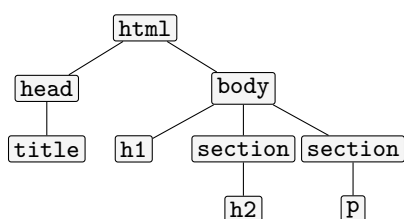


HTML DOM

Fiche de synthèse

① L'ARBRE D'UNE PAGE

Définition 1 (DOM) Le DOM, modèle objet de document, est l'**arbre** que le navigateur construit en mémoire à partir d'une page HTML : chaque balise, chaque texte et chaque attribut y devient un objet, un **nœud**. Un programme JavaScript lit et modifie la page à travers cet arbre.



Les nœuds sont des **éléments**, un par balise, des **textes**, écrits entre les balises, et des **attributs**, rattachés à leur élément. `html` est la **racine**; `head` et `body` sont ses **enfants**; `body` est le **parent** de `h1`; les deux `section` sont **frères**.

② JAVASCRIPT EN BREF

JavaScript est le langage des navigateurs. Un script s'écrit entre `<script>` et `</script>`, en fin de `body`, pour que les éléments qu'il manipule existent déjà.

Python	JavaScript
<code>x = 0</code>	<code>let x = 0;</code>
<code>TAUX = 20</code>	<code>const TAUX = 20;</code>
<code>True, False</code>	<code>true, false</code>
<code>a == b, a != b</code>	<code>a === b, a !== b</code>
<code>and, or, not</code>	<code>&&, , !</code>
<code>for x in liste:</code>	<code>for (const x of liste) {...}</code>
<code>def f(a):</code>	<code>function f(a) {...}</code>
<code>print(x)</code>	<code>console.log(x);</code>

Les blocs sont délimités par des accolades, chaque instruction finit par un point-virgule. `let` déclare une variable qui change, `const` une variable jamais réaffectée. `===` compare sans conversion : `"1" === 1` vaut `false`, alors que `"1" == 1` vaut `true`.

③ ATTEINDRE ET LIRE LES ÉLÉMENTS

- `document.getElementById("intro")` : l'élément d'identifiant `intro`.
- `document.querySelector("nav a")` : le premier élément qui correspond au sélecteur CSS.

- `document.querySelectorAll("p")` : tous ces éléments, parcourus avec `for ... of`.
- `elt.textContent`, le texte; `elt.innerHTML`, le code HTML; `elt.getAttribute("href")`, un attribut; `elt.value`, le contenu d'un champ.
- `elt.children`, `elt.parentNode`, `elt.nextElementSibling` : les enfants, le parent, le frère suivant.

④ MODIFIER LA PAGE

```

<ul id="liste"><li>Pain</li></ul>
<input type="text" id="art">
<script>
  function ajouter() {
    const li = document.createElement("li");
    const c = document.getElementById("art");
    li.textContent = c.value;
    const ul = document.getElementById("liste");
    ul.appendChild(li);
  }
</script>

```

Toute modification s'affiche aussitôt. On remplace le texte par `textContent`, un attribut par `setAttribute`, le style par `style`; `createElement` crée un élément, `appendChild` l'insère, `remove` le retire.

⑤ RÉAGIR AUX ÉVÉNEMENTS

```

<input type="text" id="prenom">
<p id="salut"></p>
<script>
  const c = document.getElementById("prenom");
  c.addEventListener("input", function (event) {
    const p = document.getElementById("salut");
    p.textContent = "Bonjour " + c.value;
  });
</script>

```

Un **événement** est un fait qui se produit dans la page; un **gestionnaire** est la fonction que le navigateur exécute alors. Le paramètre `event` décrit l'événement; `event.target` est l'élément concerné. Événements courants : `click`, `input`, `change`, `submit`, `keydown`, `mouseover`, `load`.

📌 À retenir.

- Le DOM est l'arbre de la page; `getElementById` et `querySelector` en atteignent les éléments.
- `addEventListener` associe une fonction à un événement.