Javascript dans une page web

Comment inclure du code dans une page ?

Plusieurs solutions :

- La balise `script`
- appels liés à des événements
- Les pseudo-liens javascript

La balise script

- Le code javascript inclus dans la balise script est exécuté quand il est lu.
- Généralement, on l'utilise pour déf nir des fonctions.
- Deux possibilités :
  1. code javascript écrit dans la page html:

     ```
     <script type="text/javascript">
     ... code javascript
     </script>
     ```

     (à réserver à du code court et non réutilisable)
  2. inclusion de code, pour tout ce qui est réutilisable :

     ```
     <script type="text/javascript" src="monSource.js"></script>
     ```

Remarque

Si le code javascript comporte des caractères "spéciaux" pour HTML, comme '>', '< ou '&', les

logiciels de validations émettent un warning. En théorie, il faudrait "protéger" ces caractères. Cependant, aucun des systèmes existant n'est parfaitement supporté pour le XHTML. La seule solution (à part d'ignorer superbement le problème, ce que nous faisons ici) est d'utiliser de préférence l'inclusion de code.

Appels javascript lié à des événements

- On peut demander qu'un bout de code javascript soit exécuté lors de certains événements:
  - on a pressé un bouton donné (onclick)
  - le pointeur de la souris est passé sur un élément (onmouseover)
  - on a pressé une touche (onkeyup,...)
  - la page a terminé de se charger (onload)
- Le code est alors écrit comme la valeur d'un attribut de l'élément concerné: clickez moi

```
<button onclick="alert('click !')">clickez moi</button>
```

- On écrira du code très court. idéalement un appel de fonction.

Pseudo-url

On peut aussi remplacer une URL par du javascript, en précisant `javascript:` comme protocole:

comme pour les événements, on gardera le code le plus court possible.

```
        <a href="javascript:alert('bonjour')">URL</a>
```

Constructions de base

Le code JavaScript est une suite *d'instructions*, dont certaines peuvent être des *déclarations de fonctions* ou des *affectations*.

Variables en javascript

- Les variables en javascript sont dynamiques et faiblement typées.
- Si une variable n'existe pas, lui donner une valeur la déclare automatiquement
- Une variable peut changer de type:
  montrer a

```
        <script type="text/javascript">
                a= 'salut';
                a= 3;
                a= a + 2;
                a= a+7;
        </script>
        <button onclick="alert('valeur de a '+ a)">montrer a</button>
```

- Il est conseillé de déclarer explicitement les variables à l'aide du mot-clef `var` :

```
<script type="text/javascript">
        var a;
</script>
```

Notez que ça ne f xe pas son type pour autant.

les fonctions

- Bout de code réutilisable, avec un nom.
- Déclarées à l'aide du mot-clef `function`.
- Une fonction *peut* retourner une valeur, à l'aide du mot-clef `return`

```
<script type="text/javascript">
        function somme(a,b) {
                        return a+b;
        }
</script>
<button onclick="alert(somme(3,2))"> montrer somme (entiers) </button>
<button onclick="alert(somme('bonjour ','monde'))"> montrer somme (chaînes)</bu
```

montrer somme (entiers) montrer somme (chaînes)

à noter:

- on n'explicite pas le type des arguments
- les fonctions sont uniquement identif ées par leur nom. On ne peut pas avoir deux fonctions qui s'appellent `somme`

Variables globale

- Une variables déclarée en dehors d'une fonction est *globale*.

montrer exempleVariable1

```
        <script type="text/javascript">
        // marche aussi sans var...
        var exempleVariable1= 0;

        function augmenter1() {
                        exempleVariable1= exempleVariable1 + 1;
        }
        </script>
        <button onclick="augmenter1(); alert(' la variable globale vaut '+ exem
                montrer exempleVariable1 </button>
```

Dans cet exemple, la variable exempleVariable1 modif ée par la fonction est la même que la variable globale.

Variables locale

- Une variable est locale si elle est déclarée dans une **fonction**.

- On utilise `var`.

montrer exempleVariable2

```
<script type="text/javascript">
var exempleVariable2= 0;
function augmenter2() {

        var exempleVariable2=55; // variable locale.
        exempleVariable2= exempleVariable2 + 1;
        alert("dans la fonction : "+exempleVariable2);
}
</script>

<button onclick="augmenter2(); alert('la var. globale vaut '+exempleVariable2);
        montrer exempleVariable2
</button>
```

Types de données

- Types de base : entiers, réels, booléens (true et false) ;
- chaînes de caractères ;
- les tableaux
- fonctions !
- les objets

les chaînes

- sont des objets
- constantes déclarées entre "..." *ou* entre '...'.
- ont une propriété, `length` (leur longueur)
- se comparent avec "==" !!!
- de nombreuses méthodes:
  - `charAt(pos)` retourne une String
  - `charCodeAt(pos)` retourne le code ISO 88591 du caractère
  - `split(separateur)` : renvoie le tableau obtenu en découpant la chaîne.

```
s= "un,deux,trois";
t= s.split(",");
```

  - indexOf(sousChaine,index), indexOf(sousChaine): premier indice de la sous-chaîne, à partir de l'index donné, ou -1.

```
s= "un,deux,trois";
i= s.indexOf("tr"); // i vaut 8
```

  - substr(debut,longueur) : renvoie la sous-chaîne démarrant à "debut", de longueur "longueur".
  - toLowerCase() : renvoie la chaîne en minuscules ;
  - toUpperCase() : renvoie la chaîne en majuscules.
  - String.fromCharCode(c1,c2....) : permet de passer d'une suite de codes ascii à la

chaîne correspondante.

Chaînes et nombres

Deux fonctions de javaScript sont particulièrement intéressantes:

- parseInt(CHAINE) : analyse chaîne comme un entier. Renvoie la valeur NaN (not a number) en cas d'échec.
- parseFloat(CHAINE) : analyse chaîne comme un entier. Renvoie la valeur NaN en cas d'échec.

Notez que ces deux fonctions sont très permissives. Elles acceptent des chaînes qui *commencent* par un nombre, et, s'il y a du texte après, elle l'ignorent. Pour tester si elles ont réussi, on utilisera la fonction isNaN().

```
s="1234";
i= parseInt(s);
if (isNaN(i)) {
        alert(s + " n'est pas un nombre");
}
```

les tableaux (Array)

- doivent être créés, mais on n'a pas besoin de f xer leur taille:

```
tab= new Array(); // Création du tableau
tab[0]= 2;
tab[1]= 3;
s= 0;
for (i= 0; i < tab.length; i++) {
        s= s + tab[i];
}
```

- peuvent être initialisés à la création :

```
tab= new Array("un", "deux", "trois");
```

- longueur donnée par l'attribut `length`
- accès aux cases par la notation habituelle.
- passage d'un tableau à une chaîne de caractères par "join": l'argument de join est inséré entre les éléments.

```
s= tab.join(""); // s vaut undeuxtrois
s= tab.join(":"); // s vaut un:deux:trois
```

Structures de contrôle

if

```
if (CONDITION) {
  ACTIONS1;
} else {
  ACTIONS2;
}
```

for

```
for (INIT; TEST; INCREMENT) {
  ACTIONS;
}
```

exemple

```
s= 0;
for (i= 0; i < tab.length; i++) {
  s= s+ tab[i];
}
```

while

```
while (CONDITION) {
  ACTIONS;
}
```

do...while

```
do {
  ACTIONS;
} while (CONDITION);
```

switch

```
switch (VALEUR_A_TESTER) {
  case VAL1:
        ACTIONS;
  break;
  case VAL2:
        ACTIONS;
  break;
  default:
        ACTIONS;
}
```

Contrairement au C et au Java, la valeur à tester peut être une chaîne de caractères:

```
switch (reponse) {
        case "oui":
                alert("merci");
        break;
        case "non":
                alert("au revoir");
        break;
}
```

Manipulation de la page Web

But: pouvoir lire et modif er des éléments de la page web par programme.

- Pour changer leur aspect
- Pour lire et écrire des éléments de formulaire
- On passe par le DOM « Document objets model ».

Première approche du DOM

- On dispose d'un objet nommé **`document`**, qui représente la page web.
- Pour manipuler un élément, on lui donne un identif ant.
- On peut alors récupérer l'élément à l'aide de la méthode **`getElementById`**

Un texte

peindre

```
<p id="dom1">Un texte</p>
<button onclick="document.getElementById('dom1').style.color='red';">peinc
```

Propriétés des éléments

Un élément retourné par getElementById a plusieurs propriétés, que l'on peut lire et/ou écrire.

- style : permet l'accès à toutes les caractéristiques graphiques CSS de l'élément.
- value : valeur d'un champ de formulaire. En lecture et en écriture.
- innerHTML : propriété non standard, mais très généralement utilisée, qui permet de modif er le contenu de l'élément. Ne fonctionne pas pour tous les éléments.
- id : l'id de l'élément
- className : classe de l'élément (pour les CSS)

Nous allons les détailler.

DOM et CSS (Cascading Style Sheets)

- Les attributs de style CSS d'un élément du DOM permettent de changer son aspect.
- Ils sont accessibles par la propriété "style" des éléments.

Les caractéristiques de cet élément sont contrôlées par les CSS.

color | black | Changer Css

```
<div id="controleParCSS" style="border-style: solid;">
	Les caractéristiques de cet élément sont contrôlées par les CSS.
</div>


<p>
	<select name="cssName" id="cssName">
		<option value="color">color</option>
		<option value="background">background</option>
		<option value="fontSize">fontSize</option>
		<option value="display">display</option>
		<option value="position">position</option>
		<option value="left">left</option>
		<option value="right">right</option>
		<option value="top">top</option>
		<option value="bottom">bottom</option>
		<option value="width">width</option>
		<option value="float">float</option>
		<option value="clear">clear</option>
	</select>
	<input id="cssValue" name="cssValue" value="black">
	<button type="button" onclick="setCSS()">Changer Css</button>
</p>


<script type="text/javascript">
function setCSS() {
	var elt= document.getElementById('controleParCSS');

	var cssValue= document.getElementById('cssValue').value;

	var cssName= document.getElementById('cssName').value;

	switch (cssName) {
		case 'background':
			elt.style.background= cssValue;
		break;
		case 'color':
			elt.style.color= cssValue;
		break;
		case 'fontSize':
			elt.style.fontSize= cssValue;
		break;
			case 'display':
			// none : caché ; inline : dans le cours du texte;
			// block : comme un bloc.
			elt.style.display= cssValue;
		break;
		case 'position':
			// static, absolute, relative ou fixed
			// static : le défaut.
			// ne prend pas en compte left, right, top ou bottom.
			// absolute : position donnée par rapport au conteneur
			// relative : relatif à l'élément précédent
			// fixed : relatif à la fenêtre du navigateur. (pas sur
			elt.style.position= cssValue;
		break;
		case 'left':
			elt.style.left= cssValue;
		break;
		case 'right':
			elt.style.right= cssValue;
		break;
```

DOM et CSS (suite)

On utilise en particulier la propriété "display", et sa valeur "none", pour faire apparaître et disparaître des éléments.

cacher

Vous le voyez ?

```
<button onclick="document.getElementById('coucou').style.display='none';">cache
<div style="border: 2px solid red;" id="coucou">
Vous le voyez ?
</div>
```

Exercice: écrire une fonction qui bascule l'élément entre l'état "caché" et l'état visible, et vice-versa.

Manipulation des images

On peut changer ce qu'une image aff che en modif ant sa propriété "src":

une bille

```
<img id="img1" src="ledlightblue.png"
                onmouseover="document.getElementById('img1').src='ledgreen.png'
                onmouseout="document.getElementById('img1').src='ledlightblue.p
                alt="une bille"
        />
```