une `‹div›`; cette `‹div›` est elle-même imbriquée quelque part à l'intérieur d'une autre `‹div›`.

Si vous voulez sélectionner un **enfant direct**, c'est-à-dire un élément qui est **directement** imbriqué à l'intérieur d'un autre élément et **sans aucun autre élément entre les deux**, vous devez utiliser le symbole `**>**`, comme ceci :

```
div > p {
    /* votre CSS*/
}
```

En procédant ainsi, seuls les `‹p›` qui sont imbriqués directement à l'intérieur d'une `‹div›` seront sélectionnés. Cela ne sélectionnera pas les paragraphes qui sont, par exemple, imbriqués dans des `‹div›` qui sont elles-mêmes imbriquées dans d'autres `‹div›`.

8. Utiliser les CLASS et les ID pour des styles spécifiques sur votre page HTML

8/20

Comme vous venez de le voir, certains sélecteurs en "annulent" d'autres quand ils ont un plus grand degré de **spécificité**.

Cependant, deux autres types de sélecteurs sont encore plus spécifiques que ceux déjà vu : les **class** et les **id**.

9. Fonctionnement du sélecteur CLASS en CSS

9/20

Le sélecteur `**class**` est utile quand vous souhaitez appliquer le même style à plusieurs éléments HTML différents. Plutôt que d'appliquer, en la répétant, la même règle aux différents sélecteurs, vous assignez **une seule et même class aux éléments HTML concernés**, puis définissez le style de cette class dans l'onglet CSS.

Les "class" sont assignées aux éléments HTML grâce à l'attribut `class` suivi du signe `=`, comme ceci :

```
<h1 class="NomDeLaClass"></h1>
<p class="NomDeLaClass"></p>
```

Dans le fichier .css, la class s'écrit comme ceci :

```
.NomDeLaClass {
    color: red;
}
```

N'oubliez pas le point "**.**" devant le nom de votre class.

Dans l'exemple ci-dessus, le titre h1 et le paragraphe p seront tous les deux en rouge.

10. Fonctionnement du sélecteur ID en CSS

10/20

Le sélecteur `**id**` est utile quand vous souhaitez appliquer un style à **un seul et unique** élément HTML.

Si le sélecteur `class` a vocation à être réutilisé plusieurs fois, chaque sélecteur `id` ne sera utilisé qu'une seule fois sur un site web.

La syntaxe dans le fichier HTML est la suivante :

```
‹h1 id="NomIdUnique"›‹/h1›
```

Dans le fichier .css, le sélecteur `id` s'écrit comme ceci :

```
#NomIdUnique {
    color: red;
}
```

N'oubliez pas le dièse "#" devant le nom de votre id.

11. Combiner CLASS et ID sur une feuille de style CSS

11/20

Dans cet exercice, vous allez rassembler vos nouvelles connaissances et utiliser les deux nouveaux sélecteurs `.class` et `#id`.

12. Utiliser un sélecteur de pseudo-classe en CSS

12/20

Vous venez de découvrir les sélecteurs de classe. Il est temps à présent d'aborder **les sélecteurs de pseudo-classe**.

Un sélecteur de pseudo-classe est un moyen d'accéder à un élément HTML qui se trouve dans un état particulier, par exemple les liens quand ils sont **survolés**, **cliqués** ou encore **visités**.

Il existe un grand nombre de pseudo-classes, vous allez voir ici les plus communs.

La syntaxe d'un sélecteur pseudo-classe dans un fichier CSS est la suivante :

```
sélecteur:pseudo-classe {

        propriété: valeur;

    }
```

N'oubliez pas les deux points `:` entre le sélecteur et la pseudo-classe

13. Différents styles pour un lien cliqué, survolé ou visité avec les attributs Link, Visited ou Hover

13/20

Voici 3 sélecteurs de pseudo-classe pour les liens (il y en a d'autres). Ils permettent de définir le style pour :

`a:link` : un lien non cliqué
`a:visited` : un lien qui a été cliqué
`a:hover` : un lien survolé par la souris et non cliqué

14. Sélecteur de pseudo-classe first-child en CSS

14/20

Un autre sélecteur de pseudo-classe utile est `:first-child`. Il est utilisé pour appliquer un style uniquement aux éléments qui sont le premier enfant de leur parent.

15. Sélecteur de pseudo-classe nth-child(x) en CSS

15/20

En fait, vous pouvez sélectionner n'importe quel enfant d'un élément parent en utilisant le sélecteur de pseudo-classe `:nth-child(x)` où "x" correspond à la place de l'enfant dans la descendance de l'élément parent.

Par exemple :

```
p:nth-child(2) {

    color: red;

}
```

Cela va sélectionner les paragraphes `p` qui sont le deuxième enfant de leur parent.

L'élément représentant **l'enfant que l'on veut sélectionner** est spécifié juste avant `:nth-child`. Dans l'exemple ci-dessus, l'élément ciblé est un `p`. Son élément parent est celui qui le contient.

16. Rappel : Les sélecteurs multiples en CSS

16/20

Cet exercice est un rappel sur les sélecteurs multiples.

Rappelez-vous comment utiliser les sélecteurs pour cibler des éléments imbriqués les uns dans les autres. Si vous avez un paragraphe `p` dans un `div`, ce `div` étant lui-même imbriqué dans un autre `div`, vous pouvez cibler le paragraphe `p` en question comme cela :

```
  div div p {


        /*Votre CSS*/


    }
```

Ce sélecteur va appliquer le style à tous les paragraphes imbriqués dans deux div, sans toucher aux paragraphes qui ne correspondent pas à ces critères

17. Rappel : le sélecteur "class" en CSS

17/20

Cet exercice est un rappel sur le **sélecteur "class"**. Il permet d'attribuer un même style à différents éléments (que ce soit à un titre, un paragraphe, un lien ou un tableau).

18. Rappel : le sélecteur "id" en CSS

18/20

Cet exercice est un rappel sur le **sélecteur "id"**. Il permet de cibler un élément HTML unique dans un site web.

19. Rappel : les pseudo-classes 1 en CSS

19/20

Cet exercice est un premier rappel sur les pseudo-classes.

20. Rappel : les pseudo-classes 2 en CSS

20/20

Cet exercice est un second rappel sur les sélecteurs de pseudo-classes.

- - - - - - - - - - - - - - - - - - -

LEÇON 2 Trier les propriétés et les Styles de votre CSS

1. Utiliser des sélecteurs CSS pour regrouper des éléments HTML

1/7

Dans les quelques exercices qui suivent vous allez utilisez l'ensemble des sélecteurs que vous connaissez pour regrouper de façon distincte des éléments HTML, comme l'exemple qui est présenté ici.

2. Utiliser des DIV pour diviser le contenu de votre page HTML

2/7

Pour commencer, vous aurez besoin de créer la structure HTML de base.

3. Personnaliser les DIV avec le CSS

3/7

Vous allez appliquer une mise en forme de base aux `div`. Vous utiliserez le sélecteur `class` plus tard afin de distinguer les `div` selon leur type de contenu (ami, famille ou ennemi).

Note :

Par défaut, les `div` sont vides dans le fichier `index.html`. N'oubliez pas de récupérer leur contenu selon ce que vous avez inséré à l'exercice précédent. Un simple copier-coller fera l'affaire.

4. Mettre des attributs CLASS sur des DIV pour définir le style CSS

4/7

Dans cet exercice, vous allez classifier les `div` selon leur attribut `class` qui pourra prendre comme valeur `ami`, `famille` ou `travail`.

Note :

Par défaut, les `div` sont vides dans le fichier `index.html`. N'oubliez pas de récupérer leur contenu selon ce que vous avez inséré à l'exercice précédent. Un simple copier-coller fera l'affaire.

5. Utiliser le sélecteur "class"

5/7

Dans cet exercice, vous allez appliquer un style particulier aux `div` selon leur attribut `class`.

Note :

Par défaut, les `div` sont vides dans le fichier `index.html`. N'oubliez pas de récupérer leur contenu et leur attribut `class` selon ce que vous avez inséré à l'exercice précédent. Un simple copier-coller fera l'affaire.

6. Utiliser le sélecteur "id"

6/7

Dans cet exercice, vous allez appliquer un style particulier pour le `div` qui contient votre **meilleur ami**.

Note :

Par défaut, les `div` sont vides dans le fichier `index.html`. N'oubliez pas de récupérer leur contenu et leur attribut `class` selon ce que vous avez inséré à l'exercice précédent. Un simple copier-coller fera l'affaire.

7. Personnaliser votre "meilleur ami"

7/7

Dans cet exercice, vous allez appliquer un style particulier pour le `div` qui contient votre **meilleur ami**.

Note :

Par défaut, les `div` sont vides dans le fichier `index.html`. N'oubliez pas de récupérer leur contenu et leur attribut `class` et `id` selon ce que vous avez inséré à l'exercice précédent. Un simple copier-coller fera l'affaire.

- - - - - - - - - - - - - - - - - -

Positionnement des éléments avec le CSS

LEÇON 1 Comment positionner un élément HTML avec du CSS

1. Différence entre PADDING MARGIN et BORDER en CSS

1/25

Maîtriser la position des éléments HTML vous permet d'avoir un contrôle très précis sur le rendu final de vos pages web. Les éléments de type bloc (`div`, `h1`, `table`, `p`, etc.) ne s'empileront plus forcément les uns sur les autres (à moins que vous ne le souhaitiez).

Chaque élément HTML agit donc comme un conteneur qui peut contenir soit d'autres éléments html soit du texte brut.

Cet exercice propose une illustration du modèle de boîte et les propriétés CSS associées : border, padding, margin. Vous avez utilisé `border` dans les exercices précédents, vous verrez les autres dans les exercices suivants.

2. Utiliser la propriété DISPLAY pour afficher un élément HTML

2/25

Il est possible de changer le comportement par défaut de l'affichage d'un élément html au moyen de la propriété `**display**`. Cette propriété peut prendre plusieurs valeurs :

> **block :** les éléments "block" (`div p h1-h5 table`) prennent toute la largeur disponible par défaut et se positionnent donc sous l'élément précédent. Vous pouvez leur appliquer des dimensions (width, height) et des marges (margin, padding);
> **inline-block :**les éléments "inline-block" ont les mêmes caractéristiques que les "block" mais peuvent être disposés sur une même ligne et pas les uns en dessous des autres;
> **inline :**les éléments "inline" (`a span strong em`) prennent les dimensions du texte qu'il contiennent. Il n'est pas possible de leur donner des dimensions propres (width, height) ni de leur appliquer des marges externes hautes et basses (margin-top,

margin-bottom). Mais ils peuvent avoir des marges internes (padding);
none :les éléments "none" et leur contenu disparaissent complètement de la page;Il est possible de changer le comportement par défaut de l'affichage d'un élément html au moyen de la propriété `**display**`. Cette propriété peut prendre plusieurs valeurs :
Block : les éléments "block" (`div p h1-h5 table`) prennent toute la largeur disponible par défaut et se positionnent donc sous l'élément précédent. Vous pouvez leur appliquer des dimensions (width, height) et des marges (margin, padding);
Inline-block : les éléments "inline-block" ont les mêmes caractéristiques que les "block" mais peuvent être disposés sur une même ligne et pas les uns en dessous des autres;
Inline : les éléments "inline" (`a span strong em`) prennent les dimensions du texte qu'il contiennent. Il n'est pas possible de leur donner des dimensions propres (width, height) ni de leur appliquer des marges externes hautes et basses (margin-top, margin-bottom). Mais ils peuvent avoir des marges internes (padding);
None : les éléments "none" et leur contenu disparaissent complètement de la page;

3. La propriété Inline-block en CSS

3/25

Si vous ne constatez pas beaucoup de différence, c'est normal. Les `div` sont par défaut des éléments de type `block`.

Comme indiqué, tout élément qui se positionne comme un bloc viendra automatiquement occuper toute la largeur de la page, quel que soit le contenu (ou le peu de contenu) que vous aurez mis à l'intérieur.

Mais, si vous spécifiez un affichage de type `inline-block`, les éléments html vont venir se positionner les uns à côté des autres.

4. Afficher une DIV HTML avec le type Inline sans le Block

4/25

La valeur `inline-block` vous permet donc de placer plusieurs blocs sur la même ligne. La valeur `inline` positionne aussi tous vos éléments les uns à côté des autres mais pas comme des blocs : ils ne conservent pas leur dimension.

5. Utiliser la valeur None pour la propriété DISPLAY

5/25

La propriété "display" peut aussi prendre comme valeur "none" qui permet de ne pas afficher un élément sur la page web. Un élément HTML qui a cette valeur ne sera pas affiché sur la page et la place qu'il occupait sera libérée afin que les éléments HTML voisins puissent se positionner par rapport à l'emplacement libéré. La propriété "display" peut aussi prendre comme valeur "none" qui permet de ne pas afficher un élément sur la page web. Un élément HTML qui a cette valeur ne sera pas affiché sur la page et la place qu'il occupait sera libérée afin que les éléments HTML voisins puissent se postionner par rapport à l'emplacement libéré.

6. Marges et bordures en CSS

6/25

Maintenant que vous connaissez les propriétés d'affichage et le modèle de boîte, plongeons-nous dans les détails concernant le comportement individuel des boîtes.

La marge externe : `margin` est l'espace entourant un élément. Plus la marge d'un élément est grande, plus la distance entre cet élément et ceux qui l'entourent sera grande. Nous pouvons ajuster la marge pour rapprocher ou éloigner nos éléments HTML les uns des autres;
La marge interne : `padding` est l'espace entre le contenu d'un élément et sa bordure. Nous pouvons ajuster cette valeur avec CSS pour placer les bordures plus ou moins loin du contenu;
La bordure : `border` est la bordure de chaque élément (entre `margin` et `padding`). C'est ce que l'on rend visible en utilisant la propriété `border`;

Le contenu : `content` est le contenu de la boîte. Par exemple, si on parle d'un élément `p`, le contenu sera le texte du paragraphe.

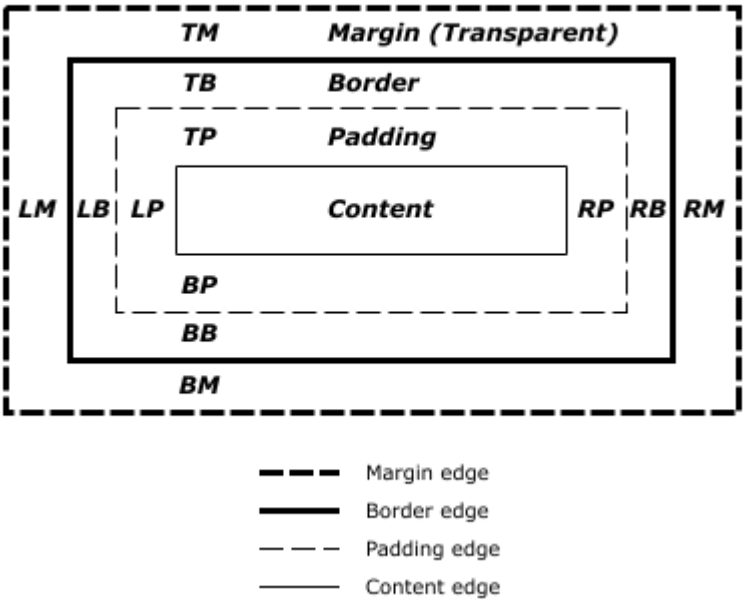
Dans l'illustration de cet exercice, il y a des abbréviations **TM, TB, TP** ils correspondent respectivement aux propriétés "top margin", "top border" et "top padding", soit "marge externe du haut", "bordure du haut" et "marge interne du haut.

Vous verrez qu'il est tout à fait possible de spécifier des valeurs différentes pour le haut, le bas, la gauche et la droite concernant toutes ces propriétés.Maintenant que vous connaissez les propriétés d'affichage et le modèle de boîte, plongeons-nous dans les détails concernant le comportement individuel des boîtes.

La marge externe : `margin` est l'espace entourant un élément. Plus la marge d'un élément est grande, plus la distance entre cet élément et ceux qui l'entourent sera grande. Nous pouvons ajuster la marge pour rapprocher ou éloigner nos éléments HTML les uns des autres;
La marge interne : `padding` est l'espace entre le contenu d'un élément et sa bordure. Nous pouvons ajuster cette valeur avec CSS pour placer les bordures plus ou moins loin du contenu;
La bordure : `border` est la bordure de chaque élément (entre `margin` et `padding`). C'est ce que l'on rend visible en utilisant la propriété `border`;
Le contenu : `content` est le contenu de la boîte. Par exemple, si on parle d'un élément `p`, le contenu sera le texte du paragraphe;

Dans l'illustration de cet exercice, il y a des abréviations **TM, TB, TP** ils correspondent respectivement aux propriétés "top margin", "top border" et "top padding", soit "marge externe du haut", "bordure du haut" et "marge interne du haut.

Vous verrez qu'il est tout à fait possible de spécifier des valeurs différentes pour le haut, le bas, la gauche et la droite concernant toutes ces propriétés.



7. Marges externes en CSS avec la propriété MARGIN

7/25

Commençons par les marges externes (`margin`). Ajuster les marges externes va non seulement modifier la position de l'élément relativement à tous les autres éléments de la page, mais aussi relativement aux "frontières" du document HTML.

Dans cet exercice, les tirets rouges représentent les bordures des éléments `html` et `body`. Vous constatez que par défaut, `body` possède déjà des marges sans qu'il y ait besoin de le spécifier. C'est le cas de nombreux éléments qui sont de type `block` par défaut (`body p h1-h5 ...`) mais ce n'est pas le cas du `div` qui est une balise générique.

Le `div` est donc collé aux bordures de `body`.

8. Marge externe : haut, bas, gauche, droite en CSS

Si vous voulez ajuster une marge en particulier, vous pouvez le faire comme cela :

```
   margin-top: /* Valeur de la marge externe du haut */
    margin-right: /* Valeur de la marge externe de droite */
    margin-bottom: /* Valeur de la marge externe du bas */
    margin-left: /* Valeur de la marge externe de gauche */
```

Vous pouvez aussi définir toutes les marges externes d'un élément avec la **notation raccourcie**. Il suffit juste de partir du haut et de tourner dans le sens des aiguilles d'une montre (Haut Droite Bas Gauche), comme cela :

```
   margin: 1px 2px 3px 4px
```

9. Les bordures en CSS

Vous utilisez la propriété `border` depuis de nombreux exercices sous la forme suivante :

```
border: 1px solid black;
```

Cette syntaxe est en fait une **notation raccourcie** pour affecter à chaque bordure d'un élément html (haut, bas, gauche, droite). Il est tout à fait possible de préciser à quel côté vous souhaitez appliquer un style spécifique. Il suffit d'utiliser les propriétés suivantes :

```
   border-top:
    border-bottom:
    border-left:
    border-right:
```

10. Marge interne ou Padding en CSS

Tout comme les marges externes `margin`, les marges internes `padding` peuvent être définies spécifiquement pour chaque côté d'un élément html.

Si vous voulez ajuster une marge interne en particulier, vous pouvez le faire comme cela :

```
padding-top: /* Valeur de la marge interne du haut */
 padding-right: /* Valeur de la marge interne de droite */
 padding-bottom: /* Valeur de la marge interne du bas */
 padding-left: /* Valeur de la marge interne de gauche */
```

Vous pouvez aussi définir toutes les marges internes d'un élément avec la **notation raccourcie**. Il suffit juste de partir du haut et de tourner dans le sens des aiguilles d'une montre (Haut Droite Bas Gauche), comme cela :

```
padding: 1px 2px 3px 4px
```

Astuce: Avec la notation raccourcie vous pouvez aussi appliquer des valeurs nulles si vous n'avez pas besoin de marge sur chaque côté. Attention, n'oubliez pas que l'ordre de déclaration des valeurs se fait dans le sens horaire : haut, droite, bas, gauche. L'exemple suivant applique une marge interne de 5px à droite et 10px à gauche. Les autres marges sont nulles (**quand la valeur est nulle, pas besoin de préciser l'unité**) :
`padding: 0 5px 0 10px`

11. Utiliser des valeurs négatives pour les marges d'un contenu HTML

Affecter une marge externe gauche (`margin-left`) de `50px` va déplacer l'élément en question dans la direction opposée, ici à droite.

En CSS, il est aussi possible de donner des valeurs négatives aux propriétés `margin` et `padding`. Ainsi, appliquer un `margin-left` de `-50px` va déplacer l'élément en question de 50px vers la gauche.

12. Rappel : Les marges internes, externes et les bordures en CSS

Cet exercice est un rappel sur ce que vous venez de voir concernant les marges externes, les marges internes et les bordures.

13. Le positionnement flottant (droite) en CSS

13/25

Dans les exercices précédents, vous avez vu différentes façons d'affecter le positionnement d'un élément html en utilisant les propriétés `display` ou `margin`.

Il existe cependant d'autres propriétés CSS pour positionner des éléments de manière plus "radicale": `float` et `position`.

Le flux courant

Comprendre la notion de "flux" est primordiale pour comprendre comment `float` et `position` peuvent affecter le rendu final de vos pages web.

Le "flux" est le comportement par défaut d'affichage de vos éléments html. Il suit l'ordre de déclaration des éléments tel que vous l'avez défini dans la structure de votre document HTML.

Si les propriétés `padding` et `margin` n'affectent pas le positionnement des éléments HTML dans le flux, c'est le cas des propriétés `float` et `position`.

> **le positionnement flottant** (`float: left|right`) : permet de faire "flotter" un élément à gauche ou à droite de son élément conteneur, le contenu qui suit vient naturellement **englober** l'élément flottant qui est retiré du "flux courant";

14. Le positionnement flottant (gauche) en CSS

14/25

Dans cet exercice, vous allez repositionner l'élément `div` tel qu'il était au départ, c'est-à-dire sur la gauche de la page.

15. Le positionnement flottant (gauche et droite) en CSS

15/25

Dans cet exercice, vous allez utiliser les deux valeurs de `float` afin de positionner deux éléments HTML côte à côte.

16. Utiliser la propriété CLEAR pour libérer une DIV

16/25

En utilisant la propriété `float`, vous pourrez rencontrer des situations où le comportement de vos éléments HTML n'est pas celui que vous attendiez. Certains éléments se retrouvant les uns au-dessus/dessous des autres.

C'est le cas, dans cet exercice, de l'élément `div` avec l'id `footer`. Il devrait se trouver en bas de page. Mais il se retrouve tout en haut car les autres éléments flottent et il a donc pris leur place.

Il existe une propriété CSS qui permet d'annuler ce comportement et de repositionner un élément HTML à son emplacement par défaut :

```
clear: (left|right|both)
```

Si vous dites à un élément `clear: left`, il se déplacera immédiatement en dessous de tout élément flottant sur la gauche. La valeur `right` a le même effet, mais pour des éléments flottant sur la droite.. Si vous lui dites `clear: both`, les éléments flottant à gauche et à droite seront pris en compte.

17. Le positionnement statique en CSS

17/25

Vous venez de voir la première propriété (`float`) pour positionner vos éléments HTML de façon "radicale".

Il en existe une deuxième (`position`) et qui affecte aussi le flux courant selon les valeurs qu'elle peut prendre.

> Positionnement **statique** (`position: static`) - n'affecte pas le flux. En fait c'est la valeur par défaut de tout élément HTML qui est dans le flux courant. Evidemment, cette valeur n'affecte pas le flux courant, elle est utilisée pour ramener un élément HTML qui était hors du flux à sa valeur par défaut;
> Positionnement **relatif** (`position: relative`) - n'affecte pas le flux. Cette valeur permet de décaler un élément dont la position est calculée par rapport à sa position de référence. L'élément reste dans le flux. Cela sert pour effectuer des ajustements de position, comme avec `margin` par exemple;
> Positionnement **absolu** (`position: absolute`) - affecte le flux. Un élément est positionné en absolu par rapport, soit au premier élément ancêtre lui-même positionné (avec `position`), sinon à l'élément `html`. Un tel élément sort complètement du flux courant, là où `float` positionne un élément par rapport à son parent direct. De plus, il ne prend plus toute la largeur disponible par défaut;
> Positionnement **fixe** (`position: fixed`) - affecte le flux.
> Très similaire à la valeur `absolute`, la différence vient du fait que si la position est précisée (`top, bottom, left, right`), la référence sera la fenêtre du navigateur. De plus, comme son nom l'indique, l'élément restera fixé à sa position, même en cas de défilement vertical;

Important :

Pour les valeurs `absolute, relative, fixed` si vous ne précisez pas les valeurs de décalage (ex: `top: 5px; left: 5px; bottom: 10px; right: 10px;`) par rapport au nouveau point de référence, alors les éléments resteront à leur emplacement par défaut dans le flux.

Les deux `div` de cet exercice sont par défaut en position `static`. Dans les exercices suivants, vous allez modifier ce comportement par défaut avec les différentes valeurs vues ci-dessus.

18. Le positionnement absolu en CSS

Dans cet exercice, vous allez attribuer un positionnement absolu au `div` enfant.

Son parent n'a pas de `position` définie, donc le `div` enfant se positionnera par rapport à l'élément `html`.

19. Le positionnement relatif en CSS

19/25

Dans cet exercice, vous allez attribuer un positionnement relatif au `div` enfant.

Il s'agit donc de dire à l'élément concerné de se placer relativement à l'endroit où il se trouve par défaut.

20. Le positionnement fixe en CSS

20/25

Dans cet exercice, vous allez attribuer un positionnement fixe au `div` enfant.

Il s'agit donc de dire à l'élément concerné de se placer à un endroit fixe par rapport aux limites de la page web.

21. Rappel 1 sur 5

21/25

Les exercices qui suivent vont vous permettre de réviser les notions essentielles sur le positionnement en CSS.

Sur la base de la structure HTML et du CSS qui vous sont données, l'objectif sera de repositionner les blocs afin de créer une mise en page de base avec un en-tête, un menu, le corps de page et le pied de page

Ce sera aussi l'occasion de voir quelques astuces supplémentaires utiles à la création d'un layout en CSS.

22. Rappel 2 sur 5

22/25

L'objectif de cet exercice est de positionner de manière permanente le `div` "header" en haut de page.

23. Rappel 3 sur 5

23/25

Le `div` "header" est maintenant fixé. Le problème est qu'un élément fixé ne prend plus toute la largeur disponible par défaut. Il va falloir préciser que l'on souhaite que le "header" prenne toute la largeur disponible.

Vous remarquez également que le `div` "header" se trouve visuellement en-dessous des autres éléments HTML. Cela arrive souvent quand on manipule des éléments hors du flux courant. Or il est préférable que le "header" soit visible par dessus tout autre élément HTML étant donné qu'il est fixé en haut de page. Vous allez utiliser la propriété `z-index` pour cela.

Cette propriété prend comme valeur un nombre entier. Plus il est élevé (par rapport aux autres z-index), plus votre élément HTML se trouvera au dessus des autres, comme un système de calques qui se superposent.

24. Rappel 4 sur 5

24/25

Dans cet exercice, vous allez mettre les `div` "menu" et "body" côte à côte.

Il y a deux options possibles :

- la technique `float`;
- la technique `inline-block`;

Ici, vous allez utiliser la deuxième technique. Elle a pour avantage que les éléments HTML resteront dans le flux, donc le `div` "footer" qui est en-dessous restera à sa place (pas besoin de la propriété `clear`). Mais elle a aussi des inconvénients que vous allez devoir corriger par la suite.

25. Rappel 5 sur 5

Les `div` "menu" et "body" devraient se trouver côte à côte grâce à `inline-block`. Mais ce n'est pas le cas pour deux raisons :

- Entre des éléments HTML positionnés en `inline-block`, il peut y avoir un espace vide dû au retour à la ligne dans le code HTML. Pour y remédier, il suffit de mettre des **commentaires HTML** entre les éléments en question;
- Selon les navigateurs, la largeur `width` d'un élément peut prendre en compte soit la largeur de son contenu, soit cette largeur plus le `padding` et la `border`. Un élément peut donc prendre plus de largeur que ce qui est précisé avec `width`. Pour que `width` soit égal à la largeur de l'ensemble il faut utiliser la propriété `box-sizing: border-box`;

- - - - - - - - - - - - - - - - - -

LEÇON 2 Créer un layout CSS

1. Créer un layout de base avec HTML et CSS

Maintenant que vous connaissez les **grands principes du positionnement en CSS**, vous allez vous en servir afin de créer un **layout** de base.
C'est-à-dire un modèle de page qui pourra servir comme structure de base pour un site web.

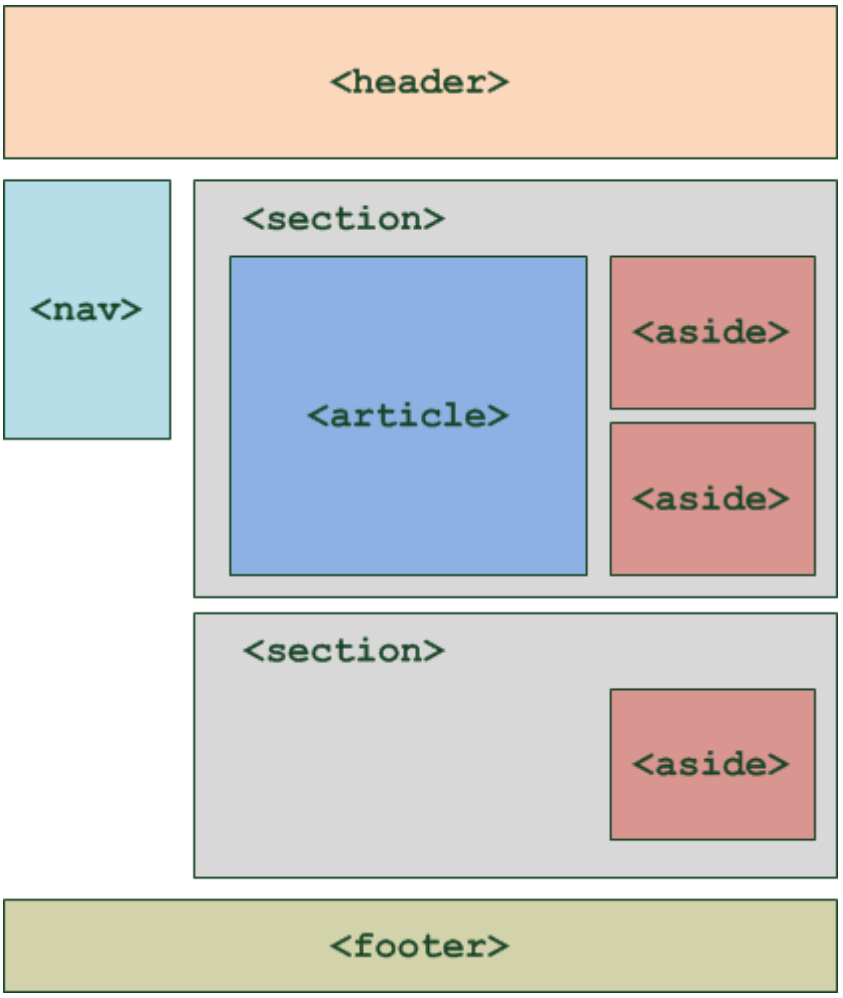
Vous allez reproduire en HTML/CSS le schéma illustré ci-contre.

Quelques précisions avant de commencer :

- Vous connaissez la balise `div` pour créer des blocs et structurer une page web;
- Le layout que vous allez mettre en page possède une structure HTML utilisant non pas la balise générique `div` mais de nouvelles balises HTML5 qui ont un sens sémantique, tout comme la balise `p` pour les paragraphes;

header : pour l'en-tête d'une page web;
footer : pour le pied de page;
nav : pour la navigation, contient généralement les liens du menu;
section : pour délimiter les parties thématiques d'un site web. Une `section` peut avoir elle-même un `header` ou un `footer`;
article : pour présenter du contenu;
aside : pour présenter des informations additionnelles;

Il n'y a pas de restrictions à proprement parler quant à l'organisations des balises : un article peut lui-même avoir un header et un footer, etc.



2. La structure de base HTML et CSS pour créer un layout

La structure HTML de base est déjà en place, il n'y a pas besoin d'y toucher. Le code CSS de base est aussi en place.

Pour l'instant cela ne ressemble pas à un layout. Tous les blocs sont par défaut en gris, il est donc difficile de les distinguer.

Dans cet exercice vous allez mettre un peu de couleur pour y voir plus clair.

En fait, toutes les couleurs sont déjà définies dans le code CSS mais elles ne sont pas interprétées car elles sont **écrasées par les valeurs indiquées lignes 3 et 7**.

Note :

Dans un code CSS, les propriétés sont **interprétées dans l'ordre dans lequel elles sont déclarées**. Si une même propriété est définie plusieurs fois pour un même élément HTML, c'est la **dernière valeur déclarée** qui sera retenue. Ceci est vrai **sauf** si la première propriété contient le mot clé **!important**.

C'est le cas dans cet exercice : on déclare d'abord une couleur pour toutes les balises (lignes 3 et 7), puis on redéfini par la suite une nouvelle couleur pour chacune de ces balises.

3. Différencier le Layout de la structure HTML

Maintenant que vous distinguez plus clairement la structure HTML du layout, il faut prendre quelques précautions afin de travailler sur une base propre.

Par défaut, la balise `body` possède des marges `margin` de `8px`. Quand vous travaillez sur une mise en page web, il est préférable d'annuler ces marges afin de pouvoir travailler sur l'ensemble de l'espace disponible. Il serait dommage de se limiter dès le début dans la création du layout à cause de comportements par défaut non souhaités.

C'est aussi le moment de faire en sorte que tous vos blocs soient dimensionnés tel que vous l'indiquez avec `width` et ce quelque soit la valeur de leur `border` et `padding` qui peuvent changer pendant la création du layout. Ainsi, quand vous dites à un bloc de faire par exemple `100px` de large, vous êtes sûr que sa largeur totale sera bien de 100px et pas plus.

4. Fixer le header en CSS

4/9

Maintenant que vous avez une copie de travail propre, vous allez pouvoir commencer à mettre en forme votre layout de site web.

Dans cet exercice, vous allez devoir faire en sorte que le `header` s'affiche de manière fixe en haut de la page.

5. Inline-block

5/9

Dans cet exercice, vous allez faire en sorte que les blocs `nav` et `section` soient côte à côte. Pour cela vous allez utiliser la méthode `inline-block`.

6. Insérer des commentaires dans le code HTML

6/9

Vous remarquez que malgré l'ajout des propriétés pour mettre les deux blocs côte à côte, ils sont toujours l'un en-dessous de l'autre.

Cela est dû à un **espace invisible** qui se créé quand il y a un retour à la ligne entre la balise fermante du `nav` et du `section`. Rappelez-vous, pour y remédier il suffit d'ajouter des **commentaires HTML** à cet endroit précis.

C'est un point important à vérifier quand vous faites un layout avec des blocs disposés en `inline-block`.

7. Aligner les blocs avec la propriété Vertical-align en CSS

7/9

Vous remarquez que les deux blocs positionnés en `inline-block` sont alignés vers le bas sur l'axe vertical. C'est le comportement par défaut mais ce n'est pas très esthétique.

Il est préférable que les blocs soient alignés vers le haut et pour cela il faut utiliser la propriété `vertical-align`.

De manière générale, dès que vous positionnez des éléments avec `inline-block` il faudra aussi préciser comment ils s'alignent entre eux avec `vertical-align`.

8. Utiliser la méthode FLOAT en CSS

8/9

Bien, le layout commence à prendre forme mais il reste à aligner les blocs `article` et `aside` pour qu'ils soient côte à côte.

Dans ce cas précis, vous allez utiliser la méthode `float`.

9. Utiliser la propriété CLEAR en CSS

9/9

Les deux blocs `article` et `aside` sont maintenant l'un à côté de l'autre. Mais à cause de la propriété `float`, ils sont maintenant **hors du flux courant**. Ils flottent donc à l'intérieur de leur parent `section`, mais le bloc `footer` qui se trouve en-dessous de `article` et `aside` dans la structure HTML est donc maintenant remonté en haut de son élément parent (`section`).

Dans cet exercice, vous allez utiliser la propriété `clear` afin de remettre par défaut le comportement de l'élément auquel elle s'applique (`footer`). Ainsi il va se positionner comme si les éléments au-dessus étaient restés dans le flux courant.

La valeur de `clear` à préciser sera `both` car il faut remettre par défaut à la fois `float: left` et `float: right`.

Note :

Après cela vous aurez fini la mise en page d'un layout de base pour un site web. Avec ces exercices vous aurez vu les principes essentiels du **positionnement** en CSS. Connaître ces astuces est important car il s'agit de la mise en place de la **structure** visuelle d'un site web.