# Javascript

# I. De l'Algo au Javascript

## 1. Déclaration et utilisation de variables

```
# Algo
Variable a en Entier
a ← 12
```

```
// Javascript
var a;
a = 12 ;
```

```
# Algo
Variable nom en Caractères
nom ← "Toto"
```

```
// Javascript
var nom;
nom = "Toto";
```

## 2. Convertir une chaine en valeur numérique

```
// Javascript
var a = "12";
var c = 12;
var b = parseInt(a);
```

## 3. Problème avec la concaténation ou l'addition

```
// Javascript
var a = "123";
var b = "45";
var c = a+b; // c contient "12345"
// Javascript
var a = 123;
var b = 45;
var c = a+b; // c contient 168
// Javascript
var a = "123";
var b = "45";
var c = parseInt(a)+parseInt(b); // c contient 168
```

## 4. Lire et Ecrire

```
# Algo
Variable nom en Caratère
Ecrire "Entrez votre nom"
Lire nom
```

```
// Javascript
var nom;
nom = prompt("Entrez votre nom");
```

```
# Algo
Variable a en Entier
Ecrire "Entrez un nombre"
Lire a
```

```
// Javascript
var a;
a = parseInt(prompt("Entrez un
nombre"));
```

```
# Algo
Variable message en Caractère
message ← "Bonjour toi"
Ecrire message
```

```
// Javascript
var message = "Bonjour toi";
alert(message);
```

## 5. Conditions

```
# Algo
Si a>12 Alors
    Ecrire "C'est beaucoup"
Sinon
    Ecrire "C'est peu"
```

```
// Javascript
if (a>12) {
    alert("C'est beaucoup");
}
else {
    alert("C'est peu");
}
```

```
# Algo
Si a>12 Alors
    Ecrire "C'est beaucoup"
SinonSi a<0
    Ecrire "C'est très peu"
Sinon
    Ecrire "C'est moyen"
```

```
// Javascript
if (a>12) {
    alert("C'est beaucoup");
}
else if (a<0) {
    alert("C'est très peu");
}
else {
    alert("C'est moyen");
}
```

ATTENTION : pour comparer si deux valeurs sont égales, vous devez utiliser `==`

Exemple

```
// Javascript
var a;

a = 12; // Affectation

if (a==7) { // Comparaison
    alert("Bien essayé !");
}
else {
    alert("Dommage !");
}
```

## 6. Exécuter le code

- Pour Firefox : cliquer sur les touches `MAJ` + `F4` pour lancer l'ardoise Javascript. Saisir votre code javascript puis touches `CTRL` + `R` pour l'exécuter.
- Pour Chrome, installer l'extension ScratchJS. Cliquer sur `F12`, puis sur l'onglet `ScratchJS` du panneau qui s'ouvre (panneau appelé outils développeurs/console). Saisir votre code javascript puis cliquer sur le bouton `Run` (ou touches `Ctrl` + `Entrée`).

# II. Structure d'un programme

## 1. Objectifs

- Comprendre les généralités de la programmation et savoir créer un premier programme.
- Connaître les règles et les conventions de syntaxe et de présentation d'un programme.
- Comment déclarer des variables et des constantes, comment les utiliser dans le corps d'un programme.
- Ajouter des commentaires à vos programmes

## 2. Les variables

Les données en mémoire sont stockées dans des variables. Il peut par exemple s'agir de données saisies par l'utilisateur ou de résultats obtenus par le programme, intermédiaires ou définitifs.

Pour employer une image, une variable est une boîte, que le programme (l'ordinateur) va repérer par une étiquette. Pour avoir accès au contenu de la **boîte**, il suffit de la désigner par son étiquette.

Les noms de variables peuvent être choisis librement par le programmeur (sauf parmi les mots-clés). La liste des mots-clés se trouve dans la documentation en ligne, mais dans un problème, une variable est choisie pour jouer un rôle. Il est souhaitable que l'auteur de l'algorithme s'en souvienne en choisissant des noms évocateurs et en respectant, autant que possible, l'unicité du rôle pour chaque variable.

Les caractères suivants peuvent être utilisés :

- lettres majuscules : de `A` à `Z`
- lettres minuscules : de `a` à `z` (les minuscules sont considérées comme des caractères différents des majuscules)
- chiffres, de 0 à 9
- les caractères trait souligné (underscore) : `_` et arobase : `@`

Exemple : nom des variables à retenir dans un problème de calcul de moyenne de notes :

Exemple : nom des variables à retenir dans un problème de calcul de moyenne de notes :

- `totalNotes` : total des notes d'un(e) élève
- `nombreNotes` : nombre des notes de cet(te) élève
- `moyenneNotes` : moyenne des notes de cet(te) élève

Remarquez le style de nommage des variables : le premier mot est entièrement en lettre minuscules, le second mot commence par une majuscule; on appelle ça la convention *camelCase*.

Une variable est typée, c'est-à-dire qu'elle possède **un type** : elle peut représenter, par exemple, un nombre (entier ou décimal) ou une chaîne de caractères (nous verrons plus tard d'autres types).

Les chapitres suivants vous donneront plus de détails sur l'ensemble des types de données en Javascript.

Une variable peut être déclarée à tout moment à l'intérieur du corps du programme. Cependant, la déclaration et l'initialisation d'une variable doivent impérativement précéder la première utilisation de celle-ci. Il est également et généralement préférable de déclarer les variables en début de bloc (juste après l'accolade ouvrante) pour une question de lisibilité.

Chaque déclaration de variable est construite sur le modèle suivant :

```
Type nomDeLaVariable ;
```

Une variable peut être initialisée lors de sa déclaration :

```
type nomDeLaVariable = valeurInitiale;
```

Les variables ne sont visibles que dans le bloc d'instructions - bloc défini par des accolades `{` et `}` - dans lequel elles sont définies.

Comme toutes les instructions du langage Javascript, chaque déclaration de variable DOIT absolument être terminée par un point-virgule `;`.

Le point-virgule ne constitue pas un séparateur, mais plutôt un terminateur d'instructions.

## 3. Les instructions

Une instruction est une ligne de traitement.

Le caractère point-virgule `;` est un terminateur. TOUTES les instructions doivent se terminer par un `;`.

Les différents identificateurs sont séparés par un séparateur.

Les séparateurs peuvent être indifféremment l'espace, la tabulation et le saut de ligne.

## 4. Les commentaires

Les commentaires dans un programme servent :

- à donner aux développeurs des indications sur le fonctionnement d'un programme
- à désactiver (temporairement) des blocs de code

**Les parties de code mises en commentaires ne sont pas exécutées.**

- Commentaires sur une seule ligne : on utilise les signes `//` :

```
// Demande le nom de famille
var nom = prompt("Entrez votre nom");
```

Dans cet exemple, on fournit une information complémentaire au code (1ère ligne); le code de la seconde ligne sera exécuté (demande à l'utilisateur d'entrer un nom).

- Commentaires sur plusieurs lignes : les caractères compris entre `/* et*/` ne seront pas interprétés. On utilise cette notation quand il y a plusieurs lignes à commenter.

```
/* Demande le nom de famille
var nom = prompt("Entrez votre nom");
*/
```

Dans cet exemple, les 2 lignes sont en commentaires, cette fois le code de la seconde ligne ne sera **pas exécuté**.

### 5. Les conventions d'écriture

Ces conventions ne sont pas des contraintes de syntaxe des langages informatiques. Elles n'existent que pour en faciliter la lecture et font partie d'une sorte de norme implicite que tous les bons développeurs s'obligent à respecter :

- Une seule instruction par ligne. Même si tout un programme peut-être écrit sur une seule ligne.
- Les délimiteurs d'un bloc `{` et `}` doivent se trouver sur des lignes différentes et être alignés sur la première colonne de sa déclaration.
- A l'intérieur d'un bloc `{}` les instructions sont indentées (décalées) par un caractère tabulation ou un espace.
- A l'intérieur d'un bloc `{}` la partie déclaration des variables et la partie instructions sont séparées par une ligne vide.

## III. Introduction au langage Javascript

### 1. Présentation

JavaScript est un langage de scripts intégré aux pages web (HTML) qui permet d'améliorer la présentation et surtout l'interactivité des pages web.

JavaScript rend les pages web « dynamiques » en manipulant le Document Object Model (D.O.M.), c'est-à-dire la représentation objet de la page web (le « document »). Le Javascript est en effet un langage orienté objet et possède des objets dits natifs (par exemple les objets `Window`, `Date`, `XmlHttpRequest`). Possibilités offertes par le Javascript :

- Animer du texte et des images
- Ajouter des conditions, des boucles, effectuer des calculs
- Intercepter les événements (souris, clavier etc.)
- Contrôler les formulaires
- Modifier les caractéristiques des éléments HTML et CSS
- Gérer des menus
- Détecter le navigateur
- Aller chercher des données de façon asynchrone (avec Ajax et l'objet `XmlHttpRequest`).

Aujourd'hui, il existe des librairies et frameworks en Javascript aussi nombreux que perfectionnés tels que Jquery, AngularJS, VueJS, ReactJS, MeteorJS ou encore NodeJS, une application Javascript côté serveur.

**JavaScript est un langage sensible à la casse. Il est donc obligatoire de respecter les majuscules et les minuscules dans le code ou les instructions (évènements, fonctions natives).**

## 2. Un peu d'histoire

Au milieu des années 1990, la société *Netscape* développe pour son navigateur web homonyme un langage nommé *LiveScript*, lequel est repris par la société *Sun* (elle-même rachetée depuis par Oracle) à l'origine de Java, qui lui donne le nom de *Javascript*, mais il faut bien retenir que ces 2 langages sont totalement différents (contrairement à ce qu'on peut lire parfois), leurs seuls points communs étant leur nom commercial en lien avec la société *Sun*.

De son côté, *Microsoft* créée un langage similaire, le *JScript*, pour son navigateur *Internet Explorer*. Ce langage sera abandonné quelques années plus tard.

Aujourd'hui, le Javascript est standardisé sous la norme E.C.M.A. qui en est à la version ES6 également connue sous le nom ECMAScript2015 et qui est implémentée par tous les navigateurs récents.

##Syntaxe

### Bases syntaxiques

Pour l'écriture des instructions JavaScript, on utilisera l'alphabet ASCII classique (à 128 caractères) comme en HTML. Les noms de variables et de fonctions ne peuvent comporter d'espaces, d'accents ni de tirets. Seul le undercore (_) est autorisé. La bonne pratique est d'utiliser la méthode camelCase.

Une instruction Javacript est écrite sur une ligne et se termine par un point-virgule : alert("Hello l'AFPA !");

Enfin, JavaScript ignore les espaces, les tabulations et les sauts de lignes. Pour déclarer une chaîne de caractères, les guillemets " et l'apostrophe ' peuvent être utilisés à condition de ne pas les mélanger. Si vous souhaitez utiliser des guillemets dans vos chaînes de caractères, tapez \" ou \' pour les différencier vis à vis du compilateur. Mots réservés

En javascript comme dans les autres langages, il existe des mots réservés correspondant aux instructions du langage qu'il ne faut donc pas utiliser donc comme nom de variables

## 3. Sensibilité à la casse

JavaScript est un langage sensible à la casse. Il est donc obligatoire de respecter les majuscules et les minuscules dans le code ou les instructions (évènements, fonctions natives).

## 4. Commentaires

+++ TODO : DOUBLONS AVEC JS 01 +++

Les commentaires sur une ligne débutent par un double slash :

// Cette ligne est commentée alert("Ce message sera affiché");

Pour commenter un bloc de code de plusieurs lignes, les commentaires débutent par / *et se terminent par* / :

/ *Bloc de plusieurs lignes en commentaires alert("Hello l'AFPA !"); console.log("Ce message ne sera pas affiché car commenté dans le code");* /

# IV. Les variables

## 1. La déclaration de variables

Les variables contiennent des données qui peuvent être modifiées lors de l'exécution d'un programme. On y fait référence par le nom de cette variable. Pour nommer une variable ou une fonction, le développeur définit des identificateurs. Un identificateur doit commencer par une lettre (alphabet ASCII) ou le signe _ et se composer de lettres, de chiffres et des caractères _ et `$` (à l'exclusion du blanc).

Pour rappel JavaScript est sensible à la casse.

Les variables peuvent se déclarer de deux façons :

- soit de façon **explicite**. On dit à JavaScript que ceci est une variable. La commande qui permet de déclarer une variable est le mot `var`. Depuis ES6, il est également possible de remplacer `var` par `let` dont le comportement est similaire, sauf que let n'est pas reconnu par les versions anciennes des navigateurs.

Par exemple :

```
var numAdherent = 1 ;
var prenomAdherent = "Jean" ;
```

- soit de façon **implicite**. On écrit directement le nom de la variable suivi de la valeur que l'on lui attribue et JavaScript s'en accommode.

Par exemple :

```
numAdherent = 2;
prenomAdherent = "Luc";
```

Attention : malgré cette apparente facilité, la façon dont on déclare la variable aura une grande importance pour la « visibilité » de la variable dans le programme Javascript. Voir à ce sujet la distinction entre variable locale et variable globale plus loin dans ce cours.

Pour la clarté de votre script et votre facilité, on ne peut que conseiller d'utiliser à chaque fois le mot-clé `var` pour déclarer une variable.

## 2. Portée des variables

### Variables de portée globale

Les variables déclarées en début de script, en dehors et avant toute structure de code (fonction, condition etc.) fonctionnent comme des variables globales : on pourra donc les exploiter partout dans le script.

Exemple :

```
var myVar = 123;

maFonction();

function maFonction()
{
    console.log("myVar fonction : "+myVar);
}

if (myVar > 1)
{
       console.log("myVar condition : "+myVar);
}
console.log("myVar fin : "+myVar);
```

L'objet par défaut est l'objet `window`.

Par exemple, la déclaration implicite de la variable `maVariable` sous-entend qu'elle est une propriété de l'objet window. On pourrait l'appeler ainsi : `window.maVariable`.

*Exercice*

Testez l'exemple de code ci-dessus et observez ce qu'il se passe.

### Variables de portée locale

Toute variable déclarée à l'intérieur d'une structure de code ne sera valide que dans cette structure.

Exemple :

```
var compteur = 2;

function maFonction()
{
      var myVar = 456;
       console.log("myVar : "+myVar);
}

if (compteur > 1)
{
    let myVar2 = "Wazaa !";
    console.log("myVar2 : "+myVar2);
}

/* Ici, myVar n'est pas disponible
* car déclarée dans la fonction
* sa portée ne concerne que le code de la fonction
*/
console.log("myVar : "+myVar);

/* Ici, myVar2 n'est pas disponible
* car déclarée dans le bloc de condition
* avec une portée uniquement pour ce bloc
*/
console.log("myVar2 : "+myVar2);
```

*Exercice*

Testez l'exemple de code ci-dessus et observez ce qu'il se passe.

## 3. Les types de données

JavaScript utilise 5 types de données :

- Nombre : Tout nombre entier ou avec virgule tel que `22` ou `3.1416`. Tout entier en octal ou hexadécimal tel que `0387`, `0xFFA8`.
- Chaîne : toute suite de caractères comprise entre guillemets ou côtes telle que "suite de caractères" ou 'suite de caractères'
- Booléen : les mots `true` pour vrai et `false` pour faux
- Objet : toute utilisation de variable par référence vers tout objet natif JavaScript (Array, Date, String …) ou tout objet du DOM.
- `null` : mot réservé qui ne représente pas de valeur. La valeur null est affectée volontairement à une variable.
- `undefined` : mot réservé qui est renvoyé par une variable référencée qui n'a pas encore été définie (la variable existe mais n'a reçu aucune affectation de valeur).

Exemples :

```
var myVar; // le type de la variable myVar est undefined
myVar = 324;    // type number
myVar = "Bonjour";    // type string
myVar = true; // type boolean
myVar = new Array(); // type object
```

Notez que, contrairement au langage C ou C++, il ne faut pas déclarer le type de données d'une variable (pas besoin préciser `int`, `float`, `char` etc.).

*Connaître le type d'une donnée*

Une variable peut changer de type après une affectation. On peut vérifier le type en cours d'une variable avec l'instruction `typeof` :

```
console.log(typeof 42); // Affiche "number"
console.log(typeof 'blubber'); // Affiche "string"
console.log(typeof true); // Affiche "boolean"

var myVar;
console.log(typeof myVar); // Affiche "
```

# V. Afficher du texte

## 1. Les méthodes de l'objet document

JavaScript a la possibilité d'accéder aux différents éléments de la page HTML comme des objets. La partie du code HTML qui s'affiche dans la fenêtre du navigateur est un objet de type document (nous décrirons la notion d'objet dans un autre cours).

A chaque objet JavaScript, le concepteur du langage a prévu un ensemble de méthodes (ou fonctions dédiées à cet objet) qui lui sont propres. A l'objet document, JavaScript a dédié la méthode *écrire dans le document*, c'est la méthode `write()`.

L'appel de la méthode se fait selon la notation :

```
nom_de_l'objet.nom_de_la_méthode()
```

Pour appeler la méthode `write()` du document, on note :

```
document.write();
```

### La méthode `write()`

La syntaxe est assez simple soit :

```
document.write("votre âge");
```

On peut aussi écrire une variable, soit la variable `age` :

```
var age = 35;
document.write(age);
```

Pour associer du texte (chaînes de caractères) et des variables, on utilise l'instruction :

```
document.write("Votre âge est " + age);
```

On peut utiliser les balises HTML pour agrémenter ce texte :

```
document.write("<b>Votre âge est </b>" + age);
document.write("<b>" + " Votre âge est " + "</b>" + age);
```

## 2. Les boîtes de dialogue de l'objet `window`

Avec des méthodes de l'objet window, JavaScript a la possibilité d'appeler des boîtes de dialogue modales (bloquantes).

### La méthode `alert()`

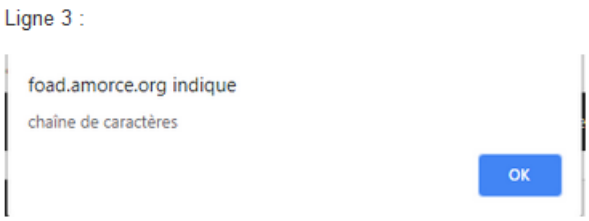
La méthode `alert()` affiche une boîte de dialogue dans laquelle est reproduite la valeur (variable et/ou chaîne de caractères) de l'argument qui lui a été fourni. Cette boîte bloque le programme en cours tant que l'utilisateur n'aura pas cliqué sur "OK".

La méthode alert() sera aussi très utile pour vous aider à débugger les scripts.

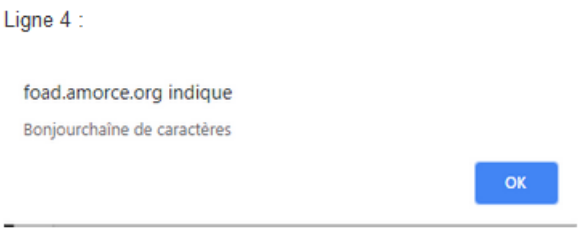
```
MyVar = "Bonjour";
window.alert(myVar);
window.alert("chaîne de caractères");
window.alert(myVar + "chaîne de caractères");
```

Ligne 2 :


Ligne 3 :




Ligne 4 :



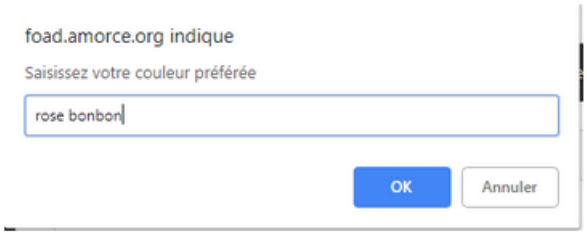

Pour écrire sur plusieurs lignes, utiliser un saut de ligne qui se fait avec le signe `\n`.

## La méthode `prompt()`

JavaScript vous propose une autre boîte de dialogue, dans le cas présent appelée boîte d'invite, qui est composée d'un champ comportant une entrée à compléter par l'utilisateur. Cette entrée possède aussi une valeur par défaut facultative.

```
var reponse1 = window.prompt("Saisissez votre nom");
var reponse2 = window.prompt("Saisissez votre couleur préférée");
```

Ligne 2 :



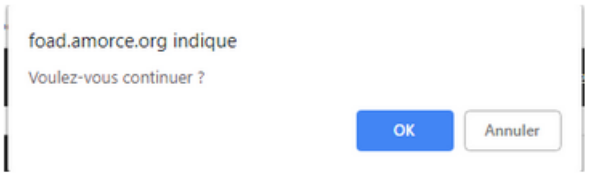

- En cliquant sur *OK*, la méthode renvoie la valeur tapée par l'utilisateur ou la réponse proposée par défaut
- Si l'utilisateur clique sur *Annuler*, la valeur `null` est alors renvoyée

## La méthode `confirm()`

Cette méthode affiche une boîte de dialogue avec 2 boutons *OK* et *Annuler*.

- En cliquant sur *OK*, la méthode renvoie la valeur `true`, ou `false` si on a cliqué sur *Annuler*. Ce qui peut permettre, par exemple, de choisir une option dans un programme.

```
if (window.confirm("Voulez-vous continuer ?") == true)
{
   …
}
```




## `alert()` … rouge

Joujou des débutants en JavaScript, cette petite fenêtre est à utiliser avec parcimonie pour attirer l'attention du lecteur pour des choses vraiment importantes.

```
alert("Vous avez confirmé !");
```

Précision : le code HTML n'est pas interprété dans les méthodes de l'objet `window`.

## 3. Afficher des informations dans la console

L'objet *console* est un élément essentiel lors de vos développements. Il permet d'afficher des informations de débogage (avertissements et erreurs de code, affichage de la valeur des variables etc.), précieuses pour la mise au point des scripts.

La console permet aussi de visualiser le code HTML qui n'est pas interprété dans les méthodes de l'objet `window`.

La console est un outil conçu uniquement pour les développeurs ; on ne doit pas y afficher d'informations pour dialoguer avec l'internaute (utiliser `alert()` ou `prompt()`).

### La méthode `log`

La console est incluse dans les outils de développement des navigateurs Chrome et Firefox, pour l'afficher faire touche `F12` puis cliquer sur l'onglet *console*.

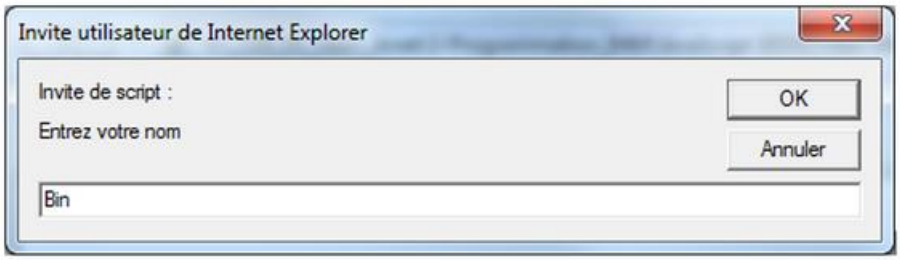
Ensuite, pour y afficher quelque chose, il faut utiliser la méthode `log` de permet d'afficher des informations dans la console.

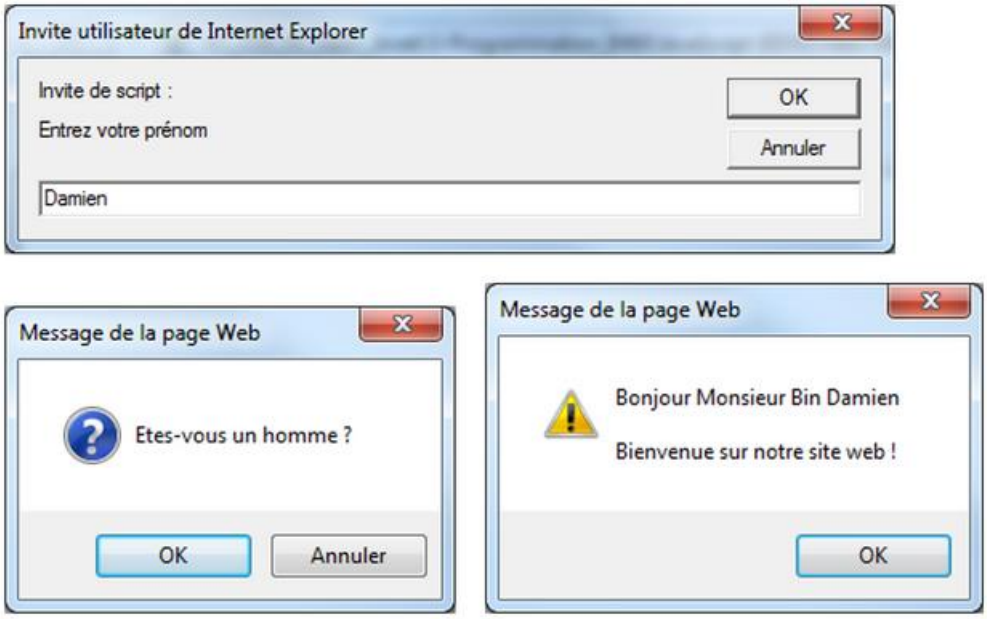
```
console.log("texte affiché dans la console");
```

## 4. Exercice

- Créer une page HTML qui demande successivement à l'utilisateur son nom puis son prénom.
- La page demande ensuite à l'utilisateur s'il est un homme ou une femme.
- Enfin, la page affiche les informations sur l'utilisateur.

Résultats à obtenir :






# VI. Opérateurs

Dans tous les exemples de ce cours, on considère que la valeur initiale de `x` est égale à *11*.

## 1. Les opérateurs de calcul

| Signe | Nom | Signification | Exemple | Résultat |
|---|---|---|---|---|
| + | Plus | Addition | x + 3 | 14 |
| - | Moins | Soustraction | x - 3 | 11 |
| * | Multiplié par | Multiplication | x*2 | 22 |
| / | Divisé par | Division | x/2 | 5.5 |
| % | Modulo | Reste de la division | x%5 | 1 |
| = | Egal | Reçoit la valeur de droite | x = 5 | 5 |

L'opérateur `+` sert aussi à concaténer des chaînes de caractères.

## 2. Les opérateurs de comparaison

| Signe | Nom | Exemple | Résultat |
|---|---|---|---|
| == | Egal | `x == 11` | `true` |
| < | Inférieur | `x < 11` | `false` |
| <= | Inférieur ou égal | `x <= 11` | `true` |
| > | Supérieur | `x > 11` | `false` |

| Signe | Nom | Exemple | Résultat |
|---|---|---|---|
| >= | Supérieur ou égal | `x > 11` | `true` |
| != | Différent | `x != 11` | `false` |

On confond souvent = avec ==. Le = est un opérateur de valeur tandis que le == est un opérateur de comparaison. Cette confusion est une source classique d'erreur de programmation.

## 3. Les opérateurs associatifs

On appelle ainsi les opérateurs qui réalisent un calcul dans lequel une variable intervient des deux côtés du signe = (ce sont donc en quelque sorte des opérateurs d'attribution).

Dans les exemples suivants `x` vaut toujours *11* et `y` aura comme valeur *5*.

| Signe | Description | Exemple | Signification | Résultat |
|---|---|---|---|---|
| `+=` | plus égal | `x += y` | `x = x + y` | 16 |
| `-=` | moins égal | `x -= y` | `x = x - y` | 6 |
| `*=` | multiplié égal | `x *= y` | `x = x * y` | 55 |
| `/=` | divisé égal | `x /= y` | `x = x / y` | 2.2 |

## 4. Les opérateurs logiques

Aussi appelés opérateurs booléens, ils servent à vérifier deux ou plusieurs conditions.

| Signe | Nom | Exemple | Signification |
|---|---|---|---|
| `&&` | et | (condition1) && (condition2) | condition1 et condition2 |
| \|\| | ou | (condition1) \|\| (condition2) | condition1 ou condition2 |

## 5. Les opérateurs d'incrémentation

Ces opérateurs vont augmenter ou diminuer la valeur de la variable d'une unité. Ce qui sera fort utile, par exemple, pour mettre en place des boucles.

Dans les exemples ci-dessous, `x` vaut 3.

| Signe | Description | Exemple | Signification | Résultat |
|---|---|---|---|---|
| `x++` | incrémentation (x++ est le même que x=x+1) | `y = x++` | 3 puis plus 1 | 4 |
| `x--` | décrémentation (x-- est le même que x=x-1) | `y = x--` | 3 puis moins 1 | 2 |

## 6. La priorité des opérateurs JavaScript

Les opérateurs s'effectuent dans l'ordre suivant de priorité (du degré de priorité le plus faible ou degré de priorité le plus élevé).

Dans le cas d'opérateurs de priorité égale, de gauche à droite.

| Opération | Opérateur |
|---|---|
| , | virgule ou séparateur de liste |
| = += -= *= /= %= | affectation |
| ? : | opérateur conditionnel |
| \|\| | ou logique |
| && | et logique |
| == != | égalité, différence |
| < <= >= > | relationnel |
| + - | addition, soustraction |
| * / | multiplication, soustraction |
| ! ++ -- | unaire |
| ( ) | parenthèses |