Manipulation simple en Javascript
PSB
2020
Y. STROPPA

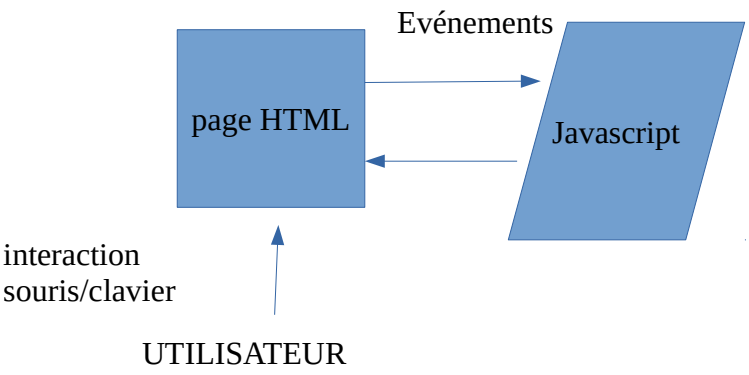
Comment se constitue un site Web .... une partie en HTML accompagnée de CSS et de fichiers JS ; La vocation des fichiers Javascript est de permettre le côté dynamique du site Web c'est à dire de pouvoir répondre aux interactions de l'utilisateur, de pouvoir échanger des données avec le serveur web ou d'autres services ....

Pour ce faire, il est nécessaire de pouvoir communiquer entre la page HTML et JavaScript comme on le fait en CSS.

Si on démarre du fichier HTML qui est chargé dans votre navigateur. Comme se passe se chargement, le navigateur lorsqu'on sollicite une URL, demande au serveur de lui fournir en réponse une structure HTML qu'il télécharge et charge dans une page. A la lecture de cette page, plusieurs événements se passent : le chargement de la page, la lecture de fichiers d'inclusions en JS eou en CSS et éventuellement leur interprétation par la navigateur. En fonction de l'écriture d'instructions Javascript qui peut se faire au chargement de la page, à la fin du chargement de la page et à l'interaction de l'utilisateur.

Tout d'abord à la lecture du fichier HTML, le navigateur définit une première structure d'éléments à partir du fichier HTML. c'est ce que l'on appelle la DOM **Document Object Model** qui permet de définir les éléments avec des propriétés visuels, contenu et comportements le tout accessible à l'aide de la variable en Javascript document.

Ensuite l'ensemble sera piloté entre événement (interaction utilisateur, Timer Javascript...) et instructions en réponse à l'événement.



Comment se passe l'interaction entre les deux éléments chargés dans la même application (navigateur). Les événements sont reçus pas la page HTML qui la relie éventuellement à un traitement associés en Javascript (click, moudown, mouseup, mousemove ...).

Plusieurs façons d'associer des événements à des objets contenu dans la DOM, c'est directement à la déclaration du fichier HTML lorsqu'on définir les Tags de notre structure avec les instructions de type onchange="code javascript", onclick="code javascript" .....

Quelques exemples :

```
<!DOCTYPE HTML>
<html>
<head>
<title>Projet PSB</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<div id="conteneur" class="center">
        <div id="contenu">
      <div id="commande_top" class="commandes">
        <input type="button" id="bt3" value="RàZ" onClick="choixRaz();"/>
        <input type="button" id="bt2" value="Verif" onClick="verifie();"/>
        <select name="config" id="config-select" onchange="chargement_();">
        </select>
      </div>
     <input type="text" id="saisie" name="saisie" valeur="54">
      <svg xmlns="http://www.w3.org/2000/svg"  id="mySVG"
            width="500" height="400" viewBox="0 0 500 500">
      </svg>
      </div>
</div>
<div id='descriptif' class="center_texte">
   <p>
      condition
   </p>
 </div>
</body>
</html>
```

Maintenant si on essaye ce fichier sans le charger sur un serveur mais uniquement en local, que se passe t-il ?

Tout se passe comme il le faut, mais lorsqu'on clique sur les boutons et si on consulte la console on s'aperçoit qu'il y a des erreurs .... normal car on n'a pas défini le contenu des fonctions en réponse au sollicitations de l'utilisateur sur les boutons ....

Plusieurs possibilités, on incorpore directement dans le fichier html le code de réponse ....

```
<input type="button" id="bt3" value="RàZ" onClick="alert('salut');"/>
```

Le comportement sera bien géré par la navigateur qui lors du clicque sur le bouton va bien relier l'événement au comportement souhaité ... mais on s'aperçoit des limitations de cette méthode au niveau du bloc d'instructions .... c'est limité et difficile à maintenir ...

Donc on va préférer définir l'appel à une fonction que l'on va définir dans le corps de notre document html dans un premier temps ...

Attention à encapsuler par des balises <script> </script> qui permettent d'indiquer au navigateur le changement de langages à utiliser

```
<script>
function choixRaz() {
	// corps de la méthode
	alert("je suis dans le corps de la methode") ;
	// fin du corps de la mtéhode
}
<script>
```

Maintenant si on souhaite séparer le corps des fonctions Javascript du HTML il suffit d'inclure une instruction HTML pour indiquer au navigateur l'insertion d'un fichier js par l'instruction suivante :
<script src="premier_fichier.js"></script>

On génére un nouveau fichier premier_fichier.js dans lequel on copie le bloc d'instructions défini :

```
function choixRaz() {
	// corps de la méthode
	alert("je suis dans le corps de la methode") ;
	// fin du corps de la mtéhode
}
```

Ensuite dans le fichier html on ajoute les lignes supplémentaires en entête du fichier
<script src="premier_fichier.js"></script>

On a le début de notre démarche, on préférera organiser notre projet en répertoire du style /js ou /css dans lesquels on installera les différents fichiers nécessaires pour définir les comportements ....

Interaction entre la partie Javacsript et HTML ....

Dans ce petit exemple on s'aperçoit que le comportement est défini à l'initialisation de la page HTML dans les instructions en HTML5 ... et qu'il permet au navigateur de connecter événement --> instructions ... maintenant la question que l'on peut se poser c'est comment le JavaScript peut interagir avec la DOM et que peut-il faire ...

Pour cela nous allons tout d'abord travailler directement dans la console pour examiner les différentes instructions et utilisation possible en "live" ...

Le tout est de récupérer un descripteur sur l'élément que l'on veut modifier grâce à la variable document.getElementById ou document.getElementByName

Exemple sur le premier bouton : var but=document.getElementById("bt3");
Si, on observe les attributs et méthodes associés à cet objet :

On peut interagir sur les propriétés de l'objet directement :

but.style.background="red" ou "yellow"
but.disabled=true ou false
but.value="nouveau contenu"
.....

Définition d'un style associé à nos boutons style1

A rajouter dans l'entête de votre fichier

```
<style>
.style1 {
      background:red;
}
</style>
```

Ensuite on peut récupérer tous les descripteurs associés à un style donné et modifier ainsi toutes les propriétés des éléments ../..

```
<input class="style1" type="button" id="bt3" value="RàZ" onClick="choixRaz();"/>
<input class="style1" type="button" id="bt2" value="Verif" onClick="verifie();"/>
```

Ce qui nous donne d'un point de vue visuel :

Maintenant si on souhaite travailler sur l'ensemble des éléments associés à un même style on peut exécuter les instructions suivan,tes :

```
var btns=document.querySelectorAll(".style1") ;

btns.forEach(e=> {
   e.style.background="yellow";
}
```

.style
balise
   input
   div
    ....

On commence à voir l'étendue des possibilités

On peut remplacer ".style1" par "input pour interagir avec tous les éléments input de la page ou avec la balise div pour interagir avec toutes les div de la page ...

Sur la partie input de type text on peut récupérer le contenu de la saisie et le modifier ....

exemple

```
var texte=document.getElementById("saisie") ;
texte.value ;
texte.value="dldl" ;
```

On souhaite maintenant modifier le comportement du bouton1 ....il suffit de récupérer un descripteur sur le bouton, et de redéfinir un autre comportement par exemple

```
function autrecomportement() {
        console.log("affiche un autre comportement") ;
}
```

Ensuite il faute le relier à la fonction :

```
var btn1=document.getElementById("bt3") ;
btn1.onclick ;
btn1.onclick=autrecomportement ;

// on peut tester
```

Donc on a la possibilité à modifier avec Javascript le comportement de notre page web à la volée en fonction de certaines situations ...

Autre exemple : essayons d'ajouter et relier les éléments entre eux, lors d'un clique sur le bouton verif on souhaite que la zone de texte se charge du contenu VERIF

Donc pour ce faire il suffit d'ajouter un module d'écoute pour le bouton ou de le surcharger si il existe et de créer une fonction qui modifie le contenu de l'input saisie

```
var verif=document.getElementById("bt2") ;
verif.onclick ;
// on peut tester
function charge_input() {
        var zone=document.getElementById("saisie") ;
        zone.value="VERIF"
} ;
verif.onclick=charge_input;
```

Une fois que vous avez mis au point les différentes

Maintenant si on souhaite générer un effet lors du survole de la souris saur la zone de saisie par exemple mettre en majuscules et en minuscules ...

Il faut ajouter deux modules d'écoute :
mouseover et mouseleave

```
var verif=document.getElementById("saisie") ;
function minuscules() {
   var sai= document.getElementById("saisie");
   sai.value= sai.value.toLowerCase();
}
function majuscules() {
   var sai= document.getElementById("saisie");
   sai.value= sai.value.toUpperCase();
}
verif.onmouseover=majuscules;
verif.onmouseleave=minuscules;
```

Maintenant on peut prendre le tout et venir l'incorporer dans un fichier js et ainsi avoir le comportement attendu.

On peut également modifier le contenu d ela DOM on venant ajouter des éléments ou en enlever ....
par exemple rendre les choses visibles ou invisibles
rajouter des items

Veuillez rajouter un bouton qui rende visible ou invisible la zone de saisie

```
<input class="style1" type="button" id="bt3" value="hide" onClick="masque();"/>
```

```
function masque() {
		var zone_saisie=document.getElementById("saisie");
		if (zone_saisie.style.display=="block"){
			zone_saisie.style.display="none";
		} else {
			zone_saisie.style.display="block";
		}
}
```

Ensuite lors de la sélection on souhaite rajouter un nombre d'élément en plus
Il faut ajouter une fonction addElement() qui permettra d'ajouter un nouveau élément à partir du positionnement fixe pour le moment.

Pour ajouter une zone de texte

```
var newdiv=document.createElement("div") ;
var newContentText=document.createTextNode("salut") ;
newdiv .appendChild(newContentText) ;
var currentDiv=document.getElementById("saisie") ;
document.body.insertBefore(newdiv, currentDiv) ;
```

Pour ajouter une nouvel élément de saisie

```
var newdiv=document.createElement("INPUT") ;
newdiv.setAttribute("type","text") ;
newdiv.setAttribute("id","textYS") ;
newdiv.setAttribute("value","text") ;
document.body.appendChild(newdiv) ;
```

Utilisation de Bootstrap pour le positionnement et le caractère responsive d'un site

Notion de grid : positionnement : https://getbootstrap.com/docs/4.0/layout/grid/

Suite des exercices voir la cas étude N°1