JavaScript

Master Pro ILI

Année 2013-14

JavaScript
- client-side scripting :
  - animation du site
  - validation de formulaires
  - ...
- approche objet
- utilise DOM

Un premier exemple en JavaScript

```
<!DOCTYPE  html>
<html>
  <body>
    <script type="text/javascript">
      document.write("Bonjour tout le monde");
    </script>
  </body>
</html>
```

et voilà le premier exemple.

Un deuxième exemple en JavaScript

```
<!DOCTYPE  html>
<html>
  <body>
    <script type="text/javascript">
      document.write("<h1>Bonjour tout le monde</h1>");
      document.write("<p>Le résultat de 2+3 est</p>");
      document.write("<center
      style='font-size : x-large; color : red;'><b>",
  2+3,"</b></center>");
    </script>
  </body>
</html>
```

et voilà le deuxième exemple.

Un langage de script

- dans un document HTML : balise `script` et attribut `type="text/javascript"`
- bien placer le code :
  - dans le corps du document (balise `body`), le code qui doit être exécuté au chargement du document, les appels de fonctions, la réponse à des événements ;
  - dans l'entête du document (balise `head`), le code qui n'est exécuté que lorsqu'il est appelé. C'est l'endroit idéal pour les fonctions, les déclarations de variables globales.
  - on peut importer du code javascript :

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

Écrire des fonctions

Syntaxe :

`function nomFonction(param1,param2){...}`

Exemple : monScript.js

```
function afficheUnTitreEtUnSousTitre(titre,sousTitre){
  document.write("<h1>",titre,"</h1>");
  document.write("<h2>",sousTitre,"</h2>");
}
```

Écrire des fonctions

```
<html>
  <head>
    <script type="text/javascript" src="monScript.js" >
    </script>
  </head>
  <body>
    <script type="text/javascript">
      afficheUnTitreEtUnSousTitre("bonjour","à tous");
      afficheUnTitreEtUnSousTitre("et même","aux autres");
    </script>
  </body>
</html>
```

Résultat du troisième exemple.

Écrire des fonctions

Et pour retourner un résultat ...
instruction `return`
Exemple : monScript.js

```
function conversionCelsiusFahrenheit(celsius){
   return 9*celsius/5 + 32;
}
```

Écrire des fonctions

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

<body>
  <h3>Saviez-vous que 40 degrés Celsius font
  <script type="text/javascript">
    document.write(conversionCelsiusFahrenheit(40));
  </script> degrés Fahrenheit ?</h3>
</body>
```

Résultat de l'exemple.

Variables en JavaScript

- les variables sont déclarées avec l'instruction `var`, mais aussi :
  ```
  var a = 3;
  b = "Bonjour";
  ```
- les noms des variables respectent les contraintes syntaxiques habituelles ;
- JavaScript est sensible à la casse des caractères (minuscule/majuscule) ;

Variables en JavaScript

- une variable peut prendre successivement des valeurs de types différents :
  ```
  var c = "Bonjour";
  c = 123.5;
  ```
- les expressions sont typées
- opérateurs classiques (*à la Java*)
- opérateur `===` teste la valeur et le type

Portée des variables

- une variable déclarée avec *`var`* à l'intérieur d'une fonction a une portée locale ;
- une variable déclarée à l'extérieur d'une fonction ou sans le mot-clé `var` est globale. Elle existe depuis l'endroit où elle est créée jusqu'à la fin de la page.

Portée des variables

monScript.js

```
function conversionCelsiusFahrenheit(celsius){
   return neuf*celsius/5 + uneConstante;
}
var uneConstante = 32;
```

cinquiemeExemple.html

```
<h3>Saviez-vous que 40 degrés Celsius font
<script type="text/javascript">
  var neuf = 9;
  document.write(conversionCelsiusFahrenheit(40));
</script> degrés Fahrenheit ?</h3>
<script type="text/javascript">
  document.write("<hr/",cinq);
  var cinq = 5;
</script>
```

Structures conditionnelles

- si sinon `if (condition) {traitement_alors} else {traitement_sinon}`
- si sinon si sinon si ... `if (...) {...} else if (...) ... else ...`
- énumération de cas `switch(variable){`
  `case valeur_1 : {traitement_1; break;}`
  `case valeur_2 : {traitement_2; break;}`
  `...`
  `default : {traitement_defaut; break;}}`

Structures itératives

- tant que `while(condition) {traitement}`
- jusqu'à `do {traitement} while(condition)`
- pour `for(init ; test_arrêt ; increment) {traitement}`
- instruction `break` pour sortir d'une boucle
- instruction `continue` pour arrêter le traitement courant et continuer la boucle avec la valeur suivante.

Boucle For – exemple

```
<html>
  <body>
  <!-- exemple tiré de w3schools.com -->
    <script type="text/javascript">
      for (i = 1; i <= 6; i++)
      {
        document.write("<h" + i
            + ">ceci est un titre de niveau " + i);
        document.write("</h" + i + ">");
      }
```

```
    </script>
  </body>
</html>
```

Résultat de l'exemple.

JavaScript : un langage objet

- un langage orienté objet ;
- possibilité de créer ses propres types ;
- ou utilisation de types prédéfinis :
  - chaîne de caractères : `String`
  - tableau : `Array`
  - date : `Date`
  - tous les types associés au modèle objet du document HTML...
- avec leurs méthodes prédéfinies.

Les tableaux

- instanciation : `tab = new Array(5);`
- ou : `tabBis = ["elt 1","elt 2", "elt 3"];`
- les éléments sont indicés de 0 à la taille du tableau :

```
for(i = 0; i<tab.length; i++){
  tab[i] = 2*i;
}
```

- pour les tableaux associatifs

```
var monTabAssoc = new Array();
monTabAssoc["JAI"] = "Emmanuel Lonca";
monTabAssoc["SECU1"] = "Salem Benferhat";
monTabAssoc["RES"] = "Bertrand Mazure";
```

HTML DOM

- le Document Object Model de HTML décrit le modèle de tout document HTML ;
- DOM est associé à tout document XML ;
- un document HTML est vu comme un arbre dont les noeuds sont soit des balises, soit des attributs, et les feuilles les valeurs, les zones de texte ;
- DOM fournit les objets pour accéder à tous les noeuds d'un document
- `document` est en javascript l'objet qui décrit le document HTML complet.

Associer un script à un évènement

Via les événements HTML :

- `onclick`, `ondblclick`,
- `onkeypressed`,
- `onkeydown`, `onkeyup`,
- `onload`,
- `onchange`,

- onfocus, onblur,
- oninput,
- ...

Exemple

monScript.js :

```
function init(){
  fahr = document.getElementById('fahrenheit');
  cels = document.getElementById('celsius');
}

function conversionCelsiusFahrenheit() {
   fahr.value = 9*cels.value/5 + 32;
}
```

Exemple

septiemeExemple.html

```
<html>
<head>
  <script type="text/javascript" src="monScript.js">
  </script>
</head>
<body onload="init()">
  <h3>Conversion Celsius - Fahrenheit</h3>
  <form>
    <input type="textarea" id="celsius"
           size="8"/>
    <input type="button" value="->"
           onclick="conversionCelsiusFahrenheit()"/>
    <input type="textarea" id="fahrenheit"
   size="8" />F
  </form>
</body>
</html>
```

DOM HTML

- le niveau 1 est défini depuis 1998 ;
- le niveau 2 est défini depuis 2003 ;
- en IDL Interface Definition Language
- référence de la documentation DOM HTML : http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-200
- mais il existe des extensions ne faisant pas partie de la recommandation du W3C.

Le document HTML

```
interface HTMLDocument : Document {
           attribute  DOMString title;
  readonly attribute  DOMString referrer;
  readonly attribute  DOMString domain;
  readonly attribute  DOMString URL;
           attribute  HTMLElement body;
  readonly attribute  HTMLCollection images;
  readonly attribute  HTMLCollection applets;
  readonly attribute  HTMLCollection links;
  readonly attribute  HTMLCollection forms;
  readonly attribute  HTMLCollection anchors;
           attribute  DOMString cookie;
};
```

Exemple

Premier exemple avec DOM

```
<html>
<head>
  <title>Premier exemple</title>
</head>
<body>
  <script type="text/javascript">
    document.write("<table><tr>");
    document.write("<th>"+document.title+"</th>");
    document.write("<th>"+document.URL+"</th>");
    document.write("</tr></table>");
    document.title="Ceci n'est pas un premier exemple";
  </script>
</body>
</html>
```

Le document HTML

```
interface HTMLDocument : Document {
  void write(in DOMString text);
  Element getElementById(in DOMString elementId);
  NodeList getElementsByName(in DOMString elementName);
};
```

Exemple

```
<html>
  <head>
```

```
    <title>Deuxième exemple</title>
  </head>
  <body>
    <h1 id="titre">Ceci est un titre</h1>
    <script type="text/javascript">
      document.getElementById("titre")
                      .style.backgroundColor="red";
    </script>
  </body>
</html>
```

Résultat de l'exemple.

Une collection d'éléments HTML

```
interface HTMLCollection {
  readonly attribute  unsigned long length;
  Node item(in unsigned long index);
  Node namedItem(in DOMString name);
};
```

Exemple

```
<script type="text/javascript">
  document.write("Nombre d'ancres : "
      +document.anchors.length+"<br/>");
  for(i = 0; i<document.anchors.length; i++){
    var lien = document.anchors.item(i);
    document.write(i+" -- "+lien.name+"<br/>");
    lien.style.color = "green";
  }
</script>
```

Résultat de l'exemple.

Un élément HTML

```
interface HTMLElement : Element {
  attribute  DOMString id;
  attribute  DOMString title;
  attribute  DOMString lang;
  attribute  DOMString dir;
  attribute  DOMString className;
};
```

Les méthodes...

```
interface Element : Node {
  readonly attribute  DOMString tagName;
  DOMString getAttribute(in DOMString name);
  void setAttribute(in DOMString name,
                    in DOMString value)
                    raises(DOMException);
  void removeAttribute(in DOMString name)
                       raises(DOMException);
  ...
  NodeList getElementsByTagName(in DOMString name);
  void normalize();
};
```

Une liste de noeuds

```
interface NodeList {
  Node item(in unsigned long index);
  readonly attribute  unsigned long length;
};
```

Un noeud

```
interface Node {
  // un certain nombre de constantes
  readonly attribute  DOMString nodeName;
           attribute  DOMString nodeValue;
  ...
  readonly attribute  Node parentNode;
  readonly attribute  NodeList childNodes;
  readonly attribute  Document ownerDocument;
};
```

Un noeud

```
interface Node {
  Node insertBefore(in Node newChild,
                    in Node refChild)
                    raises(DOMException);
  Node replaceChild(in Node newChild,
                    in Node oldChild)
                    raises(DOMException);
  Node removeChild(in Node oldChild)
                  raises(DOMException);
  Node appendChild(in Node newChild)
```

```
                    raises(DOMException);
  boolean hasChildNodes();
};
```

Le corps de document

```
interface HTMLBodyElement : HTMLElement {
           attribute  DOMString aLink;
           attribute  DOMString background;
           attribute  DOMString bgColor;
           attribute  DOMString link;
           attribute  DOMString text;
           attribute  DOMString vLink;
};
```

Exemple

```
<h1 onMouseOver="document.bgColor='yellow'"
      onMouseOut="document.bgColor='white'">
   Attention! Je change de couleur quand on s'approche!
</h1>
```

Résultat de l'exemple.

Un formulaire

```
interface HTMLFormElement : HTMLElement {
  readonly attribute  HTMLCollection elements;
  readonly attribute  long length;
           attribute  DOMString name;
           attribute  DOMString acceptCharset;
           attribute  DOMString action;
           attribute  DOMString enctype;
           attribute  DOMString method;
           attribute  DOMString target;
};
```

Un formulaire

```
interface HTMLFormElement : HTMLElement {
  void submit();
  void reset();
};
```

Un élément Input

```
interface HTMLInputElement : HTMLElement {
           attribute  DOMString defaultValue;
           attribute  boolean defaultChecked;
  readonly attribute  HTMLFormElement form;
           attribute  DOMString alt;
           attribute  boolean checked;
           attribute  boolean disabled;
           attribute  long maxLength;
           attribute  DOMString name;
           attribute  boolean readOnly;
           attribute  DOMString value;
};
```

Un élément Input

```
interface HTMLInputElement : HTMLElement {
  void blur();
  void focus();
  void select();
  void click();
};
```

Exemple

```
<form>
  <input
     type="button"
     value="c'est parti"
     onClick=
        "document.getElementById('zoneDeTexte')
            .disabled=false;
        document.getElementById('zoneDeTexte')
            .focus();"/>
  <input
     type="textarea" disabled="true"
     size="10" id="zoneDeTexte"/>
</form>
```

Résultat de l'exemple.

Une extension

- IE : la propriété `innerHTML`
- permet de modifier facilement le contenu d'un noeud : `document.getElementById("unElt").innerHTML = "<p>Ceci est un nouveau contenu</p>";`

- n'existe pas dans la recommandation du w3c pour l'api DOM
- mélange texte et éléments
- une version plus rigoureuse :

```
ndText=document.createTextNode(
        "Ceci est un nouveau contenu");
nbElt=document.createElement("p");
nbElt.appendChild(ndText);
document.getElementById("unElt").appendChild(nbElt);
```

La propriété Style

```
  // Introduced in DOM Level 2:
interface ElementCSSInlineStyle {
  readonly attribute CSSStyleDeclaration  style;
};
```

- permet ensuite d'accéder à chacune des propriétés qu'on peut mettre à jour dans une CSS.

Les propriétés de style

```
  // Introduced in DOM Level 2:
interface CSS2Properties {
           attribute DOMString        background;
           attribute DOMString        backgroundColor;
           attribute DOMString        border;
           attribute DOMString        clear;
           attribute DOMString        color;
           attribute DOMString        content;
           attribute DOMString        display;
           attribute DOMString        cssFloat;
           attribute DOMString        font;
};
```

Les propriétés de style

```
// Introduced in DOM Level 2:
 interface CSS2Properties {
            attribute DOMString        listStyle;
            attribute DOMString        margin;
            attribute DOMString        outline;
            attribute DOMString        padding;
            attribute DOMString        textAlign;
            attribute DOMString        textDecoration;
            attribute DOMString        textIndent;
            attribute DOMString        textShadow;
```

```
            attribute DOMString        visibility;
            attribute DOMString        width;
};
```

JavaScript - L'exemple Aalm

Donner une couleur rouge à l'arrière-plan d'une section lorsque la souris passe dessus.

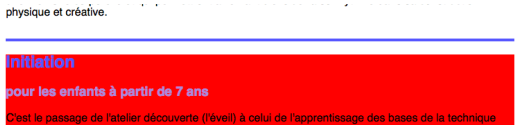
```
<script type="text/javascript">
  function initialisation(){
     lesSections =
        document.getElementsByClassName("section");
     for(i = 0; i < lesSections.length; i++){
        var uneSection = lesSections.item(i);
        uneSection.
           setAttribute("onMouseOver","onMouseOver(this)");
        uneSection.
           setAttribute("onMouseOut","onMouseOut(this)");
     }
  }
```

JavaScript - L'exemple Aalm (2)

```
   function onMouseOver(elt){
      elt.style.backgroundColor = "red";
   }
   function onMouseOut(elt){
      elt.style.backgroundColor = "white";
   }
  </script>
</head>
<body  onLoad="initialisation()">
  <!-- plus de javascript !! -->
</body>
</html>
```

JavaScript - L'exemple Aalm (3)

Sur l'écran



Le DOM de la page

```
<div class="principale">
    <h1>Les Cours</h1>
  ▶ <div class="section" onmouseover="onMouseOver(this)" onmouseout="onMouseOut(this)" style="background-color: white;">
  ▶ <div class="section" onmouseover="onMouseOver(this)" onmouseout="onMouseOut(this)" style="background-color: red;">
  ▶ <div class="section" onmouseover="onMouseOver(this)" onmouseout="onMouseOut(this)" style="background-color: white;">
  ▶ <div class="section" onmouseover="onMouseOver(this)" onmouseout="onMouseOut(this)" style="background-color: white;">
```