

# Javascript : Les bases

Sébastien Besnier - CSM

1.

Où écrire le Javascript

# Où dois-je écrire mon code ?


Dans l'élément head de la page HTML

Et/Ou dans l'élément body de la page HTML

Et/Ou un fichier javascript (extension .js)



```
<!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>
```



```
<!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>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Où écrire le JavaScript ?</title>
    <meta charset="utf-8">
    <script>
      alert('Script n°1');
    </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>

    <script>
      alert('Script n°2');
    </script>
  </body>
</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>
```

cours-js.html x script.js x

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

```
<!DOCTYPE html>
<html>
<body>

<h2>Mon premier javascript</h2>

<h3 id="demo"></h3>
<p id="name"></p>
<script>
document.getElementById("demo").innerHTML = "Hello Javascript";
document.getElementById("name").innerHTML = "My Name " + "is Sébastien";
</script>

</body>
</html>
```

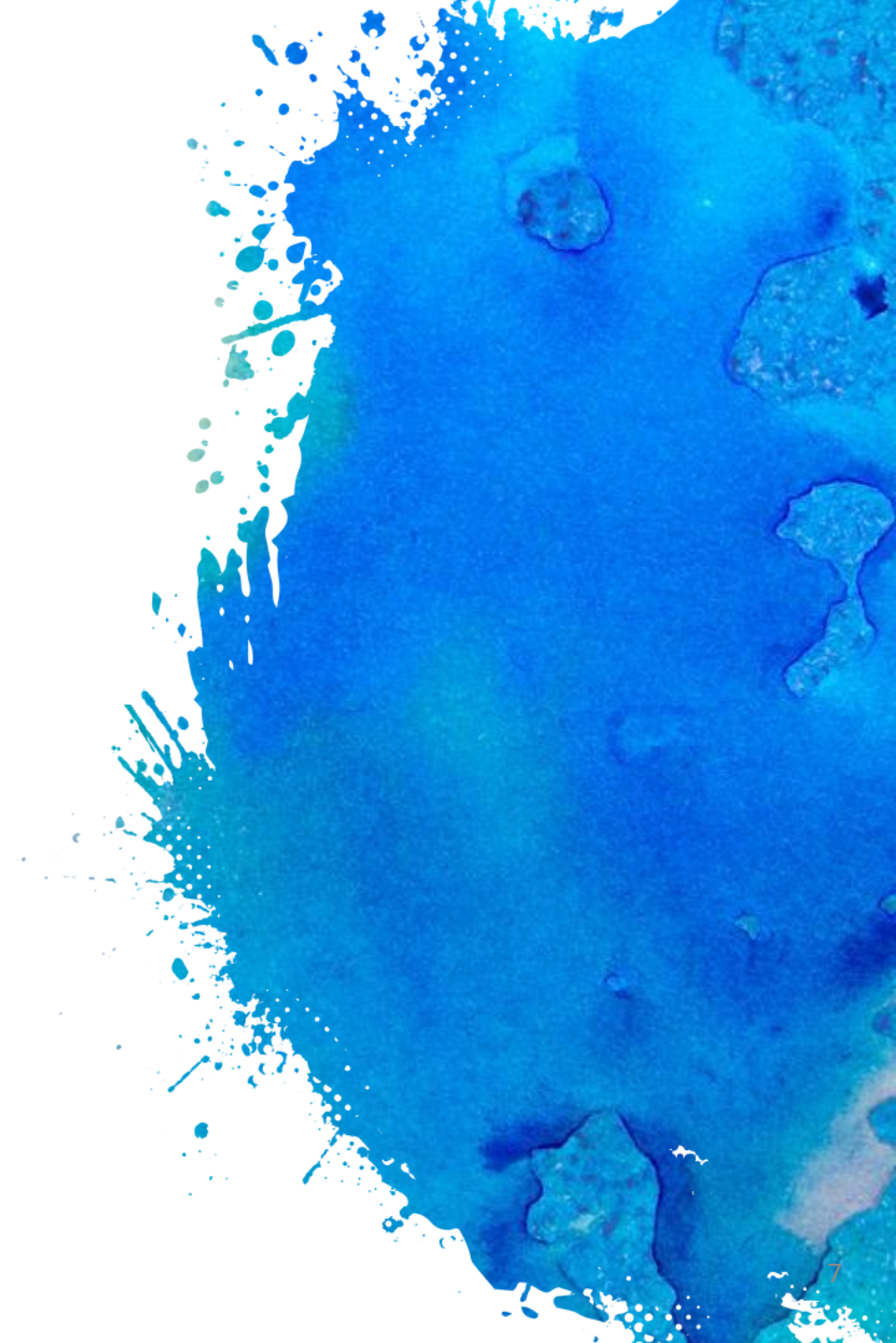
## Mon premier javascript

Hello Javascript

My Name is Sébastien

2.

# Les variables et opérateurs



# Les variables

- Les variables permettent de stocker des valeurs
- Le nom peut être court (x, y, ...) ou long (maValeur, total, ...)
- Les règles:
  - Les noms peuvent contenir des lettres, chiffres, underscore, dollar
  - Le nom doit commencer par une lettre ou underscore ou dollar
  - Les nom sont sensibles à la casse (maValeur est différent de Mavaleur)
  - Les mots réservés de javascript ne peuvent pas servir de nom (ex: function, var, ...)
- L'opérateur = permet d'assigner une valeur à une variable



```
<!DOCTYPE html>
<html>
<body>

<h2>Les variable en JavaScript</h2>

<p>You can declare many variables in one statement.</p>

<p id="demo"></p>

<script>
var maVariable = "une valeur";

var quantité = 1, animal = "vache", prix = 2000 + 500, couleur = "beige";

document.getElementById("demo").innerHTML = quantité + " " + animal + " => " + prix + "€";
</script>

</body>
</html>
```

# Les opérateurs arithmétiques

Opérateur	Opération
+	Addition
-	Soustraction
*	Multiplication
/	Division
**	Exponentiel
%	Reste de la division
++	Incrément
--	Décrément

# Les opérateurs d'assignation

Opérateur	Exemple	Equivalent
=	x = y	N/A
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	X = x ** y

# Les opérateurs de comparaison

Opérateur	Exemple	
==	x == y	x est égal à y
===	x === y	x est égal à y en valeur et en type
!=	x != y	x est différent de y
!==	X !==	x est différent de y en valeur ou en type
>	x > y	x est supérieur strictement à y
<	x < y	x est inférieur strictement à y
>=	x >= y	x est supérieur ou égal à y
<=	x <= y	x est inférieur ou égal à y
? (opérateur ternaire)	var z = (age > 50) ? « vieux » : « jeune »;	Si age > 50 alors « vieux » sinon « jeune »



# Les tableaux

- Un tableau est utilisé pour stocker plusieurs valeurs dans la même variable
- `var monTableau = [elem1, elem2, elem3, ...];`
- Le premier élément du tableau se trouve à la position 0
- Le dernier élément du tableau se trouve à la position `[taille du tableau] - 1`
- Pour connaître la taille d'un tableau on utilise la propriété `length`
  - `var taille = monTableau.length`

# Les tableaux

- Pour supprimer le dernier élément, on utilise la méthode `pop()`
- Pour supprimer le premier élément, on utilise la méthode `shift()`
- Pour ajouter un élément en fin de tableau, on utilise `push()`
- Pour ajouter un élément en début de tableau, on utilise `unshift()`
- Pour ajouter/supprimer un élément en milieu de tableau on utilise `splice()`
  - `monTableau.splice (a, b, elem1, elem2, ...)`;
  - `a` est la position ou le nouvel element doit être ajouté
  - `b` est la position ou l'élément est supprimé
  - `elem1, elem2`, sont les éléments à ajouter

# Le type Object

- Les objets sont des variables qui contiennent plusieurs valeurs

Exemples:

```
var user = { type: « admin », email: « admin@gmail.com, age: 28 };  
var produit = {  
    name: « short »,  
    price: 18,  
    size: [12, 14, 16, 18]  
};
```

# Le type Object

- On peut ensuite accéder aux propriétés de l'objet de deux manières:

`objectName.propertyName`

`objectName[« propertyName »]`

- Exemple:

```
console.log (produit.name + «  prix =>  » + produit[price]);
```



# Un tableau peut contenir des objets

```
<!DOCTYPE html>
<html>
<body>

<h2>Les objets et les tableaux</h2>

<p id="demo"></p>
<p id="liste"></p>

<script>
// Création des objets:
var voiture1 = {type:"Citroen", model:"C4", color:"white"};
var voiture2 = {type:"Peugeot", model:"308", color:"black"};

var tableau = [voiture1, voiture2];

// Affichage d'informations sur le tableau et les objets:
document.getElementById("demo").innerHTML = "Nombre de voitures : " + tableau.length;

var result = "";

for (var i = 0; i < tableau.length; i++){
    result += "Type : " + tableau[i].type + " => Model: " + tableau[i].model + "<br>";
}

document.getElementById("liste").innerHTML = result;
</script>

</body>
</html>
```

# 3. Les conditions

# if ... then...else

- if est utilisé pour exécuter un bloc de code dans le cas où une condition est vraie
- else est utilisé pour exécuter un bloc de code dans le cas où la même condition n'est pas vraie
- else if est utilisé pour spécifier une nouvelle condition si la première condition est fausse
- switch est utilisé pour spécifier plusieurs blocs de code alternatifs

```
if (condition est vraie) {  
    // bloc de code à exécuter  
}
```

```
if (condition est vraie) {  
    // bloc de code à exécuter  
} else {  
    // bloc de code à exécuter si la condition est fausse  
}
```

```
if (condition1 est vraie) {  
    // bloc de code à exécuter  
} else if (condition2 est vraie) {  
    // bloc de code à exécuter si condition1 est fausse et condition2 est vraie  
} else {  
    // bloc de code à exécuter si la condition1 et condition2 sont fausses  
}
```



# switch...case

```
switch(expression) {  
  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    case a:  
    case b:  
        // code block  
        break;  
    default:  
        // code block  
}
```

# 4. Les boucles

# La boucle for

```
for (declaration1; declaration2; declaration3){  
  // exécution du code en boucle  
}
```

declaration1 est exécutée une seule fois

declaration2 décrit la condition d'exécution

declaration3 est exécutée à chaque fois, une fois le bloc de code exécuté

// Attention aux boucles infinies...

```
for (var i = 0; i = 10; i += 6){  
  //tourne indéfiniment  
}
```

# La boucle while

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

Tant que (condition est vraie), on exécute le bloc de code

```
var i = 0;  
while (i < 10){  
  // du code  
  i++;  
}
```



# La boucle do / while

- C'est une variante de la boucle while. Elle est exécutée au moins une fois

```
do {
```

```
// bloc de code à exécuter
```

```
}
```

```
while (condition est vraie);
```

# Exercice

- Créer un tableau tab avec une quinzaine de valeurs (des chiffres)
- Ajouter 100 valeurs prises au hasard (entre 10 et 100) en utilisant la fonction `Math.random()`
- Parcourir le tableau pour trouver la valeur Max
- Parcourir le tableau pour trouver la valeur Min
- Calculer la somme des valeurs
- Calculer la moyenne des valeurs du tableau
- Si la valeur est  $\leq 10$ , placer cette valeur dans un tableau tab1
- Si la valeur est  $> 10$ , placer cette valeur dans un autre tab2
- Trier les tableaux tab1 et tab2
- Concaténer les deux tableaux tab1 et tab2 dans un tableau tab3 trié (croissant)
- Supprimer toutes les valeurs de tab en utilisant une boucle for
- Supprimer toutes les valeurs de tab1 en utilisant une boucle while
- Supprimer toutes les valeurs de tab2 en utilisant une boucle do...while

# Exercice

- La machine génère un entier entre 0 et 1000 appelé X
- La machine vous demande votre nom (Utiliser la commande prompt)
- Vous devez retrouver le nombre généré par la machine
  - La machine vous demande de saisir un entier
  - En fonction de la valeur saisie, la machine vous indique si votre chiffre est largement supérieur ou largement inférieur (la différence entre X et votre saisie est supérieure à 100), si votre chiffre est supérieur ou inférieur (la différence entre X et votre saisie est inférieure à 100) ou si vous avez trouvé le bon résultat.
  - Si vous saisissez le mot « triche » à la place d'un chiffre, la machine vous montre X.
  - Lorsque X est trouvé, la machine vous indique le nombre de tentatives que vous avez effectuées pour trouver ce chiffre

# 5. Les fonctions

# Une fonction... pourquoi?

- Une fonction est un bloc de code qui fait une tâche bien spécifique
- Une fonction est exécutée quand quelque chose l'invoque:
  - Un appel directement à cette fonction dans du code javascript
  - Quand un événement se passe dans une page HTML (ex: On clique sur un bouton)
  - Une fonction peut être appelée par elle-même
- Une fonction possède un nom, des paramètres optionnels, et peut retourner une valeur

# functions

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>

// Fonction sans paramètre
function functionWithoutParameter() {
    console.log(3+6);
}

// Fonction avec paramètres
function sum(param1, param2, param3, param4){
    console.log(param1+param2+param3+param4);
}

// Fonction avec paramètres et valeur de retour
function affichesum(param1, param2, param3, param4){
    return (param1+param2+param3+param4);
}

// Je stock le résultat de ma fonction dans une variable
var maSomme = affichesum(4, 3, 2, 1);

document.getElementById("demo").innerHTML = "Le résultat est " + maSomme;
</script>

</body>
</html>
```

# 6. Ateliers



# Puissance 4

« Alignez 4 pions de la même couleur, et c'est gagné !!! »

- Le joueur joue contre la machine
- Le programme demande le nom du joueur
- A tour de rôle, le joueur et la machine choisissent un numéro de colonne dans laquelle ils peuvent jouer (S'il n'est pas possible de jouer dans une colonne, le programme doit redemander un numéro de colonne)
- A chaque tour, le programme affiche la grille (10 x 10) du puissance 4 afin de voir où on peut jouer.
- La partie s'arrête lorsque le joueur ou la machine a aligné verticalement, horizontalement ou en diagonal 4 pions de la même couleur, ou s'il n'est plus possible de jouer
- Bonne chance et que le meilleur gagne !!!

# Il a coulé mon porte-avion !!!!

- Deux joueurs (vous et la machine)
- Chaque joueur possède un plateau de 10 colonnes (0 à 9) et 10 lignes (A à J).  
Chaque joueur possède un porte-avion (5 cases), un cuirassier (4 cases), un destroyer et un sous-marin (3 cases), un patrouilleur (2 cases).
- Le programme place aléatoirement la flotte sur chaque plateau (pas de diagonale)
- Le programme désigne aléatoirement qui démarre le jeu
- Pour désigner un emplacement, on commence toujours par désigner la référence de la ligne. Ex: « A4 », « J0 », ...
- Si un emplacement a déjà été désigné, le programme l'indique et on rejoue.
- Si à la place de désigner une case, on tape le mot « triche », le programme montre le plateau de l'ordinateur.
- A l'attaque!!!!