

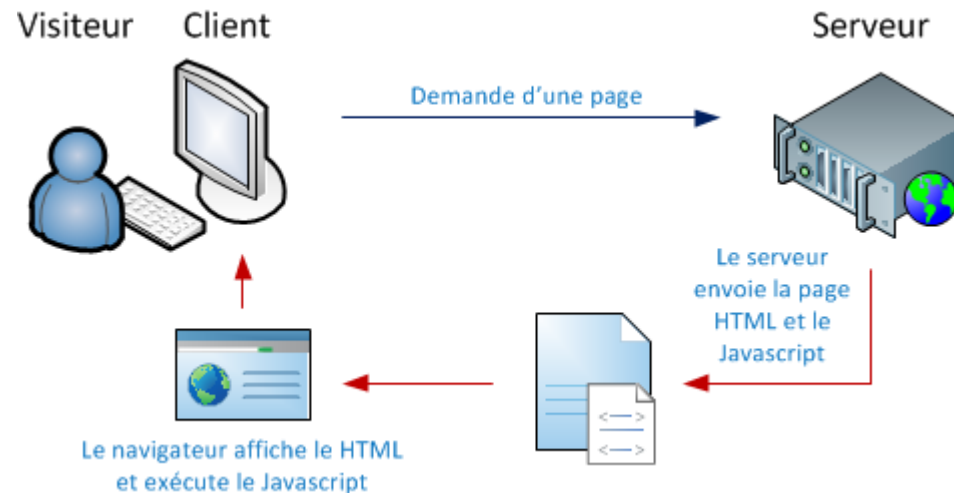
Le langage Javascript pour documents HTML

JavaScript : Principe

- Le JavaScript est un langage de programmation de scripts orienté objet.
- Le JavaScript est utilisé sur Internet, conjointement avec les pages Web HTML.
- Le JavaScript s'inclut directement dans la page Web (ou dans un fichier externe) et permet de *dynamiser* une page HTML :
 - Afficher/masquer du texte ;
 - Faire défiler des images ;
 - Créer un diaporama avec un aperçu « en grand » des images ;
 - Etc.

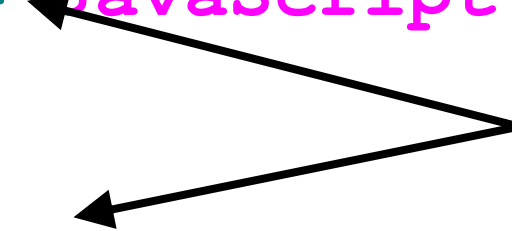
JavaScript : Principe

- Le JavaScript est un langage dit **client-side**, c'est-à-dire que les scripts sont exécutés par le navigateur chez l'internaute (le **client**). Cela diffère des langages de scripts dits **server-side** qui sont exécutés par le serveur Web. C'est le cas des langages comme le [PHP](#).



JavaScript : Balise script

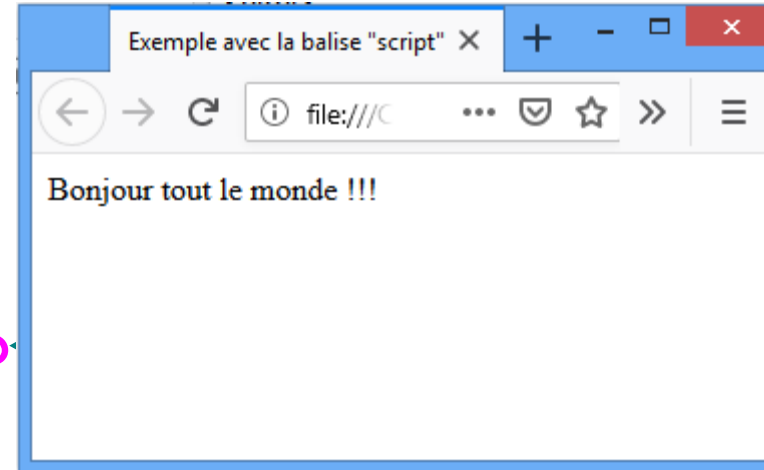
```
<script type="text/javascript"  
  language="JavaScript">  
<!--  
script  
// -->  
</script>
```



Masquer le script aux navigateurs non compatibles avec JavaScript

JavaScript : Exemple

```
<html>
  <head>
    <title>Ma première page Web
  </head>
  <body>
    <script type="text/javascript" language="JavaScript">
      <!--
        document.writeln("Bonjour tout le monde !") ;
      // -->
    </script>
  </body>
</html>
```



JavaScript : un fichier externe (.js)

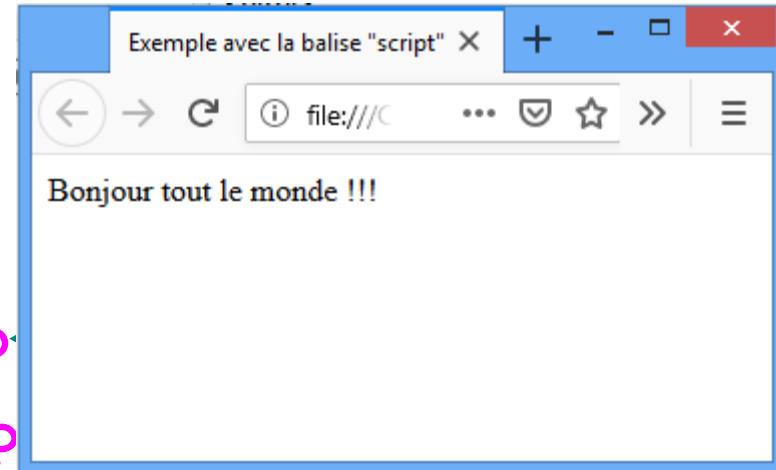
```
<script type="text/javascript"  
  language="JavaScript" src="URI">  
</script>
```



Le chemin à emprunter pour accéder
au fichier

JavaScript : Exemple

```
<html>
  <head>
    <title>Ma première page Web
  <script type="text/javascript"
    src="monscript.js">
  </script>
</head>
<body>
</body>
</html>
```



```
document.writeln("Salut!") ;
```

Variables

- Déclaration de variables facultative
- Variables non typées à la déclaration

`var nom_variable ;`

- Typage dynamique à l'affectation
- Types gérés:
 - Nombres (10, 3.14)
 - Booléens (true, false)
 - Chaînes ("Salut !", 'Salut !')
 - null
 - undefined

Les opérateurs arithmétiques

- Les opérateurs de base sont

Opérateur	Signe
addition	+
soustraction	-
multiplication	*
division	/
modulo	%

- Concernant le dernier opérateur, le modulo est tout simplement le reste d'une division.
- Exemple : 5 modulo 2 vaut 1.

La concaténation

- Une concaténation consiste à ajouter une chaîne de caractères à la fin d'une autre, comme dans cet exemple :

```
<html>
  <head>
    <title>Ma première page Web</title>
  </head>
  <body>
    <script type="text/javascript" language="JavaScript">
      var ch1="bonjour",ch2="Ali";
      message=ch1+ch2;
      Alert(message);
    </script>
  </body>
</html>
```

Interagir avec l'utilisateur

- La fonction `prompt()` permet l'interaction avec l'utilisateur :
- Exemple :

```
<html>
  <head>
    <title>Ma première page Web</title>
  </head>
  <body>
    <script type="text/javascript" language="JavaScript">
      var Nom = prompt('Entrez votre prénom :');
      alert(Nom); // Affiche le prénom entré par
                  l'utilisateur
    </script>
  </body>
</html>
```

Exercice1

- Ecrire un code qui calcule la somme de deux entiers entrés par l'utilisateur.

```
<html>
  <head>
    <title>Ma première page Web</title>
  </head>
  <body>
    <script type="text/javascript" language="JavaScript">
      var n1 = parseInt(prompt('Entrez un entier:'));
      var n2 = parseInt(prompt('Entrez un entier:'));
      var s=n1+n2;
      alert("s="+s); // Affiche le prénom entré par
      l'utilisateur
    </script>
  </body>
</html>
```

Instructions conditionnelles

- Il s'agit des instructions à utiliser pour tester les conditions.
- Les opérateurs de comparaison :

Opérateur	Signification
==	égal à
!=	différent de
===	contenu et type égal à
!==	contenu ou type différent de
>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à

Instructions conditionnelles

- Les opérateurs logiques :

Opérateur	Type de logique	Utilisation
&&	ET	valeur1 && valeur2
	OU	valeur1 valeur2
!	NON	!valeur

Instructions conditionnelles

- L'instruction if :

```
if (condition)
```

```
{
```

```
    instructions ;
```

```
}
```

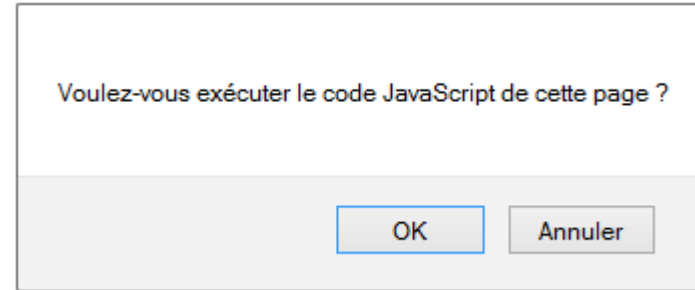
- Exemple :

```
    if confirm("voulez-vous exécuter le code  
                JavaScript de cette page?")
```

```
    {
```

```
        Alert("Le code a bien été exécuté");
```

```
    }
```



Instructions conditionnelles

- Si on souhaite exécuter un code si la condition est fausse :

```
if (condition)
{
    instructions 1 ;
}
[ else
{
    instructions 2;
} ]
```


Instructions conditionnelles

- Exemple :

```
if confirm("pour accéder à ce site vous  
devez avoir 18 ans ou plus cliquez sur ok  
si c\'est le cas'.")){  
    Alert("Vous allez être redirigé vers le  
site.");  
}  
else{  
    Alert("Désolé, vous n\'avez pas accès à ce  
site.« ) ;  
}
```

Instructions conditionnelles

- Si vous avez plusieurs cas à traiter, on peut utiliser l'instruction `else if` pour traiter à chaque fois le sinon d'une condition :

```
if (condition 1){  
    instructions 1 ;  
}  
  
else if (condition 2) {  
    instructions 2;  
}  
  
else {  
    instructions 3;  
}
```

Instructions conditionnelles

- C'est une instruction qui permet de traiter plusieurs cas (plus pratique que else si) :

```
switch (expression) {  
    case étiquette :  
        instructions ;  
        break ;  
    case étiquette :  
        instructions ;  
        break ;  
    default :  
        instructions ;  
}
```

Exercice d'application

- Écrire un code qui permet de fournir un commentaire sur quatre tranches d'âge selon le tableau ci-dessous :

Tranche d'âge	Exemple de commentaire
1 à 17 ans	« Vous n'êtes pas encore majeur. »
18 à 49 ans	« Vous êtes majeur mais pas encore senior. »
50 à 59 ans	« Vous êtes senior mais pas encore retraité. »
60 à 120 ans	« Vous êtes retraité, profitez de votre temps libre ! »

Instructions répétitives

```
while (condition)
```

```
{
```

```
    instructions ;
```

```
}
```

```
do
```

```
{
```

```
    instructions ;
```

```
}
```

```
while (condition) ;
```

Instructions répétitives

```
for (instr ; condition ; instr)  
{  
    instructions ;  
}
```

```
for (variable in objet)  
{  
    instructions ;  
}
```

L'utilisation de break

- C'est une instruction qui permet de quitter la boucle.
- Exemple :

```
Mot_de_passe=prompt("Vous trois essais !!!");
Compteur=0;
while (i<3){
    if(mot_de_passe!="123"){
        Alert("mot de passe invalide il vous reste "+compteur+"
        essais");
        compteur++;
    }
    else {
        break;}
}
If(i==3){Alert("Désolé tu as passé le nombre d'essais");}
```

Commentaires

```
// Commentaire ligne
```

```
/* Commentaire multi-lignes */
```


Les fonctions

- Nous avons déjà vu quatre fonctions : `alert()`, `prompt()`, `confirm()` et `parseInt()`.
- Il s'agit des fonctions prédéfinies, vous pouvez créer vos propres fonctions.
- La syntaxe :

```
function ma_fonction(arguments)
{
    instructions ;
    return quelque_chose; // ou pas...
}
```

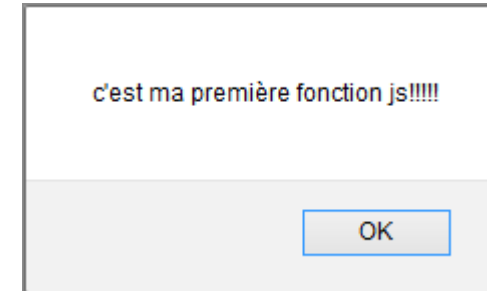
```
ma_fonction(12) ; // Appel
```

Les fonctions

- Exemple sans arguments:

```
function affiche_message()  
{  
    alert("C'est ma première fonction JS");  
}
```

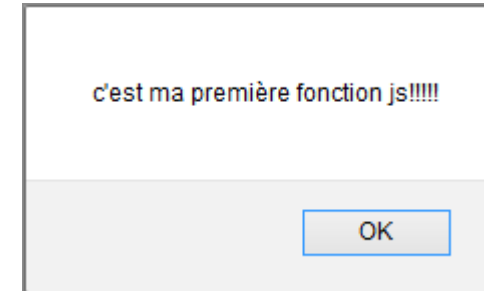
```
Affiche_message() ; // Appel
```



Les fonctions

- Exemple avec arguments:

```
function affiche_message(message)
{
    alert(message);
}
Affiche_message("C'est ma première
fonction js!!!") ; // Appel
```



Quelques consignes sur les fonctions

- Valeur de retour non typée
- Arguments non types
- Les variables déclarées avec au sein d'une fonction ne sont accessibles que dans cette fonction.
- Il faut éviter le plus possible d'avoir recours aux variables globales.
- Une fonction peut recevoir un nombre défini ou indéfini de paramètres.
- Elle peut aussi retourner une valeur ou ne rien retourner du tout.

Les fonctions et la portée des variables

- Si une variable est déclarée avec le mot clé **var** en dehors de la fonction, elle aura une portée globale, et si elle est déclarée dans la fonction, elle aura une portée locale à cette fonction. Par contre, si une variable est déclarée sans le mot clé **var**, elle aura une portée globale quelque soit l'endroit où elle a été déclarée.
- Exemple :

```
a=10;
var b=20;
function f(){
  c=30;
  var d=40;
  alert(a); // Affiche 10
  alert(b); // Affiche 20
}
f();
alert(c); // Affiche 30
alert(d); // Erreur
```

Les événements en Javascript

- Javascript est un langage événementiel. Donc les scripts déclarés dans la page ne sont pas destinés à s'exécuter séquentiellement, ligne après ligne, du début à la fin. Mais ils attendent jusqu'à ce qu'un événement soit détecté pour qu'une partie du code s'exécute.
- Le **clic**, sur un objet quelconque (lien, image, bouton...), est un exemple d'**événement**. Le navigateur écoute les événements qui se produisent sur la page Web et exécute le script qui correspond à cet événement.
- La syntaxe :

onÉvénement = "code Javascript édité directement ici, ou fonction appelée à cet endroit"

Les événements en Javascript

- Exemple de l'événement **click** :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function f(){
        alert("Vous avez bien cliqué sur le bouton!");
      }
    </script>
  </head>
  <body>
    <input type="button" value="Cliquez ici" onClick="f()" />
  </body>
</html>
```

Liste des événements Javascript

Événement	Description
click (onClick)	Cet événement est capturé sur un objet quand on clique dessus. Idéal pour les boutons, images, hyperliens, vidéos...
dblClick (onDblClick)	Quand on clique sur un objet deux fois de suite (double clic).
mouseOver (onMouseOver)	Quand on survole un objet avec le curseur de la souris.
mouseOut (onMouseOut)	Quand on quitte l'objet avec la souris après l'avoir survolé.
focus (onFocus)	Quand on active un élément (quand on place le curseur dans un champ de formulaire, par click ou par tabulation, pour commencer la saisie par exemple).

Liste des événements Javascript (suite)

Événement	Description
blur (onBlur)	Quand un élément perd le focus (quitter un champ de formulaire après être activé par exemple).
keyDown (onKeyDown)	Quand une touche du clavier est enfoncée.
keyUp (onKeyUp)	Quand une touche du clavier est relâchée.
keyPress (onKeyPress)	Quand une touche du clavier est maintenue enfoncée.
load (onLoad)	Quand un élément est chargé par le navigateur. Elle peut être appliquée à la page entière (balise <body>), dans ce cas l'événement se produira quand tous les éléments de la page seront chargés.

Liste des événements Javascript (suite)

Événement	Description
unLoad (onUnload)	Quand la page en cours est fermée ou quittée.
resize (onResize)	Quand l'internaute redimensionne la taille de la fenêtre du navigateur.
select (onSelect)	Quand l'internaute essaie de sélectionner le texte contenu dans l'objet accueillant l'événement.
change (onChange)	Quand l'internaute change le contenu d'un élément (liste de sélection ou zone de texte par exemple).
submit (onSubmit)	Quand l'internaute clique sur n'importe quel bouton de type submit présent dans la page (ou dans le formulaire).
mouseDown (onMouseDown)	Quand l'internaute appuie sur n'importe quel bouton de la souris.

Liste des événements Javascript (suite)

Événement	Description
mouseUp (onMouseUP)	Quand le bouton de la souris est relâché.
mouseMove (onMouseMove)	Quand l'internaute fait bouger le curseur de la souris dans la zone accueillant l'événement.
dragStart (onDragStart)	Se produit quand l'internaute commence le déplacement d'un élément par "glisser-déposer" (Drag & Drop).
dragEnter (onDragEnter)	Se produit quand l'internaute entre dans la zone où sera déposé l'élément glissé.
dragOver (onDragOver)	Se produit quand l'internaute survole la zone où sera déposé l'élément glissé (même si le concept est proche de celui de onDragEnter , ils sont des événements légèrement différents).
drop (onDrop)	Se produit quand l'internaute dépose l'élément glissé dans la zone prévue à cet effet.

Liste des événements Javascript

- Exemple :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function f(){
        alert("Vous avez cliqué sur le bouton 1.");
      }
      function g(){
        alert("Vous avez survolé le bouton 2.");
      }
      function h(){
        alert("Vous avez cliqué 2 fois sur le bouton 3.");
      }
    </script>
  </head>
  <body onLoad="alert('Page chargée.')">
    <input type="button" value="Bouton 1" onClick="f()" /><br />
    <input type="button" value="Bouton 2" onMouseOver="g()" /><br />

    <input type="button" value="Bouton 3" onDbClick="h()" /><br />
  </body>
</html>
```

Exercice :

- Améliorer le code de l'exemple précédent, au lieu de créer 3 fonctions, créer une seule.

Objets, méthodes et attributs

- Un **objet** peut représenter un élément HTML (image, texte, formulaire...) ou un élément propre au Javascript qui sert à exécuter des traitements spéciaux.
- Une **méthode** est une **fonction** prédéfinie appartenant à l'objet et qui permet de faire les traitement sur celui-ci (on peut dire qu'elle décrit le comportement de l'objet).
- Un **attribut** est une **variable** qui appartient à l'objet . Elle représente une propriétés ou caractéristique de celui-ci. On peut le manipuler grâce aux **méthodes** ou directement.
- **Exemple** : L'objet le plus élevé est la fenêtre du navigateur, il est identifié par le nom **window**. Il possède des méthodes comme **alert()** qui permet d'afficher des messages.

Objets, méthodes et attributs

- L'objet **window** contient un sous-objet, c'est le document HTML lui même. Cet objet est identifié par le nom **document**. Cet objet peut contenir les éléments HTML conventionnels comme du texte, les images, les formulaires... Tous ces éléments constituent des sous-objets de l'objet **document**.
- Par exemple l'objet **forms** désigne les formulaires contenus dans le document, et l'objet **images** englobe toutes les images intégrées...

Comment accéder aux différents objets

- Exemple :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
  </head>
  <body>
    <form name="fo">
      <input type="text" name="login" />
      <input type="button" value="Action" />
    </form>
  </body>
</html>
```


Comment accéder aux différents objets

- La déclaration de l'objet **window** n'est pas obligatoire.
- Pour l'exemple précédent, l'objet formulaire est identifié par son **fo** donc pour accéder à cet objet on utilise la syntaxe suivante : **document.fo**
- Idem pour la zone de texte login : **document.fo.login**.

Comment accéder aux attributs (propriétés)

- La même syntaxe : **Document.nomObjet.propriété.**
- **Exemple** : les attributs de l'objet "zone de texte" que nous avons identifié par **document.fo.login** :
 - **value**: qui désigne la valeur que contient la zone de texte.
 - **size**: qui désigne sa taille.
 - **etc.**
- En général c'est l'attribut **value** qui est souvent utilisé. Pour accéder à la valeur de la zone de texte on fait **document.fo.login.value**.

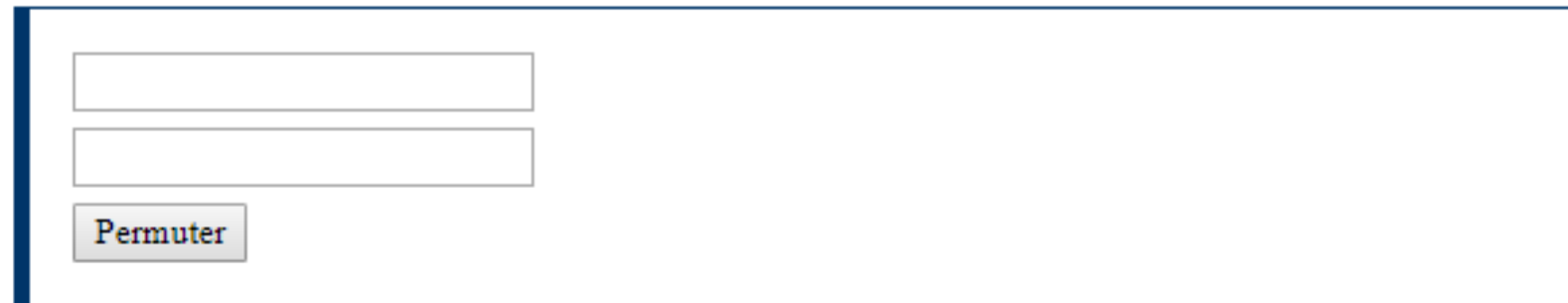
Comment accéder aux attributs (propriétés)

- Exemple d'application :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function f(){
        alert(document.fo.login.value);
      }
    </script>
  </head>
  <body>
    <form name="fo">
      <input type="text" name="login" />
      <input type="button" value="Action" onClick="f()" />
    </form>
  </body>
</html>
```

Exercices


- **1. Permutation** : Il s'agit d'un exercice des plus classiques en programmation. L'objectif est de créer un formulaire qui contient deux zones de texte et un bouton de commande. Le fait de cliquer sur le bouton permute le contenu des deux zones de texte.



The image shows a web form with a blue border. Inside the form, there are two text input fields stacked vertically on the left side. Below these fields is a button labeled 'Permuter' in a grey box. The rest of the form area is empty.

Exercices

- **2. Calculatrice simple** : Dans cet exercice, on va essayer de créer une calculatrice simple qui exécute les opérations basiques, à savoir, addition, soustraction, multiplication et division. La page contiendra trois zones de texte qui représenteront respectivement: nombre 1, nombre 2 et résultat de l'opération, ainsi que 4 boutons qui représenteront les 4 opérations prévues..



The image shows a simple calculator interface within a blue-bordered box. It consists of three text input fields stacked vertically on the left. Between the second and third input fields, there are four square buttons with icons for addition (+), subtraction (-), multiplication (*), and division (/). The third input field is positioned below these buttons.