

TP - Javascript

Partie 4

Sélectionner toutes les cellules diagonales

Recherche d'éléments

Compter les descendants

Balise dans le commentaire

Où est le document dans la hiérarchie

Partie 5

Obtenez l'attribut

Rendre un lien externe orange

Commande createTextNode vs innerHTML vs textContent

Effacer l'élément

Pourquoi "aaa" reste-t-il

Créer une liste

Créer un arbre à partir de l'objet

Afficher les descendants dans un arbre

Créer un calendrier

Horloge colorée avec setInterval

Insérer le HTML dans la liste

Trier le tableau

Quel est le défilement à partir du bas ?

Quelle est la largeur de la barre de défilement ?

Trouver les coordonnées de la fenêtre du champ

Afficher une note près de l'élément

Cacher sur clic

Se cacher

Quels handlers exécutent ?

Ballon

Déplacer le ballon à travers le terrain

Créez un menu glissant

Ajoutez un bouton de fermeture

Carrousel

Partie 6

Liste sélectionnable

Info-bulle "Intelligente"

Partie 4

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) ?

Pour accéder au noeud `<div>` DOM on peut utiliser `document.querySelector('div')` qui va sélectionner le premier élément '`<div>`' dans le document.

On peut également choisir d'utiliser `document.getElementsByTagName('div')[0]`, qui récupère tous les éléments '`<div>`' sous forme de collection HTML et on prend le premier (index 0).

Et pour finir on pourrait utiliser `document.getElementById('titre')`, si le '`<div>`' avait un attribut 'id' (par exemple '`<div id="titre">`')

Pour accéder au noeud `` DOM on peut utiliser `document.querySelector('ul')` qui sélectionne le premier élément '``' dans le document.

Ou bien `document.getElementsByTagName('ul')[0]` qui récupère tous les `` et on prend le premier.

On peut aussi le trouver par sa position relative au <div> avec document.querySelector('div + ul')(sélectionne un '' qui suit immédiatement un '<div>').

Pour accéder au deuxième '' (celui avec "Pete") on peut utiliser, document.querySelectorAll('li')[1] qui sélectionne tous les '' et on prend le deuxième (index 1, car les tableaux commencent à 0).

Ou bien utiliser, `document.getElementsByTagName('li')[1]` : Pareil mais avec `getElementsByTagName`.

En partant du '', on peut utiliser `ulElement.children[1]` où `ulElement` est la référence au ''.

Avec un sélecteur CSS plus précis : **`document.querySelector('ul li:nth-child(2)')`**.

Comme il n'y a que deux '', le deuxième est aussi le dernier, donc document.querySelector('ul li:last-child') fonctionne.

Sélectionner toutes les cellules diagonales

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

```

panache / ② html
<!DOCTYPE html>
<html>
<body>
| | <script>
// Sélectionner le tableau
let table = document.querySelector('table');

// Parcourir toutes les lignes
for (let i = 0; i < table.rows.length; i++) {
    // Pour chaque ligne, prendre la cellule à l'index i (diagonale principale)
    let td = table.rows[i].cells[i];

    // Vérifier que la cellule existe
    if (td) {
        td.style.backgroundColor = 'red'; // Exactement comme dans la consigne
    }
}
| | </script>
</body>
</html>

```

Recherche d'éléments

```

// 1. Le tableau avec id="age-table"
let table = document.getElementById('age-table');
console.log("1. Tableau :", table);

// 2. Tous les éléments label dans ce tableau
let labels = table.querySelectorAll('label');
console.log(`2. ${labels.length} label(s) trouvé(s)`);

// 3. Le premier td dans ce tableau
let firstTd = table.querySelector('td');
console.log("3. Premier td :", firstTd.textContent);

// 4. Le form avec name="search"
let form = document.querySelector('form[name="search"]');
console.log("4. Formulaire :", form);
// 5. Le premier input dans ce formulaire
let firstInput = form.querySelector('input');
console.log("5. Premier input :", firstInput);

// 6. Le dernier input dans ce formulaire
let inputs = form.querySelectorAll('input');
let lastInput = inputs[inputs.length - 1];
console.log("6. Dernier input :", lastInput);
</script>

```

ⓘ DevTools is now available in French

[Don't show again](#) [Always match Chrome's language](#) [Switch DevTools to French](#)

Elements [Console](#) Sources Network >

Default levels ▾ | No Issues |

1. Tableau :	<table id="age-table">...</table>	table.html:45
2. 3 label(s) trouvé(s)		table.html:49
3. Premier td :	Age:	table.html:53
4. Formulaire :	<form name="search">...</form>	table.html:57
5. Premier input :	<input type="text" name="search">	table.html:60
6. Dernier input :	<input type="submit" value="Search!">	table.html:65

Compter les descendants

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>
```

Cette page indique

1

OK

Balise dans le commentaire

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>
```

Cette page indique

BODY

OK

Où est le document dans la hiérarchie

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 ?

Le document est un Node, est un Document, n'est pas un élément et n'est pas un HTMLElement.

Partie 5

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>
```

```

Users > pierric > Music > monprojet > <--> scriptandui > name.html
<!DOCTYPE html>
<html>
<body>

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

<script>
let element = document.querySelector('[data-widget-name]');
let value = element.getAttribute('data-widget-name');
alert(value); // "menu"
</script>
</body>
</html>

```

Rendre un lien externe 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>

```

<!DOCTYPE html>
<html>
<body>
<a name="list">the list</a>
<ul>
<li><a href="http://google.com">http://google.com</a></li>
<li><a href="/tutorial.html">/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>
// Selectionne tous les liens <a> qui ont un attribut href
let links = document.querySelectorAll('a[href]');

for (let link of links) {
  let href = link.getAttribute('href');

  // Vérifie si c'est un lien externe selon les critères
  if (href.includes('://') && !href.startsWith('http://internal.com')) {
    link.style.color = 'orange';
  }
}
</script>
</body>
</html>

```

la liste

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

Commande 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`

```
ers > pierrette > Music > monprojet > script.html > html > body > sc
<!DOCTYPE html>
<html>
<body>
<script>
    // Les deux commandes équivalentes sont :
    elem.append(document.createTextNode(text)); // Méthode 1
    elem.textContent = text; // Méthode 3

    // La commande différente est :
    elem.innerHTML = text; // Méthode 2

    // Réponse : Les commandes 1 et 3 sont équivalentes
</script>
</body>
</html>
```

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>
```

```
:> Users > pierrette > Music > monprojet > script.html > html > body > sc
1  <!DOCTYPE html>
2  <html>
3  <body>
4  <ol id="elem">
5  |   <li>Hello</li>
6  |   <li>World</li>
7  </ol>
8
9  <script>
10 |
11 |   clear(elem); // efface la liste
12 |
13 </script>
14 </body>
15 </html>
```

```

<!DOCTYPE html>
<html>
<body>
<ol id="elem">
  <li>Hello</li>
  <li>World</li>
</ol>

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

```

1. Hello
2. World

Pourquoi "aaa" reste-t-il

Le texte "aaa" reste visible parce qu'il n'est pas à l'intérieur du tableau (`<table>`). En HTML, un élément `<table>` ne peut contenir directement que des éléments spécifiques comme `<tr>`, `<thead>`, `<tbody>`, `<tfoot>`, etc...

Pourquoi "aaa" reste-t-il ?

Dans l'exemple ci-dessous, l'appel `table.remove()` supprime le tableau du document. mais si vous l'exéutez, 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>

```

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.

```

<!DOCTYPE html>
<html>
<body>

<ul id="list"></ul>

<script>
let ul = document.getElementById('list');

while (true) {
    let content = prompt("Entrez le contenu de l'élément de liste :","");
}

// Si l'utilisateur appuie sur Esc ou entre une chaîne vide
if (content === null || content === '') {
| break;
}

// Crée un élément <li>
let li = document.createElement('li');

// Traite le contenu comme du texte (pas du HTML)
li.textContent = content;

// Ajoute à la liste
ul.append(li);
}

</script>

</body>
</html>

```

- pomme
- tomate
- laitue
- sucre

Cette page indique

Entrez le contenu de l'élément de liste :

OK

Annuler

Créer un arbre à partir de l'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

```

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

```

<!DOCTYPE html>
<html>
<body>

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

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

function createTree(container, data) {
  container.innerHTML = '<ul>' + makeHTML(data) + '</ul>';
}

function makeHTML(data) {
  let html = '';
  for (let key in data) {
    html += '<li>' + key;
    if (Object.keys(data[key]).length > 0) {
      html += '<ul>' + makeHTML(data[key]) + '</ul>';
    }
    html += '</li>';
  }
  return html;
}

createTree(document.getElementById('container'), data);
</script>
</body>
</html>

```

```

function makeHTML(data) {
  let html = '';
  for (let key in data) {
    html += '<li>' + key;
    if (Object.keys(data[key]).length > 0) {
      html += '<ul>' + makeHTML(data[key]) + '</ul>';
    }
    html += '</li>';
  }
  return html;
}

createTree(document.getElementById('container'), data);
</script>
</body>
</html>

```

Afficher les descendants dans un arbre

Afficher les descendants dans un arbre

Il y a un arbre organisé comme un `ul/li` imbriqué.

Écrivez le code qui ajoute à chaque `` le nombre de ses descendants. Sautez les feuilles (nœuds sans enfants).

Le résultat :

- Animals [9]
 - Mammals [4]
 - Cows
 - Donkeys
 - Dogs
 - Tigers
 - Other [3]
 - Snakes
 - Birds
 - Lizards
 - Fishes [5]
 - Aquarium [2]
 - Guppy
 - Angelfish
 - Sea [1]

```
<script>
// Trouve tous les <li> dans la page
let allLi = document.querySelectorAll('li');

// Pour chaque <li>
for (let li of allLi) {
    // Regarde si ce <li> contient un <ul> (donc a des enfants)
    let ulsInside = li.getElementsByTagName('ul');

    // S'il y a au moins un <ul> à l'intérieur
    if (ulsInside.length > 0) {
        // Trouve tous les <li> à l'intérieur de cet élément
        let lisInside = li.getElementsByTagName('li');

        // Ajoute le nombre entre crochets
        li.firstChild.textContent = li.firstChild.textContent + ' [' + lisInside.length + ']';
    }
}
</script>
</body>
</html>
```

• Animals [9]

- Mammals [4]
 - Cows
 - Donkeys
 - Dogs
 - Tigers
- Other [3]
 - Snakes
 - Birds
 - Lizards

- Birds

- Lizards

• Fishes [5]

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

Créer un calendrier

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.

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

<div id="calendar"></div>

<script>
function createCalendar(elem, year, month) {
    let date = new Date(year, month - 1, 1);

    // Création du tableau
    let table = '<table><tr>';

    // En-têtes des jours (lundi à dimanche)
    let days = ['MO', 'TU', 'WE', 'TH', 'FR', 'SA', 'SU'];
    for (let day of days) {
        table += '<th>' + day + '</th>';
    }
    table += '</tr><tr>';

    // Cases vides avant le premier jour
    let firstDay = date.getDay();
    if (firstDay === 0) firstDay = 7; // dimanche devient 7
    for (let i = 1; i < firstDay; i++) {
        table += '<td></td>';
    }

    // Jours du mois
    while (date.getMonth() === month - 1) {
        table += '<td>' + date.getDate() + '</td>';
        date.setDate(date.getDate() + 1);
    }
}
</script>
```

```

// Jours du mois
while (date.getMonth() === month - 1) {
    table += '<td>' + date.getDate() + '</td>';

    // Nouvelle ligne chaque dimanche
    if (date.getDay() === 0) {
        table += '</tr><tr>';
    }

    // Jour suivant
    date.setDate(date.getDate() + 1);
}

table += '</tr></table>';
elem.innerHTML = table;
}

// Exemple : septembre 2012 (mois = 9)
createCalendar(document.getElementById('calendar'), 2012, 9);

```

</script>

</body>

</html>

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
					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

Horloge colorée avec setInterval

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.

```

body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    background-color: #f5f5f5;
}

#clock {
    text-align: center;
    font-family: monospace;
}

#time {
    font-size: 48px;
    font-weight: bold;
    margin-bottom: 20px;
}

#controls {
    font-size: 18px;
}

button {
    font-size: 16px;
    padding: 8px 20px;
    margin: 0 5px;
    border: 1px solid #ccc;
    background-color: white;
}

```

```

button {
    font-size: 16px;
    padding: 8px 20px;
    margin: 0 5px;
    border: 1px solid #ccc;
    background-color: white;
    cursor: pointer;
    border-radius: 3px;
}

button:hover {
    background-color: #f0f0f0;
}

```

```

!DOCTYPE html>
<html>
|   <link rel="stylesheet" href="styles.css">
<body>
|       <div id="clock">
|           <div id="time">
|               <span id="hours">hh</span>:<span id="minutes">mm</span>:<span id="seconds">ss</span>
|           </div>
|           <div id="controls">
|               <button id="startBtn">Start</button> |
|               <button id="stopBtn">Stop</button>
|           </div>
|       </div>
|
|       <script>
|           let timer = null;
|
|           function updateClock() {
|               const now = new Date();
|
|               document.getElementById('hours').textContent =
|                   String(now.getHours()).padStart(2, '0');
|               document.getElementById('minutes').textContent =
|                   String(now.getMinutes()).padStart(2, '0');
|               document.getElementById('seconds').textContent =
|                   String(now.getSeconds()).padStart(2, '0');
|           }

```

```

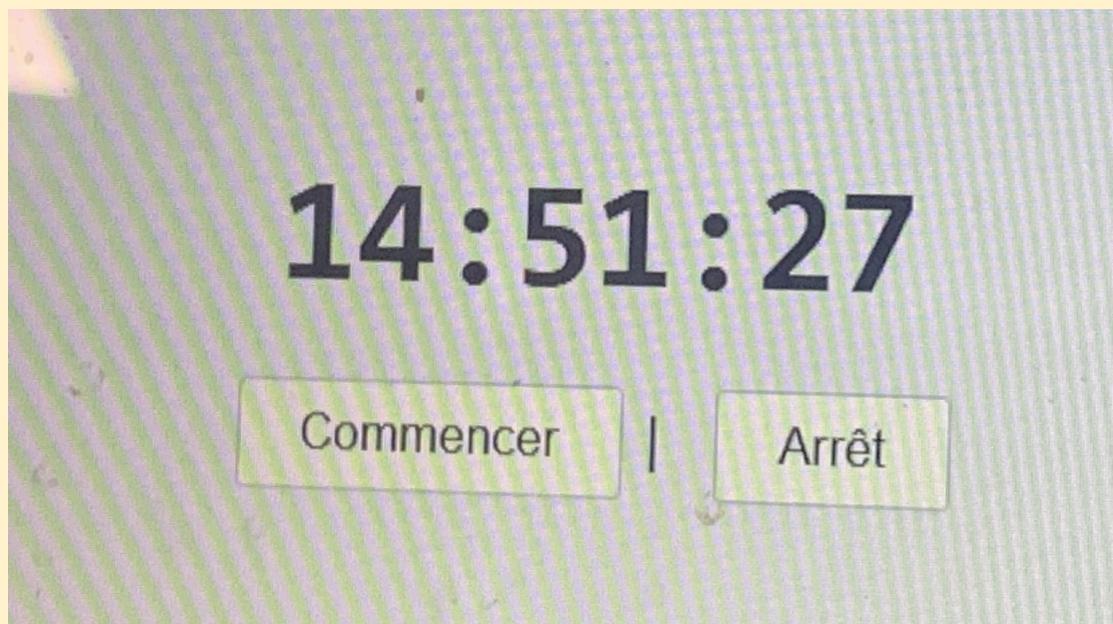
function startClock() {
    if (!timer) {
        updateClock();
        timer = setInterval(updateClock, 1000);
    }
}

function stopClock() {
    if (timer) {
        clearInterval(timer);
        timer = null;
    }
}

document.getElementById('startBtn').onclick = startClock;
document.getElementById('stopBtn').onclick = stopClock;

startClock();
</script>
</body>
</html>

```



Insérer le HTML dans la liste

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>
```

```
<!DOCTYPE html>
<html>
<body>
  <ul id="ul">
    <li id="one">1</li>
    <li id="two">4</li>
  </ul>

  <script>
    document.getElementById('one').insertAdjacentHTML('afterend',
      '<li>2</li><li>3</li>');
  </script>
</body>
</html>
```

- 1
- 2
- 3
- 4

Trier le tableau

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>
```

Name	Surname	Age
John	Smith	10
Pete	Brown	15
Ann	Lee	5
Emma	Wilson	25

```
<html>
<body>
  <table>
    <thead>
      <tr>
        <th>Name</th><th>Surname</th><th>Age</th>
      </tr>
    </thead>
    <tbody>
      <tr><td>John</td><td>Smith</td><td>10</td></tr>
      <tr><td>Pete</td><td>Brown</td><td>15</td></tr>
      <tr><td>Ann</td><td>Lee</td><td>5</td></tr>
      <tr><td>Emma</td><td>Wilson</td><td>25</td></tr>
    </tbody>
  </table>

  <script>
    let table = document.querySelector('table');
    let tbody = table.querySelector('tbody');
    let rows = Array.from(tbody.rows);

    // Trier par la première colonne (Name)
    rows.sort((a, b) => {
      return a.cells[0].textContent.localeCompare(b.cells[0].textContent);
    });

    // Remettre les lignes dans l'ordre trié
    rows.forEach(row => tbody.appendChild(row));
  </script>
```

Name	Surname	Age
Ann	Lee	5
Emma	Wilson	25
John	Smith	10
Pete	Brown	15

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 });
```

```
<!DOCTYPE html>
<html>
<body>
<!DOCTYPE html>
<html>
<body>
  <button onclick="test()">Tester la notification</button>

  <script>
    function showNotification(options) {
      // 1. Crée la div de notification
      let notification = document.createElement('div');

      // Ajoute la classe de base
      notification.className = 'notification';

      // Ajoute la classe optionnelle
      if (options.className) {
        notification.className += ' ' + options.className;
      }

      // Définit le contenu HTML
      notification.innerHTML = options.html;

      // Ajoute au document
      document.body.appendChild(notification);
    }
  </script>
```

```
// Supprime après 1.5 secondes
setTimeout(() => {
  notification.remove();
}, 1500);

// Fonction de test
function test() {
  showNotification({
    top: 10,
    right: 10,
    html: "Hello!",
    className: "welcome"
  });
}</script>
```



Quel est le défilement à partir du bas ?

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`.

```
<!DOCTYPE html>
<html>
<body>
    <!-- Un élément avec défilement -->
    <div id="myElement" style="height: 100px; width: 200px; overflow: auto; border: 1px solid black;">
        <div style="height: 300px;">
            Contenu très long qui nécessite un défilement...
        </div>
    </div>

    <script>
        // Ton code ici
        let elem = document.getElementById('myElement');

        // Code pour obtenir le scrollBottom
        let scrollBottom = elem.scrollHeight - elem.scrollTop - elem.clientHeight;

        // Version avec vérification pour éviter les valeurs négatives
        function getScrollBottom(elem) {
            return Math.max(0, elem.scrollHeight - elem.scrollTop - elem.clientHeight);
        }

        // Exemple d'utilisation
        console.log("scrollBottom:", getScrollBottom(elem));
    </script>
</body>
</html>
```

Contenu très long qui nécessite un défilement...

Quelle est la largeur de la barre de défilement ?

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.

```

<!DOCTYPE html>
<html>
<body>
    <div id="result"></div>

    <script>
        // Crée un div avec défilement
        let div = document.createElement('div');
        div.style.cssText =
            `width: 100px;
            height: 100px;
            overflow: scroll;
            position: absolute;
            top: -9999px;
            left: -9999px;
            `;
        document.body.appendChild(div);

        // Calcule la largeur de la barre de défilement
        let scrollbarWidth = div.offsetWidth - div.clientWidth;

        // Supprime le div
        div.remove();

        // Affiche le résultat
        document.getElementById('result').textContent =
            `Largeur de la barre de défilement : ${scrollbarWidth}px`;

        console.log(`Largeur barre de défilement : ${scrollbarWidth}px`);
    </script>

```

Largeur de la barre de défilement : 15px

Trouver les coordonnées de la fenêtre du champ

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)

champ.html

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.

```

html, body {
    margin: 0;
    padding: 0;
}

#coords {
    margin-bottom: 10px;
}

#field {
    width: 200px;
    height: 150px;
    border: 10px groove black;
    background-color: #00FF00;
    position: relative;
    overflow: hidden;
    cursor: pointer;
}

.triangle-right {
    position: absolute;
    width: 0;
    height: 0;
    border-top: 6px solid transparent;
    border-bottom: 6px solid transparent;
    border-left: 20px solid red;
    font-size: 0;
    line-height: 0;
}

```

```

.triangle-right:after {
  content: attr(style);
  position: relative;
  top: -6px;
  left: 8px;
  font-size: 12px;
  color: black;
}

<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="styles.css">
</head>
<body>

  <div id="coords">(les coordonnées du clic apparaissent ici)</div>
  <div id="field">
    ...
  </div>

  <div class="triangle-right" style="left:-20px;top:-176px">1</div>
  <div class="triangle-right" style="left:-10px;top:-178px">3</div>
  <div class="triangle-right" style="left:10px;top:-48px">4</div>
  <div class="triangle-right" style="left:20px;top:-42px">2</div>

  <script>
    // Fonction pour afficher les coordonnées du clic
    document.onclick = function(e) {
      document.getElementById('coords').innerHTML = e.clientX + ':' + e.clientY;
    }
  </script>

```

```

let field = document.getElementById('field');

// 1. Coin extérieur supérieur gauche
let rect = field.getBoundingClientRect();
let answer1 = [rect.left, rect.top];

// 2. Coin extérieur inférieur droit
let answer2 = [rect.right, rect.bottom];

// 3. Coin intérieur supérieur gauche
let answer3 = [rect.left + field.clientWidth, rect.top + field.clientHeight];

// 4. Coin intérieur inférieur droit
let answer4 = [
  rect.left + field.clientWidth + field.clientWidth,
  rect.top + field.clientHeight + field.clientHeight
];

// Pour vérifier dans la console
console.log('1. Extérieur supérieur gauche:', answer1);
console.log('2. Extérieur inférieur droit:', answer2);
console.log('3. Intérieur supérieur gauche:', answer3);
console.log('4. Intérieur inférieur droit:', answer4);

```

189:393



Afficher une note près de l'élément

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 incident voluptatum error fugiat animi amet! Odio temporibus nulla id unde quaerat dignissimos enim nisi rem provident molestias sit tempore omnis recusandae esse sequi 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`

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

`note below`

```

<script>
// Fonction pour positionner la note
function positionAt(anchor, position, elem) {
    let anchorCoords = anchor.getBoundingClientRect();
    let scrollY = window.pageYOffset || document.documentElement.scrollTop;
    let scrollX = window.pageXOffset || document.documentElement.scrollLeft;

    if (position == "top") {
        elem.style.left = (anchorCoords.left + scrollX + anchorCoords.width/2 - elem.offsetWidth/2) + "px";
        elem.style.top = (anchorCoords.top + scrollY - elem.offsetHeight - 10) + "px";
    }

    if (position == "right") {
        elem.style.left = (anchorCoords.right + scrollX + 10) + "px";
        elem.style.top = (anchorCoords.top + scrollY + anchorCoords.height/2 - elem.offsetHeight/2) + "px";
    }

    if (position == "bottom") {
        elem.style.left = (anchorCoords.left + scrollX + anchorCoords.width/2 - elem.offsetWidth/2) + "px";
        elem.style.top = (anchorCoords.bottom + scrollY + 10) + "px";
    }
}

```

```

// Fonction pour afficher une note
function showNote(anchor, position, html) {
    let note = document.createElement('div');
    note.className = "note";
    note.innerHTML = html;
    document.body.append(note);
    positionAt(anchor, position, note);
}

// Afficher les notes
let quote = document.getElementById('quote');

// Note au-dessus
showNote(quote, "top", "note above");

// Note en dessous
showNote(quote, "bottom", "note below");

// Note à droite
showNote(quote, "right", "note at the right");
</script>

```

```

body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
}

.note {
    position: absolute;
    z-index: 1000;
    padding: 10px;
    border: 1px solid black;
    background: white;
    text-align: center;
    font: italic 14px Georgia;
    width: 200px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.2);
}

blockquote {
    background: #f9f9f9;
    border-left: 10px solid #ccc;
    margin: 40px 0;
    padding: 20px;
    position: relative;
}

```

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

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 incident
 voluptatum error fugiat animi amet! Odio temporibus nulla id unde quaerat dignissimos enim nisi rem provident
 molestias sit tempore omnis recusandae esse sequi officia sapiente.

Cacher sur clic

Cacher sur clic

Ajouter JavaScript à la `button` pour faire `<div id="text">` disparaître quand on clique dessus.

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

<button onclick="changer()">Bouton</button>

<div id="text">Texte qui apparaît et disparaît</div>

<script>
function changer() {
    let texte = document.getElementById("text");
    if (texte.style.display === "none") {
        texte.style.display = "block";
    } else {
        texte.style.display = "none";
    }
}
</script>

</body>
</html>|
```

Cacher/Montrer

Texte qui apparaît et disparaît

Cacher/Montrer

Se cacher

Se cacher

Crée un bouton qui se cache tout seul au clic.

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

<button onclick="cacher()">se cache</button>

<script>
function cacher() {
    document.getElementsByTagName("button")[0].style.display = "none";
}
</script>

</body>
</html>|
```

Quels handlers exécutent ?

Quels handlers exécutent?

Il y a un bouton dans la variable. Il n'y a aucun handlers dessus.

Quels handlers s'exécutent au clic après le code suivant? Quelles alertes apparaissent?

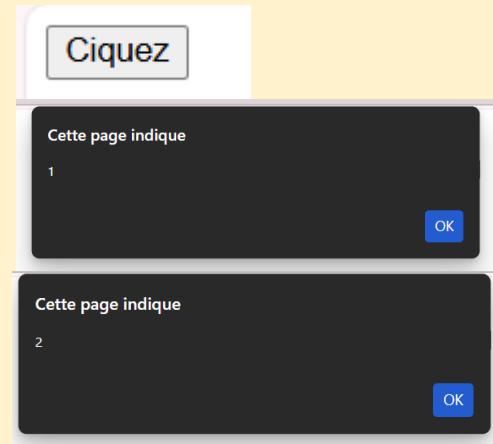
```
1 button.addEventListener("click", () => alert("1"));
2
3 button.removeEventListener("click", () => alert("1"));
4
5 button.onclick = () => alert(2);
```

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

<button id="btn">Cliquez</button>

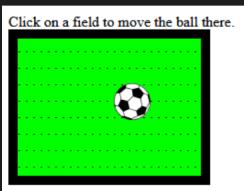
<script>
let button = document.getElementById('btn');
button.addEventListener("click", () => alert("1"));
button.removeEventListener("click", () => alert("1"));
button.onclick = () => alert(2);
</script>

</body>
</html>
```



Deux alertes apparaissent au clic : "1" puis "2".

Ballon



Exigences:

- Le centre de la balle devrait passer exactement sous le pointeur au clic (si possible sans franchir le bord du champ).
- L'animation CSS est la bienvenue.
- La balle ne doit pas franchir les limites du terrain.
- Quand la page est défilée, rien ne devrait se casser.

Notes:

- Le code doit aussi fonctionner avec différentes tailles de balle et de champ, sans être lié à des valeurs fixes.
- Propriétés d'utilisation `event.clientX/event.clientY` pour les coordonnées de clic.

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#field {
    width: 400px;
    height: 300px;
    border: 10px solid black;
    background-color: #00FF00;
    position: relative;
    overflow: hidden;
    cursor: pointer;
}
#ball {
    position: absolute;
    transition: left 0.5s, top 0.5s;
}
</style>
</head>
<body>
<div id="field">
    
</div>
<script>
let field = document.getElementById('field');
let ball = document.getElementById('ball');
```

Déplacer le ballon à travers le terrain

```
<div id="field">
  
</div>

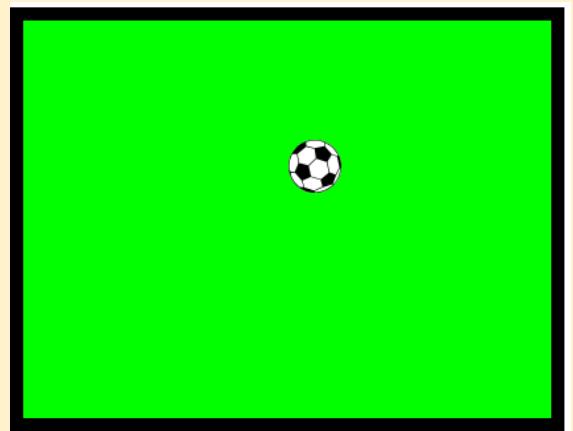
<script>
  let field = document.getElementById('field');
  let ball = document.getElementById('ball');

  // Position initiale
  ball.style.left = (field.clientWidth - ball.offsetWidth) / 2 + 'px';
  ball.style.top = (field.clientHeight - ball.offsetHeight) / 2 + 'px';

  field.addEventListener('click', function(e) {
    // Position relative au terrain
    let x = e.clientX - field.getBoundingClientRect().left - field.clientLeft - ball.offsetWidth/2;
    let y = e.clientY - field.getBoundingClientRect().top - field.clientTop - ball.offsetHeight/2;

    // Limites
    x = Math.max(0, Math.min(x, field.clientWidth - ball.offsetWidth));
    y = Math.max(0, Math.min(y, field.clientHeight - ball.offsetHeight));

    // Déplacement
    ball.style.left = x + 'px';
    ball.style.top = y + 'px';
  });
</script>
```



Créez un menu glissant

Créez un menu glissant

Crée un menu qui s'ouvre/replie au clic:

Créez un menu glissant

Crée un menu qui s'ouvre/replie au clic:

▶ Sweeties (click me)!

▼ Sweeties (click me)!
Cake
Donut
Honey

► Sweeties (click me)!

▼ Sweeties (click me)!
Cake
Donut
Honey

Ajoutez un bouton de fermeture

Horse [x]
The horse is one of two extant subspecies of Equus ferus. It is an odd-toed ungulate mammal belonging to the taxonomic family Equidae. The horse has evolved over the past 45 to 55 million years from a small multi-toed creature, Eohippus, into the large, single-toed animal of today.

Donkey [x]
The donkey or ass (Equus africanus asinus) is a domesticated member of the horse family, Equidae. The wild ancestor of the donkey is the African wild ass, E. africanus. The donkey has been used as a working animal for at least 5000 years.

Cat [x]
The domestic cat (Latin: Felis catus) is a small, typically furry, carnivorous mammal. They are often called house cats when kept as indoor pets or simply cats when there is no need to distinguish them from other felids and felines. Cats are often valued by humans for companionship and for their ability to hunt vermin.

```
<script>
  // Ajouter un bouton de fermeture à chaque panneau
  const panes = document.querySelectorAll('.pane');

  panes.forEach(pane => {
    // Créer le bouton
    const button = document.createElement('button');
    button.className = 'remove-button';
    button.textContent = '[x]';
    button.style.position = 'absolute';
    button.style.top = '10px';
    button.style.right = '10px';

    // Ajouter le bouton au panneau
    pane.style.position = 'relative';
    pane.appendChild(button);

    // Ajouter l'élément de clic
    button.onclick = function() {
      pane.style.display = 'none';
    };
  });
</script>
```

Donkey [x]
The donkey or ass (Equus africanus asinus) is a domesticated member of the horse family, Equidae. The wild ancestor of the donkey is the African wild ass, E. africanus. The donkey has been used as a working animal for at least 5000 years.

Cat [x]
The domestic cat (Latin: Felis catus) is a small, typically furry, carnivorous mammal. They are often called house cats when kept as indoor pets or simply cats when there is no need to distinguish them from other felids and felines. Cats are often valued by humans for companionship and for their ability to hunt vermin.

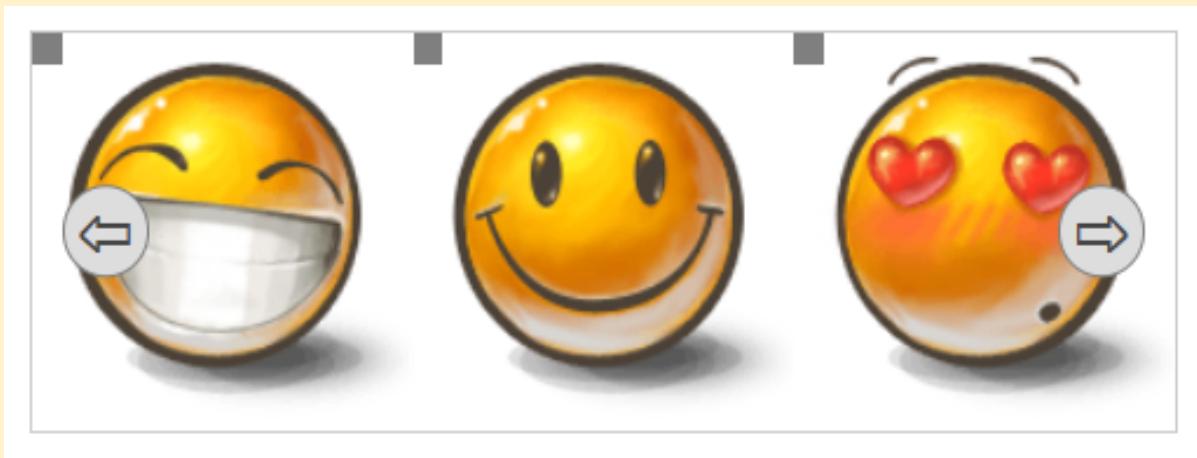
Carrousel

Carrousel

Créez un « carrousel » – un ruban d'images que l'on peut faire défiler en cliquant sur des flèches.



P.S. Pour cette tâche, la structure HTML/CSS représente en fait 90% de la solution.



```
<script>
  const carousel = document.getElementById('carousel');
  const ul = carousel.querySelector('ul');
  const prevBtn = carousel.querySelector('.prev');
  const nextBtn = carousel.querySelector('.next');

  let position = 0;
  const step = 130;

  prevBtn.onclick = function() {
    position += step;
    position = Math.min(position, 0);
    ul.style.marginLeft = position + 'px';
  };

  nextBtn.onclick = function() {
    position -= step;
    const max = -(ul.children.length * step - carousel.offsetWidth);
    position = Math.max(position, max);
    ul.style.marginLeft = position + 'px';
  };

  let i = 1;
  for(let li of ul.querySelectorAll('li')) {
    li.style.position = 'relative';
    li.insertAdjacentHTML('beforeend', `<span style="position: absolute; left:0; top:0; background:rgba(0,0,0,0.5); color:white; padding:5px;">${i}</span>`);
    i++;
  }
</script>
```

Partie 6

Liste sélectionnable

Liste sélectionnable

Créer une liste dont les éléments sont sélectionnables, comme dans le gestionnaire de fichiers

- Un click sur un élément de la liste sélectionne seulement cet élément (ajoute la classe `.selected`), désélectionne tous les autres.
- Si un click est effectué avec `Ctrl` (`Cmd` sur Mac), alors la sélection est inversée sur l'élément, mais les autres éléments ne sont pas modifiés

Cliquez sur un élément de la liste pour le sélectionner.

- Christopher Robin
- Winnie-the-Pooh
- Tigger
- Kanga
- Rabbit. Just rabbit.

Cliquez sur un élément de la liste pour le sélectionner.

- Christopher Robin
- Winnie-the-Pooh
- **Trigger**
- Kanga
- Rabbit. Just rabbit.

```
<script>
  const ul = document.getElementById('ul');

  ul.addEventListener('click', function(event) {
    event.preventDefault();

    const ctrlPressed = event.ctrlKey || event.metaKey;

    if (!ctrlPressed) {
      const items = ul.querySelectorAll('li');
      items.forEach(item => item.classList.remove('selected'));
    }

    if (event.target.tagName === 'LI') {
      event.target.classList.toggle('selected');
    }
  });
</script>
```

Info-bulle “Intelligente”

Info-bulle “Intelligente”

Écrire JavaScript qui montre une infobulle sur un élément avec l'attribut data-tooltip. La valeur de cet attribut devrait devenir le texte de l'infobulle.

C'est comme la tâche Comportement info-bulle, mais ici les éléments annotés peuvent être imbriqués. L'infobulle la plus profondément imbriquée est montrée.

Une seule infobulle peut apparaître en même temps.

Veuillez passez votre curseur sur le texte en bleu est l'info bulle s'affichera.....
[Hover over me](#)

