JAVA SCRIPT

Le langage le plus populaire, On peut s’en service pour créer des scripts

Ça peut modifier le contenu html. LE BUT c’est de rendre notre page interactive

Comme l’exemple de la lampe : https://www.w3schools.com/js/tryit.asp?filename=tryjs\_intro\_lightbulb

Pour cela on définit des id à nos éléments et/ou des classes

Comment cibler un élément ?

On lui met un id et dans le css on va le relier avec onclick= document.getElementById("notrefichiers").innerHTML

(Sur le html dans le corps) One click = quand on clique

Si on veut le mettre en script, on le met juste avant la fermeture de body

Chaque script pour une fonctionnalité, Il est préférable d’utiliser cette pratique

Les variables

Sont des données qui vont contenir des valeurs différentes

Pour des valeurs constantes qui ne changent pas on utilise : const X …. (ça ne changera pas)

Le mot - clé let pour définir une variable à portée restreinte.

Une ligne de variable s’appelle instruction (ça a donner un ordre au pc)

<script> const x = 10; let y =20; y = 40; </script>

On a intégré la valeur y=40 a la let 20

Pour le debuggage ;

En dessous de la ligne const on met :

-Console.log(x) ;

En dessous de la ligne let on met :

-Console.log(y) ;

En dessous de la ligne y on met :

-Console.log(y) ;

///

‘ = simple quotte

‘’ = double quotte

Les anti slash « \ » servent à rajouter d’autre guillemets. On y met un caractère d’échappement, c’est-à-dire un caractère qui sera afficher tel quel, dans son état normal.

Exemple :

Let chaine = "chaine \"mot en quotte\" reste de la chaine » ;

Si on ne les met pas, il va penser que la commande s’arrête au mot chaine

(Je n’ai toujours rien compris)

Chaine = 'it\'s a good day';

Sur cette exemple on utilise des simple quotte pour éviter que les apostrophes soient mal interprétées

Le double \\ :

Le premier \ lui dit que c’est un caractère d’échappement donc il ne va pas s’afficher donc à l’image on va en voir qu’un.

Exemple : let dossier = "dossier \\mes images\\ image.png » ;

console.log(dossier) ; 🡪 Il va afficher \mes images\

Objet

Exemple : voiture

Un objet contient des attribues : -moteur -4 roues -puissance

Et les méthodes : -rouler-accélérer-freiner

Fonction à écrire

Objet.Attribut === voiture.nbroues

Objet.Méthode() === voiture.Rouler()

Pour récupérer un élément dans l’objet, on met objet.nomde l’attribut

* JSON : JavaScriptObjetNotation = comment on va construire un objet

Objet = { attr : val,

‘’ :’’

Fonct: fonction }

<https://www.w3schools.com/js/js_objects.asp>

La différence entre l’attribut et la méthode, c’est que après la méthode on met les ‘’()’’

Pour l’afficher dans la console, on met (console.log(on met notre contenu))

Dans le contenu on met le nom de l’objet ou de l’attribut, et après le point on met la méthode ou le résultat de la fonction.

This , référence l’objet en cours 🡪 Le mot - clé this

Dans une définition de fonction, this fait référence au "propriétaire" de la fonction.

Dans l'exemple ci-dessus, this est l’objet personne qui "possède" la fonction fullName.

* En d'autres termes, this.firstName signifie la propriété firstName de cet objet .

(On l’utilise à l’intérieur de l’objet), (Toujours le même lien)

Math.random() = pour afficher une valeur aléatoire qu’on peut paramétrer

Math.floor(Math.random() \* 5.4);

Le 5.4 Est le paramètre

Le floor, arrondi le chiffre

Ce sont des fonctions prédéfinies par JS

(Une majuscule pour chaque nouveau mot, à part le premier)

Une fonction se déclare avec le terme function afficherMessage()

La fonction s’exécute seulement au moment où l’on fait l’appel

function afficherMessage()

{console.log("salut les mecs");

}

afficherMessage();

En bleu : l’appel

Entre les accolade : les valeurs fictives et en dessous, les valeurs réelles

Function somme( a,b)

{Console.log (a+b) ;

}

Somme (2,3) ;

Regarder le VS code dans le github

Val1 = Math.random() \* 10 ;

En entrant cette formule, on dit que la val1 (que j’ai créé) me donne un chiffre au piff entre 0 et 10. (avec virgule)

Donc la valeur de cette formule équivaut à une ‘’valeur X’’

Pour qu’il m’affiche un nombre entier on écrit :

X = Math.floor (Math.random () \*10) ;

console.log(X)

Math.floor définie le nombre entier,

Soit un met entre parenthèse, la formule entière comme au-dessus,

Ou on met la valeur finale :

Math.floor(X) ;

Tout en précisant la valeur de X

X = math.random() \* 10

Booléen :

‘’False’’, est une chaine de caractère ce n’est pas le false booléen quand c’est entre guillemet.

False, c’est le false booléen, donc la valeur est fausse.

Tout ce qui n’a pas de « valeur » est faux

Cas exceptionnel :

Chaine="’’ ; = false

Chaine=" ’’ ; = true

Il y a un espace dans la deuxième c’est pour ça qu’il met true

UN ESPACE EST CONSIDERER COMME UN CARACTETRE

X= 1 🡺 nombre entier

Y= 1.0 // équivaut à 1 = entier

X === y (triple égale c’est égalité + du même type)

Opérateur logique

A= true

(!a) ; // false

Le point d’exclamation c’est la négation.

A= true ; b = false ;

A != b => true

La négation de A est égale à b

! == Différentes valeurs ou de types différents

1 === ‘’1’’ => False ‘’1’’= chaine de caractère

1 ! == ‘’1’’ => True 1 = nombre entire

True and false = false

True and true = true

False and false = false

&& = and = et

(S’il y a une opérante qui est false, tout est faux)

True OU true = true

True OU false = true

False OU false = false

|| = or = ou

Condition js

Si/if (condition) {Traitement 1}

Sinon/else (condition) {Traitement 2}

‘’Else if’’ on peut en rajouter autant de fois qu’on veut

On peut aussi l’écrire d’une manière différente :

Exemple :

A<B ? X=’’inférieur’’ : X=’’supérieur’’ ;

Qui veux dire :

? = la question

: = sinon

Si à est inférieur à B il affiche inférieur sinon il affiche supérieur

Pour la fonction, il faut mettre les bases : (définition)

Function lenomdelafonction()

{

}

C’est le début

Entre les deux accolades, on fait le traitement de la fonction.

La valeur qui est entre parenthèse (paramètre fictif) quand je fais appel à la fonction,

Je lui inculque un paramètre réel rattaché à une variable (paramètre réel)

* Tout fonction qui fait retourner une valeur, doit être mémoriser dans une variable, sinon sa valeur sera perdue.

function age2(chiffre) {

  let message = "";

  if (chiffre <= 10) {

    message = "enfant";

  } else if (chiffre <= 15) {

    mesage = "adolescent";

  } else if (chiffre <= 20) {

    message = "jeune";

  } else if (chiffre <= 45) {

    message = "adulte";

  } else if (chiffre <= 100) {

    message = "senior";

  } else {

    message = "trop vieux zakaria";

  }

  return message;

}

let resultats = age2(152);

console.log(resultats);

Switch

Switch (expression)

{ Cas 1 :

Traitement 1 ; break ;

Traitement 2 ; break ;

-

-

-

Default (dernier cas) :

Traitement DernierCas ;

}

Exemple :

function lesages(age3) {

let numbre = "";

switch (true) {

case (age3 <= 10):

numbre = "enfant";

break;

case (age3 <= 15):

numbre = "adolescent";

break;

case (age3 <= 20):

numbre = "jeune";

break;

case (age3 <= 45):

numbre = "adulte";

break;

case (age3 <= 100):

numbre = "senior";

break;

default:

numbre = "trop vieux";}

return numbre;}

console.log(lesages(19));

console.log(lesages(14));

console.log(lesages(45));

console.log(lesages(79));

console.log(lesages(120));

les boucles forte

Exemple : boucle forte avec function et tableau

function entiers(tab) {

  let message =""

let i;

for (i = 0; i < cars.length; i++)

{

  message+= tab[i] +"\n";

}

return message

}

let tab1= [1,2,3,4,5,6,7];

console.log(entiers(tab));

La boucle For / In

on utilise la boucle for lorsque on connais la valeur initial et la valeur final

L' for/in instruction JavaScript parcourt les propriétés d'un objet:

Syntaxe

for (key in object) {  
  // *code block to be executed*  
}

Exemple

//for (initialization ; condition ;iteration)

For (var i=1 ;i<10 ; i\*=2){

Console.log(i) ;}

let person = {fname:"John", lname:"Doe", age:25};

var text = "";  
let x;  
for (x in person) {  
  text += person[x];  
}

Exemple expliqué

* La boucle **for in** itère sur un objet **personne**
* Chaque itération renvoie une **clé** (x)
* La clé est utilisée pour accéder à la **valeur** de la clé
* La valeur de la clé est **person [x]**

## Array. forEach()

La forEach()méthode appelle une fonction une fois pour chaque élément d'un tableau, dans l'ordre.

**Remarque:** la fonction n'est pas exécutée pour les éléments de tableau sans valeurs.

### **Example**

var txt = "";  
var numbers = [45, 4, 9, 16, 25];  
numbers.forEach(myFunction);  
  
function myFunction(value, index, array) {  
  txt = txt + value + "<br>";  
}

The Break Statement

L' breakinstruction peut également être utilisée pour sortir d'une boucle:

### **Exemple**

for (i = 0; i < 10; i++) {  
  if (i === 3) { break; }  
  text += "The number is " + i + "<br>";  
}

La déclaration Continue

L' continueinstruction interrompt une itération (dans la boucle), si une condition spécifiée se produit, et continue avec l'itération suivante dans la boucle.

Cet exemple ignore la valeur de 3:

Exemple

for (i = 0; i < 10; i++) {  
  if (i === 3) { continue; }  
  text += "The number is " + i + "<br>";  
}

## les boucles While Loops do while

## dans la boucle while en fait incrementation manuel

## exemple

function exemple(tab) {

  let message = "";

  let i;

  for (i = 0; i < tab.length; i++) {

    if (tab[i] === "non") {

      continue;

    }

    message += tab[i] + "\n";

  }

  return message;

}

let tab1 = ["yes", "oui", "no", "non", "yeah", "nous", "merci"];

console.log(exemple(tab1));

## La boucle While

## exemple

function exemple(x) {

  let message = "";

  let i = 0;

  while (x[i]) {

    if (x[i] === "Saab") {

      i++;

      continue;

    }

    message += x[i] + "\n";

    i++;

  }

  return message;

}

message1 = ["BMW", "Volvo", "Saab", "Ford"];

console.log(exemple(message1));

## Cas exep

function mots(tab) {

    let i = -1;

    let message = "";

    while (i < tab.lengt-1) {

        i++;

        if (tab [i]=== "non" ){

continue;

        }

message += tab[i]+"\n";

    }

return message;

}

    let tab1 = ["aaa", "bbb", "ccc", "ddd"];

## Creating an Array

## L'utilisation d'un littéral de tableau est le moyen le plus simple de créer un tableau JavaScript.

## Syntaxe:

var array\_name = [item1, item2, ...];

Exemple

var cars = ["Saab", "Volvo", "BMW"];

## Modification d'un élément de tableau

Cette instruction modifie la valeur du premier élément dans cars: Cette instruction modifie la valeur du premier élément dans cars:

cars[0] = "Opel";

Exemple

var cars = ["Saab", "Volvo", "BMW"];  
cars[0] = "Opel";  
document.getElementById("demo").innerHTML = cars[0];

## Numeric Sort (Tri numérique)

pour faire lordre alphabétique et numeriques

.sort()

let fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.sort();

console.log(fruits)

## Ajout des éléments aux tableau

Le moyen le plus simple d'ajouter un nouvel élément à un tableau consiste à utiliser la push()méthode:

Exemple

let fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.push("Lemon"); //pour **ajouter un nouvel élément à la fin**

au lieux faire fruits[fruits.length (n.des éléments de tableaux)]=‘’lemon’’;

fruits.unshift("Lemon");//pour **ajouter un nouvel élément au debut**

fruits.pop();// pour **supprime le dernier élément de tableau**

fruits.shift();// pour **supprime le premier élément de tableau**

Tableaux associatifs

### **Exemple:**

let person = [];  
person["firstName"] = "John";

clé  
person["lastName"] = "Doe";

doe  
person["age"] = 46;  
let x = person.length;     // person.length will return 0  
let y = person[0];         // person[0] will return undefined

JavaScript Array Methods

## Pushing

## La push()méthode ajoute un nouvel élément à un tableau (à la fin):

### **Exemple**

var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.push("Kiwi");       //  Adds a new element ("Kiwi") to fruits

Exemple : tableau d’objet

let objTab = [

{nom : ‘’james’’ ; age : 21 ;@ :’’roubaix’’}

{nom : ‘’farouk’’ ; age : 28 ;@ :’’lille’’}

{nom : ‘’directeur’’ ; societe ;@ :’’afpa’’}

]

JavaScript Math Object

Exemple

Math.PI;            // returns 3.141592653589793

Math.pow ()

Math.pow(x, y) renvoie la valeur de x à la puissance de y:

Exemple

Math.pow(8, 2);      // returns 64

Math.sqrt ()

Math.sqrt(x) renvoie la racine carrée de x:

Exemple

Math.sqrt(64);      // returns 8

https://www.w3schools.com/js/js\_math.asp

JavaScript Date Objects

Example

let d = new Date();

new (instructeur)

les mois en JS sont indexer de 0-11 mois

JavaScript ISO Dates

ISO 8601 est la norme internationale pour la représentation des dates et des heures.

La syntaxe ISO 8601 (AAAA-MM-JJ) est également le format de date JavaScript préféré:

### Exemple (date complète)

let d = new Date("2015-03-25");

## Recherche d'une chaîne dans une chaîne

La indexOf()méthode retourne l'index de (la position de) l' firstoccurrence d'un texte spécifié dans une chaîne:

### **Exemple**

var str = "Please locate where 'locate' occurs!";  
var pos = str.indexOf("locate");

## La méthode charAt ()

La charAt()méthode renvoie le caractère à un index (position) spécifié dans une chaîne:

### **Exemple**

var str = "HELLO WORLD";  
str.charAt(0);            // returns H

D

DOM : Document Objet Model

SYSTEM DOMBOITEMEnt <body>(<header>(<h1><p>))

## Événements HTML courants

Voici une liste de quelques événements HTML courants:

|  |  |
| --- | --- |
| **un événement** | **La description** |
| onchange (sur le changement | Un élément HTML a été modifié |
| Onclick (sur clic | L'utilisateur clique sur un élément HTML |
| Onmouseover | L'utilisateur déplace la souris sur un élément HTML |
| Onmouseout (à la souris | L'utilisateur éloigne la souris d'un élément HTML |
| onkeydown | L'utilisateur appuie sur une touche du clavier |
| Onload (en charge | Le navigateur a fini de charger la page |

Methode de integre javasecrept

<script> </script>

Sdn

Une autre fichier js

**Définition ( function )**

Une fonction est un groupe de ligne(s) de code de programmation destiné à exécuter une tâche bien spécifique et que l'on pourra, si besoin est, utiliser à plusieurs reprises. De plus, l'usage des fonctions améliorera grandement la lisibilité de votre script.

En Javascript, il existe deux types de fonctions :

* les fonctions propres à Javascript. On les appelle des "méthodes". Elles sont associées à un objet bien particulier comme c'était le cas de la méthode Alert()
* les fonctions écrites par vous-même pour les besoins de votre script.

Les tableaux : les fonctions slice splice

La fonction **Slice** elle affiche une partie de tableau grâce a un indices sans modification de tableau Ex : let noms=noms.slice(2.4)🡨l’indice d’où il commence a cope 2 jusqu’à l’argument d’arrivé 3 et déclarer le prochain 4 (1+)

https://www.youtube.com/watch?v=qZGtDqP66So

La fonction **Splice** elle peux ajouter et suprimer et remplacier

Ex : let noms=noms.slice(2.1)🡨 (2,…) le premier argument sa veux dire cest a partir de la quil va supprime ou modifier ,et le deuxième argument yomathilo le nombre des éléments a supprime (il va supprime que le 2eme element)

.Splice il supprime l’élément est-il va le mettre dans un autre élément

Ex : let noms=noms.splice(2,0,‘’farouk’’)🡨 il va ajouter l’élément âpre le premier indice 2

.Splice il va modifier le tableau principal et il va ajouter farouk apre 2eme

Ex : let noms=noms.splice(-1,0,‘’farouk’’)🡨il va commencercon par la fin du tableau

les fonctions concat- reverse - join - sort

1. La concat()méthode est utilisée pour joindre deux ou plusieurs tableaux.

Cette méthode ne modifie pas les tableaux existants, mais renvoie un nouveau tableau, contenant les valeurs des tableaux joints.

Syntaxe… Ex

*<script>*

*function myFunction() {*

*var hege = ["Cecilie", "Lone"];*

*var stale = ["Emil", "Tobias", "Linus"];*

*var kai = ["Robin"];*

*var children = hege.concat(stale,kai);*

*document.getElementById("demo").innerHTML = children;*

*}*

*</script>*

1. La reverse()méthode inverse l'ordre des éléments dans un tableau.

**Remarque:** cette méthode modifiera le tableau d'origine.

Syntaxe… Ex

*<script>*

*var fruits = ["Banana", "Orange", "Apple", "Mango"];*

*document.getElementById("demo").innerHTML = fruits;*

*function myFunction() {*

*fruits.reverse();*

*document.getElementById("demo").innerHTML = fruits;*

*}*

*</script>*

1. La join()méthode renvoie le tableau sous forme de chaîne.

Les éléments seront séparés par un séparateur spécifié. Le séparateur par défaut est la virgule (,).

**Remarque:** cette méthode ne modifiera pas la matrice d'origine.

Syntaxe… Ex

*<script>*

*function myFunction() {*

*var fruits = ["Banana", "Orange", "Apple", "Mango"];*

*var x = document.getElementById("demo");*

*x.innerHTML = fruits.join();*

*}*

*</script>*

1. La méthode sort () trie les éléments d'un tableau.

L'ordre de tri peut être alphabétique ou numérique, et croissant (vers le haut) ou décroissant (vers le bas).

Par défaut, la méthode sort () trie les valeurs sous forme de chaînes par ordre alphabétique et croissant.

Cela fonctionne bien pour les chaînes ("Apple" vient avant "Banana"). Cependant, si les nombres sont triés sous forme de chaînes, "25" est plus grand que "100", car "2" est plus grand que "1".

Pour cette raison, la méthode sort () produira un résultat incorrect lors du tri des nombres.

Vous pouvez résoudre ce problème en fournissant une «fonction de comparaison» (voir «Valeurs des paramètres» ci-dessous).

Remarque**:** cette méthode modifie le tableau d'origine.

Syntaxe… Ex

*<script>*

*var points = [40, 100, 1, 5, 25, 10];*

*document.getElementById("demo").innerHTML = points;*

*function myFunction() {*

*points.sort(function(a, b){return a-b});*

*document.getElementById("demo").innerHTML = points;*

*}*

*</script>*

<https://www.w3schools.com/jsref/jsref_sort.asp>