

HTML 5

Rappels, accessibilité et quelques nouveautés



Introduction

Ce chapitre destiné permet un retour sur certains points du HTML ainsi que la présentation de quelques nouveautés du langage HTML.

Ceci n'est pas une référence complète. Chaque section présente de brèves explications et des exemples, ce qui vous permettra d'approfondir le sujet.

Bien qu'il ne s'agisse pas d'un cours d'accessibilité, nous aborderons les bonnes pratiques et des problèmes spécifiques en la matière.

Structurer une page



Installation

Commencez par créer un nouveau projet VS Code avec un nouveau fichier appelé **index.html** dans le dossier **html** avec le balisage suivant.



*Attention penser à créer votre dossier en tant que descendant du dossier **SCAP** pour pouvoir le commiter sur votre espace Github.*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Les HTML !</title>
  </head>
  <body>
    </body>
</html>
```

Sémantique et accessibilité

Avec plus de 100 éléments HTML et la possibilité de créer des éléments personnalisés, il existe une infinité de façons de baliser votre contenu.

Certaines méthodes, notamment sémantiquement, sont plus efficaces que d'autres.

Sémantique signifie "lié au sens". Écrire du code HTML sémantique signifie utiliser des éléments HTML pour structurer votre contenu en fonction de la signification de chaque élément, et non de son apparence.

Choisir les bons éléments pour le code signifie que vous n'aurez pas à factoriser ou à commenter votre code HTML ultérieurement ! **Keep it Simple !**

Du HTML accessible, mais encore ?

HTML est accessible ! Mais dans certains cas spécialement pour certaines interactions (js) le niveau d'accessibilité ne peut être atteint aussi simplement,

Pour résoudre ces problèmes, il existe “**Accessible Rich Internet Applications**” (**ARIA**) (qu'on peut traduire par « applications internet riches et accessibles ») qui est un ensemble de rôles et d'attributs qui définissent comment rendre le contenu et les applications web accessibles (notamment celles développées avec JavaScript) pour les personnes ayant des handicaps.

Accessibilité

La première règle **d'ARIA** s'énonce ainsi : « Si vous pouvez utiliser un élément natif HTML ou un attribut avec la sémantique et le comportement voulu qui existe nativement, faites-le plutôt que d'utiliser un autre élément en lui ajoutant un rôle, un état ou une propriété ARIA afin de le rendre accessible. ».



Note : On peut parfois lire l'expression « Mieux vaut ne pas utiliser ARIA que de l'utiliser incorrectement ». Lors d'un sondage WebAIM sur plus d'un million de pages d'accueil de sites, il a été observé que les pages utilisant ARIA avaient 41% d'erreurs supplémentaires détectées par rapport aux pages sans ARIA. Bien qu'ARIA soit conçu pour rendre les pages web plus accessibles, lorsqu'il est utilisé incorrectement, il fait plus de mal que de bien.

- Règle 1: Ne pas utiliser ARIA
- Règle 2: N'ajoutez pas (inutilement) d'attributs ARIA au code HTML
- Règle 3: Toujours permettre la navigation au clavier
- Règle 4: Ne pas masquer les éléments sélectionnables

Header

Créons un en-tête de site. Vous commencerez par utiliser le balisage non sémantique, puis vous trouverez une solution adaptée afin de découvrir les avantages de la section HTML et des éléments de titre.

Si vous accordez peu ou pas de considération à la sémantique de notre en-tête, vous pouvez utiliser du code comme celui-ci. L'utilisation des `<div>` non sémantiques va vous créer du travail supplémentaire. Pour cibler plusieurs `<div>`s avec CSS, vous finissez par utiliser des ID ou des classes pour identifier le contenu. Le code inclut également un commentaire pour chaque `</div>` de fermeture afin d'indiquer quelle balise d'ouverture chaque `</div>` a fermé.



Vous pouvez inclure des attributs "role" pour fournir la sémantique permettant de créer un bon modèle d'objet d'accessibilité (AOM) pour les lecteurs d'écran. mais nous verrons cela plus tard...

```
<!-- header -->
<div id="pageHeader">
  <div id="title">L'école du code</div>
  <!-- navigation -->
  <div id="navigation">
    <a href="#reg">S'enregistrer</a>
    <a href="#about">À propos</a>
    <a href="#teachers">Professeurs</a>
    <a href="#feedback">Témoignages</a>
  </div>
  <!-- end navigation -->
</div>
<!-- end header -->
```

Header

Ce code utilise deux éléments sémantiques `<header>` et `<nav>`.

Il s'agit de l'en-tête principal.

Lorsque `<header>` est de niveau supérieur, il s'agit de la bannière du site, Par contre lorsqu'un élément `<header>` est imbriqué dans `<main>`, `<article>` ou `<section>`, il l'identifie simplement comme en-tête de cette section et n'est pas un point de repère.

L'élément `<nav>` identifie le contenu en tant que navigation. Comme cet élément `<nav>` est imbriqué dans l'en-tête du site, il s'agit de la navigation principale du site.

Si elle était imbriquée dans un `<article>` ou un `<section>`, il s'agirait de navigation interne pour cette section uniquement. À l'aide d'éléments sémantiques, vous avez créé un modèle d'objet d'accessibilité pertinent.

```
<header>
  <h1>L'école du code</h1>
  <nav>
    <a href="#reg">S'enregistrer</a>
    <a href="#about">À propos</a>
    <a href="#teachers">Professeurs</a>
    <a href="#feedback">Témoignages</a>
  </nav>
</header>
```


Footer

Comme pour l'élément **<header>**, le fait que le pied de page **<footer>** soit ou non un “point de repère” dépend de son emplacement d'imbrication.

Le pied de page du site doit contenir les informations souhaitées sur chaque page, telles que la déclaration sur les droits d'auteur, des coordonnées, ainsi que des liens vers vos politiques de confidentialité et de cookies.

Lorsqu'un élément **<footer>** est un descendant d'un élément **<article>**, **<aside>**, **<main>**, **<nav>** ou **<section>**, il ne s'agit pas d'un point de repère. Si le message apparaît seul, selon le balisage, ce pied de page peut être mis en avant

Nous avons commencé par les éléments **<header>** et **<footer>**, étudions à présent les éléments de section en examinant les mises en page les plus courantes:

```
<footer>  
  <p>&copy; Mon Site</h1>  
</footer>
```

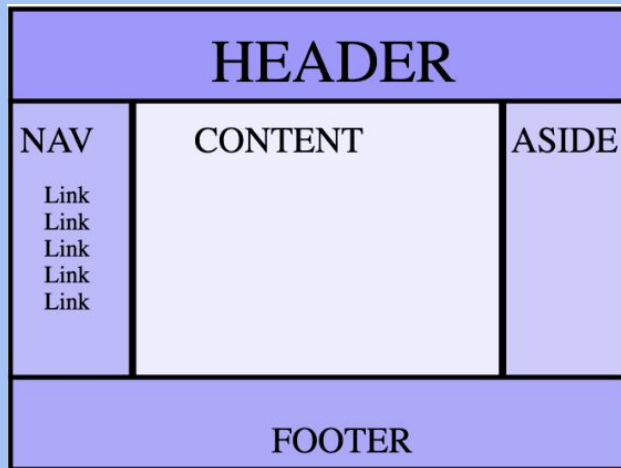
Structure du document

Une mise en page composée d'un en-tête, de deux barres latérales et d'un pied de page est appelée mise en page du “Saint Graal”, “*holly grail*” layout en VO.

Il existe de nombreuses façons de baliser ce contenu, notamment :

```
<body>
<header>Header</header>
<nav>Nav</nav>
<main>Main</main>
<aside>Aside</aside>
<footer>Footer</footer>
</body>
```

Si vous créez un blog, vous trouverez peut-être une série d'articles dans
<main>.



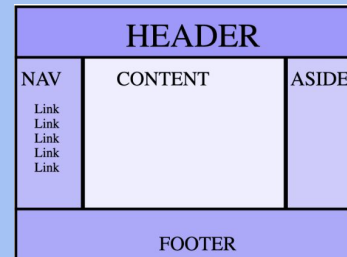
```
<body>
<header>Header</header>
<nav>Nav</nav>
<main>
  <article>Article A</article>
  <article>Article B</article>
</main>
<aside>Aside</aside>
<footer>Footer</footer>
</body>
```

Structure du document

<main>

L'élément <main> identifie le contenu principal du document. Il ne doit y avoir qu'un seul

<main> par page.



<aside>

Le <aside> désigne le contenu indirectement ou tangentiellement lié au contenu principal du document.

<article>

Nous avons ajouté deux éléments <article> au niveau de <main>.

Ce n'était pas nécessaire dans le premier exemple où le contenu principal ne comportait qu'un seul mot, ou dans le monde réel, une seule section de contenu. Toutefois, si vous écrivez un blog, comme dans notre deuxième exemple, chaque article doit se trouver dans un élément <article> imbriqué dans <main>.

Une <article> représente une section de contenu complète ou autonome qui est, en principe, réutilisable de manière indépendante. Pensez à un article comme à un article de journal.

Structure du document

| HEADER | | |
|-------------------------------------|---------|-------|
| NAV Link Link Link Link | CONTENT | ASIDE |
| FOOTER | | |

<section>

L'élément **<section>** permet d'englober les sections génériques d'un document lorsqu'il n'y a plus d'élément sémantique spécifique à utiliser. À quelques exceptions près, les sections doivent avoir un titre.

Pour reprendre l'exemple du journal, sur la page d'accueil du journal, la bannière inclurait le nom du journal, suivi d'une seule <main>, divisée en plusieurs éléments <section>, chacun avec un en-tête, comme "Actualités" et "Politique". Chaque section comporte une série de <article>. Dans chaque <article>, vous pouvez également trouver un ou plusieurs éléments <section>. Si vous consultez cette page, l'intégralité de la partie "en-têtes et sections" correspond à l'<article>. Ce <article> est ensuite divisé en plusieurs <section>, y compris site header, site footer et la structure du document. L'article lui-même a un en-tête, tout comme chacune des sections.

<h1>-<h6>

Il y a six éléments de titre de section: **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>** et **<h6>**. Chacun représente l'un des six niveaux de titre de section, **<h1>** étant le niveau de section le plus élevé ou le plus important, et **<h6>** le niveau le plus bas. Lorsqu'un titre est imbriqué dans une bannière de document <header>, il s'agit de l'en-tête de l'application ou du site. Lorsqu'il est imbriqué dans **<main>**, qu'il soit imbriqué ou non dans un **<header>** de **<main>**, il s'agit de l'en-tête de cette page, et non de l'ensemble du site.

Lorsqu'elle est imbriquée dans une **<article>** ou une **<section>**, elle correspond à l'en-tête de cette sous-section de la page.

Accessibilité

Points de repère - landmark-

Les utilisateurs d'outils d'assistance technologiques -AT- s'appuient sur des éléments structurels pour leur donner des informations sur la mise en page globale de la page. Lorsque vous coupez de vastes zones de contenu, vous pouvez utiliser des rôles de repère ARIA ou les nouveaux éléments de repère HTML pour ajouter du contexte structurel à votre page.

Ils garantissent que le contenu se trouve dans des régions navigables. Nous vous recommandons de fournir au moins un point de repère par page.

Certaines ressources suggèrent de combiner des points de repère ARIA et HTML pour fournir une meilleure couverture d'AT. Bien que ce type de redondance ne pose pas de problèmes pour vos utilisateurs, testez les schémas à l'aide d'un lecteur d'écran pour vous en assurer. En cas de doute, il est préférable de n'utiliser par défaut que les éléments HTML les plus récents, car la compatibilité avec les navigateurs est particulièrement élevée.

Accessibilité

Comparons les éléments de repère HTML mappés avec leurs rôles de repère ARIA équivalents.

| Tag HTML | Landmark ARIA |
|-------------------------------------|----------------------|
| <header> | banner |
| <aside> | complementary |
| <footer> | contentinfo |
| <nav> | navigation |
| <main> | principal |
| <form> ¹ | form |
| <section> ¹ | region |

1 Nécessite l'inclusion de l'attribut name pour être accessible. Une section doit être nommée de manière accessible en raison de son rôle ARIA implicite (region) afin d'être visible par la technologie d'assistance.

Comparons maintenant des exemples de structure de contenu accessible.

Mais ajoutons le code suivant à notre page

```
<body>
  <header>
    <h1>L'école du code</h1>
    <nav>
      <a href="#reg">S'enregistrer</a>
      <a href="#about">À propos</a>
      <a href="#teachers">Professeurs</a>
      <a href="#feedback">Témoignages</a>
    </nav>
  </header>
  <main>
    <section id="reg"><h2>S'enregistrer</h2></section>
    <section id="about"><h2>A propos</h2></section>
    <section id="teachers"><h2>Professeurs</h2></section>
    <section id="feedback"><h2>Témoignages</h2></section>
  </main>
<footer>L'école du code depuis 2024 &copy;</footer>
</body>
```



N.B. : Étant donné qu'aucun contenu n'est autonome et complet, `<section>` est plus approprié que `<article>`.

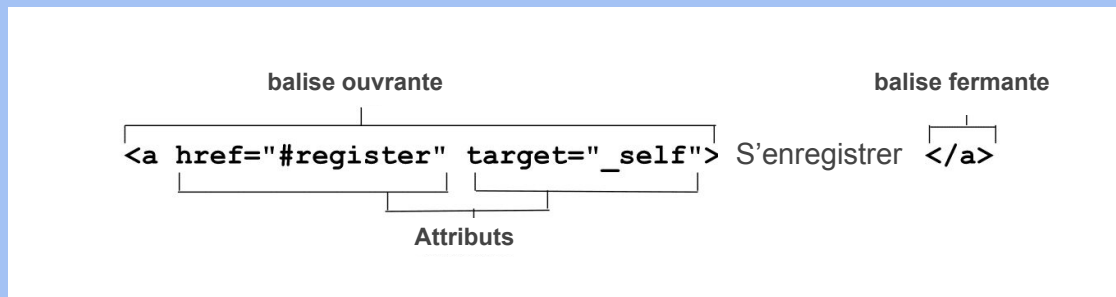
Bien que chacune des sections ait un titre, aucune section ne mérite de `<footer>`.

Les principaux attributs



Les attributs du HTML

Ce sont les attributs qui font du HTML si puissant. Les attributs sont des noms séparés par un espace et des paires nom/valeur apparaissant dans la balise d'ouverture. Ils fournissent des informations sur l'élément et ses fonctionnalités.



Les attributs définissent le comportement, les liaisons et la fonctionnalité des éléments. Certains attributs sont globaux, ce qui signifie qu'ils peuvent apparaître dans la balise d'ouverture de n'importe quel élément.

D'autres s'appliquent à plusieurs éléments, mais pas à tous, tandis que d'autres sont spécifiques à un élément et ne concernent qu'un seul élément.

En HTML, tous les attributs à l'exception des attributs booléens et, dans une certaine mesure, des attributs énumérés, nécessitent une valeur.

Les attributs du HTML

Si une valeur d'attribut comprend un espace ou des caractères spéciaux, la valeur doit être placée entre guillemets. C'est pourquoi, et pour améliorer la lisibilité, il est toujours recommandé d'utiliser des doubles quotes..

Bien que le code HTML ne soit pas sensible à la casse, certaines valeurs d'attributs le sont. Les valeurs qui font partie de la spécification HTML ne sont pas sensibles à la casse. Les valeurs de chaînes définies, telles que les noms de classe et d'identifiant, sont sensibles à la casse.

Si la valeur d'un attribut est sensible à la casse en HTML, elle l'est lorsqu'elle est utilisée dans un sélecteur d'attribut en CSS et en JavaScript. Sinon, elle ne l'est pas.

```
<!-- l'attribut type est insensible à la casse : ils sont équivalents -->
<input type="text">
<input type="TeXt">

<!-- l'attribut id est sensible à la casse : ils ne sont pas équivalents-->
<div id="myId">...</div>
<div id="MyID">...</div>
```

Les attributs booléens

Si un attribut booléen est présent, la valeur est toujours "true". Les attributs booléens incluent **autofocus**, **inert**, **checked**, **disabled**, **required**, **reversed**, **allowfullscreen**, **default**, **loop**, **autoplay**, **controls**, **muted**, **readonly**, **multiple**, et **selected**. Les valeurs booléennes peuvent être omises, définies sur une chaîne vide ou correspondre au nom de l'attribut. Toutefois, il n'est pas nécessaire que la valeur soit définie à "true". Toutes les valeurs, y compris true, false et 😊, bien qu'elles ne soient pas valides, seront remplacées par "true".

Ces trois tags sont équivalents:

```
<input required>  
<input required="">  
<input required="required">
```

Si la valeur de l'attribut est "false", omettez l'attribut. Si l'attribut est vrai, incluez-le mais n'indiquez aucune valeur. Par exemple, `required="required"` n'est pas une valeur valide en HTML. Cependant, comme `required` est une valeur booléenne, les valeurs non valides sont associées à la valeur "true".

Toutefois, étant donné que les attributs énumérés non valides ne renvoient pas nécessairement la même valeur que les valeurs "manquantes", il est plus facile de prendre l'habitude d'omettre des valeurs que de se rappeler quels attributs sont booléens ou énumérés et de fournir potentiellement une valeur non valide.

Les attributs énumérés

Les attributs énumérés sont parfois confondus avec les attributs booléens. Ce sont des attributs HTML qui ont un ensemble limité de valeurs valides prédéfinies. Comme les attributs booléens, ils ont une valeur par défaut si l'attribut est présent, mais que la valeur est manquante.

Par exemple, si vous utilisez l'élément **<input>**, la valeur par défaut est **<input type="text">**.

Cependant, contrairement aux attributs booléens, l'omission de l'attribut ne signifie pas qu'il est faux.

Un attribut présent avec une valeur manquante n'est pas nécessairement vrai. De plus, la valeur par défaut pour les valeurs non valides n'est pas obligatoirement identique à une chaîne nulle.

Dans la plupart des cas, avec des attributs énumérés, les valeurs manquantes et non valides sont les mêmes. Par exemple, si l'attribut **type** d'un élément **<input>** est manquant, présent mais sans valeur, ou s'il a une valeur non valide, il est défini par défaut sur **text**.

Bien que ce comportement soit courant, il ne s'agit pas d'une règle. Pour cette raison, il est important de savoir quels attributs sont booléens ou énumérés. Si possible, omettez des valeurs pour ne pas vous tromper et recherchez les valeurs si nécessaire.

Les attributs globaux

Les attributs globaux peuvent être définis pour n'importe quel élément HTML, y compris les éléments <head>. **Il existe plus de 30 attributs globaux.** Bien que tous ces éléments puissent en théorie être ajoutés à n'importe quel élément HTML, certains attributs globaux n'ont aucun effet lorsqu'ils sont définis sur certains éléments ; par exemple, définir hidden sur un **<meta>**, car le métacontenu n'est pas affiché.

L'attribut global **id** permet de définir un identifiant unique pour un élément. Il remplit de nombreuses fonctions, y compris :

- Cible de l'identifiant de fragment d'un lien.
- Identifier un élément pour l'écriture de script.
- Associer un élément de formulaire à son libellé.
- Fournir un libellé ou une description pour les technologies d'assistance.
- Ciblage de styles avec une forte spécificité ou en tant que sélecteurs d'attributs dans CSS.

La valeur id est une chaîne sans espaces. Tous les autres caractères sont valides.

La valeur **id** peut être 😊 ou **.class**, mais ce n'est pas une bonne idée. Pour faciliter la programmation, actuelle et future, faites du premier caractère de l'id une lettre et n'utilisez que des caractères ASCII, des chiffres, ainsi que des caractères _ et -.

Il est recommandé de définir une convention d'attribution de noms id et de s'y tenir, car les valeurs id sont sensibles à la casse.

Les attributs globaux : id

L'id doit être unique au document. La mise en page de votre page ne sera probablement pas affectée si vous utilisez l'id plusieurs fois, mais vos interactions JavaScript, les liens et les éléments risquent de ne pas se comporter comme prévu.

Identifiant de fragment de lien

La barre de navigation comprend quatre liens. Les liens ne sont pas limités aux URL HTTP. Il peut s'agir d'identifiants de fragments de sections de la page dans le document actuel (ou dans d'autres documents).

Dans notre exemple, la barre de navigation dans l'en-tête de page comprend quatre liens (voir le code ci contre).

L'attribut **"href"** fournit le lien hypertexte vers lequel l'utilisateur est redirigé en activant le lien. Lorsqu'une URL comprend un signe de hachage (#) suivi d'une chaîne de caractères, cette chaîne est un identifiant de fragment.

Si cette chaîne correspond à l'id d'un élément de la page Web, le fragment est l'ancre de cet élément. Le navigateur fait défiler la page jusqu'au point où l'ancre est définie.

```
<nav>
  <a href="#reg">S'enregistrer</a>
  <a href="#about">À propos</a>
  <a href="#teachers">Professeurs</a>
  <a href="#feedback">Témoignages</a>
</nav>
```

Les attributs globaux : id

Ces quatre liens pointent vers quatre sections de notre page identifiées par leur attribut id. Lorsque l'utilisateur clique sur l'un des quatre liens de la barre de navigation, l'élément lié par l'identifiant de fragment, c'est-à-dire l'élément contenant l'**ID** correspondant moins l'élément **#**, défile.

Le contenu **<main>** de notre exemple comporte quatre sections avec des identifiants. Lorsque le visiteur du site clique sur l'un des liens du **<nav>**, la section associée à cet identifiant de fragment défile. Le balisage est semblable à celui-ci:

```
<section id="reg"><h2>S'enregistrer</h2></section>
<section id="about"><h2>A propos</h2></section>
<section id="teachers"><h2>Professeurs</h2></section>
<section id="feedback"><h2>Témoignages</h2></section>
```

En comparant les identifiants de fragment dans les liens **<nav>**, vous remarquerez que chacun correspond au id d'une **<section>** dans **<main>**. Le navigateur affiche un lien sans frais en haut de page.

Le séparateur de marque de hachage dans href ne fait pas partie de l'identifiant de fragment. L'identifiant de fragment est toujours la dernière partie de l'URL. Il n'est pas envoyé au serveur.

En CSS, vous pouvez cibler chaque section à l'aide d'un sélecteur d'ID tel que **#feedback** ou, pour moins de précision, d'un sélecteur d'attributs sensible à la casse, **[id="feedback"]**.

Les attributs globaux : class

Sélecteur de classe

L'attribut **class** offre un moyen supplémentaire de cibler des éléments avec CSS (et JavaScript), mais ne remplit aucune autre fonction en HTML (bien que les frameworks et les bibliothèques de composants puissent les utiliser). La valeur de l'attribut de classe est une liste des classes sensibles à la casse de l'élément, séparées par des espaces.

La construction d'une structure sémantique permet de cibler les éléments en fonction de leur emplacement et de leur fonction. La structure permet d'utiliser des sélecteurs d'éléments descendants, des sélecteurs relationnels et des sélecteurs d'attributs.

En examinant les attributs tout au long de cette section, réfléchissez à la manière dont les éléments ayant les mêmes attributs ou valeurs d'attribut peuvent être stylisés. Vous ne devez pas toujours utiliser l'attribut "**class**", mais la plupart des développeurs n'ont pas conscience qu'ils n'ont souvent pas besoin de le faire.

Jusqu'à présent, notre exemple n'a utilisé aucune classe.

Un site peut-il être lancé sans un seul nom de classe ? Nous verrons...

Les attributs globaux : style

L'attribut **style** permet d'appliquer des styles intégrés, c'est-à-dire des styles appliqués à l'élément unique sur lequel l'attribut est défini. L'attribut style utilise comme paire valeur/clé de propriété CSS, la syntaxe de la valeur étant identique au contenu d'un bloc de style CSS: les propriétés sont suivies du signe deux-points, comme dans CSS, et des points-virgules terminent chaque déclaration, après la valeur.

Les styles ne sont appliqués qu'à l'élément sur lequel l'attribut est défini. Les descendants héritent des valeurs de propriété héritées s'ils ne sont pas remplacés par d'autres déclarations de style sur des éléments imbriqués ou dans des feuilles de style ou des blocs <style>.

Bien que style soit en effet un attribut global, son utilisation n'est pas recommandée. Définissez plutôt les styles dans un ou plusieurs fichiers distincts. Cela dit, l'attribut style peut s'avérer utile pendant le développement, car il permet d'appliquer un style rapide, par exemple à des fins de test. Prenez ensuite le style "solution" et collez-le dans votre fichier CSS associé.

Les attributs globaux : tabindex

L'attribut **tabindex** peut être ajouté à n'importe quel élément pour qu'il soit sélectionné. La valeur **tabindex** détermine si elle est ajoutée à l'ordre de tabulation et, éventuellement, dans un ordre de tabulation autre que celui par défaut.

La valeur de l'attribut tabindex correspond à un entier. Une valeur négative (la convention consiste à utiliser -1) rend un élément capable de recevoir le focus, par exemple via JavaScript, mais n'ajoute pas l'élément à la séquence de tabulation. Une valeur tabindex de 0 rend l'élément sélectionnable et accessible via la touche de tabulation, en l'ajoutant à l'ordre de tabulation par défaut de la page, dans l'ordre du code source. Une valeur supérieure ou égale à 1 place l'élément dans une séquence de focus prioritaire, ce qui n'est pas recommandé.



Modifier l'ordre de tabulation peut créer une très mauvaise expérience utilisateur. Il est difficile d'utiliser des technologies d'assistance (claviers et lecteurs d'écran) pour naviguer dans votre contenu.

En tant que développeur, il est également difficile à gérer et à entretenir.

Les attributs globaux : role

L'attribut **role** fait partie de la spécification ARIA, et non de la spécification HTML. L'attribut **role** peut être utilisé pour donner une signification sémantique au contenu, ce qui permet aux lecteurs d'écran d'informer les utilisateurs du site de l'interaction utilisateur attendue pour un objet.

Il existe quelques widgets d'interface utilisateur courants, tels que les boîtes de sélections, les barres de menu, les listes d'onglets et les arborescences, qui n'ont pas d'équivalent HTML natif.

Par exemple, lors de la création d'un modèle de conception à onglets, vous pouvez utiliser les rôles **tab**, **tablist** et **tabpanel**. Une personne qui peut voir physiquement l'interface utilisateur a appris par expérience à naviguer dans le widget et à rendre différents panneaux visibles en cliquant sur les onglets associés. L'inclusion du rôle tab avec **<button role="tab">** lorsqu'un groupe de boutons est utilisé pour afficher différents panneaux permet à l'utilisateur du lecteur d'écran de savoir que le **<button>** sélectionné peut afficher un panneau associé au lieu d'implémenter une fonctionnalité typique de bouton. L'attribut role ne modifie pas le comportement du navigateur ni les interactions avec le clavier ou les pointeurs. Ajouter **role="button"** à un **** ne le transforme pas en **<button>**. C'est pourquoi nous vous recommandons d'utiliser les éléments HTML sémantiques à bon escient. Toutefois, lorsqu'il n'est pas possible d'utiliser l'élément approprié, l'attribut role permet d'informer les utilisateurs de lecteurs d'écran lorsqu'un élément non sémantique a été adapté au rôle d'un élément sémantique.

Les attributs globaux : contenteditable

Un élément dont l'attribut **contenteditable** est défini sur **true** est modifiable, sélectionnable et ajouté à l'ordre de tabulation comme si l'élément `tabindex="0"` avait été défini. **Contenteditable** est un attribut énuméré qui prend en charge les valeurs **true** et **false**. La valeur par défaut est **inherit** si l'attribut est absent ou s'il a une valeur non valide. Ces trois balises d'ouverture sont équivalentes:

```
<style contenteditable>  
<style contenteditable="">  
<style contenteditable="true">
```

Si vous incluez **<style contenteditable="false">**, l'élément n'est pas modifiable (sauf s'il est modifiable par défaut, comme `<textarea>`). Si la valeur n'est pas valide (**<style contenteditable="😊">** ou **<style contenteditable="contenteditable">**, par exemple), la valeur par défaut est **inherit**.

Les attributs globaux peuvent être appliqués à tous les éléments, même aux éléments **<style>**.

Vous pouvez utiliser des attributs et un peu de CSS pour créer un éditeur CSS en direct.

Ajouter ce bloc dans la dernière section de votre code.

*Essayez de remplacer le **color** de style par une valeur autre que “red”.*

*Essayez ensuite de remplacer style par un sélecteur **p**.*

Ne supprimez pas la propriété display, sinon le bloc de style disparaîtra.

```
<style contenteditable>  
style {  
  color: red;  
  display: block;  
  padding: 1rem;  
}  
</style>
```

Les attributs globaux : Attributs personnalisés

Nous avons abordé uniquement les attributs globaux HTML. D'autres attributs ne s'appliquent qu'à un seul élément ou à un ensemble limité d'éléments. Même avec des centaines d'attributs définis, il se peut que vous ayez besoin d'un attribut non inclus dans la spécification. HTML est là pour vous aider.

Vous pouvez créer les attributs personnalisés de votre choix en ajoutant le préfixe **data-**.

Vous pouvez nommer votre attribut avec n'importe quel nom commençant par **data-**, suivi de toute série minuscule de caractères ne commençant pas par xml et ne contenant pas le signe deux-points (:).

Bien que le code HTML soit indulgent et ne pose pas de problème si vous créez des attributs non compatibles qui ne commencent pas par **data**, ou même si vous commencez votre attribut personnalisé par **xml** ou incluez un :, créer des attributs personnalisés valides commençant par **data-** présente des avantages.

Grâce aux attributs de données personnalisés, vous savez que vous n'utilisez pas par erreur un nom d'attribut existant. Les attributs de données personnalisées sont évolutifs.

*Bien que les navigateurs n'implémentent pas de comportements par défaut pour un attribut spécifique précédé de **data-**, il existe une API d'ensemble de données intégrée permettant d'itérer vos attributs personnalisés.*

Les propriétés personnalisées sont un excellent moyen de communiquer des informations spécifiques à une application via JavaScript. Ajoutez des attributs personnalisés aux éléments sous la forme data-name et accédez-y via le DOM à l'aide de dataset[name] sur l'élément en question. *Mais nous y reviendrons plus tard...*

Les principes de base du texte



Principes de base du texte

De la même manière que votre éditeur de texte fournit **<h1>** aux en-têtes **<h6>**, ainsi que de nombreuses façons de mettre en forme des sections de texte de manière pertinente et visuelle, le langage HTML fournit un ensemble très similaire d'éléments sémantiques et non sémantiques pour donner un sens à la prose.

Nous allons décrire les principales méthodes de balisage du texte, ou les principes de base du texte.

Les titres revisités

Il y a six éléments d'en-tête de section : **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>** et **<h6>**.

<h1> est le plus important et **<h6>** le moins. Pendant de nombreuses années, les développeurs ont appris que les en-têtes étaient utilisés par les navigateurs pour décrire les documents. C'était à l'origine un objectif, mais les navigateurs n'ont pas mis en œuvre les fonctionnalités de description. Toutefois, les utilisateurs de lecteurs d'écran utilisent les titres comme stratégie d'exploration pour en savoir plus sur le contenu de la page, à l'aide de la touche "h". Par conséquent, vous assurer que les niveaux de titre sont mis en œuvre comme vous le feriez pour décrire un document rend votre contenu accessible et reste très encouragé.

Par défaut, les navigateurs appliquent le style **<h1>** le plus grand et **<h2>** légèrement plus petit, chaque niveau de titre suivant étant plus petit par défaut. Par défaut, les navigateurs diminuent également la taille de police **<h1>** en fonction du nombre d'éléments **<article>**, **<aside>**, **<nav>** ou **<section>** dans lesquels elle est imbriquée.

Principes de base du texte

En dehors des titres, la plupart des textes structurés sont composés d'une série de paragraphes. En HTML, les paragraphes sont balisés avec la balise `<p>`. La balise de fermeture est facultative, mais elle est toujours recommandée. La section **#about** comporte un titre et quelques paragraphes:

```
<section id="about">
  <h2>A propos</h2>
  <p>Bienvenue au cours, où notre formation à l'apprentissage du HTML, du CSS et du JS. </p>
  <p>Nous enseignons les matières nécessaire à l'acquisition des compétences d'un développeur Front-End !</p>
  <p>Alors du courage et de l'abnégation afin de se préparer à un métier bousculé par l'IA.</p>
</section>
```

Cette section n'est pas un "point de repère", "**landmark**" en VO, car son nom n'est pas accessible. Pour en faire une "**region**", qui est un rôle de repère, vous pouvez utiliser **aria-labelledby** pour fournir le nom accessible:

```
<section id="about" aria-labelledby="#about_title">
  <h2 id="about_title">A propos</h2>
  ...
</section>
```



Ne créez des points de repère que si et quand c'est approprié. Avoir trop de points de repère peut rapidement être désorienté pour les utilisateurs de lecteurs d'écran.

Principes de base du texte : Citations

Lorsque vous balisez un article ou un article de blog, vous pouvez inclure une citation.

Il existe des éléments pour ces trois composants: **<blockquote>**, **<q>** et **<cite>** pour une citation visible, ou l'attribut **cite** pour fournir plus d'informations dans le cadre d'une recherche.

La section **#feedback** contient un en-tête et deux avis. Ces avis sont des citations, dont certaines contiennent des citations, suivies d'un paragraphe contenant la citation de la citation. Omettre le troisième avis pour économiser de l'espace, le balisage est:

```
<section id="feedback" class="feedback" >
  <h2>Témoignages</h2>
  <ul>
    <li>
      <blockquote>Lorem ipsum sit amet dolor.</blockquote>
      <p>Will Good</p>
    </li>
    <li>
      <blockquote>Lorem ipsum sit amet dolor.</blockquote>
      <p>John Doe <br><cite>Code en stock</cite></p>
    </li>
    <li>
      <blockquote cite="https://institutducode.com">
        Lorem ipsum sit amet dolor.
      </blockquote>
      <p>Bill Murray</p>
    </li>
  </ul>
  <p lang="fr-FR">Marcel dit : <q>Je suis désolé mais je ne
    connais pas ce langage !</q></p>
</section>
```

Principes de base du texte : Citations

Les informations sur l'auteur ou la citation ne font pas partie de la citation, et donc pas dans la balise **<blockquote>**, mais sont placées après. Bien qu'il s'agisse de citations au sens courant du terme, elles ne citent pas réellement une ressource spécifique et sont donc encapsulées dans un élément de paragraphe **<p>**.

Dans le deuxième exemple, l'utilisation de **
** crée un saut de ligne dans un bloc de texte. Il peut être utilisé dans les adresses physiques, dans la poésie et dans les blocs de signature. Les sauts de ligne ne doivent pas être utilisés comme retour chariot vers des paragraphes distincts. Au lieu de cela, fermez le paragraphe précédent et ouvrez-en un nouveau. L'utilisation de paragraphes est non seulement bonne pour l'accessibilité, mais elle permet également d'appliquer des styles. L'élément **
** n'est qu'un saut de ligne. Il est affecté par très peu de propriétés CSS.

Bien que nous ayons fourni des informations sur les citations dans un paragraphe après chaque citation, celles présentées précédemment sont codées de cette manière, car elles ne proviennent pas d'une source externe. Si c'est le cas, la source peut (devrait ?) être citée.

Principes de base du texte : Citations

Si l'avis a été extrait d'un site Web, d'un livre ou d'une autre œuvre, l'élément **<cite>** peut être utilisé comme titre d'une source. Le contenu de **<cite>** peut être le titre d'un livre, le nom d'un site Web, d'une série TV, ou même le nom d'un programme informatique. L'encapsulation **<cite>** peut être utilisée, que la source soit mentionnée de manière passe, ou qu'elle soit citée ou référencée. Le contenu de **<cite>** est l'œuvre, pas son auteur.

Afin d'indiquer que le contenu doit être mentionné lorsque vous ne pouvez pas rendre le contenu visible, l'attribut **cite** utilise l'URL du document source ou du message pour les informations citées comme valeur.

Cet attribut est valide à la fois pour **<q>** et pour **<blockquote>**. Bien qu'il s'agisse d'une URL, elle est lisible par un ordinateur, mais pas visible pour le lecteur.

Bien que la balise de fermeture **</p>** soit facultative (mais toujours recommandée), la balise de fermeture **</blockquote>** est toujours obligatoire.

La plupart des navigateurs ajoutent une marge intérieure à la fois aux directions intégrées **<blockquote>** et le contenu **<cite>** en italique. Ceci peut être contrôlé avec CSS. L'élément **<blockquote>** n'ajoute pas de guillemets, mais ceux-ci peuvent être ajoutés avec du contenu généré par CSS.

Principes de base du texte : Citations

L'élément `<q>` ajoute des guillemets par défaut en utilisant des guillemets adaptés à la langue.

L'attribut **lang** est inclus pour indiquer au navigateur que la langue de base de la page a été définie sur l'anglais dans la balise d'ouverture `<html lang="fr">`, mais que ce paragraphe est dans une autre langue. Cela permet aux commandes vocales comme Siri, Alexa et voiceOver d'utiliser la prononciation en anglais. Il indique également au navigateur le type de guillemets à afficher.

Comme `<blockquote>`, l'élément `<q>` accepte l'attribut `cite`.

```
<section id="feedback" class="feedback" >
...
  <p lang="en-EN">John says: <q>I'am sorry i do not kwow this language !</q></p>
  <p>Jean dit: <q cite="https://www.youtube.com/watch?v=NeKy63suy_w" >Je suis
désolé mais je ne      connais pas ce langage !</q></p>
</section>
```

Principes de base du texte : Entités HTML

Vous avez peut-être remarqué la séquence d'échappement, ou "entité". Comme `<` est utilisé en HTML, vous devez l'échapper à l'aide de `<` ou d'un encodage `<`; moins facile à mémoriser. Quatre entités réservées en HTML sont les suivantes: `<`, `>`, `&` et `"`. Leurs références de caractères sont respectivement **`<`**, **`>`**, **`&`** et **`"`**.

Vous utiliserez souvent les entités **`©`** pour les droits d'auteur (©), **`™`** pour les marques (TM) et **` `** pour les espaces insécables. Les espaces insécables sont utiles lorsque vous souhaitez insérer un espace entre deux caractères ou mots tout en empêchant un saut de ligne à cet endroit.

Il existe plus de 2 000 références de caractères nommées. Toutefois, si nécessaire, chaque caractère, y compris les emoji, a un équivalent encodé qui commence par **`&#`**.

Seuls les caractères non compatibles avec le jeu de caractères doivent être échappés. Si nécessaire, il existe de nombreux outils pour permettre l'échappement de différents caractères.

Vous pouvez également vous assurer d'inclure **`<meta charset="UTF-8">`** dans le **`<head>`** de la page.

Même lorsque vous spécifiez le jeu de caractères au format UTF-8, vous devez toujours échapper la `<` lorsque vous voulez imprimer ce caractère à l'écran. En règle générale, vous n'avez pas besoin d'inclure les références de caractères nommées pour `>`, `"` ou `&`.

Principes de base du texte : Entités HTML

Vous avez peut-être remarqué la séquence d'échappement, ou "entité". Comme `<` est utilisé en HTML, vous devez l'échapper à l'aide de `<` ou d'un encodage `<`; moins facile à mémoriser. Quatre entités réservées en HTML sont les suivantes: `<`, `>`, `&` et `"`. Leurs références de caractères sont respectivement **`<`**, **`>`**, **`&`** et **`"`**.

Vous utiliserez souvent les entités **`©`** pour les droits d'auteur (©), **`™`** pour les marques (TM) et **` `** pour les espaces insécables. Les espaces insécables sont utiles lorsque vous souhaitez insérer un espace entre deux caractères ou mots tout en empêchant un saut de ligne à cet endroit.

Il existe plus de 2 000 références de caractères nommées. Toutefois, si nécessaire, chaque caractère, y compris les emoji, a un équivalent encodé qui commence par **`&#`**.

Seuls les caractères non compatibles avec le jeu de caractères doivent être échappés. Si nécessaire, il existe de nombreux outils pour permettre l'échappement de différents caractères.

Vous pouvez également vous assurer d'inclure **`<meta charset="UTF-8">`** dans le **`<head>`** de la page.

Même lorsque vous spécifiez le jeu de caractères au format UTF-8, vous devez toujours échapper la `<` lorsque vous voulez imprimer ce caractère à l'écran. En règle générale, vous n'avez pas besoin d'inclure les références de caractères nommées pour `>`, `"` ou `&`.

Les liens



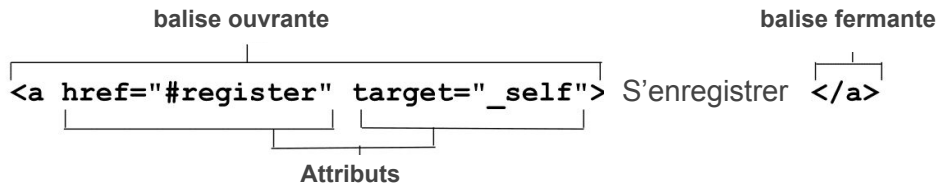
Les liens

La balise d'ancrage **<a>** et l'attribut **href** créent un lien hypertexte. Les liens sont la colonne vertébrale d'Internet. La première page Web contenait 25 liens, indiquant "Tout ce qui existe en ligne concernant W3 est directement ou indirectement associé à ce document". Il est très probable que tout ce qu'il y a en ligne à propos de W3 est également lié directement ou indirectement à ce document !

La puissance du Web c'est la balise **<a>** !

Les liens représentent une connexion entre deux ressources, dont l'une est le document actuel. Les liens peuvent être créés par **<a>**, **<area>**, **<form>** et **<link>**.

*Nous allons détailler la balise **<a>** composée d'une seule lettre, mais pas si simple...*



Les liens : l'attribut href

Bien que cela ne soit pas obligatoire, l'attribut **href** figure dans presque toutes les balises `<a>`.

L'adresse du lien hypertexte permet de transformer `<a>` en lien.

L'attribut **href** permet de créer des liens hypertextes vers des emplacements de la page active, d'autres pages d'un site ou d'autres sites.

Il peut également être codé pour télécharger des fichiers ou pour envoyer un e-mail à une adresse spécifique, y compris en incluant un objet et une suggestion de contenu du corps de l'e-mail.

```
<a href="https://institutducode.com">Institut du code</a>
<a href="#teachers">Professeurs</a>
<a href="https://institutducode.com#teachers">IDC Professeurs</a>
<a href="mailto:codeur@institutducode.com">Envoyer un mail à John</a>
<a href="tel:8005551212">Appeler John</a>
```

Le premier lien inclut une URL absolue, qui peut être utilisée sur n'importe quel site dans le monde pour accéder à la page d'accueil du site. Les URL absolues incluent un protocole (dans ce cas, `https://`) et un nom de domaine. Lorsque le protocole est écrit simplement sous la forme `//`, il s'agit d'un protocole implicite qui signifie "utiliser le même protocole que celui actuellement utilisé".

Les URL relatives n'incluent pas de protocole ni de nom de domaine. Elles sont "relatives" au fichier actuel. Par exemple une page, mais qui comporte plusieurs sections (id).

Les liens : l'attribut href

Les navigateurs acceptent également deux liens de haut de page: si vous cliquez sur `Top` (non sensible à la casse) ou simplement sur `Top`, l'utilisateur est fait défiler jusqu'en haut de la page, sauf si un élément dont l'ID **top** est défini dans la même casse.

```
<a href="#top">Allez au début</a>
```

L'attribut **href** peut commencer par **mailto:** ou **tel:** pour envoyer des e-mails ou passer des appels. Le traitement du lien dépend de l'appareil, du système d'exploitation et des applications installées.

Le lien **mailto** ne doit pas nécessairement inclure d'adresse e-mail, mais il peut être prérempli avec du texte **cc**, **bcc**, **subject** et **body** pour préremplir l'e-mail. Si on clique sur un lien "mailto" le client de messagerie par défaut, s'ouvre.

L'application qui s'ouvre lorsque l'utilisateur clique ou appuie sur un lien **tel** ou appuie sur la touche Entrée dépend aussi du système d'exploitation, des applications installées et des paramètres de l'appareil.

Un iPhone peut ouvrir FaceTime, l'application Téléphone ou les contacts. Un bureau Windows peut ouvrir Skype, s'il est installé.

Les liens : l'attribut href

Il existe plusieurs autres types d'URL, tels que les blobs et les URL de données (avec l'attribut `download` -voir page suivante-).

Lorsque vous insérez des liens susceptibles d'ouvrir d'autres applications installées, nous vous recommandons d'en informer l'utilisateur. Assurez-vous que le texte situé entre les balises d'ouverture et de fermeture indique à l'utilisateur le type de lien qu'il est sur le point d'activer.

Les sélecteurs d'attributs CSS, tels que `[href^="mailto:"]`, `[href^="tel:"]` et `[href$=".pdf"]`, peuvent être utilisés pour cibler des styles par type d'application.

L'attribut **download** doit être inclus lorsque **href** pointe vers une ressource téléchargeable.

La valeur de l'attribut de téléchargement correspond au nom de fichier suggéré pour la ressource à enregistrer dans le système de fichiers local de l'utilisateur.

```
<a href="blob:https://jakearchibald.github.io/944a5fc8-fdb3-458a-91ee-cdd5964b6646" download="blob.svg"></a>
```

Pour créer un lien vers une ressource téléchargeable, incluez l'URL de la ressource en tant que valeur de l'attribut **href** et le nom de fichier suggéré qui peut être utilisé dans le système de fichiers de l'utilisateur comme valeur de l'attribut **download**.

Les liens : Contexte de navigation

L'attribut **target** permet de définir le contexte de navigation pour la navigation par lien (et l'envoi de formulaire).

Les valeurs possibles incluent la valeur par défaut **_self**, qui est la fenêtre actuelle, **_blank**, qui ouvre le lien dans un nouvel onglet, **_parent**, qui est le parent si le lien actuel est imbriqué dans un objet ou un iFrame, et **_top**, qui est l'ancêtre de niveau supérieur, particulièrement utile si le lien actuel est profondément imbriqué.

_top et **_parent** sont identiques à **_self** si le lien n'est pas imbriqué.

L'attribut **target** ne se limite pas à ces quatre termes clés: n'importe quel terme peut être utilisé.

Chaque contexte de navigation (c'est-à-dire chaque onglet de navigateur) est associé à un nom de contexte de navigation. Les liens peuvent ouvrir des liens dans l'onglet actuel, dans un nouvel onglet sans nom, ou dans un onglet nommé nouveau ou existant.

Par défaut, le nom est une chaîne vide. Pour ouvrir un lien dans un nouvel onglet, l'utilisateur peut effectuer un clic droit et sélectionner "Ouvrir dans un nouvel onglet".

Les développeurs peuvent forcer cette opération en incluant **target="_blank"**.

Les listes



Les listes

HTML propose plusieurs façons de baliser les listes. Il existe des listes numérotées (****), des listes à puces (****) et des listes de descriptions (**<dl>**). Les éléments de liste (****) sont imbriqués dans des listes numérotées et des listes à puces. Dans une liste de descriptions, vous trouverez les termes descriptifs (**<dt>**) et les détails de la description. **<dd>**.

Dans les formulaires HTML, les listes d'éléments **<option>** constituent le contenu de **<datalist>**, **<select>** et **<optgroup>** dans un élément **<select>**. Celles-ci sont utilisées dans les formulaires HTML.

Dans les menus et les listes à puces, les éléments sont généralement affichés sous forme de puces. Dans les listes ordonnées, elles sont généralement précédées d'un compteur croissant tel qu'un chiffre ou une lettre. Les puces et l'ordre de numérotation peuvent être contrôlés et inversés avec HTML, CSS, ou les deux.

Par défaut, les éléments de liste (triés ou non) sont précédés de chiffres ou de puces. Toutefois, même si vous ne voulez pas que les listes ressemblent à des listes, vous aurez toujours d'une liste d'éléments, comme dans les barres de navigation, une liste de tâches avec des cases à cocher au lieu de puces, ou des questions vrai et faux dans un test à choix multiples.

Pour toutes ces listes sans puces, il est approprié d'utiliser des éléments de liste HTML.

Les listes

Listes à puces

L'élément `` est l'élément parent des listes d'éléments non ordonnées.

Les seuls enfants d'un élément `` sont un ou plusieurs éléments d'élément de liste ``.

Créons une liste de fruits. Nous utilisons une liste non ordonnée, car l'ordre n'a pas d'importance :

```
<ul>
  <li>Ananas</li>
  <li>Citron</li>
  <li>Raisin</li>
</ul>
```

Listes ordonnées

L'élément `` est l'élément parent des listes d'éléments ordonnées. Les seuls enfants d'un élément `` sont un ou plusieurs éléments `` ou des éléments de liste.

Dans ce cas, les "puces" représentent des nombres de tous types. Le type peut être défini en CSS avec la propriété `list-style-type` ou via l'attribut `type`.

L'élément `` comporte trois attributs spécifiques à un élément: `type`, `reversed` et `start`.

```
<ol>
  <li>Ananas</li>
  <li>Citron</li>
  <li>Raisin</li>
</ol>
```

L'attribut **type** énuméré définit le type de numérotation. Cinq valeurs sont valides pour `type`. La valeur par défaut est **1** pour les nombres, mais vous pouvez également utiliser **a**, **A**, **i** ou **I** pour les lettres minuscules et majuscules, ou les chiffres romains. La propriété `list-style-type` fournit beaucoup plus de valeurs.

```
<ol type="A">
  <li>Ananas</li>
  <li>Citron</li>
  <li>Raisin</li>
</ol>
```

Éléments de liste

Nous avons utilisé l'élément ``, mais nous ne l'avons pas encore introduit officiellement.

L'élément `` peut être un enfant direct d'une liste non ordonnée (``), d'une liste numérotée (``) ou d'un menu (`<menu>`).

L'élément `` doit être imbriqué en tant qu'enfant de l'un de ces éléments et n'est valide nulle part ailleurs.

La spécification n'exige pas la fermeture d'un élément de liste, car il est fermé implicitement lorsque le navigateur rencontre la prochaine balise d'ouverture `` ou la balise de fermeture de liste requise: ``, ``, `</menu>`.

Bien que la spécification ne l'exige pas, fermez vos balises `` !

Cela rend votre code plus facile à lire et vous en serez reconnaissant par la suite. Il est plus facile de fermer tous les éléments que de se souvenir des balises qui doivent être fermées et de celles qui comportent une balise de fermeture facultative.

Il n'existe qu'un seul attribut `` spécifique à l'élément: **value**, un entier. **Value** n'est utile que sur un `` lorsque le `` est imbriqué dans une liste numérotée et n'a aucune signification pour les listes ou les menus non ordonnés. Elle remplace la valeur du début de `` en cas de conflit.

Éléments de liste

```
<ol start="4">
  <li value="7">Ananas</li>
  <li value="25">Citron</li>
  <li>Raisin</li>
</ol>
```

value correspond au numéro de l'élément dans une liste numérotée. Pour les éléments de liste suivants, poursuivez la numérotation à partir de la valeur définie, sauf si cet élément possède également un attribut value défini. Il n'est pas nécessaire que la valeur soit dans l'ordre. Toutefois, si elle ne l'est pas, il devrait y avoir une bonne raison. Lorsque vous combinez **reversed** sur **** avec des attributs **value** pour les éléments de la liste, le navigateur définit cet élément **** sur la valeur fournie, puis comptabilise les **** qui le précèdent et génère un compte à rebours pour ceux qui suivent. Si un deuxième élément de liste possède un attribut de valeur, le compteur est réinitialisé au niveau de ce deuxième élément de liste et la valeur suivante diminue d'une unité. Tout cela peut également être contrôlé à l'aide de **compteurs CSS** fournissant du contenu généré pour le pseudo-élément **::marker**.

Si le numéro est purement de la présentation, utilisez du code CSS.

 *Si la numérotation est importante d'un point de vue sémantique ou a un sens, utilisez ces attributs.*

Listes de descriptions

Lors de la création d'une liste de termes et de leurs définitions ou descriptions, ou de listes similaires de paires clé/valeur, les éléments des listes de description fournissent la sémantique appropriée.

Le rôle implicite d'un `<dt>` est **term**, **listitem** étant un autre rôle autorisé.

Le rôle implicite d'un `<dd>` est **definition**. Aucun autre rôle n'est autorisé.

Contrairement à `` et ``, `<dl>` n'a pas de rôle ARIA implicite. C'est logique, car `<dl>` n'est pas toujours une liste. En revanche, les rôles **list** et **group** sont acceptés.

Le plus souvent, vous rencontrerez des listes de descriptions contenant le même nombre d'éléments `<dt>` et `<dd>`. Toutefois, les listes de descriptions ne sont pas toujours et n'ont pas besoin d'être des paires terme/description. Vous pouvez en avoir plusieurs à un ou un à plusieurs, comme un terme de dictionnaire ayant plusieurs définitions.

Chaque `<dt>` est associée à au moins un `<dd>`, et chaque `<dd>` est associé à au moins un `<dt>`.

```
<dl>
  <dt>Banane</dt>
  <dd>Fruit jaune</dd>
  <dt>Orange</dt>
  <dd>A la fois fruit et couleur</dd>
</dl>
```

La navigation



Navigation

Il existe plusieurs types de navigation et plusieurs façons de les afficher.

La navigation locale qui consiste en une série de liens affichant la hiérarchie de la page actuelle par rapport à la structure du site ou les pages que l'utilisateur a suivis pour accéder à la page actuelle, est appelée **fil d'Ariane**.

La **table des matières** d'une page est un autre type de navigation locale.

Une page contenant des liens hiérarchiques vers chacune des pages d'un site est appelée "**plan de site**".

La **section de navigation menant aux pages de premier niveau du site Web** qui se trouve sur chaque page est appelée navigation générale.

La navigation générale peut être affichée de différentes manières, y compris les barres de navigation, les menus déroulants et les menus déroulants. Le même site peut afficher sa navigation générale différemment, en fonction de la taille de la fenêtre d'affichage.

Assurez-vous toujours que les utilisateurs peuvent accéder à n'importe quelle page de votre site avec le moins de clics possible, tout en veillant à ce que la navigation soit intuitive et simple.

Cela dit, il n'y a pas d'exigences spécifiques pour les éléments de navigation.

Navigation : Accéder au contenu

Le premier élément de la page est un lien interne:


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

Lorsque l'utilisateur clique dessus ou clique sur “Entrée”, il fait défiler la page et sélectionne le contenu principal (un élément `<main>` avec un id défini sur “main” :

```
<main id="main">....</main>
```

Vous n'avez peut-être jamais vu le lien sur ce site, même si vous avez lu toutes les sections précédentes. Il ne s'affiche que lorsqu'il est sélectionné.

Pour améliorer la facilité d'utilisation et l'accessibilité, il est important de permettre aux utilisateurs de contourner les blocs de contenu qui sont répétés sur chaque page. Le lien d'ancrage est inclus. Ainsi, lorsqu'un utilisateur du clavier appuie sur la touche “tab” lors du chargement, il peut accéder rapidement au contenu principal du site, ce qui évite d'avoir à parcourir les liens du menu ou le fil d'Ariane.

 Il est possible de masquer le lien tout en gardant à l'esprit que lorsque le lien est sélectionné, ce qui se produit lorsqu'un utilisateur clique sur le lien à l'aide d'un clavier sur la page, ce lien doit être visible par tous les utilisateurs. Ne masquez le contenu à l'état non sélectionné et inactif qu'à l'aide d'un sélecteur semblable à `.visually-hidden:not(:focus):not(:active)`

Navigation : principale

La navigation principale est la section de navigation qui mène aux pages de premier niveau du site Web, qui sont les mêmes sur toutes les pages.

Elle peut également être constituée d'onglets qui ouvrent des listes imbriquées de liens renvoyant vers toutes les sous-sections d'un site ou d'autres menus. Il peut inclure des sections intitulées, des boutons et des champs de recherche. Ces fonctionnalités supplémentaires ne sont pas obligatoires.

La navigation doit apparaître sur chaque page et être identique sur toutes les pages.

Comme pour la navigation principale, le pied de page doit être identiques sur toutes les pages. Mais c'est la seule similitude. La navigation principale permet de naviguer vers toutes les parties du site. Les éléments de navigation dans un pied de page n'ont pas d'exigences spécifiques. En règle générale, le pied de page inclura des liens vers l'entreprise, tels que des mentions légales, concernant l'entreprise et ses carrières, et peut conduire à des sources externes, telles que des icônes de réseaux sociaux.

Navigation : principale

Dans notre code, nous avons déjà notre navigation principale.
Nous n'avons pas de code HTML supplémentaire à ajouter.

Par contre, quand nous sommes dans le cas d'une navigation principale sur plusieurs pages il faut alors se préoccuper de plusieurs points d'accessibilités.

La page sélectionnée doit être signalée par un attribut "aria-current" avec la valeur "page". De plus il est bon d'ajouter un signe graphique indiquant que la page est sélectionnée. par exemple en soulignant le lien et en ajoutant une icône (check) après le nom de la page.

```
<nav>
  <ul>
    <li><a href="\reg.html">S'enregistrer</a></li>
    <li aria-current="page">A propos</li>
    <li><a href="\teachers.html">Professeurs</a></li>
    <li><a href="\feedback.html">Témoignages</a></li>
  </ul>
</nav>
```

Navigation : le fil d'Ariane

Les fils d'Ariane fournissent une navigation secondaire pour aider les utilisateurs à comprendre où ils se trouvent sur un site Web. Ils indiquent généralement la hiérarchie des URL du document actuel et l'emplacement de la page actuelle dans la structure du site. Du point de vue de l'utilisateur, la structure du site peut différer de la structure de fichiers sur le serveur. Ce n'est pas un problème. L'utilisateur n'a pas besoin de savoir comment vous organisez vos fichiers, mais il doit pouvoir naviguer dans votre contenu.

Si le site présente une structure de répertoires hiérarchique simple, le fil d'Ariane sera souvent composé d'un lien vers la page d'accueil, avec un lien vers le fichier d'index de chaque répertoire dans le chemin de l'URL. L'inclusion de la page actuelle est facultative et nécessite un peu plus d'attention.

```
<nav aria-label="Fil d'ariane">
  <ul role="list">
    <li><a href="/">Accueil</a></li>
    <li><a href="/about">A propos</a></li>
    <li aria-current="page">Les acteurs</li>
  </ul>
</nav>
```


Les tables



Les tables

Les tableaux HTML sont utilisés pour afficher des données tabulaires avec des lignes et des colonnes.

La décision d'utiliser un **<table>** doit être basée sur le contenu que vous présentez et sur les besoins de vos utilisateurs par rapport à ce contenu.

Si des données sont présentées, comparées, triées, calculées ou croisées, **<table>** est probablement le bon choix. Si vous souhaitez simplement disposer de manière soignée le contenu non tabulaire, comme un grand groupe de vignettes, les tableaux ne sont pas appropriés.

À la place, il est préférable de créer une liste d'images et de styliser la grille avec CSS.

Nous allons voir tous les éléments qui composent le tableau, ainsi que de certaines fonctionnalités d'accessibilité et d'usabilité à prendre en compte lorsque vous présentez des données tabulaires. Nous aborderons certaines propriétés CSS spécifiques aux tables.

Les tables

Éléments du tableau, dans l'ordre

La balise **<table>** encapsule le contenu du tableau, y compris tous ses éléments.

Le rôle ARIA implicite d'un **<table>** est **table**. Les technologies d'assistance savent que cet élément est une structure de table contenant des données organisées en lignes et en colonnes. Si le tableau conserve un état de sélection, propose une navigation bidimensionnelle ou permet à l'utilisateur de réorganiser l'ordre des cellules, définissez **role="grid"**. Si les lignes de grid peuvent être développées et réduites, utilisez plutôt **role="treegrid"**.

Dans l'élément **<table>**, vous trouverez les en-têtes de tableau (**<thead>**), les corps de tableau (**<tbody>**) et, éventuellement, les pieds de page (**<tfoot>**).

Chacun de ces éléments est composé de lignes du tableau (**<tr>**). Les lignes contiennent des en-têtes de tableau (**<th>**) et des cellules de données (**<td>**) qui, à leur tour, contiennent toutes les données. Dans le DOM, avant tout cela, vous pouvez trouver deux fonctionnalités supplémentaires: la légende de la table (**<caption>**) et les groupes de colonnes (**<colgroup>**). Selon que **<colgroup>** comporte ou non un attribut **span**, il peut contenir des éléments de colonne de tableau (**<col>**) imbriqués.

Les tables

Les enfants de la table sont, dans l'ordre:

- Élément **<caption>**
- Éléments **<colgroup>**
- Éléments **<thead>**
- Éléments **<tbody>**
- Éléments **<tfoot>**

Légende du tableau

En tant qu'élément sémantique natif, **<caption>** est la méthode privilégiée pour donner un nom à une table. **<caption>** fournit un titre de tableau descriptif associé à un programme. Par défaut, il est visible et disponible pour tous les utilisateurs.

L'élément **<caption>** doit être le premier élément imbriqué dans l'élément **<table>**. L'inclure permet à tous les utilisateurs de connaître immédiatement l'objectif de la table, sans avoir à lire le texte qui l'entoure. Vous pouvez également utiliser **aria-label** ou **aria-labelledby** sur **<table>** pour fournir un nom accessible en tant que légende. L'élément **<caption>** ne comporte aucun attribut spécifique à l'élément.

Les tables

La légende apparaît en dehors du tableau. L'emplacement de la légende peut être défini à l'aide de la propriété CSS `caption-side`, ce qui est une bonne pratique que d'utiliser l'attribut obsolète `align`. Cela permet de placer la légende en haut et en bas. Les positionnements à gauche et à droite, avec `inline-start` et `inline-end`, ne sont pas encore entièrement pris en charge. La partie supérieure correspond à la présentation par défaut du navigateur.

```
<table>
  <caption>EDC étudiants</caption>
</table>
```

De préférence, les tableaux de données doivent avoir des en-têtes clairs et une légende, et être suffisamment simples pour être presque explicites. Gardez à l'esprit que tous les utilisateurs n'ont pas les mêmes capacités cognitives. Lorsque le tableau "fait un point" ou qu'il a besoin d'une interprétation, fournissez un bref résumé du point principal ou de la fonction du tableau. L'emplacement de ce résumé dépend de sa longueur et de sa complexité. S'il est bref, utilisez-le comme texte intérieur de la légende. S'il est plus long, résumez-le dans la légende et fournissez le résumé dans le paragraphe précédent le tableau, en associant les deux à l'attribut `aria-describedby`. Une autre option consiste à placer la table dans un `<figure>` et à placer le résumé dans `<figcaption>`.

Les tables

Section de données

Le contenu des tableaux peut comprendre jusqu'à trois sections: zéro ou plusieurs, l'en-têtes de tableau (**<thead>**), le corps de tableau (**<tbody>**) et le pied de page de tableau (**<tfoot>**).

Toutes ces sections sont facultatives. Aucune ou plusieurs sont acceptées.

Ces éléments n'ont pas d'impact sur l'accessibilité du tableau, mais ils sont utiles en termes de convivialité. Ils permettent d'y ajouter des styles.

Par exemple, le contenu de l'en-tête **<thead>** peut être persistant, tandis que le contenu de **<tbody>** peut être fait pour défiler. Les lignes qui ne sont pas imbriquées dans l'un de ces trois éléments contenant des éléments sont implicitement encapsulées dans un élément **<tbody>**.

Tous les trois partagent le même rôle implicite (**rowgroup**).

Aucun de ces trois éléments ne comporte d'attribut spécifique à un élément.

```
<table>
  <caption>EDC étudiants</caption>
  <thead></thead>
  <tbody></tbody>
  <tfoot></tfoot>
</table>
```

Les tables

Contenu de la table

Les tableaux peuvent être divisés en en-têtes, corps et pieds de page, mais cela n'a aucun effet si les tableaux ne contiennent pas de lignes, de cellules ni de contenu.

Chaque ligne du tableau **<tr>** contient une ou plusieurs cellules. Si une cellule est une cellule d'en-tête, utilisez **<th>**, sinon, utilisez **<td>**.

Il y a des attributs pour ajouter une marge intérieure entre les cellules et dans les cellules, pour les bordures et pour l'alignement du texte. La marge intérieure et l'espacement des cellules, qui définissent l'espace entre le contenu d'une cellule et sa bordure, ainsi qu'entre les bordures des cellules adjacentes, doivent être définis respectivement par les propriétés CSS **border-collapse** et **border-spacing**.

Border-spacing n'aura aucun effet si “**border-collapse**” est défini à “**collapse**”.

Si “**border-collapse**” est défini à “**separate**”, il est possible de masquer complètement les cellules vides avec “**empty-cells: hide;**”.

Les tables

Dans cet exemple, nous avons une légende, un en-tête de tableau et un corps de tableau.

L'en-tête comporte une ligne contenant trois cellules `<th>` d'en-tête, créant ainsi trois colonnes. Le corps contient trois lignes de données: la première cellule est une cellule d'en-tête pour la ligne. Nous utilisons donc `<th>` au lieu de `<td>`.

La cellule `<th>` a une signification sémantique, avec des rôles ARIA implicites d'en-tête de colonne ou d'en-tête de ligne.

Il définit la cellule comme en-tête de la colonne ou de la ligne de cellules du tableau, en fonction de la valeur de l'attribut **scope**. Le navigateur utilisera **col** ou **row** par défaut si **scope** n'est pas explicitement défini.

Les autres valeurs de **scope** incluent **rowgroup** et **colgroup**, qui s'avèrent utiles avec les tables complexes.

```
<table>
  <caption>EDC étudiants</caption>
  <thead>
    <tr>
      <th>Nom</th>
      <th>Année</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>John Doe</th>
      <th>2003</th>
    </tr>
    <tr>
      <th>Simon Says</th>
      <th>2002</th>
    </tr>
  </tbody>
</table>
```

```
table,
tr > * {
  border: 1px solid;
}
```


Les images



Les images

Les images décoratives, telles que les dégradés d'arrière-plan sur des boutons ou les images de fond sur des sections de contenu ou sur la page entière, sont des images de présentation et doivent être appliquées en CSS. Lorsqu'une image ajoute du contexte à un document, il s'agit d'un contenu qui doit être intégré au code HTML.

La principale méthode pour inclure des images consiste à utiliser la balise **** avec l'attribut **src** faisant référence à une ressource image et l'attribut **alt** décrivant l'image.

```

```

L'attribut **srcset** sur **** et l'élément **<picture>** permettent d'inclure plusieurs sources d'images avec les requêtes média associées, chacune avec une source d'image de remplacement.

Cela permet d'utiliser le fichier image le plus approprié en fonction de la résolution de l'appareil, des fonctionnalités du navigateur et/ou de la taille de la fenêtre d'affichage. L'attribut **srcset** permet de fournir plusieurs versions d'image en fonction de la résolution et, avec l'attribut **sizes**, de la taille de la fenêtre d'affichage du navigateur.

```

```

Les images

*Chaque image doit au minimum inclure les attributs **src** et **alt**.*

L'attribut **src** correspond au chemin d'accès et au nom de l'image intégrée et permet de fournir l'URL de l'image. Le navigateur récupère ensuite l'élément et l'affiche sur la page. Cet attribut est requis par ****. Sans lui, il n'y a rien à afficher.

L'attribut **alt** fournit un texte alternatif pour l'image, avec une description de l'image pour les personnes qui ne peuvent pas voir l'écran (par exemple, les moteurs de recherche et les technologies d'assistance, et même Alexa, Siri et l'Assistant Google). Le navigateur peut l'afficher si l'image ne se charge pas. Outre les utilisateurs dont les réseaux sont lents ou dont la bande passante est limitée, le texte alt est incroyablement utile dans les e-mails HTML, car de nombreux utilisateurs bloquent les images dans leurs applications de messagerie.

Si l'image est au format **SVG**, incluez également **role="img"**, qui est nécessaire en raison de bugs VoiceOver.

```

```

Les images : règles d'accessibilité

Avant qu'une seule ligne de code ne soit écrite, vous devez d'abord réfléchir au point de l'image et à la façon dont elle sera utilisée. Se poser des questions sur l'objectif et le contexte de l'image peut vous aider à déterminer la meilleure façon de transmettre ces informations à une personne à l'aide de technologies d'assistance (TA) telles que des lecteurs d'écran.

Vous pouvez vous demander:

- *L'image est-elle essentielle pour comprendre le contexte de la fonctionnalité ou de la page ?*
- *Quel type d'informations l'image cherche-t-elle à transmettre ?*
- *L'image est-elle simple ou complexe ?*
- *L'image suscite-t-elle une émotion ou incite-t-elle l'utilisateur à agir ?*
- *Ou bien l'image est-elle un simple objet visuel qui n'a pas de véritable intérêt ?*

Images décoratives

Une image décorative est un élément visuel qui n'ajoute pas de contexte ni d'informations supplémentaires permettant à l'utilisateur de mieux comprendre le contexte. Les images décoratives sont complémentaires et peuvent apporter du style plutôt que de la substance.

Les images : règles d'accessibilité

Si vous décidez qu'une image est décorative, elle doit être masquée de manière programmatique pour les TA. Lorsque vous programmez une image pour qu'elle soit masquée, cela indique à l'AT que l'image n'est pas nécessaire pour comprendre le contenu, le contexte ou l'action de la page.

Il existe de nombreuses façons de masquer des images, y compris en utilisant une alternative au texte vide/null, en appliquant des flux ARIA ou en ajoutant l'image en tant qu'arrière-plan CSS.

Vous trouverez ci-dessous quelques exemples de méthodes pour masquer une image décorative.



Attention : Réfléchissez bien avant de faire ce choix, car le terme décoratif peut avoir différentes significations selon les utilisateurs. Certains utilisateurs de TA veulent entendre des descriptions pour chaque élément visuel à l'écran.

Les utilisateurs peuvent choisir d'ignorer les descriptions de vos images s'ils les considèrent redondantes ou détaillées, mais ils ne peuvent pas imaginer des descriptions qui n'existent pas. En cas de doute, ajoutez des descriptions à vos images.

Un attribut de texte alternatif vide/null diffère d'un attribut de texte alternatif manquant. Si l'attribut texte alternatif est manquant, la TA peut lire le nom du fichier ou le contenu environnant pour fournir à l'utilisateur plus d'informations sur l'image.

Les images : règles d'accessibilité

Rôle défini sur presentation ou none

Un rôle défini sur presentation ou none supprime la sémantique d'un élément de l'exposition à l'arborescence d'accessibilité. En attendant, **aria-hidden="true"** supprime l'intégralité de l'élément (et tous ses enfants) de l'API d'accessibilité.

```
  
  

```

Utilisez **aria-hidden** avec prudence, car il peut masquer des éléments que vous ne souhaitez pas masquer.

Images en CSS

```
.acteur{ background-image: url("../img/acteur.jpg");}
```

Lorsque vous ajoutez une image de fond avec CSS, un lecteur d'écran ne détecte pas le fichier image. Veillez à ce que l'image soit masquée avant d'appliquer cette méthode.

Les images : règles d'accessibilité

Images informatives

Une image informative est une image qui transmet un concept, une idée ou une émotion simple. Voici quelques types d'images informatives : photos d'objets du monde réel, icônes essentielles, dessins simples et images de texte.

Si votre image est informative, vous devez inclure un texte alternatif programmatique décrivant l'objectif de l'image. Les descriptions alternatives d'images (souvent abrégées en "texte alternatif") fournissent aux utilisateurs de TA plus de contexte sur une image et les aident à mieux comprendre le message ou l'intention d'une image.

Les descriptions alternatives simples utilisant des éléments `` sont obtenues en incluant l'attribut **alt**, quel que soit le type de fichier vers lequel il pointe (par exemple, **.jpg**, **.png**, **.svg**, etc.).

```

```

Les images : règles d'accessibilité

Toutefois, lorsque vous utilisez des éléments `<svg>` intégrés, vous devez faire attention à leur accessibilité.

Tout d'abord, étant donné que les **SVG** sont codés sémantiquement, AT les ignore par défaut. Si vous avez une image décorative, ce n'est pas un problème. L'AT l'ignorera comme prévu.

Toutefois, si vous disposez d'une image informative, un **role="img"** ARIA doit être ajouté au modèle pour que la TA la reconnaisse comme une image.

Deuxièmement, les éléments `<svg>` n'utilisent pas l'attribut **alt**, par conséquent, vous devez utiliser différentes méthodes de codage pour ajouter d'autres descriptions à vos images informatives.

```
<svg role="img">  
<title>John l'acteur principal du film</title>  
</svg>
```


Les images : règles d'accessibilité

Images fonctionnelles

Une image fonctionnelle est connectée à une action. Un logo qui renvoie vers la page d'accueil, une loupe utilisée comme bouton de recherche ou une icône de réseaux sociaux qui vous redirige vers un autre site Web ou une autre application sont des exemples d'image fonctionnelle.

Comme les images informatives, les images fonctionnelles doivent inclure une autre description pour informer tous les utilisateurs de leur objectif. Contrairement à une image informative, chaque image fonctionnelle doit décrire l'action de l'image, et non ses aspects visuels.

```
<a href="/"></a>  
<a href="/"><span class="visually-hidden">Fiche Wikipedia</span></a>
```

```
.visually-hidden {  
  position: absolute !important;  
  height: 1px;  
  width: 1px;  
  overflow: hidden;  
  clip: rect(1px, 1px, 1px, 1px);  
  color: red;  
}
```

Les images : règles d'accessibilité

Images complexes

Une image complexe nécessite souvent plus d'explications qu'une image décorative, informative ou fonctionnelle. Il nécessite à la fois une description courte et une description longue pour transmettre l'intégralité du message. Les images complexes comprennent les infographies, les cartes, les graphiques/tableaux et les illustrations complexes. Comme pour les autres types d'images, vous pouvez utiliser différentes méthodes pour ajouter d'autres descriptions à vos images complexes.

```

```

Une façon d'ajouter une explication supplémentaire à une image consiste à créer un lien vers une ressource ou à fournir un lien permettant d'accéder à une explication plus longue plus loin sur la page. Cette méthode est un bon choix, non seulement pour les utilisateurs de TA, mais elle aide également les personnes ayant des handicaps, tels que des troubles cognitifs, d'apprentissage et de lecture, qui pourraient bénéficier d'un accès facile à ces informations d'images supplémentaires à l'écran au lieu d'être englouties dans le code.

```
  
<p id="description">Avec cet organigramme, vous connaîtrez les différents services de la société.</p>
```

Les images : règles d'accessibilité

Une autre méthode consiste à ajouter l'attribut **aria-describedby** à l'élément ****. Vous pouvez associer l'image de manière programmatique à un **ID** contenant une description plus longue. Cette méthode crée une association forte entre l'image et la description complète. La description étendue peut être affichée à l'écran ou masquée visuellement. Toutefois, pensez à la garder visible pour répondre aux besoins d'un plus grand nombre d'utilisateurs.

```
<figure role="group">
  
  <figcaption><a href="peterlipp.html">Consulter le site de Peter Lipp</a></figcaption>
</figure>
```

Pour regrouper des descriptions courtes et plus longues, vous pouvez également utiliser les éléments HTML5 **<figure>** et **<figcaption>**. Ces éléments agissent de la même manière que **aria-describedby**, dans la mesure où ils regroupent sémantiquement les éléments, formant ainsi une association plus forte entre l'image et sa description. L'ajout d'ARIA **role="group"** garantit la rétrocompatibilité avec les anciens navigateurs Web qui ne sont pas compatibles avec la sémantique native de l'élément **<figure>**.

Les images : Rédiger des descriptions d'images alt efficaces

Les attributs **alt** ont pour but d'être courts et concis. Ils fournissent toutes les informations pertinentes transmises par l'image tout en omettant les informations redondantes par rapport à d'autres contenus du document ou inutiles. Dans la rédaction du texte, le ton doit refléter le ton du site.

Pour rédiger un texte alternatif efficace, imaginez que vous lisez la page entière à une personne qui ne peut pas la voir. En utilisant l'élément ****, les utilisateurs de lecteurs d'écran et les robots sont informés que l'élément est une image. Il est inutile d'inclure "*C'est une image/ une photo de*" dans **alt**.

L'utilisateur n'a pas besoin de savoir qu'il y a une image, mais il a besoin de savoir quelles informations l'image transmet.

Normalement, vous ne devez pas dire : "Voici une image granuleuse d'un homme portant un chapeau rouge".

Au lieu de cela, vous devez transmettre ce que l'image véhicule par rapport au contexte du reste du document, et ce que vous transmettez changera en fonction du contexte et du contenu du texte environnant.

Cela va dépendre du contexte !

Les images : Chargement différé

L'attribut “**loading**” indique au navigateur compatible JavaScript comment charger l'image.

La valeur “**eager**” par défaut signifie que l'image est chargée immédiatement lors de l'analyse du code HTML, même si l'image se trouve en dehors de la fenêtre d'affichage visible.

Si vous définissez **loading="lazy"**, le chargement de l'image est différé jusqu'à ce qu'elle soit susceptible d'entrer dans la fenêtre d'affichage. La valeur "probable" est définie par le navigateur en fonction de la distance entre l'image et la fenêtre d'affichage. Cette information est mise à jour lorsque l'utilisateur fait défiler la page.

Le chargement différé permet d'économiser la bande passante et le processeur, ce qui améliore les performances pour la plupart des utilisateurs. Pour des raisons de sécurité, si JavaScript est désactivé, toutes les images seront définies par défaut sur “**eager**”.

```

```

Audio et vidéo



Audio et vidéo

Le langage HTML permet d'intégrer des contenus multimédias dans une page Web.

Nous allons découvrir les fichiers audio et vidéo, y compris comment les intégrer, les commandes utilisateur, l'ajout d'un espace réservé à une image statique pour vos vidéos, ainsi que les bonnes pratiques, y compris pour rendre les fichiers audio et vidéo accessibles.

<audio> et <video>

Les éléments `<video>` et `<audio>` peuvent être utilisés pour intégrer directement des lecteurs multimédias avec l'attribut `src` ou servir d'élément conteneur pour une série d'éléments `<source>`, chacun fournissant une suggestion de fichier `src`. Bien que `<video>` puisse être utilisé pour intégrer un fichier audio, l'élément `<audio>` est préférable pour intégrer des fichiers audio.

Les balises d'ouverture `<video>` et `<audio>` peuvent contenir plusieurs autres attributs, y compris `controls`, `autoplay`, `loop`, `mute` et `preload`, ainsi que les attributs globaux. L'élément `<video>` accepte également les attributs `height`, `width` et `poster`.

Audio et vidéo

```
<video src="videos/machines.webm" poster="images/machine.jpg" controls>  
  <p>Voir <a href="https://youtube.com/link">video sur Youtube</a></p>  
</video>
```

Cet exemple **<video>** comporte une source unique avec l'attribut **src** qui redirige vers la source vidéo. L'attribut **poster** fournit une image à afficher lors du chargement de la vidéo. Enfin, l'attribut **controls** fournit des commandes vidéo utilisateur.

Le contenu de remplacement est inclus entre les balises d'ouverture et de fermeture. Si le navigateur n'est pas compatible avec **<video>** (ou **<audio>**), ce contenu s'affiche. La balise de fermeture **</video>** est obligatoire, même s'il n'y a pas de contenu entre les deux (mais il doit toujours y avoir du contenu de remplacement, n'est-ce pas ?).

Si aucun attribut **src** n'est inclus dans les balises d'ouverture **<video>** ou **<audio>**, incluez un ou plusieurs éléments **<source>**, chacun avec un attribut **src** à un fichier multimédia. L'exemple suivant inclut trois options de fichier multimédia, un contenu de remplacement et des sous-titres en anglais et en français entre les balises d'ouverture et de fermeture.

Audio et vidéo

```
<video poster="images/machine.jpg" controls>
  <source src="videos/machines.webm" type="video/webm">
  <source src="videos/machines.mp4" type="video/mp4">
  <source src="videos/machines.ogv" type="video/ogv">
  <track label="English" kind="subtitles" srclang="en" src="vtt/subtitles-en.vtt" default />
  <track label="Francais" kind="subtitles" srclang="fr" src="vtt/subtitles-fr.vtt" />
  <p>Voir <a href="https://youtube.com/link">video sur Youtube</a></p>
</video>
```

Chaque élément enfant `<source>` inclut un attribut `src` pointant vers la ressource, et l'attribut `type` informe le navigateur du type de contenu multimédia du fichier associé. Cela empêche le navigateur de récupérer les fichiers multimédias qu'il ne pourrait pas décoder.

Dans l'attribut `type`, vous pouvez inclure un paramètre `codecs`, qui spécifie exactement la manière dont la ressource est encodée.

Les codecs vous permettent d'inclure des optimisations multimédias qui ne sont pas encore prises en charge par tous les navigateurs. Le codec est séparé du type de contenu par un point-virgule.

Audio et vidéo

Par exemple, le codec peut être écrit à l'aide d'une syntaxe intuitive, telle que

<source src="videos/machines.webm" type="video/webm; codecs=vp8,vorbis">, qui indique que les fichiers WebM contiennent du contenu vidéo VP8 et du contenu audio vorbis. Les codecs peuvent également être plus difficiles à déchiffrer, tels que **<source src="videos/machines.mp4" type="video/mp4; codecs=avc1.4d002a">**, qui indique que l'encodage MP4 est Advanced Video Coding Main Profile Level 4.2.

Par défaut, lors de l'affichage d'une vidéo, la première image de la vidéo est une image fixe lorsqu'elle est disponible. C'est quelque chose qui peut être contrôlé. L'attribut poster permet d'afficher la source d'une image pendant le téléchargement de la vidéo et jusqu'à sa lecture. Si la vidéo est lue, puis mise en pause, l'affiche n'est plus affichée.

Veillez à définir le format de votre vidéo. En effet, lors du chargement de la vidéo, toute différence de taille entre l'affiche et la vidéo entraîne un ajustement de la mise en page et un nouvel affichage.

Audio et vidéo

Incluez toujours l'attribut boolean controls. Les commandes utilisateur sont ainsi visibles, ce qui permet aux utilisateurs de contrôler les niveaux audio, de couper complètement le son et d'afficher la vidéo en plein écran. Omettre controls nuit à l'expérience utilisateur, surtout si l'attribut booléen autoplay est inclus.

Notez que certains navigateurs ne tiennent pas compte de la directive d'attribut autoplay si l'attribut booléen muted est omis. En effet, la lecture automatique d'un fichier multimédia nuit généralement à l'expérience utilisateur, même lorsque le son est coupé et que les commandes sont visibles.

Titres

Entre les balises d'ouverture et de fermeture obligatoires de l'audio et de la vidéo, incluez un ou plusieurs éléments <track> pour spécifier des pistes de texte temporisées. L'exemple suivant inclut deux fichiers <track>, fournissant des sous-titres avec codes temporels en anglais et en français.

```
<track label="English" kind="subtitles" srclang="en" src="vtt/subtitles-en.vtt" default />  
<track label="Francais" kind="subtitles" srclang="fr" src="vtt/subtitles-fr.vtt" />
```

Audio et vidéo

Les fichiers de suivi, spécifiés dans l'attribut `src`, doivent être au format WebVTT (.vtt). Les fichiers doivent appartenir au même domaine que le document HTML, sauf si l'attribut `crossorigin` est inclus.

Il existe cinq valeurs énumérées pour l'attribut `kind` de suivi: subtitles, captions, descriptions, chapters et d'autres metadata.

Incluez subtitles avec l'attribut `srclang` pour la transcription et la traduction des dialogues. La valeur de chaque attribut `label` est présentée à l'utilisateur en tant qu'option. Le contenu de l'option de VTT sélectionnée s'affiche par-dessus la vidéo. Vous pouvez styliser l'apparence des sous-titres en ciblant `::cue/` `::cue()`.

La valeur `kind="caption"` doit être réservée à la transcription et aux traductions incluant des effets sonores et d'autres informations audio pertinentes. Cette fonctionnalité n'est pas réservée aux spectateurs sourds. Peut-être qu'un utilisateur ne trouve pas son casque, alors qu'il a activé les sous-titres. Ou peut-être n'a-t-il pas bien compris les derniers points de discussion d'un podcast favori, alors qu'il souhaite lire la transcription pour confirmer sa compréhension. Disposer d'autres moyens d'accéder au contenu audio et vidéo est à la fois important et pratique.

Audio et vidéo

Le kind="description" est destiné à la description textuelle du contenu visuel de la vidéo pour les utilisateurs qui ne peuvent pas voir la vidéo, qu'ils utilisent un système sans écran comme Google Home ou Alexa, ou qu'ils soient aveugles.

Fournissez des sous-titres au format WebVTT pour rendre la vidéo accessible aux personnes malentendantes. N'oubliez pas que le handicap est un décalage entre une personne et son environnement actuel. Les déficiences auditives peuvent être dues au fait que l'utilisateur se trouve dans un environnement bruyant, que ses haut-parleurs sont défectueux ou qu'il a un casque cassé, ou qu'il souffre d'une perte auditive ou est sourde. Veiller à ce que votre contenu soit accessible aide beaucoup plus de personnes que vous ne le pensez ; cela aide tout le monde.

Transcriptions

Comme les cousins des sous-titres, les transcripts sont des documents textuels détaillés qui capturent tous les mots, les sons et les informations visuelles importantes de vos contenus multimédias. Les transcriptions aident principalement les personnes malentendantes ou sourdes, tandis que les transcriptions descriptives aident les personnes sourdes et sourdes.

Audio et vidéo

Les transcriptions sont également utiles pour les personnes souffrant de troubles cognitifs ou pour les personnes qui souhaitent consulter le contenu à leur propre rythme.

Si les transcriptions sont généralement plus détaillées que les sous-titres, elles sont très similaires en termes de format et d'objectif. Elles sont tellement similaires que de nombreuses personnes ajoutent d'abord des sous-titres à leurs contenus multimédias, les exportent, puis les utilisent comme base pour leurs transcriptions. Remanier vos sous-titres pour créer vos transcriptions vous permet de gagner du temps.

Les robots de recherche ne peuvent pas accéder à vos sous-titres, mais peuvent explorer vos transcriptions de texte. Lorsque vous incluez des transcriptions dans vos fichiers multimédias, vous améliorez l'optimisation du référencement. Il s'agit de l'une des rares exceptions dans lesquelles le contenu en double n'est pas source de confusion pour les utilisateurs ou n'est pas pénalisé par des algorithmes de moteur de recherche.

Chaque lecteur multimédia gère les transcriptions de manière différente. Il est possible que certains fournisseurs n'intègrent pas cette fonctionnalité dans leur lecteur multimédia et que, même dans ce cas, certains utilisateurs n'aient pas accès à l'interface de transcription.

Audio et vidéo

Pour vous assurer que votre transcription est accessible à tous les utilisateurs:

- en incluant le texte de la transcription directement en contexte, sur la page contenant la vidéo intégrée.
- Ajouter un lien vers un PDF accessible contenant la transcription.
- Lien vers la copie figurant sur une autre page.
- Inclure un lien vers la transcription, où qu'elle se trouve, dans la description de la vidéo sur la plate-forme de lecteur multimédia que vous avez utilisée (comme YouTube ou Vimeo).
- Par exemple, accédez à YouTube pour regarder la vidéo Password format? | Il n'y a pas meilleur navigateur que Chrome et découvrez un exemple de transcription.

Audiodescriptions

La description audio est un autre support utilisé pour aider les personnes handicapées. Ce type de média alternatif utilise un narrateur pour expliquer des informations visuelles importantes aux personnes qui ne peuvent pas voir le contenu visuel. Ces descriptions incluent des informations non verbales telles que des expressions faciales, des actions non prononcées et l'environnement en arrière-plan dans les contenus vidéo uniquement et multimédias.

Audio et vidéo

Les audiodescriptions ont parfois besoin d'être très détaillées en raison de la grande quantité d'informations à partager avec le spectateur. S'il n'y a pas assez de pauses naturelles dans la vidéo pour les audiodescriptions, les audiodescriptions étendues sont utilisées. Dans les audiodescriptions étendues, la vidéo est mise en pause pour laisser suffisamment de temps au narrateur pour transmettre toutes les informations dans le contenu multimédia avant de lire le reste de la vidéo.

Interprétation en langue des signes

Un autre type de média majeur que vous pouvez rencontrer est l'interprétation de la langue des signes, où un interprète commente la partie auditive du contenu audio uniquement ou multimédia en langue des signes. C'est très important pour de nombreuses personnes sourdes, car la langue des signes est leur première et leur langue la plus courante. L'interprétation en langue des signes est souvent plus expressive et détaillée que les documents écrits, offrant ainsi une expérience beaucoup plus riche que les sous-titres ou les transcriptions seules. Cela dit, l'interprétation en langue des signes peut être chronophage et coûteuse pour de nombreuses organisations. Et même si vous avez le temps et le budget nécessaires pour ajouter l'interprétation en langue des signes à vos supports, il existe plus de 300 langues des signes différentes dans le monde. Ajouter une seule interprétation en langue des signes à vos supports ne suffirait pas à répondre aux besoins d'un public international.

Audio et vidéo

Remarques sur les vidéos en arrière-plan

Votre équipe marketing ou de conception peut souhaiter que votre site inclue une vidéo de fond. En général, cela signifie qu'ils souhaitent une vidéo en boucle, en lecture automatique et sans contrôle, sans commande. Le code HTML peut se présenter comme suit:

```
<video autoplay loop muted poster="images/machine.jpg" role="none">  
  <source src="videos/machines.webm" type="video/webm">  
  <source src="videos/machines.mp4" type="video/mp4">  
  <source src="videos/machines.ogv" type="video/ogv">  
</video>
```

Les vidéos en arrière-plan ne sont pas accessibles. Aucun contenu ne doit être transmis via des vidéos en arrière-plan sans donner aux utilisateurs le contrôle total de la lecture et l'accès à l'ensemble des sous-titres. Cette vidéo étant purement décorative, elle inclut le rôle ARIA de **none** et omet tout contenu de remplacement. Pour améliorer les performances des vidéos dont le son est toujours coupé, supprimez la piste audio de vos sources multimédias. Si votre vidéo est lue pendant cinq secondes ou moins, les consignes d'accessibilité n'exigent pas de mécanisme de mise en pause. Cependant, tout élément comportant l'attribut booléen **loop** sera lu en boucle indéfiniment par défaut, au-delà de cette limite de temps de cinq secondes ou de toute autre durée. Pour une expérience utilisateur de qualité, incluez toujours une méthode pour mettre la vidéo en pause. Pour ce faire, il suffit d'inclure controls.

Comprendre le “focus”



Comprendre le “focus”

Par défaut, les éléments interactifs, y compris les commandes de formulaire, les liens et les boutons, sont sélectionnables et accessibles avec la touche de tabulation. Les éléments accessibles via la touche de tabulation font partie de l'ordre de navigation séquentiel du document. Les autres éléments sont inertes, ce qui signifie qu'ils ne sont pas interactifs. Avec les attributs HTML, il est possible de rendre les éléments interactifs inertes et de les rendre interactifs.



Remarque : Pour des raisons d'usabilité, assurez-vous toujours que l'utilisateur sache sur quel élément est sélectionné. Incluez les styles CSS `:focus`, `:focus-visible` et, éventuellement, `:focus-within`. C'est très important: les parcours de formation CSS et Accessibilité comportent des sections "Apprendre" consacrées aux styles de focus.

Par défaut, l'ordre du focus de navigation est le même que l'ordre visuel, qui correspond à l'ordre du code source. Des attributs HTML peuvent modifier cet ordre, et des propriétés CSS peuvent modifier l'ordre visuel du contenu. Modifier l'ordre de tabulation en HTML ou l'ordre d'affichage visuel avec CSS peut nuire à l'expérience utilisateur.

Ne modifiez pas l'ordre de tabulation perçu et réel avec CSS et HTML. Comme le montrent les deux exemples suivants, les commandes de tabulation différentes de l'ordre visuellement attendu sont déroutantes pour les utilisateurs et nuisent à l'expérience utilisateur.

Comprendre le “focus”

Dans cet exemple, la valeur de l'attribut `tabindex` a rendu l'ordre de tabulation chaotique:

```
<p>Cliquer dans un champs et utiliser la touche “tab”.</p>
<ol>
  <li><input tabindex="3"></li>
  <li><input tabindex="6"></li>
  <li><input tabindex="2"></li>
  <li><input tabindex="0"></li>
  <li><input tabindex="-1"></li>
  <li><input tabindex="0"></li>
</ol>
```

```
input, ol{ font-size: 2rem }
input { width: 5rem;}
input:focus {
  background-color: lightgray;
  border-color: green;
  outline: 2px solid red;
}
ol{
  columns: 2 7rem;
}
```

Dans cet exemple, CSS a créé une divergence entre l'ordre de tabulation et l'ordre visuel du contenu:

Comprendre le “focus”

Dans cet exemple, la valeur de l'attribut `tabindex` a rendu l'ordre de tabulation chaotique:

```
<p>Cliquer dans le champs et utiliser la touche “tab”.  
<label><input></label></p>  
<div>  
  <span tabindex="0">Flexbox.</span>  
  <span tabindex="0">la propriété CSS </span>  
  <span tabindex="0">avec</span>  
  <span tabindex="0">stylée </span>  
  <span tabindex="0">et </span>  
  <span tabindex="0">Cette</span>  
  <span tabindex="0">l'ordre</span>  
  <span tabindex="0">inverse</span>  
  <span tabindex="0">dans</span>  
  <span tabindex="0">écrive</span>  
  <span tabindex="0">à été </span>  
  <span tabindex="0">phrase</span>  
</div>
```

```
div {  
  display: inline-flex;  
  flex-flow: row-reverse;  
  gap: 0.5em;  
}  
  
span:nth-of-type(6){  
  order: 3;  
}  
  
span {  
  border: 1px dashed;  
}  
  
span:focus {  
  background-color: lightgray;  
  border-color: green;  
  outline: 2px solid red;  
}
```

Comprendre le “focus”

La déclaration **flex-flow: row-reverse;** a inversé l'ordre visuel.

En outre, la propriété CSS **order** a été appliquée au sixième mot, "Cette", qui a déplacé ce mot de manière visuelle.

La séquence de tabulation correspond à l'ordre du code, qui ne correspond plus à l'ordre visuel, ce qui crée une déconnexion pour les utilisateurs de clavier.



Remarque : Les fonctionnalités CSS, comme flexbox, grid, position, transforms et multi-column, peuvent modifier l'ordre visuel du contenu. Assurez-vous que votre contenu suit toujours un ordre de tabulation logique pour toutes les tailles de fenêtre d'affichage. Testez votre contenu en le parcourant avec la touche de tabulation à l'aide du clavier: appuyez sur Maj+Tabulation pour revenir en arrière. Assurez-vous toujours que l'élément sélectionné dans CSS est clairement identifiable, et évitez les problèmes d'accessibilité en ne réorganisant pas les éléments sélectionnables avec CSS.

Rendre les éléments inertes interactifs

Les attributs **contenteditable** et **tabindex**, qui sont des attributs globaux, peuvent être ajoutés à n'importe quel élément, ce qui les rend sélectionnables dans le processus. Les éléments sélectionnables peuvent également être sélectionnés avec la souris ou le pointeur en définissant l'attribut autofocus, ou à l'aide d'un script, par exemple avec `element.focus()`.

L'attribut **tabindex**

L'attribut global **tabindex** permet aux éléments qui, autrement, ne pourraient pas être sélectionnés, d'être sélectionnés, généralement à l'aide de la touche de tabulation, d'où le nom.

La valeur de l'attribut **tabindex** correspond à un entier. Une valeur négative rend un élément sélectionnable, mais pas avec la touche de tabulation. Une valeur **tabindex** de 0 rend l'élément sélectionnable et accessible via la touche de tabulation, en ajoutant l'élément sur lequel il est appliqué à l'ordre de navigation séquentielle dans l'ordre du code source. Une valeur supérieure ou égale à 1 rend l'élément sélectionnable et accessible via la touche de tabulation, mais l'ajoute à une séquence de tabulation prioritaire et, comme nous l'avons vu ci-dessus, doit être évitée.

Comprendre le “focus”

```
<div data-action="click" data-icon="share" data-label="instagram" role="button" tabindex="0">
  <svg aria-label="share" role="img" xmlns="http://www.w3.org/2000/svg">
    <use href="#shareIcon" />
  </svg>
  <span>Partager</span>
</div>
```



Remarque : `role="button"` indique aux utilisateurs de lecteurs d'écran que cet élément doit se comporter comme un `<button>`. Lorsque vous créez des éléments personnalisés qui imitent des éléments sémantiques existants, l'ajout d'un rôle ARIA est approprié et attendu. L'élément doit fournir toutes les caractéristiques de l'élément répliqué. Pour ce faire, étendez l'élément répliqué, par exemple en étendant `HTMLButtonElement`, ou ajoutez `tabindex="0"` et utilisez JavaScript pour programmer toutes les fonctionnalités de l'élément qu'il imite, y compris la gestion des événements de pointeur et les pressions sur les touches Entrée et Espace. Si `<button>` avait été utilisé pour le bouton au lieu de créer un élément personnalisé, les attributs `tabindex` et `role` auraient été inutiles, et le navigateur a fourni les événements de pointeur et de clavier.

Rendre les éléments inertes interactifs

Un attribut **tabindex** avec une valeur négative rend l'élément sélectionnable, mais pas accessible via la touche de tabulation. L'élément est capable de recevoir le focus, par exemple via `HTMLElement.focus()`, mais il ne fait pas partie de l'ordre de navigation de sélection séquentielle.

La convention pour les éléments sélectionnables et non tabables consiste à utiliser **tabindex="-1"**. Notez que si vous ajoutez **tabindex="-1"** à un élément interactif, il ne sera plus accessible via la touche de tabulation.

Évitez de modifier l'ordre du DOM avec **tabindex**. Les commandes de tabulation modifiées peuvent créer de mauvaises expériences utilisateur, mais elles sont également difficiles à gérer et à entretenir pour les développeurs.

Rendre les éléments inertes interactifs

L'attribut **contenteditable** a été abordé précédemment. Définir **contenteditable="true"** sur n'importe quel élément le rend modifiable, sélectionnable et fait partie de l'ordre de tabulation. Le comportement de sélection est semblable à celui du paramètre **tabindex="0"**, mais il est différent. Les éléments **contenteditable** imbriqués sont sélectionnables, mais pas accessibles via la touche de tabulation. Pour rendre un élément **contenteditable** imbriqué accessible via la touche tablable, ajoutez **tabindex="0"**, qui l'ajoutera à l'ordre de navigation séquentielle.

Attribut autofocus

Bien que la valeur booléenne **autofocus** soit un attribut global pouvant être défini sur n'importe quel élément, elle ne rend pas un élément inerte interactif. Lors du chargement de la page, le premier élément sélectionnable avec l'attribut **autofocus** défini est sélectionné, tant qu'il est affiché et qu'il n'est pas imbriqué dans un élément **<dialog>**.

Les éléments de la séquence précédant l'élément sélectionné automatiquement sont simplement ignorés. Il est donc déconseillé d'inclure l'attribut **autofocus**.

Rendre les éléments interactifs inertes

Il existe également des attributs HTML permettant de supprimer les éléments interactifs de la séquence de tabulation. L'ajout d'un **tabindex** négatif aux éléments sélectionnables, de l'ajout de l'attribut **disabled** aux commandes de formulaire compatibles et de l'ajout de l'attribut global inert à un conteneur rend les éléments impossibles à onglets. Ces trois attributs ne sont PAS interchangeables.

Valeur tabindex négative

Comme nous l'avons vu ci-dessus, un attribut **tabindex** avec une valeur négative rend un élément sélectionnable, mais pas tabable. Bien qu'il ne soit pas nécessaire d'ajouter **tabindex="0"** à un élément sélectionnable par défaut, y compris les liens, les boutons, les commandes de formulaire et les éléments qui sont **contenteditable**, l'ajout d'un **tabindex** avec une valeur négative supprime les éléments normalement tabables de l'ordre de navigation du focus séquentiel.

Une valeur tabindex négative empêche les utilisateurs du clavier de se concentrer sur les éléments interactifs, mais ne désactive pas l'élément. Les utilisateurs du pointeur peuvent toujours se concentrer sur l'élément. Pour désactiver un élément, utilisez l'attribut disabled.

Rendre les éléments interactifs inertes

Disabled

L'attribut booléen “**disabled**” rend les commandes de formulaire auxquelles il est appliqué et leurs descendants, le cas échéant, non sélectionnables. Les commandes de formulaire désactivées ne peuvent pas être sélectionnées, ne génèrent pas d'événements de clic et ne sont pas envoyées lors de l'envoi du formulaire. Notez que disabled n'est pas un attribut global. Elle s'applique à <button>, <input>, <optgroup>, <option>, <select>, <textarea>, aux éléments personnalisés associés à un formulaire et à <fieldset>. Lorsque ce paramètre est défini sur <optgroup> ou <fieldset>, toutes les commandes du formulaire enfant sont désactivées, à l'exception du contenu du premier <legend> de <fieldset>.

Les éléments compatibles avec disabled peuvent également être ciblés avec les pseudo-classes :disabled et :enabled. Les éléments désactivés avec l'attribut disabled sont généralement stylisés en gris clair via la feuille de style du navigateur.

L'attribut disabled ne s'applique pas aux éléments inert normalement qui sont sélectionnables via tabindex ou contenteditable. Il ne s'applique pas non plus à l'élément <form> lui-même. Pour les désactiver, vous pouvez utiliser l'attribut inert global.

Rendre les éléments interactifs inertes

Attribut inert

Lorsque l'attribut booléen global **inert** est ajouté à un élément, celui-ci et tout le contenu imbriqué sont désactivés (ni cliquables, ni tabulaires) et supprimés de l'arborescence d'accessibilité.

Bien que **inert** puisse être appliqué à n'importe quel élément, il est généralement utilisé pour des sections de contenu, comme le contenu hors écran ou masqué.

Lorsque vous appliquez **disabled** aux commandes de formulaire, le navigateur fournit un style par défaut et peut être stylisé à l'aide de la pseudo-classe **:disabled**. L'attribut **inert** ne fournit aucun indicateur visuel et n'a pas de pseudo-classe correspondante (bien que le sélecteur d'attributs **[inert]** corresponde).

L'utilisation de **inert** sur du contenu visible sans styles indiquant l'inertie peut nuire à l'expérience utilisateur. Comme le contenu inerte n'est pas disponible pour les utilisateurs de lecteurs d'écran, cela peut prêter à confusion lorsque les utilisateurs voyants voient à l'écran du contenu qui n'est pas accessible aux outils d'accessibilité. Rendez l'inertie très apparente via CSS.

Assurez-vous que le curseur ne se déplace jamais sur un contenu non visible. Tout élément affiché en dehors de l'écran qui n'apparaît pas automatiquement au moment de la sélection doit être rendu inerte.

Typographie



Typographie

Créer et concevoir du contenu accessible ne se résume pas au choix d'une police facile à lire. Même avec des familles de polices accessibles, les personnes souffrant de déficiences visuelles, cognitives, linguistiques ou d'apprentissage peuvent avoir du mal à traiter le texte en raison d'autres éléments tels que les variations de police, la taille, l'espacement et le crénage, pour n'en citer que quelques-uns.

Voyons les principes de base à prendre en compte pour créer un contenu plus inclusif et toucher encore plus de lecteurs. Nous reverrons cela lorsque nous étudierons le CSS.

Police

La police de caractères est un facteur majeur qui peut avoir un impact important sur l'accessibilité de la copie. Votre choix de police et de style peut influencer sur la conception de n'importe quelle page.

Les personnes souffrant de troubles de la lecture, de l'apprentissage et de l'attention tels que la dyslexie et le trouble d'hyperactivité du déficit de l'attention (TDAH), ainsi que les personnes malvoyantes, peuvent tous tirer profit de polices de caractères accessibles.

Typographie

Choisissez des polices de caractères courantes Le moyen le plus rapide de créer une conception accessible consiste à choisir une police de caractères commune (par exemple, Arial, Times New Roman, Calibri, Verdana, etc.).

De nombreuses études de police portant sur les personnes ayant un handicap montrent que les polices de caractères courantes accélèrent la lecture et améliorent la compréhension par rapport aux polices peu courantes. Bien que ces polices de caractères courantes ne soient pas intrinsèquement plus accessibles que d'autres polices de caractères, certaines personnes handicapées ont plus de facilité à les lire parce qu'elles ont une grande expérience de travail avec (ou autour) ces polices de caractères.

En plus de choisir une police de caractères courante, évitez les polices richement décorées ou manuscrites, ainsi que celles ne comportant qu'une seule casse de caractères (par exemple, uniquement les majuscules). Ces polices de caractères spéciales avec des motifs cursifs, des formes originales ou des éléments artistiques comme les lignes fines peuvent sembler belles, mais elles sont beaucoup plus difficiles à lire pour certaines personnes handicapées que les polices de caractères courantes.

Typographie

Caractéristiques des lettres et crénage

Les recherches sur la facilité de lecture des polices de caractères avec ou sans empattement ne sont pas concluantes, mais certains chiffres, lettres ou combinaisons peuvent prêter à confusion pour les personnes souffrant de troubles cognitifs et d'apprentissage du langage. Pour les personnes présentant ces types de handicap, chaque lettre et chaque chiffre doivent être clairement définis et présenter des caractéristiques uniques, de sorte que les lettres ne soient pas confondues avec des chiffres.

Les problèmes de lisibilité les plus courants sont le "l" majuscule (lnde), le "l" minuscule (laltue) et le chiffre "1". De même, les paires de lettres comme b/d, p/q, f/t, i/j, m/w et n/u peuvent parfois basculer vers la gauche, la droite ou le haut pour certains lecteurs.

La lisibilité du texte diminue également lorsque l'espacement des lettres ou le crénage sont trop serrés. Portez une attention particulière au crénage, en particulier entre la paire de lettres r/n problématique.

Les collections de polices Open Source telles que Google Fonts peuvent vous aider à sélectionner la police de caractères la plus inclusive pour votre prochaine conception. Si vous utilisez des produits Adobe, vous pouvez intégrer des familles de polices accessibles provenant de partenaires de fonderie directement dans vos conceptions. Cela inclut la sélection de Google Fonts.

Typographie

Lorsque vous recherchez votre prochaine police de caractères, soyez particulièrement attentif aux points suivants:

Dans la mesure du possible, utilisez des polices courantes.

Évitez d'utiliser des polices élaborées ou manuscrites et celles qui ne contiennent qu'une seule casse de caractères.

- Choisissez une police de caractères aux caractéristiques uniques, en accordant une attention particulière au I majuscule, au l minuscule et au 1.
- Examinez certaines combinaisons de lettres pour vous assurer qu'elles ne correspondent pas exactement à celles de l'autre.
- Vérifiez le crénage, en particulier entre la paire de lettres r/n.

Taille de police et style typographique

Les gens partent souvent du principe qu'il suffit de choisir une famille de polices accessible pour créer un contenu inclusif. Cependant, il est également important de tenir compte de la taille de la police et du style du texte sur la page.

Typographie

Par exemple, les personnes malvoyantes ou daltoniennes peuvent être incapables de lire une partie du texte s'il est trop petit, en utilisant un TA, comme un zoom de navigateur, pour le lire.

D'autres utilisateurs, comme ceux qui souffrent de dyslexie ou de troubles de la lecture, peuvent avoir des difficultés à lire du texte en italique. Les lecteurs d'écran ignorent souvent les méthodes de mise en forme, comme le gras et l'italique, de sorte que l'intention de ces styles n'est pas communiquée aux utilisateurs aveugles ou malvoyants.

Étant donné que vous ne pouvez pas prédire quels sont les besoins de chaque utilisateur, lorsque vous ajoutez des polices à vos produits numériques, tenez compte des directives suivantes:

- Les tailles de police de base doivent être définies avec une valeur relative (% , rem ou em) pour faciliter le redimensionnement.
- Limitez le nombre de variantes de police de caractères, comme la couleur, le texte gras, tout en MAJUSCULES et l'italique pour améliorer la lisibilité. Utilisez plutôt des méthodes pour mettre en avant des mots dans votre texte, tels que des astérisques, des tirets ou la mise en surbrillance d'un mot individuel.
- Dans la mesure du possible, utilisez le balisage au lieu du texte sur les images. Les lecteurs d'écran ne peuvent pas lire le texte intégré dans les images (sans ajout de code supplémentaire), et le texte intégré peut également être pixélisé s'il est agrandi par des utilisateurs malvoyants.

Typographie

Structure et mise en page

Si la police de caractères, la taille de la police et le style typographique sont importants pour la typographie accessible, la structure et la mise en page du texte sur une page peuvent être tout aussi importantes pour la compréhension de l'utilisateur.

Les mises en page complexes peuvent représenter un véritable obstacle pour les personnes atteintes de déficience visuelle ou de lecture, ainsi que pour les 6, 1 millions de personnes aux États-Unis atteintes de TDAH. Avec ces types de handicaps, il est plus difficile pour les utilisateurs de conserver leur place et de suivre le flux du texte en raison de l'absence de chemins linéaires clairs, de titres manquants et d'éléments non regroupés.

Un aspect important des conceptions de mise en page accessibles consiste à distinguer les éléments critiques les uns des autres et à regrouper des éléments similaires. Si les éléments sont trop proches, il peut être difficile de savoir où commence et se termine, surtout s'ils ont un style similaire.

Considérez votre texte comme un ensemble de puces individuelles sur un plan. Cela vous aidera à planifier la structure globale de la page et à utiliser des en-têtes, des sous-titres et des listes chaque fois que nécessaire.

Typographie

Espacement

L'espacement entre les paragraphes, les phrases et les mots est également important, car il permet aux lecteurs de rester concentrés sur le texte et d'améliorer la compréhension visuelle globale de la page. Les longues lignes de texte peuvent être un obstacle pour les lecteurs en situation de handicap, car ils ont du mal à ne pas perdre le fil et à suivre le rythme des contenus. Un bloc de texte restreint permet aux lecteurs de passer plus facilement à la ligne suivante.

Alignement du contenu

De nombreuses personnes handicapées rencontrent un autre problème : lire une copie justifiée. L'espacement inégal entre les mots dans un texte justifié peut entraîner la formation de "rivières d'espace" sur la page, ce qui rend la copie difficile à lire.

La justification de texte peut également entraîner un regroupement ou une étirement artificielle des mots. Les lecteurs peuvent donc avoir du mal à localiser les limites des mots.

Typographie

Heureusement, il existe des consignes claires sur l'espacement et des outils tels que la bonne hauteur de ligne et le calculateur de ratio d'or pour rendre notre texte plus accessible.

L'intégration de ces directives aide les personnes souffrant de troubles du déficit de l'attention, de la lecture et de troubles de la vision à se concentrer davantage sur le texte et moins sur la mise en page.

Bonnes pratiques concernant la structure et la mise en page

Lorsque vous réfléchissez à la structure et à la mise en page, assurez-vous de:

- Utilisez des éléments tels que des en-têtes, des sous-titres, des listes, des nombres, des guillemets et d'autres regroupements visuels pour diviser la page en sections.
- Utilisez des paragraphes et des phrases, et un espacement entre les mots clairement définis.
- Créez des colonnes de texte qui ne dépassent pas 80 caractères de largeur (40 caractères pour les logogrammes).
- Évitez un alignement justifié des paragraphes, ce qui crée des "rivières d'espace" dans le texte.

Autres éléments de texte



Autres éléments de texte

Nous avons abordé la plupart des éléments HTML, mais pas tous. Nous n'avons pas abordé le sujet des éléments de texte. Contrairement aux idées reçues, le HTML était initialement destiné au partage de documents, et non à des vidéos de chats. De nombreux éléments fournissent la sémantique du texte à la documentation.

Exemples de code et rédaction technique

Lorsque vous documentez des exemples de code, utilisez l'élément `<code>`. Par défaut, le contenu textuel est affiché en police à chasse fixe. Lorsque vous incluez plusieurs lignes de code, imbriquez `<code>` dans un élément `<pre>`, qui représente du texte préformaté.

```
<p>La méthode push() ajoute un ou plusieurs éléments à la fin d'un tableau et renvoie la nouvelle longueur du  tableau.</p>
```


Autres éléments de texte

L'élément **<data>** associe un contenu donné à une traduction lisible par un ordinateur. L'attribut `value` de l'élément fournit la traduction lisible par un ordinateur du contenu de l'élément. Si le contenu **<data>** est lié à l'heure ou à la date, l'élément **<time>**, qui représente une période spécifique, doit être utilisé à la place.

L'élément **<time>** peut inclure l'attribut **`datetime`** pour fournir des dates et des heures lisibles par l'ordinateur. L'attribut **`datetime`** étant exploitable par un ordinateur, il fournit des informations utiles pour des applications telles que les agendas et les moteurs de recherche.

Lorsque vous fournissez un exemple de sortie d'un programme, utilisez l'élément **<samp>** pour délimiter le texte. De manière générale, le navigateur affiche également cet échantillon ou le résultat entre guillemets dans une police à chasse fixe.

Lorsque vous fournissez des instructions via une interaction avec le clavier, vous pouvez utiliser l'élément **<kbd>**. Il représente l'entrée textuelle de l'utilisateur sur un clavier, une saisie vocale ou tout autre périphérique de saisie de texte.

Autres éléments de texte

L'élément **<var>** peut être utilisé pour les expressions mathématiques ou les variables de programmation. La plupart des navigateurs affichent le contenu textuel en italique, dans la police qui l'entoure. Si vous faites beaucoup de mathématiques, envisagez d'utiliser *MathML*, le langage de balisage mathématique basé sur XML pour décrire la notation mathématique.

```
<p>Si <var>a</var> et <var>b</var> sont les côtés et <var>c</var> est l'hypoténuse alors  
<var>a<sup>2</sup></var> + <var>b<sup>2</sup></var> = <var>c<sup>2</sup></var>.</p>
```

La puissance de deux dans le théorème de Pythagore utilisait l'élément en exposant **<sup>**. Il existe un élément d'indice **<sub>** similaire qui spécifie le texte intégré qui doit être affiché en indice pour des raisons typographiques uniquement. Les exposants et les indices sont des nombres, des chiffres, des symboles ou d'autres annotations plus petits que la ligne normale et placés respectivement légèrement au-dessus ou en dessous de la ligne.

Utilisez **** pour indiquer le texte qui a été supprimé. Vous pouvez également inclure le cite défini sur la ressource qui explique la modification et l'attribut **datetime** avec la date ou l'heure au format de date et d'heure lisible par un ordinateur. L'élément barré **<s>** peut être utilisé pour indiquer que le contenu n'est plus pertinent, mais qu'il n'a pas été supprimé du document.

Autres éléments de texte

L'élément **<ins>** est l'opposé de l'élément ****. Il est utilisé pour indiquer le texte qui a été ajouté, ou "inséré", en incluant éventuellement les attributs **cite** ou **datetime**.

Définitions et langues acceptées

Lorsque vous incluez des abréviations ou des acronymes, fournissez toujours la version complète et développée du terme en texte brut dès la première utilisation, car vous introduisez la représentation abrégée du terme entre les balises d'ouverture et de fermeture **<abbr>**, sauf si le terme est bien connu du lecteur, comme "HTML" ou "CSS" dans cette série. **<abbr>** n'est nécessaire que pour cette première occurrence, lorsque l'abréviation ou l'acronyme est défini. L'attribut **title** n'est ni nécessaire, ni utile.

Lorsque vous définissez un terme qui n'est ni une abréviation ni un acronyme, utilisez l'élément de définition **<dfn>** pour identifier le terme défini dans le contenu qui l'entoure.

Si le terme en cours de définition n'est pas dans la même langue que le texte qui l'entoure, veillez à inclure l'attribut **lang** pour identifier la langue.

Autres éléments de texte

Lorsque vous écrivez des langues dans différentes directions, le code HTML fournit l'élément **<bdi>** pour traiter le texte potentiellement bidirectionnel indépendamment du texte qui l'entoure. Cet élément d'internationalisation est particulièrement utile lorsque du contenu dont l'orientation est inconnue est inséré de manière dynamique dans la page. L'élément **<bdo>** remplace l'orientation actuelle du texte et affiche le texte dans une direction différente. Le W3C fournit une introduction aux algorithmes bidirectionnels.

Certains jeux de caractères incluent de petites annotations placées au-dessus ou à droite des caractères pour fournir des informations sur la prononciation. L'élément **<ruby>** est le conteneur à utiliser pour contenir ces annotations, qui facilitent la lecture de langues écrites telles que le coréen, le chinois et le japonais. Le Ruby peut également être utilisé pour l'hébreu, l'arabe et le vietnamien.

La parenthèse rubis (**<rp>**) a été incluse dans la spécification pour contenir des parenthèses d'ouverture et de fermeture pour les navigateurs qui ne sont pas compatibles avec l'affichage de **<ruby>**. Lorsque les navigateurs prennent en charge **<ruby>**, ce qui est le cas de tous les navigateurs permanents, le contenu des éléments **<rp>** n'est pas affiché. L'élément de texte en rubis (**<rt>**) contient les annotations réelles. Ces deux éléments sont imbriqués dans **<ruby>**.

Autres éléments de texte

Mise en valeur du texte

Plusieurs éléments peuvent être utilisés pour accentuer du texte en fonction de la raison sémantique de cette mise en forme (plutôt que pour des raisons de présentation, car il s'agit d'un travail pour CSS).

Utilisez l'élément **** pour mettre en avant une partie du contenu. L'élément **** peut être imbriqué, chaque niveau d'imbrication indiquant un degré d'accentuation plus élevé. Cet élément a une signification sémantique et peut être utilisé pour informer les user-agents auditifs tels que les lecteurs d'écran, Alexa et Siri, afin de le mettre en valeur.

Utilisez l'élément **<mark>** pour identifier ou mettre en surbrillance le texte qui est pertinent, par exemple pour mettre en surbrillance (ou "marquer") l'occurrence de termes de recherche dans les résultats de recherche. Cela permet d'identifier rapidement le contenu marqué sans le mettre en valeur ni le mettre en avant.

L'élément **** identifie le texte comme ayant une grande importance. Les navigateurs affichent généralement le contenu avec une épaisseur de police plus importante.

L'élément **<cite>**, abordé dans les principes de base du texte, est utilisé pour marquer les titres de livres, d'articles ou d'autres œuvres créatives, ou une référence abrégée ou des métadonnées de citation, comme le numéro ISBN d'un livre.

Autres éléments de texte

Trois éléments ont été temporairement obsolètes, mais ont été rajoutés au code HTML. Ils doivent être utilisés avec parcimonie, le cas échéant, car ils n'apportent que peu ou pas de valeur sémantique, et le CSS doit toujours être utilisé pour appliquer des styles aux éléments HTML.

<i>

Les éléments <i> peuvent être utilisés pour des termes techniques, des mots étrangers (là encore, avec l'attribut lang identifiant la langue), des pensées ou des noms de navires. L'élément permet de différencier le contenu intégré du texte qui l'entoure pour une raison spécifique, telle qu'un texte idiomatique, des termes techniques et des désignations taxonomiques. Cet élément ne doit pas être utilisé uniquement pour mettre du texte en italique.

Par défaut, l'élément <i> affiche son style en italique. Dans cet exemple, nous définissons font-style: normal, car les caractères utilisés ne sont pas disponibles en italique.

<u>

L'élément <u> est destiné aux contenus comportant des annotations non textuelles. Par exemple, vous pouvez annoter les mots mal orthographiés en connaissance de cause. Par défaut, le contenu est souligné, mais cela peut être contrôlé avec CSS, par exemple en ajoutant un trait de soulignement rouge ondulé pour imiter les indicateurs d'erreur grammaticale du traitement de texte.

Autres éléments de texte

**** L'élément **** permet d'attirer l'attention sur du texte qui n'est pas important autrement. Cet élément ne transmet aucune information sémantique spéciale et ne doit être utilisé que si aucun des autres éléments de cette section ne remplit l'objectif. Aucun exemple n'est fourni, car je n'ai pas pu trouver de cas d'utilisation valide. C'est ainsi qu'est cet élément de type "dernier recours".

Espaces

Lorsque vous souhaitez ajouter des sauts de ligne, par exemple lors de l'écriture d'un poème ou d'une adresse physique, l'élément de saut de ligne à fermeture automatique, **
**, permet d'ajouter un retour chariot.

<address>

**Machine Learning Workshop
**

**100 Google Drive
**

Mountain View, CA 94040

</address>

Pour ajouter un séparateur ou une pause thématique entre les sections du contenu tangentiel (par exemple, entre les chapitres d'un livre ou entre le monologue de 5 000 mots et la recette recherchée par les utilisateurs), incluez un élément **<hr>**. C'est l'abréviation de « règle horizontale ». Bien que les navigateurs affichent généralement une ligne horizontale, cet élément a une signification sémantique. Le rôle par défaut est separator.

Autres éléments de texte

HTML comporte également un élément qui permet de casser des mots. L'élément `<wbr>` qui se ferme automatiquement indique au navigateur que si un mot est susceptible de dépasser de son conteneur, le navigateur risque de rompre la ligne.

Cette fonction est généralement utilisée pour séparer les mots dans les URL longues. Elle n'ajoute pas de trait d'union.

Les éléments `
`, `<hr>` et `<wbr>` sont tous des éléments vides, ce qui signifie qu'ils ne peuvent avoir aucun nœud enfant (ni des éléments imbriqués, ni du texte).

Aucune balise de fin n'est associée à ces éléments, car aucun "intérieur" ne permet de stocker le contenu.

Détails et résumés



Détails et résumé

Les éléments `<details>` et `<summary>` sont des éléments d'interface utilisateur qui permet de masquer et d'afficher du contenu. Le contexte d'utilisation peut être celui d'une FAQ qui contient une liste de questions visibles. Un clic sur une question permet de déplier la réponse à cette question.

Les éléments `<details>` et `<summary>` ne sont entièrement compatibles avec tous les navigateurs récents que depuis janvier 2020. Vous pouvez désormais créer des widgets fonctionnels, quoique moins attrayants, qu'en utilisant uniquement du code HTML sémantique. Les éléments `<details>` et `<summary>` sont tout ce dont vous avez besoin: il s'agit d'un moyen intégré de gérer l'expansion et le repli du contenu. Lorsqu'un utilisateur clique ou appuie sur une `<summary>`, ou utilise la touche Entrée lorsque `<summary>` est sélectionné, le contenu du `<details>` parent devient visible.

```
<details>
<summary>J'ai des clés mais pas de portes. J'ai de l'espace mais pas de place.
Vous pouvez entrer mais vous ne pouvez pas sortir. Qu'est-ce que je suis ?</summary>
Un clavier.
</details>
```

Détails et résumé

Comme pour tout contenu sémantique, vous pouvez améliorer progressivement les fonctionnalités et l'apparence par défaut. Dans le cas présent, nous n'avons ajouté qu'une petite partie de CSS:

```
body { background: #a4bacc99; color: #226daa; font-family: Arial, sans-serif; margin: 1rem; }
details { border: 1px solid; padding: 0 1rem; background: white; }
details + details { border-top: none; }
details[open] { padding-bottom: 1em; }
summary { padding: 1rem 2em 1rem 0; font-size: 1.25rem; font-weight: bold; cursor: pointer; }
```

Modification de la visibilité: attribut open

L'élément **<details>** est le conteneur du widget d'information. **<summary>** est le résumé ou la légende de son parent **<details>**. Le résumé est toujours affiché et fonctionne comme un bouton qui active ou désactive l'affichage du reste du contenu du parent. L'interaction avec **<summary>** active ou désactive l'affichage des frères et sœurs résumé auto-étiquetés en activant/désactivant l'attribut open de l'élément **<details>**.

Détails et résumé

L'attribut `open` est un attribut booléen. S'il est présent, peu importe sa valeur ou son absence, tous les contenus de `<details>` sont présentés à l'utilisateur. Si l'attribut `open` n'est pas présent, seul le contenu de `<summary>` est affiché. Étant donné que l'attribut `open` est ajouté et supprimé automatiquement lorsque l'utilisateur interagit avec la commande, il peut être utilisé en CSS pour appliquer un style différent à l'élément en fonction de son état.

Vous pouvez créer un accordéon avec une liste de plusieurs éléments `<details>`, chacun avec un enfant `<summary>`. Si vous omettez l'attribut `open` dans votre code HTML, les `<details>` seront tous réduits ou fermés, et seuls les en-têtes de résumé seront visibles au chargement de la page. Chaque titre ouvre le champ du reste du contenu dans l'élément `<details>` parent. Si vous incluez l'attribut `open` dans votre code HTML, `<details>` s'affiche en grand format avec le contenu visible au chargement de la page.

Le contenu masqué à l'état réduit peut faire l'objet de recherches dans certains navigateurs, mais pas dans d'autres, même si le contenu réduit ne fait pas partie du DOM. Si vous effectuez une recherche dans Edge ou Chrome, les détails contenant un terme de recherche sont développés pour afficher l'occurrence. Ce comportement n'est pas reproduit dans Firefox ou Safari.

Détails et résumé

`<summary>` doit être le premier enfant d'un élément `<details>` et doit représenter un résumé, une légende ou une légende pour le reste du contenu de l'élément `<details>` parent dans lequel il est imbriqué. Le contenu de l'élément `<summary>` peut être n'importe quel titre, texte brut ou code HTML pouvant être utilisé dans un paragraphe.

Activer/Désactiver le repère de résumé

Dans les deux précédents exemples, vous noterez la flèche du côté `inline-start` du résumé. Un widget d'affichage est généralement présenté à l'écran sous la forme d'un petit triangle qui pivote (ou se tourne) pour indiquer qu'il est ouvert ou fermé. Un libellé s'affiche à côté du triangle. Le contenu de l'élément `<summary>` définit le widget d'information. La flèche qui pivote en haut de chaque section est un `::marker` défini sur l'élément `<summary>`. Comme les éléments de liste, l'élément `<summary>` accepte la propriété abrégée `list-style` et ses propriétés de longue durée, y compris `list-style-type`. Vous pouvez styliser le triangle d'affichage avec CSS, en remplaçant le repère d'un triangle par tout autre type de puce, y compris une image avec `list-style-image`.

Dialog



Dialog

Une boîte de dialogue modale est un type particulier de fenêtre pop-up sur une page Web: un pop-up qui interrompt l'utilisateur pour qu'il se concentre sur lui-même. Il existe des cas d'utilisation valides pour afficher une boîte de dialogue, mais une grande attention doit être prise au préalable. Les boîtes de dialogue modales obligent les utilisateurs à se concentrer sur un contenu spécifique et, au moins temporairement, à ignorer le reste de la page.

Les boîtes de dialogue peuvent être modales (seul le contenu de la boîte de dialogue peut être interagi) ou non modales (il est toujours possible d'interagir avec le contenu en dehors de la boîte de dialogue). Les boîtes de dialogue modales s'affichent au-dessus du reste du contenu de la page. Le reste de la page est inerte et, par défaut, est masqué par un fond semi-transparent.

L'élément HTML `<dialog>` sémantique permettant de créer une boîte de dialogue est fourni avec la sémantique, les interactions avec le clavier, ainsi que toutes les propriétés et méthodes de l'interface `HTMLDialogElement`.

Dialog

Voici un exemple de `<dialog>`.

```
<dialog id="dialog">
<form action="">
  <button type="submit" aria-label="close" formmethod="dialog" formnovalidate>X</button>
  <h2 id="dialogid">MLW Registration</h2>
  <p>All fields are required</p>
  <p><label>Name: <input type="text" name="name" required /> </label> </p>
  <p><label>Warranty:<input type="number" min="0" max="10" step="1" name="warranty" required /></label></p>
  <p><button type="submit" formmethod="post">Submit</button></p>
<hr/>
<p>Additional buttons</p>
<button id="jsbutton">JS close</button> <button id="reset" type="reset">Reset</button>
  </p>
</form>
</dialog>
<button id="modal">Open Modal dialog</button>
```


Dialog

```
<script>
const dialog = document.getElementById('dialog');
const jsbutton = document.getElementById('jsbutton');
const modal = document.getElementById('modal');
const reset = document.getElementById('reset');

modal.addEventListener('click', (event) => {
  dialog.showModal();
});

jsbutton.addEventListener('click', (event) => {
  dialog.close();
});
</script>
```

```
body {
  background: #a4bacc99;
  color: #226daa;
  font-family: Raleway, sans-serif;
  accent-color: #226DAA;
}

a:hover, a:focus {
  text-underline-offset: 0.25em;
}

[aria-label="close"] {
  appearance: none;
  float: right;
  border: 1px solid;
  border-radius: 50%;
}

dialog :focus {
  outline: 2px solid #226DAA;
}
```

Conclusion



Les nouveautés de CSS3 permettent d'ajouter un peu de vie à vos pages !

Nous avons vu de nouvelles façons de positionner et de modifier des éléments grâce à la propriété transform.

Nous avons ajouté des comportements à nos éléments HTML sans javascript en quelques règles CSS.

Avec les transitions CSS3 vous avez la possibilité de modifier l'apparence et le comportement d'un élément à chaque fois qu'un changement d'état se produit, par exemple quand la souris est placée au-dessus, le focus est mis dessus.

Les animations CSS3 permettent de changer l'aspect et le comportement d'un élément en plusieurs images clés. Les transitions fournissent un changement d'un état à un autre, alors que les animations peuvent définir plusieurs points de transition sur les différentes images clés