Le HTML, abréviation de "HyperText Markup Language" (langage de balisage hypertexte), est le langage standard utilisé pour décrire la structure des documents affichés sur le Web. C'est un langage constitué d'une série d'éléments et d'attributs qui permettent de baliser tous les composants d'un document pour le structurer de manière significative.

Les documents HTML se présentent sous forme d'une hiérarchie de nœuds comprenant des éléments HTML et des nœuds de texte. Les éléments HTML fournissent la signification et la mise en forme des documents, incluant la création de paragraphes, de listes, de tableaux, l'intégration d'images et de formulaires. Chaque élément peut avoir plusieurs attributs spécifiés, et de nombreux éléments peuvent contenir d'autres éléments et du texte.

Il existe plusieurs catégories d'éléments, chacune servant un but spécifique, comme les métadonnées, les sections, le texte, les formulaires, les médias, etc. Mais avant d'explorer ces catégories, comprenons ce qu'est un "élément".

Éléments

Un élément HTML encapsule différentes parties du contenu pour leur donner une signification ou un comportement spécifique. Ils sont définis par des balises, qui sont des symboles tels que `<` et `>`. Par exemple, un titre de niveau 1 est défini par la balise `<h1>`.

Les balises d'ouverture et de fermeture entourent le contenu de l'élément. Il est important de noter que les éléments peuvent être imbriqués, mais ils doivent être correctement fermés dans l'ordre inverse de leur ouverture pour éviter toute erreur.

Balises et Contenus des Éléments HTML

Une balise ouverte `<tag>` est toujours suivie de son homologue fermante `</tag>`. Même si certaines balises peuvent être omises dans certains cas, il est préférable de les inclure pour éviter toute confusion.

Par exemple, dans une liste non ordonnée (`<ul>`), chaque élément (`<li>`) doit être entouré de balises de début et de fin. Omettre ces balises peut entraîner des comportements imprévus du navigateur.

Le HTML est indulgent, ce qui signifie qu'il peut souvent interpréter le code même s'il n'est pas parfaitement formaté. Cependant, il est préférable de respecter les bonnes pratiques pour garantir la cohérence et la compatibilité entre les navigateurs.

Éléments Remplacés et Vides

Les éléments HTML peuvent être remplacés ou vides. Les éléments remplacés sont ceux qui sont substitués par des objets, comme les images ou les éléments d'interface utilisateur. Les éléments vides n'ont pas de contenu interne et peuvent être auto-fermants.

Les attributs jouent un rôle crucial dans la définition du comportement et de l'apparence des éléments. Ils fournissent des informations supplémentaires sur l'élément et sont spécifiés dans la balise d'ouverture.

Apparence des Éléments

L'apparence par défaut des éléments est déterminée par les feuilles de style des navigateurs. Bien que les balises puissent affecter l'apparence, leur principale fonction est de structurer le contenu. Les technologies d'assistance et les moteurs de recherche utilisent la sémantique des balises pour interpréter le contenu de la page.

Éléments, Attributs et JavaScript

Le HTML crée une représentation structurée du document à travers le DOM (Document Object Model). Chaque élément et chaque morceau de texte sont représentés par des objets JavaScript dans le DOM. L'API DOM HTML permet d'accéder et de contrôler ces éléments via JavaScript, offrant ainsi une interaction dynamique avec la page web.

Structure des Documents HTML

Les documents HTML sont la base de la création de pages Web. Ils fournissent une structure et des informations essentielles pour que les navigateurs interprètent et affichent correctement le contenu. Dans cet exposé, nous allons explorer la structure fondamentale d'un document HTML, en mettant l'accent sur les éléments clés qui composent l'en-tête (<head>) du document.

Déclaration de Type de Document et Élément Racine

- Un document HTML commence par une déclaration de type de document (`<!DOCTYPE html>`) pour indiquer au navigateur le type de document qu'il doit interpréter.

- L'élément racine `<html>` enveloppe l'intégralité du contenu du document et définit le contexte de la page.

Langue du Contenu

- L'attribut `lang` de l'élément `<html>` permet de déclarer la langue principale du document, ce qui est crucial pour l'accessibilité et l'interprétation par les moteurs de recherche.

- L'utilisation appropriée de l'attribut `lang` garantit que le contenu est compris et interprété correctement par les lecteurs d'écran et les services de traduction.

En-tête du Document (<head>)

- L'en-tête du document contient des métadonnées importantes telles que le titre, le jeu de caractères, les liens vers les feuilles de style, etc.

- Le titre du document, défini par l'élément `<title>`, est crucial car il s'affiche dans l'onglet du navigateur et les résultats de recherche.

Métadonnées du Document

- Les métadonnées incluent des informations telles que le jeu de caractères (`<meta charset="utf-8" />`), les informations de la fenêtre d'affichage (`<meta name="viewport" content="width=device-width" />`), et les liens alternatifs et canoniques.

Feuilles de Style CSS

- Les feuilles de style définissent l'apparence et la mise en forme du contenu HTML.

- Elles peuvent être incluses à l'aide de l'élément `<link>` ou `<style>`, en interne ou externe au document.

Scripts JavaScript

- Les scripts JavaScript permettent d'ajouter des fonctionnalités interactives et dynamiques à la page.

- Ils peuvent être inclus directement dans le document ou à partir de fichiers externes à l'aide de l'élément `<script>`.

En conclusion, la structure d'un document HTML est essentielle pour assurer la cohérence, l'accessibilité et l'interprétation correcte du contenu par les navigateurs et les utilisateurs. En comprenant les composants clés de l'en-tête HTML, les développeurs peuvent créer des pages Web efficaces et bien structurées.

L'article se concentre sur l'importance des métadonnées dans la structuration des documents HTML, en particulier dans la section `<head>`. Voici un résumé :

Métadonnées

Les métadonnées sont des informations sur les données elles-mêmes. En HTML, elles sont souvent placées dans la section `<head>` du document. Bien que des éléments tels que `<title>`, `<link>`, `<script>`, `<style>`, et `<base>` soient des métadonnées, la balise `<meta>` est spécifiquement conçue pour représenter des métadonnées qui ne peuvent pas être exprimées par d'autres éléments.

Balise `<meta>` obligatoire

Deux balises `<meta>` essentielles ont été récapitulées : la déclaration de jeu de caractères et la balise Meta de la fenêtre d'affichage. La spécification a évolué pour standardiser la façon dont ces balises sont déclarées.

Types de balises `<meta>` définies officiellement

Il existe deux principaux types de balises `<meta>` : les directives pragma et les métatypes nommés. Les directives pragma sont définies par l'attribut `http-equiv`, tandis que les métatypes nommés utilisent l'attribut `name`.

Directives Pragma

Les directives pragma décrivent comment la page doit être analysée. La directive la plus courante est `refresh`, mais elle n'est pas recommandée car elle peut perturber l'expérience utilisateur.

Métatypes nommés

Les métadonnées nommées incluent des informations telles que la description, les mots-clés, les robots, et la couleur du thème. Les mots-clés ne sont plus utiles pour le référencement naturel, mais la description est importante pour le SEO. La balise `<meta name="robots">` contrôle l'indexation des moteurs de recherche, tandis que `<meta name="theme-color">` personnalise l'interface du navigateur.

Open Graph et balises Meta similaires

Les balises Meta Open Graph sont utilisées pour contrôler la façon dont les sites de réseaux sociaux affichent les liens vers votre contenu. Vous pouvez spécifier des titres, descriptions et images personnalisés pour les publications sur Facebook, Twitter, etc.

Autres informations utiles

D'autres balises `<meta>` peuvent être utilisées pour indiquer des informations telles que la compatibilité avec les applications Web, les icônes d'application pour les appareils mobiles, etc.

En résumé, les métadonnées jouent un rôle crucial dans la structuration des documents HTML, permettant un meilleur référencement, une personnalisation de l'interface utilisateur et un contrôle sur la façon dont le contenu est affiché sur les réseaux sociaux.

HTML sémantique

L'article met en lumière l'importance de l'HTML sémantique dans la structuration du contenu web pour améliorer à la fois la lisibilité pour les développeurs et l'accessibilité pour les utilisateurs, en particulier ceux qui dépendent des technologies d'assistance comme les lecteurs d'écran.

L'HTML sémantique consiste à utiliser des balises qui expriment le sens du contenu plutôt que simplement son apparence. En comparant deux extraits de code, l'article illustre la différence entre l'utilisation d'éléments non sémantiques tels que et , et l'utilisation d'éléments sémantiques comme <header>, <main>, <footer>, etc.

En plus de rendre le code plus lisible pour les développeurs, l'HTML sémantique facilite la compréhension du contenu par les outils automatisés, notamment les lecteurs d'écran. L'article explique comment les éléments sémantiques comme <header>, <footer>, etc., fournissent des points de repère clairs pour les utilisateurs de lecteurs d'écran, facilitant ainsi la navigation dans le contenu.

Il souligne également l'importance de l'attribut role dans la définition du rôle d'un élément dans le contexte du document, ce qui est crucial pour l'accessibilité. L'article encourage l'utilisation des éléments sémantiques appropriés pour structurer le contenu, soulignant que le choix des bons éléments dès le départ peut éviter la nécessité de refactoriser le code HTML.

En fin de compte, l'article encourage les développeurs à privilégier l'HTML sémantique dans la structuration du contenu web, non seulement pour faciliter la compréhension du code par les développeurs, mais aussi pour améliorer l'accessibilité pour tous les utilisateurs, quel que soit leur mode de navigation sur le web.

On explore aussi l'importance des en-têtes et des sections dans la structuration du contenu HTML de manière sémantique. Il souligne que le choix des bons éléments de code simplifie la maintenance, facilite la compréhension par les moteurs de recherche et les technologies d'assistance, et améliore l'accessibilité du site.

L'article commence par démontrer l'effet des éléments sémantiques sur la compréhension du contenu, mettant en évidence la différence entre un code non sémantique utilisant des divs et des spans et un code sémantique avec des éléments comme header, nav, et main.

Ensuite, il se concentre sur la création d'un en-tête de site, comparant des approches non sémantiques et sémantiques. Il met en avant l'utilisation d'éléments comme header et nav pour fournir une structure claire et significative au contenu, améliorant ainsi son accessibilité et sa compréhension.

Le rôle des éléments <footer> est également discuté, soulignant leur utilité pour fournir des informations importantes telles que les droits d'auteur et les coordonnées. L'article explique comment les éléments sémantiques comme header, footer, main, aside, article, et section contribuent à une structure logique et accessible du document.

Il met en garde contre l'utilisation abusive des rôles de point de repère, recommandant de les utiliser avec parcimonie pour éviter toute confusion pour les utilisateurs de lecteurs d'écran.

Enfin, l'article présente l'importance des éléments de titre (<h1>-<h6>) pour organiser hiérarchiquement le contenu, soulignant leur impact sur la compréhension du contenu par les lecteurs d'écran. Il conclut en encourageant l'utilisation de la sémantique HTML pour structurer le contenu de manière logique et pertinente.

Cet article explore en profondeur les attributs HTML, soulignant leur importance et leur utilisation dans la création de pages web puissantes et fonctionnelles. Voici un résumé détaillé de chaque section:

Attributs HTML:

Les attributs sont des paires nom/valeur séparées par un espace et apparaissant dans la balise d'ouverture des éléments HTML. Ils fournissent des informations sur l'élément et ses fonctionnalités. Certains attributs sont globaux, s'appliquant à tous les éléments, tandis que d'autres sont spécifiques à un élément particulier.

Attributs booléens:

Les attributs booléens comme autofocus, checked, required, etc., sont des attributs où la simple présence de l'attribut indique que sa valeur est "true". Les valeurs booléennes peuvent être omises ou définies sur une chaîne vide, mais toutes sont interprétées comme "true". Les valeurs non valides sont également remplacées par "true".

Attributs énumérés:

Contrairement aux attributs booléens, les attributs énumérés ont un ensemble limité de valeurs valides prédéfinies. L'omission de la valeur de l'attribut ne signifie pas qu'elle est "true". Les valeurs non valides ne sont pas nécessairement remplacées par une valeur par défaut. Il est important de connaître les attributs booléens et énumérés pour éviter les erreurs de manipulation.

Attributs globaux:

Les attributs globaux peuvent être appliqués à n'importe quel élément HTML, bien que certains n'aient aucun effet sur certains éléments. L'attribut "id" permet de définir un identifiant unique pour un élément, utilisé pour diverses fonctions telles que cibler des éléments pour le CSS ou les scripts.

Utilisations spécifiques des attributs:

Divers exemples sont fournis, notamment l'utilisation de l'attribut "id" pour créer des liens d'ancrage, des sélecteurs CSS, des scripts JavaScript, des étiquettes de formulaire, etc. Chaque utilisation est expliquée en détail avec des exemples de code.

Attributs personnalisés:

Les attributs personnalisés peuvent être créés en ajoutant le préfixe "data-" suivi d'un nom d'attribut personnalisé. Ils permettent de stocker des informations spécifiques à une application et peuvent être accédés via JavaScript à l'aide de la propriété dataset de l'élément.

L'article se termine en abordant deux attributs spécifiques à l'élément image, "target" et "href", ainsi que d'autres attributs spécifiques à l'élément lors de l'examen plus détaillé des liens.

Cet article décrit les principes fondamentaux du balisage du texte en HTML, en mettant en parallèle les fonctionnalités des éditeurs de texte avec les éléments sémantiques et non sémantiques disponibles dans le langage HTML pour structurer le contenu de manière significative et visuelle.

Titres revisités

Les éléments `<h1>` à `<h6>` sont utilisés pour structurer les titres de section, allant du titre principal `<h1>` au titre de niveau inférieur `<h6>`. Bien que les navigateurs n'utilisent pas toujours ces en-têtes pour décrire les documents, ils sont cruciaux pour l'accessibilité des lecteurs d'écran. Les niveaux de titre doivent être hiérarchisés correctement pour faciliter la navigation.

Paragraphes

Les paragraphes sont balisés avec `<p>` en HTML. Bien que la balise de fermeture soit facultative, il est recommandé de l'inclure pour assurer la validité du code et une meilleure compréhension.

Citations

Pour les citations dans un article, les éléments `<blockquote>`, `<q>`, et `<cite>` sont utilisés. Les informations sur l'auteur ou la source de la citation sont généralement placées après la citation dans un élément `<p>`. L'attribut `cite` peut être utilisé pour spécifier l'URL de la source de la citation.

Entités HTML

Les entités HTML sont utilisées pour échapper les caractères spéciaux tels que `<`, `>`, `&`, et `&quot;`. Elles sont nécessaires pour assurer la validité du code HTML. Les espaces insécables sont également utilisés pour empêcher les sauts de ligne indésirables. Les caractères spéciaux peuvent être représentés soit par leur entité nommée, comme `&copy;`, soit par leur équivalent encodé en nombre, comme `&#x1F600;` pour un emoji.

En somme, cet article fournit un guide complet sur la structuration et la présentation du texte en HTML, en mettant l'accent sur l'accessibilité, la validité du code et les bonnes pratiques de codage.

Voici un résumé des points principaux sur les liens HTML:

1. Introduction aux liens:

- Les liens sont essentiels sur le Web et sont créés avec l'élément `<a>`.

- Ils permettent de connecter différentes ressources sur Internet.

2. Attribut `href` :

- L'attribut `href` est utilisé pour spécifier l'adresse de destination du lien.

- Il peut pointer vers différentes destinations telles que des pages Web, des sections de la même page, des adresses e-mail ou des numéros de téléphone.

3. Types de liens:

- Les liens peuvent être absolus (avec un nom de domaine complet) ou relatifs (par rapport au fichier actuel).

- Des liens vers des sections spécifiques d'une page peuvent être créés en utilisant des identifiants de fragments.

- Les liens peuvent également intégrer des adresses e-mail ou des numéros de téléphone pour envoyer des e-mails ou passer des appels.

4. Attribut `download` :

- L'attribut `download` est utilisé pour proposer le téléchargement de fichiers lorsque le lien est cliqué.

- Il spécifie le nom du fichier à télécharger sur le système de fichiers local de l'utilisateur.

5. Attribut `target`:

- L'attribut `target` contrôle le contexte de navigation du lien.

- Les valeurs courantes incluent `_self`, `_blank`, `_parent` et `_top`, qui déterminent où s'ouvrira le lien.

6. Attribut `rel`:

- L'attribut `rel` définit la relation entre le document actuel et la ressource liée par le lien.

- Il peut être utilisé pour indiquer si le lien est externe, s'il fournit une aide contextuelle, ou s'il renvoie à des documents précédents ou suivants dans une série.

7. Suivi des clics sur les liens:

- L'attribut `ping` peut être utilisé pour notifier des URL lorsque les liens sont activés, permettant de suivre les interactions des utilisateurs.

8. Conseils sur l'expérience utilisateur:

- Les liens doivent être descriptifs et différenciés visuellement du texte environnant.

- Ils doivent fournir des informations claires sur leur destination.

- Les interactions utilisateur doivent être prises en compte pour une meilleure expérience de navigation.

9. Liens et JavaScript:

- JavaScript peut être utilisé pour manipuler les liens HTML, par exemple pour changer la destination URL.

L'article met également en avant l'importance de bien concevoir les liens pour améliorer l'expérience utilisateur et garantir la compréhension et l'accessibilité du contenu.

Les listes

L'article porte sur les différentes utilisations et balisages des listes en HTML, en mettant en évidence les listes non ordonnées, les listes ordonnées et les listes de descriptions.

Les listes sont omniprésentes dans le développement web, souvent utilisées pour structurer et organiser l'information de manière logique. Par exemple, lors de la création d'une liste de courses, d'une liste de tâches ou même de questions à choix multiples dans un test, on utilise des listes.

En HTML, il existe trois types principaux de listes :

1. Les listes non ordonnées `<ul>` : Utilisées lorsque l'ordre des éléments n'a pas d'importance. Chaque élément est précédé d'une puce par défaut. On les ferme avec `</ul>`.

2. Les listes ordonnées `<ol>` : Utilisées lorsque l'ordre des éléments est important, chaque élément étant numéroté. Les attributs spécifiques comme `type`, `reversed`, et `start` peuvent être définis pour contrôler le comportement de la numérotation.

3. Les listes de descriptions `<dl>` : Utilisées pour créer des paires de termes et de descriptions. Elles se composent de termes `<dt>` et de leurs descriptions `<dd>`.

Les éléments de liste `<li>` sont utilisés pour définir les éléments individuels à l'intérieur des listes `<ul>` et `<ol>`. Ils peuvent également être utilisés dans les listes de descriptions `<dl>`.

L'article explore également des exemples concrets, comme la création de listes d'enseignants et d'avis dans une application fictive appelée MLW (Machine Learning World). Il montre comment imbriquer des listes et utiliser des éléments `<img>`, `<blockquote>`, et `<p>` pour enrichir le contenu des listes.

Enfin, l'article souligne l'importance de fermer correctement les balises `<li>` pour assurer la lisibilité du code et discute de l'ajout éventuel d'attributs ARIA pour améliorer l'accessibilité des listes.

En somme, l'article offre un aperçu complet de l'utilisation des listes en HTML, couvrant leur syntaxe, leurs attributs et leurs bonnes pratiques.

Navigation

L'article aborde différents aspects de la navigation sur les sites web, en mettant l'accent sur les bonnes pratiques et les techniques pour améliorer l'accessibilité et l'expérience utilisateur.

Navigation Locale

La navigation locale concerne les liens qui guident les utilisateurs à l'intérieur du site web. Elle peut inclure des éléments tels que les fils d'Ariane, les barres latérales, ou les menus déroulants. Ces éléments aident les utilisateurs à se repérer sur le site et à accéder facilement à différentes sections ou pages.

Fils d'Ariane

Les fils d'Ariane fournissent une navigation secondaire en indiquant la hiérarchie des pages dans le site web. Ils permettent aux utilisateurs de savoir où ils se trouvent et de naviguer rapidement vers des sections spécifiques. Les fils d'Ariane peuvent être composés de liens vers la page d'accueil, les répertoires, et la page actuelle.

Navigation Globale

La navigation globale offre un moyen uniforme de se déplacer sur tout le site web. Elle comprend généralement des liens vers les pages principales du site et peut être présentée sous forme de barres de navigation, de menus déroulants, ou d'onglets. La navigation globale doit être cohérente sur toutes les pages du site.

L'article met en avant l'importance d'une navigation intuitive et simple pour garantir une expérience utilisateur optimale. Il donne également des conseils sur l'accessibilité, en soulignant l'importance de fournir des alternatives textuelles pour les éléments visuels et en expliquant l'utilisation des attributs ARIA pour améliorer la navigation pour les utilisateurs de lecteurs d'écran.

En résumé, l'article offre un aperçu complet des différentes techniques de navigation sur les sites web, en mettant en avant les bonnes pratiques pour garantir une expérience utilisateur fluide et accessible.

Les tableaux

Les tableaux HTML sont des outils pour afficher des données tabulaires avec des lignes et des colonnes. Le choix d'utiliser un tableau doit être basé sur le contenu et les besoins des utilisateurs. S'il s'agit de présenter, comparer, trier ou calculer des données, alors <table> est approprié. Cependant, si vous voulez simplement organiser le contenu non tabulaire, comme des vignettes, les tableaux ne sont pas appropriés. Dans ce cas, une liste d'images avec stylisation CSS est préférable.

Éléments du tableau

- <table>: Encapsule tout le contenu du tableau.

- <caption>: Donne un nom à une table pour une meilleure compréhension.

- <colgroup>: Contient des groupes de colonnes dans le tableau.

- <col>: Définit une colonne dans un groupe de colonnes.

- <thead>, <tbody>, <tfoot>: Sections facultatives du tableau pour les en-têtes, le corps et le pied de page, respectivement.

- <tr>: Ligne du tableau.

- <th>: Cellule d'en-tête.

- <td>: Cellule de données.

Légende du tableau

L'élément <caption> donne un titre descriptif à une table, favorisant l'accessibilité et la compréhension du contenu.

Section de données

Les tableaux peuvent inclure des sections facultatives pour organiser les données en en-têtes, corps et pieds de page, améliorant la convivialité.

Contenu de la table

Les lignes du tableau contiennent des cellules, utilisant <th> pour les en-têtes et <td> pour les données.

Fusion des cellules

Il est possible de fusionner plusieurs cellules dans une seule cellule à l'aide des attributs colspan et rowspan.

Appliquer un style à des tableaux

Les éléments <colgroup> et <col> permettent de définir des styles de colonne, bien que leur utilisation soit limitée.

Présenter des données

Utilisez des tableaux uniquement pour des données tabulaires, pas pour la présentation. Pour les formulaires, utilisez les éléments de formulaire appropriés plutôt que des tableaux.

En résumé, cette partie met en avant l'importance de l'accessibilité, de la sémantique et de l'utilisation appropriée des tableaux pour présenter des données de manière claire et compréhensible. Il fournit des conseils sur la structuration, la conception et le style des tableaux en HTML pour améliorer l'expérience utilisateur.

Les formulaires

Cette partie met en lumière l'importance des formulaires dans la conception de sites Web et d'applications, soulignant qu'ils constituent un élément omniprésent dans la plupart des plateformes en ligne. Même des sites comme MachineLearningWorkshop.com, à l'origine conçus comme des farces pour le 1er avril, intègrent des formulaires pour des interactions utilisateur, même si ceux-ci peuvent être fictifs.

La pièce maîtresse des formulaires réside dans la compréhension de l'élément HTML `<form>`, qui encapsule les commandes interactives permettant l'envoi d'informations. Elle met en avant la puissance du langage HTML dans la création de formulaires, sans nécessairement recourir à JavaScript. Il souligne également l'importance des attributs HTML dans la validation des données du formulaire côté client, offrant ainsi une expérience utilisateur fluide et évitant les soumissions erronées.

Elle aborde également les différents types de méthodes HTTP pour l'envoi des données du formulaire, mettant en évidence les différences entre GET et POST, avec des recommandations sur le choix de la méthode appropriée en fonction du type de données à envoyer.

En outre, il traite des éléments tels que les boutons d'envoi de formulaire, la manipulation des données après l'envoi du formulaire, les options de sélection, les cases à cocher et leur utilisation appropriée, ainsi que l'association des libellés aux commandes de formulaire pour une accessibilité et une convivialité accrues.

En somme, la partie offre un aperçu complet de la création et de la gestion des formulaires HTML, mettant en lumière leur importance dans l'interaction utilisateur et offrant des conseils pratiques pour une implémentation efficace.

Les images

Cette partie donne un aperçu complet de l'utilisation des images en HTML, en couvrant divers aspects tels que l'inclusion d'images, les méthodes pour rendre les images réactives, la rédaction de descriptions alternatives efficaces, et d'autres fonctionnalités et bonnes pratiques.

Il commence par expliquer que les images décoratives doivent être appliquées en CSS, tandis que celles qui ajoutent du contexte au document doivent être intégrées au code HTML à l'aide de la balise `<img>` avec l'attribut `src` pour référencer la ressource image et l'attribut `alt` pour décrire l'image.

Ensuite, couvre les méthodes d'inclusion de plusieurs sources d'images avec les attributs `srcset` et `sizes` pour rendre les images réactives en fonction de la résolution de l'appareil et de la taille de la fenêtre d'affichage. Il explique également l'utilisation de l'élément `<picture>` avec des enfants `<source>` pour le même but.

Elle souligne l'importance de rédiger des descriptions alternatives (attribut `alt`) concises mais informatives pour les utilisateurs qui ne peuvent pas voir l'écran, en mettant l'accent sur la transmission des informations pertinentes de l'image par rapport au contexte du document.

En outre, mentionne d'autres fonctionnalités comme le chargement différé des images, la spécification de dimensions avec les attributs `width` et `height`, et d'autres attributs moins couramment utilisés comme `crossorigin`, `decoding`, etc.

Enfin, elle donne des conseils spécifiques sur la rédaction de descriptions alternatives en fonction du contexte de l'image, encourageant la concision et la pertinence.

Dans l'ensemble, cette partie fournit une ressource complète pour comprendre et utiliser efficacement les images en HTML, en mettant en lumière les meilleures pratiques pour l'accessibilité et les performances.

Audio et videos

Cette partie aborde la manière d'intégrer et de contrôler des fichiers audio et vidéo dans une page web en utilisant HTML. Voici un résumé :

Intégration de fichiers audio et vidéo

Les éléments `<audio>` et `<video>` permettent d'intégrer des fichiers multimédias dans une page web. L'attribut `src` peut être utilisé pour spécifier directement la source du fichier ou plusieurs éléments `<source>` peuvent être inclus pour donner des suggestions de fichiers sources. Il est préférable d'utiliser `<audio>` pour les fichiers audio.

Contrôles utilisateur

Les attributs tels que `controls`, `autoplay`, `loop`, `mute` et `preload` permettent de contrôler le comportement de la lecture. L'élément `<video>` accepte également les attributs `height`, `width` et `poster` pour spécifier la taille et l'image de couverture de la vidéo.

Sous-titres et accessibilité

Les éléments `<track>` sont utilisés pour spécifier des pistes de texte temporisées, comme des sous-titres. Les fichiers de suivi doivent être au format WebVTT (.vtt). Il existe différentes valeurs pour l'attribut `kind` de `<track>` selon le type de contenu texte, comme les sous-titres, les descriptions, etc.

Bonnes pratiques et accessibilité

Il est recommandé d'inclure des sous-titres au format WebVTT pour rendre les vidéos accessibles aux personnes malentendantes. Les vidéos en arrière-plan, bien que populaires pour des raisons esthétiques, doivent être utilisées avec précaution car elles ne sont pas accessibles par défaut.

L'utilisation de commandes multimédias personnalisées est également abordée pour offrir une meilleure expérience utilisateur.

La connaissance de HTMLMediaElement ainsi que d'autres API liées aux médias permet de créer des expériences multimédias riches et accessibles sur le web.

Modele, emplacement et ombre

Cette partie met en lumière l'importance de l'accessibilité dans la conception de sites web et offre des conseils pratiques pour intégrer des fichiers audio et vidéo de manière accessible et conviviale pour tous les utilisateurs.

Elle porte sur la création de composants web réutilisables en utilisant les modèles HTML, les éléments personnalisés et le Shadow DOM. Voici un résumé de ses points principaux :

1. Introduction aux composants web : Les composants web permettent de créer des widgets UI réutilisables. Ils sont composés de modèles HTML, d'éléments personnalisés et du Shadow DOM, ce qui les rend facilement intégrables dans d'autres applications web.

2. Création d'un élément personnalisé : Pour créer un composant web, on commence par définir un élément personnalisé en utilisant `customElements.define`. Il est recommandé d'utiliser un nom en minuscules avec un tiret pour distinguer les éléments personnalisés des éléments standards.

3. Utilisation de l'élément `<template>`: L'élément `<template>` permet de déclarer des fragments de HTML à cloner et à insérer dans le DOM avec JavaScript. Son contenu n'est pas affiché par défaut, mais il peut être instancié avec JavaScript.

4. Utilisation de l'élément `<slot>`: L'élément `<slot>` permet d'inclure un emplacement pour du contenu personnalisé à l'intérieur d'un élément personnalisé. Il crée un "emplacement nommé" qui peut être rempli avec du contenu lors de l'utilisation de l'élément personnalisé.

5. Définition du Shadow DOM : Le Shadow DOM définit des styles CSS et isole le contenu d'un composant web du reste du document. Cela signifie que les styles externes ne s'appliquent pas au composant et que les styles du composant n'affectent pas le reste du document.

6. Stylisation des composants web : Les styles CSS pour les composants web sont encapsulés dans le Shadow DOM. Cela permet de créer des styles spécifiques au composant sans affecter le reste de la

page. La pseudo-classe `:host` et le pseudo-élément `::slotted()` sont utilisés pour cibler les éléments du composant.

En résumé, la partie explique comment créer des composants web réutilisables en utilisant les modèles HTML, les éléments personnalisés et le Shadow DOM, tout en fournissant des instructions sur la manière de les styliser de manière encapsulée pour éviter les collisions avec d'autres styles de la page.

Les API HTML

Cette partie explore les API HTML et leurs utilisations pour interroger et manipuler les éléments du Document Object Model (DOM). Voici un résumé des points principaux :

1. DOM et AOM : Le DOM (Document Object Model) est une API permettant d'accéder et de manipuler les documents HTML. Il représente l'arborescence de tous les nœuds du document, comprenant des éléments, des attributs et des nœuds de texte. L'AOM (Accessibility Object Model) est basé sur le DOM et représente une arborescence d'accessibilité.

2. API HTML Element : Chaque nœud de l'arborescence du document est un objet manipulable avec JavaScript. Les navigateurs fournissent de nombreuses API permettant de manipuler les nœuds HTML, y compris des méthodes, des événements et des propriétés pour interroger et mettre à jour les éléments.

3. Interfaces des éléments disponibles: Il existe de nombreuses interfaces HTML spécifiques à chaque élément HTML, telles que HTMLAnchorElement pour `<a>`, HTMLImageElement pour `<img>`, etc. Ces interfaces fournissent des méthodes et des propriétés spécifiques à chaque type d'élément.

4. Interface Node : Tous les types de nœuds DOM sont représentés par une interface basée sur Node, qui fournit des informations et des méthodes pour interroger et ajouter des nœuds à l'arborescence DOM. Par exemple, la méthode `appendChild()` est utilisée pour ajouter un nœud enfant à un nœud parent.

5. Interfaces Document et HTMLDocument** : L'interface Document représente la page web chargée dans le navigateur, tandis que HTMLDocument fournit des informations spécifiques au document HTML telles que l'URL, la date de dernière modification, etc.

6. Interface Window : L'interface Window représente la fenêtre du navigateur dans laquelle le script s'exécute. Elle fournit des fonctionnalités pour manipuler la fenêtre du navigateur et accéder aux informations sur l'appareil, telles que la taille de l'écran et le rapport de pixels de l'appareil.

En résumé, les API HTML offrent une gamme de fonctionnalités pour interagir avec les éléments HTML d'une page web, que ce soit pour les interroger, les manipuler ou accéder à des informations sur le document et la fenêtre du navigateur.

Concentration

Le texte traite de la concentration, en mettant en lumière les éléments interactifs et la manière de les rendre accessibles et utilisables pour les utilisateurs, en particulier ceux qui naviguent sur le web avec un clavier. Voici un résumé des points clés :

1. Ordre de tabulation et d'accessibilité: Les éléments interactifs sont normalement accessibles via la touche de tabulation, mais il est possible de les rendre inertes ou interactifs en utilisant des attributs HTML appropriés.

2. Attribut tabindex : L'attribut `tabindex` permet de modifier l'ordre de tabulation des éléments. Une valeur négative rend l'élément sélectionnable mais pas tabulable, tandis qu'une valeur de 0 ou plus le rend tabulable. Cependant, modifier l'ordre de tabulation peut perturber l'expérience utilisateur.

3. Attribut contenteditable : Cet attribut permet de rendre un élément modifiable et sélectionnable, le plaçant dans l'ordre de tabulation.

4. Attribut autofocus : Cet attribut sélectionne automatiquement le premier élément sélectionnable lors du chargement de la page, mais son utilisation doit être prudente pour éviter de perturber l'expérience utilisateur.

5. Éléments désactivés : Les attributs `disabled` et `inert` permettent de rendre les éléments non sélectionnables et d'indiquer visuellement qu'ils sont inactifs. Cependant, ils ne sont pas interchangeables et leur utilisation doit être appropriée pour ne pas confondre les utilisateurs.

En résumé, le texte met en garde contre la modification de l'ordre de tabulation et souligne l'importance de rendre les éléments interactifs accessibles et utilisables pour tous les utilisateurs, en particulier ceux qui dépendent de la navigation au clavier ou des technologies d'assistance.

Autres elements de texte integrés

Nous avons couvert la plupart des éléments HTML, mais il en reste encore à explorer, notamment les éléments de texte intégrés. À l'origine, le HTML était conçu pour partager des documents, pas pour diffuser des vidéos de chats. De nombreux éléments HTML fournissent une sémantique utile pour la documentation. Parmi eux, nous trouvons les liens avec l'élément `<a>`, ainsi que d'autres éléments présentés brièvement ici.

Lorsque vous documentez des exemples de code, utilisez l'élément `<code>`. Le contenu textuel est affiché en police à chasse fixe par défaut. Pour des blocs de code plus longs, placez `<code>` à l'intérieur de `<pre>` pour représenter du texte préformaté.

L'élément `<data>` associe un contenu à une traduction lisible par ordinateur. Utilisez l'attribut `value` pour fournir la traduction. Pour les dates ou heures, utilisez `<time>` avec l'attribut `datetime` pour les ordinateurs.

Pour afficher des résultats de programme, utilisez `<samp>`. Pour des saisies clavier, utilisez `<kbd>`. `<var>` est utile pour les expressions mathématiques ou les variables de programmation.

Utilisez `<sup>` pour les exposants et `<sub>` pour les indices. `<del>` indique du texte supprimé, tandis que `<ins>` montre du texte ajouté. Utilisez `<s>` pour du texte barré.

Pour définir des abréviations ou des acronymes, utilisez `<abbr>` pour la première utilisation. `<dfn>` sert à définir un terme, en spécifiant la langue avec l'attribut `lang` si nécessaire.

Utilisez `<bdi>` pour le texte bidirectionnel, `<bdo>` pour changer l'orientation du texte. `<ruby>` est utilisé pour des annotations sur des caractères dans des langues comme le chinois ou le japonais.

Pour mettre en valeur du texte, utilisez `<em>` pour l'emphase, `<mark>` pour le surlignage, `<strong>` pour l'importance, et `<cite>` pour les références.

Évitez `<i>`, `<u>`, et `<b>` sauf si vraiment nécessaires. Utilisez `<br>` pour les sauts de ligne, `<hr>` pour les séparateurs thématiques, et `<wbr>` pour les coupures de mots.