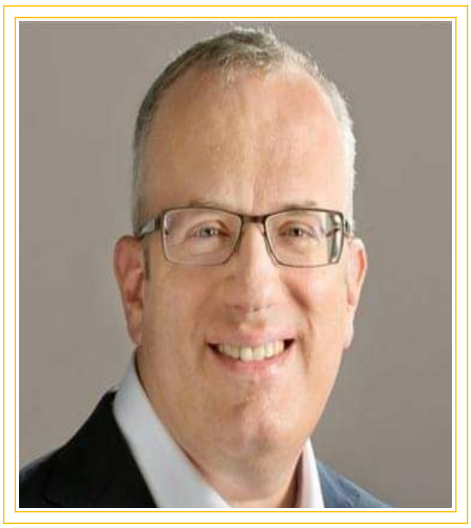
COURS JAVASCRIPT <intro/>

/*---------Présentation---------*/



- Créer en 1995 par Brendan Eich (Netscape/fondation Mozilla)
- Paradigmes = script, impératif, oriente-objet
- Langage interprété (exécuté ligne par ligne/non compilé)
- Norme ECMAscript (ES5: fonctionne sur toute les navigateurs)
- Sensible a la casse

/*---------Domaine d'utilisation---------*/

➔ Utilisation cote **client**

➔ Utilisation cote **serveur**

/*---------Tips---------*/

✗ UTF-8 (<head><meta charset="utf-8"></head>) est le **meilleur encodage** pour les langages de programmation

✗ F12 pour inspecter une page web

✗ Site de développement HTML/CSS/JS: www.jsfiddle.net

/*---------Structure de base---------*/

- Code JS intégré sur une page HTML

```
<! DOCTYPE>
<html>
    <head>
        <title> </title>
    </head>
<body>
    <script type="text/javascript">
        CODE JS;
    </script>
</body>
</html>
```

- Code JS importé

```
<! DOCTYPE>
<html>
    <head>
        <title> </title>
    </head>
<body>

<script src = "script.js"></script>
</body>
</html>
```

=> Dans l'entête de la page, a l'intérieur des balises HEAD : souvent utilise pour les fonctions dans la mesure ou le contenu de la section <head></head> est **exécuté en premier**.

=> A l'intérieur d'une balise HTML: on appelle cela un gestionnaire d'evenements, permet au script d'interagir avec des éléments HTML.

/*---------Variable---------*/

✔ mot clé VAR

Dans JS, il n'y a pas de typage de variable, il suffit juste le mot clé "var" pour déclarer une variable.

```
var myInt;
var myInt = 0; //déclaration et définition a 0
    myInt = 1; //réaffectation, pas très recommanda si vous voulez déclarer une variable locale
```

✔ mot clé LET

Si une fonction contient plusieurs **bloc de code**, on met le mot clé "let" si on veut qu'une variable n'appartient qu'a une de ce bloc

```
let myCar;
let myCar = "Bye !";
```

* **Portée d'une variable** (rappel) **:**

> Variables globales : accessible dans tout le script ou même dans d'autres scripts du même document HTML

> Variables locales : ne peuvent êtres employées qu'a l'intérieur de la fonction dans la-quelles elles ont été crées

/*---------Valeur spéciaux---------*/

null/undefined absence d'une valeur (vide)

NaN = **N**ot **a** **N**umber (pas un nombre)

/*---------Qualification---------*/

constante
const PI = 3.14;

/*---------Opérateur---------*/

=	: affectation
==	: égale (même type)
===	: identique (même symbole)
!=	: différent de
<	: inférieur a
>	: supérieur a
<=	: inférieur ou égale
>=	: supérieur ou égale
||	: OU logique
&&	: ET logique

/*---------Affichage---------*/

- **chaîne | entier** (affichage)

```
alert ("PUB !"); //un pop-up (message simple | publicité)
document.write ("JS est le langage du vrai dev :)"); //équivalent de print (écrit les chaînes directement sur la page web)
confirm ("confirmed ?") //message pour demander une confirmation
```

***N.B** : Utilise le symbole ` au lieu de '' afin que la fonction est compatible avec les balise HTML*

- **variable** (affichage)

```
var myVar = 2*7;

document.write (myVar); //output : 14
document.write(`Mon variable est : ${myVar}`) ; //output : Ma variable est : 14

alert ("Affichage variable = " + myVar); //on met juste le "+" pour concaténer une chaîne avec une variable (sa valeur)
```

- **Console** (affichage)

```
console.log ("Welcome into the console");
console.error ("ERROR test"); //Pour afficher une message d'erreur
```

/*---------Saisi---------*/

- **chaîne | entier** (saisi)

```
var saisi = prompt ("Entrez une valeur: ") ;
alert("saisi = " + saisi);
```

***N.B** : Pour les entiers, il faut convertissez en utilisant la fonction parseInt(<var>) ou parsefloat(<var>) afin d'être considérer comme une vrai entier (sinon, vous pouvez utiliser la méthode ci-dessous pour une saisi d'un entier).*

```
var saisi = number (prompt ("Entrez une valeur: ") );
```

/*---------Cast(conversion)---------*/

- **Vers integer**

```
var age = prompt("Quelle est votre age? ")
parseInt(age);
alert(typeof(age));
```

- **Vers float**

```
parseFloat(age);
```

/*---------Structure conditionnel---------*/

<var1> === <var2> : test rapide si <var1> est **identique** a <var2> | retourner true/false

<var1> == <var2> : test rapide si <var1> est **égale** (meme type) a <var2> | retourner true/false

Test imbriqué (**if**)

```
if (condition) {
        instruction;
    }
    else if (condition){
        instruction;
    }
    else{
        instruction;
    }
```

Test multiple (**switch**)

```
switch (condition){
            case condition :
            instruction;
            break;

            case condition :
            instruction;
            break;

            case condition :
            instruction;
            break;

            case condition :
            instruction;
            break;

        default :
            instruction;

    }
```

/*---------Boucle---------*/

While

```
while (condition){
        instruction;
        casseur de boucle;
    }
```

For

```
for (var i = <valeur>; i <comparaison> <valeur>; casseur de boucle) {
        instruction;
    }
```

continue : utilise a l'intérieur d'une boucle pour ignorer le reste de l'itération actuelle.

While (continue)

```
while (condition){

        if (condition) {i++; continue;}
        //Eviter une boucle infini

        instruction;
        i++;
    }
```

For (continue)

```
for (var i = <valeur>; i <comparaison> <valeur>; casseur de boucle) {

    if (i === <valeur>) {continue;}
    /*
    Si la condition est vraie,
    la boule va ignorer
    la valeur indiquEe y compris
    l'instruction qui le suit jusqu'a la fin de la boucle
    */

    instruction;
}
```

/*---------Tableau---------*/

index de départ = 0 (par convention)

- **tableau vide**

notation 1 var tab = [ ];
notation 2 var tab2 = Array();
notation 3 var tab3 = new Array(<taille>);

- **tableau avec valeur**

var tab = ["a", "rakoto", 10, true, 3.14, false];

- **tableau imbriqué** (2D)

var tab2D = ["a", "rakoto", ["b", "c", "d"], 10, true, 3.14, false];

- **affichage d'un tableau**

aff1 Simple: document.write ("tab[]"); //tous les éléments
aff2 Simple: document.write ("tab[3]"); //4e élément => true
aff3 2D: document.write ("tab2D[2][0]"); //1er élément du 2e tableau => b

- **affectation d'un tableau**

=> tab[4] = <valeur>; // affectation d'une nouvelle valeur au 4e élément

- **Type référencer**

Si tab0[ ] = tab1[ ]; // Valeur de tab0[] est toujours valeur de tab1[] car ils utilisent le meme pointeur

/*---------Fonction(prédéfini)---------*/

Color index
violet : variable
rouge : fonction
bleu : paramètre(s)/argument(s)

STRUCTURE (fonction)

```
function nameFunction(arg0, arg1, ... , argn){
    instruction;

    retrun <var>; // retrun [<var>, <var>];
}
//Appel: nameFunction();
```

CHAÎNE (fonction)
var.length : longueur d'une chaîne
var.toUpperCase() : Majuscule (valeur du var)
var.toLowerCase() : Minuscule (valeur du var)

var.indexOf("chaine") : position de <chaîne> par rapport a une chaîne de caractère

exemple de indexOf
{
var Alpha = "abcdefghijklmnopqrstuvwxyz";
Alpha.indexOf("y") => 24
}

NB: Si <chaîne> n'existe pas => position = -1

var.substring(pos.debut , pos.fin + 1) : extraction d'une chaîne de caractère

exemple de substr
{
var Alpha = "abcdefghijklmnopqrstuvwxyz";
Alpha.substring(0,4) => abcd
Alpha.substring(6,7) => g
}

var.charAt(pos) : extraction d'une seule caractère, si pos = 25 => z

ENTIER (fonction)

parseInt(var) : cast int, conversion en entier
parseFloat(var) : cast float, conversion en virgule flottante
isNaN(var) : is Not a Number, test si <var> n'est pas un nombre

MATH (fonction)

Math.round(var) : arrondissement (ex: Math.round(12.9) = 13)
Math.random() : valeur aléatoire

Exemple de fonction random (retourne une valeur aléatoire sur y)

```
{
    function randOm(nb){

    var x = Math.random();
    x = x * nb;
    var y = Math.floor(x) + 1;

    return y;
}
```

Math.pow(valeur, puissance) : puissance (ex: Math.pow(2, 8) = 256)
Math.max/min(valeur.min , ... , valeur.max)
Math.PI : valeur de PI (3.14)
Math.sqrt(var) : racine carrée de <var>

window.open() (POPUP)

window.open(page[.nom][.option])

- **page** : contient l'adresse de la page a afficher
- **nom** : contient le nom du popup qui va être ouvert (non obligatoire)
- **option** : une chaîne de caractères contenant les options de paramétrage du popup (non obligatoire)

exemple (simple)

```
{
    <script type="text/javascript">
        function pub(pageHtml){
            window.open(pageHtml);
        }
    </script>
    <body>
        <center>
            <a href="javascript:pub('index.html')">Cliquez ici pour continuer...</a>
        </center>
    </body>>
}
```

exemple (avec DOM)

```
{
        <script type="text/javascript">
            function pub2(pageHtml){
                elem = getElementById("buttonContinued");
                elem.window.open(pageHtml);
            }
        </script>
        <body>
            <center>
                <label for="popup"> Veuillez clickez sur le boutton pour continuer : </label>
                <button onclick="pub2('index.html')"> Continuer </button>
            </center>
        </body>>
    }
}
```

TABLEAU

```
//<var> => tab[]
//<valeur> => "chaîne" ou "123"
```

var.push(valeur) : ajouter un nouveau élément au tableau
var.pop() : retirer le dernier élément du tableau
delete var : retirer un élément du tableau (ex: delete tab[5])
var.sort() : trie alphabétique/ordre croissant des éléments du tableau
var.reverse() : trie inversé alphabétique/ordre croissant des éléments du tableau
var.join(",") : afficher tous les éléments en les séparant par
var.includes(valeur) : retourne true si <valeur> existe dans le tableau sinon false
var.length : taille du tableau

OTHER

new Date() : afficher la date/heure actuelle (date de la machine)

```
//spécification de date
        var anniversaire = new Date();
        anniversaire = new Date(9,20,2020) //20sept2020
```

typeof(var) : permet d'afficher le type du variable

/*---------Contrôle de saisi(un peu de HTML)---------*/

<INPUT>

=> Lors de l'utilisation d'une balise <input>, vous pouvez contrôler le saisi d'un utilisateur a l'aide des attributs suivants

Attributs (INPUT)

id="<id>" : l'un des attributs le plus important dans le monde du développement web car il nous permet d'interagir avec les éléments(valeur) d'une balise identifiée.

max/minlength="<valeur>" : le nombre de caractères max/min qui peut être écrit dans le champ (pour être valide).

placeholder="Un peu de texte" : une valeur d'exemple qui sera afficher (si aucune valeur n'est saisie).

autocomplete="on/off" : afin de permettre au gestionnaire de mot de passe de saisir automatiquement/ou non le mot de passe.

type="text": permettent a l'utilisateur de saisir un **texte**.

type="password": permettent a l'utilisateur de saisir un **mot de passe** sans que celui-ci ne soit lisible a l'écran.

type="button": permettent de créer un bouton, parfois suivi d'un attribut **onclick**="function()" qui déclenche une fonction JS lors du clic.

required : le champ doit contenir une valeur non vide afin d'être validé.

pattern : le contenu du contrôle doit respecter l'expression rationnelle indiquée par l'attribut.

<LABEL>

=> Afin d'indiquer le rôle du champ, on utilise la balise **<label>** avec l'attribut **for=**"<id>" (liE avec l'attribut id="<id>" de la balise <input> qui le suit)

exemple

```
{
    <label for="userPassword" > Mot de passe : </label>
    <input type="password" id="userPassword">
}
```

<FORM>

=> Parfois utilise pour les formulaires mais peut être employées d'une autre façon

Petit calculatrice (somme)

```
<form onsubmit="return false" oninput="somme.value = parseInt(a.value) + parseInt(b.value)">
    <input type="number" name="a"  step="any"> +
    <input type="number" name="b" step="any"> =
    <output name="somme"></output>
</form>
```

/*---------Class & Objet(initiation a la POO)---------*/

OBJET

//Principe

> une variable qui sert a stocker plusieurs données dans des sous-variables (plusieurs autres variable) /analogue a un objet dans le monde réel

> un objet = variable(sous-variables(clé-valeur : propriete-valeur))

//Types

- ✗ **Objets natifs :** Math, Array, Boolean ...
- ✗ **Objets d'hôtes :** console, alert, document...
- ✗ **Objets créer**

//Création d'un objet (avec méthode)

```
var personne = {

    //Les sous-variables
    nomComplet : "RAKOTO Nirina",
    age : 34,
    accesAdmin : false

    //Les methodes(fonctions)
    message : function(){
        alert ("Bonjour " + personne.nomComplet);
    }
    };
```

//Manipulation

=> Accès a une valeur:

notation1 personne["age"] ; //Pour accéder a la valeur de la variable age de l'objet personne

notation2 personne.age ;

=> Appel d'une fonction (objet) :

personne.message() ;

=> Affectation:

personne.acces = true;

=> Type référence :

Si objet1 = objet2 ;

```
//objet1.clé-valeur est toujours identique a objet2.clé-valeur car ils utilisent le meme pointeur

//Parcours d'un objet - Toujours la notation1
for (var cleVal in personne) {
        document.write (cleVal, personne[cleVal] + "<br>");
}
```

output (sortie)

```
{

        nomCompletRAKOTO Nirina
        age34
        accesfalse

}
```

//Mot clé "with" (procedure d'un objet)

```
with(personne){
    alert("Bonjour " + nomComplet + "!");
    confirm("Veuillez confirmer si ces information sont vraies: <br> Age : " + age + "<br> Acces Admin : " + accesAdmin);
}
```

#CLASS (explication nv intermédiaire)

//Principe

>> **Class** : modèle de fabrication des objets (entité propre/qlqc de concret/abstrait)

> principaux caractéristiques d'une class :

Lors de le création d'une class, **2 questions** devraient être posées:

1. En quoi consiste un/une "nom de la class" ? (éléments)
2. Qu'est ce qu'on peut faire avec ? (méthodes)

(lié au Q1) > **État (éléments/propriétés/attributs) :** ce sont des variables qui sont lui propre (concret/abstrait)
(lié au Q2) > **Comportement (services/actions/méthodes) :** ce sont des fonctions

> mots cles : **constructor / this / new**

* constructor permet de créer des objets (éléments) d'une class

* this permet de lier les éléments a un attribut (sert aussi a l'accès interne sur un élément)

```
class <nom de la class>
        {
            //les etats (elements/proprietes/attributs)
            constructor(<attribut>)
            {
                this.<elements 1> = <attribut>;
                this.<elements 2> = <attribut>;
                ...
                this.<elements n> = <attribut>;
            }

            //les methodes (comportements/fonctions)
            fonction1()
            {
                instruction;
            }
            fonction2()
            {
                instruction;
            }
            ...
            fonctionN()
            {
                instruction;
            }
```

exemple

```
/*
    Dans cette exemple, on va voir:
        - La definition d'une class a partir du mot cle "class"
        - La creation d'un objet (etat/methode) grace au mot cle "constructor"
        - 2 differentes facons d'instancier et de maniere securisE une class (simple/par des fonctions get)
        - Modification de valeur par la fonction set
    */
    {
        class musicPlayer //une superfonction qui permet de lire des audios
        /*
         Autre syntax pour la definition d'une class
            let MP = class musicPlayer
            {

            }
            let MP = class
            {

            }
            let MP = {

            }
        */
```

(suite)

```
{
    constructor(format) //creation d'un objet format
    {
        this.format = format; //attribuer qlqc l'element this.format

        //Appel interne d'une class (dans la class elle meme)
        document.write("Le format est " + this.format);
    }

    //Appel interne d'une class grace au methode getformat()
        getformat() //sans parametre
        {
            return document.write("Le format est " + this.format);
        }

    //Modification de valeur par la methode setformat()
        setformat(newFormat)
        {
            this.format = newFormat;
        }

    //methode pour ecouter l'audio
        play()
        {
            console.log("Lecture en cours ...");
            /*cette instruction sert a une exemple
            (ce n'est pas un algo pour lire des audios)
            */
        }
}
```

* new permet de créer une instance d'une class

syntaxe

```
{
    let <var> = new <nom de la class>(<attribut>);
}
```

exemple

```
/*
Dans cette exemple, on va voir:
    - Acces exerne d'un element (2 exemples)
    - Lecture de l'audio
    - Modification du format
*/
{
    //definition de la valeur du format (dans une variable)
    let player = new musicPlayer("mp3");

    //1er exemple: appelation simple
```

(suite)

```
document.write("Le format est" + player.format);

    //Lecture de l'audio
    player.play();

    //Modification du format
    player.setformat("ogg");

    //2e exemple: appelation securisE
    document.write("Le format est" + player.getformat());
```

output (sortie)

```
{
        Le format est mp3

        Lecture en cours...

        Le format est ogg
}
```

***N.B**: Parfois, il nous faut au moins un élément pour accomplir une action (Ex : Des yeux pour voir)*

Exemples (explication)

```
class = humain //modèle concret
    objet = homme/femme //nom de l'objet
    État {Yeux, cheveux, taille, poids, force, vitesse, intelligence, ...} /*les elements qui
    constitue l'objet (caracteristique/capacitE)*/
    Comportement {Boire, manger, dormir, parler, ...} /*les actions possibles par l'objet*/

class = animal
    objet = Oiseau, chat, poisson, ...
    Etat {Ailes, Vision, AgressivitE, santE, ...}
    Comportement {Chasser, voler, courir, manger, aboyer, ...}
```

//Dans la POO, on peut rattacher un objet a une autre classe (ici, un humain peut voler, comme dans un jeu video)

class = voler (abstrait)
objet = humain
Etat = {powerToFly = 1} //l'élément qui permet de voler (un pouvoir)
Comportement {gravity(0)} //fonction qui permet de voler

N.B:
+ Les objets peuvent interagir entre eux (Un humain qui conduit une voiture), on peut trouver une tonne de manipulation d'objet dans un jeu vidéo
+ A savoir qu'on peut réutiliser une class déjà créer par un autre développeur sans jamais analysé le code (a condition que ce dernier est standard) sinon il y aura des bugs
+ On peut aussi améliorer/modifier une class existant (évolutivité d'un code, a condition que c'est en opensource)

- ✔ Objet : instance produit a partir d'un modèle (qui est une class)
- ✔ Class (global) != Objet (spécifique)
- ✔ Class = super-fonction (factorisation de plusieurs fonctions différentes, d'où l'intérêt de la POO)

/*---------DOM (Document Objet Model---------*/

//Principe

- Permette aux développeurs d'accéder a tous les éléments d'une page web
- Changer littéralement quoi que ce soit sur une page web en modifiant les propriétés DOM
- Imaginer le DOM pour comprendre comment il fonctionne
- DOM = Structure en arbre (inversé) d'une page HTML

#getElementById()

Permet de **recuperer les informations** d'une balise identifiée par son **id**

Syntaxe : let infoArecuperer = document.getElementById("id");

document : représente la page HTML, plus précisément l'intérieur de la fenêtre du navigateur (la zone d'affichage sans la barre d'adresse et les boutons de navigation)
id : désigne la valeur de l'attribut id d'une balise unique située dans la page HTML

***N.B** : getElementById peut accéder a des éléments meme dans une fonction*

#LES MÉTHODES de getElementById()

- getElementById().style : permet de changer les style CSS d'un élément

exemple

```
<script>
function changeColor(newcolor){
    document.getElementById("paragraphe").style.color = newcolor;
}
</script>
<body>
    <p id="paragraphe"> Un peu de texte </p>
    <button onclick="changeColor('blue')"> blue </button>
    <button onclick="changeColor('red')"> red </button>
</body>
```

- getElementById("Input").value : permet de recuperer la valeur d'une entrer (d'une balise HTML)

  exemple [1]

```
<script>
    function control(){
        saisi = document.getElementById("input").value;
        alert ("Vous avez saisi : " + saisi);
    }
</script>
<body>
    <form>
        <input type="text" id="input" name="entrE" value=""><br>
        <input type="button" id="bouton" value="ctrl" onclick="control()">
    </form>
</body>
```

exemple [2]

```
<script>
    function control(){
        var saisi = document.getElementById("Ztext1").value;
        document.getElementById("Ztext2").value = saisi;
    }
</script>
<body>
    <center>
        <form>
            <input type="text" id="Ztext1" value=""><br>
            <input type="button" value="Ctrl" onclick="control()"><br>
            <input type="text" id="Ztext2" value="">
        </form>
    </center>

</body>
```

COURS JAVASCRIPT <fin/>