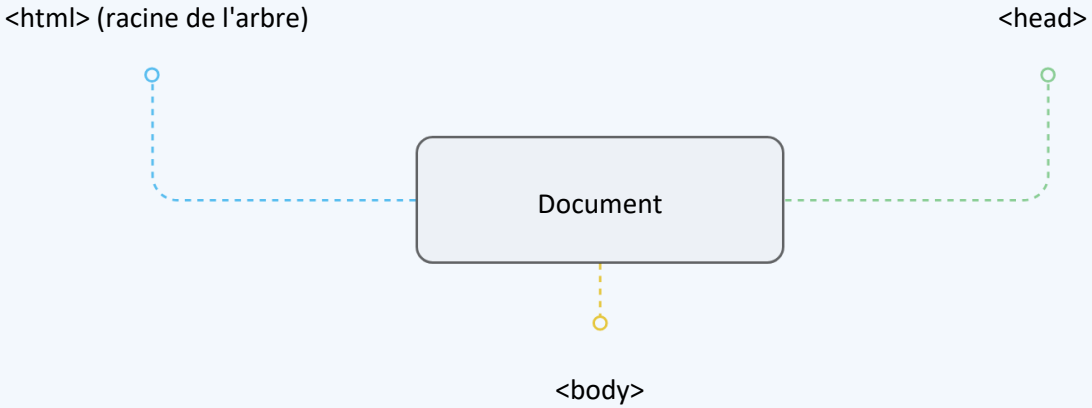
Introduction au DOM : Qu'est-ce que le Document Object Model ?

Explorez comment le DOM transforme un simple document HTML en une page web interactive et dynamique. Ce cours vous guidera des bases aux techniques avancées.

Le DOM : Une Représentation Structurée du Document

Imaginez votre page web comme un arbre généalogique. Le Document Object Model (DOM) est une représentation de cette page sous forme cette page sous forme d'une structure arborescente. Chaque élément HTML (comme un titre, un paragraphe ou une image) est une "branche" est une "branche" ou une "feuille" de cet arbre, appelée un "nœud".



Cette structure nous permet de naviguer et de manipuler chaque partie de la page avec un langage de script comme JavaScript. JavaScript.

Accéder aux Éléments du DOM : Premiers Pas

Pour manipuler un élément, il faut d'abord le trouver ! JavaScript nous offre plusieurs "chercheurs" pour sélectionner des nœuds spécifiques dans l'arbre du DOM.

1

getElementById('id')

La méthode la plus rapide pour trouver un élément unique grâce à son attribut `id`.

2

querySelector('sélecteur')

Un outil polyvalent qui trouve le premier élément correspondant à un sélecteur CSS (ex: '.maclasse', '#monid', 'p').

3

querySelectorAll('sélecteur')

Similaire à querySelector, mais renvoie tous les éléments correspondants sous forme de liste.

Manipuler le DOM : Modifier le Contenu

Une fois un élément sélectionné, vous pouvez facilement changer ce qu'il contient.

Propriétés Clés

- `textContent` : Modifie ou récupère le contenu textuel brut d'un élément, en ignorant le HTML.
- `innerHTML` : Modifie ou récupère le contenu HTML complet à complet à l'intérieur d'un élément. Attention à la sécurité avec sécurité avec cette méthode !

Exemple de Code

```
// HTML: <h1 id="titre">Bonjour</h1>

const monTitre = document.getElementById('titre');

// Change le texte
monTitre.textContent = 'Bonjour le Monde !';
```

Manipuler le DOM : Modifier la Structure

Au-delà du contenu, vous pouvez réorganiser l'arbre du DOM en déplaçant, ajoutant ou supprimant des éléments.

Méthodes Fondamentales

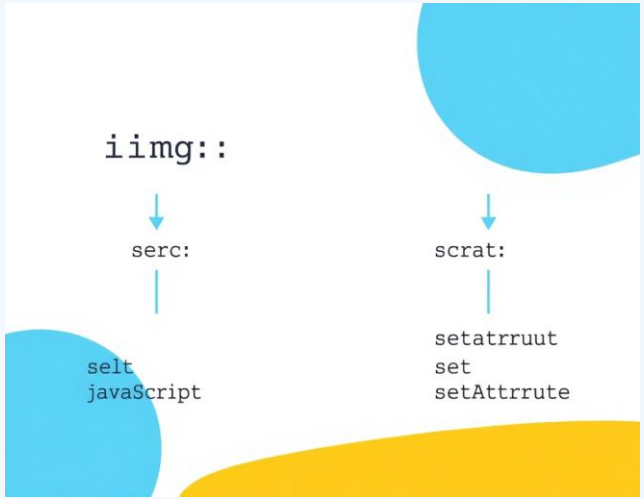
- `createElement()` : Crée un tout nouvel élément (nœud). (nœud).
- `appendChild()` : Ajoute un élément comme dernier enfant d'un autre élément.
- `removeChild()` : Supprime un élément enfant d'un élément parent.

Exemple : Ajouter un paragraphe

```
// HTML: <div id="conteneur"></div>

const conteneur = document.getElementById('conteneur');
const nouveauParagraphe = document.createElement('p');
nouveauParagraphe.textContent = 'Ceci est un nouveau paragraphe.';
conteneur.appendChild(nouveauParagraphe);
```

Les Attributs du DOM : Lecture et Modification



Les attributs HTML (comme `src` pour une image ou `href` pour un lien) sont aussi accessibles et modifiables via le DOM.

getAttribute('nom')

Permet de lire la valeur d'un attribut. Par exemple, `image.getAttribute('src')`.

setAttribute('nom', 'valeur')

Permet de définir ou de modifier la valeur d'un attribut. Par exemple, `lien.setAttribute('href', 'https://nouvelle-url.com')`.

removeAttribute('nom')

Supprime complètement un attribut d'un élément.

Événements DOM : Interagir avec l'Utilisateur

Les événements sont des actions qui se produisent dans le navigateur, comme un clic de souris, une pression de touche ou le chargement de la page. Vous pouvez "écouter" ces événements pour déclencher du code JavaScript.

La méthode `addEventListener`

C'est la manière moderne et la plus recommandée pour gérer les événements. Elle vous permet d'attacher une fonction à un événement sur un élément spécifique.

`element.addEventListener('typeEvenement', fonctionADeclencher);`

- `click` : Déclenché lors d'un clic de souris.
- `mouseover` : Quand le curseur de la souris passe sur l'élément.
- `keydown` : Quand une touche du clavier est pressée.

Manipuler les Classes CSS

Changer le style d'un élément est souvent fait en ajoutant ou en retirant des classes CSS. La propriété `classList` rend cela très simple.
très simple.

Ajouter une classe

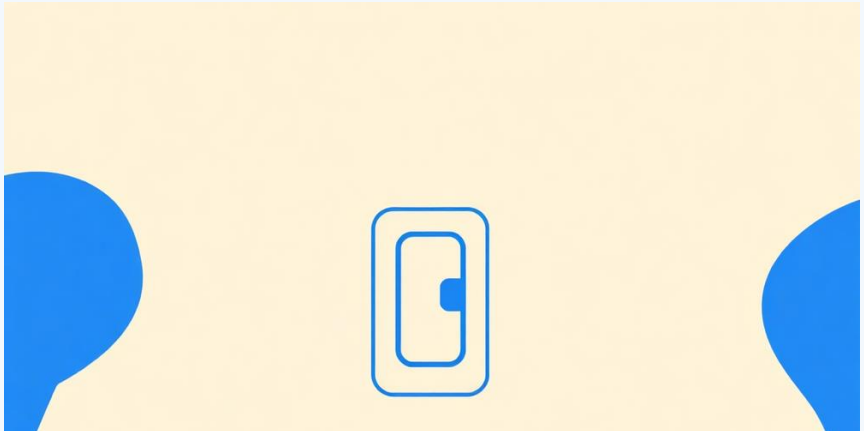
`element.classList.add('ma-classe');`

Retirer une classe

`element.classList.remove('ma-classe');`

Alterner une classe

`element.classList.toggle('ma-classe');`
(l'ajoute si absente, la retire si présente)



Performances et Optimisation

Des manipulations intensives du DOM peuvent ralentir votre page. Voici quelques principes pour garder une application fluide.

- Minimiser les accès au DOM : Chaque interaction avec le DOM a un coût. Stockez les éléments fréquemment utilisés dans des variables dans des variables (`const monElement = document.getElementById('id');`).
- Modifier les éléments hors du DOM : Si vous devez faire de nombreuses modifications, retirez l'élément du DOM, modifiez-le, puis réinsérez-le.
- Éviter les "Reflows" : Changer la géométrie d'un élément (largeur, hauteur) est coûteux. Essayez de grouper les modifications de style. modifications de style.
- Utiliser les `DocumentFragment` pour les ajouts en masse (voir diapositive suivante).

Concepts Avancés : Fragments et Shadow DOM

DocumentFragment

Un `DocumentFragment` est un conteneur de nœuds DOM "allégé" DOM "allégé" et sans parent. Vous pouvez y ajouter de multiples multiples éléments, puis ajouter le fragment entier au DOM en une DOM en une seule opération. C'est idéal pour la performance lors performance lors de la construction de listes ou de tables complexes.

Shadow DOM

Le Shadow DOM permet d'encapsuler une partie de l'arbre DOM. Le style et le script à l'intérieur du Shadow DOM sont isolés du reste de la page. C'est la technologie qui alimente les Web Components et évite les conflits de style CSS.

Bonnes Pratiques et Outils

Soyez Spécifique

Utilisez les sélecteurs les plus précis possibles pour améliorer les performances (`getElementById` est le plus rapide).

Mettez en Cache

Ne sélectionnez pas le même élément plusieurs fois. Stockez-le dans une variable.

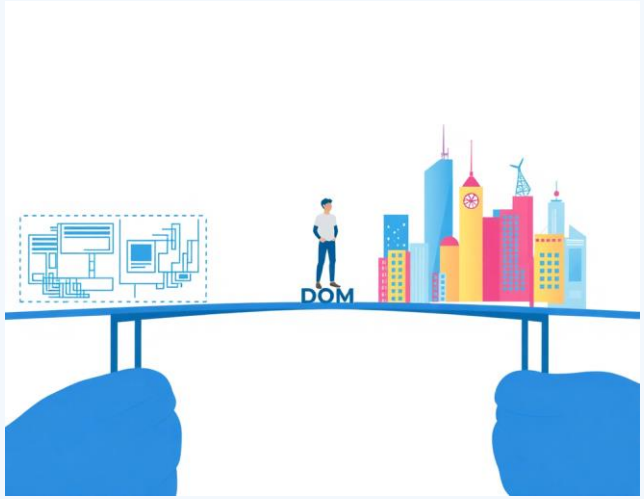
Utilisez les Outils de Développement

L'inspecteur d'éléments de de votre navigateur est votre meilleur ami pour explorer et déboguer le DOM en temps réel.

Commentez votre Code

Expliquez pourquoi vous manipulez le DOM, surtout surtout pour les logiques complexes.

Conclusion : Maîtriser le DOM pour des Pages Web Dynamiques



Le DOM est le pont entre votre code HTML et votre logique JavaScript. Le comprendre et le maîtriser est la compétence fondamentale pour transformer des pages statiques en expériences utilisateur riches et interactives.

Points Clés à Retenir

- Le DOM est un arbre de nœuds représentant votre HTML.
- Sélectionnez des éléments avec `querySelector` ou `getElementById`. `getElementById`.
- Modifiez le contenu, les attributs et les styles avec les méthodes appropriées.
- Utilisez `addEventListener` pour réagir aux actions de l'utilisateur.
- Pensez à la performance en minimisant les manipulations directes du DOM.