Workshop III

Les délais

Il est possible d'exécuter ou de répéter une instruction après un délai avec les méthodes setTimeout, clearTimeout, setInterval et clearInterval.

Les délais : setTimeout

Il est possible d'exécuter une instruction après un délai avec setTimeout et clearTimeout.

```
var identifiant = setTimeout(instruction, delai, param1, param2, …, paramN);
clearTimeout(identifiant);
```

La durée (delai) est en millisecondes.

Les paramètres sont facultatifs.

```
var timer;
if (timer) {
    clearTimeout(timer);
}
timer = setTimeout(function() {
    console.log(42);
}, 1000);
// (Après 1 seconde) => 42
```

Les délais : setInterval

Il est possible d'exécuter une instruction de manière répétitive après un délai avec setInterval et clearInterval.

```
var identifiant = setInterval(instruction, delai, param1, param2, ..., paramN);
clearInterval(identifiant);
```

La durée (delai) est en millisecondes.

Les paramètres sont facultatifs.

```
var timer;
if (timer) {
    clearInterval(timer);
}
timer = setInterval(function() {
    console.log(42);
}, 1000);
// (Tout les secondes) => 42
```

Les closures

Les closures permettent d'enfermer une ou plusieurs variables dont les valeurs ne peuvent pas être modifiées durant l'exécution des instructions.

```
function maFonction(nombre) {
    function addition(valeur) {
    return nombre + valeur;
    }
return addition;
}

var closure = maFonction(40);
console.log(closure(2));
// => 42
```

nombre est la variable qui est enfermé.

Les closures : les délais

Les closures sont très utiles pour enfermer des variables qui ont des valeurs susceptibles de changer avant l'exécution du délai.

```
var maTable = ['Hello', 'world'];
for (let item of maTable) {
    setTimeout(function() {
        console.log(item);
    }, 1000);
}
// => Hello && => world

for (var item of maTable) {
    setTimeout(function() {
        console.log(item);
    }, 1000);
}
// => world && => world

for (var item of maTable) {
    setTimeout((function(item) {
        return function() {
            console.log(item);
        };
    }) (item), 1000);
}
// => Hello && => world
```

Le DOM
(Document Object Model)

Le DOM est la structure d'une page web. L'arborescence du document est organiser de manière à pouvoir entrer et sortir de balise en balise.

Comme Javascript, chaque balise est un objet qui a des propriétés et des méthodes.

Pour accéder à la racine de l'arborescence, on utilise la propriété document

```
console.log(document.body);
// => <body> ... </body>
```

Chaque élément est un nœud (node), ces nodes peuvent avoir des enfants (children).

Le DOM : Outils de développement

Il est conseillé d'utiliser l'Outil de développement présent sur Google Chrome qui permet beaucoup plus de chose que celui des autres navigateurs.

L'onglet Properties permet de visualiser directement l'intégralité des propriétés et des méthodes.

La console permet de visualiser le retour de l'instruction console.log(); et d'insérer directement des instructions sur le navigateur.

Le DOM : chemin d'accès

Il existe différentes solutions pour accéder à chaque objet.

```
<body>
    <header>Hello World !</header>
    <div id="monID" class="class1 class2">42</div>
    <div class="class1 class3">100</div>
</body>
```

```
console.log(document.body.children[0]);
console.log(document.body.firstElementChild);
// => <header> ... </header>
console.log(document.body.lastElementChild);
// => <div class="class1 class3">100</div>
console.log(document.getElementsByClassName('class1'));
// => Object {
    0 : <div id="monID" class="class1 class2">42</div>
    1 : <div class="class1 class3">100</div>
}
console.log(document.getElementsByClassName('class1')[0]);
console.log(document.getElementsByClassName('class1').monID;
// => <div id="monID" class="class1 class2">42</div>
console.log(document.getElementById('monID'));
// => <div id="monID" class="class1 class2">42</div>
console.log(document.getElementById('monID').parentElement);
// => <body> ... </body>
console.log(document.getElementById('monID').nextElementSibling);
// => <div class="class1 class3">100</div>
console.log(document.getElementById('monID').childNodes[0]);
// => '42'
```

Le DOM : chemin d'accès

Il existe différentes solutions pour accéder à chaque objet.

```
<body>
    <header>Hello World !</header>
    <div id="monID" class="class1 class2">42</div>
    <div id="monID2" class="class1">
        <div class="class3">100</div>
        <div id="monID3" class="class1 class2">8</div>
    </div>
</body>
```

```
console.log(document.querySelector('.class1'));
console.log(document.querySelector('#monID'));
// => <div id="monID" class="class1 class2">42</div>
console.log(document.querySelector('.class1 .class1'));
// => <div id="monID3" class="class1 class2">8</div>
console.log(document.querySelectorAll('.class2'));
// => Object {
    0 : <div id="monID" class="class1 class2">42</div>
    1 : <div id="monID3" class="class1 class2">8</div>
}
console.log(document.querySelectorAll('.class2')[0]);
// => <div id="monID" class="class1 class2">42</div>
console.log(document.querySelectorAll('div'));
// => Object {
    0 : <div id="monID" class="class1 class2">42</div>
    1 : <div id="monID2" class="class1"> ... </div>
    3 : <div class="class3">100</div>
    4 : <div id="monID3" class="class1 class2">8</div>
}
```

Le DOM : chemin d'accès

Les propriétés firstElementChild et lastElementChild retournent respectivement le premier objet enfant et le dernier objet enfant d'un objet parent.

La méthode getElementsByClassName retourne un tableau des objets qui possèdent la/les attribut(s) class qui correspondent au masque de recherche.

On peut parcourir le tableau via l'index ou via la key si l'objet possède un attribut id.

```
document.getElementsByClassName('class1');
document.getElementsByClassName('class1 class2');
document.getElementsByClassName('class1 class2', 'class3');
```

La méthode getElementByID retourne le premier objet trouvé qui possède l'attribut id qui correspond au masque de recherche.

La propriété parentElement retourne l'objet parent de l'objet dans lequel on se trouve.

Les propriétés nextElementSibling et previousElementSibling retournent respectivement l'objet enfant suivant et l'objet enfant précédent de l'objet parent dans lequel on se trouve.

La propriété childNodes retourne un tableau des objets enfants de l'objet dans lequel on se trouve. Un saut de ligne est considéré comme un objet enfant.

Le DOM : chemin d’accès

La méthode querySelector retourne le premier objet qui correspond au masque de recherche.

La méthode querySelectorAll retourne un tableau des objets qui correspondent au masque de recherche.

On peut parcourir le tableau via l’index.

Le DOM : Manipuler l'attribut class

La propriété classList possède différentes méthodes pour manipuler les class.

```
<div id="monID" class="class1">42</div>
```

La méthode add ajoute une ou plusieurs class.

```
document.getElementById('monID').classList.add('class2', 'class3');
// => <div id="monID" class="class1 class2 class3">42</div>
```

La méthode remove supprime une ou plusieurs class.

```
document.getElementById('monID').classList.remove('class2', 'class3');
// => <div id="monID" class="class1">42</div>
```

La méthode replace remplace une class par une autre.

```
document.getElementById('monID').classList.replace('class1', 'class2');
// => <div id="monID" class="class2">42</div>
```