

INTRODUCTION AU JAVASCRIPT

CONSOLE.LOG("BIENVENUE À TOUS");

LET'S BEGIN....

PRÉSENTATION

- PERMET D'ANIMER ET DE DYNAMISER DES PAGES WEB —
- JAVASCRIPT EST SURTOUT UTILISÉ CÔTÉ CLIENT —
- PEUT ÊTRE UTILISÉ POUR CRÉER D'AUTRES APPLICATIONS (TOM TOM) —

HISTOIRE

— 1995 —

— SUN DÉVELOPPA UN LANGAGE INFORMATIQUE PERFORMANT APPELÉ JAVA —
— TROP PUISSANT ET COMPLIQUÉ POUR UNE UTILISATION AU SEIN D'APPLICATIONS WEB —

— SUN ET NETSCAPE —

— LIVESCRIPT —

— NOTORIÉTÉ JAVA => JAVASCRIPT —

AJOUTER VERSION JS

ECMASCRIPT 2018
JS ES6

JQUERY

— BIBLIOTHÈQUE OU UNE API (APPLICATION PROGRAMMING INTERFACE) —
— 2006 —
— JOHN RESIG —

JAVASCRIPT : DOCUMENT.GETELEMENTSBYTAGNAME("LI")[0].INNERHTML="LAIT";
JQUERY : \$("LI:FIRST").TEXT("LAIT");

HTML/CSS VS JS

LANGAGE DE CONCEPTION VS LANGAGE DE PROGRAMMATION

LES OUTILS POUR PRATIQUER

— SUBLIMETEXT —
— UN NAVIGATEUR GOOGLE CHROME —
— CONSOLE GOOGLE CHROME —

OU Ecrire du code Java Script ?

— SOIT DANS UN FICHIER .JS —

— SOIT DIRECTEMENT DANS NOTRE FICHIER HTML ENTRE DEUX BALISES <SCRIPT></SCRIPT> —

OU Ecrire du code Java Script ?



```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>Jqueri UI resizable</title>
    <link rel="stylesheet" href="style.css" />

    <script type="text/javascript">
      Écrire mon code JAVASCRIPT ici
    </script>
  </head>
  <body>
    <h1>Titre de la page</h1>
  </body>
</html>
```

The image shows a code editor interface with an HTML file open. A red rectangular box highlights the content of a `<script>` tag. Inside this box, the text "Écrire mon code JAVASCRIPT ici" is displayed, indicating where the user should write their JavaScript code. The rest of the HTML structure, including the DOCTYPE declaration, language attribute, title, link to a stylesheet, and the main body content with an h1 tag, is visible outside the highlighted area.

IL N'EST PLUS NÉCESSAIRE D'INDIQUER LE « TYPE = 'TEXT/JAVASCRIPT' » SOUS HTML 5.

OU Ecrire du code Java Script ?

```
1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <h2>External JavaScript</h2>
6
7     <p>Tout mon code JS est dans le fichier appelé app.js</p>
8
9     <script src="app.js"></script>
10
11   </body>
12 </html>
```

BALISE <NOSCRIPT>

```
1 <noscript>
2   <!-- anchor linking to external file -->
3   <a href="https://www.mozilla.com/">External Link</a>
4 </noscript>
5 <p>Rocks!</p>
```

PLAN DU COURS

— SYNTAXE & AFFICHAGE —

LES DÉCLARATIONS & LES AFFECTATIONS

LES OPÉRATEURS D'AFFECTATION

LES VARIABLES, LEURS TYPES ET LEUR PORTÉE

DÉCLARATIONS ET AFFECTATIONS

STRINGS / CONCATÉNATION

NOMBRES & MÉTHODES

TABLEAUX & MÉTHODES

BOOLEANS

CONSTANTES

DATES, FORMAT & GET

MATH / RANDOM

ARRAY & OBJECT

TYPE CONVERSION

REGEXP

PLAN DU COURS

— LES STRUCTURES ITÉRATIVES —

LES BOUCLES
WHILE
FOR
DO WHILE
BREAK
CONTINUE

PLAN DU COURS

— LES CONDITIONS & COMPARAISONS —

IF ELSE ELSE IF
LES OPÉRATEURS D'AFFECTATION
LES OPÉRATEURS DE COMPARAISON
LES OPÉRATEURS LOGIQUES
EXPRESSIONS
SYNTAXE DES CONDITIONS DE CHOIX
SWITCH

PLAN DU COURS

— LES FONCTIONS & CLASSES —

FONCTIONS
ARROW FONCTIONS
FONCTIONS SCOPE
FONCTIONS PRÉDÉFINIES
FONCTIONS UTILISATEUR

FE
FONCTION ET PORTÉE DE VARIABLES
LES CLASSES
ERRORS / UNDEFINED

PLAN DU COURS

— MANIPULATION DU DOM —

LES SÉLECTEURS
LES ÉVÉNEMENTS
COOKIES & SESSIONS
JS JSON
JS FORMS
BEST PRACTICES
LIBRAIRIES
NOUVEAUTÉS JAVASCRIPT
JQUERY

JAVASCRIPT

- SYNTAXE & AFFICHAGE -

SYNTAXE & AFFICHAGE

- CASE SENSITIVE -
(SENSIBLE À LA CASSE)

- MESSAGE AFFICHÉ EN QUOTES OU GUILLEMETS -

- UNE INSTRUCTION JS SE TERMINERA TOUJOURS PAR UN POINT VIRGULE -

JAVASCRIPT

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Title of the document</title>

    <!-- Popup d'alert -->
    <script> alert("Bonjour");</script>

    <!-- Popup de confirmation -->
    <script type="text/javascript"> confirm('Tu commences le JS');</script>

    <!-- Affiche un message en console -->
    <script type="text/javascript"> console.log('Voici la console');</script>
  </head>
  <body>
  </body>

  <script src="app.js"></script>

</html>
```

LES COMMENTAIRES

1
2
3 // mon commentaire

LES COMMENTAIRES

```
1
2
3 /* 
4 mes commentaires
5 sur plusieurs lignes */
```

DÉCLARATIONS & AFFECTATIONS

NOUS DÉCLARONS UNE DONNÉE (VARIABLE, ARRAY, OBJECT, CONSTANTE, LET...)

NOUS LUI AFFECTONS UNE VALEUR

TYPE CONVERSION

OBJETS NATIF

EN JAVASCRIPT IL Y A **5 TYPES DE DONNÉES** QUI PEUVENT CONTENIR DES VALEURS :

- STRING
- NUMBER
- BOOLEAN
- OBJECT
- FUNCTION

ET IL Y A EN JAVASCRIPT **6 TYPES D'OBJETS** :

- OBJET
- DATE
- ARRAY
- STRING
- NUMBER
- BOOLEAN

VARIABLES

EMPLACEMENT DE STOCKAGE NOMMÉ

-> PEUT ÊTRE MODIFIÉ PENDANT L'EXÉCUTION D'UN SCRIPT

NOM UNIQUE, PREND UNE PLACE EN MÉMOIRE ET PERMET DE STOCKER LA VALEUR

VARIABLES

```
var monMessage= 'Coucou !';  
  
alert(monMessage);  
// Affichera son contenu DONC 'Coucou !'  
  
alert('monMessage');  
// Affichera 'monMessage' car se trouve entre quottes
```

VARIABLES

```
1 <script>
2
3 // affichera Pierre
4
5 var monPrenom = "Pierre";
6 alert(monPrenom) ;
7
8 var monAge = 27;
9
10 //les chiffres ne prennent pas de quotes
11 // Ici j'affecte la valeur que
12 //l'internaute rentrera dans le champ généré par prompt
13 // La valeur de monAge ne sera donc potentiellement plus le chiffre 27
14
15 var monAge = prompt('Quel âge as-tu ?');
16 alert(monAge);
17
18 // Ici Marie prendra la place de Pierre dans la variable
19 // affichera Marie
20 var monPrenom = "Marie";
21 alert(monPrenom) ;
22
23 </script>
```

TYPE CONVERSION

IL EXISTE ÉGALEMENT DEUX TYPES DE DONNÉES QUI NE PEUVENT CONTENIR DE VALEUR:

- NULL
- UNDEFINED

C'EST TYPE DE DONNÉES SONT IDENTIFIABLES GRÂCE AU **TYPEOF()**

OPÉRATEURS D'AFFECTION

OPÉRATEUR ARITHMÉTIQUES

OPÉRATEURS D'AFFECTATION

| Nom | Opérateur (raccourci) | Signification |
|---|-------------------------------|-----------------------------------|
| Affectation | <code>x = y</code> | <code>x = y</code> |
| Affectation après addition | <code>x += y</code> | <code>x = x + y</code> |
| Affectation après soustraction | <code>x -= y</code> | <code>x = x - y</code> |
| Affectation après multiplication | <code>x *= y</code> | <code>x = x * y</code> |
| Affectation après division | <code>x /= y</code> | <code>x = x / y</code> |
| Affectation du reste | <code>x %= y</code> | <code>x = x % y</code> |
| Affectation après exponentiation | <code>x **= y</code> | <code>x = x ** y</code> |
| Affectation après décalage à gauche | <code>x <= y</code> | <code>x = x << y</code> |
| Affectation après décalage à droite | <code>x >= y</code> | <code>x = x >> y</code> |
| Affectation après décalage à droite non-signé | <code>x >>> y</code> | <code>x = x >>> y</code> |
| Affectation après ET binaire | <code>x &= y</code> | <code>x = x & y</code> |
| Affectation après OU exclusif binaire | <code>x ^= y</code> | <code>x = x ^ y</code> |
| Affectation après OU binaire | <code>x = y</code> | <code>x = x y</code> |

PORTEE DES VARIABLES

NOUS POUVONS MANIPULER LA VALEUR DES VARIABLES

LA VARIABLE EST DITE ACCESSIBLE, MAIS CELA DEPEND DE L'ENDROIT OÙ ELLE EST DECLARÉE.

VARIABLE GLOBALE

VARIABLE LOCALE

VARIABLE LOCALE

VAR MONMESSAGE = 'SALUT' ;

ICI LA PORTÉE DE LA VARIABLE DÉPEND DE L'ENDROIT OÙ ELLE EST DÉCLARÉE.

VARIABLE GLOBALE

MONMESSAGE = 'SALUT' ;

(PAS DE MOT CLÉ VAR)

ACCESSIBLE DE PARTOUT PEU IMPORTE SA POSITION

LOCALE VS GLOBALE

UNE VARIABLE DÉCLARÉE **EXPLICITEMENT EN DEHORS** DE TOUTE FONCTION, POURRA ÊTRE UTILISÉE N'IMPORTE OÙ DANS LE SCRIPT. ON PARLE DE VARIABLE GLOBALE.

UNE VARIABLE DÉCLARÉE **EXPLICITEMENT À L'INTÉRIEUR** D'UNE FONCTION AURA UNE PORTÉE LIMITÉE À CETTE SEULE FONCTION.

EXEMPLE

```
var a = 12;

function MultipliePar2(b) {
    var a = b * 2;
    return a;
}

document.write("Le double de 4 est ",MultipliePar2(4));
document.write('<br />');
document.write("La valeur de a est ",a);
```

EXEMPLES DE PORTÉES DE VARIABLES

```
1 <script>
2
3     // Ceci est une variable globale
4     var maVariable = 2;
5     function myfunction(){
6
7         console.log(maVariable);
8
9         // Ceci est une variable locale
10        var maVariableLocale = "Ne se voit pas à l'extérieur de la fonction";
11
12    };
13
14    // Va afficher en console 2
15    myfunction();
16
17    // Va afficher une erreur en console
18    console.log(maVariableLocale);
19
20 </script>
```

TYPES DE VARIABLES

STRING
BOOLEAN
INTEGER
BIG INTEGER
ARRAY
OBJECT

....

TYPEOF()

CHAÎNE DE CARACTÈRES (STRINGS)

```
var a = "Salut !";  
//console.log("type of");  
alert(typeof(a)); // Affichera « string »
```

MÉTHODE DE CHAÎNE (STRING)

LENGTH
INDEXOF()
LASTINDEXOF()
SEARCH()
SLICE() SUBSTRING()
SUBSTR()
REPLACE()
TOUPPERCASE()
TOLOWERCASE()
CONCAT()
TRIM()
CHARAT()
SPLIT()
...

NUMBERS

```
var a = 36 ;  
alert(typeof(a)) ; // Affichera « number »
```

NUMBERS METHODS

TOSTRING()
TOEXPONENTIAL()
TOFIXED()
TOPRECISION()
VALUEOF()
NUMBER()
PARSINT()
PARSEFLOAT()
MAX_VALUE
MIN_VALUE
.NAN

JS GLOBAL

ISNAN()

```
// isNaN  
if(!isNaN(varb)) document.write('ce n`est pas un number - Not a Number<br>');
```

BOOLEANS

```
var a = true ;  
var b = false ;
```

UNE DONNÉE A TOUJOURS UNE VALEUR, C'EST-À-DIRE QU'ELLE EST SOIT VRAIE SOIT FAUSSE (TRUE OU FALSE).

CONSTANTES

```
const PI = 3.141592653589793;  
PI = 3.14;           // Cela donnera une erreur  
PI = PI + 10;      // Cela donnera une erreur
```

LES CONSTANTES NE PEUVENT ÊTRE RÉAFFECTÉES

TABLEAUX (ARRAY)

```
// Les tableaux JavaScript sont utilisés pour stocker des valeurs multiples dans une seule variable.  
var cars = ["Saab", "Volvo", "BMW"];  
console.log(cars);
```

```
▼ (3) ["Saab", "Volvo", "BMW"] ⓘ  
  0: "Saab"  
  1: "Volvo"  
  2: "BMW"  
  length: 3  
► __proto__: Array(0)
```

>

MÉTHODES DE TABLEAUX

TOSTRING()
JOIN()
PUSH()
SHIFT()
UNSHIFT()
DELETE

...

ARRAY MULTIDIMENSIONNEL

```
var deuxDimensions = [ [1, 'pierre', 3], [1, 1, 'laurie'] ];  
document.write(deuxDimensions[0][1]);
```

DATES

```
var d = new Date();
console.log(d);
// Tue Dec 17 2019 20:24:58 GMT+0100 (Central European Standard Time)
```

DATES FORMAT & GET

| Method | Description |
|-------------------|---|
| getFullYear() | Get the year as a four digit number (yyyy) |
| getMonth() | Get the month as a number (0-11) |
| getDate() | Get the day as a number (1-31) |
| getHours() | Get the hour (0-23) |
| getMinutes() | Get the minute (0-59) |
| getSeconds() | Get the second (0-59) |
| getMilliseconds() | Get the millisecond (0-999) |
| getTime() | Get the time (milliseconds since January 1, 1970) |
| getDay() | Get the weekday as a number (0-6) |
| Date.now() | Get the time. ECMAScript 5. |

DATES FORMAT & GET

```
var d = new Date("2015-03-25");
var d = new Date("2015-03");
var d = new Date("2015");
var d = new Date("2015-03-25T12:00:00Z");
var d = new Date("2015-03-25T12:00:00-06:30");
var d = new Date("03/25/2015");
var d = new Date("2015/03/25");
var d = new Date("25-03-2015");
var d = new Date("Mar 25 2015");
var d = new Date("25 Mar 2015");
var d = new Date("January 25 2015");
var d = new Date("Jan 25 2015");
var d = new Date("JANUARY, 25, 2015");
```

MATH

```
// L'objet JavaScript Math vous permet d'effectuer des tâches  
// mathématiques sur les nombres.  
  
Math.PI; // returns 3.141592653589793  
  
// Math.round(x) renvoie la valeur de x arrondi au  
// nombre entier le plus proche:  
  
Math.round(4.4); // returns 4  
  
// Math.pow(x, y) Renvoie la valeur de x à la puissance y:  
Math.pow(8, 2); // returns 64  
  
// Math.floor(x)renvoie la valeur de x arrondi vers le bas  
// à son entier le plus proche:  
Math.floor(4.7); // returns 4
```

RANDOM

```
// // renvoie un nombre aléatoire compris entre 0 (inclus) et 1 (exclusif):  
Math.random();|
```

REGEXP

UNE EXPRESSION RÉGULIÈRE EST UNE SÉQUENCE DE CARACTÈRES QUI FORME UN MOTIF DE RECHERCHE .

**LORSQUE VOUS RECHERCHEZ DES DONNÉES DANS UN TEXTE, VOUS POUVEZ UTILISER CE MODÈLE DE
RECHERCHE POUR DÉCRIRE CE QUE VOUS RECHERCHEZ.**

LES REGEXP SONT UTILISÉS POUR DES RECHERCHES ET REMplacements DE TEXTE

EXEMPLE DE REGEX

```
// La méthode search va chercher la chaîne de caractères
// IFOCOP dans str
var str = "Vive IFOCOP!";
var n = str.search("IFOCOP");
console.log(n);
```

```
// Ici nous avons créé une expression régulière
// Le i indique la recherche n'est pas sensible à la casse
var str = "Vive Ifocop";
var n = str.search(/Ifocop/i);
console.log(n);
```

LES OBJETS

COMME DANS LA LANGUE FRANÇAISE, NOUS AVONS LES NOMS (ASSIMILABLES AUX "CHOSES") ET LES VERBES (ASSIMILABLES AUX "ACTIONS").

AVEC LES OBJETS, NOUS POUVONS STOCKER **NOTRE INFORMATION** ET **LES FONCTIONS QUI UTILISENT CETTE INFORMATION EN MÊME TEMPS.**

LES **INFORMATIONS** CONTENUES DANS NOTRE OBJET SE NOMMENT DES **PROPRIÉTÉS**. ET LES **FONCTIONS** (DONC LES ACTIONS FAISABLES) CONTENUES DANS NOTRE OBJET SE NOMMENT DES **MÉTHODES**.

LES OBJETS

LES OBJETS SONT UNE **COMBINAISON DE CLÉ – VALEUR**

LES VALEURS DES OBJETS PEUVENT ÊTRE DE N'IMPORTE QUEL TYPE (STRING, NUMBER, ARRAY OU MÊME
OBJET)

LES OBJETS

```
var monObjet = {  
    name : 'pierre', // valeur de type string  
    age : 27, // valeur de type number  
    interets : ['jouer', 'rire', true, 25] // valeur de type objet  
};  
  
// IMPORTANT : le POINT VIRGULE à la fin du stockage, les DEUX  
// POINTS pour chaque rapport clé : valeur. La VIRGULE à la fin de chaque déclaration  
// de clé-valeur sauf pour la dernière déclaration.
```

RAPPEL SUR LES OBJETS

PROPRIÉTÉS

IHS PROPRIÉTÉS SONT IHS VAIS ASSOCIÉS À IIN ORIFT JAVASCRIPT

```
// Différentes façons d'accéder à un objet  
objectName.property // person.age  
objectName[ "property" ] // person[ "age" ]  
objectName[ expression ] // x = "age"; person[ x ]
```

```
var person = {  
    firstname: "John",  
    lastname: "Doe",  
    age: 50,  
    eyecolor: "blue"  
};  
  
console.log(person.firstname + " is " + person.age + " years old.");
```

RAPPEL SUR LES OBJETS

MÉTHODES

```
var person = {  
    firstName: "John",  
    lastName : "Doe",  
    id       : 5566,  
    fullName : function() {  
        return this.firstName + " " + this.lastName;  
    }  
}
```

DANS UNE DÉFINITION DE FONCTION, THIS FAIT RÉFÉRENCE <<AU PROPRIÉTAIRE >> DE LA FONCTION.
ICI, THIS EST L'OBJET <PERSONNE> QUI POSSÈDE LA FONCTION <FULLNAME>

NIVEAU AVANCÉ

ACCESSEURS

CONSTRUCTOR

PROTOTYPES

VOIR DIAPO SUR LES ACCESSEURS, LES CONSTRUCTEURS ET LES PROTOTYPES
(NIVEAU PLUS AVANCÉ)

LES OBJETS - DÉCLARATION ET REMPLISSAGE

```
var monObjet = {};// déclaration d'un objet vide
```

```
var monObjet = new Object(); // on crée l objet. ATTENTION au « O » majuscule  
monObjet["name"] = "Charlie"; // on remplit l objet.  
monObjet.name = "Charlie"; // autre façon de remplir l objet.
```

EXEMPLE

```
var contactRepertoire = {};// On crée l'objet  
  
contactRepertoire.nom = 'Pierre DUPONT'; // syntaxe clé-valeur. nom est la clé  
contactRepertoire.numero = '06 68 62 54 95';  
contactRepertoire.telephoner = function() {  
    document.write('Appel ' + this.nom + ' au ' + this.numero + '...');  
};  
  
// contactRepertoire est l'objet  
// nom est une propriété  
// numero est une propriété  
// telephoner est une méthode
```

IMPORTANT : LE MOT CLÉ « THIS » SIGNIFIE « LUI-MÊME » (RÉFÉRENCE À L'OBJET)

CIBLER UNE PROPRIÉTÉ OU UNE MÉTHODE

```
// affiche Appel Pierre DUPONT au 0668625495...
contactReperoire.telephoner() ;
// affiche Pierre
contactReperoire.nom ;
```

AUTRE FAÇON D'ÉCRIRE UN OBJET

```
var monObjet = new Object();
```

OBJET DANS L'OBJET

```
monObjet = {  
    prenom : 'Arnaud',  
    loisirs : {  
        sport : 'football',  
        cinema : 'the artist'  
    },  
    nom : 'DELACRE',  
};
```

“THIS”

```
var contactReperoire = {};
contactReperoire.nom = 'Pierre DUPONT';
contactReperoire.numero = '06 68 62 54 95';
contactReperoire.telephoner = function() {
    document.write('Appel ' + this.nom + ' au ' + this.numero + '...');
};
```

IMPORTANT : LE MOT CLÉ « THIS » SIGNIFIE « LUI-MÊME » (RÉFÉRENCE À L'OBJET)

JAVASCRIPT

- STRUCTURE ITÉRATIVE -

STRUCTURE ITÉRATIVE

NOUS APPELONS STRUCTURE ITÉRATIVE (OU RÉCURSIVE), LA STRUCTURE QUI PERMET D'EXÉCUTER PLUSIEURS FOIS LES MÊMES INSTRUCTIONS

ELLE PERMET DE FAIRE "EN BOUCLE" UN BLOC D'INSTRUCTIONS.

TANT QUE LA CONDITION SERA RESPECTÉE, LES INSTRUCTIONS DE CODE SE RÉPÉTERONT.

LA BOUCLE FOR

DÉPART - CONDITION - INCRÉMENTATION

LA BOUCLE FOR

```
for(var monChiffre = 1 ; monChiffre < 100 ; monChiffre++){
    document.write(monChiffre) ;
}
```

PARCOURIR UN TABLEAU AVEC LA BOUCLE FOR

```
var txt = "";
var numbers = [45, 4, 9, 16, 25];
numbers.forEach(myFunction);
document.getElementById("demo").innerHTML = txt;

function myFunction(value, index, array) {
    console.log("value");
    console.log(value);
    console.log("index");
    console.log(index);
    console.log("array");
    console.log(array);
    txt = txt + value + "<br>";
}
```

PARCOURIR UN OBJET AVEC LA BOUCLE FOR

```
var personne = {prenom:"John", nom:"Doe", age:25};  
  
var text = "";  
var x;  
for (x in personne) {  
    console.log(personne);  
}
```

LA BOUCLE WHILE

LA BOUCLE WHILE EST UNE AUTRE MANIÈRE D'ÉCRIRE UNE BOUCLE.

LA BOUCLE WHILE

```
var monChiffre = 1; // déclaration et affectation de ma variable
// je pose ma condition ici, qui sera vérifiée à
// chaque tour de boucle (tant que mon chiffre est inférieur à 100)
while(monChiffre < 100){
    // mes instructions à l'intérieur des accolades (écris ce chiffre)
    document.write(monChiffre);
    monChiffre++; // ajoute lui 1
}
```

LA BOUCLE DO WHILE

1. EXÉCUTERA D'ABORD UNE PREMIÈRE FOIS LE CODE SANS TESTER LA CONDITION

2. PUIS ENSUITE, N'EXÉCUTERA LE CODE QUE SI LA CONDITION EST REMPLIE POUR TOUTES LES FOIS SUIVANTES.

LA BOUCLE DO WHILE

```
var maCondition = false;  
do{  
    document.write('ce code ne s\'exécutera qu\'une seule fois  
    parce-que je ne rempli pas la condition...');  
} while(maCondition == true);  
// Remarque : ici je mets un point virgule car j'écris  
// ce code après la fermeture d'accolade.
```

BREAK

```
for (i = 0; i < 10; i++) {  
    if (i === 3) {  
        break;  
    }  
    text += "Le chiffre de l'itération est " + i + "<br>"  
}
```

CONTINUE

```
for (i = 0; i < 10; i++) {  
    if (i === 3) {  
        continue;  
    }  
    text += "Le chiffre de l'itération est " + i + "<br>"  
}
```

JAVASCRIPT

CONDITIONS ET COMPARAISONS

LES STRUCTURES CONDITIONNELLES

NOUS APPELONS STRUCTURE CONDITIONNELLE, LA STRUCTURE QUI PERMET DE RÉALISER UNE (OU PLUSIEURS) INSTRUCTION(S) SOUS CERTAINES CONDITIONS.

ELLE NÉCESSITE L'UTILISATION DES OPÉRATEURS DE COMPARAISONS ET PARFOIS DES OPÉRATEURS LOGIQUES.

IF ELSE ELSE IF

NOUS APPELONS STRUCTURE CONDITIONNELLE, LA STRUCTURE QUI PERMET DE RÉALISER UNE (OU PLUSIEURS) INSTRUCTION(S) SOUS CERTAINES CONDITIONS.

OPÉRATEURS DE COMPARAISONS & OPÉRATEURS LOGIQUES

OPÉRATEURS DE COMPARAISON

| Opérateur | Description | Exemples qui renvoient true |
|-------------------------------|---|---|
| Égalité (==) | Renvoie true si les opérandes sont égaux après conversion en valeurs de mêmes types. | <code>3 == var1</code> <code>"3" == var1</code> <code>3 == '3'</code> |
| Inégalité (!=) | Renvoie true si les opérandes sont différents. | <code>var1 != 4</code> <code>var2 != "3"</code> |
| Égalité stricte (===) | Renvoie true si les opérandes sont égaux et de même type. Voir Object.is() et égalité de type en JavaScript . | <code>3 === var1</code> |
| Inégalité stricte (!==) | Renvoie true si les opérandes ne sont pas égaux ou s'ils ne sont pas de même type. | <code>var1 !== "3"</code> <code>3 !== '3'</code> |
| Supériorité stricte (>) | Renvoie true si l'opérande gauche est supérieur (strictement) à l'opérande droit. | <code>var2 > var1</code> <code>"12" > 2</code> |
| Supériorité ou égalité (>=) | Renvoie true si l'opérande gauche est supérieur ou égal à l'opérande droit. | <code>var2 >= var1</code> <code>var1 >= 3</code> |
| Infériorité stricte (<) | Renvoie true si l'opérande gauche est inférieur (strictement) à l'opérande droit. | <code>var1 < var2</code> <code>"2" < "12"</code> |
| Infériorité ou égalité (<=) | Renvoie true si l'opérande gauche est inférieur ou égal à l'opérande droit. | <code>var1 <= var2</code> <code>var2 <= 5</code> |

OPÉRATEURS LOGIQUES

| Opérateur (nom) | Opérateur (symbole) | Description |
|--------------------|------------------------|---|
| AND (ET) | && | Lorsqu'il est utilisé avec des valeurs booléennes, renvoie <code>true</code> si toutes les comparaisons sont évaluées à <code>true</code> ou <code>false</code> sinon |
| OR (OU) | | Lorsqu'il est utilisé avec des valeurs booléennes, renvoie <code>true</code> si au moins l'une des comparaisons est évaluée à <code>true</code> ou <code>false</code> sinon |
| NO (NON) | ! | Renvoie <code>false</code> si une comparaison est évaluée à <code>true</code> ou renvoie <code>true</code> dans le cas contraire |

SYNTAXE DES CONDITIONS DE CHOIX SWITCH

```
var couleur = prompt("quelle est ta couleur?");
switch(couleur){
    case "jaune":
        alert("tu aimes le jaune");
        break;
    case "bleu":
        alert("tu aimes le jaune");
        break;
    case "rouge":
        alert("tu aimes le jaune");
        break;
    case "vert":
        alert("tu aimes le jaune");
        break;
    default:
        alert("tas couleur n'est pas dans la liste");
        break;
}
```

CONDITION TERNAIRE

```
// Condition ternaire.  
var voiture = 'bm';  
document.write((voiture == 'bm') ? 'jolie<br>' : 'muche<br>');
```

JAVASCRIPT

FONCTIONS ET CLASSES

FONCTIONS

A L'INVERSE ET EN COMPLÉMENT, UNE FONCTION UTILISATEUR EST UNE FONCTION ÉCRITE PAR LE DÉVELOPPEUR

L'INTÉRÊT DES FONCTIONS EST DE PERMETTRE LA RÉUTILISATION À L'ENVI D'UN ENSEMBLE D'INSTRUCTIONS.

FONCTIONS PRÉDÉFINIES VS FONCTIONS UTILISATEURS

FONCTIONS PRÉDÉFINIES

```
var maVariable = 15;  
typeof(maVariable);  
isNaN(maVariable);
```

```
maVariable = 'Hamburger';  
maVariable.substring(3,9);
```

LES FONCTIONS UTILISATEUR

```
// sans argument
var coucou = function(){
    alert('salut') ;
};

// Executer cette fonction
coucou();

// Avec argument
var carre = function(nombre){
    return nombre * nombre ;
};

// Executer cette fonction
carre(5);

// Avec plusieurs arguments
var calculTva = function(montant, taux){
    return montant * taux;
};

// Exécuter la fonction
calculTva(10, 1.2);
```

DÉCLARATION

INVOCATION

PARAMÈTRE

PARAMÈTRES

LES FONCTIONS UTILISATEUR

```
// Autre façon de déclarer une fonction  
  
function carre(nombre){  
    return nombre * nombre;  
}
```

FUNCTION SCOPE

SANS LE MOT CLÉ VAR, LA VARIABLE EST ACCESSIBLE DE PARTOUT DANS LE SCRIPT = **GLOBALE**

AVEC LE MOT CLÉ VAR, LA PORTÉE DE LA FONCTION DÉPEND DE L'ENDROIT OÙ ELLE EST DÉCLARÉE:

- EN DEHORS D'UNE FONCTION = **GLOBALE**
- À L'INTÉRIEUR D'UNE FONCTION = **LOCALE**

FUNCTION SCOPE

```
// La variable est globale
var a = 12;

function MultipliePar2(b) {
    // La variable est locale
    var a = b * 2;
    return a;
}

document.write("Le double de 4 est ",MultipliePar2(4));
document.write('<br />');
document.write("La valeur de a est ",a);
```

MOTS CLEFS “THIS”

LE MOT-CLÉ THIS, FONCTIONNE À LA MANIÈRE D’UN PLACEHOLDER PERMETTANT D’INDIQUER À NOTRE MÉTHODE QUELLE DEVRA FONCTIONNER POUR TOUS LES OBJETS FAISANT APPEL À ELLE.
EN CLAIR, THIS EST UNE SORTE D’OBJET TÉMOIN.

MOTS CLEFS “THIS”

```
var vieillir = function(nouvelAge) {
    this.age = nouvelAge; // ici THIS va servir pour tous les objets qui vont
    //utiliser la méthode « vieillir »
}; // REMARQUE IMPORTANTE : Nous avons crée notre méthode en DEHORS de tout
// objet.

var pierre = new Object(); // Maintenant nous créons notre objet
pierre.age = 30; // Nous lui donnons un âge initial
pierre.changeAge = vieillir; // nous donnons à notre objet la méthode qui
//permet de changer d'âge.

pierre.changeAge(50); // en appelant la propriété changeAge, celle-ci appelle
//la méthode vieillir, qui s'occupe de faire le changement pour l'objet concerné
// grâce à THIS.
```

CALL()

```
var person = {  
    fullName: function() {  
        return this.firstName + " " + this.lastName;  
    }  
}  
var person1 = {  
    firstName: "John",  
    lastName: "Doe"  
}  
var person2 = {  
    firstName: "Mary",  
    lastName: "Doe"  
}  
  
person.fullName.call(person1);
```

AVEC LA MÉTHODE CALL() VOUS POUVEZ CRÉER UNE MÉTHODE QUI PEUT ÊTRE APPELÉ SUR DIFFÉRENTS OBJETS

ERRORS

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

<script>
  try {
    adddalert("Welcome guest!");
  }
  catch(err) {
    document.getElementById("demo").innerHTML = err.message;
  }
```

UNDEFINED

LA PROPRIÉTÉ GLOBALE `UNDEFINED` PRÉSENTE LA VALEUR `UNDEFINED`. CETTE VALEUR EST L'UN DES TYPES PRIMITIFS DE JAVASCRIPT.

JAVASCRIPT

MANIPULATION DU DOM

DOCUMENT OBJECT MODEL

ÉLÉMENT - ÉVÉNEMENT - ATTRIBUT

EST

RÉCUPÉRABLE - MANIPULABLE - EXPLOITABLE

COMMENT?

DOCUMENT HTML = OBJET MANIPULABLE EN JS

DOM

RÉCUPÉRER UN ÉLÉMENT HTML (LE SÉLECTIONNER)

LES SÉLECTEURS HTML

DOCUMENT.GETELEMENTBYID() ; // PAR SON ID

DOCUMENT.GETELEMENTBYCLASSNAME ; // PAR SA CLASSE

DOCUMENT.GETELEMENTSBYNAME() ; // PAR SON ATTRIBUT « NAME »

DOCUMENT.GETELEMENTSBYTAGNAME() ; // PAR SON NOM DE BALISE HTML

AUTRES SÉLECTEURS

QUERYSELECTOR()

RETOURNE LE PREMIER ÉLÉMENT TROUVÉ SATISFAISANT AU SÉLECTEUR (TYPE DE RETOUR : ELEMENT), OU NULL SI AUCUN OBJET CORRESPONDANT N'EST TROUVÉ.

QUERYSELECTORALL()

RETOURNE TOUS LES ÉLÉMENTS SATISFAISANT AU SÉLECTEUR, DANS L'ORDRE DANS LEQUEL ILS APPARAISSENT DANS L'ARBRE DU DOCUMENT (TYPE DE RETOUR : Nodelist), OU UN TABLEAU Nodelist VIDE SI RIEN N'EST TROUVÉ.

DOM

MANIPULER UN ÉLÉMENT HTML

MANIPULATION D'UN ÉLÉMENT

LES **MÉTHODES** HTML DOM SONT LES **ACTIONS** QUE VOUS POUVEZ EFFECTUER (SUR DES ÉLÉMENTS HTML).

LES **PROPRIÉTÉS** HTML DOM SONT DES **VALEURS** (DES ÉLÉMENTS HTML) QUE VOUS POUVEZ DÉFINIR OU MODIFIER.

CHANGEMENTS D'ÉLÉMENTS HTML

| Property | Description |
|---|---|
| <code>element.innerHTML = new html content</code> | Change the inner HTML of an element |
| <code>element.attribute = new value</code> | Change the attribute value of an HTML element |
| <code>element.style.property = new style</code> | Change the style of an HTML element |
| Method | Description |
| <code>element.setAttribute(attribute, value)</code> | Change the attribute value of an HTML element |

AJOUT ET SUPPRESSION DES ÉLÉMENTS

| Method | Description |
|--|-----------------------------------|
| <code>document.createElement(element)</code> | Create an HTML element |
| <code>document.removeChild(element)</code> | Remove an HTML element |
| <code>document.appendChild(element)</code> | Add an HTML element |
| <code>document.replaceChild(new, old)</code> | Replace an HTML element |
| <code>document.write(text)</code> | Write into the HTML output stream |

LES OBJETS HTML

| Property | Description | DOM |
|------------------------------|--|-----|
| document.anchors | Returns all <a> elements that have a name attribute | 1 |
| document.applets | Returns all <applet> elements (Deprecated in HTML5) | 1 |
| document.baseURI | Returns the absolute base URI of the document | 3 |
| document.body | Returns the <body> element | 1 |
| document.cookie | Returns the document's cookie | 1 |
| document.doctype | Returns the document's doctype | 3 |
| document.documentElement | Returns the <html> element | 3 |
| document.documentElementMode | Returns the mode used by the browser | 3 |
| document.documentElementURI | Returns the URI of the document | 3 |
| document.domain | Returns the domain name of the document server | 1 |
| document.domConfig | Obsolete. Returns the DOM configuration | 3 |
| document.embeds | Returns all <embed> elements | 3 |

LES OBJETS HTML

| | | |
|------------------------------|--|---|
| document.forms | Returns all <form> elements | 1 |
| document.head | Returns the <head> element | 3 |
| document.images | Returns all elements | 1 |
| document.implementation | Returns the DOM implementation | 3 |
| document.inputEncoding | Returns the document's encoding (character set) | 3 |
| document.lastModified | Returns the date and time the document was updated | 3 |
| document.links | Returns all <area> and <a> elements that have a href attribute | 1 |
| document.readyState | Returns the (loading) status of the document | 3 |
| document.referrer | Returns the URI of the referrer (the linking document) | 1 |
| document.scripts | Returns all <script> elements | 3 |
| document.strictErrorChecking | Returns if error checking is enforced | 3 |
| document.title | Returns the <title> element | 1 |
| document.URL | Returns the complete URL of the document | 1 |

MODIFICATION D'UN ÉLÉMENT

```
// selectionne toutes les divs
var mesDivs = document.getElementsByTagName("div");

// supprime leur attribut « style »
mesDivs.removeAttribute("style");
```

DOM

EXPLOITER UN ÉLÉMENT HTML

LES ÉVÉNEMENTS

LES ÉVÉNEMENTS SONT DES ACTIONS DE L'UTILISATEUR, QUI VONT POUVOIR DONNER LIEU À UNE INTERACTIVITÉ.

LES ÉVÉNEMENTS

ONBLUR()
ONCHANGE()
ONCLICK()
ONFOCUS()
ONFOCUSOUT()
ONKEYPRESS()
ONKEYUP()
ONMOUSEDOWN()
ONMOUSEUP()
ONRESIZE()
ONSROLL()
EVENTPREVENTDEFAULT()

....

ÉVÉNEMENTS, FONCTIONS ET MÉTHODES

CE SONT LES **GESTIONNAIRES D'ÉVÉNEMENTS** QUI PERMETTENT D'ASSOCIER UNE ACTION À UN ÉVÉNEMENT.

```
onEvenement="Action_Javascript_ou_Fonction();"
```

ÉVÉNEMENTS, FONCTIONS ET MÉTHODES

```
<img src='une_image.jpg' onclick='nom_de_lafonction(parametre1,parametre2)' />
```

```
<img src='une_image.jpg' onclick='fonction1() ; fonction2() ; fonction3()' />
```

ÉVÉNEMENTS, FONCTIONS ET MÉTHODES

```
<!Doctype html> HTML mixed mode ▾
<html>
  <head>
  </head>
  <body>
    <script type="text/javascript">

      function resultat(){

    }

    </script>

    <form name="formulaire" action="" method="post" id="monForm">
      <label for="pseudo">Pseudo</label>
      <input type="text" name="pseudo" id="pseudo" placeholder="pseudo" />
      <br />
      <label for="mdp">Mdp</label>
      <input type="password" name="mdp" id="mdp" placeholder="mdp" />
      <br />
      <input type="submit" name="ok" value="ok" id="monBoutton" onclick="resultat()" />
    </form>
  </body>
</html>
```

LA FONCTION ANONYME

UNE FONCTION ANONYME EST UNE FONCTION SANS NOM QUE L'ON VAS ASSOCIÉ A UN ÉVÉNEMENT CRÉE DYNAMIQUEMENT ELLE EST EN TOUT POINT SIMILAIRE A UNE FONCTION CLASSIQUE, UNE DE CES UTILITÉ EST DE POUVOIR AJOUTER DES PARAMÈTRES A UN ÉVÉNEMENT CREE DYNAMIQUEMENT

```
document.getElementById('mon_element').onclick=function(param1,param2,event,this){  
    du code du code  
}
```

PLUSIEURS FAÇONS DE FAIRE

```
/////////// Evènement(s)
```

```
// Appel de l'évènement "click"
var button = document.getElementById("boutton");

button.onclick = function(){
    alert("hey man");
};
```

```
// Autre façon d'écrire le même code
document.getElementById("boutton2").onclick = function(){
    alert("yo body");
};
```

ÉVÉNEMENTS, FONCTIONS ET MÉTHODES

```
// Première façon de lier un événement à une action (fonction ou méthodes)
<input type="text" onblur="myFunction()">

// Deuxième façon de lier un événement à une action (fonction ou méthodes)

document.getElementById("fname").onblur = function() {myFunction()};

function myFunction() {
    alert("Input field lost focus.");
}

// Façon identique à la deuxième sans appeler d'autre fonction

document.getElementById("fname").onblur = function() {
    alert("Input field lost focus.");
};
```

LA FONCTION ANONYME

LA PLUS ANCIENNE:

DOCUMENT.GETELEMENTBYID('MON_ELEMENT').ONCLICK=NOM_DE_LA_FONCTION

LE PROBLÈME DE CETTE MÉTHODE EST QUE NE L'ON PEUT ADJOINDRE QU' UNE SEULE FONCTION POUR LE MEME
ÉVÉNEMENT

ASSOCIER DYNAMIQUEMENT UN ÉVÈNEMENT A UN ÉLÉMENT

“ADDEVENTLISTENER”

```
DOCUMENT.GETELEMENTBYID('MON_ELEMENT').ADDEVENTLISTENER("CLICK", NOM_DE_LA_FONCTION, FALSE);
```

REMOVEEVENTLISTENER

ADDEVENTLISTENER

```
<!Doctype html>
<html>
    <head>
    </head>
    <body>
        <script type="text/javascript">

            document.getElementById("div1").addEventListener("click", maFonction);

            function maFonction(){
                alert('maFonction - clic div1');
            }

        </script>

        <div id="div1"> Div1 </div>

    </body>
</html>
```

Attention, Ici le script doit être chargé
après le DOM pour fonctionner.

SETATTRIBUTE

“SETATTRIBUTE”

DOCUMENT.GETELEMENTBYID('MON_ELEMENT').SETATTRIBUTE('ONCLICK','NOM_DE
LAFONCTION('PARAMÈTRE',EVENT));

REMOVEATTRIBUTE

FONCTION ANONYME VS ADDEVENTLISTENER

```
// Script d'initialisation  
// Ici seul la fonction 2 sera appelée  
document.onload=fonction_1  
document.onload=fonction_2  
  
// Ici les deux fonctions seront appelées au démarrage de la page  
document.addEventListener("load", fonction_1, false);  
document.addEventListener("load", fonction_2, false);
```

LES PARAMÈTRES DE FONCTION DANS UN ÉVÈNEMENT

LES PARAMÈTRES SONT MIS A L'INTÉRIEUR DES PARENTHÈSES DE LA FONCTION ET CHAQUE PARAMÈTRE EST SÉPARÉ PAR UNE VIRGULE.

```
fonction la_fonction(param_1,param_2,param_3)
```

LE PARAMÈTRE THIS

POINTE L'ÉLÉMENT D'OÙ PROVIENT L'ÉVÈNEMENT IL EVITERA DE METTRE UN ID A L'ELEMENT ET DONC D'UTILISER DOCUMENT.GETELEMENTBYID().

```
<div onclick='couleur(this)'  
      style='height:50px;width:50px;background-color:green'>clic moi  
</div>
```

LE PARAMÈTRE EVENT

LE PARAMÈTRE EVENT FAIT PARTIE DE LA GESTION GLOBALE DES ÉVÉNEMENTS PAR LE NAVIGATEUR IL EST CONSIDÉRÉ COMME UN OBJET ET POSSÈDE DONC DES MÉTHODES ET PROPRIÉTÉ.

```
<div onmousedown='lien_a(event)' id='div_rouge'>texte texte texte texte </div>

function lien_a(evt){

    evt.stopPropagation();
    alert(evt.currentTarget.id);
}
```

MÉTHODES EVENT

| Méthodes | Description |
|--------------------------|--|
| preventDefault() | empêche l'action par défaut de se produire pas |
| stopPropagation() | limite l'évènement à l'Élément cible afin d'éviter la propagation aux parents afin d'éviter pour chaque parent possédant le même évènement le déclenchement de la fonction associée. |

ADDEVENTLISTENER & EVENT

SI ON VEUT UTILISER UNIQUEMENT LE PARAMÈTRE EVENT ON A PAS BESOIN DE PASSER PAR UNE FONCTION ANONYME CAR IL EST IMPLICITEMENT AJOUTÉ

ADDEVENTLISTENER & EVENT

```
<!Doctype html>
<html>
  <head>
  </head>
  <body>
    <script type="text/javascript">

      document.getElementById("monBoutton").addEventListener("click", maFonction);

      function maFonction(event){
        console.log(event);
        event.preventDefault();
      }

    </script>
    <form name="formulaire" action="" method="post" id="monForm">
      <label for="pseudo">Pseudo</label>
      <input type="text" name="pseudo" id="pseudo" placeholder="pseudo" />
      <br />
      <label for="mdp">Mdp</label>
      <input type="password" name="mdp" id="mdp" placeholder="mdp" />
      <br />
      <input type="submit" name="ok" value="ok" id="monBoutton" />
    </form>
  </body>
</html>
```

HTML mixed mode ▾

Attention, Ici le script doit être chargé après le DOM pour fonctionner.

EN RÉSUMÉ

```
document.getElementById("div1").onclick = div1; // <== evenement sur l'objet,  
//function déclarée plus loin.
```

```
document.getElementById("div1").onblur=function(){myScript}; // <== evenement sur  
l'objet, function déclarée ici.
```

```
document.getElementById("div1").addEventListener("blur", myScript); // <== evenement  
ajouté sur l'objet via addEventListener, function déclarée plus loin.
```

```
document.getElementById("div1").addEventListener("submit", function(event){  
event.preventDefault(); alert('div14 - submit! ')}); // <== evenement ajouté sur  
l'objet via addEventListener, function déclarée ici.
```

```
// <div onBlur="myScript">...</div> <== méthode initiale/ancienne.
```

EN RÉSUMÉ

```
// ----- CHARGEMENT DE LA PAGE (plusieurs possibilité d'écriture)
window.addEventListener("load", page);

window.onload = function(){};

document.addEventListener("DOMContentLoaded", function(event) {
    alert('dom loadé');
});
```

ÉVÈNEMENT DE LA SOURIS ET DU CLAVIER

| propriete | description |
|----------------------|--|
| type | retourne le type de l'évènement (onclick; onmouseover...etc) |
| currentTarget | retourne l'Élément qui a déclenché l'évènement |
| target | retourne l'Élément survolé dont le parent est l'Élément qui a déclenché l'évènement. |
| button | Retours le bouton de la souris qui a été cliqué (0:gauche; 1:centre; 2:droite) |
| clientX | Retour de la coordonnée horizontale du pointeur de la souris, par rapport à la fenêtre du navigateur. |
| clientY | Retour de la coordonnée verticale du pointeur de la souris, par rapport à la fenêtre du navigateur. |
| pageX | Retour de la coordonnée horizontale du pointeur de la souris, par rapport à la fenêtre du navigateur plus la position de la barre de scroll. |

ÉVÈNEMENT DE LA SOURIS ET DU CLAVIER

| | |
|----------------------|--|
| pageY | Retour de la coordonnée verticale du pointeur de la souris, par rapport à la fenêtre du navigateur plus la position de la barre de scroll. |
| screenX | Retour de la coordonnée horizontale du pointeur de la souris, par rapport à l'écran |
| screenY | Retour de la coordonnée verticale du pointeur de la souris, par rapport à l'écran |
| charCode | Récupère le code de caractère Unicode de la touche qui a généré l'événement onkeypress. |
| altKey | Retours si oui ou non la touche "ALT" a été pressé quand un événement a été déclenché |
| shiftKey | Retours si oui ou non la touche "SHIFT" a été pressé quand un événement a été déclenché |
| ctrlKey | Retours si oui ou non la touche "CTRL" a été pressé quand un événement a été déclenché |
| metaKey | Retours si oui ou non la touche "meta" a été pressé quand un événement a été déclenché |
| relatedTarget | Retourne l'élément lié à l'élément qui a déclenché l'événement |

CSS

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

    <p id="p1">Hello World!</p>
    <p id="p2">Hello World!</p>

    <script>
      document.getElementById("p2").style.color = "blue";
      document.getElementById("p2").style.fontFamily = "Arial";
      document.getElementById("p2").style.fontSize = "larger";
    </script>

    <p>The paragraph above was changed by a script.</p>

  </body>
</html>
```

ANIMATIONS

```
<script>
    function myMove() {
        var elem = document.getElementById("animate");
        var pos = 0;
        var id = setInterval(frame, 5);
        function frame() {
            if (pos == 350) {
                clearInterval(id);
            } else {
                pos++;
                elem.style.top = pos + "px";
                elem.style.left = pos + "px";
            }
        }
    }
</script>
```

COLLECTIONS

LA MÉTHODE **GETELEMENTSBYTAGNAME()** ET **GETELEMENTBYCLASSNAME()** RETOURNE UN OBJET DE COLLECTION HTML (“HTMLCOLLECTION”) :

C’EST UNE SORTE DE TABLEAU CONTENANT DES ÉLÉMENTS HTML.

LA PROPRIÉTÉ “LENGTH” DÉFINIE LE NOMBRE D’ÉLÉMENTS DANS UNE COLLECTION D’OBJET HTML (“HTMLCOLLECTION”)

COLLECTIONS

```
▼ HTMLCollection(4) ⓘ                                         app.js:15
  ► 0: span
  ► 1: span
  ► 2: span
  ► 3: span
  ► __proto__: HTMLCollection

▼ HTMLCollection(4) ⓘ
  length: 4
  ▼ 0: span
    title: ""
    lang: ""
    translate: true
    dir: ""
    hidden: false
    accessKey: ""
    draggable: false
    spellcheck: true
    autocapitalize: ""
    contentEditable: "inherit"
    isContentEditable: false
    inputMode: ""
  ► offsetParent: body
  offsetTop: 320
  offsetLeft: 8
  offsetWidth: 33
  offsetHeight: 18
  ► style: CSSStyleDeclaration {alignContent: "", alignItems: "", alignSelf: "", alignmentBaseline: "", all: "", ...}
  innerText: "span"
```

LISTE DES ÉVÉNEMENTS

| Événement | Description |
|-----------------------------|---|
| Abort (onAbort) | Cet événement a lieu lorsque l'utilisateur interrompt le chargement de l'image |
| Blur (onBlur) | Se produit lorsque l'élément perd le focus, c'est-à-dire que l'utilisateur clique hors de cet élément, celui-ci n'est alors plus sélectionné comme étant l'élément actif. |
| Change (onChange) | Se produit lorsque l'utilisateur modifie le contenu d'un champ de données. |
| Click (onClick) | Se produit lorsque l'utilisateur clique sur l'élément associé à l'événement. |

LISTE DES ÉVÉNEMENTS

| | |
|-----------------------------------|--|
| dblclick (onDbclick) | Se produit lorsque l'utilisateur double-clique sur l'élément associé à l'événement (un lien hypertexte ou un élément de formulaire). |
| dragdrop (onDragdrop) | Se produit lorsque l'utilisateur effectue un <i>glisser-déposer</i> sur la fenêtre du navigateur. |
| error (onError) | Se déclenche lorsqu'une erreur apparaît durant le chargement de la page. |
| Focus (onFocus) | Se produit lorsque l'utilisateur donne le focus à un élément, c'est-à-dire que cet élément est sélectionné comme étant l'élément actif |
| keydown (onKeydown) | Se produit lorsque l'utilisateur appuie sur une touche de son clavier. |
| keypress (onKeyPress) | Se produit lorsque l'utilisateur maintient une touche de son clavier enfoncée. |
| keyup (onKeyUp) | Se produit lorsque l'utilisateur relâche une touche de son clavier préalablement enfoncée. |
| Load (onLoad) | Se produit lorsque le navigateur de l'utilisateur charge la page en cours |
| MouseOver (onMouseOver) | Se produit lorsque l'utilisateur positionne le curseur de la souris au-dessus d'un élément |

LISTE DES ÉVÉNEMENTS

| | |
|---------------------------------|--|
| MouseOut (onMouseOut) | Se produit lorsque le curseur de la souris quitte un élément. |
| Reset (onReset) | Se produit lorsque l'utilisateur efface les données d'un formulaire à l'aide du bouton Reset. |
| Resize (onResize) | Se produit lorsque l'utilisateur redimensionne la fenêtre du navigateur |
| Select (onSelect) | Se produit lorsque l'utilisateur sélectionne un texte (ou une partie d'un texte) dans un champ de type "text" ou "textarea" |
| Submit (onSubmit) | Se produit lorsque l'utilisateur clique sur le bouton de soumission d'un formulaire (le bouton qui permet d'envoyer le formulaire) |
| Unload (onUnload) | Se produit lorsque le navigateur de l'utilisateur quitte la page en cours |

LISTE DES ÉVÉNEMENTS ASSOCIÉS À LEURS OBJETS

| | |
|-----------|---|
| abort | Image |
| blur | Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, window |
| change | FileUpload, Select, Submit, Text, TextArea |
| click | Button, document, Checkbox, Link, Radio, Reset, Select, Submit |
| dblclick | document, Link |
| dragdrop | window |
| error | Image, window |
| focus | Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, window |
| keydown | document, Image, Link, TextArea |
| keypress | document, Image, Link, TextArea |
| keyup | document, Image, Link, TextArea |
| load | Image, Layer, window |
| mousedown | Button, document, Link |
| mousemove | Aucun spécifiquement |
| mouseout | Layer, Link |
| mouseover | Area, Layer, Link |
| mouseup | Button, document, Link |
| move | window |
| reset | form |
| resize | window |
| select | text, Textarea |
| submit | Form |
| unload | window |

LES COOKIES

LES COOKIES SONT DES DONNÉES CONTENUS DANS DES PETITS FICHIER TEXT SUR VOTRE ORDINATEUR
LES COOKIES ONT ÉTÉ CRÉÉ POUR NE PAS "OUBLIER LES INFORMATIONS SUR L'UTILISATEUR"

IL SONT UNE ASSOCIATION DE CLEF - VALEUR :

USERNAME = SAMIH HABBANI

LES COOKIES

NOUS POUVONS CRÉER, LIRE ET SUPPRIMER DES COOKIES AVEC LE LA PROPRIÉTÉ DOCUMENT.COOKIE

```
document.cookie = "username=John Doe";
```

LES COOKIES

VOUS POUVEZ AJOUTER UNE DATE D'EXPIRATION À VOTRE COOKIE :

```
document.cookie = "username=Samih Habbani; expires=Thu, 18 Dec 2013 12:00:00 UTC";
```

LES COOKIES

VOUS POUVEZ ÉGALEMENT PRÉCISER À QUEL CHEMIN LE COOKIE CORRESPOND (PAR DÉFAUT À LA PREMIÈRE PAGE)

```
document.cookie = "username=Samih Habbani; expires=Thu, 18 Dec 2013 12:00:00 UTC;  
path=/";
```

LES COOKIES

```
// Pour lire les cookies
var x = document.cookie;

// Les cookies sont modifiables de la même façon que vous les avez créé
document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";

// Pour supprimer un cookie il suffit de mettre une date d'expiration passée
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";
```

LES COOKIES

SI VOUS CRÉER UN NOUVEAU COOKIE APRÈS EN AVOIR DÉJÀ CRÉÉ UN, L'ANCIEN COOKIE NE SERA PAS ÉCRASÉ ET LES DEUX SERONT MANIPULABLES. IL SUFFIRA SIMPLEMENT DE TROUVER DANS LA STRING LA CLÉ - VALEUR QUI CORRESPONDRA À VOTRE COOKIE.

LES COOKIES

HTML mixed mode ▾

```
// Fonction pour créer un cookie
function setCookie(cname, cvalue, exdays) {
    var d = new Date();
    d.setTime(d.getTime() + (exdays*24*60*60*1000));
    var expires = "expires=" + d.toUTCString();
    document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}

// Fonction pour en récupérer un
function getCookie(cname) {
    var name = cname + "=";
    var decodedCookie = decodeURIComponent(document.cookie);
    var ca = decodedCookie.split(';');
    for(var i = 0; i <ca.length; i++) {
        var c = ca[i];
        while (c.charAt(0) == ' ') {
            c = c.substring(1);
        }
        if (c.indexOf(name) == 0) {
            return c.substring(name.length, c.length);
        }
    }
    return "";
}
```

LES COOKIES

```
function checkCookie() {  
    var user = getCookie("username");  
    if (user != "") {  
        alert("Welcome again " + user);  
    } else {  
        user = prompt("Please enter your name:", "");  
        if (user != "" && user != null) {  
            setCookie("username", user, 365);  
        }  
    }  
}
```

SESSIONSTORAGE & LOCALSTORAGE

SONT DES PROPRIÉTÉS QUI PERMETTENT DE GARER UN ENSEMBLE DE CLEF-VALEUR DANS LE NAVIGATEUR

SESSIONSTORAGE & LOCALSTORAGE

```
// Stocker
sessionStorage.setItem("nom", "Habbani");
// Récupérer
document.getElementById("result").innerHTML = sessionStorage.getItem("nom");
```

SESSIONSTORAGE & LOCALSTORAGE

LA SESSIONSTORAGE STOCK UN ENSEMBLE DE DONNÉES POUR LE NAVIGATEUR POUR UNE SEULE SESSION SEULEMENT. LORSQUE LE NAVIGATEUR FERME, LES DONNÉES SONT PERDUES.

SESSIONSTORAGE & LOCALSTORAGE

```
function clickCounter() {
    if(typeof(Storage) !== "undefined") {
        if (sessionStorage.clickcount) {
            sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;
        } else {
            sessionStorage.clickcount = 1;
        }
        document.getElementById("result").innerHTML = "You have clicked the
button " + sessionStorage.clickcount + " time(s) in this session.";
    } else {
        document.getElementById("result").innerHTML = "Sorry, your browser does
not support web storage...";
    }
}
</script>
</head>
<body>
    <p><button onclick="clickCounter()" type="button">Click me!</button></p>
    <div id="result"></div>
    <p>Click the button to see the counter increase.</p>
    <p>Close the browser tab (or window), and try again, and the counter is reset.</p>
</body>
```

SESSIONSTORAGE & LOCALSTORAGE

```
// Stocker
localStorage.setItem("lastname", "Smith");
// Récupérer
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```

JS JSON = (J)AVA(S)CRIPT (O)BJECT (N)OTATION

LE FORMAT SON EST UN FORMAT POUR STOCKER ET TRANSPORTER DES DONNÉES

IL EST SOUVENT UTILISÉ QUAND DES DONNÉES SON ENVOYÉS DU SERVEUR VERS LA PAGE WEB

“TEXT-ONLY”

JS JSON = (J)AVA(S)CRIPT (O)BJECT (N)OTATION

```
{  
  "employees": [  
    {"firstName": "John", "lastName": "Doe"},  
    {"firstName": "Anna", "lastName": "Smith"},  
    {"firstName": "Peter", "lastName": "Jones"}  
  ]  
}
```

JS JSON = (J)AVA(S)CRIPT (O)BJECT (N)OTATION

SON FORMAT EST TRÈS SIMILAIRE AU FORMAT OBJECT EN JAVASCRIPT ET DONC FACILEMENT MANIPULABLE:

- LES DONNÉES SONT CLASSÉES EN CLEF-VALEUR
- LES DONNÉES SONT SÉPARÉES PAR DES VIRGULES
- LES ACCOLADES CONTIENNENT DES OBJETS
- LES CROCHETS CONTIENNENT DES ARRAYS

JS JSON = (J)AVA(S)CRIPT (O)BJECT (N)OTATION

`JSONPARSE()`

JS JSON = (J)AVA(S)CRIPT (O)BJECT (N)OTATION

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

    <h2>Create Object from JSON String</h2>

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

    <script>
      var text = '{"employees":[' +
        '{"firstName":"John","lastName":"Doe"},' +
        '{"firstName":"Anna","lastName":"Smith"},' +
        '{"firstName":"Peter","lastName":"Jones"}]';

      obj = JSON.parse(text);
      document.getElementById("demo").innerHTML =
        obj.employees[1].firstName + " " + obj.employees[1].lastName;
    </script>

  </body>
</html>
```

HTML mixed mode ▾

JS FORMS

```
function validateForm() {  
    var x = document.forms["myForm"]["fname"].value;  
    if (x == "") {  
        alert("Name must be filled out");  
        return false;  
    }  
}
```

JS FORMS

| Attribute | Description |
|------------------|---|
| disabled | Specifies that the input element should be disabled |
| max | Specifies the maximum value of an input element |
| min | Specifies the minimum value of an input element |
| pattern | Specifies the value pattern of an input element |
| required | Specifies that the input field requires an element |
| type | Specifies the type of an input element |

ALLER PLUS LOIN DANS L'OBJET

ACCESEURS D'OBJET
CONSTRUCTEURS
OBJET PROTOTYPES

BEST PRACTICES

[HTTPS://THEMEFOREST.NET/](https://themeforest.net/)

...

NIVEAU AVANCÉ

HÉRITAGE ET CHAÎNE DE PROTOTYPES

LE MODE STRICT

LES TABLEAUX TYPÉS EN JAVASCRIPT

LA GESTION DE LA MÉMOIRE EN JAVASCRIPT

GESTION DE LA CONCURRENCE ET BOUCLE DES ÉVÉNEMENTS

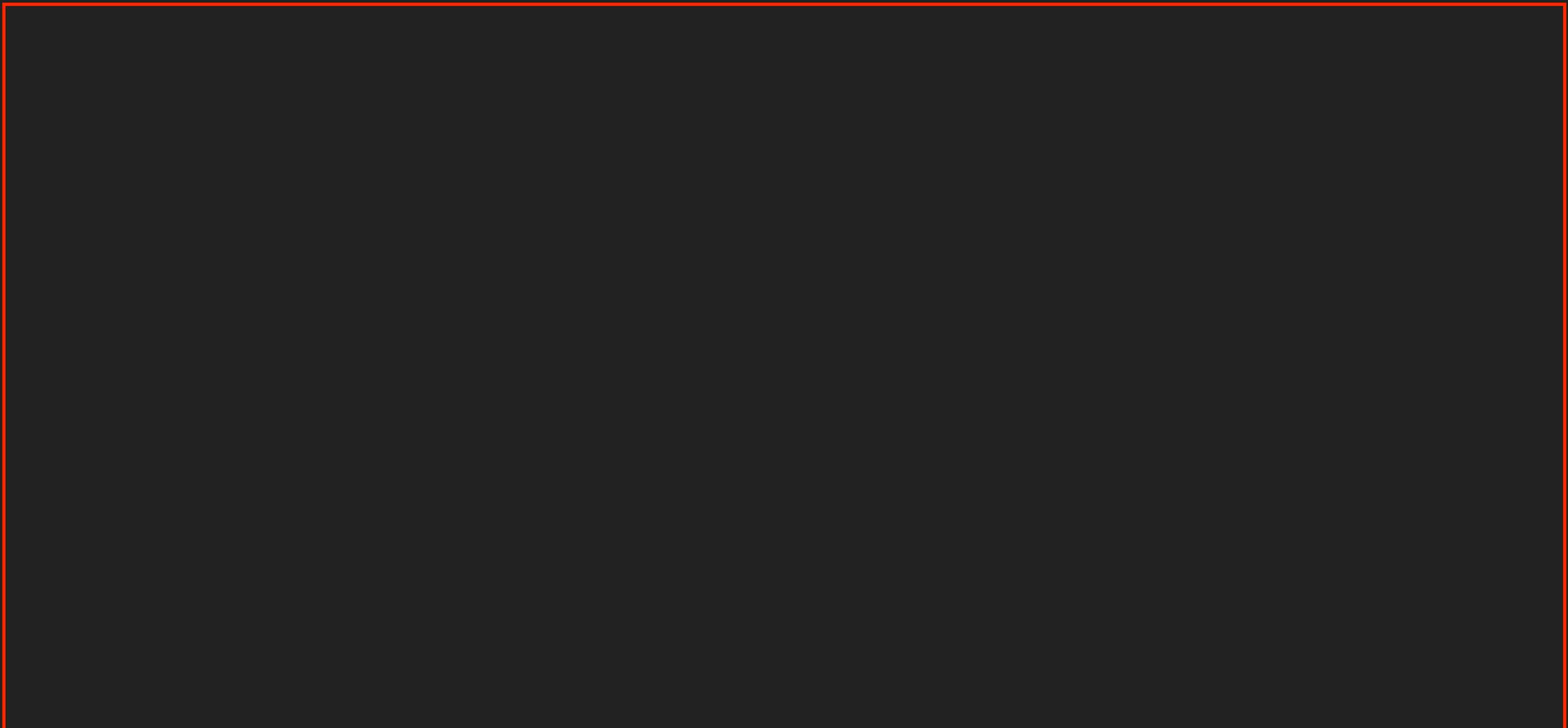
...

ETUDE DES DIFFÉRENTES LIBRAIRIES

[HTTPS://THEMEFOREST.NET/](https://themeforest.net/)

....

NOUVEAUTÉS JAVASCRIPT



JQUERY : WRITE LESS DO MORE

SÉLECTEURS ÉVÉNEMENTS FONCTIONS

JQUERY : WRITE LESS DO MORE

RÉSUMÉ DE TOUT CE QUE L'ON A FAIT

CE QUE L'ON A PAS EU LE TEMPS DE VOIR

LES CLASSES

NODES

COLLECTIONS

NODE LISTS

AJOUTER LES PARSIINT ETC...

REMETTRE TOUT EGO HTML CSS ET JS SUR RÉSEAU

CE QUE L'ON A PAS EU LE TEMPS DE VOIR

NODES
COLLECTIONS
NODE LISTS

LES CLASSES

REFAIRE DIAPO SUR PARAMÈTRES