⬅ Revenir au sommaire.

eqla Cours d'HTML

Sommaire

1. Première page web faite à la va-vite

Nous allons voir ici comment faire une première page web. Nous n'allons respecter aucune règle du HTML. Nous allons juste faire une page web qui fonctionne.

De plus, nous créerons un projet dans PHPStorm, comment créer une page web et comment l'afficher dans un navigateur.

Nous modifierons ce code petit à petit en ajoutant des balises et toujours sans respecter les règles de l'art. Nous verrons ainsi comment structurer une page web et comment la mettre en forme.

Cependant, ce n'est pas la méthode classique pour créer une page web. Nous verrons plus tard comment faire une page web proprement. Nous allons simplement et naïvement créer une page web qui fonctionne.

Je vous donnerai plus tard la "recette" pour créer une page web qui respecte les règles du HTML.

1.1 Création du projet

Si vous êtes déjà dans un projet PHPStorm, fermez-le :

- Dans le menu `File` cliquez sur `Close Project`.

Sinon, suivez les étapes suivantes :

1. Ouvrez `PHPStorm`.
2. Dans le champ texte `Location` entrez le chemin vers le dossier dans lequel vous voulez créer votre projet.
   Par exemple, si vous voulez créer votre projet dans le dossier `C:\Users\Johnny\Documents\ProjetsHTML`, entrez ce chemin dans le champ texte Location et à la fin du chemin, ajoutez le nom du projet `CoursHTML`. Le chemin complet sera donc dans notre exemple `C:\Users\Johnny\Documents\ProjetsHTML\CoursHTML`.
3. Décochez la case `Add 'composer.json'`.
4. Cliquez sur le bouton `Create`.

1.2 Création de la page web

Maintenant, nous allons créer notre première page web.
On va la faire évoluer petit à petit dans ce chapitre.

Pour cela, suivez les étapes suivantes :

1. Avec les flèches de direction, sélectionnez le dossier `CoursHTML` dans la fenêtre de gauche.
2. Dans le menu `File` cliquez sur `New` (`ALT+INSERT`) puis sur `File` .
3. Dans le champ texte `Name` entrez le nom de votre fichier `index.html` .
4. Dans **index.html**, vous allez taper/copier le texte suivant en respectant les espaces et les retours à la ligne :

   ```
   Bonjour les amis ! Présentation Je m'appelle Johnny, enchanté de faire votre connaissance ! Ceci est ma première page web !
   ```

5. Veuillez adapter le code en remplaçant `Johnny` par votre prénom.
6. Enregistrez le fichier (`CTRL+S`).
7. Exécutez votre page dans un navigateur Internet (`SHIFT+F10`).
8. Vérifiez que votre page s'affiche correctement.

Comme vous pouvez le constater, nous avons une page web qui s'affiche dans notre navigateur. C'est déjà un bon début. Mais nous pouvons faire mieux.

Dans la page web, nous souhaiterions avoir un titre, un sous-titre et deux paragraphes or nous en sommes loin.

On voit quelques problèmes :

- Il n'y a pas de titre à notre page: dans l'onglet et dans la barre des titres de la fenêtre, nous avons le nom du fichier.
- Présentation n'est pas un sous-titre, c'est juste un texte. Il est collé au texte de la ligne suivante.
- Il n'y a pas de structure à notre page.
- Il n'y a pas de mise en forme à notre page: les retours à la ligne ne sont pas respectés.
- Et surtout, nous avons un problème d'encodage de caractères. Nous avons nos

caractères accentués qui ne s'affichent pas correctement. Et donc, c'est illisible.

Nous allons petit à petit corriger cela.

1.3 Structuration de la page web

Nous allons maintenant structurer notre page web. Pour cela, nous allons ajouter des balises HTML.

1.3.1 Balise

Oui, je veux bien, mais qu'est-ce qu'une balise ?

> **Une balise** (**tag** en anglais) est un élément qui permet de structurer un document. Une balise est composée d'un `nom` et d'un `contenu`. Le nom d'une balise est entouré de `chevrons` : chevron ouvrant et chevron fermant. Symbolisés par `<` (chevron ouvrant) et `>` (chevron fermant).
>
> Le contenu d'une balise est placé entre la balise ouvrante et la balise fermante. Une balise peut être vide, c'est-à-dire qu'elle n'a pas de contenu, mais nous verrons cela plus tard.
>
> **Syntaxe :**
>
> ```
> <nom>Je suis le contenu</nom>
> ```
>
> Dans l'exemple ci-dessus, le nom de la balise est `nom` et le contenu de la balise est `Je suis le contenu`.

1.3.2 Ajout de la balise <title>

Nous allons ajouter la balise `title` au début de notre page et y mettre le titre de notre page : *Je suis la page d'accueil.*

> **La balise `title`**
> La balise `title` est une balise qui sert à donner un titre à la page. Ce titre est affiché dans l'onglet du navigateur et dans la barre de titre de la fenêtre.
>
> *Syntaxe :*

```
<title>Je suis le titre de la page</title>
```

SPOILER ALERT ! 😀
Elle fait en fait partie de la balise `head`. Nous verrons plus tard ce qu'est la balise `head`.
Mais pour l'instant, retenez juste que la balise `title` fait partie de la balise `head`.

Et oui, il y a des balises qui font partie d'autres balises. C'est comme les poupées russes. 😀

Donc, nous allons ajouter `<title>` Je suis la page d'accueil `</title>` au début de votre page.

Le code de votre page ressemblera à ceci :

```
<title>Je suis la page d'accueil</title> Bonjour les amis ! Présentation Je m'appelle Johnny, enchanté de faire votre connaissance ! Ceci est ma première page web !
```

- Enregistrez votre fichier.
- Vérifiez que votre page s'affiche correctement dans votre navigateur. (`SHIFT+F10`)

On constate que la balise `title` n'est pas affichée dans la page. C'est normal, la balise `title` est une balise qui sert à donner un titre à la page. Ce titre est affiché dans l'onglet du navigateur et dans la barre de titre de la fenêtre. Mais cette balise est importante surtout pour le référencement de votre page web et pour les lecteurs d'écran.

1.3.3 Ajout de la balise <h1>

Les Balises de titre `h1` à `h6`
Dans une page web, il y a des balises de titre avec des niveaux allant de 1 à 6, le niveau 1 est le titre principal, le niveau 2 est le sous-titre du niveau 1, le niveau 3 est le sous-titre du niveau 2, etc.
Syntaxe :

```
<h1>Je suis le titre principal</h1>
```

Ou de manière plus générique :

```
<hx>Je suis le titre principal</hx>
```

Où `x` est un nombre entre 1 et 6.

ATTENTION !
Ne confondez pas les balises de titre (de h1 à h6) avec la balise `title`. La balise `title` sert à donner un titre à la page. Les balises de titre servent à structurer le contenu de la page via des titres de différentes importances.

Nous allons maintenant ajouter la balise `h1` au début de votre page et y mettre le titre principal de notre page : *Bonjour les amis !*

J'avoue il y a mieux comme titre, mais c'est juste pour l'exemple. 😀

Ensuite, nous allons ajouter un titre de niveau 2 : *Présentation*

```
<title>Je suis la page d'accueil</title>
<h1>Bonjour les amis !</h1>
<h2>Présentation</h2>

Je m'appelle Johnny, enchanté de faire votre connaissance ! Ceci est ma première page web !
```

- Enregistrez le fichier (`CTRL+S`).
- Vérifiez que votre page s'affiche correctement dans votre navigateur. (`SHIFT+F10`)

Et voilà, vous avez un titre principal qui s'affiche en gros caractères et un sous-titre qui s'affiche en caractères un peu moins gros.

C'est déjà mieux. Continuons à l'améliorer.

1.3.4 Ajout de la balise <p>

La balise `p`
La balise `p` est une balise qui sert à structurer notre texte via un paragraphe.
Tout comme dans un livre, vous pouvez avoir plusieurs paragraphes.

Syntaxe :

```
  <p>Je suis un paragraphe.</p>
```

Nous allons ajouter deux balises `p` à notre page. Une pour chaque paragraphe :

```
<meta charset="UTF-8" />
<title>Je suis la page d'accueil</title>
<h1>Bonjour les amis !</h1>
<h2>Présentation</h2>
<p>Je m'apppelle Johnny, enchanté de de faire votre connaissance !</p>
<p>Ceci est ma première page web !</p>
```

- Enregistrez le fichier (`CTRL+S`).
- Vérifiez que votre page s'affiche correctement dans votre navigateur. (`SHIFT+F10`)
- Vérifiez que votre page est bien structurée : nous avons un titre principal et deux paragraphes.

Ici, nous avons donc nos deux paragraphes. Nous aurions peut-être pu n'en faire qu'un seul, mais c'est juste pour l'exemple.

Ok ! Notre page commence à ressembler à quelque chose. Mais nous avons toujours nos problèmes d'encodage de caractères.

1.3.5 Encodage de caractères

Depuis que nous avons commencé à faire cette page web, nous avons un problème d'encodage de caractères. Nous avons nos caractères accentués qui ne s'affichent pas correctement. Et donc, c'est illisible.
Pour y remédier, nous allons devoir ajouter une balise `meta` avec un attribut `charset` = `UTF-8` à notre page.

Un Attribut ? Késako ?

> **Attribut d'une balise**
> L'attribut d'une balise est un élément qui permet de modifier le comportement d'une balise. Un attribut est composé d'un `nom` et d'une `valeur` . La valeur de l'attribut est entouré de `guillemets` : guillemet ouvrant et guillemet fermant. Symbolisés par `"` (guillemet ouvrant) et `"` (guillemet fermant).

Ok, admettons, et elle sert à quoi la balise `<meta>` ?

> **La balise `meta`**
> La balise `<meta>` est une balise auto-fermante utilisée pour spécifier les métadonnées du document HTML. Ces métadonnées ne sont généralement pas visibles par l'utilisateur, mais sont importantes pour le navigateur, les moteurs de recherche et d'autres services web. Tout comme la balise `<title>` , la balise `<meta>` est toujours placée à l'intérieur de la balise `<head>` d'un document HTML.
>
> *Exemple pour l'encodage en UTF8 :*
>
> ```
> <meta charset="UTF-8" />
> ```
>
> Ici la balise `<meta>` a un attribut `charset` avec la valeur `UTF-8` . Cet attribut permet de spécifier l'encodage des caractères de la page. Ici, nous spécifions que nous utilisons l'encodage `UTF-8` . Cet encodage permet d'afficher les caractères accentués.

Donc, nous allons ajouter `<meta charset="UTF-8">` au début de votre page:

```
<meta charset="UTF-8" />
<title>Je suis la page d'accueil</title>
<h1>Bonjour les amis !</h1>
<h2>Présentation</h2>
<p>Je m'apppelle Johnny, enchanté de de faire votre connaissance !</p>
<p>Ceci est ma première page web !</p>
```

- Enregistrez le fichier (`CTRL+S`).
- Vérifiez que votre page s'affiche correctement dans votre navigateur. (`SHIFT+F10`)
- Vérifiez que votre page est bien structurée : nous avons un titre principal, deux paragraphes et nos caractères accentués s'affichent correctement !
- Félicitations ! Vous avez fait votre première page web ! 🎉

2. Première page web faite proprement

Bon, nous sommes heureux de notre première page web. Mais nous avons fait n'importe quoi. Nous avons juste fait une page web qui fonctionne. Nous n'avons pas respecté les règles du

HTML. Nous allons maintenant voir comment faire une page web proprement.

Je vais vous afficher le code source du navigateur. Normalement, ce code devrait correspondre au code que vous avez créé. Mais je vous ai dit que les navigateurs d'aujourd'hui sont intelligents. Ils corrigent parfois certaines erreurs que nous faisons. Et donc, le code source affiché par le navigateur n'est pas forcément le code que nous avons créé. Il est possible que le code source affiché par le navigateur soit différent du code que nous avons créé.

Voici le code source que le navigateur a généré pour notre page web :

```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Je suis la page d'accueil</title>
  </head>
  <body>
    <h1>Bonjour les amis !</h1>
    <p>Je m'apppelle Johnny, enchanté de de faire votre connaissance !</p>
    <p>Ceci est ma première page web !</p>
  </body>
</html>
```

Oula ! Mais il y a plein de balises que nous n'avons pas créées ! Une balise html, head, body.

Effectivement, nous n'avons pas créé ces balises. Mais elles sont obligatoires. Nous allons voir pourquoi.

2.1 La balise <html>

La balise `html` est une balise qui sert à définir le début et la fin du document HTML. Elle est toujours placée au début et à la fin du document HTML.

Elle est composée de deux balises enfants : la balise `<head>` et la balise `<body>`.
Il est recommandé d'indiquer la langue du document HTML dans l'attribut `lang` de la balise `<html>`. Cela permet aux moteurs de recherche et aux lecteurs d'écran de mieux comprendre le contenu de la page.

> *Syntaxe :*

```
<html lang="fr"></html>
```

Ell est présentée pour la syntaxe seule sans les balises enfants `<head>` et `<body>` . C'est pour que vous puissiez voir la balise `<html>` toute seule.
Syntaxe complète:

```
<html lang="fr">
  <head>
    <!-- Contenu de la balise head -->
  </head>
  <body>
    <!-- Contenu de la balise body -->
  </body>
</html>
```

Je n'ai pas développé les balises `head` et `body` exprès. Pour garder le code plus lisible. Nous allons développer ces balises plus tard.

Notre page web devient donc avec l'attribut `lang` :

```
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <title>Je suis la page d'accueil</title>
  </head>
  <body>
    <h1>Bonjour les amis !</h1>
    <h2>Présentation</h2>
    <p>Je m'apppelle Johnny, enchanté de de faire votre connaissance !</p>
    <p>Ceci est ma première page web !</p>
  </body>
</html>
```

2.2 La balise <!-- -->

Vous avez découvert une étrange balise : `<!-- -->` lorsque je vous ai présenté la syntaxe de la balise `html` : C'est une balise de commentaire.

Elle permet de mettre des commentaires dans le code HTML. Ces commentaires ne sont pas affichés dans la page web. Ils sont juste là pour nous aider à comprendre le code.

Elle n'a aucune influence sur l'affichage, c'est juste pour le webdeveloppeur qu'elle est utile.

> **Syntaxe :**
>
> ```
> <!-- Je suis un commentaire -->
> ```
>
> Parfois, on teste du code html et on se rend compte qu'une partie du code ne fonctionne pas. On peut alors mettre cette partie de code en commentaire pour la désactiver. Cela permet de tester le code sans cette partie. Et donc, de voir si le problème vient de cette partie de code ou pas.

2.3 La balise <head>

La balise `head` est une balise qui sert à définir des informations sur le document HTML. Elle est toujours placée entre les balises `<html>` et `<body>`.
Cette balise est très importante car on y ajoute:

- le titre de notre page via la balise `<title>`,
- les métadonnées de notre page via la balise `<meta>`,
- les ressources externes:
    - les liens vers les fichiers CSS via la balise `<link>`,
    - les liens vers les fichiers JavaScript via la balise `<script>` si on veut les placer dans le `<head>` car on peut aussi les placer juste avant la balise fermante `</body>`.

> *Syntaxe :*
>
> ```
> <head>
>   <!-- Contenu de la balise head -->
> </head>
> ```
>
> Pour rappel, la balise `head` fait partie de la balise `html`. Donc, elle est toujours placée entre les balises `<html>` et `<body>`.

Exemple :

```
<html lang="fr">
  <head>
    <!-- Contenu de la balise head -->
  </head>
  <body>
    <!-- Contenu de la balise body -->
  </body>
</html>
```

2.4 La balise <body>

La balise `body` est une balise qui sert à définir le contenu du document HTML. Elle est toujours placée entre les balises `<html>` et `</html>`.
Cette balise est très importante car on y ajoute:

- le contenu de notre page,
- les liens vers les fichiers JavaScript via la balise `<script>` si on veut les placer dans le `<body>` car on peut aussi les placer juste avant la balise fermante `</body>`.

Dans la balise `body`, on peut mettre toutes sortes de balises HTML: des balises de titre, des paragraphes, des images, des liens, des tableaux, etc.

> *Syntaxe :*
>
> ```
> <body>
>   <!-- Contenu de la balise body -->
> </body>
> ```

2.5 La balise DOCTYPE

Alors dans notre approche naïve de faire notre page web, le navigateur n'a pas ajouté la balise `DOCTYPE` au code source qu'il a généré.

La balise `DOCTYPE` est une balise qui sert à définir le type de document HTML. Elle est toujours

placée au début du document HTML. Elle est obligatoire. Elle permet au navigateur de savoir quel type de document HTML il doit afficher.
Elle se met toujours au début du document HTML. Elle n'a pas de balise fermante. Elle est auto-fermante. Elle n'a pas de contenu. Elle est composée d'un nom et d'une valeur. Le nom est `DOCTYPE` et la valeur est `html`.

Elle a eu différentes syntaxes au fil du temps. Mais aujourd'hui, en HTML 5, la syntaxe est la suivante :

```
<!DOCTYPE html>
```

Elle peut vous sembler compliquée, mais c'est juste une balise auto-fermante avec un nom et une valeur. C'est tout.
Juste pour info, voici les différentes syntaxes qu'elle a eu au fil du temps :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<!-- HTML 4.01 Strict -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4
<!-- HTML 4.01 Transitional -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/fra
<!-- HTML 4.01 Frameset -->

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/
<!-- XHTML 1.0 Strict -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml
<!-- XHTML 1.0 Transitional -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DT
<!-- XHTML 1.0 Frameset -->

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml1
<!-- XHTML 1.1 -->

<!DOCTYPE html>
<!-- HTML5 -->
```

Il ne faut bien entendu par retenir toutes ces syntaxes. Je vous les donne juste pour info.
Donc, comme vous pouvez le constater, la dernière syntaxe est la plus simple. C'est celle que nous utiliserons.

> *Syntaxe pour le HTML 5:*
>
> ```
> <!DOCTYPE html>
> ```

Donc en prenant en compte cette balise, notre code ressemblera à ceci :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <title>Je suis la page d'accueil</title>
  </head>
  <body>
    <h1>Bonjour les amis !</h1>
    <h2>Présentation</h2>
    <p>Je m'apppelle Johnny, enchanté de de faire votre connaissance !</p>
    <p>Ceci est ma première page web !</p>
  </body>
</html>
```

Et là, nous avons une page web correcte ! 🎉 🎉 🎉

2.6 La balise <meta>

Nous avons déjà vu cette balise mais je la remets ici pour la forme.

La balise `meta` est une balise auto-fermante utilisée pour spécifier les métadonnées du document HTML. Ces métadonnées ne sont généralement pas visibles par l'utilisateur, mais sont importantes pour le navigateur, les moteurs de recherche et d'autres services web. Tout comme la balise `<title>`, la balise `<meta>` est toujours placée à l'intérieur de la balise `<head>` d'un document HTML.

> *Syntaxe :*
>
> ```
> <meta attribut="valeur" />
> ```
>
> On voit donc qu'elle a toujours un attribut et une valeur.

A parti d'un exemple, je vais maintenant vous montrer différentes balises meta que l'on rencontre souvent dans les pages web.

```
<meta charset="UTF-8" />
<meta name="description" content="Ceci est ma première page web !" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

- La première balise `meta` sert à spécifier l'encodage des caractères de la page. Ici, nous spécifions que nous utilisons l'encodage `UTF-8`. Cet encodage permet d'afficher les caractères accentués.
- La deuxième balise `meta` sert à spécifier la description de la page.
    - Cette description est utilisée par les moteurs de recherche pour afficher un résumé de la page dans les résultats de recherche.
    - Lorsqu'une URL est partagée sur des plateformes de médias sociaux comme Facebook ou LinkedIn, ces plateformes peuvent utiliser la méta-description pour fournir un aperçu du contenu de la page.
- La troisième balise `meta` sert à spécifier la largeur de la page. Ici, nous spécifions que la largeur de la page est égale à la largeur de l'écran de l'appareil. Cela permet d'adapter la page à la taille de l'écran de l'appareil. C'est ce qu'on appelle le responsive web design. Mais en gros, copiez-la et collez-la dans vos pages web. Elle est très importante. Nous verrons cela plus tard.

Modifions notre page web pour prendre en compte ces balises `meta` :

```html
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <meta name="description" content="Ceci est ma première page web !" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Je suis la page d'accueil</title>
  </head>
  <body>
    <h1>Bonjour les amis !</h1>
    <h2>Présentation</h2>
    <p>Je m'apppelle Johnny, enchanté de de faire votre connaissance !</p>
    <p>Ceci est ma première page web !</p>
  </body>
</html>
```

Précédemment notre page web était correcte. Mais là, elle est encore mieux ! En effet, notre page web sera mieux référencée par les moteurs de recherche. Et elle sera mieux affichée sur les appareils mobiles.

3. Le DOM

Le `DOM` est l'acronyme de **D**ocument **O**bject **M**odel. C'est un modèle de document qui permet de représenter un document HTML sous forme d'arbre. C'est-à-dire que chaque balise HTML est représentée par un noeud de l'arbre. Et chaque noeud peut avoir des enfants. Et chaque enfant peut avoir des enfants, etc.

Quand on parle du DOM, on parle du DOM HTML.

Voici une représentation visuelle du DOM HTML (non accessible) :

```
Document html (par exemple index.html)
|-- DOCTYPE
|-- html
    |-- head
    |   |-- meta (charset="UTF-8")
    |   |-- title
    |       |-- Text ("Titre de la page")
    |-- body
        |-- h1
        |   |-- Text ("Mon titre principal")
        |-- p
            |-- Text ("Ceci est un paragraphe.")
```

En voici une représentation accessible textuelle :
La flèche ==> signifie a pour balise enfant

```
Document html  ==> DOCTYPE
Document html  ==> html
html ==> head
head ==> meta (charset="UTF-8")
head ==> title
title ==> Text ("Titre de la page")
html ==> body
body ==> h1
h1 ==> Text ("Mon titre principal")
body ==> p
p ==> Text ("Ceci est un paragraphe.")
```

Il n'y a donc rien de nouveau vraiment, juste savoir que cette représentation existe appelée DOM. Cela nous permet de mieux comprendre la structure d'une page web. Et puis, ça le fait dire que l'on connaît le terme `DOM`. 😄

4. On récapitule

Nous avons vu beaucoup de choses dans ce chapitre. Je vais donc vous faire un petit récapitulatif de ce que nous avons vu.

4.1 Les balises

Nous avons vu les balises suivantes :

- La balise `DOCTYPE` qui sert à définir le type de document HTML. Elle est toujours placée au début du document HTML. Elle est obligatoire. Elle permet au navigateur de savoir quel type de document HTML il doit afficher.
- La balise `html` qui sert à définir le début et la fin du document HTML. Elle est toujours placée au début (après `DOCTYPE`) et à la fin du document HTML. Elle est composée de deux balises enfants : la balise `<head>` et la balise `<body>`.
    - La balise `head` qui sert à définir des informations sur le document HTML. Elle est toujours placée entre les balises `<html>` et `<body>`.
        - La balise `title` qui sert à donner un titre à la page. Ce titre est affiché dans l'onglet du navigateur et dans la barre de titre de la fenêtre.
        - La balise `meta` qui sert à spécifier les métadonnées du document HTML. Ces métadonnées ne sont généralement pas visibles par l'utilisateur, mais sont importantes pour le navigateur, les moteurs de recherche et d'autres services web. Tout comme la balise `<title>`, la balise `<meta>` est toujours placée à l'intérieur de la balise `<head>` d'un document HTML.
    - La balise `body` qui sert à définir le contenu du document HTML. Elle est toujours placée entre les balises `<html>` et `</html>`.
        - Les balises de titre `h1` à `h6` qui servent à structurer le contenu de la page via des titres de différentes importances.
        - La balise `p` qui sert à structurer notre texte via un paragraphe.
- La balise `<!-- -->` qui sert à mettre des commentaires dans le code HTML. Ces commentaires ne sont pas affichés dans la page web. Ils sont juste là pour nous aider à comprendre le code.

4.2 Les attributs

Nous avons vu les attributs suivants :

- L'attribut `lang` de la balise `<html>` qui permet d'indiquer la langue du document HTML. Cela permet aux moteurs de recherche et aux lecteurs d'écran de mieux comprendre le contenu de la page.
- L'attribut `charset` de la balise `<meta>` qui permet de spécifier l'encodage des caractères de la page. Ici, nous spécifions que nous utilisons l'encodage `UTF-8`. Cet encodage permet d'afficher les caractères accentués.
- L'attribut `name` de la balise `<meta>` :
    - avec comme valeur `description` permet de spécifier la description de la page dans l'attribut `content`. Cette description est utilisée par les moteurs de recherche pour afficher un résumé de la page dans les résultats de recherche.

```
`<meta name="description" content="Ceci est ma première page web !" />`
```

    - avec comme valeur `viewport` permet de spécifier la largeur de la page et le zoom initial. Ici, nous spécifions, dans l'attribut `content`, que la largeur de la page (`width`) est égale à la largeur de l'écran de l'appareil (`device-width`). Cela permet d'adapter la page à la taille de l'écran de l'appareil. Grâce à cela notre page se rapproche de la notion de responsive web design. Enfin, le facteur de zoom est remis à 100% (`initial-scale=1.0`). Notez enfin, que nous pouvons spécifier plusieurs valeurs séparées par une virgule. Ici, nous n'en avons qu'une seule.

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

- L'attribut `content` de la balise `<meta>` qui dépent de la valeur `name` de la balise `<meta>`. Comme vous avez pu le voir précédemment.

4.3 Notre code commenté

Je vais reprendre notre code et le commenter pour que vous puissiez mieux comprendre ce que nous avons fait.

```
<!-- La déclaration DOCTYPE indique au navigateur que ce document est de type HTML5. -->
<!DOCTYPE html>
<!-- L'élément <html> est la racine du document et contient tout le contenu de la page. L
<html lang="fr">
  <!-- L'élément <head> contient des métadonnées (informations sur le document) qui ne so
  <head>
    <!-- L'élément <meta> avec l'attribut "charset" définit l'encodage des caractères du
    <meta charset="UTF-8" />
    <!-- L'élément <meta> avec l'attribut "name" à "description" fournit une courte descri
    <meta name="description" content="Ceci est ma première page web !" />
    <!-- L'élément <meta> avec l'attribut "viewport" rend la page web responsive, c'est-à
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <!-- L'élément <title> définit le titre du document, qui s'affiche dans l'onglet du na
    <title>Je suis la page d'accueil</title>
  </head>

  <!-- L'élément <body> contient le contenu visible de la page web. -->
  <body>
    <!-- L'élément <h1> est un titre de niveau 1. Il est généralement utilisé pour le titr
    <h1>Bonjour les amis !</h1>
    <!-- L'élément <h2> est un titre de niveau 2. Il est généralement utilisé pour le sous
    <h2>Présentation</h2>
    <!-- L'élément <p> est un paragraphe. Il contient du texte qui est affiché sur la page
    <p>Je m'appelle Johnny, enchanté de faire votre connaissance !</p>
    <!-- Un autre paragraphe. -->
    <p>Ceci est ma première page web !</p>
  </body>
  <!-- Fin du document HTML -->
</html>
```

Il faut juste noter que j'ai mis un commentaire juste avant la balise DOCTYPE mais ce n'est pas permis. Je l'ai fait juste pour vous expliquer ce qu'est la balise DOCTYPE. DOCTYPE doit être la première balise du document HTML.

Voyons ensemble le résultat de notre travail : Notre première page web.
Et le code source de celle-ci: Code source.

On est d'accord, c'est beaucoup mieux que notre première page web sans le moindre code HTML car elle respecte les normes du HTML et visuellement elle est plus esthétique. 😀 Elle

n'est pas très jolie, il faut bien l'avouer. Lors du cours de CSS, vous apprendrez à la rendre plus jolie.

4.4 Exercices d'entraînement

Maintenant, je vous propose de faire quelques exercices pour vous entraîner.
Faites l'Exercice 1 et l'Exercice 2.

5. la page par défaut

Lorsque l'on va sur un site web, on arrive sur une page d'accueil. C'est la page qui s'affiche par défaut.

Parfois, on voit écrit dans l'url : https://google.com/index.html où `index.html`` est la page par défaut. C'est-à-dire que si on ne précise pas de page, c'est cette page qui sera affichée.

En d'autres termes, si on reprend l'url d'exemple, elle signifie que si on va sur https://google.com, on arrive sur la page https://google.com/index.html.

C'est totalement transparent pour nous. On ne voit pas que l'on arrive sur la page https://google.com/index.html. On voit juste https://google.com.

En fonction du serveur, la page par défaut peut-être sensiblement différente.
Par exemple, sur un serveur web:

- Apache, la page par défaut est **index.html**
- Apache + PHP, la page par défaut est/peut être **index.php** ou index.html si **index.php** n'existe pas.
- IIS, la page par défaut est **default.html** ou **default.asp** ou **index.htm** ou **index.html**.
- Nginx, la page par défaut est **index.html**.
- Tomcat : Pour les applications Java, Tomcat utilise **index.jsp** ou **index.html**
- Etc.

Donc pour résumer, soit on trouve comme page par défaut:

- **index.html**
- **index.php**

- **default.html**
- **default.asp**
- **index.htm**

Tout dépendra du serveur web.

6. La balise <a> - Les liens hypertextes

Page d'information sur la balise a: https://developer.mozilla.org/fr/docs/Web/HTML/Element/a

Nous y voilà !

Nous allons maintenant voir comment créer des liens hypertextes. C'est-à-dire des liens qui permettent de naviguer d'une page web à une autre page web.

6.1 Les liens externes

Pour cela, nous allons utiliser la balise `a` qui est une balise qui sert à créer un lien hypertexte. Elle est composée d'un attribut `href` qui contient l'URL de la page web vers laquelle on veut créer un lien. Le contenu de la balise `a` est le texte du lien.

> *Syntaxe :*
>
> ```
> <a href="URL">Texte du lien</a>
> ```
>
> *Exemple :*
>
> ```
> <a href="https://www.google.com">Lien vers Google</a>
> ```
>
> *Résultat :* > Lien vers Google

Dans le code ci-dessus, nous avons créé un lien vers le site web de Google. Le texte du lien est `Google`. Si on clique sur ce lien, on est redirigé vers le site web de Google.

Maintenant, ce n'est pas toujours une `URL` (**U**niform **R**esource **L**ocator) comme valeur pour l'attribut `href` :

- les liens vers des fichiers

```
Télécharger le fichier<a href="docs/HTML5-Cheat-Sheet.pdf">
  HTML5-Cheat-Sheet.pdf</a
>
```

  Il faut juste noter que le fichier dans l'exemple doit être dans le même dossier que la page web. Sinon, il faut préciser le chemin vers le fichier.
- des liens vers des adresses mail (on pourrait définir le sujet du mail et le corps du mail)

```
<a href="mailto:johnny.piette@eqla.be">Envoyer un mail à Johnny</a>
```

- des liens vers des liens internes/ancres (que nous verrons plus tard)

```
<a href="#sommaire">Revenir au sommaire</a>
```

- des liens vers des numéros de téléphone (évidemment sur un smartphone)

```
<a href="tel:+32475252525">Appeler Johnny</a>
```

- des sms (évidemment sur un smartphone) et on pourrait définir le corps du sms

```
<a href="sms:+32475252525">Envoyer un sms à Johnny</a>
```

- du code JavaScript

```
html   <a href="javascript:alert('Vous avez cliqué sur le lien !');">Cliquez ici
```

  Résultats: Cliquez ici pour voir le résultat de ces liens.

Faites l'Exercice suivant: Exercices - liens hypertextes: Partie 2

6.2 Attribut target de la balise <a>

Nous allons maintenant voir l'attribut `target` de la balise `a`. Cet attribut permet de spécifier comment le lien doit être ouvert. Par défaut, le lien s'ouvre dans la même fenêtre. Mais on peut spécifier qu'il doit s'ouvrir dans une nouvelle fenêtre ou un nouvel onglet.

> *Syntaxe :*

```
<a href="URL" target="valeur">Texte du lien</a>
```

Exemple :

```
<a href="https://www.google.com" target="_blank">Lien vers Google</a>
```

Résultat : > Lien vers Google

Dans le code ci-dessus, nous avons créé un lien vers le site web de Google. Le texte du lien est `Google`. Si on clique sur ce lien, on est redirigé vers le site web de Google. Mais cette fois-ci, le lien s'ouvre dans un nouvel onglet.

Pour l'attribut `target`, il peut y avoir plusieurs valeurs mais j'en verrai qu'une:

- `_blank` : le lien s'ouvre dans un nouvel onglet.

Faites l'Exercice suivant: Exercices - liens hypertextes: Partie 3

6.3 Attribut rel de la balise <a>

Nous allons maintenant voir l'attribut `rel` de la balise `a`. Cet attribut permet de spécifier la relation entre la page web en cours et la page web vers laquelle on veut créer un lien.

Voici les différentes valeurs possibles (nous n'allons pas toutes les voir) pour l'attribut `rel` :

- `nofollow` : le lien pointe vers une page web qui n'est pas approuvée par le propriétaire de la page web en cours.
- `noreferrer` : le lien pointe vers une page web qui ne doit pas envoyer de référence à la page web en cours.
- `noopener` : le lien pointe vers une page web qui ne doit pas ouvrir la page web en cours.

Syntaxe :

```
<a href="URL" rel="valeur">Texte du lien</a>
```

Exemples :

```
<a href="https://www.google.com" rel="nofollow">Google</a>
<a href="https://www.google.com" rel="noreferrer">Google</a>
<a href="https://www.google.com" rel="noopener">Google</a>
```

Dans les 3 cas, il y a un lien vers le site web de Google.

Nous allons expliquer chaque cas :

- Dans le premier cas de `rel="nofollow"`, le lien dirige vers une page web sans transmettre d'autorité SEO (la valeur ou le "poids" qu'un site donne à un autre site aux yeux des moteurs de recherche via un lien) de la part du propriétaire de la page actuelle. Cela signifie que le propriétaire de la page actuelle ne souhaite pas endosser ou promouvoir activement le site web cible, comme celui de Google, par exemple.
- Dans le deuxième cas `rel="noreferrer"`, le lien pointe vers une page web qui ne doit pas envoyer de référence à la page web en cours. C'est-à-dire que le propriétaire de la page web en cours ne veut pas que Google sache que le lien vers son site web se trouve sur la page web en cours.
- Dans le troisième cas `rel="noopener"`, le lien pointe vers une page web qui ne doit pas ouvrir la page web en cours. C'est-à-dire que le propriétaire de la page web en cours ne veut pas que Google puisse modifier la page web en cours. Nous allons le voir dans le prochain chapitre.

Enfin, on peut les combiner. Par exemple, on peut avoir `rel="nofollow noreferrer noopener"`. Bien que noreferrer et noopener soient redondants. En effet, noopener implique noreferrer. Mais on peut les combiner.

Faites l'Exercice suivant: Exercices - liens hypertextes: Partie 4

6.4 Danger de l'attribut target="_blank"

L'attribut `target="_blank"` est très pratique pour ouvrir un lien dans un nouvel onglet. Mais il peut être dangereux. En effet, si on utilise cet attribut, on peut ouvrir une page web malveillante dans un nouvel onglet. Cette page web malveillante peut alors modifier la page web en cours. Par exemple, elle peut modifier le texte de la page web en cours. Elle peut aussi modifier le lien vers lequel on a cliqué. C'est ce qu'on appelle une attaque de type `tabnabbing`.

> *Le **tabnabbing** est une technique où une page malveillante ouverte dans un nouvel onglet peut prendre le contrôle de l'onglet d'origine et le rediriger vers une autre URL, souvent une fausse page de connexion. Ou modifier le comportement de la page d'origine. Profiter que l'utilisateur est connecté pour récupérer des informations personnelles (identifiants, mots de passe, etc.). Cela peut ensuite être utilisé pour tromper l'utilisateur et recueillir ses informations.*

La plupart des navigateurs web ont corrigé ce problème. Mais il y a encore des navigateurs qui ne l'ont pas corrigé.

Pour éviter cela, il faut ajouter l'attribut `rel="noopener"` à la balise `a`. Cet attribut permet de spécifier que le lien pointe vers une page web. Et que cette page de distination ne peut pas ouvrir la page d'origine en cours.

Je vais vous donner un exemple de page d'origine qui pointe vers une page malveillante. Dans cet exemple, la page malveillante va modifier le texte de la page d'origine. Elle va aussi modifier le lien vers lequel on a cliqué. C'est ce qu'on appelle une attaque de type `tabnabbing` comme vu précédemment.

> *Exemple :*

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Page d'Origine</title>
  </head>
  <body>
    <h1>Bienvenue sur la page d'origine</h1>
    <a href="p2.html" target="_blank"
      >Cliquez ici pour ouvrir une nouvelle page</a
    >
  </body>
</html>
```

Dans le code ci-dessus, nous avons créé une page web qui contient un lien vers une autre page web. Ce lien s'ouvre dans un nouvel onglet. Si on clique sur ce lien, on arrive sur la page web

suivante :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Page Malveillante</title>
    <script>
      window.onload = function () {
        if (window.opener) {
          window.opener.document.body.innerHTML =
            "<h1>Cette page a été piratée!</h1>";
          window.opener.alert("Message depuis la page malveillante!");
        }
      };
    </script>
  </head>

  <body>
    <h1>Page Externe</h1>
    <p>Regardez la page d'origine maintenant.</p>
  </body>
</html>
```

Dans le code ci-dessus, nous avons créé une page web qui contient du code JavaScript. Ce code JavaScript est exécuté lorsque la page web est chargée.

Ce code JavaScript vérifie si la page web a été ouverte par une autre page web. Si c'est le cas:

- le code JavaScript modifie le texte de la page web d'origine (la page appelante) en cours.
- le code affiche aussi un popup avec le message "Message depuis la page malveillante".

C'est ce qu'on appelle une attaque de type `tabnabbing` .

> Pour éviter ce type d'attaque lorsque vous voulez ouvrir une fenêtre dans un onglet, utilisez l'attribut `rel="noopener noreferrer"` .

```
<a href="p2.html" target="_blank" rel="noopener noreferrer"
  >Cliquez ici pour ouvrir une >nouvelle page</a
>
```

6.5 Attribut title de la balise <a>

Nous allons maintenant voir l'attribut `title` de la balise `a`. Cet attribut permet de spécifier un titre pour le lien. Ce titre est affiché lorsque l'utilisateur passe la souris sur le lien.

Syntaxe :

```
<a href="URL" title="valeur">Texte du lien</a>
```

Exemple :

```
<a href="https://www.google.com" title="Lien vers Google">Google</a>
```

Résultat : > Google

Dans le code ci-dessus, nous avons créé un lien vers le site web de Google. Le texte du lien est `Google`. Si on passe la souris sur ce lien, on voit apparaître le titre `Lien vers Google`.

Faites l'Exercice suivant: Exercices - liens hypertextes: Partie 5

6.6 Les liens internes/ancres

Nous allons maintenant voir comment créer des liens internes appelés aussi ancres. C'est-à-dire des liens qui permettent de naviguer d'une partie de la page web en cours à une autre partie de la même page web. C'est très utile pour naviguer dans une page web qui est très longue.

Par exemple, la page du cours contient un sommaire. Ce sommaire contient des liens qui permettent de naviguer vers les différentes parties de la page web. Cliquez sur ce lien pour revenir au sommaire tout en haut de la page: Sommaire.

Ou bien on peut directement aller sur le lien interne d'une autre page web pour que votre utilisateur puisse directement aller sur la partie de la page web qui l'intéresse.

Par exemple sur wikipedia le lien interne "Avenir du HTML" de la page web HTML permet d'aller directement sur la partie de la page web qui parle de l'avenir du HTML via ce lien : Avenir du HTML.

L'intérêt dans l'exemple précédent est que l'utilisateur n'a pas besoin de scroller la page web pour trouver la partie qui l'intéresse. Il peut directement cliquer sur le lien interne pour aller sur la partie qui l'intéresse en l'occurence "Avenir du HTML".

Pour cela, nous allons à nouveau utiliser la balise `a` qui est une balise qui sert à créer un lien hypertexte. Elle est composée d'un attribut `href` qui contient l'URL de la page web vers laquelle on veut créer un lien. Mais cette fois-ci, l'URL sera précédée d'un `#` suivi de l'identifiant de la balise vers laquelle on veut créer un lien.

Cet identifiant est défini par l'attribut `id` que l'on mettra sur la balise vers laquelle on veut créer un lien.

```
<h1 id="sommaire">Sommaire</h1>
```

Cet identifiant doit être unique dans la page web. C'est-à-dire qu'il ne doit pas y avoir deux balises avec le même identifiant.

> *Syntaxe :*
>
> ```
> <a href="#identifiant">Texte du lien</a>
> ```
>
> *Exemple :*
>
> ```
> <a href="#sommaire">Revenir au sommaire</a>
> ```
>
> *Résultat :* > Revenir au sommaire

Dans le code ci-dessus, nous avons créé un lien vers l'identifiant `sommaire`. Cet identifiant est défini sur la balise `<h1>` qui contient le texte `Sommaire`. Si on clique sur ce lien, on est redirigé vers la balise `<h1>` qui contient le texte `Sommaire`.

Faites l'Exercice suivant: Exercices - liens hypertextes: Partie 6

6.7 Importance de l'attribut id

L'attribut `id` est très importants Il permet de créer des liens vers des ancres. Mais il permet aussi de cibler une balise pour la modifier avec du CSS ou du JavaScript.

Par exemple, si on veut modifier la couleur du texte de la balise `<h1>` qui contient le texte `Sommaire`, on peut utiliser le code CSS suivant :

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

6.8 Liens relatifs

Un lien relatif est un lien qui pointe vers une autre page ou un fichier en se basant sur l'emplacement actuel de la page. Quand je parle d'emplacement, je parle du dossier dans lequel se trouve la page web.

Exemples :

- Si vous avez deux pages web : index.html et page2.html dans le même dossier, vous pouvez créer un lien de index.html vers page2.html en utilisant simplement le nom du fichier, comme ceci :

  ```
  <a href="page2.html">Aller sur la page 2</a>
  ```

- Si page2.html est dans un sous-dossier nommé cours, alors le lien relatif depuis index.html serait :

  ```
  html   <a href="cours/page2.html">Aller sur la page 2 du cours</a>
  ```

  Pourquoi utiliser un lien relatif ?
  C'est pratique lorsque vous voulez lier des pages ou des fichiers qui sont proches les uns des autres, car vous n'avez pas besoin de spécifier le chemin complet.

On peut donc appeler la page index via un relatif de deux manières différentes :

- `./index.html`

  ```
  <a href="./index.html">Aller sur la page d'accueil</a>
  ```

- `index.html`

```
<a href="index.html">Aller sur la page d'accueil</a>
```

Quand on utilise le symbole ./ devant le nom du fichier, cela signifie que le fichier se trouve dans le même dossier que la page web en cours. Donc, on peut créer un lien relatif vers ce fichier via `./index.html`.

Mais comme le fichier se trouve dans le même dossier que la page web en cours, on peut aussi créer un lien relatif vers ce fichier via `index.html`.

Si le fichier se trouve dans un sous-dossier, on peut créer un lien relatif vers ce fichier via `sous-dossier/index.html` :

```
<a href="sous-dossier/index.html">Aller sur la page d'accueil</a>
```

Peut s'écrire aussi avec ./ :

```
<a href="./sous-dossier/index.html">Aller sur la page d'accueil</a>
```

6.9 Liens absolus

Un lien absolu est un lien qui pointe vers une page ou un fichier en utilisant son adresse complète, que ce soit pour des sites web externes ou pour des fichiers dans des dossiers spécifiques de votre propre site.

Exemples :

- Si vous voulez créer un lien vers le site web de Google depuis n'importe quelle page, vous utiliserez son adresse complète :

```
<a href="https://www.google.com">Aller sur Google</a>
```

- Si vous avez une page page3.html dans un sous-dossier nommé articles sur votre propre site web, et que vous voulez créer un lien depuis la page d'accueil, vous pourriez utiliser un lien absolu comme ceci :

```
html   <a href="/articles/page3.html">Lire l'article</a>
```

Ici, le / au début indique la racine du site, suivi du chemin complet vers le fichier.

Pourquoi utiliser un lien absolu ?
Les liens absolus sont utiles lorsque vous voulez lier à une page ou un fichier en spécifiant son chemin complet, que ce soit sur un site externe ou dans un emplacement spécifique de votre propre site.

6.10 Liens d'évitement / skip links

> **On passe ce point car il est plus intéressant de le voir lorsque nous aurons vu les balises de navigation. Nous y reviendrons plus tard.**

Les liens d'évitement sont des liens qui permettent de sauter des éléments de navigation et d'aller directement au contenu principal de la page. Ils sont très utiles pour les personnes qui utilisent un lecteur d'écran. En effet, cela leur permet d'aller directement au contenu principal de la page sans devoir écouter tous les éléments de navigation.

Nous verrons plus tard comment créer des liens d'évitement lorsque nous aurons vu les balises de navigation. Mais je vous donne déjà un exemple de lien d'évitement :

```
<a href="#main">Aller au contenu principal</a>
```

Dans le code ci-dessus, nous avons créé un lien vers l'identifiant `main`. Cet identifiant est défini sur la balise `<main>` qui contient le contenu principal de la page web. Si on clique sur ce lien, on est redirigé vers la balise `<main>` qui contient le contenu principal de la page web.

Vous comprenez pourquoi les id sont importants ? 😉 En effet, ils permettent aussi de créer des liens internes/ancres et des liens d'évitement.

On applique un style css sur les liens d'évitement pour les cacher visuellement mais pas pour les lecteurs d'écran. Cela permet de ne pas encombrer visuellement la page web avec des liens d'évitement. Mais les lecteurs d'écran peuvent toujours les lire.

Exemple de style css pour cacher les liens d'évitement visuellement :

```css
.skip-link {
  position: absolute;
  top: -40px;
  left: 0;
  background-color: #000;
  color: #fff;
  padding: 1rem;
  z-index: 100;
}

.skip-link:focus {
  top: 0;
}
```

Donc notre lien d'évitement devient :

```html
<a href="#main" class="skip-link">Aller au contenu principal</a>
```

7. Les images

Page d'information sur la balise img: https://developer.mozilla.org/fr/docs/Web/HTML/Element/img

Nous allons maintenant voir comment insérer des images dans une page web. Pour cela, nous allons utiliser la balise `img` qui est une balise qui sert à insérer une image dans une page web. Elle est composée d'un attribut `src` qui contient l'URL de l'image.

> *Syntaxe :*
>
> ```html
> <img src="URL" />
> ```
>
> *Exemple :*
>
> ```html
> <img
>   src="https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.p
> />
> ```

Résultat : >

Ici l'image est affichée directement mais les lecteurs d'écran ne pourront pas indiquer à l'utilisateur qu'il y a une image. Il faut donc ajouter un texte alternatif à l'image via l'attribut `alt` de la balise `img`. Ce texte alternatif est affiché si l'image ne peut pas être affichée ou ce texte sera lu par les lecteurs d'écran.

Syntaxe :

```
<img src="URL" alt="Texte alternatif" />
```

Exemple :

```
<img
  src="https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.p
  alt="Logo de Google"
/>
```

Résultat : >

L'attribut `alt` est utilisé pour décrire une image non décorative. C'est-à-dire une image qui contient des informations importantes. Par exemple, une image qui contient du texte. Cela permet aux lecteurs d'écran de lire le texte alternatif de l'image. Cela permet aussi aux moteurs de recherche de comprendre le contenu de l'image.

Maintenant, si vous avez une image décorative vous pouvez faire en sorte qu'elle soit ignorée par les lecteurs d'écran en ajoutant l'attribut `role="presentation"` à la balise `img`. Cela permet de ne pas encombrer les lecteurs d'écran avec des images décoratives. Et en mettant l'attribut `alt=""`. De plus, on ajoute l'attribut `aria-hidden="true"` pour indiquer aux lecteurs d'écran que l'image est cachée visuellement.

Syntaxe :

```
<img src="URL" alt="" role="presentation" aria-hidden="true" />
```

Exemple :

```
<img
  src="https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.p
  alt=""
  role="presentation"
  aria-hidden="true"
/>
```

Résultat : >

Dans le code ci-dessus, nous avons créé une image décorative. Cela permet de ne pas encombrer les lecteurs d'écran avec des images décoratives. Ils vont passer outre cette image.

7.1 Formats d'images pour le web

Il existe plusieurs formats d'images adaptés au web, chacun ayant ses propres avantages :

- **JPG** : Idéal pour les photographies ou les images avec de nombreux détails et couleurs. Il utilise une compression avec perte, ce qui signifie que certaines données de l'image sont perdues pour réduire sa taille. C'est un bon compromis entre qualité et taille de fichier.
- **PNG** : Ce format est adapté pour les images avec des zones de transparence. Il utilise une compression sans perte, ce qui signifie que la qualité de l'image est préservée, mais la taille du fichier peut être plus grande que celle d'un JPEG.
- **GIF** : Principalement utilisé pour les animations, il ne supporte que 256 couleurs contre 16 millions pour le PNG et le JPEG. Ce qui le rend moins adapté pour les photographies. Depuis que le format PNG supporte la transparence, le GIF est de moins utilisé. Mais, il est fort utilisé pour les memes et les animations.

- **SVG** : C'est un format d'image vectorielle, idéal pour les logos et les icônes. Il est basé sur XML et peut être redimensionné sans perte de qualité.

7.2 miniatures

Il est possible de créer des miniatures d'images. C'est-à-dire des images plus petites qui pointent vers une image plus grande. Cela permet de réduire le temps de chargement de la page web. En effet, l'image plus grande ne sera chargée que si l'utilisateur clique sur l'image plus petite.

Imaginons que vous faites une application médicale qui affiche des images de rayons X. Vous ne voulez pas que l'utilisateur doive attendre que toutes les images soient chargées avant de pouvoir utiliser l'application. Vous pouvez donc créer des miniatures d'images qui pointent vers les images de rayons X. Cela permet de réduire le temps de chargement de la page web. En effet, les images de rayons X ne seront chargées que si l'utilisateur clique sur les miniatures.

Exemple :

```
<a href="rayon-x-1.jpg">
  <img src="rayon-x-1-miniature.jpg" alt="Rayon X 1" />
</a>
<a href="rayon-x-2.jpg">
  <img src="rayon-x-2-miniature.jpg" alt="Rayon X 2" />
</a>
```

Dans le code ci-dessus, nous avons créé deux miniatures d'images qui pointent vers deux images de rayons X. Si on clique sur une miniature, on est redirigé vers l'image de rayon X correspondante.

7.2 balise title de la balise img

Nous allons maintenant voir l'attribut `title` de la balise `img`. Cet attribut permet de spécifier un titre pour l'image. Ce titre est affiché lorsque l'utilisateur passe la souris sur l'image. Evidemment, au niveau accessibilité, ce n'est pas un attribut accessible. Donc est-ce vraiment pertinent de l'utiliser ? 😉

> *Syntaxe :*

```
<img src="URL" title="valeur" />
```

Exemple :

```
<img
  src="https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.p
  title="Logo de Google"
/>
```

Résultat : >

Dans le code ci-dessus, nous avons créé une image qui pointe vers le logo de Google. Si on passe la souris sur cette image, on voit apparaître le titre `Logo de Google`.

7.3 Base 64

Une autre façon d'insérer une image dans une page web est d'utiliser le format Base 64. C'est un format qui permet d'encoder une image en texte. Cela permet d'insérer une image directement dans le code HTML ou dans du CSS. C'est-à-dire que l'image n'est pas chargée depuis un serveur mais directement depuis le code HTML ou CSS. Cela permet d'éviter une requête HTTP. C'est-à-dire que le navigateur n'a pas besoin de faire une requête HTTP pour récupérer l'image. Cela permet donc d'améliorer les performances de la page web.

Ce n'est pas spécialement courant mais il faut le savoir. Cela peut être utile dans certains cas.

Pour encoder une image en Base 64, il faut utiliser un convertisseur. Par exemple, vous pouvez utiliser le convertisseur suivant : Base64 Image Encoder.

Voici un exemple d'image encodée en Base 64 :

```
<img
  src="
  alt="Image d'un carré"
/>
```

Résultat : Testez dans une page web et dites-moi si vous voyez une image. ☺ Si oui, quelle est l'image ?

Il faut bien comprendre que l'image est encodée en Base 64. C'est-à-dire que le code HTML contient le texte de l'image. C'est pour cela que la valeur de l'attribut `src` commence par ``. C'est le format de l'image qui est indiqué. Ici, c'est une image PNG. Ensuite, il y a le texte de l'image encodée en Base 64. C'est pour cela que le texte est très long.

Elle possède différents attributs qui permettent de modifier l'image :

- `alt` : texte alternatif qui s'affiche si l'image ne peut pas être affichée.
- `width` : largeur de l'image en pixels ou en pourcentage.
- `height` : hauteur de l'image en pixels ou en pourcentage.
- `title` : titre de l'image qui s'affiche lorsque l'utilisateur passe la souris sur l'image. Mais elle n'est pas lue par les lecteurs d'écran et donc n'est pas un attribut accessible.

Normalement, on ne manipule pas les attributs `width` et `height` en HTML. On les manipule en CSS. Mais je vous les donne quand même: Vade Retro Satanas ! 😄

Faites l'Exercice suivant: Exercices - images: Exercice 4

8. La balise

Nous allons maintenant voir la balise `br` qui est une balise qui sert à insérer un saut de ligne dans une page web. Elle ne possède pas de balise de fermeture.

> *Syntaxe :*
>
> ```
> <br />
> ```
>
> *Exemple :*

```
<p>
  Ceci est la première ligne.<br />
  Ceci est la deuxième ligne après un saut de ligne.
</p>
```

Résultat :

Ceci est la première ligne.
Ceci est la deuxième ligne après un saut de ligne.

Dans le code ci-dessus, nous avons créé un paragraphe qui contient deux lignes. La première ligne est suivie d'un saut de ligne. La deuxième ligne est affichée après le saut de ligne.

Si vous voulez insérer plusieurs sauts de ligne, vous pouvez utiliser la balise `br` plusieurs fois.

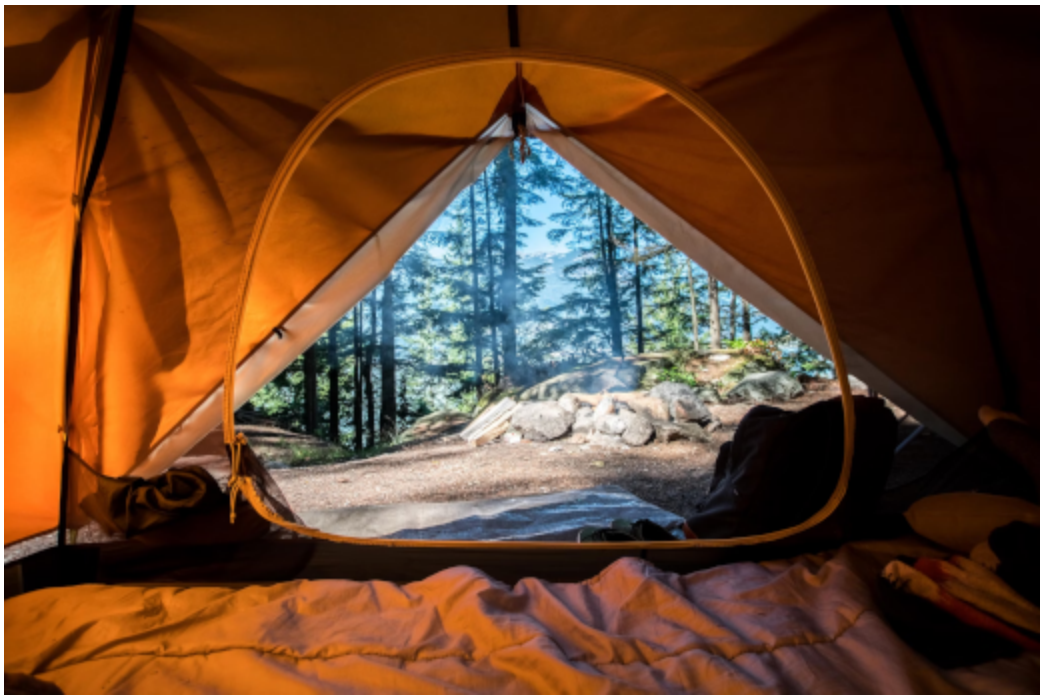
Si vous ne voulez pas coller un texte à une image,
n'utilisez pas la balise `br` pour insérer un saut de ligne entre le texte et l'image.

Mauvais Exemple fonctionnel:

```
<p>
  Ceci est un texte.<br />
  <img
    src="https://raw.githubusercontent.com/ZamBoyle/Eqla_HTML/master/Exercices/Images/ten
    alt="Vue d'une tente"
  />
</p>
```

Résultat :

Ceci est un texte.



En HTML, il est tout à fait valide de placer une image (`<img>`) à l'intérieur d'un élément de paragraphe (`<p>`).Voici un lien vers la spécification WHATWG pour l'élément `<p>` : WHATWG HTML Living Standard - The p element.

La spécification déclare que l'élément `<p>` peut contenir du contenu phrasé, et une image (`<img>`) fait partie du contenu phrasé, comme le décrit la spécification here. Vous pouvez donc inclure une image à l'intérieur d'un paragraphe de cette façon:

```
<p>
  Voici un paragraphe avec une
  <img src="image.jpg" alt="Description de l'image" /> image à l'intérieur.
</p>
```

9. Les listes

Nous allons maintenant voir comment créer des listes dans une page web. Il existe deux types de listes :

- les listes ordonnées
- les listes non ordonnées

9.1 Les listes ordonnées

Nous allons maintenant voir comment créer des listes ordonnées dans une page web. Pour cela, nous allons utiliser la balise `ol` qui est une balise qui sert à créer une liste ordonnée. Elle est composée d'une ou plusieurs balises `li` qui sont des balises qui servent à créer un élément de liste.

> Syntaxe :
>
> ```
> <ol>
>   <li>Élément 1</li>
>   <li>Élément 2</li>
>   <li>Élément 3</li>
> </ol>
> ```
>
> Résultat :
>
> 1. Élément 1
> 2. Élément 2
> 3. Élément 3

Dans le code ci-dessus, nous avons créé une liste ordonnée qui contient trois éléments de liste. Chaque élément de liste est créé avec la balise `li` et est numéroté automatiquement.

Si vous voulez changer le numéro de départ de la liste, vous pouvez utiliser l'attribut `start` de la balise `ol`. Cet attribut permet de spécifier le numéro de départ de la liste.

On peut aussi changer le type de numérotation de la liste. Pour cela, on utilise l'attribut `type` de la balise `ol`. Cet attribut permet de spécifier le type de numérotation de la liste. Il peut prendre les valeurs suivantes :

- `1` : numérotation décimale (1, 2, 3, 4, 5, ...)
- `a` : numérotation alphabétique minuscule (a, b, c, d, e, ...)
- `A` : numérotation alphabétique majuscule (A, B, C, D, E, ...)
- `i` : numérotation romaine minuscule (i, ii, iii, iv, v, ...)
- `I` : numérotation romaine majuscule (I, II, III, IV, V, ...)

> *Syntaxe :*

```
<ol start="numéro de départ" type="type de numérotation">
  <li>Élément 1</li>
  <li>Élément 2</li>
  <li>Élément 3</li>
</ol>
```

Exemple :

```
<ol start="5" type="i">
  <li>Élément 1</li>
  <li>Élément 2</li>
  <li>Élément 3</li>
</ol>
```

Résultat :

5. Élément 1
6. Élément 2
7. Élément 3

Dans le code ci-dessus, nous avons créé une liste ordonnée qui contient trois éléments de liste. Chaque élément de liste est créé avec la balise `li`. Le numéro de départ de la liste est 5. Le type de numérotation de la liste est romaine minuscule.

9.2 Les listes non ordonnées

Nous allons maintenant voir comment créer des listes non ordonnées dans une page web. Pour cela, nous allons utiliser la balise `ul` qui est une balise qui sert à créer une liste non ordonnée. Elle est composée d'une ou plusieurs balises `li` qui sont des balises qui servent à créer un élément de liste.

Syntaxe :

```
<ul>
  <li>Élément 1</li>
  <li>Élément 2</li>
  <li>Élément 3</li>
</ul>
```

Résultat :

- Élément 1
- Élément 2
- Élément 3

Dans le code ci-dessus, nous avons créé une liste non ordonnée qui contient trois éléments de liste. Chaque élément de liste est créé avec la balise `li`.

9.3 Les listes imbriquées

Il est possible d'imbriquer des listes. C'est-à-dire de mettre une liste dans une autre liste. Cela permet de créer des sous-listes.

Exemple :

```
<ol>
  <li>Élément 1</li>
  <li>
    Élément 2
    <ul>
      <li>Sous-élément 1</li>
      <li>Sous-élément 2</li>
      <li>Sous-élément 3</li>
    </ul>
  </li>
  <li>Élément 3</li>
</ol>
```

Résultat :

1. Élément 1

2. Élément 2
    - Sous-élément 1
    - Sous-élément 2
    - Sous-élément 3
3. Élément 3

Dans le code ci-dessus, nous avons créé une liste ordonnée qui contient trois éléments de liste. Le deuxième élément de liste contient une liste non ordonnée qui contient trois éléments de liste.

Faites l'Exercice suivant: Exercices - listes: Exercice 5

10. Mise en évidence

Nous allons voir comment mettre en évidence notre texte: en italique, en gras et un marqué le texte.

En effet, il est souvent utile de mettre en évidence des portions de texte pour attirer l'attention.

10.1 Mettre en italique et <i>

Nous allons maintenant voir comment mettre en italique une portion de texte. Pour cela, nous allons utiliser la balise `em` qui est une balise qui sert à mettre en évidence une portion de texte.

Exemple:

```
<em>Ce texte en italique</em> mais pas celui-ci.
```

Résultat:
Ce texte en italique mais pas celui-ci.

Dans le code ci-dessus, nous avons créé une portion de texte qui est mise en évidence en italique.

Concernant les lecteurs d'écran, voici ce que dit la documentation de Mozilla Developer Network :
An example for `<em>` could be: "Just do it already!", or: "We had to do something about it". A person or software reading the text would pronounce the words in italics with an emphasis, using

verbal stress.

On peut utiliser la balise `<u>` pour souligner une portion de texte. Mais il est préférable d'utiliser la balise `<em>` pour mettre en évidence une portion de texte. En effet, les lecteurs d'écran ne liront pas le texte souligné. D'un point accessibilité, il est donc préférable d'utiliser la balise `<em>` pour mettre en évidence une portion de texte.

Exemple :

```
Un texte comme <em>important</em> serait lu avec une intonation qui indique
l'importance du mot.
```

Résultat :
Un texte comme *important* serait lu avec une intonation qui indique l'importance du mot.

10. Mettre en gras et <b>

Nous allons maintenant voir comment mettre en gras une portion de texte. Pour cela, nous allons utiliser la balise `strong` qui est une balise qui sert à mettre en gras une portion de texte. En fait, `strong` veut dire important. On met donc en évident des mots importants.

On peut utiliser la balise `<b>` pour mettre en gras une portion de texte. Mais il est préférable d'utiliser la balise `<strong>` pour mettre en gras une portion de texte. En effet, les lecteurs d'écran ne liront pas le texte en gras. D'un point accessibilité, il est donc préférable d'utiliser la balise `<strong>` pour mettre en évidence une portion de texte.

Exemple:

```
<strong>Ce texte en gras</strong> mais pas celui-ci.
```

Résultat:
Ce texte en gras mais pas celui-ci.

Dans le code ci-dessus, nous avons créé une portion de texte qui est mise en gras.

10.3 Marquer le texte <mark>

Nous allons maintenant voir comment marquer une portion de texte. Pour cela, nous allons

utiliser la balise `mark` qui est une balise qui sert à marquer une portion de texte.

Exemple:

```
<mark>Ce texte est marqué</mark> mais pas celui-ci.
```

Résultat:
Ce texte est marqué mais pas celui-ci.

10.4 Souligner le texte <u>

Nous allons maintenant voir comment souligner une portion de texte. Pour cela, nous allons utiliser la balise `u` qui est une balise qui sert à souligner une portion de texte.

Exemple:

```
<u>Ce texte est souligné</u> mais pas celui-ci.
```

Résultat:
Ce texte est souligné mais pas celui-ci.

Les lecteurs d'écran ne liront pas le texte souligné. D'un point accessibilité, il est donc préférable d'utiliser la balise `em` pour mettre en évidence une portion de texte.

Faîtes l'Exercice suivant: Exercices - mise en évidence: Exercice 6

11. Structuration d'une page web / Sémantique

Pour le moment, nous avons vu comment structurer le texte d'une page web. Mais nous n'avons pas encore vu comment structurer la page web elle-même. C'est-à-dire comment structurer les différentes parties de la page web.

On parle aussi de sémentique. C'est-à-dire que l'on va donner du sens à notre page web. Cela permet d'améliorer l'accessibilité de la page web. Cela permet aussi d'améliorer le référencement de la page web. Et les lecteurs d'écran peuvent utiliser ces balises pour naviguer dans la page web.

Nous allons maintenant voir comment structurer une page web. Pour cela, nous allons utiliser les balises suivantes :

- `<header>` : balise qui sert à créer l'en-tête de la page web.
- `<nav>` : balise qui sert à créer la barre de navigation de la page web.
- `<main>` : balise qui sert à créer le contenu principal de la page web.
- `<footer>` : balise qui sert à créer le pied de page de la page web.
- `<aside>` : balise qui sert à créer une section de la page web qui est indépendante du contenu principal de la page web.
- `<section>` : balise qui sert à créer une section de la page web.
- `<article>` : balise qui sert à créer un article de la page web.
- `<div>` : balise qui sert à créer une division de la page web.

Elles ne sont pas obligatoires mais il est recommandé de les utiliser. En effet, cela permet de structurer la page web et donc d'améliorer l'accessibilité de la page web. Cela permet aussi d'améliorer le référencement de la page web. Et les lecteurs d'écran peuvent utiliser ces balises pour naviguer dans la page web.

11.1 La balise <header>

Nous allons maintenant voir la balise `header` qui est une balise qui sert à créer l'en-tête de la page web. Elle est composée d'une ou plusieurs balises `h1` à `h6` qui sont des balises qui servent à créer un titre.

> Exemple :
>
> ```
> <header>
>   <h1>Titre principal</h1>
>   <img
>     src="https://live.mdnplay.dev/fr/docs/Web/HTML/Element/header/mdn-logo-sm.png"
>     alt="Logo de MDN"
>   />
> </header>
> ```
>
> Résultat :

Titre principal

Dans le code ci-dessus, nous avons créé l'en-tête de la page web. L'en-tête de la page web contient un titre principal et un logo.

11.2 La balise <nav>

Nous allons maintenant voir la balise `nav` qui est une balise qui sert à créer la barre de navigation de la page web. Elle est composée d'une ou plusieurs balises `a` qui sont des balises qui servent à créer un lien.

On peut mettre nav avant, après voire dans header. Cela dépend de la structure de la page web.

Voici un exemple de barre de navigation :

```
<nav role="navigation">
  <ul>
    <li><a href="#main">Aller au contenu principal</a></li>
    <li><a href="https://www.example.com/">Accueil</a></li>
    <li><a href="https://www.example.com/page1.html">Page 1</a></li>
    <li><a href="https://www.example.com/page2.html">Page 2</a></li>
  </ul>
</nav>
```

Résultat :

- Aller au contenu principal
- Accueil
- Page 1
- Page 2

Dans le code ci-dessus, nous avons créé la barre de navigation de la page web. La barre de navigation contient quatre liens.

Ceux-ci se présente comme une liste et non sur une seule ligne (block).

Vous verrez plus tard comment mettre en forme cette liste dans le cours de CSS.

Vous noterez que le premier lien est un lein d'évitement / skip link. Il permet d'aller directement au contenu principal de la page web. Cela permet aux utilisateurs qui utilisent un clavier de ne pas avoir à parcourir toute la barre de navigation pour accéder au contenu principal de la page web.

Normalement ce lien d'évitement est caché visuellement mais pas pour les lecteurs d'écran. Pour le cacher visuellement, on utilise généralement la technique suivante :

```css
.navigation-container ul li {
  list-style-type: none; /* Supprime les puces */
}

.skip-link {
  position: absolute;
  top: -40px;
  left: 0;
  background: #000;
  color: #fff;
  padding: 8px;
  z-index: 100;
  transition: top 0.3s;
}

.skip-link:focus {
  top: 0;
}
```

Vous le verrez plus tard dans le cours de CSS / Accessibilité.

> Voici un exemple de barre de navigation :

```
<nav class="navigation-container" role="navigation">
  <ul>
    <li><a href="#main" class="skip-link">Aller au contenu principal</a></li>
    <li><a href="https://www.example.com/">Accueil</a></li>
    <li><a href="https://www.example.com/page1.html">Page 1</a></li>
    <li><a href="https://www.example.com/page2.html">Page 2</a></li>
  </ul>
</nav>
```

Résultat avec le skip link caché:

- Accueil
- Page 1
- Page 2

11.3 La balise <main>

Nous allons maintenant voir la balise `main` qui est une balise qui sert à créer le contenu principal de la page web. Elle est composée d'une ou plusieurs balises `h1` à `h6` qui sont des balises qui servent à créer un titre.

Exemple :

```
<main>
  <h2>Titre de la section 1</h2>
  <p>Contenu de la section 1</p>
  <h2>Titre de la section 2</h2>
  <p>Contenu de la section 2</p>
</main>
```

Résultat :

Titre de la section 1

Contenu de la section 1

> # Titre de la section 2
>
> Contenu de la section 2

Dans le code ci-dessus, nous avons créé le contenu principal de la page web. Le contenu principal de la page web contient deux sections. Chaque section contient un titre et un paragraphe.

11.4 La balise <section>

Nous allons maintenant voir la balise `section` qui est une balise qui sert à créer une section de la page web. Elle est composée d'une ou plusieurs balises `h1` à `h6` qui sont des balises qui servent à créer un titre. Elle est aussi composée d'une ou plusieurs balises `p` qui sont des balises qui servent à créer un paragraphe. Elle fait partie de la balise `main`.

> Exemple :
>
> ```
> <main id="main-content">
>   <section>
>     <h2>Titre de la section</h2>
>     Contenu de la section
>   </section>
> </main>
> ```
>
> Résultat :
>
> # Titre de la section
>
> Contenu de la section

Dans le code ci-dessus, nous avons créé une section de la page web. Cette section contient un titre et un paragraphe.

11.4 La balise <footer>

Nous allons maintenant voir la balise `footer` qui est une balise qui sert à créer le pied de page de la page web. Elle est composée d'une ou plusieurs balises `p` qui sont des balises qui servent à créer un paragraphe.

> Exemple :
>
> ```
> <footer>
>   Contenu du pied de page
> </footer>
> ```
>
> Résultat :
>
> Contenu du pied de page

Dans le code ci-dessus, nous avons créé le pied de page de la page web. Le pied de page de la page web contient un paragraphe.

11.5 La balise <aside>

Nous allons maintenant voir la balise `aside` qui est une balise qui sert à créer une section de la page web qui est indépendante du contenu principal de la page web. Elle est composée d'une ou plusieurs balises `p` qui sont des balises qui servent à créer un paragraphe.

Ell est souvent utilisée pour créer une barre latérale.

> Exemple :
>
> ```
> <aside>
>   Contenu de la section indépendante
> </aside>
> ```
>
> Résultat :
>
> Contenu de la section indépendante

Dans le code ci-dessus, nous avons créé une section de la page web qui est indépendante du contenu principal de la page web. Cette section contient un paragraphe.

11.6 La balise <article>

Nous allons maintenant voir la balise `article` qui est une balise qui sert à créer un article de la page web. Elle est composée d'une ou plusieurs balises `h1` à `h6` qui sont des balises qui servent à créer un titre. Elle est aussi composée d'une ou plusieurs balises `p` qui sont des balises qui servent à créer un paragraphe.

Exemple :

```
<article>
  <h2>Titre de l'article</h2>
  <p>Contenu de l'article</p>
</article>
```

Résultat :

Titre de l'article

Contenu de l'article

Dans le code ci-dessus, nous avons créé un article de la page web. Cet article contient un titre et un paragraphe.

11.7 Différence entre <section> et <article>

Différence entre section et article :

- `<section>` : une section est une partie d'une page web. Elle peut contenir plusieurs articles. Elle peut aussi contenir d'autres sections. Utilisé pour regrouper du contenu thématiquement cohérent, comme les chapitres d'un document.
- `<article>` : Représente un contenu indépendant et autonome qui pourrait être

extrait et réutilisé dans un autre contexte tout en restant compréhensible (par exemple, un article de blog, un commentaire utilisateur).

Exemple :

```
<section>
  <article>
    <h2>Titre de l'article 1</h2>
    <p>Contenu de l'article 1...</p>
  </article>
  <article>
    <h2>Titre de l'article 2</h2>
    <p>Contenu de l'article 2...</p>
  </article>
</section>
```

Résultat :

Titre de l'article 1

Contenu de l'article 1...

Titre de l'article 2

Contenu de l'article 2...

Dans le code ci-dessus, nous avons créé une section de la page web. Cette section contient deux articles. Chaque article contient un titre et un paragraphe.

11.9 Exemple complet

Voici un exemple complet de structuration d'une page web : Cliquez ici.

12. Balises de type block et inline

En HTML, les éléments sont catégorisés en tant qu'éléments de block ou éléments en ligne (inline), en fonction de la manière dont ils sont affichés dans le navigateur et de leur comportement dans le flux du document.

12.1 Balise de Type Block

Les balises de type block sont des éléments qui occupent toute la largeur disponible de leur conteneur parent et provoquent un retour à la ligne avant et après leur contenu. Ils sont généralement utilisés pour définir la structure du document et pour grouper d'autres éléments, à la fois de block et en ligne. Les éléments de block peuvent contenir d'autres éléments de block et des éléments en ligne.

Exemples de balises de bloc:

- `<div>`
- `<p>`
- `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`
- `<ul>`, `<ol>`, `<li>`
- `<table>`

12.2 Balise de Type Inline

À l'inverse, les éléments en ligne (inline) n'occupent que l'espace nécessaire pour afficher leur contenu et ne provoquent pas de retour à la ligne. Ils sont généralement utilisés pour styliser ou manipuler du texte ou d'autres éléments en ligne sans perturber le flux du document. Les éléments en ligne ne peuvent contenir que d'autres éléments en ligne.

Exemples de balises en ligne:

- `<span>`
- `<a>`
- `<img>`
- `<strong>`
- `<em>`

Dans cet exemple, la balise `<div>` et la balise `<p>` sont des éléments de bloc, tandis que la balise `<span>` est un élément en ligne utilisé pour changer la couleur du texte à l'intérieur de l'élément de bloc `<p>`.

12.3 La balise - type block

En HTML, la balise `<div>` est une balise de `bloc` (block element).

Elle est utilisée pour regrouper d'autres éléments HTML. Elle n'a pas de signification spécifique et ne représente aucune mise en forme ou style par défaut. Cependant, elle est très utile pour appliquer du style ou effectuer des manipulations sur le contenu groupé à l'aide de CSS et de JavaScript.

Voici un exemple simple de l'utilisation de la balise `<div>` :

- site.css:

```css
.maDiv {
  background-color: lightblue;
  padding: 20px;
  text-align: center;
}

h1 {
  text-align: center;
  text-decoration: underline;
}

body{
  margin-inline: 10%;
}
```

- site.html:

```
<!DOCTYPE html>
<html lang="fr-BE">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width" />
    <link rel="stylesheet" href="site.css" />
    <title>Exemple de la balise div</title>
  </head>
  <body>
    <header>
        <h1>Exemple de la balise div</h1>
    </header>
    <main>
        <article>
            <div class="maDiv">
                <p>Voici un paragraphe à l'intérieur d'une div.</p>
                <p>La div est utilisée pour grouper des éléments et les styler ensemble.</
            </div>
        </article>
    </main>
    <footer>
        <p>© 2023 Mon Site Web. Tous droits réservés.</p>
    </footer>
  </body>
</html>
```

Pour voir le résultat cliquez ici.

Dans cet exemple, la balise `<div>` contient deux paragraphes, et la classe CSS `.maDiv` est utilisée pour appliquer un style à tout le contenu de la balise `<div>`. Le style CSS définit ici une couleur de fond, une marge intérieure (padding), et un alignement du texte au centre pour la `<div>`.

En résumé:

- La balise `<div>` est une balise de conteneur utilisée pour grouper d'autres éléments HTML.
- Elle sert souvent à appliquer du style ou à manipuler plusieurs éléments en tant que groupe à l'aide de CSS et de JavaScript.

- C'est une balise de block, ce qui signifie qu'elle occupe toute la largeur disponible de son conteneur parent et provoque un retour à la ligne avant et après son contenu.
- Les `<div>` sont souvent utilisés en conjonction avec les classes et les identifiants CSS pour appliquer un style spécifique aux éléments groupés.

CSS:

```css
#monIdentifiant {
  color: red;
  font-size: 20px;
}

.maClasse {
  background-color: yellow;
  padding: 10px;
}

div {
  margin-top: 5px;
  border-radius: 10px;
  border-style: dotted;
}
```

HTML:

```
<!DOCTYPE html>
<html lang="fr-BE">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width" />
    <link rel="stylesheet" href="site.css" />
    <title>Exemples de la balise div</title>
  </head>
  <body>
    <h1>Exemples de la balise div</h1>
    <h2>1. Un div avec un style via un id et une classe</h2>
    <div id="monIdentifiant" class="maClasse">
        <p>Voici un paragraphe à l'intérieur d'une div.</p>
        <p>Le div est utilisée pour grouper des éléments et les styler ensemble.</p>
    </div>
    <h2>2. Un div avec un style via une classe</h2>
    <div class="maClasse">
        <p>Voici un autre paragraphe à l'intérieur d'une div.</p>
        <p>Le div est utilisée pour grouper des éléments et les styler ensemble.</p>
    </div>
    <h2>3. Un div avec un style via la balise</h2>
    <div>
        <p>Voici un autre paragraphe à l'intérieur d'une div.</p>
        <p>Le div est utilisée pour grouper des éléments et les styler ensemble.</p>
    </div>
  </body>
</html>
```

Pour voir le résultat cliquez ici.

Dans le premier div, nous avons utilisé un identifiant et une classe. Il y aura trois styles appliqués à ce div: le style de l'identifiant, le style de la classe et le style de la balise div.

Dans le deuxième div, nous avons utilisé uniquement une classe. Cela montre que nous pouvons utiliser un identifiant et/ou une classe pour styler une div. Il y aura deux styles appliqués à ce div: le style de la classe et le style de la balise div.

Dans le troisième div, nous n'avons utilisé ni identifiant ni classe. Il n'y aura qu'un seul style appliqué à ce div: le style de la balise div.

12.4 La balise - type inline

La balise `<span>` en HTML est une balise de type `inline` (inline element) qui est utilisée pour grouper ou appliquer un style à une portion de texte dans le document, sans changer la sémantique du contenu. Par elle-même, la balise `<span>` ne provoque aucun changement visuel ou de formatage. Cependant, elle est très utile lorsqu'elle est utilisée avec du CSS ou du JavaScript pour appliquer des styles, des animations ou d'autres manipulations à une partie spécifique du texte.

Exemple:
Voici un exemple montrant à la fois des éléments de bloc et des éléments en ligne:

- site.css:

```
.important {
  color: red;
  text-decoration: underline;
  font-weight: bold;
  font-style: italic;
}
```

- site.html:

```
<!DOCTYPE html>
<html lang="fr-BE">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width" />
    <link rel="stylesheet" href="site.css" />
    <title>Exemple de balises Block et Inline</title>
  </head>
  <body>
    <h1>Exemple de balises Block et Inline</h1>
    <div>
        <p>
            La balise span est un <span class="important">élément en ligne (inline)</span>
            l'intérieur d'un élément de block (le p) et qui est lui-même à l'intérieur d'u
        </p>
    </div>
  </body>
</html>
```

Dans cet exemple, la balise `<span>` est utilisée pour appliquer un style à une partie du texte à l'intérieur de la balise `<p>`. La classe CSS `.important` est utilisée pour appliquer un style à tout le contenu de la balise `<span>`. Le style CSS définit ici une couleur de texte rouge et en gras pour la balise `<span>`.
Nous avons également utilisé la balise `<div>` pour grouper le contenu de la balise `<p>` et la balise `<span>`. Nous avons donc un bloc (p) dans un bloc (div).

Résultat:
Pour voir le résultat cliquez ici.

12.5 Exercices - Div & Span

Faîtes l'Exercice suivant: Exercices - Div & Span: Exercice 8

13. Les tableaux

Un tableau est une structure de données qui permet de stocker des données sous forme de lignes et de colonnes. Il est composé de cellules qui sont organisées en lignes et en colonnes, où

chaque cellule contient une donnée.

13.1 Création d'un tableau en HTML

En HTML, un tableau est structuré en utilisant différentes balises, chacune ayant un rôle spécifique dans la création du tableau :

- `table` : Définit l'ensemble du tableau.
- `thead` : Contient les en-têtes des colonnes du tableau.
- `tbody` : Contient les données ou le contenu principal du tableau.
- `tfoot` : Utilisé pour regrouper les pieds de colonne du tableau (souvent pour des résumés ou des totaux).
- `tr` : Représente une ligne du tableau.
- `th` : Représente une cellule d'en-tête, définissant le titre d'une colonne.
- `td` : Représente une cellule standard contenant des données.

Exemple de Code HTML pour un Tableau
page web:

```html
<!DOCTYPE html>
<html lang="fr-BE">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="site.css">
    <title>Exemple de tableau en HTML</title>
</head>
<body>
    <h1>Exemple de tableau en HTML</h1>
    <table>
        <thead>
            <tr>
                <th scope="col">Entête 1</th>
                <th scope="col">Entête 2</th>
                <th scope="col">Entête 3</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Ligne 1, Colonne 1</td>
                <td>Ligne 1, Colonne 2</td>
                <td>Ligne 1, Colonne 3</td>
            </tr>
            <tr>
                <td>Ligne 2, Colonne 1</td>
                <td>Ligne 2, Colonne 2</td>
                <td>Ligne 2, Colonne 3</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <td colspan="3">Pied du tableau</td>
            </tr>
        </tfoot>
    </table>
</body>
</html>
```

Résultat:
Pour voir le résultat: Exemple de tableau simple.

Dans cet exemple, chaque balise joue un rôle clé dans la structuration du tableau :

- Les balises `<th>` dans `<thead>` créent l'en-tête du tableau, et ont un style distinct pour se démarquer du reste du tableau.
- Les balises `<td>` dans `<tbody>` contiennent les données du tableau.
- Les balises `<td>` dans `<tfoot>` offrent un espace pour des informations supplémentaires ou des résumés à la fin du tableau.

Cependant, on constate qu'il n'est pas toujours facile de distinguer les données du tableau. En effet, nous n'avons pas de bordures pour délimiter les cellules du tableau. Nous n'avons pas non plus de couleur de fond pour les en-têtes de colonne du tableau.

13.2 Affichage de bordures et de couleurs de fond

Améliorons donc notre tableau en ajoutant des bordures et des couleurs de fond.

site.css:

```
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    border: 1px solid black;
    padding: 8px;
}
th {
    background-color: #958d8d;
}
tfoot {
    background-color: #958d8d;
}
```

Résultat:
Pour voir le résultat: Tableau avec bordures et couleur de fond.

Les CSS intégrées garantissent que le tableau est visuellement clair et lisible: bordures, couleurs de fond, espacement, etc.

Cet exemple illustre un tableau en HTML, bien que les tables puissent devenir beaucoup plus complexes selon les besoins des données à afficher.

Notons que nous avons utilisé l'attribut `collaspe` pour la balise `<table>`. Cet attribut permet de fusionner les bordures des cellules du tableau. Cela permet d'avoir un tableau plus lisible. Sinon, nous aurions un effet de double bordure.

13.3 colspan et rowspan

Les attributs `colspan` et `rowspan` sont utilisés pour fusionner des cellules de tableau. Ils sont utilisés pour fusionner des cellules de tableau horizontalement et verticalement respectivement.

Dans l'exemple précédent, nous avons utilisé l'attribut `colspan` pour fusionner les cellules de la dernière ligne du tableau (`tfoot`). Sinon nous n'aurions eu qu'une seule cellule dans cette ligne.

```
<table>
    <thead>
        <tr>
            <th>Dessin animé</th>
            <th>Pour enfants ?</th>
            <th>Pour adolescents ?</th>
            <th>Pour adultes ?</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Dragon Ball</td>
            <td colspan="3" id="col1">C'est la base, c'est universel ! :-)</td>
        </tr>
        <tr>
            <td>Ken le survivant</td>
            <td>Non trop violent</td>
            <td>Limite, encore trop violent </td>
            <td>Si on aime l'hémoglobine</td>
        </tr>
        <tr>
            <td>Les Titounis</td>
            <td>Les tout-tout petits aimeront</td>
            <td>Ils fredonneront les chansons !</td>
            <td id="row1" rowspan="2">Les parents apprécieront le silence.</td>
        </tr>
        <tr>
            <td>Les Minikeums</td>
            <td>Oui sans problème !</td>
            <td>Se lasseront vite</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <th>Dessin animé</th>
            <th>Pour enfants ?</th>
            <th>Pour adolescents ?</th>
            <th>Pour adultes ?</th>
        </tr>
```

```
    </tfoot>
</table>
```

On peut évidemment faire des colspan et rowspan ailleurs dans le tableau: n'importe où dans le `<thead>`, `<tbody>` ou `<tfoot>`.

Voici un exemple de colspan et rowspan: Tableau avec colspan et rowspan.

13.4 Colorer les lignes paires et impaires

Lorsqu'un tableau contient beaucoup de lignes, il peut être difficile de suivre les données d'une ligne à l'autre. Pour faciliter la lecture des données, il est possible de colorer les lignes paires et impaires d'un tableau. Pour cela, on utilise le pseudo-sélecteur `nth-child` avec la valeur `even` pour les lignes paires et la valeur `odd` pour les lignes impaires.

```
tr:nth-child(even) {
    background-color: #f2f2f2;
}
```

Voici un exemple de coloration des lignes paires et impaires d'un tableau: Tableau ligns paires/impaires

13.5 Caption, scope, headers, id

La balise `caption` est utilisée pour ajouter un titre à un tableau. Elle est placée juste après la balise `<table>`.

L'attribut `scope` est utilisé pour définir le champ d'application d'un en-tête de tableau. Il peut prendre les valeurs `row` ou `col` pour définir le champ d'application d'un en-tête de tableau à une ligne ou à une colonne.

L'attribut `headers` est utilisé pour définir les en-têtes de colonne associés à une cellule de données. Il contient une liste d'identifiants séparés par des espaces. Chaque identifiant correspond à l'attribut `id` d'un en-tête de colonne.

L'attribut `id` est utilisé pour définir un identifiant unique pour un élément HTML. Il est utilisé pour associer un en-tête de colonne à une cellule de données.

```
<table>
    <caption>Dessins animés et leur public</caption>
    <thead>
        <tr>
            <th id="c1" scope="col">Dessin animé</th>
            <th id="c2" scope="col">Pour enfants ?</th>
            <th id="c3" scope="col">Pour adolescents ?</th>
            <th id="c4" scope="col">Pour adultes ?</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td headers="c1">Dragon Ball</td>
            <td colspan="3" headers="c2 c3 c4">C'est la base, c'est universel ! :-)</td>
        </tr>
        <tr>
            <td>Ken le survivant</td>
            <td>Non trop violent</td>
            <td>Limite, encore trop violent </td>
            <td>Si on aime l'hémoglobine</td>
        </tr>
        <tr>
            <td>Les Titounis</td>
            <td>Les tout-tout petits aimeront</td>
            <td>Ils fredonneront les chansons !</td>
            <td rowspan="2">Les parents apprécieront le silence.</td>
        </tr>
        <tr>
            <td>Les Minikeums</td>
            <td>Oui sans problème !</td>
            <td>Se lasseront vite</td>
        </tr>
    </tbody>
</table>
```

Résultat: Tableau avec caption, scope, headers, id

Dans l'exemple ci-dessus, nous avons ajouté un titre au tableau à l'aide de la balise `<caption>`.

Nous avons aussi utilisé l'attribut `scope` pour définir le champ d'application des en-têtes de colonne. Nous avons aussi utilisé l'attribut `headers` pour associer les en-têtes de colonne aux cellules de données.

Enfin, nous avons utilisé l'attribut `id` pour définir un identifiant unique pour chaque en-tête de colonne.

Pour le colspan="3" de la ligne 2, nous avons utilisé l'attribut `headers` pour associer les en-têtes de colonne aux cellules de données. En effet, nous avons associé les en-têtes de colonne aux cellules de données en utilisant l'attribut `headers` et l'attribut `id`. C'est à dire: headers="c2 c3 c4". Ce qui peut sembler étonnant c'est de donner un headers à "Dragon Ball", ce n'est pas nécessaire mais c'est une bonne pratique dans cet exemple avec le colspan="3" de la ligne 2.

Vous pourriez vous demander quand utiliser scope="row". C'est quand vous avez un tableau avec des en-têtes de ligne. Dans ce cas, vous pouvez utiliser scope="row" pour définir le champ d'application des en-têtes de ligne. Mais ça devient alors des tableaux plus complexes.

13.6 Pour aller plus loin

Je vous invite à visiter le lien suivant pour en savoir plus sur les tableaux en HTML: https://www.w3.org/WAI/tutorials/tables/

Vous y trouverez des exemples allant de simples tableaux à des tableaux plus complexes.

13.7 Exercices - Tableaux

Faîtes l'Exercice suivant: Exercices - Tableaux

14. Les formulaires

Un formulaire HTML est utilisé pour collecter des informations auprès des utilisateurs. Il contient des éléments de formulaire tels que des champs de saisie, des cases à cocher, des boutons radio, des boutons d'envoi, etc.

Si vous faites des vérifications côté client, n'oubliez pas de faire des vérifications côté serveur. En effet, il faut partir du principe que le formulaire peut être modifié côté client. Donc il faut toujours faire des vérifications côté serveur. La règle, c'est la double vérification: côté client et côté

serveur. Mais vous en parlerez peut-être au cours de PHP.

14.1 La balise <form>

La balise `<form>` est un élément de block utilisé pour créer un formulaire HTML pour l'entrée utilisateur. Un formulaire HTML peut contenir un ou plusieurs éléments de formulaire.

Les formulaires sont souvent l'endroit où l'accessibilité est mal implémentée sur un site web. Nous essayerons de faire au mieux pour rendre nos formulaires accessibles grâce aux attributs `aria-label`, `aria-labelledby`, `aria-describedby`, `aria-required`, `aria-

Exemple:

```
<form>
  <!-- éléments de formulaire -->
</form>
```

Ici nous avons simplement ajouté la balise form mais elle doit être utilisée avec plusieurs attributs: method, action

14.2 L'attribut `action`

Cet attribut est utilisé pour indiquer où envoyer notre formulaire. C'est à dire l'adresse qui va traîter notre formulaire complété. Il est important à noter qu'il est vraiment important de contacter un site web via un site sécurisé en HTTPS. De cette manière la transmission est cryptée de bout en bout.

Si l'attribut est absent ou n'a pas de valeur, il sera envoyé à la page en cours contenant le formulaire.

14.3 L'attribut `method`

Cet attribut sert à indiquer comment envoyer les valeurs de notre formulaire à la page de destination.
Il y a deux valeurs possibles:

- `GET`: on envoie les variables et leurs valeurs via l'url de destination définie dans l'attribut `action`.

Par exemple l'url pourrait ressembler à https://www.google.com/search?q=eqla où l'on contacte la page search, où on a passé la variable `q` avec la valeur `eqla`. Après la page on a un point d'interrogation suivit des variables et leurs valeurs. A notre qu'on utilise le symbole `&` pour ajouter une autre variable dans l'url de destination.
Notons enfin que comme les variables sont dans l'url, un petit curieux pourrait essayer de modifier celles-ci aisément en manipulant l'url... Mais s'il n'y a aucune données sensibles rien ne vous empêche d'envoyer via GET.
Enfin, nous sommes limités par la taille maximale de caractères dans l'url. Cette valeur dépend du naviateur utilisé: IE (2083), Chrome ()

- `POST` : c'est la méthode à prévilégier dans la mesure où les variables et valeurs sont envoyées dans la requête HTTP et ne figureront pas dans l'url. On peut bien entendu toujours voir les valeurs envoyées aisément avec l'outil réseau de l'outil de développement des navigateurs.

Donc notre balise `form` devrait ressemble à ceci

```
<form action="http://adresse" method="POST">
  <!-- éléments de formulaire -->
</form>
```

14.4 La balise <input>

C'est l'élément central d'un formulaire c'est via la balise `input` que vous allez encoder vos données.

Voici une liste des types de champs de formulaire les plus couramment utilisés en HTML :

1. On entre du **Texte** dans le champ et le type sera:
    - `type` =" `text` ": Un champ de texte simple pour une seule ligne de texte.
    - `type` =" `password` ": Un champ de texte qui masque les caractères saisis.
    - `type` =" `email` ": Un champ de texte qui doit contenir une adresse e-mail valide.
2. On entre un texte sur plusieurs lignes via la balise `<textarea>`
3. On entre des **nombres** dans le champ et le type sera :

    - `number` : Un champ de texte qui doit contenir un nombre.
4. On fait un **choix** dans le champ et le type sera :
    - `checkbox` : Une case à cocher.
    - `radio` : Un bouton radio.
    - `select` : Une liste déroulante.
5. On appuie sur un **Boutons** et le type sera :
    - `button` : Un bouton générique.
    - `submit` : Un bouton qui soumet le formulaire.
    - `reset` : Un bouton qui réinitialise le formulaire.
6. **Autres** :
    - `hidden` : Un champ caché utilisé pour stocker des données qui ne sont pas visibles ou modifiables par l'utilisateur. Sauf si l'utilisateur modifie la page avec un outil de développement: très facile.
    - `image` : Un champ qui permet à l'utilisateur de cliquer sur une image pour soumettre un formulaire.
    - `color` : Un champ qui permet à l'utilisateur de choisir une couleur.
7. **Attributs de validation** :
    - `required` : Spécifie que le champ doit être rempli avant de soumettre le formulaire.
    - `pattern` : Spécifie une expression régulière que la valeur du champ doit respecter.
    - `min` et `max` : Spécifient les valeurs minimale et maximale pour les champs numériques.
    - `maxlength` et `minlength` : Spécifient la longueur maximale et minimale du texte.

Chacun de ces types de champs peut être utilisé avec divers attributs HTML pour personnaliser leur comportement et leur apparence dans le formulaire HTML. Vous pouvez également utiliser JavaScript pour ajouter une logique supplémentaire et des validations de formulaire.

C'est via l'attribut `type` que vous allez définir le type de champ que vous voulez utiliser. C'est ce que nous allons voir dans la section suivante.

14.5 L'attribut type="text"

La valeur "texte" de l'attribut type indique que la balise input est un champ de texte simple pour

une seule ligne de texte. C'est le type de champ le plus couramment utilisé dans les formulaires HTML.

Exemple Démo:

```
<form action="http://zamboyle.synology.me:2727/forms/demos/demo14-5.php" method="POST">
  <label for="userName">Nom :</label>
  <input type="text" id="userName" name="userName" aria-required="true" required>

  <input type="submit" value="Envoyer">
</form>
```

Résultat: Exemple de champ de type texte

Dans cet exemple, nous avons utilisé l'attribut `type` avec la valeur `text` pour créer un champ de texte simple pour une seule ligne de texte.

Nous avons aussi utilisé l'attribut `name` pour définir le nom du champ de texte. C'est ce nom qui sera utilisé pour identifier le champ de texte lors de l'envoi du formulaire.

Nous avons aussi utilisé l'attribut `id` pour définir un identifiant unique pour le champ de texte. Cet identifiant est utilisé pour associer le champ de texte à son étiquette.

Cela permet aux lecteurs d'écran de lire le texte "Nom :" lorsqu'ils rencontrent le champ de texte.

Faites l'exercice suivant: Simple Formulaire

14.6 L'attribut type="email"

La valeur "email" de l'attribut type indique que la balise input est un champ de texte qui doit contenir une adresse e-mail valide.

Dans l'exemple précédent, nous avions utilisé l'attribut `type` avec la valeur `text` pour créer un champ de texte simple pour une seule ligne de texte. Mais nous n'avions pas de validation pour vérifier que l'utilisateur entre une adresse e-mail valide. C'est ici que l'attribut `type` avec la valeur `email` entre en jeu.

Exemple Démo:

```
<form action="http://zamboyle.synology.me:2727/forms/demos/demo14-6.php" method="POST">
  <label for="userEmail">Email :</label>
  <input type="email" id="userEmail" name="userEmail" aria-required="true" required>

  <input type="submit" value="Envoyer">
</form>
```

Résultat: Exemple de champ de email

Faites l'exercice suivant: Type Email

14.7 L'attribut required

L'attribut `required` est utilisé pour indiquer que le champ de texte est obligatoire. Cela permet aux lecteurs d'écran de lire le texte "obligatoire" lorsqu'ils rencontrent le champ de texte.

On le couple avec l'attribut `aria-required` pour indiquer que le champ de texte est obligatoire. Cela permet aux lecteurs d'écran de lire le texte "obligatoire" lorsqu'ils rencontrent le champ de texte.

Normalement avec les technologies d'assistance modernes, required devrait suffire. Mais il est recommandé d'utiliser les deux attributs pour une meilleure compatibilité avec les technologies d'assistance plus anciennes.

Exemple Démo:

```
  <input type="text" id="userName" name="userName" aria-required="true" required>
```

14.8 L'attribut type="password"

La valeur "password" de l'attribut type indique que la balise input est un champ de texte qui masque les caractères saisis. Cela permet de masquer les caractères saisis par l'utilisateur.

On va imposer une longueur minimale de 8 caractères avec l'attribut `minlength` ="8" dans l'exemple suivant:

```
    <form action="http://zamboyle.synology.me:2727/forms/demos/demo14-8.php" method="POST"
      <label for="userName">Username :</label><br>
      <input type="text" id="userName" name="userName" aria-required="true" required><br>

      <label for="userEmail">Email :</label><br>
      <input type="email" id="userEmail" name="userEmail" aria-required="true" required><

      <label for="userPassword1">Mot de passe :</label><br>
      <input type="password" id="userPassword1" name="userPassword1" minlength="8" aria-re

      <label for="userPassword2">Retaper le mot de passe :</label><br>
      <input type="password" id="userPassword2" minlength="8" name="userPassword2" aria-re
      <br>

      <input type="submit" value="Envoyer">
    </form>
```

Résultat: Exemple de champ de type password

14.9 La balise textarea

La balise `<textarea>` est utilisée pour créer un champ de texte multiligne. C'est-à-dire un champ de texte qui peut contenir plusieurs lignes de texte.

Cette balise peut être utilisée avec les attributs `cols` et `rows` pour définir le nombre de colonnes et de lignes du champ de texte.

```
<form action="http://zamboyle.synology.me:2727/forms/demos/demo14-9.php" method="POST">
    <label for="userFirstname">Prénom :</label><br>
    <input type="text" id="userFirstname" name="userFirstname" aria-required="true" requi
    <br>
    <label for="userWho">Qui es-tu ?</label><br>
    <textarea id="userWho" name="userWho" aria-required="true" required rows="10" cols="50
    <br>
    <input type="submit" value="Envoyer">
</form>
```

Résultat: Exemple de balise textarea

14.10 L'attribut type="number"

La valeur "number" de l'attribut type indique que la balise input est un champ de texte qui doit contenir un nombre.

On peut utiliser les attributs `min` et `max` pour définir les valeurs minimale et maximale du champ de texte.

```
<form action="http://zamboyle.synology.me:2727/forms/demos/demo14-10.php" method="POST">
    <label for="userFirstname">Prénom :</label><br>
    <input type="text" id="userFirstname" name="userFirstname" aria-required="true" requi
    <br>
    <label for="userAge">Age :</label><br>
    <input type="number" id="userAge" name="userAge" min="8" max="99" aria-required="true"
    <br>
    <input type="submit" value="Envoyer">
</form>
```

Résultat: Exemple de champ de type number

Faites l'exercice suivant: Type number

14.11 L'attribut type="checkbox"

La valeur "checkbox" de l'attribut type indique que la balise input est une case à cocher. C'est-à-dire une case à cocher qui peut être cochée ou décochée. On peut cocher plusieurs cases à cocher.

On peut utiliser l'attribut `checked` pour cocher la case à cocher par défaut.

```
<form action="http://zamboyle.synology.me:2727/forms/demos/demo14-11.php" method="POST">
    <label for="userFirstname">Prénom :</label><br>
    <input type="text" id="userFirstname" name="userFirstname" aria-required="true" requi
    <br>
    <label for="userAge">Age :</label><br>
    <input type="number" id="userAge" name="userAge" min="8" max="99" aria-required="true"
    <br>
    <label for="userIsStudent">Genre :</label><br>
    <input type="checkbox" id="userIsStudent" name="userIsStudent" checked> Etudiant<br>
    <br>
    <input type="submit" value="Envoyer">
</form>
```

Résultat: Exemple de champ de type checkbox

Si vous avez plusieurs cases à cocher, vous pouvez mettre dans l'attribut `name` une valeur qui contient à la fin des crochets `[]`. Cela permet de récupérer les valeurs des cases à cocher dans un tableau. Ca sera fort utile dans un langage côté serveur comme PHP.

```
<form action="http://zamboyle.synology.me:2727/forms/demos/demo14-11b.php" method="post">
    Quels fruits aimez-vous ?<br>
    <input type="checkbox" name="fruits[]" value="pomme"> Pomme<br>
    <input type="checkbox" name="fruits[]" value="orange"> Orange<br>
    <input type="checkbox" name="fruits[]" value="banane"> Banane<br>
    <input type="submit" value="Soumettre">
</form>
```

Résultat: Exemple de champ de type checkbox avec des crochets

14.12 L'attribut type="radio"

La valeur "radio" de l'attribut type indique que la balise input est un bouton radio. C'est-à-dire un bouton radio qui peut être sélectionné ou désélectionné. On ne peut sélectionner qu'un seul bouton radio.

On peut utiliser l'attribut `checked` pour sélectionner le bouton radio par défaut.

On utilise l'attribut `name` pour regrouper les boutons radio. C'est-à-dire que les boutons radio qui

ont le même nom sont regroupés. On utilise l'attribut `value` pour définir la valeur du bouton radio.

```
<form action="http://zamboyle.synology.me:2727/forms/demos/demo14-12.php" method="POST">
    <label for="userFirstname">Prénom :</label><br>
    <input type="text" id="userFirstname" name="userFirstname" aria-required="true" requi
    <br>
    <label for="userAge">Age :</label><br>
    <input type="number" id="userAge" name="userAge" min="8" max="99" aria-required="true"
    <br>
    <label for="userGender">Genre :</label><br>
    <input type="radio" id="userGender" name="userGender" checked value="masculin"> Homme<
    <input type="radio" id="userGender" name="userGender" value="feminin"> Femme<br>
    <br>
    <input type="submit" value="Envoyer">
</form>
```

Résultat: Exemple de champ de type radio

14.13 L'attribut type="select"

La valeur "select" de l'attribut type indique que la balise input est une liste déroulante. C'est-à-dire une liste déroulante qui permet de sélectionner une valeur parmi plusieurs valeurs.

On utilise la balise `<option>` pour définir les valeurs de la liste déroulante. On utilise l'attribut `selected` pour sélectionner une valeur par défaut.

On peut avoir une sélection multiple en ajoutant l'attribut `multiple` à la balise `<select>`. Ca permet en appuyant sur la touche Ctrl de sélectionner plusieurs valeurs.

Voici un exemple montrant comment créer une liste déroulante avec les valeurs "Belgique", "France", "Luxembourg" et "Suisse". La valeur "Belgique" est sélectionnée par défaut.

```
<form action="http://zamboyle.synology.me:2727/forms/demos/demo14-13.php" method="POST">
    <label for="userFirstname">Prénom :</label><br>
    <input type="text" id="userFirstname" name="userFirstname" aria-required="true" requi
    <br>
    <label for="userCountry">Choix du pays:</label><br>
    <select id="userCountry" name="userCountry">
        <option value="Belgique" selected>Belgique</option>
        <option value="France">France</option>
        <option value="Luxembourg">Luxembourg</option>
        <option value="Suisse">Suisse</option>
    </select><br>
    <br>
    <input type="submit" value="Envoyer">
</form>
```

Résultat: Exemple de champ de type select

19. Un meta pour le cache

Il est possible de spécifier le temps de mise en cache d'une page web. Pour cela, on utilise la balise `meta` avec l'attribut `http-equiv` et la valeur `Cache-Control`. On utilise aussi l'attribut `content` pour spécifier le temps de mise en cache. Par exemple, `max-age=3600` signifie que la page web doit être mise en cache pendant 3600 secondes.

Si on ne veut pas que la page soit mise en cache, on utilise la valeur `no-cache` pour l'attribut `content`.

> Exemples :
>
> ```
> <meta http-equiv="Cache-Control" content="max-age=3600" />
> <meta http-equiv="Cache-Control" content="no-cache" />
> ```

Vous n'êtes pas obligé d'utiliser cette balise. Ca dépend de vos besoins.

20. Le sitemap et le robots.txt

20.1 Sitemap HTML

Il existe aussi un sitemap au format HTML. Il est généralement nommé sitemap.html. Il est placé à la racine du site web. C'est-à-dire qu'il est placé dans le même dossier que la page d'accueil du site web. Il est généralement placé dans le pied de page du site web. Il permet aux utilisateurs de trouver facilement les pages d'un site web.

Pour l'accessibilité, il est recommandé de créer un sitemap HTML. En effet, cela permet aux utilisateurs de trouver facilement les pages d'un site web. Cela permet aussi aux moteurs de recherche de découvrir et d'indexer les pages d'un site web.

Voici un exemple de sitemap HTML :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>Sitemap</title>
  </head>
  <body>
    <nav>
      <ul>
        <li><a href="https://www.example.com/">Accueil</a></li>
        <li><a href="https://www.example.com/page1.html">Page 1</a></li>
        <li><a href="https://www.example.com/page2.html">Page 2</a></li>
      </ul>
    </nav>
  </body>
</html>
```

Vous pouvez également structurer le sitemap HTML de manière à refléter la structure de votre site web, en imbriquant des listes pour représenter la hiérarchie des pages et des sous-pages. Voici un exemple d'une structure plus complexe :

```html
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>Sitemap</title>
  </head>
  <body>
    <nav>
      <ul>
        <li><a href="https://www.example.com/">Accueil</a></li>
        <li>
          <a href="https://www.example.com/page1.html">Page 1</a>
          <ul>
            <li>
              <a href="https://www.example.com/subpage1.html">Sous-Page 1</a>
            </li>
            <li>
              <a href="https://www.example.com/subpage2.html">Sous-Page 2</a>
            </li>
          </ul>
        </li>
        <li><a href="https://www.example.com/page2.html">Page 2</a></li>
      </ul>
    </nav>
  </body>
</html>
```

Résultat:

- Accueil
- Page 1
  - Sous-Page 1
  - Sous-Page 2
- Page 2

20.2 Le sitemap XML

Le sitemap est un fichier qui contient la liste des pages d'un site web. Il permet aux moteurs de recherche de découvrir et d'indexer les pages d'un site web. Il permet aussi de donner des

informations sur les pages d'un site web comme la date de dernière modification d'une page, la fréquence de mise à jour d'une page, etc.

Le sitemap est un fichier au format XML. Il est généralement nommé sitemap.xml. Il est placé à la racine du site web. C'est-à-dire qu'il est placé dans le même dossier que la page d'accueil du site web.

Voici un exemple de sitemap :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
    <url>
        <loc>https://www.example.com/</loc>
        <lastmod>2020-01-01</lastmod>
        <changefreq>monthly</changefreq>
        <priority>1.0</priority>
    </url>
    <url>
        <loc>https://www.example.com/page1.html</loc>
        <lastmod>2020-01-01</lastmod>
        <changefreq>monthly</changefreq>
        <priority>0.8</priority>
    </url>
    <url>
        <loc>https://www.example.com/page2.html</loc>
        <lastmod>2020-01-01</lastmod>
        <changefreq>monthly</changefreq>
        <priority>0.8</priority>
    </url>
</urlset>
```

Dans le code ci-dessus, nous avons créé un sitemap qui contient trois pages. La première page est la page d'accueil du site web. Les deux autres pages sont des pages du site web. Chaque page est définie par une balise `url` qui contient les informations de la page. La balise `loc` contient l'URL de la page. La balise `lastmod` contient la date de dernière modification de la page. La balise `changefreq` contient la fréquence de mise à jour de la page. La balise `priority` contient la priorité de la page.

Généralement, ce genre de site web est généré automatiquement par un CMS (Content

Management System) comme WordPress. Il existe aussi des outils qui permettent de générer automatiquement un sitemap. Par exemple, vous pouvez utiliser l'outil suivant : XML Sitemap Generator.

20.3 Le fichier robots.txt

Le fichier robots.txt est un fichier qui permet de donner des instructions aux robots des moteurs de recherche. Il est généralement nommé robots.txt. Il est placé à la racine du site web. C'est-à-dire qu'il est placé dans le même dossier que la page d'accueil du site web.

Voici un exemple de fichier robots.txt :

```
User-agent: *
Disallow: /admin/
Disallow: /private/
```

Dans le code ci-dessus, nous avons créé un fichier robots.txt qui contient deux instructions. La première instruction indique que tous les robots sont autorisés à accéder à toutes les pages du site web. La deuxième instruction indique que les robots ne sont pas autorisés à accéder aux pages qui se trouvent dans les dossiers `admin` et `private`.

ou encore:

```
User-agent: *
Disallow: /
```

Dans le code ci-dessus, nous avons créé un fichier robots.txt qui contient une instruction. Cette instruction indique que tous les robots ne sont pas autorisés à accéder à toutes les pages du site web.

voici toutes les instructions possibles (**à ne pas retenir**) :

- `User-agent` : permet de spécifier le nom du robot. L'astérisque `*` signifie que l'instruction s'applique à tous les robots.
- `Disallow` : permet de spécifier les pages auxquelles les robots n'ont pas accès. L'astérisque `*` signifie que l'instruction s'applique à toutes les pages. Il est possible d'utiliser des expressions régulières pour spécifier les pages. Par exemple, `/admin/`

signifie que l'instruction s'applique à toutes les pages qui se trouvent dans le dossier `admin`. Il est possible d'utiliser plusieurs instructions `Disallow` pour spécifier plusieurs pages.

- `Allow` : permet de spécifier les pages auxquelles les robots ont accès. L'astérisque `*` signifie que l'instruction s'applique à toutes les pages. Il est possible d'utiliser des expressions régulières pour spécifier les pages. Par exemple, `/admin/` signifie que l'instruction s'applique à toutes les pages qui se trouvent dans le dossier `admin`. Il est possible d'utiliser plusieurs instructions `Allow` pour spécifier plusieurs pages.
- `Sitemap` : permet de spécifier l'URL du sitemap du site web. Il est possible d'utiliser plusieurs instructions `Sitemap` pour spécifier plusieurs sitemaps.
- `Crawl-delay` : permet de spécifier le délai entre deux requêtes du robot. Par exemple, `Crawl-delay: 10` signifie que le robot doit attendre 10 secondes entre deux requêtes. Il est possible d'utiliser plusieurs instructions `Crawl-delay` pour spécifier plusieurs délais.
- `Host` : permet de spécifier l'URL du site web. Il est possible d'utiliser plusieurs instructions `Host` pour spécifier plusieurs URL.

⬅ Revenir au sommaire.
