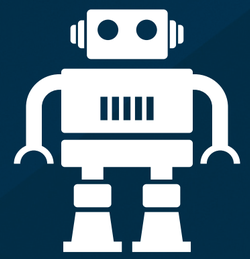


Communiquer avec un robot

JAVASCRIPT, LES FONDAMENTAUX

JAVASCRIPT, LES FONDAMENTAUX

COMMUNIQUER AVEC UN ROBOT

ECRIRE DU CODE EN JAVASCRIPT

Le Javascript est un language de programation orienté objet, c'est à dire qu'il permet de créer des boîtes virtuelles qui contiennent différentes caractéristiques. Ces boîtes et leurs caractéristiques sont ensuite utilisées pour construire des programmes, comme des Légos. La particularité de Javascript est d'être activé une fois que le rendu de la page est terminée, c'est pouquoi le code JavaScript doit être écrit à la fin du document.

BALISE <script>...</script>
La propriété type="..." permet de spécifier la nature du script

```
<script type="text/javascript">
    // Ouverture d'une alert en JavaScript
    alert("Hello World!");

</script>
```

LES COMMENTAIRES
Il est primordiale de commenter le code JavaScript

CODE JAVASCRIPT
La commande alert("...") affiche une boîte de dialogue en JavaScript

INTEGRER UN FICHIER .JS EXTERNE

Comme avec le code CSS, il est possible d'intégrer du JavaScript de différentes façons dans une page web. Pour améliorer les perfromances d'affichage de votre page, il est recommandé de placer le JavaScript dans un fichier .js extérieur au document principal (comme pour les fichiers .css). Les fichiers JavaScript sont ensuite intégrés dans une balise <script>...</script> juste avant la balise </body>.

```
<!DOCTYPE html>
<html>

  <head>
    <meta charset="utf-8" />
    <title>Bonjour le monde</title>
    <link rel="stylesheet" type="text/css" href="css/style.css">
  </head>

  <body>
    <p class="textBjr">Hello World!</p>
    <script type="text/javascript" src="js/script.js"></script>
  </body>

</html>
```

FICHIER CSS

Les styles sont intégrés dans la balise **<head>...</head>**

FICHIER JS EXTERNE

La propriété **src="..."** permet de **cibler un fichier .js** commme la propriété href="..." pour le .css

L'importation du fichier .js ce fait avant le balise </body> car il doit sexécuter une fois la page chargée

UTILISER LE MEMOIRE DE L'ORDINATEUR CLIENT

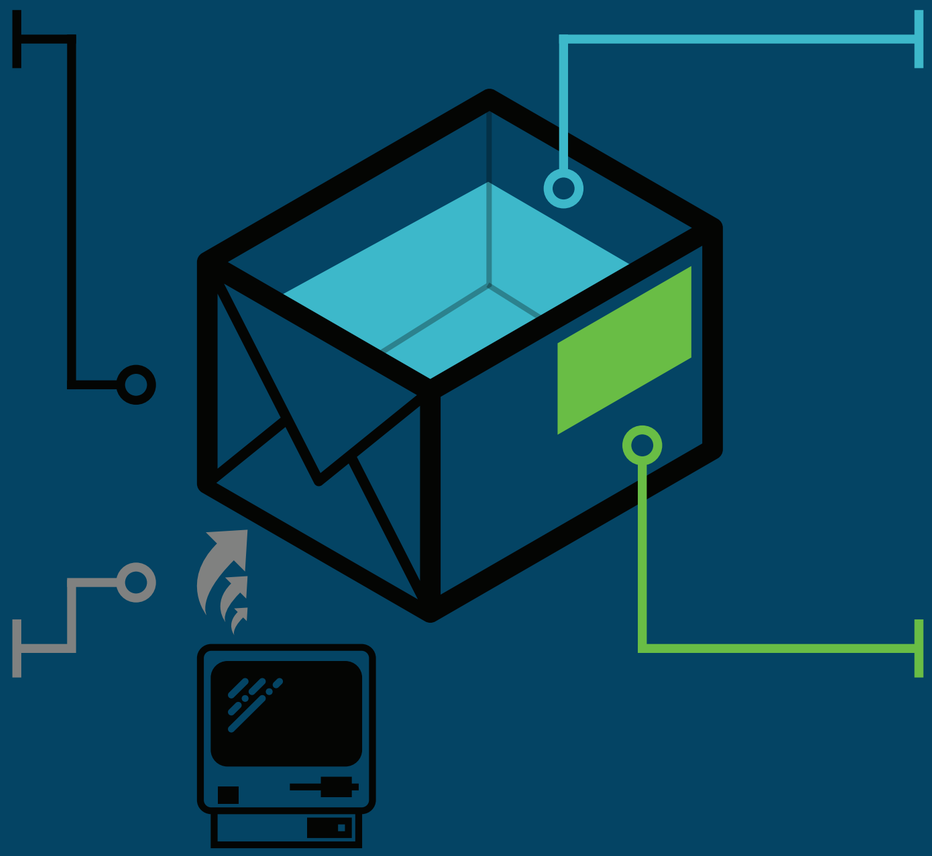
Comme vu précédemment, le JavaScript utilise le principe des boîtes, que nous appelerons des variables, pour y associer des caractéristiques. Ces variables sont stokées dans la mémoire de l'ordinateur client (l'internaute), il est alors possible de les réutiliser, de les associer à des actions et bien plus encore. Les variables sont composées de deux éléments clef : le nom de la variable et le contenu de la variable.

LA VARIABLE JAVASCRIPT

Les variable correspondent à espace mémoire de l'ordinateur du client. Pour initialiser une variable, il faut utiliser le mot-clef "var" suivi du nom de la variable.

var nomDeLaVariable;

Les variable peuvent être de différents types et doivent respecter une orthographe stricte.



L'ESPACE MEMOIRE

Les variables sont stockées dans la mémoire de l'odinateur client. Elles sont ensuite réutilisables en écrivant le nom de la variable dans le code.

nomDeLaVariable;

LE CONTENU DE LA VARIABLE

Il existe trois principaux types de contenu pour les variables JavaScript : strings, number et boolean.

var **unString** = **"Du texte"**;
var **unNumber** = **246**;
var **unBoolean** = **false**;

Le symbole égale ("=") permet de définir le contenu d'une variable.

LE NOM DE LA VARIABLE

Le nom de la variable permet de l'identifier dans le programme. Par convention, le nom s'écrit en camelCase.

var **nomDeLaVariable**;

STOCKER UNE VARIABLE DANS L'ESPACE MEMOIRE

Le Javascript s'exécutant une fois la page entièrement chargée, nous pouvons enregistrer les actions de l'utilisateur pour déclencher des scripts qui répondront à ces actions. Les scripts utiliserons donc la mémoire de l'ordinateur client pour enregistrer ces réponses et agir en conséquence.

DECLARATION DE LA VARIABLE

La variable "leNom" est stockée en mémoire

```
// Création d'une variable qui stock une réponse
var leNom = prompt("Quel est ton nom ?");

// La réponse est affichée dans la console
console.log("Bonjour " + leNom);
```

COMMANDE PROMPT("...")

Cette commande affiche un boîte de dialogue avec un champs de texte. Le contenu de ce champs sera la valeur de "leNom".

REPONSE JAVASCRIPT

La variable "leNom" est affichée dans une nouvelle boîte de dialogue.

LES VARIABLES PRIMITIVES DE JAVASCRIPT

Pour réaliser un code, JavaScript dispose de trois types de variables de base (dit type primitif) : String, Number et Boolean.

STRING

Variable de type texte ayant pour valeur une **chaîne de caractères**.

La valeur est écrite **entre guillemets**.

```
var string = “Julien Noyer“;
```

NUMBER

Variable de type numérique ayant pour valeur **un ou plusieurs chiffres**.

La valeur est écrite **sans guillemets**.

```
var number = 1638;
```

BOOLEAN

Variable représentant une **réponse vraie ou fausse**.

Les valeurs sont **true ou false**.

```
var boolean = true;
```

LES OPERATEURS ELEMENTAIRES EN JAVASCRIPT

JavaScript possède différents types d'opérateurs qui permettent de réaliser des opérations plus ou moins complèxes avec des variables. Le premier opérateur que nous avons utilisé et l'opérateur d'affectation qui permet de définir le contenu d'une variable et l'opérateur d'addition qui permet de concatèner des variables ou de les additionner.

```
var number = 10;                                    // La variable number vaut 10
number++;                                           // La variable number vaut maintenant 11
number--;                                           // La variable number vaut maintenant 10
var addition = 5 + 5;                               // La variable addition vaut 10
var soustraction = 15 - 5;                          // La variable soustraction vaut 10
var multiplication = 5 * 5;                         // La variable multiplication vaut 25
var division = 10 / 5;                              // La variable division vaut 2

var modulo = 23 % 5;                                // La variable modulo vaut 3 (reste d'une division)

var x = 10;
var y = 15;
x += y;                                             // La variable x vaut maintenant 25 (x + y)
y -= x;                                             // La variable y vaut maintenant -10 (y - x)

var firstName = "Julien";
var lastName = "Noyer";
var fullName = firstName + " " + lastName; // La variable fullName vaut "Julien Noyer"
```

EFFECTUER UN CALCUL SIMPLE EN JAVASCRIPT

En effectuant des opérations simples, il est possible d'afficher un message personnalisé à l'utilisateur qui prendra en compte les paramêtres qu'il aura renseigné dans la fenêtre prompt().

```
var currentYear = 2015;

// Trois questions sont posées à l'utilisateur
var firstName = prompt("Quel est ton prénom ?");
var lastName = prompt("Quel est ton nom ?");
var age = prompt("Quel est ton âge ?");

// Calcul de la date de naissance de l'utilisateur
var birthYear = currentYear - parseInt(age);

console.log("Bonjour " + firstName + " " + lastName + ", ton année de naissance est : " + birthYear);
```

PARSEINT(...)

La commande parseInt(...) permet de transformer une chaîne de caractère en chiffre pour pouvoir effectuer un calcul avec le retour du prompt().

ECRIRE UNE FONCTION JAVASCRIPT

Les fonctions font parties des briques fondamentales de JavaScript, une fonction est un ensemble d'instructions JavaScript effectuant une ou plusieurs tâches. Pour utiliser une fonction, il faut d'abord définir ces instructions avant de l'appeler avec ou sans paramêtre. Les fonctions sont très utiles pour réaliser des instructions à plusieurs reprise sans avoir à les écrire plusieurs fois.

SANS PARAMETRE

Le mot clés function permet de déclarer une fonction en Javascript, suivi du nom() de la fonction

Une fois défiie la fonction doit être appelée

```
function helloWorld() {
  return “Hello World!“;
}

helloWorld();
// Affiche la string “Hello World!”
```

AVEC PARAMETRES

Les paramètres de la fonction sont des variables placées entre les parenthèses et séparées par une virgule

Les paramètres sont ajoutés à l'appel de la fonction

```
function fullName(firstName, lastName) {
  return “Hello World “ + firstName + “ “ + lastName;
}

fullName(“Julien”, “Noyer”);
// Affiche la string “Hello World Julien Noyer”
```

Le mot clé return permet de renvoyer les donnés récupérées par la fonction

LES TABLEAUX EN JAVASCRIPT

Les tableaux (array en anglais) sont des variables qui ont pour particularité de pouvoir recevoir plusieurs valeurs et de différents types. Pour reprendre le principe des boîtes, une variable de type array serait un conteneur pour plusieurs boîtes. Les tableaux sont très utiles car ils permettent d'inclure dans une seul variable plusieurs entrées contenant elles-même des variables.

VARIABLE REUTILISABLE

```
var leNom = "Julien Noyer";
```

DECLARATION DU TABLEAU

Les entrées du tableau sont écrites entre crochets et séparées par une virgule.

```
var monTableau = ["Du texte", 2391, false, leNom];
```

MANIPULER LE TABLEAU

En language de programmation, la numérotation d'un tableau commence par la valeur zéro. Pour afficher une valeur il faut mettre entre crochets le numéro de la valeur.

```
// Afficher les éléments du tableau
console.log(monTableau[0]);          // Affiche "Du texte"
console.log(monTableau[1]);          // Affiche 2391
console.log(monTableau[2]);          // Affiche false
console.log(monTableau[3]);          // Affiche "Julien Noyer"
```

.LENGTH

Cette commande renvoi le nombre d'entrées du tableau en une valeur de type number.

```
// Connaître la taille d'un tableau
console.log(monTableau.length);      // Affiche 4
```

MANIPULER LES DONNEES D'UN TABLEAU

Comme pour les autres variables, le contenu d'un tableau peut être modifié après sa déclaration. Pour ce faire, il faut utiliser la commande .push(...) pour ajouter une nouvelle entrée au tableau. Il est également possible de remplacer une entrée dans le tableau en utilisant la commande .splice(...). La gestion des tableaux reprensente un principe important dans la programmation orientée objet.

.PUSH(...)

La commande .push(...) permet d'ajouter une entrée dans le tableau.

```
var users = ["Frédérique", "Pascal", "Matthieu", "Julien"];
var newUser = prompt("Quel est ton prénom ?");

users.push(newUser);
console.log(users);

var removed = users.splice(1, 2, "Jacques", "Paul");
console.log(removed);
console.log(users);
```

.SPLICE(...)

La commande .splice(...) de supprimer une entrée depuis l'index d'un tableau et de la remplacer par une autre.

LES OBJETS EN JAVASCRIPT

Les objets JavaScript sont les variables les plus puissantes, ils combinent un ensemble de propriétés, comprenant noms et valeurs associées, et peut également contenir des fonctions entières. La structure dun objet est assez proche de celle d'un tableau, l'avantage d'un objet est qu'il permet de labelliser les valeurs qu'il contient.

DECLARATION DE L'OBJET

Les paramètres de l'objet sont écrits entre accolades et séparées par une virgule

AFFICHER LES PARAMÈTRES

L'appel des paramètres ce fait avec un point et deux parenthèses après le nom de l'objet

MODIFIER UN PARAMÈTRE

Pour modifier un paramètre il suffi de l'appeler pour lui affecter une nouvelle valeur

```
var julien = {
  firstName: "Julien",
  lastName: "Noyer",
  pays: "France",
  age: 36
}

console.log(julien.firstName);  // Affiche "Julien"
console.log(julien.pays);  // Affiche "France"
console.log(julien.age);  // Affiche 36

julien.pays = "Italie";
console.log(julien.pays);  // Affiche maintenant "Italie"
```

AJOUTER UNE FONCTION A UN OBJET

Il est possible d'ajouter directement une fonction en paramètre d'un objet pour, par exemple, définir automatiquement le nom complet d'un utilisateur. Pour cela, il suffi de passer en paramètre de l'objet, une fonction qui prendra en références les paramètre de l'objet.

DECLARATION DE L'OBJET

Les paramètres de l'objet sont écrits entre accolades et séparées par une virgule

AJOUTER UNE FONCTION

Pour pouvoir accéder aux paramètres de l'objet il faut ajouter le mot-clé this. avant des les appeler

AFFICHER LE NOM COMPLET

L'appel de la function() ce fait de la même façon qu'un paramètre d'objet classique

```
var julien = {
   firstName: "Julien",
   lastName: "Noyer",
   age: 36,

   // Ajout d'une fonction qui calcule le nom complet
   fullName: function() {
      return this.firstName + " " + this.lastName;
   }
};

console.log(julien.fullName());  // Affiche "Julien Noyer"
```

Cette méthode est limitée car elle nécessite l'ajout de la fonction en paramètre de chaque objet devant réagir de la même façon

LA PROGRAMMATION ORIENTEE OBJET

Créée en 1960 par Ole-Johan Dahl et Kristen Nygaard, la programmation orientée objet applique aux langages de programmation le principe des objets physiques tel qu'une voiture, un jouet ou-bien un immeuble. Comme dans le monde réel, un objet JavaScript possède des propriétés et des comportements qui définissent leur type.

JE SUIS UN ROBOT

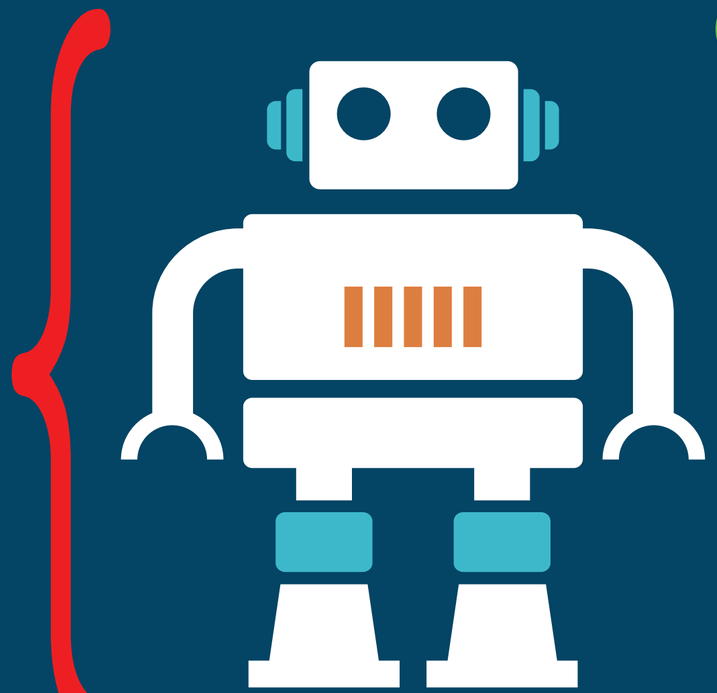
J'ai un corps solide et mobile et je m'appel Antoine

JE SUIS COLORE

Je sais reconnaitre et reproduire les couleurs

JE ME DEPLACE

J'analyse mon environnement et je me repère dans l'espace



FAIRE UNE PHOTO

Je prend une photo quand quelque chose passe devant mes yeux

FAIRE UN DESSIN

Quand quelqu'un me le demande, je peux dessiner un moutont

FAIRE UN CAFE

Je prépare un café tous les matins à 9h30

CREER UN TYPE D'OBJETS

La programmation orientée objet nous permet de créer nos propres types d'objets pour ensuite créer des objets par type. Un type d'objet s'écrit en UpperCamelCase et possède des propriétés et des fonctionnements qui sont automatiquement associés au xnouveaux objets.Il faut donc dans un premier temps définir le type d'objet, ces propriétés et ses fonctions pour ensuite créer des objets du type défini.

TYPE D'OBJET

Un type d'objet est une fonction qui prend en paramètres les propriétés du type d'objet, le mot-clé this. fait référence au tye d'objet

FONCTION DU TYPE D'OBJET

La propriété .prototype permet de définir les fonctions du type d'objet

CREATION DES OBJETS

Les objets sont des variable du type d'objet

FACTORISATION

La fonction s'applique à tous les nouveaux objets

```
function Users (firstName, lastName) {
   this.firstName = firstName;
   this.lastName = lastName;
};

Users.prototype.fullName = function() {
   return this.firstName + " " + this.lastName;
};

var julien = new Users("Julien", "Noyer");
var chuck = new Users("Chuck", "Norris");

console.log(julien.fullName());  // Affiche "Julien Noyer"
console.log(chuck.fullName());  // Affiche "Chuck Norris"
```