TD3 – JavaScript

1 TD3 / TME 3 : Initiation à JavaScript

Ce TD / TME a pour objectif de mettre en pratique les notions vues en cours sur JavaScript :

- Accès au DOM
- Définition de fonctions
- Manipulation du DOM
- Gestion d'événements

Il n'est pas directement lié au projet Birdie mais a pour but de vous familiariser avec le langage avant de vous confronter au *framework* ReactJS.

3.1 Utilisation des méthodes d'identification

Sans modifier le code HTML de l'exercice 1 :

- Comment accéder à l'élément d'identifiant `p1`
- Comment accéder aux éléments de classe `classe1` ?
- Comment accéder à l'élément `header` ?
- Comment accéder aux paragraphes de classe `classe1` ?
- Comment accéder aux éléments enfants de `main` ?
- Combien d'enfants l'élément `main` possède-t-il ? combien d'éléments enfants ?
- Quel est le «`nextSibling`» du dernier élément `h2` ?
- Comment accéder au dernier paragraphe du document ?

```
<!DOCTYPE html>
<html lang="fr">
<head>
        <meta charset="UTF-8">
        <title>Exercice 1</title>
        <script src="correction/ex1.js" defer></script>
</head>
<body>
<header>
<h1 class="classe1">Grand titre</h1>

<p>Paragraphe <a href="https://fr.wiktionary.org/wiki/liminaire">liminaire</a>.</p>
</header>

<main>
<h2>Texte de remplissage</h2>
<details>Lorem ipsum de base</details>

<p id="p1">Lorem ipsum dolor sit amet. Ut enim fugiat et aliquid debitis aut consequatur dolore et fugit
↪  sint sed voluptatum sunt. 33 itaque neque aut doloribus corporis At quia placeat ut sunt galisum.</p>

<p>Qui internos vitae hic assumenda cumque aut necessitatibus molestiae vel voluptates optio 33 autem
↪  facilis. Est aliquid dolor aut ipsam ducimus eum voluptas error. Et sapiente consequatur qui labore
↪  velit sit quisquam omnis 33 nesciunt omnis in atque perspiciatis vel veritatis earum!</p>

<p class="classe1">Nam corporis dicta in quae voluptas non assumenda exercitationem aut repellendus sunt et
↪  repellendus reiciendis. Qui maxime dolore ab ratione cupiditate ut voluptas iusto aut neque
↪  inventore.</p>

<h2>Second texte de remplissage</h2>
<details>avec la bonne distribution de longueurs de mots</details>

```

```
28 <p>Généralement, on utilise un texte en faux latin (le texte ne veut rien dire, il a été modifié), le Lorem
   ↪ ipsum ou Lipsum, qui permet donc de faire office de texte d'attente. L'avantage de le mettre en latin
   ↪ est que l'opérateur sait au premier coup d'œil que la page contenant ces lignes n'est pas valide, et
   ↪ surtout l'attention du client n'est pas dérangée par le contenu, il demeure concentré seulement sur
   ↪ l'aspect graphique.</p>
29
30 <p id="p2" class="classe1">Ce texte a pour autre avantage d'utiliser des mots de longueur variable, essayant
   ↪ de simuler une occupation normale. La méthode simpliste consistant à copier-coller un court texte
   ↪ plusieurs fois (<q>ceci est un faux-texte ceci est un faux-texte ceci est un faux-texte ceci est un
   ↪ faux-texte ceci est un faux-texte</q>) a l'inconvénient de ne pas permettre une juste appréciation
   ↪ typographique du résultat final.</p>
31
32 <p>Il circule des centaines de versions différentes du Lorem ipsum, mais ce texte aurait originellement été
   ↪ tiré de l'ouvrage de Cicéron, De Finibus Bonorum et Malorum (Liber Primus, 32), texte populaire à cette
   ↪ époque, dont l'une des premières phrases est : <q lang="la"
   ↪ cite="https://la.wikisource.org/wiki/De_finibus_bonorum_et_malorum/Liber_Primus">Neque porro quisquam
   ↪ est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit... </q> (<q>Il n'existe personne
   ↪ qui aime la souffrance pour elle-même, ni qui la recherche ni qui la veuille pour ce qu'elle est...
   ↪ </q>).</p>
33
34 <h2>Une dernière section</h2>
35
36 <p>Ceci est un court paragraphe.</p>
37
38 </main>
39
40 </body>
41 </html>
```

3.2 Variables et fonctions

Créer un fichier JavaScript et l'appeler dans un document HTML vide. Il faudra utiliser la console du navigateur dans les outils de développement.

1. Créer trois variables `var_var`, `var_let` et `var_const` avec respectivement les mots-clefs `var`, `let` et `const`. Les initialiser, puis essayer de changer leur valeur. Traiter l'exception avec un message d'erreur personnalisé complétant le message par défaut.
2. Créer un objet `{prop1: 1, prop2: 2, prop3: 4}`. Créer une fonction qui :
   — prend un objet en paramètre
   — retourne un objet de même taille que le paramètre d'entrée et indiquant si chaque propriété de l'objet d'entrée est un nombre pair ou impair. Par exemple, appliquée à l'objet précédemment défini, la fonction doit retourner `{prop1: "1 est impair.", prop2: "2 est pair.", prop3: "4 est pair."}`
3. Nous allons découvrir quelques fonctions et méthodes utiles...
   — Créer une fonction fléchée prenant deux paramètres `x` et `nombre`, et renvoyant `true` si `x` est un multiple de `nombre`, `false` sinon.
   — Sachant que `Math.floor(x)` renvoie la partie entière de x, et `Math.random()` un nombre aléatoire compris entre 0 et 1, créer un tableau `nombres` de 20 nombres entiers aléatoires compris entre 0 et 99.
   — Après avoir consulté la documentation des méthodes map et filter applicables aux tableaux, les utiliser pour :
      — créer un tableau dont tous les éléments sont égaux à la moitié des éléments du tableau `nombres` (par exemple, si `nombres` commence par [6, 7, 56...), le nouveau tableau commencera par [3, 3.5, 28...]).
      — créer un tableau ne comportant que les éléments de `nombres` multiples de 3.

3.3 Gestion d'événements

Ajouter au fichier ex3.html un code JavaScript :
— il associe au click sur le bouton une fonction `additionne` pour gérer l'événement
— la fonction `additionne` affiche dans la console la somme des deux nombres saisis dans les champs
— si l'utilisateur a maintenu la touche Shift appuyée lors du click, afficher un message supplémentaire dans la console.

```html
<!DOCTYPE html>
<html lang="fr">
<head>
        <meta charset="UTF-8">
        <title>Exercice 2</title>
        <script src="correction/ex3.js" defer></script>
</head>
<body>
<header>
<body>
<label for="nb1">Nombre 1</label><input type="number" id="nb1" min="0" max="100"><br>
<label for="nb2">Nombre 2</label><input type="number" id="nb2" min="0" max="100"><br>
<button id="btn_ok">Calcul!</button>
</body>
</html>
```

3.4 Manipulation du DOM

1. Ajouter au fichier ex4_q1.html un code JavaScript qui à chaque click sur le bouton `btn_ajout`, ajoute un élément de liste contenant le texte «texte» à la liste. Au click sur le second bouton, supprimer le dernier élément de liste, sans qu'aucune erreur JavaScript ne soit générée.

```html
<!DOCTYPE html>

<html lang="fr">
<head>
<title>Ajout / Suppression d'éléments</title>
<meta charset="UTF-8">
<script src="correction/ex4_q1.js" defer></script>

</head>
<body>

<ul id="listecommissions">
        <li>1kg de farine</li>
        <li>un pack de lait</li>
</ul>

<button id="btn_add">Ajoutez... </button><button id="btn_supp">Supprimez... </button>

</body>
</html>
```

2. En affectant le même gestionnaire aux clicks sur les deux boutons, trouver un moyen pour qu'en cliquant par exemple sur le bouton «/\», le premier élément de la seconde liste devienne le dernier de la première, et symétriquement pour le second bouton (le premier élément e la première liste devient le dernier de la seconde). Vous pourrez utiliser la propriété `target` de l'événement, qui désigne l'élément à l'origine de l'événement (ici le bouton cliqué).

```html
<!DOCTYPE html>

<html lang="fr">
<head>
<title>Ajout / Suppression d'éléments</title>
<meta charset="UTF-8">
<script src="correction/ex4_q2.js" defer></script>
</head>
<body>

<ul id="liste1">
        <li>1kg de farine</li>
        <li>un pack de lait</li>
</ul>

```

```
16  <button id="btn_2vers1">/\</button><button id="btn_1vers2">\/</button>
17
18  <ul id="liste2">
19          <li>une plaquette de beurre</li>
20          <li>une baguette</li>
21  </ul>
22
23
24
25  </body>
26  </html>
```