

Chapitre 4

JavaScript (JS)

JS

- Le JavaScript (JS) est le langage à balises qui vient compléter le HTML et CSS: il permet de rendre la page Web interactive (dynamique).
- C'est un langage interprété, structuré, Orienté Objet et inspiré essentiellement du langage C.
 - Il s'écrit directement dans la page HTML ou dans un fichier externe.
 - Il permet de gérer les événements principaux de la souris: déplacements, clicks, ...
 - Il permet de programmer les boutons.

JS

❖ Utilisation du JS dans une page web

- Embarquer directement le code JavaScript dans un document HTML à l'intérieur de la balise <script>.
 - Si cette balise est placée dans la section head, le code JavaScript sera accessible avant l'affichage de la page Web.
 - Si cette balise est placée dans la section body, le code JavaScript sera accessible lors de l'affichage de la page Web.

```
<script language="javascript"> //ou bien <script type="text/javascript">  
    alert("Le code JS est à l'intérieur de la balise <script>");  
</script>
```

- Placer le code JavaScript dans un fichier externe de type texte qui doit être liés aux documents HTML où sera exécuté ce code.

Fichier test.html

```
<!DOCTYPE html>  
<html>  
  <head>  
    <script type="text/javascript" src="test.js"></script>  
  
  </head>  
  <body>  
    <h1>Un titre de niveau 1</h1>  
  </body>  
</html>
```

Fichier test.js

```
alert("Le code JS est dans le fichier externe test.js");
```

JS

❖ Exécution du code JS

- Le code JavaScript est appelé par le pseudo protocole JavaScript: comme une valeur de l'attribut href de la balise <a>.

```
<a href="javascript:alert('Par le pseudo-protocole  
  javascript:');"> Cliquer pour exécuter le code Javascript  
</a>
```

- Le code JavaScript est exécuté suite à des événements déclenchés par l'interaction des utilisateurs avec les documents HTML. Par exemple un clic souris sur les éléments HTML ou lors de leurs survole par le pointeur de la souris.

```

```

JS

❖ Règles générales

- L'insertion des espaces peut s'effectuer n'importe où dans le script, donc n'hésitons pas à les utiliser pour éclaircir nos écritures.
- Pour rendre plus clair et plus compréhensible notre programme, on peut placer des commentaires :
 - En utilisant le double slash `//`: tout ce qui se trouve à droite est commentaire.
 - En utilisant `/*` et `*/`: pour écrire un commentaire sur plusieurs lignes.
- Un nombre à virgule est séparé par un point (.) et non par une virgule (0.5 et non 0,5).
- Javascript est sensible à la casse: le mot Nom est différent de nom et de NOM.
- Une instruction simple est une expression qui se termine avec un point-virgule ; et un bloc d'instructions est un ensemble d'instructions simples et/ou des blocs d'instructions rassemblées entre une accolade ouvrante et une accolade fermante.

```
expression;           // Instruction simple
{                     // Bloc d'instructions
  instruction(s);      // une ou plusieurs instructions simples et/ou blocs
}
```

JS

❖ Déclaration et affectation de variables

- Les variables sont des contenants qui permettent de manipuler facilement des valeurs numériques ou autres.
 - Le mot-clé var permet de déclarer une ou plusieurs variables.
 - Après la déclaration de la variable, il est possible de lui affecter une valeur. La déclaration et affectation en même temps:

```
var nom, age;      //déclaration de plusieurs variables
nom="toto";        //affectation d'une chaine de caractère
age=35;            //affectation d'une valeur numérique
var prenom=titi;   //déclaration et affectation
```

- Il n'est pas nécessaire d'indiquer le type de la variable. Il reconnu automatiquement lors de l'affectation. De ce fait, le type de la variable peut être modifié plusieurs fois.

```
var v=true;        // v est de type boolean par initialisation
v="zéro";          // v change de type après affectation
```

- Il est intéressant de choisir un nom de variable qui rappelle sa fonction dans le programme. Si cette fonction fait intervenir plusieurs mots, on essaie de choisir un nom utilisant une contraction de ces mots en faisant précéder chacun d'eux par une Majuscule ou __.

Exemple: monNom ou mon__nom

JS

❖ Opérateurs et expressions JavaScript.

- Le langage JavaScript utilise les mêmes opérateurs que ceux utilisés par le langage C (opérateurs arithmétiques, opérateurs de comparaison et opérateurs logiques). Il est de même pour les expressions qui doivent être construites en suivant les mêmes règles syntaxiques que celles du langage C.

Opérateur	Effet
<code>+, *, -, /, %(modulo)</code>	Opérateurs arithmétiques (<i>10%4=2 c'est le reste de la division entière</i>)
<code><=, >=, >, <, ==, !=</code>	Opérateurs de comparaison
<code>++, --</code>	Opérateur d'incrément et de décrémentation
<code> , &&, !</code>	Opérateurs logiques
<code>=, +=, *=, -=, /=, %=</code>	Opérateurs d'affectation (<i>x+=4 est équivalente x=x+4</i>)

❖ Structures conditionnelles en JavaScript.

- En JavaScript, les structure de programmation conditionnelles sont les même qu'en langage C :
- Test simple : if else

```
if (condition)
  Instruction(s);    // simple ou bloc d'instructions exécuté si condition est vrai
else
  Instrction(s);     // Facultative exécuté si condition est fausse
```

- Test multiple : switch

```
switch (variable) {
  case valeur1 : Instructions1;    // Instructions1 exécutée si variable==valeur1
  ...
  case valeuri : Instructionsi;    // Instructionsi exécutée si variable==valeuri
}
```

JS

❖ Structures répétitives en JavaScript.

- En JavaScript, les structure de programmation répétitives sont les même qu'en langage C.

- Boucle dont la condition est testée au début : while

```
while (condition) {  
    Instruction(s); // simple ou bloc d'instructions exécuté si condition est vrai  
}
```

- Boucle dont la condition est testée à la fin : do

```
do{  
    Instruction(s); // simple ou bloc d'instructions exécuté au moins une fois  
}  
while(condition)
```

- Généralisation de la boucle while : for

```
for (initialisation; condition; expression_arithmetique) {  
    Instruction(s); // simple ou bloc d'instructions  
}
```

JS

❖ Fonctions en JavaScript.

- La déclaration de fonctions en JavaScript se fait par l'utilisation du mot-clé **function**, du nom de la fonction et des paramètres de la fonction entre deux parenthèses .
- Le corps d'une fonction est délimité par deux accolades **{}**. Il contient le code Javascript qui est exécuté à chaque appel de la fonction et éventuellement une instruction return dans le cas où la fonction retourne une valeur.

```
function nomFonction(paramètres) {  
    // corps de la fonction  
    return (valeur);  
}
```

```
function parite(x) {  
    x = Number(x);  
    if(x)  
        document.write(x+" est impaire");  
    else  
        document.write(x+" est impaire  ");  
}
```

JS

- L'appel de la fonction se fait simplement par son nom. Les paramètres de la fonction sont séparés par le caractère virgule, et ils sont passés à la fonction par valeur pour les types primitifs et par adresse pour les objets. Il est de même pour la valeur retournée qui peut être de type simple ou un objet.

```
nomFonction(par1, par2, ...) ;  
var v=nomFonction(par1, par2, ...) ;
```

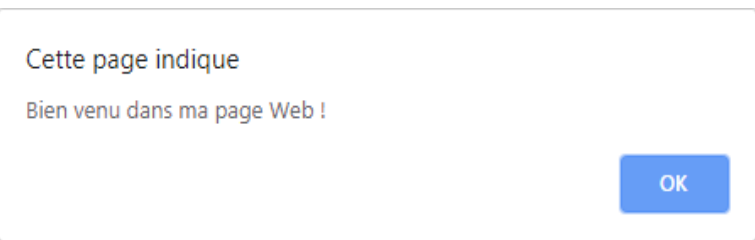
```
Parite(15)
```

JS

❖ L'objet Window.

- L'objet window représente la fenêtre du navigateur. Il possède quelques propriétés de type simple, comme l'attribut status, et d'autres attributs de type objet comme navigator, history et document. Il offre quelques méthodes, dont les méthodes d'interaction permettant le dialogue homme/machine.
- `alert(message)` : affiche une boîte de dialogue contenant le message passé en paramètre et un seul bouton OK permettant de fermer cette boîte.

```
<script language="javascript">  
  alert("Bienvenu dans ma page Web !");  
</script>
```

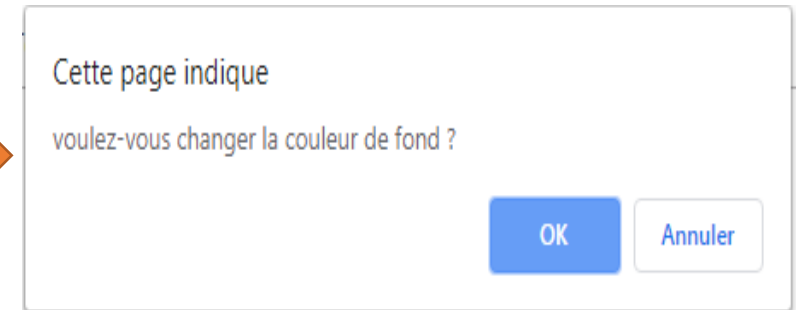


JS

❖ L'objet window.

- `confirm(message)` : affiche une boîte de dialogue contenant le message passé en paramètre et deux boutons OK et Annuler. En cliquant sur le bouton OK, cette fonction retourne la valeur logique true et si on clique sur le bouton Annuler la fonction retourne la valeur logique false.

```
<script language="javascript">  
  confirm("voulez-vous changer la couleur de fond?");  
</script>
```



JS

❖ L'objet window.

- `prompt(message)` : affiche une boîte de dialogue contenant le message passé en paramètre, une zone de saisie de texte au clavier et deux boutons Ok et Annuler. Si on clique sur le bouton OK la valeur entrée au clavier est retournée par cette fonction et si on clique le bouton Annuler la fonction retourne la valeur null.

```
<script language="javascript">  
  prompt("Saisissez votre texte :", "Texte par défaut");  
</script>
```



Cette page indique

Saisissez votre texte :

OK Annuler

L'objet window possède aussi la méthode **open()** qui permet de créer dynamiquement de nouvelles fenêtres et la méthode **close()** pour fermer les fenêtres ouvertes.

JS

❖ L'objet document.

- L'objet Document permet de manipuler tous les objets qui sont inclus dans le documents HTML (images, textes, formulaires...) par leurs noms ou leurs ids.
- Il existe de nombreux méthodes (et attributs) qui s'appliquent sur l'objet document.
 - write(): permet d'écrire la chaîne de caractères passée en paramètre directement dans le document.
 - writeln(): permet d'écrire dans le document en ajoutant un retour à la ligne.

```
document.write("Bonjour"); //Affiche "Bonjour" dans la page.  
document.writeln("Bonjour"); //Affiche "Bonjour" suivi d'un retour à la ligne dans la page.  
document.write(v); //Affiche la valeur de la variable v.
```

- Cet objet possède deux méthodes open() et close(). La première méthode permet l'ouverture des documents affichés par des fenêtres pour permettre leur modification dynamique, et la deuxième méthode permet leur fermeture.

JS

❖ Les méthodes de l'objet Document.

- getElementById(): méthode qui retourne un objet HTML à partir de son id

```
<input type="text" id="NOM">  
<script type="text/javascript">  
    document.getElementById("NOM").value="texte par  
</script>
```

texte par défaut

- getElementsByName(): méthode qui retourne un tableau d'objets HTML ayant nom défini dans la propriété name de la balise de l'objet.

```
<input type="text" name="nom"> <br>  
<input type="text" name="nom" value=" "> <br>  
<input type="text" name="nom" value=" "> <br>  
<script language="javascript">  
    document.getElementsByName("toto")[0].value = "toto 0";  
    document.getElementsByName("toto")[1].value = "toto 1";  
    document.getElementsByName("toto")[2].value = "toto 2";  
</script>
```

toto 0

toto 1

toto 2

JS

❖ Propriétés de l'objet document.

- bgcolor: Elle permet de changer dynamiquement la couleur de fond de la page Web.

```
<a href="javascript:void(document.bgColor=' #339999' )">Fond bleu</a>  
<a href="javascript:void(document.bgColor=' #66FF99' )">Fond vert</a>
```

- lastModified: retourne la date de dernière modification.
- location: retourne l'url complète de la page.
- title: retourne le type du document dans la balise title.

JS

❖ Propriétés de l'objet document.

- innerHTML: est une propriété qui permet de modifier le contenu d'un paragraphe ou d'insérer un objet dans la page.

```
<p id="paragraphe">texte texte</p>
<script type="text/javascript"> //ou bien <script language="javascript">
    document.getElementById("paragraphe").innerHTML ="Texte changé";
</script>

//Navigateur affiche : Texte changé
```

JS

❖ Sous-objet style de l'objet Document.

- Le sous-objet style permet de définir dynamiquement un attribut de style local style sur la balise concernée.

```
<span id="citation">
  Mieux vaut tard que jamais.
</span>
<script language="javascript">
  document.getElementById("citation").style.color="#888888";
  document.getElementById("citation").style.fontStyle="italic";
</script>
```

- Les attributs du sous-objets style sont les propriétés que nous avons déjà vu en CSS. Si la propriété est composée, alors le tiret est enlevé et le caractère immédiatement suivant est capitalisé; par exemple font-style devient fontStyle.
- La valeur à appliquer à l'attribut est la même que celle déclarée en CSS, juste que cette fois, il faut la placer entre des guillemets, car elle est vue en Javascript en tant que chaîne de caractères.

JS

❖ Les événements.

- **onClick**: Exécute une action lors du clic sur un élément : bouton, champs, ...

```
<form>
  <input type="text" name="champ_texte" value="valeur d'origine">
  <input type="button" name="bouton" value="Changer"
    onclick="champ_texte.value='valeur après clic'">
</form>
```

- **onChange**: Exécuter une action lorsqu'une option ou un champ est changé.

```
<form>
  <input type="text" name="champ_texte" value="valeur d'origine">
  <input type="button" name="bouton" value="Changer"
    onclick="champ_texte.value='valeur après clic'">
</form>
```

- **onBlur**: L'action est exécutée lorsque l'élément perd le focus par exemple sortir d'une zone de type "text" dans laquelle on est rentré.

```
<form>
  <input type="text" name="champ_texte" value="valeur d'origine">
  <input type="button" name="bouton" value="Changer"
    onclick="champ_texte.value='valeur après clic'">
</form>
```

Exercice:

1. Faire le nécessaire en HTML ,CSS
et JavaScript pour créer la page
suivante:

GENERER LA FACTURE

Designation

Quantité

Prix unitaire

Nom du Client

Remise de 15%

Calculer les PT

PT

PT - Remise

2. Faire le nécessaire en JavaScript pour calculer le Prix Totale et le PT-Remise et afficher une fenêtre qui contient le message :

Bonjour nom du client vous devez payer PT-remise MRU comme dans la page suivant:.

The screenshot shows a web browser window with the address bar displaying a file path. A modal dialog is open in the center of the screen, displaying the message: "Cette page indique Bonjour Aicha Vous devez payer 2550000MRU". Below the dialog, there is a form with several input fields and buttons. The form fields are: "Produit", "1000", "3000", "Aicha", and "Remise de 15%". Below these fields is a green button labeled "Calculer les PT". At the bottom of the form are two more input fields: "PT" and "PT - Remise".