Objectif du module

À la fin de ce module, l'étudiant sera capable de :

- Sélectionner efficacement des éléments HTML
- Modifier dynamiquement le contenu et l'apparence d'une page
- Réagir aux actions utilisateur (clic, saisie, formulaire, clavier)
- Comprendre le rôle de preventDefault()
- Transformer un mockup statique en interface interactive simple

Table des matières

1. DOM

Qu'est-ce que le DOM ?

Le **DOM** (*Document Object Model*) est une **représentation structurée de ta page web** que le navigateur crée en lisant ton HTML.

On peut imaginer la page comme un **arbre de nœuds** : chaque balise HTML devient un objet que JavaScript peut lire et modifier.

- **Document** → c'est la page web chargée dans le navigateur.
- **Object Model** → les éléments HTML (titre, paragraphes, images, boutons…) sont représentés comme des **objets** que l'on peut manipuler par programmation.
- Grâce à ce modèle, **JavaScript peut lire, modifier, ajouter ou supprimer des éléments, changer du texte ou des styles, et réagir aux actions de l'utilisateur**.

En pratique, lorsqu'un navigateur charge ta page HTML, il construit ce modèle DOM :

```
<body>
  <header>
    <h1>Mon site</h1>
  </header>

  <main>
    <p>Bienvenue !</p>
    <button>Cliquer</button>
  </main>
</body>
```

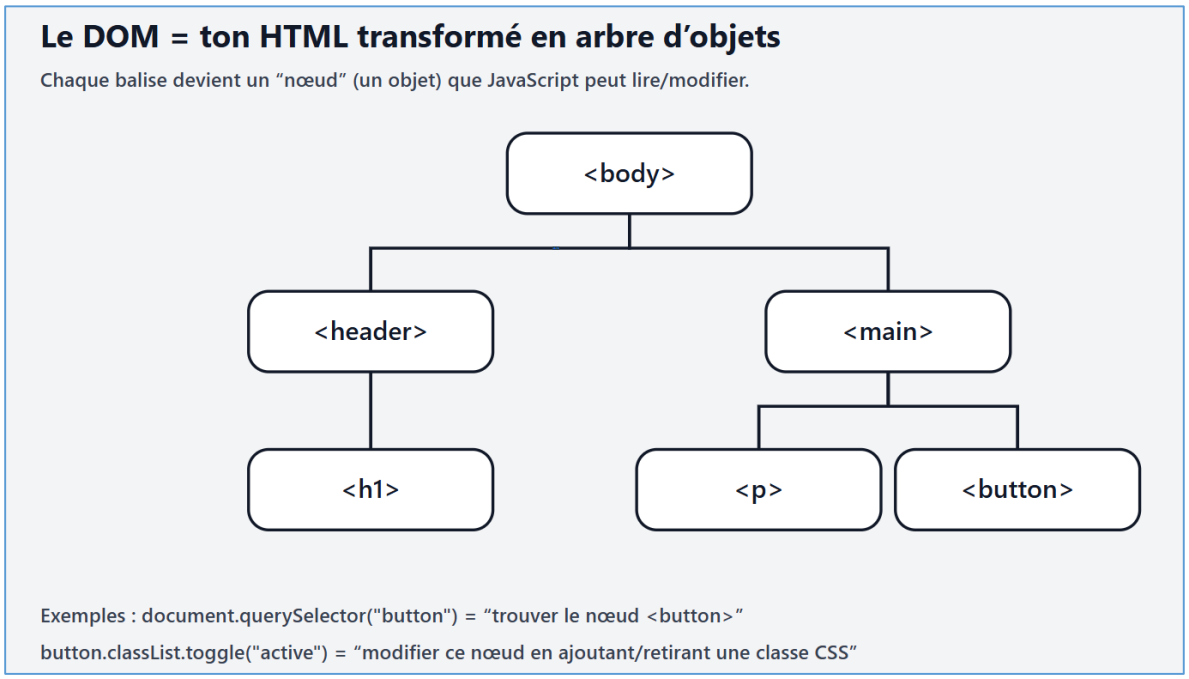
Chaque élément devient un **nœud** dans l'arbre.

Le <body> contient <header> et <main>

<main> contient <p> et <button>

Chaque élément peut avoir :

- un parent
- des enfants
- des frères/sœurs



Pourquoi le DOM est-il important ?

Le DOM est **la porte d'accès de JavaScript à la page** : sans lui, tu ne pourrais pas faire interagir ton interface.

Par exemple :

- faire apparaître/cacher un menu
- changer le texte d'un bouton
- modifier des classes CSS
- afficher dynamiquement du contenu

…tout cela se fait en manipulant le DOM par JavaScript.

Le DOM est la version « vivante » de ta page HTML. C'est une **structure en arbre** que JavaScript peut lire et modifier pour rendre la page **interactive**.

Documentation :

https://developer.mozilla.org/fr/docs/Glossary/DOM

2. Sélectionner des éléments dans la page

Dans le module précédent, nous avons utilisé :

```
document.getElementById("id")
```

C'est utile, mais limité aux IDs.

querySelector()

```
document.querySelector("sélecteur CSS")
```

Fonctionne comme un sélecteur CSS.

Exemples :

```
document.querySelector("#titre");
document.querySelector(".card");
document.querySelector("button");
```

Sélectionne le **premier élément correspondant**.

querySelectorAll()

```
document.querySelectorAll(".btn");
```

Sélectionne **tous les éléments correspondants**.

Retourne une liste d'éléments.

3. Modifier dynamiquement le contenu

Modifier le texte

```
element.textContent = "Nouveau texte";
```

Remplace le contenu texte.

Modifier le HTML

```
element.innerHTML = "<strong>Important</strong>";
```

Injecte du HTML.

4. Modifier l'apparence

Modifier un style directement

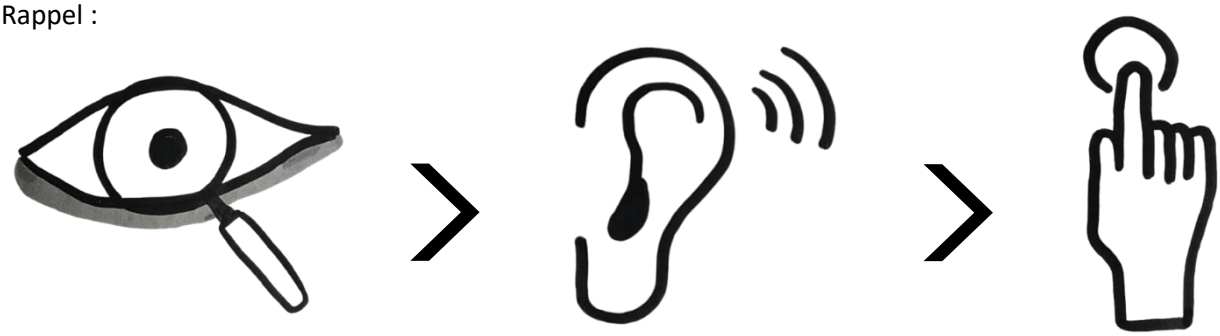
```
element.style.backgroundColor = "red";
element.style.fontSize = "20px";
```

Manipuler les classes

```
element.classList.add("active");
element.classList.remove("active");
element.classList.toggle("active");
```

4. Réagir aux actions utilisateur

Rappel :



On sélectionne **On écoute** **On agit**

Événement : click

```
button.addEventListener("click", function() {
   console.log("Bouton cliqué");
});
```

Événement : input

Déclenché quand l'utilisateur tape dans un champ.

Exemple : affichage en direct

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Introduction au javascript</title>
    <!-- Inclure Bootstrap CSS -->
<link rel="stylesheet" href="css/bootstrap.css">
</head>
<body>
    <input type="text" id="champ">
    <p id="resultat"></p>
    <br><br>
 <script>
    const champ = document.querySelector("#champ");
    const resultat = document.querySelector("#resultat");

    champ.addEventListener("input", function() {
    resultat.textContent = champ.value;
    });
</script>
</body>
</html>
```

Événement : submit

Se déclenche quand un formulaire est envoyé.

Exemple :

```
<form id="monForm">
    <input type="text" id="nom">
    <button type="submit">Envoyer</button>
</form>
```

preventDefault()

Par défaut, un formulaire recharge la page.

Pour empêcher cela :

```
const form = document.querySelector("#monForm");

    form.addEventListener("submit", function(event) {
    event.preventDefault();
    console.log("Formulaire intercepté");
    });
```

preventDefault() bloque le comportement automatique du navigateur.

Événement : keydown

Détecte une touche clavier.

```
document.addEventListener("keydown", function(event) {
   console.log("Touche pressée :", event.key);
});
```

Exercice – Couleur.html - Changement de couleur du fond

Dans cet exercice, changer la couleur de fond d'une page HTML lorsqu'on tape sur la lettre « r ».

Exercice – Couleur2.html - Changement de couleur du fond + sélecteur de couleur

Consigne :

Créer une page contenant :

- Un champ `<input type="color" id="couleur">`
- Un bouton `<button id="btnAppliquer">Appliquer</button>`

Comportement :

- Au clic sur "Appliquer", la couleur de fond de la page devient la couleur choisie.

Contraintes :

- Utiliser `.value`
- Utiliser `addEventListener("click", …)`
- Utiliser `document.body.style.backgroundColor = …`

Livrable attendu :

- Le fond de la page change de couleur après le clic.

Exemple récapitulatif

Création d'une page avec :

- Un champ texte
- Un bouton
- Une zone d'affichage

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Introduction au javascript</title>
    <!-- Inclure Bootstrap CSS -->
<link rel="stylesheet" href="css/bootstrap.css">
</head>
<body class="p-4">
    <h2>Écrire un message</h2>
    <input type="text" id="message" class="form-control mb-2">
    <button id="btnAfficher" class="btn btn-primary mb-3">
        Afficher
    </button>
    <p id="zoneAffichage"></p>
 <script>
    const message = document.querySelector("#message");
    const bouton = document.querySelector("#btnAfficher");
    const zone = document.querySelector("#zoneAffichage");

    bouton.addEventListener("click", function() {
    if (message.value === "") {
        zone.textContent = "Veuillez entrer un message.";
        zone.classList.add("text-danger");
    } else {
        zone.textContent = message.value;
        zone.classList.remove("text-danger");
    }
   });
</script>
</body>
</html>
```

Exercice – chaine.html – remplacer un mot

Demander une phrase à l'utilisateur, un mot à remplacer et un mot de remplacement, afficher ensuite la phrase avec le mot remplacé.

Créer :

- Un champ texte
- Un second champ “mot à remplacer”
- Un bouton “Remplacer”
- Une zone résultat

Comportement :

- Remplacer toutes les occurrences du mot
- Afficher le résultat dans la page

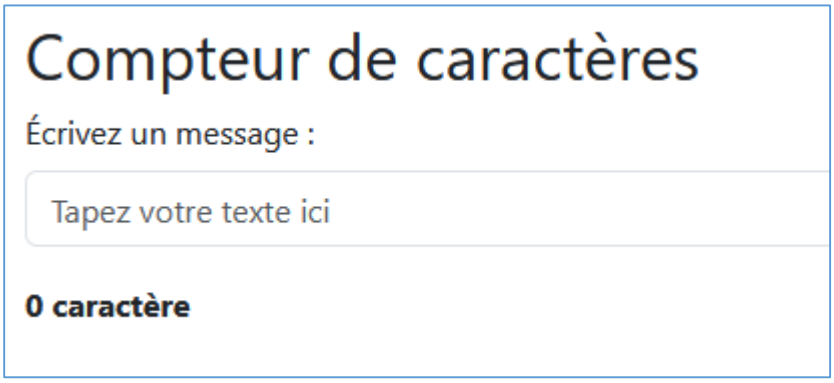
Contraintes :

- Utiliser `.value`
- Utiliser `.replace()`
- Afficher avec `textContent`

Exercice – compteur.html

Créer un formulaire contenant :

- Un champ texte
- Un bouton envoyer



Comportement :

- Si le champ est vide :
  - Empêcher l’envoi du formulaire.
  - Afficher un message d’erreur.
- Sinon :
  - Afficher un message de confirmation.

Contraintes :

- Utiliser `submit`
- Utiliser `preventDefault()`
- Ne pas utiliser `alert()`

Exercice taille du texte – tailleTexte.html

Créer une page contenant :

- Un paragraphe `<p id="texte">…</p>`
- Un bouton + (augmenter) <button id="btnPlus">+</button>
- Un bouton – (diminuer) <button id="btnMoins">-</button>

Comportement :

- Au clic sur +, la taille du texte augmente.
- Au clic sur –, la taille du texte diminue.

Contraintes :

- Utiliser `getElementById`
- Utiliser `addEventListener("click", …)`
- Utiliser une variable `let taille = …` (taille actuelle)
- Mettre à jour l'affichage via `element.style.fontSize = taille + "px"`

Livrable attendu :

- Le texte change de taille immédiatement après chaque clic.