

# Javascript

Les bases

# Plan

- Introduction
- JavaScript est un langage côté client
- Où écrire le code JavaScript
- Syntaxe
- Les variables
- Les types
- Les opérateurs
- Les actions conditionnelle
- Les boucles
- Les boites de dialogues

# Introduction

- Le JavaScript est un langage de programmation créé en 1995. Le JavaScript est aujourd'hui l'un des langages de programmation les plus populaires et il fait partie des langages web dits « standards » avec le HTML et le CSS.
- Le JavaScript vient **combler les limites** du langage HTML
- Le JavaScript va nous permettre de **créer des pages interactives** et « vivantes » à l'aide de scripts.
  - Le JavaScript est un langage dynamique ;
  - Le JavaScript est un langage côté client ;
  - Le JavaScript est un langage interprété ;
  - Le JavaScript est un langage orienté objet.

# JavaScript est un langage dynamique

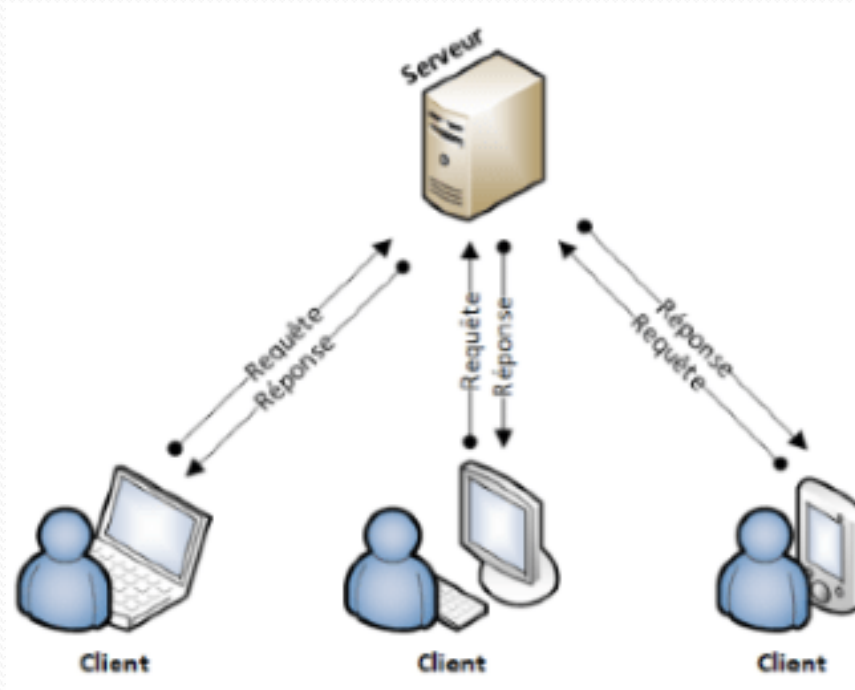
- JavaScript un langage qui va nous permettre de générer du contenu dynamique pour nos pages web.
- Un contenu « dynamique » est un contenu qui va se mettre à jour dynamiquement, c'est-à-dire changer sans qu'on ait besoin de modifier le code manuellement mais plutôt en fonction de différents facteurs externes.
- On oppose généralement les langages « dynamiques » aux langages « statiques » comme le HTML et le CSS.
- Une page statique est une page dont le contenu est le même pour tout le monde, à tout moment. En effet ni le HTML ni le CSS ne nous permettent de créer des contenus qui vont se mettre à jour par eux-mêmes.

# Langage côté client

- Il existe des langages **côté client** et des langages **côté serveur**.
- Un « **serveur** » est une sorte de super ordinateur, constamment accessible et qui va héberger les fichiers constituant un site web et le « servir » sur demande du **client**.
- Lorsqu'on demande à accéder à une page web en tapant une URL dans notre navigateur, nous sommes le client, et **navigateur est le logiciel client** qui effectue une demande au serveur,

# Langage côté client

- Un langage « côté client » ou « client side » est un langage qui va être exécuté dans le navigateur des utilisateurs qui demandent la page.



HTML  
CSS  
JAVASCRIPT

# Langage interprété

- Il existe des langages interprétés et des langages compilés.
- les langages compilés, doivent transformer le fichier source en une autre forme pour pouvoir l'exécuter.
- Les langages interprétés n'ont pas besoin d'une transformation pour s'exécuter.
- Le JavaScript est un langage interprété. Cela signifie qu'il va pouvoir être exécuté directement. L'exécution s'arrête à la première erreur rencontrée.

# Langage orienté objet

- JavaScript est un langage orienté objet. Nous parlerons de cela dans la partie consacrée aux objets.



# Framework Javascript

- un framework ou « cadre de travail » est relativement similaire dans son but à une « super librairie ».
- Les framework vont également nous fournir un ensemble de codes tout prêts pour nous faire gagner du temps en développement.
- Les framework JavaScript les plus connus aujourd'hui sont **Angular.js** et **React.js**.

# Les limites du HTML

Le langage HTML est limité, parmi ses limites on peut citer:

- **Absence des structures de contrôle algorithmiques**
- **Pas d'interaction avec l'utilisateur**
- **Aucune connectivité avec les serveurs de base de données**

# Où écrire du code JavaScript ?

On va pouvoir **placer du code JavaScript** à l'intérieur de la balise **<script>** et ceci à trois endroits différents :

- Dans l'élément **head** d'une page HTML
- Dans l'élément **body** d'une page HTML
- Dans un **fichier** portant l'extension « **.js** » séparé.

# JavaScript dans la balise Head

Dans ce cas, il faudra placer le JavaScript à l'intérieur d'un élément (balise) **script**.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Où écrire le JavaScript ?</title>
    <meta charset="utf-8">
    <script>
      alert('Ceci est affiché en JavaScript !');
    </script>
  </head>
  <body>
    <h1>On peut écrire le JavaScript dans...</h1>
    <ul>
      <li>L'élément head d'un fichier HTML</li>
      <li>L'élément body d'un fichier HTML</li>
      <li>Un fichier ".js" séparé</li>
    </ul>
  </body>
</html>
```

# JavaScript dans la balise Body

- On peut également écrire notre code JavaScript au sein de l'élément **body** d'un fichier HTML.
- Le code JavaScript est mit dans la balise **script** et préférons le placer à la fin de notre page.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Où écrire le JavaScript ?</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>On peut écrire le JavaScript dans...</h1>
    <ul>
      <li>L'élément head d'un fichier HTML</li>
      <li>L'élément body d'un fichier HTML</li>
      <li>Un fichier ".js" séparé</li>
    </ul>

    <script>
      alert('Ceci est affiché en JavaScript !');
    </script>
  </body>
</html>
```

# JavaScript dans un fichier .js (Avantages)

- C'est la méthode recommandée dans le cas de gros projets
- Permet la séparation des langages (meilleur maintenabilité)
- Possibilité de réutiliser un même code JavaScript dans plusieurs fichiers HTML.

# JavaScript dans un fichier .js (comment faire ?)

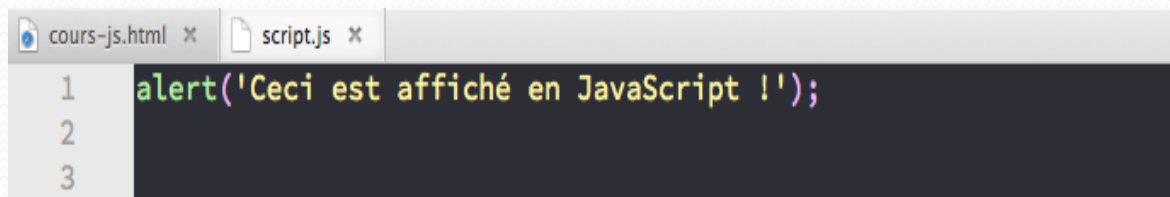
- Il faut lier nos fichiers HTML et JavaScript en utilisant à nouveau un élément **script** et son attribut **src**.
- Dans l'attribut **src**, nous allons indiquer le chemin relatif du **fichier .js** par rapport au fichier .html. Si nos deux fichiers sont dans le même dossier, par exemple, il suffira d'indiquer le nom du fichier JavaScript.

# JavaScript dans un fichier .js (Exemple)

- Fichier HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Où écrire le JavaScript ?</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>On peut écrire le JavaScript dans...</h1>
    <ul>
      <li>L'élément head d'un fichier HTML</li>
      <li>L'élément body d'un fichier HTML</li>
      <li>Un fichier ".js" séparé</li>
    </ul>
    <script src="script.js"></script>
  </body>
</html>
```

- Fichier .js :



```
1 alert('Ceci est affiché en JavaScript !');
2
3
```



# Syntaxe de JavaScript

- Chaque **instruction en JavaScript** se termine par un point virgule (;) mais ce n'est pas obligatoire.
- JavaScript est **sensible à la casse**. C'est-à-dire qu'il fait la différence entre majuscule et minuscule
- Les commentaires en JavaScript sont comme suit:

//une seule ligne

/\* plusieurs lignes \*/

# Les variables

- Il existe trois types de déclarations de variable en JavaScript.
  1. **var** : On déclare une variable, éventuellement en initialisant sa valeur.
  2. **let** : On déclare une variable dont la portée est celle du bloc courant, éventuellement en initialisant sa valeur.
  3. **Const** : On déclare une constante nommée, dont la portée est celle du bloc courant, accessible en lecture seule.

- Exemples:

```
var x=10;
```

```
var b=true;
```

```
var nom="Badaoui";
```

```
var prenom='Amal';
```

**Rq:** on peut utiliser les **guillemets** ou les **apostrophes** pour une chaînes de caractères.

# Les types des variables

- **Number** : ce type va représenter tout **nombre**, qu'il soit **positif**, **négatif**, **entier** ou à **virgule**.
  - **NaN**: (*Not A Number*) qui est obtenu lorsque l'on essaie de réaliser une opération interdite (comme par exemple diviser par zéro)
- **String**: ce type va représenter les chaînes de caractères. (les caractères d'échappement existent aussi dans javascript : \n,\t,\r...)
- **Boolean**: ce type représente une valeur **true** ou **false**.
- **Object** : Il s'agit d'un moyen d'utiliser des objets en JavaScript (à voir plus tard)
- **null** : est un dernier type possible. Il signifie qu'une variable ne contient pas de donnée.

# Conversion de type(1)

JavaScript permet de changer le type d'une variable, avec les fonctions suivantes :

- **parseInt()**: permet de convertir une chaîne en entier
- **parseFloat()** : permet de convertir une chaîne en flottant .
- **Number()**: permet de convertir une chaîne en nombre.
- **toString()**: permet de convertir un nombre en chaîne de caractères.
- **String()**: permet de convertir un nombre en chaîne de caractères.
- On peut connaître le type d'une variable avec la fonction **typeof()**

# Conversion de type(2)

- Exemple1:

```
var chaine="43.54";  
var nb1=parseInt(chaine); //nb1=43  
var nb2=parseFloat(chaine); //nb2=43.54  
var nb3=Number(chaine); //nb3=43.54
```

- Exemple 2:

```
var nb=43.54;  
var chaine=nb+""; //chaine="43.54";  
var chaine=nb.toString(); //chaine="43.54";
```

# Tests sur les types

Il est parfois nécessaire, avant de lancer un traitement sur des variables supposées être des nombres, de tester si tel est vraiment le cas.

- **isNaN()** : très utilisée, teste si le paramètre *n'est pas* un nombre.

- Exemple :

**isNaN(34.7)**                      //renvoie false

**isNaN("abc")**                    //renvoie true.

# Les opérateurs

- Les opérateurs **algébriques** :  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $\%$
- Les opérateurs **d'affectation** :  $=$ ,  $+=$ ,  $-=$ ,  $*=$ ,  $/=$ ,  $\%=$
- Les opérateurs de **comparaison**:  $==$ ,  $===$ ,  $>=$ ,  $>$ ,  $<$ ,  $<=$ ,  $!=$ ,  $!==$
- Les opérateur **d'incrément et décrémentation**:  $x++$ ,  $x--$ ,  $++x$ ,  $--x$
- Les opérateurs **logiques**:
  - **&&**: représente le **ET** logique
  - **||**: représente le **OU** logique
  - **!**: représente la **négation**

# Afficher un texte dans le document

- pour afficher un texte on va utiliser pour le moment la méthode **document.write ()** comme suit:

```
document.write("chaine") ;
```

```
document.write("Bonjour") ;  
Document.write("<p>Je suis un paragrpah</p>")
```



# Les actions conditionnelles(1)

- **Syntaxe 1:**

`if(condition) { action }`

- **Syntaxe 2:**

`if(condition){Action1} else{Action2}`

- **Syntaxe 3:**

`if(condition1){Action1}  
else if(condition2){Action2}  
else if(constion 3){Action3}  
.....  
else{action}`

# Les actions conditionnelles(2)

## Choix multiple

```
switch ( variable ){  
  
    case val1 : action1 ; break;  
    case val2 : action2; break;  
    case val3 : action3; break;  
  
    ...  
    default : action; break;  
}
```

# Les actions conditionnelles(3)

## (Opérateur ternaire)

Opérateur ternaire est un moyen d'exécuter une condition simple .  
Equivalent à : **if...else**

**(expression)? Valeur si vrai : Valeur si faux ;**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les ternaires</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Les ternaires</h1>
    <script>
      var heure = 19, bon="";

      bon = (heure <= 18) ? "Bonjour" : "Bonsoir";
      alert(bon);
    </script>
  </body>
</html>
```

# Les boucles

Les boucles dans le langage javascript sont :

- La boucle **while**
- La boucle **do... while**
- La boucle **for**

# La boucle while

- La boucle **while** est l'équivalente de la boucle Tant que vue en algorithmique.
- Syntaxe:

```
while ( condition ) {  
  
Instructions  
  
}
```

- Exemple:

```
var x=5 ;  
while ( x != 0 ) {  
  
document.write(x) ;  
x--  
}
```

# La boucle do...while

- La boucle **do...while** est l'équivalente de la boucle **Faire....Tant que** vue en algorithmique.
- Syntaxe:

```
do{  
  
Instructions  
  
} while( condition ) ;
```

- Exemple:

```
var x=5 ;  
do{  
x=x+1  
document.write(x) ;  
  
} while(x <=10) ;
```

# La boucle for

- La boucle **for** est l'équivalente de la boucle **pour** en algorithmique:
- Syntaxe:

```
for (initialisation ; condition ; incrémentation) {  
  
    Les instructions  
  
}
```

- Où:
- **Initialisation** : est généralement une initialisation d'un compteur de la boucle
- **Condition**: est la condition d'arrêt de la boucle
- **Incrémentation** : est l'incrément ou la décrémentation du compteur de la boucle

# La boucle for

- Exemple :

```
for (i=1 ; i <=10 ; i++ ) {  
    Console.write(i) ;  
}
```



# Comparaison entre la boucle for et la boucle while

- Les deux boucles **for** et **while** sont équivalentes:

```
for (i=1 ; i <=10 ; i++ ) {  
  
  Console.write(i) ;  
  
}
```

```
var i=1 ;  
while (i<=10) {  
  
  Console.write(i) ;  
  i++;  
  
}
```

# Les boites de dialogue (1)

## (alert)

- Cette fonction est très importante, elle permet d'afficher une boite de dialogue contenant un message.

Syntaxe:

- `alert ("message");` //affiche le message indiqué entre ()
- `alert (x);` //affiche la valeur de la variable x
- `alert ("la valeur vaut " + x);` //message suivi d'une valeur

# Les boites de dialogue (2)

## prompt

- **prompt()**: c'est une fonction qui affiche une boite de dialogue qui demande à l'utilisateur de saisir une valeur:

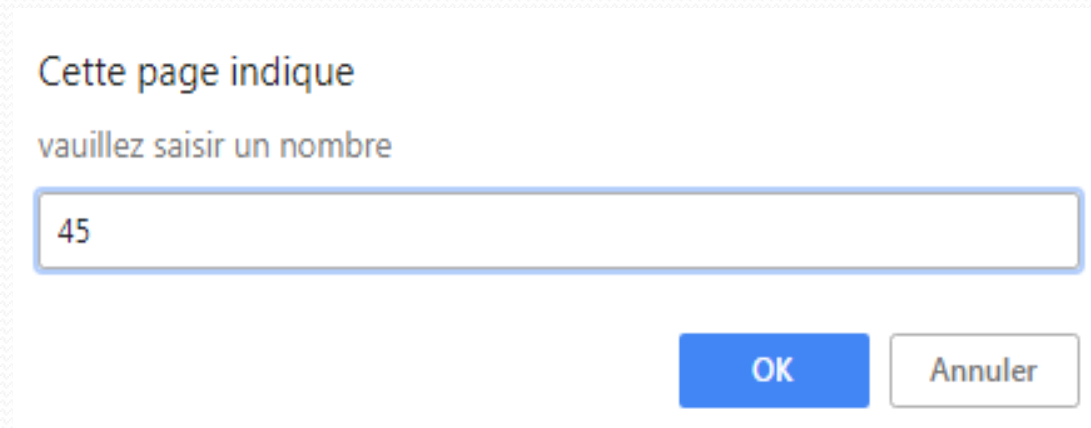
Exemple:

```
var x=prompt("veuillez saisir un nombre");  
if(x!=null){  
  alert("le nombre que vous avez entré est :"+x);  
}
```

# Les boites de dialogue (3)

## prompt

- Le résultat du code précédent est le suivant :



A screenshot of a JavaScript prompt dialog box. The dialog has a white background and a thin grey border. Inside, the text "Cette page indique" is in a dark blue font, and "veuillez saisir un nombre" is in a lighter blue font. Below the text is a text input field with a blue border, containing the number "45". At the bottom right of the dialog are two buttons: a blue "OK" button and a white "Annuler" button with a grey border.

Cette page indique  
veuillez saisir un nombre

45

OK Annuler

# Les boites de dialogue (4)

## (confirm)

- **confirm()**: c'est comme une boite **alert()**, sauf qu'elle ajoute des options de réponse; qui peuvent donner à l'utilisateur la possibilité de confirmer ou d'annuler ce qui est demandé.

### Exemple:

```
var x=confirm("Etes vous sûre de vouloir...");
```

```
if(x==true){  
    alert("ok");  
}  
else{  
    alert("non");  
}
```

# Les boites de dialogue (5) (confirm)

- Le résultat du code précédent est :

