Les liens, les images et les attributs

Certaines balises ne portent pas leur information entre les chevrons, mais dedans : c'est le rôle des attributs. Deux d'entre elles font tout le web — le lien et l'image.

4TTR | Découverte

Jusqu'ici, tout ce que tes balises avaient à dire tenait entre l'ouvrante et la fermante. Mais comment indiquer **où** mène un lien ? Le texte cliquable est déjà pris par le contenu. Il faut un autre emplacement : l'attribut.

🎯 Objectifs

À la fin de ce cours, tu seras capable de :

1. Reconnaître un attribut et l'écrire correctement.
2. Créer un lien vers un autre site ou vers une autre page.
3. Afficher une image et rédiger un texte alternatif utile.
4. Laisser un commentaire dans ton code.

L'attribut : une information dans la balise

Un **attribut** est une information écrite **dans la balise ouvrante**, sous la forme `nom="valeur"`.

```
<a href="https://safeonweb.be">consulter Safeonweb</a>
```

Décomposons :

- `a` est le nom de la balise ;
- `href` est le nom de l'attribut ;
- `"https://safeonweb.be"` est sa valeur, entre guillemets ;
- `consulter Safeonweb` reste le contenu, entre les deux balises.

Une balise peut porter **plusieurs attributs**, séparés par une espace :

```
<img src="images/cadenas.svg" alt="Un cadenas fermé">
```

Trois règles à ne jamais oublier :

- l'attribut se met dans la balise **ouvrante**, jamais dans la fermante ;
- la valeur va entre **guillemets** ;
- pas d'espace autour du signe `=`.

Le lien

`<a>` — pour GB ***anchor***, ancre — est la balise qui a donné son nom au web. Sans elle, les pages resteraient des îles.

```
<a href="https://safeonweb.be">consulter Safeonweb</a>
```

L'attribut `href` porte la destination. Le contenu porte le texte sur lequel on clique.

Trois destinations possibles

Écriture	Où ça mène
`href="https://safeonweb.be"`	Un autre site — l'adresse complète
`href="contact.html"`	Une autre page de **ton** site, dans le même dossier
`href="#materiel"`	Un endroit précis **de la page en cours**

La deuxième forme s'appelle un **chemin relatif** : `contact.html` signifie « le fichier de ce nom, à côté de celui-ci ». C'est ainsi qu'on relie les pages d'un site entre elles. Déplace un des deux fichiers, et le lien casse.

> ⚠ **ATTENTION** : sans `href`, la balise `<a>` ne mène nulle part et n'est même plus cliquable. L'attribut n'est pas décoratif : il porte l'essentiel.

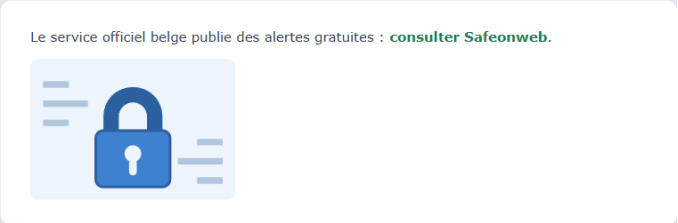
Soigne aussi le texte cliquable. « Cliquez ici » ne dit rien à qui parcourt la page en diagonale, ni à un lecteur d'écran qui liste les liens d'une page. Écris plutôt ce vers quoi on va : « consulter Safeonweb ».

L'image

`<img>` est une balise **orpheline** : elle n'encadre rien, donc pas de fermeture. Toute son information tient dans ses attributs.

```
<img src="images/cadenas.svg" alt="Un cadenas fermé">
```

- `src` — GB ***source*** — est le chemin du fichier image. Ici, le fichier `cadenas.svg` rangé dans un dossier `images` voisin de la page.
- `alt` est une description en mots de ce que montre l'image.



`alt` n'est pas optionnel

C'est le point que les débutants sautent, et c'est le plus important des deux.

Le texte alternatif sert dans trois situations :

- l'image ne se charge pas — connexion lente, fichier déplacé — et `alt` s'affiche à sa place ;
- une personne aveugle consulte la page, et son lecteur d'écran lit `alt` à voix haute ;
- un moteur de recherche indexe ta page : il ne « voit » pas l'image, il lit `alt`.

Un bon `alt` décrit ce que l'image **montre**, pas le nom du fichier :

❌	✅
`alt="image"`	`alt="Un cadenas fermé"`
`alt="cadenas.svg"`	`alt="Le Colisée vu depuis le sud"`
`alt="photo1"`	`alt="Capture d'écran du menu Encodage de VSCode"`

> 💡 **ASTUCE :** pour trouver le bon `alt`, imagine que tu décris la page à quelqu'un au téléphone. Ce que tu dirais de l'image, c'est le `alt`.

Les commentaires

Un **commentaire** s'écrit entre `<!--` et `-->`. Le navigateur l'ignore complètement : il ne s'affiche jamais dans la page.

```
<!-- la liste des visites de Rome -->
<ul>
  <li>Le Colisée</li>
</ul>
```

Deux usages :

- **expliquer** une partie du code à celui qui le lira — ton professeur, un camarade, ou toi dans trois semaines ;
- **désactiver** temporairement un morceau de code sans l'effacer, le temps de tester autre chose.

```
<!-- <img src="images/photo.jpg" alt="Une photo"> -->
```

L'image ci-dessus ne s'affiche plus, mais le code est toujours là, prêt à être réactivé.

> ℹ️ Un commentaire est invisible **DANS LA PAGE**, pas dans le code source. N'y écris jamais un mot de passe ou une remarque que tu ne voudrais pas voir lue : n'importe quel visiteur peut afficher le code source.

🧪 Vérifications

1. Dans `<a href="contact.html">Nous écrire</a>`, quel est l'attribut, et quel est le contenu ?
2. Pourquoi `<img>` n'a-t-elle pas de balise fermante ?
3. Cite deux situations où le texte du `alt` est réellement lu par quelqu'un.
4. Comment mettre temporairement une image de côté sans supprimer sa ligne ?

✍️ À retenir

- Un **attribut** s'écrit `nom="valeur"` dans la balise **ouvrante**.
- `<a href="…">` crée un lien ; sans `href`, il ne mène nulle part.
- Un **chemin relatif** comme `contact.html` désigne un fichier voisin.
- `<img src="…" alt="…">` est orpheline ; `alt` décrit l'image pour ceux qui ne la voient pas.
- `<!-- … -->` n'apparaît jamais dans la page, mais reste visible dans le code source.

Suite

Ta page dit maintenant tout ce qu'elle a à dire. Elle reste noire sur blanc, dans la police par défaut du navigateur. C'est le moment du CSS.