

Langage de Script JavaScript

Partie 2 : Gestion des Événements

Pr. Mohammed SALIHOUN

m.salihoun@emsi.ma

version éditée par: Dr. Mohammed BELATAR

3ème Année - Ingénierie en Informatique et Réseaux

4 octobre 2022

Table of Contents

1. JavaScript discret

- Qu'est-ce que le JavaScript discret
- Relier le code aux évènements depuis JavaScript
- L'évènement `window.onload`
- Fonctions anonymes
- Fonctions imbriquées

2. Les Timers

- Définir un timer

- Passer des paramètre à un timer

3. Compléments sur le DOM

- Le mot-clef `this`
- Liaison d'un handler avec `this`
- Les Objets en général
- Creation de nouveaux éléments
- Modifier l'arbre DOM
- Selection d'objets DOM
- Lire ou modifier les classes CSS
- L'attribut `classList`

Table of Contents

1. JavaScript discret

- Qu'est-ce que le JavaScript discret
- Relier le code aux évènements depuis JavaScript
- L'évènement `window.onload`
- Fonctions anonymes
- Fonctions imbriquées

2. Les Timers

- Définir un timer

- Passer des paramètres à un timer

3. Compléments sur le DOM

- Le mot-clef `this`
- Liaison d'un handler avec `this`
- Les Objets en général
- Création de nouveaux éléments
- Modifier l'arbre DOM
- Sélection d'objets DOM
- Lire ou modifier les classes CSS
- L'attribut `classList`

JavaScript discret

Qu'est-ce que le JavaScript discret

JavaScript discret :

- HTML sans code JS dans les balises.
- On utilise le DOM pour associer le JS aux évènements et aux éléments.

Permet de **séparer** le code en 3 tiers :

- Contenu (HTML) : qu'est-ce que c'est ?
- Présentation (CSS) : comment est-ce présenté ?
- Comportement (JS) : comment ça réagit aux évènements ?

JavaScript discret

Exemple d'un JavaScript indiscret

HTML (Exemple)

```
<button onclick="okayClick();" >OK</button>
```

JS (Exemple)

```
// activée quand le bouton OK est cliqué  
function okayClick() {  
    alert("booyah");  
}
```

- **Mauvais style** (le HTML et le JS sont mélangés).
- **But** : Supprimer tout le JS mélangé au HTML.

JavaScript discret

Relier le code aux évènements depuis JavaScript

JS (Modèle)

```
let objectName.onevent = function ;
```

HTML (Exemple)

```
<button id="ok">OK</button>
```

JS (Exemple)

```
let okButton = document.getElementById("ok");  
okButton.onclick = okayClick ;
```

- On peut relier du code JS (une fonction) à un évènement sur un élément HTML en **JS**.
- Remarquez qu'il ne faut pas mettre de parenthèses après le nom de la fonction !

JavaScript discret

Comment et quand le code est exécuté

HTML (Exemple)

```
<html>
  <head>
    <script src="myFile.js" type="text/javascript"></script>
  </head>
  <body> ... </body>
</html>
```

myFile.js (Exemple)

```
let x = 3;
function f(n) return n + 1;
function g(n) return n - 1;
x = f(x);
```

JavaScript discret

Comment et quand le code est exécuté

Le code présent dans le fichier JS est exécuté quand le navigateur charge la balise `script`.

- Les variables sont déclarées.
- Les fonctions sont déclarées (et pas exécutées).

A cet instant, le navigateur n'a pas encore chargé l'élément `body` du document HTML.

- Aucun objet DOM correspondant aux éléments HTML n'a encore été créé.

JavaScript discret

L'évènement window.onload

JS (Modèle)

```
function functionName() {  
    // put code to initialize the page here  
}  
  
// instruct window to run the function when the page has loaded  
window.onload = functionName;
```

Il existe un évènement appelé `window.onload` qui est déclenché à la fin du chargement de la page.

Il est possible d'attacher une fonction JS à cet évènement et donc d'exécuter un code donné JS juste après le chargement de la page.

JavaScript discret

Ajout d'un handler en JavaScript discret

HTML (Exemple)

```
<button id="ok">OK</button>
```

```
<!-- (1) -->
```

JS (Exemple)

```
// called when page loads; sets up event handlers
```

```
function pageLoad() {
```

```
    let ok = document.getElementById("ok"); // (3)
```

```
    ok.onclick = okayClick;
```

```
}
```

```
function okayClick() {
```

```
    alert("booyah"); // (4)
```

```
}
```

```
window.onload = pageLoad; // (2)
```

JavaScript discret

Fonctions anonymes

JS (Modèle)

```
function (parameters) {  
    ... statements ... ;  
}
```

- JS permet de définir des fonctions anonymes.
- Une fonction sans nom.
- Elle peut être affectée à une variable, à un handler, etc...

JavaScript discret

Fonctions anonymes et window.onload

JS (Exemple)

```
window.onload = function() {  
  let ok = document.getElementById("ok"); ok.onclick = okayClick;  
};  
function okayClick() { alert("booyah"); }
```

ou bien, plus concis, mais peut-être plus difficile à lire :

JS (Exemple)

```
window.onload = function() {  
  document.getElementById("ok").onclick = function() {  
    alert("booyah");  
  };  
};
```

JavaScript discret

Fonctions définies dans une fonction

JS (Exemple)

```
function everything() {  
    let count = 0;  
    function incr(n) {  
        count += n;  
    }  
    function reset() {  
        count = 0;  
    }  
    incr(4);  
    console.log(count);  
}  
everything();
```

// call the function to run the code

JavaScript discret

Fonctions définies dans une fonction

Dans l'exemple précédant, tout le code est à l'intérieur d'une fonction.

Les variables et les autres fonctions définies à l'intérieure sont locales à la fonction englobante.

Dans ce code, il n'y a qu'une unique entité globale, c'est la fonction `everything`.

JavaScript discret

Le modèle pour window.onload

JS (Exemple)

```
window.onload = function() {  
  
    function okayClick() {  
        alert("booyah");  
    }  
  
    let ok = document.getElementById("ok");  
    ok.onclick = okayClick;  
};
```

Dans cet exemple, tout le code est à l'intérieur d'une fonction anonyme directement associée à l'évènement `window.onload`.

Dans ce code, il n'y a plus d'entité globale !

Table of Contents

1. JavaScript discret

- Qu'est-ce que le JavaScript discret
- Relier le code aux évènements depuis JavaScript
- L'évènement `window.onload`
- Fonctions anonymes
- Fonctions imbriquées

2. Les Timers

- Définir un timer

- Passer des paramètres à un timer

3. Compléments sur le DOM

- Le mot-clef `this`
- Liaison d'un handler avec `this`
- Les Objets en général
- Création de nouveaux éléments
- Modifier l'arbre DOM
- Sélection d'objets DOM
- Lire ou modifier les classes CSS
- L'attribut `classList`

Les Timers

Définir un timer

- **setTimeout(function, delayMS)** : Appelle une fonction après un certain délai.
- **setInterval(function, delayMS)** : Appelle une fonction périodiquement à un intervalle de temps donné.
- **clearTimeout(timerID)**, **clearInterval(timerID)** : Arrête et détruit un timer.

Les fonctions `setTimeout` and `setInterval` retournent une valeur unique qui identifie le *timer*.

- Cette valeur est passée en paramètre à la fonction `clearTimeout` ou `clearInterval` pour arrêter et détruire le *timer*.

Les Timers

Exemple avec setTimeout

HTML (Output)

```
<button id="clickme">Click me !</button>  
<span id="outputText"></span>
```

JS (Exemple)

```
window.onload = function() {  
    document.getElementById("clickme").onclick = delayedMessage;  
};  
  
function delayedMessage() {  
    document.getElementById("outputText").innerHTML = "Wait for it...";  
    setTimeout(sayBooyah, 5000);  
}  
  
function sayBooyah() { // called when the timer goes off  
    document.getElementById("outputText").innerHTML = "BOOYAH!";  
}
```

Les Timers

Exemple avec setInterval

JS (Exemple)

```
let timer = null; // stores ID of interval timer

function delayMsg2() {
  timer = setInterval(mowgli, 1000);
}

function mowgli() {
  document.getElementById("outputText").innerHTML += "Mowgli!"
}
```

Les Timers

Exemple avec clearInterval

JS (Exemple)

```
let timer = null; // stores ID of interval timer

function toggleDelayMessage() {
  if (timer === null) {
    timer = setInterval(mowgli, 1000);
  } else {
    clearInterval(timer);
    timer = null;
  }
}

function mowgli() {
  document.getElementById("outputText").innerHTML += "Mowgli!"
}
```

Les Timers

Passer des paramètres à un timer

JS (Exemple)

```
function delayedMultiply() {  
    // 6 and 7 are passed to multiply when timer goes off  
    setTimeout(multiply, 2000, 6, 7);  
}  
  
function multiply(a, b) {  
    alert(a * b);  
}
```

On place les paramètres après l'intervalle de temps, cela ne marche pas sous IE : il faut créer une fonction spéciale pour ça !

Pourquoi ne pas écrire simplement :

JS (Exemple)

```
setTimeout(multiply(6, 7), 2000);
```

Table of Contents

1. JavaScript discret

- Qu'est-ce que le JavaScript discret
- Relier le code aux évènements depuis JavaScript
- L'évènement `window.onload`
- Fonctions anonymes
- Fonctions imbriquées

2. Les Timers

- Définir un timer

- Passer des paramètre à un timer

3. Compléments sur le DOM

- Le mot-clef `this`
- Liaison d'un handler avec `this`
- Les Objets en général
- Creation de nouveaux éléments
- Modifier l'arbre DOM
- Selection d'objets DOM
- Lire ou modifier les classes CSS
- L'attribut `classList`

Compléments sur le DOM

Le mot-clef this

JS (Modèle)

```
this.fieldName           // access field  
this.fieldName = value ;  // modify field  
this.functionName(parameters) ; // call method
```

Le code JS s'exécute toujours dans le contexte d'un objet.

Par défaut, cet objet est l'objet `window`.

- Toutes les variables globales et les fonctions sont dans l'objet `window`.

Le mot-clef `this` désigne l'objet courant (donc l'objet `window` par défaut).

Compléments sur le DOM

Liaison d'un handler avec this

JS (Modèle)

```
window.onload = function() {  
    document.getElementById("textbox").onmouseout = booyah ;  
};  
function booyah() { // booyah knows what object it was called on  
    this.value = "booyah" ;  
}
```

Les *handlers* peuvent être attachés aux évènements sur les éléments de façon *discrète* (fonction anonyme).

Dans la fonction *handler*, l'élément déclancheur est référencé par `this`.

Compléments sur le DOM

this peut faire référence à n'importe quel objet

JS (Modèle)

```
let objet1 = {  
  cne : 'AB32012',  
  nom : 'Ali',  
  prenom : 'Ahmed',  
  infoCNE : function(){return "Le numéro CNE :"+ this.cne;}  
};  
let var1 = objet1.infoCNE();
```

On peut créer directement un *Object* sans définir sa classe. Il suffit de déclarer des attributs et des méthodes entre `{` et `}` (comme on peut définir un tableau entre `[ ]`).

On peut faire appel aux attributs/méthodes de l'intérieur avec `this`, de l'extérieur on utilise le nom de l'objet.

Compléments sur le DOM

Creation de nouveaux éléments

JS (Modèle)

```
// create a new <h2> node  
let newHeading = document.createElement("h2");  
newHeading.innerHTML = "This is a heading";  
newHeading.style.color = "green";
```

- document.**createElement("tag")** : crée un nouvel élément (vide) et retourne l'objet DOM correspondant.
- document.**createElement("texte")** : crée un nouvel élément contenant le texte et retourne l'objet DOM correspondant.

Une fois créé, l'élément n'est pas encore ajouté au document HTML principal. Il faut l'ajouter comme nouvel enfant d'un élément existant.

Compléments sur le DOM

Modifier l'arbre DOM

JS (Modèle)

```
let p = document.createElement("p");  
p.innerHTML = "A paragraph !" ;  
document.getElementById("main").appendChild(p) ;
```

On peut appliquer ces méthodes sur n'importe quel objet DOM.

Nom	Description
<code>appendChild</code> (node)	Ajoute l'élément paramètre à la suite des enfants existants
<code>insertBefore</code> (new, old)	Ajoute l'élément new juste avant l'élément enfant old
<code>removeChild</code> (node)	Supprime l'élément paramètre de la liste des éléments enfants
<code>replaceChild</code> (new, old)	Remplace l'élément enfant old par l'élément new

Compléments sur le DOM

Exemple : suppression d'un élément

JS (Modèle)

```
function slideClick() {  
  let bullet = document.getElementById("removeme");  
  bullet.parentNode.removeChild(bullet);  
}
```

Une forme particulière : `obj.remove()` .

Compléments sur le DOM

Selection d'objets DOM

On peut appliquer ces méthodes sur n'importe quel objet DOM (et pas seulement sur l'objet `document`).

- **`getElementsByTagName(tag)`** : retourne un tableau des descendants du type passé en paramètre (comme `"div"`).
- **`getElementsByName(name)`** : retourne un tableau des descendants dont l'attribut `name` est égal à la chaîne de caractères passée en paramètre.
- **`querySelector(selector)`** : retourne le premier descendant qui correspond au sélecteur CSS passé en paramètre.
- **`querySelectorAll(selector)`** : retourne le tableau de tous les descendants qui correspondent au sélecteur CSS passé en paramètre.

Compléments sur le DOM

Selection d'objets DOM : Exemple 1

JS (Exemple)

```
let allParas = document.querySelectorAll("p");  
for (let i = 0; i < allParas.length; i++) {  
    allParas[i].style.backgroundColor = "yellow";  
}
```

HTML (Exemple)

```
<body>  
  <p>This is the first paragraph</p>  
  <p>This is the second paragraph</p>  
  <p>You get the idea...</p>  
</body>
```

Compléments sur le DOM

Selection d'objets DOM : Exemple 1

Surligne tous les paragraphes du document :

HTML (Output)

This is the first paragraph

This is the second paragraph

You get the idea...

Compléments sur le DOM

Selection d'objets DOM : Exemple 2

JS (Exemple)

```
let allParas = document.querySelectorAll("#address p");  
for (let i = 0; i < allParas.length; i++) {  
    allParas[i].style.backgroundColor = "yellow";  
}
```

HTML (Exemple)

```
<p>This won't be highlighted!</p>  
<div id="address" >  
    <p>10090 Agdal</p>  
    <p>Rabat, MAR</p>  
</div>
```

Compléments sur le DOM

Selection d'objets DOM : Exemple 2

Surligne tous les paragraphes du document contenus dans la section d'ID `address` :

HTML (Output)

This won't be highlighted !

10090 Agdal

Rabat, MAR

Compléments sur le DOM

Lire ou modifier les classes CSS

JS (Exemple)

```
function highlightField() {  
    // turn text yellow  
    let text = document.getElementById("text");  
    if (!text.className) {  
        text.className = "highlight";  
    } else if (text.className.indexOf("invalid") < 0) {  
        text.className += " highlight"; // awkward  
    }  
}
```

L'attribut DOM `className` correspond à l'attribut HTML `class`.
Pas très pratique quand un élément a plus d'une classe.

Compléments sur le DOM

L'attribut classList

JS (Exemple)

```
function highlightField() {  
    // turn text yellow  
    let text = document.getElementById("text");  
    if (!text.classList.contains("invalid")) {  
        text.classList.add("highlight");  
    }  
}
```

classList est une *collection* possédant les méthodes **add**, **remove**, **contains**, et **toggle** pour manipuler plus simplement les classes CSS.