Chapitre 4

Langage JavaScript

Plan du chapitre 4

- Introduction
- Insertion du code JavaScript dans une page HTML
- Conventions
- Variables
- Tableaux
- Opérateurs
- Tests logiques
- Boucles
- Fonctions
- Objets & propriétés
- Évènements

Plan du chapitre 4

- **Introduction**
- Insertion du code JavaScript dans une page HTML
- Conventions
- Variables
- Tableaux
- Opérateurs
- Tests logiques
- Boucles
- Fonctions
- Objets & propriétés
- Évènements

Introduction

- JavaScript a été initialement développé par Netscape en 1995, puis standardisé par l'ECMA International (European Computer Manufacturers Association) sous le nom d'ECMAScript.
- Le JavaScript est un langage de programmation de scripts (i.e. langage interprété) orienté objet (i.e. utilise les objets pour communiquer avec le monde extérieur).
- Le JavaScript est intégré directement dans le fichier HTML et permet de dynamiser une page Web, en ajoutant des interactions avec l'utilisateur, des animations (Afficher/masquer du texte, faire défiler des images, créer des info-bulles...).

Introduction

- Les scripts JavaScript sont exécutés par le navigateur Web sur la machine du client (client-side).
- Le JavaScript est un langage basé événement (event-driven) :
  - Il permet de manipuler les événements de la souris, les menus déroulants, les messages d'alerte, les fenêtres, les cadres, les données des formulaires, et leur associer des actions ou des fonctions.
  - Il peut être utilisé pour vérifier la validité des données fournies par l'internaute.

Plan du chapitre 4

- Introduction
- **Insertion du code JavaScript dans une page HTML**
- Conventions
- Variables
- Tableaux
- Opérateurs
- Tests logiques
- Boucles
- Fonctions
- Objets & propriétés
- Évènements

Insertion du JavaScript dans une page HTML

L'intégration du code JavaScript dans une page HTML peut se faire de trois façons:

- Grâce à la balise **<SCRIPT>** ;
- En mettant le code dans un fichier externe ;
- En mettant en place un gestionnaire d'événements ;

Insertion du JavaScript dans une page HTML

Insertion avec la balise <SCRIPT>

- Le code JavaScript peut être inséré dans n'importe quel endroit de la page Web, à condition que le script soit entièrement chargé avant d'exécuter une instruction.
- Généralement, le code JavaScript est placé dans la balise d'en-tête (**<HEAD>** et **</HEAD>**). Cependant, les évènements JavaScript sont insérés dans le corps de la page Web entre les balises **<BODY>** et **</BODY>** en tant qu'attribut d'un marqueur HTML.
- Syntaxe :

```
<script language = "javascript">
<!--
      Placez ici le code de votre script
-->
</script>
```

Ou

```
<script type = "text/javascript">
<!--
      Placez ici le code de votre script
-->
</script>
```

Insertion du JavaScript dans une page HTML

Insertion avec la balise <SCRIPT>

- L'attribut **language** de la balise **<SCRIPT>** spécifie le langage utilisé pour les lignes de code qui suivent. Cet attribut laisse entrevoir que JavaScript n'est pas le seul langage script qui peut être interprété par le navigateur (exple : VbScript).
- Il est possible d'utiliser plusieurs versions de JavaScript en déclarant plusieurs balises **<SCRIPT>** ayant chacune comme attribut la version du JavaScript correspondante.
- **N.B :** Pour les versions du HTML antérieures à la norme HTML5, il est souvent indispensable de mettre tout le script entre **<!–** et **–>** pour que le document soit correctement validé. Les navigateurs qui exécutent JavaScript ne tiendront pas compte de ces caractères.

Insertion du JavaScript dans une page HTML

Insertion dans un fichier externe

- Le code JavaScript est placé dans un fichier indépendant sauvegardé avec l'extension **.js**.
- Dans le fichier HTML, on insère la ligne de code suivante :

```
<SCRIPT LANGUAGE=JavaScript SRC="url/fichier.js"> </SCRIPT>
```

Insertion du JavaScript dans une page HTML

Mise en place d'un gestionnaire d'événements

- Dans le navigateur, certaines actions effectuées par l'internaute donnent lieu à des événements (exp : clic d'un des boutons de la souris).
- Un gestionnaire d'événements, une fois mis en place, sera automatiquement exécuté lorsque l'événement correspondant se présentera.
- Pour définir un gestionnaire d'événements, on utilise la syntaxe suivante :

```
<balise NomEvenement="code JavaScript à exécuter">
```

Insertion du JavaScript dans une page HTML

Mise en place d'un gestionnaire d'événements

Exemple:

```
<html>
<head>
<script language="javascript">
  function Carre (idForm)
  {
     Var formulaire=document.getElementById("idForm");
      formulaire.resultat.value = formulaire.resultat.value * formulaire.resultat.value;
  }
</script>
</head>
<body>
<form id=nomForm>
     <p> Test de la fonction <b> carré </b> : </p>
     <input type=text name=resultat size=25 />
     <input type=button name=carre value=Carré onclick="Carre(nomForm)" />
</form>
</body>
</html>
```

Insertion du JavaScript dans une page HTML

Mise en place d'un gestionnaire d'événements

Exemple:

```
<html>
<head>
<script language="javascript">
  function Carre ( X )
  {
      X = X * X ;
  }
</script>
</head>
<body>
<form id=nomForm>
     <p> Test de la fonction <b> carré </b> : </p>
     <input type=text name=resultat size=25 />
     <input type=button name=carre value=Carré
          onclick="Carre( document.getElementById("nomForm").resultat.value)" />
</form>
</body>
</html>
```

Plan du chapitre 4

- Introduction
- Insertion du code JavaScript dans une page HTML
- **Conventions**
- Variables
- Tableaux
- Opérateurs
- Tests logiques
- Boucles
- Fonctions
- Objets & propriétés
- Évènements

Conventions

- Les instructions d'un script JavaScript sont séparées par des ; ou par un retour chariot.
- Tout ce qui se trouve après \\ dans une ligne est un commentaire.
- Un commentaire multiligne est écrit entre /* et */.
- Le langage JavaScript est sensible à la casse (attention aux noms des variables et des fonctions).

Plan du chapitre 4

- Introduction
- Insertion du code JavaScript dans une page HTML
- Conventions
- **Variables**
- Tableaux
- Opérateurs
- Tests logiques
- Boucles
- Fonctions
- Objets & propriétés
- Évènements

Variables

Définition

Une variable est un objet identifié par un nom et contenant des données qui pourront être modifiées lors de l'exécution du programme.

Types de données

En JavaScript, les variables sont déclarées sans avoir besoin de préciser leur type (tel est le cas pour les langages C, C++ ...).

Variables

Types de données

- Les types de données autorisés en JavaScript sont :
  - **Les nombres:** entiers (décimal (base10), octal (base8) ou hexadécimal (base16)) ou à virgules flottantes. On y trouve aussi certaines constantes correspondent à des valeurs particulières telles que :
    - **positive Infinity** ou **+Infinity** qui peut être obtenu quand on essaie d'ajouter une quantité strictement positive au plus grand nombre géré par le langage.
    - **negative Infinity** ou **–Infinity.**
    - **positive zero** ou **+0** qui correspond à une valeur nulle positive.
    - **negative zero** ou **-0** qui correspond à une valeur nulle négative.
    - **NaN** (Not A Number) qui est obtenu lorsque l'on essaie de réaliser une opération interdite (exp : division par zéro).

Variables

Types de données

- Les types de données autorisés en JavaScript sont :
  - **Les chaînes de caractères** (string): une suite de caractères délimitée par des ” ou ’. Pour utiliser des caractères spéciaux dans les chaînes, il faut les précéder d'un antislash (\):
    - \b : touche de suppression (retour arrière)
    - \f : saut de page
    - \n : retour à la ligne
    - \r : appui sur la touche *ENTREE*
    - \t : tabulation
    - \" : guillemets doubles
    - \' : guillemets simples
    - \\ : caractère antislash

Exemple :

```
Chaine = " \"c:\\windows\\\"";
```

Variables

Types de données

- Les types de données autorisés en JavaScript sont :
  - **Les booléens:** des variables à deux états permettant de vérifier une condition :
    - **true:** si le résultat est vrai.
    - **false:** lors d'un résultat faux.
  - **Les variables de type *null*:** un mot caractéristique qui signifie qu'une variable ne contient pas de donnée.

Variables

Types de données

- Pour tester l'existence d'une variable et/ou vérifier son type, on utilise l'instruction **typeof.**

- **Exemple :**

```
var BooleanValue = true;
var NumericalValue = 354;
var StringValue = "This is a String";
alert(typeof BooleanValue);  // affiche "boolean"
alert(typeof NumericalValue);  // affiche "number"
alert(typeof StringValue);  // affiche "string"
alert(typeof nothing);  // affiche “undifined”
```

Variables

Conversion de type

JavaScript est doté de 4 fonctions de conversion de types chaîne/numérique :

- **eval()**: évaluation et conversion numérique d'une chaîne.

var a=2 eval("a*2") // retourne la valeur numérique 4.	eval(123) //retourne la valeur numérique 123.

- **parseInt()** : conversion d'une chaîne en un nombre entier.

parseInt("FF",16) /* retourne la valeur numérique 255 correspondante à la chaine 'FF' dans la base 16. */
parseInt("12",8) /* retourne la valeur numérique 10 correspondante à la chaine '12' dans la base 8. */
parseInt("essai",10) /* retourne la valeur numérique 0. */

Variables

Conversion de type

JavaScript est doté de 4 fonctions de conversion de types chaîne/numérique :

- **parseFloat()** : conversion d'une chaîne en un nombre réel.

```
parseFloat("52.96")   /* retourne la valeur numérique 52.96. */
parseFloat("X12")   /* retourne la valeur numérique 0. */
```

- **string()** : transforme un objet en chaîne de caractères.

Variables

Tests sur les types

- **isFinite()** : permet de tester si la variable passée en paramètre est bien un nombre fini (exp : isFinite(+Infinity) renvoie le booléen false).
- **isNaN()** : teste si le paramètre n'est pas un nombre. (exp : isNaN(34.7) renvoie false, isNaN("abc") renvoie true).

Variables

Déclaration & affectation de valeurs

- **Déclaration d'une variable sans affectation de valeur :**
  - Pour déclarer une variable, on utilise le mot-clé **var.**
  - **Exemple :**

    var a ; /* déclare l'existence d'une variable dont le nom est a. */

- **Opérateur d'affectation simple :**
  - L'affectation d'une valeur à une variable se fait à l'aide de l'opérateur = .
  - **Exemple :**

    a="Ceci est une chaîne"

    a=b=c=d=e=5

Variables

Déclaration & affectation de valeurs

- **Opérateurs d'affectation complexe :**
  - += : permet de réaliser une addition : si a vaut 5, a+=2; affecte 5+2=7 à a.
  - -= : permet de réaliser une soustraction : si a vaut 5, a-=2; affecte 5-2=3 à a.
  - *= : permet de réaliser une multiplication : si a vaut 5, a*=2; affecte 5*2=10 à a.
  - /= : permet de réaliser une division : si a vaut 5, a/=2; affecte 5/2=2.5 à a.
- **Opérateur ternaire**
  - Cet opérateur permet d'affecter une valeur à une variable en fonction du résultat à un test.
  - **Syntaxe :**
    variable=TestÀRéaliser?valeurSiTestVrai:ValeurSiTestFaux
  - **Exemple :**
    a=b>3?0:1 affecte 0 à a si b est strictement supérieur à 3, et 1 dans le cas contraire.

Variables

Portée d'une variable

- **Variable globale :** une variable déclarée sans le mot-clé **var** (i.e. de façon implicite) et accessible de partout dans le script.
- **Variable locale :** une variable déclarée avec le mot-clé **var** (i.e. de façon explicite) dont la portée dépend de l'endroit où elle est déclarée.

Plan du chapitre 4

- Introduction
- Insertion du code JavaScript dans une page HTML
- Conventions
- Variables
- **Tableaux**
- Opérateurs
- Tests logiques
- Boucles
- Fonctions
- Objets & propriétés
- Évènements

Tableaux

Tableau à une dimension

- Déclaration d'un tableau de n éléments :

```
var nom_tableau = new Array(n) ;
```

- Déclaration d'un tableau dont le nombre d'éléments est a priori inconnu :

```
var nom_tableau = new Array() ;
```

Tableaux

Tableau à une dimension

- Affectation de valeurs à un tableau :
  - Affectation des valeurs l'une après l'autre :

```
var tableau1 = new Array(4);
tableau1[0]="Beurre" ;
tableau1[1]="Confiture" ;
tableau1[2]="Pain" ;
tableau1[3]="Jus de fruit" ;
```

  - Affectation des valeurs en une seule ligne :

```
var tableau1 = new Array(4);
tableau1=["Beurre", "Confiture", "Pain", "Jus de fruit"];
```

Tableaux

Tableau à une dimension

- Déclaration et définition simultanées d'un tableau :

```
var tableau1 = new Array('Beurre', 'Confiture', 'Pain', 'Jus de fruit');
```

- La propriété length d'un tableau renvoie son nombre d'éléments.

```
var tableau1 = new Array('Beurre', 'Confiture', 'Pain', 'Jus de fruit');
alert(tableau1.length);
// renvoie 4.
```

Tableaux

Tableau à une dimension

- Il est possible de remplir un tableau de données de différents types.
- Exemple :

```
var tableau = new Array (10,"jeune",22.5);
```

Tableaux

Tableau multidimensionnel

```
var nom_tableau = new Array(n);
for (i=0;i<nom_tableau.length;i++)
 {
    nom_tableau[i] = new Array(m);
 }
```

Tableaux

Tableaux associatifs

- Tableaux dont les éléments sont accessibles grâce à des noms utilisés comme **clés** (au lieu des indices).
- Exemple :

```
Preferences["Saison"] ="Ete";
Preferences["Couleur"] = "Bleu";
Preferences["Jour"] = "Vendredi";
Preferences["Musique"] = "Jazz";
```

Plan du chapitre 4

- Introduction
- Insertion du code JavaScript dans une page HTML
- Conventions
- Variables
- Tableaux
- **Opérateurs**
- Tests logiques
- Boucles
- Fonctions
- Objets & propriétés
- Évènements

Opérateurs

Opérateurs mathématiques

+	permet d'additionner deux nombres ou de concaténer deux chaînes de caractères.
-	permet de faire la soustraction de deux nombres.
*	permet de faire la multiplication de deux nombres.
/	permet de faire la division du premier nombre par le second.
%	renvoie le reste de la division euclidienne de deux nombres.

Opérateurs

Opérateurs de comparaison

<	permet de tester si le premier nombre est strictement inférieur au second. Il renvoie true ou false.
<=	permet de tester si le premier nombre est inférieur ou égal au second. Il renvoie true ou false.
>	permet de tester si le premier nombre est strictement supérieur au second. Il renvoie true ou false.
>=	permet de tester si le premier nombre est supérieur ou égal au second. Il renvoie true ou false.
==	permet de tester l'égalité de deux nombres. Il renvoie true ou false.
!=	permet de tester si deux nombres sont différents. Il renvoie true ou false.
===	permet de tester à la fois le type et l'égalité de deux expressions : 1===2 renvoie false, . 1===1 renvoie true mais 1==="1" renvoie false, puisque l'on compare un nombre à une chaîne de caractères.

Opérateurs

Opérateurs logiques

&&	"et logique". Renvoie true quand les deux quantités sur laquelle il opère valent true. Par exemple, (1<2) && (chaine == "a") renvoie true si la variable chaine vaut "a", et false dans le cas contraire.
||	"ou logique". Renvoie true quand au moins une des deux quantités sur laquelle il opère vaut true. Par exemple, (1<2) || (chaine == "a") renvoie true dans tous les cas, car 1<2 est toujours vrai.

Opérateurs

Opérateurs unaires

Les opérateurs unaires s'appliquent à une seule quantité.

!	Opérateur logique de négation. Il s'applique aux booléens. Exemple : !(2<3) renvoie false.
++	Opérateur d'incrémentation : il ajoute 1 à la variable concernée. Cet opérateur peut être placé en préfixe (l'opération est réalisée avant tout calcul) ou en postfixe (l'opération est réalisée à la toute fin du calcul).
--	Opérateur de décrémentation : il soustrait 1 à la variable concernée. Cet opérateur peut être placé en préfixe (l'opération est réalisée avant tout calcul) ou en postfixe (l'opération est réalisée à la toute fin du calcul).

Plan du chapitre 4

- Introduction
- Insertion du code JavaScript dans une page HTML
- Conventions
- Variables
- Tableaux
- Opérateurs
- **Tests logiques**
- Boucles
- Fonctions
- Objets & propriétés
- Évènements

Tests logiques

Instruction if

- Instruction **if** simple :

```
if (choix == 1)
{
        alert ("Vous avez fait le premier choix") ;
}
```

- Instruction **if... else** :

```
if (choix == 1)
 {
        alert ("Vous avez fait le premier choix") ;
 }
else
 {
      alert ("Vous n'avez pas fait le premier choix") ;
 }
```

Tests logiques

Instruction switch

```
switch (choix)
{
    case 1:
            alert ("Vous avez fait le premier choix") ;
    break ;

    case 2:
            alert ("Vous avez fait le deuxième choix") ;
    break ;

    case 3:
            alert ("Vous avez fait le troisième choix") ;
    break ;

    default :
             alert ("Vous devez faire un choix entre 1 et 3") ;
}
```

Plan du chapitre 4

- Introduction
- Insertion du code JavaScript dans une page HTML
- Conventions
- Variables
- Tableaux
- Opérateurs
- Tests logiques
- **Boucles**
- Fonctions
- Objets & propriétés
- Évènements

Boucles

Boucle do... while

```
var compteur = 1;
do
 {
        compteur++;
        alert(compteur);
 }
while (compteur<n)
```

Boucles

Boucle while

```
var compteur = 1;
while (compteur<n)
 {
      compteur++ ;
      alert (compteur);
 }
```

Boucles

Boucle for

```
for (i=0; i<n; i++)
 {
    //Suite d'instructions...
 }
```

Boucles

Boucle for ... in

- Elle est souvent utilisée avec les tableaux.
- **Exemple :**

```
var Langages = new Array("C++", "Java",  "JavaScript", "C#", "Perl");
for (i in Langages)
{
   document.write(Langages[i] + "<hr />");
}
```

Plan du chapitre 4

- Introduction
- Insertion du code JavaScript dans une page HTML
- Conventions
- Variables
- Tableaux
- Opérateurs
- Tests logiques
- Boucles
- **Fonctions**
- Objets & propriétés
- Évènements

Fonctions

- Une fonction est un sous-programme qui permet d'effectuer un ensemble d'instructions par simple appel dans le corps du programme principal.
- Une fonction doit être définie avant d'être utilisée.
- La déclaration d'une fonction se fait grâce au mot-clé **function** selon la syntaxe suivante:

```
function Nom_De_La_Fonction (argument1, argument2, ...)
{
    // liste d'instructions
}
```

Fonctions

- On invoque une fonction avec son nom et sa liste d'arguments entre parenthèses :

  Nom_De_La_Fonction (argument1, argument2, ...);

- Une fonction peut éventuellement retourner une valeur à l'aide de l'instruction **return.**

```
function surfaceRectangle (longueur, largeur)
{
        return longueur*largeur ;
}
```

Fonctions

Fonctions prédéfinies

setTimeout() :

- permet de spécifier un temps après lequel une certaine action doit s'exécuter.
- Syntaxe :

```
setTimeout(MaFonction,duree) ;
// Appelle la fonction MaFonction après duree millisecondes.
```

- Exemple :

```
<script type="text/javascript">
        setTimeout("alert('Trente secondes sont passees !" );",30000);
</script>
```

Fonctions

Fonctions prédéfinies

clearTimeout() :

- permet d'arrêter une exécution avec **setTimeout()**.
- Syntaxe :

```
var vTimeout=setTimeout(MaFonction,duree) ;
clearTimeout(vTimeout) ;
```

Fonctions

Fonctions prédéfinies

setInterval() :

- Syntaxe :

```
setInterval(MaFonction,duree);
/* Permet de déclencher l'exécution de la fonction MaFonction
   toutes les duree millisecondes. */
```

- Elle peut être arrêtée par clearTimeout :

```
var vTimeout=setInterval(MaFonction,duree) ;
clearTimeout(vTimeout) ;
```

Plan du chapitre 4

- Introduction
- Insertion du code JavaScript dans une page HTML
- Conventions
- Variables
- Tableaux
- Opérateurs
- Tests logiques
- Boucles
- Fonctions
- **Objets & propriétés**
- Évènements

Objets & propriétés

Définition

- JavaScript traite les éléments qui s'affichent dans le navigateur comme des objets.
- Quatre objets prédéfinis sont instantanément associés à un document lors de son chargement dans le navigateur :
  - L'objet **window** représente la fenêtre affichée dans le navigateur.
  - L'objet **location** contient l'URL du document courant.
  - L'objet **document** donne accès aux propriétés du document : titre, couleur, image de fond…
  - L'objet **history** donne accès aux URL des pages précédemment visualisées.
- Les objets sont accessibles par une syntaxe hiérarchique "à point" :

  objet1.objet2.objet3
- Cette syntaxe à point se termine souvent par une propriété associée à un objet.

  window.document.form.resultat.value
- Un objet peut posséder plus qu'une seule propriété.

Objets & propriétés

Le mot-clé This

Le mot clé **this** fait référence à l'objet en cours et vous évite d'avoir à définir l'objet en tapant **window.objet1.objet2...** ainsi pour manipuler les propriétés de l'objet il suffira de taper **this.propriete** .

Objets & propriétés

Définition d'un nouveau type d'objet

- JavaScript permet de définir de nouveaux objets parfaitement adaptés à chaque situation.
- La création d'un nouvel objet se fait en deux temps :
  - Définition du type de l'objet à l'aide d'une fonction ;
  - Instanciation de l'objet avec l'opérateur **new**.
- **Exemple :**

  On désire créer un objet **Livre** avec les deux propriétés suivantes :
  - Titre ;
  - NombreDePages ;

Définition :	**Instanciation :**
function Livre (Titre, NomDePages) *{* *Livre.Titre=Titre;* *Livre.NombreDePages=NomDePages;* *}*	*livre1 = new Livre ('Programmation Web ', 102)*

Objets & propriétés

Méthodes

- Une méthode est une fonction associée à un objet.
- Les méthodes des objets du navigateur sont des fonctions prédéfinies par les normes HTML, qu'on ne peut pas les modifier.
- Il est toutefois possible de créer une méthode personnelle pour un objet que l'on a créé soi-même.
- L'appel d'une méthode se fait suivant la syntaxe suivante :

```
window.objet1.objet2.methode ()
```

Objets & propriétés

Objets prédéfinis

L'objet String :

- Propriétés de l'objet String :
  - Cet objet ne possède qu'une seule propriété, **length**, qui renvoie sa longueur.
  - **Exemple :**

```
chaine="Bonjour";
alert (chaine.length);     // renvoie 7.
```

- Méthodes de l'objet String :

Voir complément du cours : Tableau 1

Objets & propriétés

Objets prédéfinis

L'objet String :

- L'objet String possède aussi d'autres méthodes de formatage, à savoir :
  big(), blink(), bold(), fontcolor(), fontsize(), italics(), sub(), sup(), …
- **Exemple :**

```
<script type="text/javascript">
    var texte = "Lorem ipsum dolor sit amet !";
    document.write(texte.length + "<br />");
    document.write(texte.big()+ "<br />");
    document.write(texte.strike()+ "<br />");
    document.write(texte.bold()+ "<br />");
    document.write(texte + texte.sub());
    document.write(texte + texte.sup() + "<br />");
    document.write(texte.fontcolor("red") + "<br />");
    document.write(texte.replace("ipsum dolor","texte bidon")+ "<br />");
    document.write(texte.toLowerCase() + "<br />");
    document.write(texte.toUpperCase()+ "<br />");
    document.write(texte.slice(4,12)+"<br />")
    document.write(texte.split("m") + "<br />");
</script>
```

Objets & propriétés

Objets prédéfinis

L'objet Math :

- Propriétés de l'objet Math :
  - E est la constante d'Euler (environ 2,718)
  - PI est le nombre pi (environ 3,14159)
  - LN2 est le logarithme naturel de 2 (environ 0,693)
  - LN10 est le logarithme naturel de 10 (environ 2,302)
  - LOG2E est le logarithme en base 2 de e (environ 1,442)
  - LOG10E est le logarithme en base 10 de e (environ 0,434)
  - SQRT2 est la racine carrée de 2 (environ 1,414)
  - SQRT1_2 est la racine carrée de 1/2 (environ 0,707)
- Méthodes de l'objet Math :

Voir complément du cours : Tableau 2

Objets & propriétés

Objets prédéfinis

L'objet Math :

☐ Exemple :

```
<html>
<head>
<script language="javascript">
  function Carre ( X )
  {
      X = Math.pow( X , 2);
  }
</script>
</head>
<body>
<form id=nomForm>
     <p> Test de la fonction <b> carré </b> : </p>
     <input type=text name=resultat size=25 />
     <input type=button name=carre value=Carré
      onclick="Carre( document.getElementById("nomForm").resultat.value)" />
</form>
</body>
</html>
```

Objets & propriétés

Objets prédéfinis

L'objet Date :

- Constructeur de l'objet Date :
  - Le constructeur permet de créer une nouvelle « instance » de la classe Date.
  - Syntaxe :

```
var nouvDate = new Date(annee, mois, jour [, heure, minute, seconde, milliseconde]) ;
```

  - Les paramètres heure, minute, seconde et milliseconde sont optionnels.
  - Exemple :

```
var nouvelleDate = new Date(2003, 10, 11, 9, 32, 34, 123) ;
var nouvelleDate = new Date;   /*la date renvoyée est la date courante.*/
```

- Méthodes de l'objet Date :

Voir complément du cours : Tableau 3

Objets & propriétés

Objets prédéfinis

L'objet Date :

- Exemple :

```
<html>
<head>
<script language="javascript">
  function DateToday ()
  {
     Var days = new Array ("Sunday", "Monday", "Tuesday", "Thursday", "Friday",
     "Saturday");
     Var months = new Array ("January", "February", "March", "April", "May", "June",
     "July", "August", "September", "October", "November", "December");
     Var date = new Date();
     return ("<H4> " + days[date.getDay()]+" " + date.getDate()+" " +
     months[date.getMonth()]+" " +date.getFullYear()+"</H4>");
  }
</script>
</head>
<body>
     <script language="javascript">
          document.write(DateToday ());
     </script>
</body>
</html>
```

Objets & propriétés

Objets prédéfinis

L'objet Array :

- Propriété de l'objet Array :
  - length: nombre d'éléments dans le tableau.
- Méthodes de l'objet Array :
  - join() : liste des éléments du tableau.
  - pop() : enlève et retourne le dernier élément du tableau.
  - shift() : enlève et retourne le premier élément du tableau.
  - push() : ajoute un nouvel élément à la fin du tableau.
  - slice(debut, fin) : sous-tableau débutant à debut et se terminant à fin.
  - splice(debut,n) : enlève n éléments à partir de debut.
  - concat() : fait une jointure de deux tableaux.
  - reverse() : renverse l'ordre des éléments.
  - sort() : trie les éléments du tableau.
  - toString() : convertit un tableau en chaine.
  - unshift() : ajoute un élément au début du tableau.

Objets & propriétés

Objets prédéfinis

L'objet Array :

☐ Exemple :

```
var employes=new Array("Jean", "Pierre", "Alain", "Sophie", "Ali");
 var salaires=new Array(75000, 60000,80000, 60000, 65000);
var autresEmployes=new Array("Pedro", "Manuel", "Alejandro");
var tousLesEmployes=employes.concat(autresEmployes);
document.write("<b>Resultatde la concatenation: </b>"+tousLesEmployes+"<br>");
document.write("<b>Tableau des employes inverse :</b> "+employes.reverse()+"<br>");
var chaineEmployes1=employes.join(" - ");
var chaineEmployes2=employes.toString();
document.write("<b>Resultat du  join: </b>"+chaineEmployes1+"<br>");
document.write("<b>Resultat du  toString: </b>"+chaineEmployes2+"<br><br>");
document.write("<b>Tableau des salaires apres tri: </b>"+salaires.sort()+"<br>");
document.write("<b>Position du premier 60000:</b>"+salaires.indexOf(60000)+"<br>");
document.write("<b>Position du dernier 60000:</b>"+salaires.lastIndexOf(60000)+"<br>");
var nouveauxEmployes=employes.splice(2,0,"Michel","Shophie");
document.write("<b> Apres ajout de nouveaux employes:</b>" + employes +"<br>");
```

Objets & propriétés

Objets prédéfinis

L'objet Array :

☐ Exemple :

Résultat affiché

```
Resulat de la concatenation: Jean,Pierre,Alain,Sophie,Ali,Pedro,Manuel,Alejandro
Tableau des employes inverse : Ali,Sophie,Alain,Pierre,Jean
Resultat du join: Ali - Sophie - Alain - Pierre - Jean
Resultat du toString: Ali,Sophie,Alain,Pierre,Jean
Tableau des salaires apres tri: 60000,60000,65000,75000,80000
Position du premier 60000:0
Position du dernier 60000:1
Apres ajout de nouveaux employes:Ali,Sophie,Michel,Shophie,Alain,Pierre,Jean
```

Objets & propriétés

Objets prédéfinis

L'objet window :

- Propriétés de l'objet window :
  - *closed*: Indique si la fenêtre est fermée.
  - *document*: document de la fenêtre
  - *frames*: Tableau des cadre (frames) dans le document.
  - *history*: Historique de la fenêtre
  - *length*: Nombre de cadres (frames) dans la fenêtre
  - Location: Emplacement (URL) de la fenêtre
  - *name*: Nom de la fenêtre
  - *navigator*: Type de navigateur.
  - *opener*: Fenêtre qui a créé la fenêtre courante
  - *parent*: Parent de la fenêtre
  - *self*: La fenêtre elle -même
  - *status*: Barre d'état de la fenêtre
  - *top*: Fenêtre de plus haut niveau.

Objets & propriétés

Objets prédéfinis

L'objet window :

- Méthodes de l'objet window :

Voir complément du cours : Tableau 4

Objets & propriétés

Objets prédéfinis

L'objet window :

- Exemple :

```
<html>
<head>
<script type="text/javascript">
function ouvrirFenetre(){
  fenetre=window.open('','','width=400,height=200');
  fenetre.document.write("<h3>Une nouvelel fenetre est creee</h3>");
  fenetre.document.write("<a href=''>Un lien est cree </a>");
  fenetre.focus();
}
function fermerFenetre() {fenetre.close(); }
function deplacerFenetre() {fenetre.moveBy(-100,-200);   fenetre.focus();}
</script>
</head>
<body>
<input type="button" value="Ouvrir" onclick="ouvrirFenetre()" />
<input type="button" value="Fermer" onclick="fermerFenetre()" />
<input type="button" value="Deplacer" onclick="deplacerFenetre()" />
</body>
</html>
```

Objets & propriétés

Objets prédéfinis

L'objet document :

- Représente le contenu de la fenêtre et possède plusieurs collection d'objets tels que :
  - *anchors[]* : tableau des ancrages dans le document
  - *forms[]* : tableau des formulaire dans le document.
  - *images[]* : tableau des images dans le document.
  - *links[]* : tableau des liens dans le document.
- **Méthodes de l'objet document :**
  - *write()* et *writeln()* pour modifier le contenu d'un document
  - *close()* : ferme la communication avec le document
  - *getElementById()* : l'élement ayant un identificateur donné
  - getElementsByName() : tous les éléments identifiés par un nom (attribut *name*) donné
  - *getElementsByTagName()* : les éléments ayant une balise donné

Objets & propriétés

Objets prédéfinis

L'objet Image :

☐ **Propriétés de l'objet Image :**

- **src :** adresse de l'image
- **width :** largeur de l'image
- **height :** hauteur de l'image
- **complete :** indique si l'image est totalement chargée (true) ou non (false)

☐ **Évènements associés à l'objet Image :**

- **onLoad :** se produit une fois l'image est entièrement chargée
- **onError :** se produit si l'image ne peut pas être chargée
- **onAbort :** se produit lorsque le chargement de l'image est interrompu

Objets & propriétés

Objets prédéfinis

L'objet Image :

- Exemple :

```
<html>
<head>
<script language="javascript">
        function ChangerImage  (imageAM,adresseNI)
        {
                Var image = new Image ();
                 Image.onerror = function ()
                {
                        alert("Erreur lors du chargement de l'image");
                }
                Image.onabort = function ()
                {
                        alert("Chargement interrompu");
                 }
                Image.onload = function ()
                {
                        imageAM.src = image.src;
                        imageAM.width = image.width ;
                        imageAM.height = image.height ;
                 }
                 image.src = adresseNI ;
        }
</script>
</head>
<body>
      <img src="image1.png"  id="monImage"  onclick=" ChangerImage
      (document.getElementById('monImage '),'image2.png');" />
</body>
</html>
```

Plan du chapitre 4

- Introduction
- Insertion du code JavaScript dans une page HTML
- Conventions
- Variables
- Tableaux
- Opérateurs
- Tests logiques
- Boucles
- Fonctions
- Objets & propriétés
- **Évènements**

Évènements

- Les évènements sont des actions de l'internaute, qui vont pouvoir donner lieu à une interactivité (clic de souris, passage de la souris au-dessus d'une zone, changement d'une valeur...).

- Grâce aux gestionnaires d'événements, on peut associer une action à un événement selon la syntaxe suivante:

```
onEvenement ="Action_Javascript_ou_Fonction ();"
```

Évènements

Liste des événements

Evénement (gestionnaire d'événement)	Description
Click (onClick)	Se produit lorsque l'utilisateur clique sur l'élément associé à l'événement
Load (onLoad)	Se produit lorsque le navigateur de l'utilisateur charge la page en cours
Unload (onUnload)	Se produit lorsque le navigateur de l'utilisateur quitte la page en cours
MouseOver (onMouseOver)	Se produit lorsque l'utilisateur positionne le curseur de la souris au-dessus d'un élément
MouseOut (onMouseOut)	Se produit lorsque le curseur de la souris quitte un élément Cet événement fait partie du Javascript 1.1
Focus (onFocus)	Se produit lorsque l'utilisateur donne le focus à un élément, c'est-à-dire que cet élément est sélectionné comme étant l'élément actif
Blur (onBlur)	Se produit lorsque l'élément perd le focus, c'est-à-dire que l'utilisateur clique hors de cet élément, celui-ci n'est alors plus sélectionné comme étant l'élément actif
Change (onChange)	Se produit lorsque l'utilisateur modifie le contenu d'un champ de données
Select (onSelect)	Se produit lorsque l'utilisateur sélectionne un texte (ou une partie d'un texte) dans un champ de type "text" ou "textarea"
Submit (onSubmit)	Se produit lorsque l'utilisateur clique sur le bouton de soumission d'un formulaire (le bouton qui permet d'envoyer le formulaire)

Évènements

Objets auxquels sont associés des événements

Objet	Evénements associables
Lien hypertexte	onClick, onMouseOver, onMouseOut
Page du navigateur	onLoad, onUnload
Bouton, Case à cocher, Bouton radio	onBlur, onClick, onFocus
Liste de sélection d'un formulaire	onBlur, onChange, onFocus
Bouton Submit, Bouton Reset	onSubmit, onReset
Champ de texte et zone de texte	onBlur, onChange, onFocus, onSelect