

Atelier 12 : Javascript 2/2

Sommaire

<u>Introduction</u>	1
<u>Navigateur 1</u>	2
<u>Exercice : DOM</u>	2
<u>Exercice : Recherche d'éléments</u>	3
<u>Exercice : descendants,balise,hiérarchie</u>	5
<u>Navigateur 2</u>	7
<u>Exercice : Obtenez l'attribut</u>	7
<u>Exercice : Rendre les liens externes orange</u>	8
<u>Exercice : createTextNode vs innerHTML vs textContent</u>	9
<u>Exercice : Effacer l'élément</u>	9
<u>Exercice : « pourquoi aaa reste-t-il ? »</u>	10
<u>Exercice : Créer une liste</u>	11
<u>Exercice : Créer un arbre à partir d'un objet</u>	12
<u>Exercice : Afficher les descendants d'un arbre</u>	12
<u>Exercice : Calendrier / Date/ HTML</u>	13
<u>Exercice : Notification</u>	15
<u>Exercice : Ballon</u>	16
<u>Exercice : Coordonnées</u>	18
<u>Conclusion</u>	19

Introduction

Dans le cadre de mes premiers travaux pratiques en JavaScript, j'ai découvert les bases du langage et la manière d'intégrer des scripts dans une page HTML. J'ai appris à utiliser la balise <script>.

Ce TP marque le début de mon apprentissage du fonctionnement et des règles fondamentales de JavaScript.

Navigateur 1

Exercice : DOM

Enfants DOM

Regardez cette page :

```
1 <html>
2 <body>
3   <div>Users:</div>
4   <ul>
5     <li>John</li>
6     <li>Pete</li>
7   </ul>
8 </body>
9 </html>
```

Pour chacun des éléments suivants, donnez au moins un moyen d'y accéder :

- Le noeud `<div>` du DOM ?
- Le noeud `` du DOM ?
- Le deuxième `` (avec Pete) ?

Accéder au nœud `<div>`

Méthode : par le type de balise

```
document.querySelector("div");
```

Accéder au nœud ``

Méthode : `querySelector`

Accéder au deuxième `` (Pete)

Méthode : en partant du ``

La question des frères et sœurs

Si `element` – est un nœud élément arbitraire du DOM ...

- Est-il vrai que `elem.lastChild.nextSibling` est toujours `null` ?
- Est-il vrai que `elem.children[0].previousSibling` est toujours `null` ?

Les deux affirmations sont fausses, car `nextSibling` et `previousSibling` peuvent référencer des nœuds texte.

Il faut utiliser `nextElementSibling` et `previousElementSibling` pour garantir `null`.

Sélectionner toutes les cellules diagonales

Écrivez le code pour colorer toutes les cellules du tableau diagonal en rouge.

Vous devrez obtenir toutes les diagonales `<td>` de la `<table>` et les colorer en utilisant le code :

```
1 // td doit être la référence à la cellule du tableau
2 td.style.backgroundColor = 'red';
```

Le résultat devrait être :

1:1	2:1	3:1	4:1	5:1
1:2	2:2	3:2	4:2	5:2
1:3	2:3	3:3	4:3	5:3
1:4	2:4	3:4	4:4	5:4
1:5	2:5	3:5	4:5	5:5

Pour sélectionner toutes les cellules `<td>` du tableau :

```
const cells = document.querySelectorAll('td');
```

Exercice : Recherche d'éléments

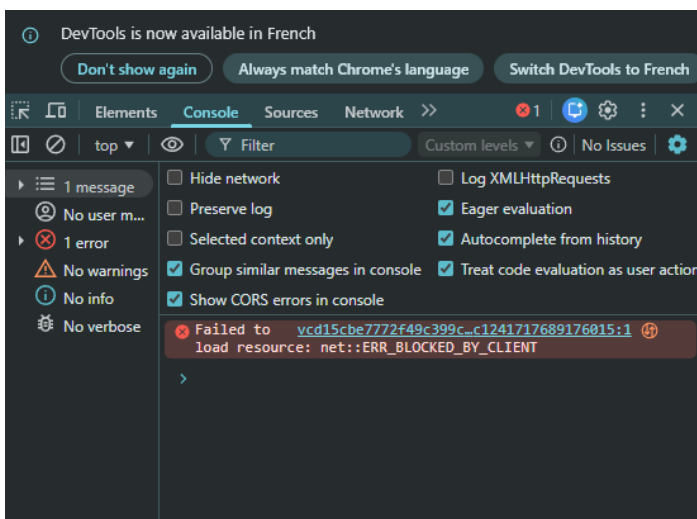
Recherche d'éléments

Voici le document avec le tableau et formulaire

Comment trouver ?...

1. Le tableau avec `id="age-table"`.
2. Tous les éléments `label` dans ce tableau (il devrait y en avoir 3).
3. Le premier `td` dans ce tableau (avec le mot "Age").
4. Le `form` avec `name="search"`.
5. Le premier `input` dans ce formulaire.
6. Le dernier `input` dans ce formulaire.

Après avoir ouvert table.html puis utiliser F12 je n'ai pas pu taper les ligne de code dans la console du navigateur a cause d'une erreur



Donc voici les ligne de code que j'aurai taper une par une.

// 1. Le tableau avec id="age-table"

```
let table = document.getElementById('age-table');
```

// 2. Tous les éléments label dans ce tableau

```
let labels = table.getElementsByTagName('label');
```

// OU

```
let labels = table.querySelectorAll('label');
```

// 3. Le premier td dans ce tableau (avec le mot "Age")

```
let firstTd = table.getElementsByTagName('td')[0];
```

// OU

```
let firstTd = table.querySelector('td');
```

// 4. Le form avec name="search"

```
let form = document.querySelector('form[name="search"]');  
// OU  
let form = document.forms.search;
```

// 5. Le premier input dans ce formulaire

```
let firstInput = form.querySelector('input');  
// OU  
let firstInput = form.getElementsByTagName('input')[0];
```

// 6. Le dernier input dans ce formulaire

```
let inputs = form.querySelectorAll('input');  
let lastInput = inputs[inputs.length - 1];
```

getElementById() : recherche par ID

getElementsByTagName() : recherche par balise HTML

querySelector() : recherche avec sélecteur CSS (premier élément)

querySelectorAll() : recherche avec sélecteur CSS (tous les éléments)

document.forms : accès aux formulaires par nom

Exercice : descendants, balise, hiérarchie

Compter les descendants

Qu'affiche le script ?

```
1 <html>  
2  
3 <body>  
4 <script>  
5   alert(document.body.lastChild.nodeType);  
6 </script>  
7 </body>  
8  
9 </html>
```

Balise dans le commentaire

Qu'affiche ce code ?

```
1 <script>  
2   let body = document.body;  
3  
4   body.innerHTML = "<!--" + body.tagName + "-->";  
5  
6   alert( body.firstChild.data ); // Qu'est ce qu'il y a ici ?  
7 </script>
```

Où est le "document" dans la hiérarchie ?

À quelle classe appartient le `document` ?

Quelle est sa place dans la hiérarchie DOM ?

Hérite-t-il de `Node` ou `Element`, ou peut-être de `HTMLElement` ?

Compter les descendants

Le script affiche 1.

Explication :

Dans ce document HTML, le dernier enfant de <body> est l'élément <script> lui-même .

La propriété `nodeType` retourne :

- 1 pour un nœud élément (ELEMENT_NODE)
- 8 pour un commentaire (COMMENT_NODE)
- 3 pour un texte (TEXT_NODE)

Ainsi, `document.body.lastChild.nodeType` retourne 1 car <script> est un élément.

Balise dans le commentaire

Le script affiche "BODY".

Explication :

1. `body.tagName` retourne "BODY" (le nom de la balise en majuscules)

2. `body.innerHTML = "<!--" + body.tagName + "-->"` transforme le contenu du body en un commentaire HTML : <!--BODY-->

3. `body.firstChild` accède au premier enfant, qui est maintenant le nœud commentaire

4. `data` sur un nœud commentaire retourne son contenu textuel, soit "BODY"

Où est le "document" dans la hiérarchie ?

Classe du document :

Le document appartient à la classe `HTMLDocument`. Cette classe est spécifique aux documents HTML.

Hiérarchie dans le DOM :

Le document suit cette chaîne d'héritage :

`EventTarget` → `Node` → `Document` → `HTMLDocument`

Héritage :

Oui, le document hérite de `Node`. Cela signifie qu'il dispose des propriétés et méthodes communes à tous les nœuds du DOM, comme `parentNode`, `childNodes`, etc.

Non, le document n'hérite pas de `Element`. Un document est le conteneur racine qui possède les éléments, mais il n'est pas lui-même un élément.

Non, le document n'hérite pas de `HTMLElement`. Cette classe est réservée aux éléments HTML spécifiques comme <div>, <p>, etc.

Dans l'arbre DOM, le document représente la racine de l'arborescence. Il sert de point d'entrée pour accéder à tous les éléments de la page. Bien qu'il soit un nœud (`Node`), il se situe à un niveau supérieur aux éléments qu'il contient.

Vérification pratique :

On peut confirmer cela avec quelques lignes de JavaScript :

```
javascript
// Vérifications
console.log(document instanceof HTMLDocument); // true
console.log(document instanceof Document);      // true
console.log(document instanceof Node);           // true
console.log(document instanceof Element);        // false
console.log(document instanceof HTMLElement);    // false
```

Navigateur 2

Exercice : Obtenez l'attribut

Obtenez l'attribut

Écrivez le code pour sélectionner l'élément avec l'attribut `data-widget-name` dans le document et pour lire sa valeur.

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5   <div data-widget-name="menu">Choose the genre</div>
6
7   <script>
8     /* your code */
9   </script>
10 </body>
11 </html>
```

Réponse :

```
<!DOCTYPE html>
<html>
<body>

<div data-widget-name="menu">Choose the genres</div>

<script>
  // mon code:
  let element = document.querySelector('[data-widget-name]');
  let valeur = element.getAttribute('data-widget-name');
  alert(valeur); // Affiche "menu"
</script>
</body>
</html>
```

Exercice : Rendre les liens externes orange

Rendre les liens externes orange

Mettez tous les liens externes en orange en modifiant leur propriété `style`.

Un lien est externe si :

- Son `href` contient `://`
- Mais ne commence pas par `http://internal.com`.

Exemple :

```

1 <a name="list">the list</a>
2 <ul>
3   <li><a href="http://google.com">http://google.com</a></li>
4   <li><a href="/tutorial">/tutorial.html</a></li>
5   <li><a href="local/path">local/path</a></li>
6   <li><a href="ftp://ftp.com/my.zip">ftp://ftp.com/my.zip</a></li>
7   <li><a href="http://nodejs.org">http://nodejs.org</a></li>
8   <li><a href="http://internal.com/test">http://internal.com/test</a></li>
9 </ul>
10
11 <script>
12   // setting style for a single link
13   let link = document.querySelector('a');
14   link.style.color = 'orange';
15 </script>

```

Le résultat devrait être :

The list:

- <http://google.com>
- </tutorial.html>
- <local/path>
- <ftp://ftp.com/my.zip>
- <http://nodejs.org>
- <http://internal.com/test>

Réponse :

```

<a name="list">the list</a>
<ul>
  <li><a href="http://google.com">http://google.com</a></li>
  <li><a href="/tutorial">/tutorial.html</a></li>
  <li><a href="local/path">local/path</a></li>
  <li><a href="ftp://ftp.com/my.zip">ftp://ftp.com/my.zip</a></li>
  <li><a href="http://nodejs.org">http://nodejs.org</a></li>
  <li><a href="http://internal.com/test">http://internal.com/test</a></li>
</ul>

<script>
  // Sélectionner tous les liens
  let allLinks = document.querySelectorAll('a');

  // Appliquer le style orange aux liens externes
  allLinks.forEach(link => {
    let href = link.getAttribute('href');

    if (href) {
      // Vérifier les critères pour un lien externe
      let isExternal = href.includes('://') && !href.startsWith('http://internal.com');

      if (isExternal) {
        link.style.color = 'orange';
      }
    }
  });
</script>

```

Exercice : createTextNode vs innerHTML vs textContent

createTextNode vs innerHTML vs textContent

Nous avons un élément DOM vide `elem` et une chaîne de caractères `text`.

Lesquelles de ces 3 commandes feront exactement la même chose ?

1. `elem.append(document.createTextNode(text))`
2. `elem.innerHTML = text`
3. `elem.textContent = text`

Les commandes qui font exactement la même chose sont :

1 `elem.append(document.createTextNode(text))`

3 `elem.textContent = text`

La commande différente est :

2 `elem.innerHTML = text`

`createTextNode()` et `textContent` traitent le contenu comme du texte pur

• `innerHTML` interprète et exécute le contenu comme du HTML

Donc les méthodes 1 et 3 sont équivalentes, tandis que la méthode 2 se comporte différemment si `text` contient des balises HTML.

Exercice : Effacer l'élément

Effacer l'élément

Créez une fonction `clear(elem)` qui supprime tout de l'élément.

```
1 <ol id="elem">
2   <li>Hello</li>
3   <li>World</li>
4 </ol>
5
6 <script>
7   function clear(elem) { /* votre code */ }
8
9   clear(elem); // efface la liste
10 </script>
```

```
<ol id="elem">
  <li>Hello</li>
  <li>World</li>
</ol>

<script>
  // Solution simple
  function clear(elem) {
    elem.innerHTML = '';
  }

  // Récupérer l'élément et appeler la fonction
  let elem = document.getElementById('elem');
  clear(elem);
</script>
```

Pour cet exercice j'ai utilisé, `innerHTML = ''` car c'est la solution la plus adaptée et la plus simple.

Exercice : « pourquoi aaa reste-t-il ? »

Pourquoi "aaa" reste-t-il ?

Dans l'exemple ci-dessous, l'appel `table.remove()` supprime le tableau du document. mais si vous l'exécutez, vous pouvez voir que le texte "aaa" est toujours visible.

Pourquoi cela se produit-il ?

```
1 <table id="table">
2   aaa
3   <tr>
4     <td>Test</td>
5   </tr>
6 </table>
7
8 <script>
9   alert(table); // la table, comme il se doit
10
11   table.remove();
12   // pourquoi y a-t-il encore "aaa" dans le document ?
13 </script>
```

Le texte "aaa" n'est pas à l'intérieur d'un élément `<tr>` ou `<td>`, ce qui est syntaxiquement incorrect pour une table HTML.

Le HTML est mal formé : Le texte "aaa" est directement à l'intérieur de `<table>`, ce qui n'est pas autorisé

Le navigateur corrige automatiquement l'HTML lors du parsing :

- Il place "aaa" AVANT la table
- Il crée une structure de table valide

Exercice : Créer une liste

Créer une liste

Écrivez une interface pour créer une liste à partir des entrées utilisateur.

Pour chaque élément de la liste :

1. Interrogez un utilisateur sur son contenu en utilisant `prompt`.
2. Créez le `` avec et ajoutez-le à ``.
3. Continuez jusqu'à ce que l'utilisateur annule l'entrée (en appuyant sur la touche `Esc` ou une entrée vide).

Tous les éléments doivent être créés dynamiquement.

Si un utilisateur tape des balises HTML, elles doivent être traitées comme un texte.

Réponse :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Créer une liste</title>
</head>
<body>

  <ul id="liste"></ul>

  <script>
    const ul = document.getElementById("liste");

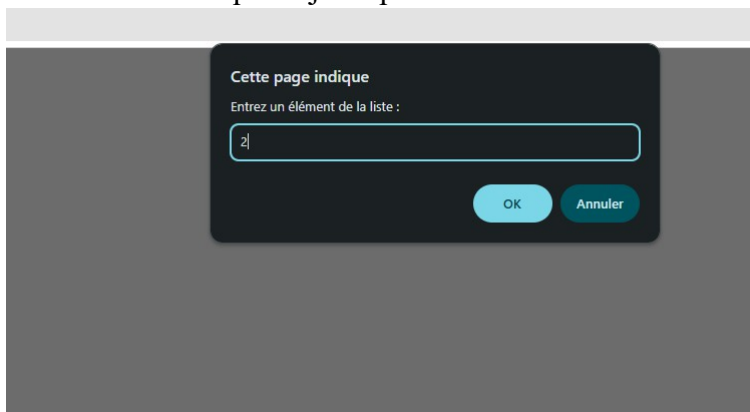
    while (true) {
      const texte = prompt("Entrez un élément de la liste :");

      if (texte === null || texte === "") {
        break;
      }

      const li = document.createElement("li");
      li.textContent = texte;
      ul.appendChild(li);
    }
  </script>

</body>
</html>
```

L'entrée dans laquelle j'ai tapé tous les chiffres



J'ai fait le test de 1 jusqu'à 9 voici donc la liste afficher après avoir tapé sur échap (Cela fonctionne également avec la touche entrée.)

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

Exercice : Créer un arbre à partir d'un objet

Créer un arbre à partir de l'objet

Écrivez une fonction `createTree` qui crée une liste imbriquée `ul/li` à partir de l'objet imbriqué.

Par exemple :

```
1 let data = {  
2   "Fish": {  
3     "trout": {},  
4     "salmon": {}  
5   },  
6  
7   "Tree": {  
8     "Huge": {  
9       "sequoia": {},  
10      "oak": {}  
11    },  
12    "Flowering": {  
13      "apple tree": {},  
14      "magnolia": {}  
15    }  
16  }  
17 };
```

La syntaxe :

```
1 let container = document.getElementById('container');  
2 createTree(container, data); // crée l'arbre dans le conteneur
```

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Créer un arbre</title>
</head>
<body>

  <div id="container"></div>

<script>
  let data = {
    "Fish": {
      "trout": {},
      "salmon": {}
    },
    "Tree": {
      "Huge": {
        "sequoia": {},
        "oak": {}
      },
      "Flowering": {
        "apple tree": {},
        "magnolia": {}
      }
    }
  };

  function createTree(container, data) {
    const ul = document.createElement("ul");

    for (let key in data) {
      const li = document.createElement("li");
      li.textContent = key;

      if (Object.keys(data[key]).length > 0) {
        createTree(li, data[key]);
      }

      ul.appendChild(li);
    }

    container.appendChild(ul);
  }

  const container = document.getElementById("container");
  createTree(container, data);
</script>
</body>
</html>
```

L'affichage sur l' HTML

Exercice : Afficher les descendants d'un arbre

- Fish
 - trout
 - salmon
- Tree
 - Huge
 - sequoia
 - oak
 - Flowering
 - apple tree
 - magnolia

scendants. Sauter les feuilles (nœuds sans

- Birds
- Lizards

- Fishes [5]
 - Aquarium [2]
 - Guppy
 - Angelfish
 - Sea [1]
 - Sea trout

Exercice : Calendrier / Date/ HTML

Créer un calendrier

Écrivez une fonction `createCalendar(elem, year, month)`.

L'appel doit créer un calendrier pour l'année/le mois donné et le mettre dans `elem`.

Le calendrier doit être un tableau, où une semaine est un `<tr>` et un jour est un `<td>`. Le dessus du tableau doit être un `<th>` avec les noms des jours de la semaine : le premier jour doit être le lundi, et ainsi de suite jusqu'au dimanche.

Par exemple, `createCalendar(cal, 2012, 9)` devrait générer dans l'élément `cal` le calendrier suivant :

MO	TU	WE	TH	FR	SA	SU
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

P.S. Pour cette tâche, il suffit de générer le calendrier, il ne doit pas encore être cliquable.

Voici le calendrier il n'est pas cliquable

← → ↻ ⓘ Fichier D:/bts/tp/mr%20sotoca/atelier/index.html

MO	TU	WE	TH	FR	SA	SU
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

```

index.html  nouveau 1.js
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4    <meta charset="UTF-8">
5    <title>Calendrier</title>
6  </head>
7  <body>
8    <div id="cal"></div>
9
10   <script>
11     function createCalendar(elem, year, month) {
12       let table = document.createElement("table");
13
14       // En-tête avec les jours
15       let days = ["MO", "TU", "WE", "TH", "FR", "SA", "SU"];
16       let tr = document.createElement("tr");
17
18       for (let day of days) {
19         let th = document.createElement("th");
20         th.textContent = day;
21         tr.appendChild(th);
22       }
23       table.appendChild(tr);
24
25       let date = new Date(year, month - 1, 1);
26       let currentRow = document.createElement("tr");
27
28       // Décalage du premier jour (lundi = 1)
29       let startDay = date.getDay();
30       if (startDay === 0) startDay = 7;
31
32       for (let i = 1; i < startDay; i++) {
33         currentRow.appendChild(document.createElement("td"));
34       }
35
36       // Remplir les jours
37       while (date.getMonth() === month - 1) {
38         let td = document.createElement("td");
39         td.textContent = date.getDate();
40         currentRow.appendChild(td);
41
42         if (date.getDay() === 0) {
43           table.appendChild(currentRow);
44           currentRow = document.createElement("tr");
45         }
46
47         date.setDate(date.getDate() + 1);
48       }
49
50       // Ajouter la dernière ligne
51       if (currentRow.children.length > 0) {
52         table.appendChild(currentRow);
53       }
54     }
55   </script>

```

Horloge colorée avec setInterval

Créez une horloge colorée comme ici :

hh:mm:ss
Start Stop

Utilisez HTML/CSS pour le style, JavaScript ne met à jour que le temps dans les éléments.

Insérez le HTML dans la liste

Écrivez le code pour insérer `23` entre deux `` ici :

```
1 <ul id="ul">
2   <li id="one">1</li>
3   <li id="two">4</li>
4 </ul>
```

Trier le tableau

Il y a un tableau :

```
1 <table>
2 <thead>
3   <tr>
4     <th>Name</th><th>Surname</th><th>Age</th>
5   </tr>
6 </thead>
7 <tbody>
8   <tr>
9     <td>John</td><td>Smith</td><td>10</td>
10  </tr>
11  <tr>
12    <td>Pete</td><td>Brown</td><td>15</td>
13  </tr>
14  <tr>
15    <td>Ann</td><td>Lee</td><td>5</td>
16  </tr>
17  <tr>
18    <td>...</td><td>...</td><td>...</td>
19  </tr>
20 </tbody>
21 </table>
```

Il peut y avoir plus de lignes.

Écrivez le code pour le trier par la colonne "name".

Exercise : Notification

Create a notification

Write a function `showNotification(options)` that creates a notification: `<div class="notification">` with the given content. The notification should automatically disappear after 1.5 seconds.

The options are:

```
1 // shows an element with the text "Hello" near the right-top of the window
2 showNotification({
3   top: 10, // 10px from the top of the window (by default 0px)
4   right: 10, // 10px from the right edge of the window (by default 0px)
5   html: "Hello!", // the HTML of notification
6   className: "welcome" // an additional class for the div (optional)
7 });
```

Notification is on the right

Hello 6

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dolorum aspernatur quam ex eaque inventore quod voluptatem adipisci omnis nemo nulla fugit iste numquam ducimus cumque minima porro ea quidem maxime necessitatibus beatae labore soluta voluptatum magnam consequatur sit laboriosam velit excepturi laborum sequi eos placeat et quia deleniti? Corrupti velit impedit autem et obcaecati fuga debitis nemo ratione iste veniam amet dicta hic ipsam unde cupiditate incidunt aut iure ipsum officiis soluta temporibus. Tempore dicta ullam delectus numquam consectetur quisquam explicabo culpa excepturi placeat quo sequi molestias reprehenderit hic at nemo cumque voluptates quidem repellendus maiores unde earum molestiae ad.

Use CSS positioning to show the element at given top/right coordinates. The source document has the necessary styles.

Exercice : Ballon

Quel est le défilement à partir du bas ?

La propriété `elem.scrollTop` est la taille de la partie déroulante à partir du haut. Comment obtenir la taille du défilement inférieur (appelons-le `scrollBottom`) ?

Écrivez le code qui fonctionne pour un `element` arbitraire.

P.S. Veuillez vérifier votre code: s'il n'y a pas de défilement ou que l'élément est entièrement défilé vers le bas, alors il devrait retourner `0`.

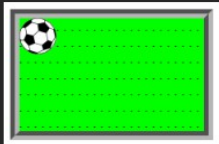
Quelle est la largeur de la barre de défilement ?

Écrivez le code qui renvoie la largeur d'une barre de défilement standard.

Pour Windows, il varie généralement entre `12px` et `20px`. Si le navigateur ne lui réserve pas d'espace (la barre de défilement est à moitié translucide sur le texte, cela arrive également), alors il peut s'agir de `0px`.

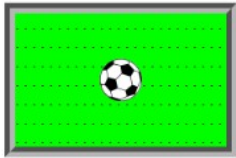
P.S. Le code devrait fonctionner pour tout document HTML, ne dépend pas de son contenu.

Placer la balle au centre du terrain



Quelles sont les coordonnées du centre de terrain ?

Calculez et utilisez-les pour placer la balle au centre du champ vert :



- L'élément doit être déplacé par JavaScript, pas CSS.
- Le code doit fonctionner avec n'importe quelle taille de boule (10 , 20 , 30 pixels) et n'importe quelle taille de champ, ne pas être lié aux valeurs données.

P.S. Bien sûr, le centrage pourrait être effectué avec CSS, mais ici, nous voulons exactement JavaScript. De plus, nous rencontrerons d'autres sujets et des situations plus complexes lorsque JavaScript doit être utilisé. Ici, nous faisons un "échauffement".

La différence: largeur CSS vs clientWidth

Quelle est la différence entre `getComputedStyle(elem).width` et `elem.clientWidth` ?

Donnez au moins 3 différences. Plus c'est mieux.

Exercice : Coordonnées

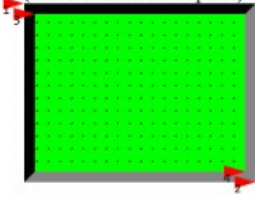
Trouver les coordonnées de la fenêtre du champ

Dans l'iframe ci-dessous, vous pouvez voir un document avec le "champ" vert.

Utilisez JavaScript pour trouver les coordonnées de la fenêtre des coins pointés par des flèches.

Il y a une petite fonctionnalité implémentée dans le document pour plus de commodité. Un clic à n'importe quel endroit montre les coordonnées là-bas.

Click anywhere to get window coordinates.
That's for testing, to check the result you get by JavaScript.
(click coordinates show up here)



Votre code doit utiliser DOM pour obtenir les coordonnées de la fenêtre de :

1. Coin extérieur supérieur gauche (c'est simple).
2. En bas à droite, coin extérieur (simple aussi).
3. Coin intérieur supérieur gauche (un peu plus dur).
4. En bas à droite, coin intérieur (il y a plusieurs façons, choisissez-en une).

Les coordonnées que vous calculez doivent être les mêmes que celles renvoyées par le clic de souris.

P.S. Le code devrait également fonctionner si l'élément a une autre taille ou bordure, qui n'est lié à aucune valeur fixe.

Afficher une note près de l'élément

Créez une fonction `positionAt(anchor, position, elem)` qui positionne `elem`, en fonction de `position` près de l'élément `anchor`.

La `position` doit être une chaîne de caractères avec l'une des 3 valeurs :

- "top" – position `elem` juste au dessus de `anchor`
- "right" – position `elem` immédiatement à droite de `anchor`
- "bottom" – position `elem` juste en dessous `anchor`

Il est utilisé à l'intérieur de la fonction `showNote(anchor, position, html)`, fournie dans le code source de la tâche, qui crée un élément "note" avec `html` donné et l'affiche à la `position` donnée près de `anchor`.

Voici la démo des notes :

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Reprehenderit sint atque dolorum fuga ad incididunt voluptatum error fugiat animi amet! Odio temporibus nulla id unde quacrat dignissimos enim nisi rem provident molestias sit tempore omnis recusand in officia sapiente.

note above

Teacher: Why are you late?
Student: There was a man who lost a hundred dollar bill.
Teacher: That's nice. Were you helping him look for it?
Student: No. I was standing on it.

note at the right

note below

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Reprehenderit sint atque dolorum fuga ad incididunt voluptatum error fugiat animi amet! Odio temporibus nulla id unde quacrat dignissimos enim nisi rem provident molestias sit tempore omnis recusandae esse sequi officia sapiente.

Conclusion

Ce TP m'a permis de comprendre les premiers éléments essentiels de JavaScript et d'apprendre à exécuter du code dans une page web. J'ai découvert les instructions, les points-virgules, les commentaires et le mode strict. Ces notions me serviront de base pour progresser vers des scripts plus complexes et développer mes compétences en programmation web.