INTRODUCTION ET RAPPEL JAVASCRIPT

INTRODUCTION

JavaScript est un langage de programmation qui ajoute de l'interactivité à votre site web (par exemple : jeux, réponses quand on clique sur un bouton ou des données entrées dans des formulaires, composition dynamique, animations).

JavaScript (« JS » en abrégé) est un langage de programmation dynamique complet qui, appliqué à un document HTML, peut fournir une interactivité dynamique sur les sites Web.

JavaScript est plutôt compact tout en étant très souple. Les développeurs ont écrit de nombreux outils sur le cœur du langage JavaScript, créant des fonctionnalités supplémentaires très simplement. Parmi ces outils, il y a :

- **des Interfaces de Programmation d'Applications pour navigateurs (API) — API intégrées aux navigateurs web permettant de créer dynamiquement du HTML, de définir des styles de CSS, de collecter et manipuler un flux vidéo depuis la webcam de l'utilisateur ou de créer des graphiques 3D et des échantillonnages audio.**
- **des API tierces-parties permettant aux développeurs d'incorporer dans leurs sites des fonctionnalités issues d'autres fournisseurs de contenu, comme Twitter ou Facebook.**
- **des modèles et bibliothèques tierces-parties applicables à votre HTML permettant de mettre en œuvre rapidement des sites et des applications.**

- Javascript permet
  - de programmer des actions en fonction d'événements utilisateurs (déplacements de souris, focus, *etc.*)
  - d'accéder aux éléments de la page HTML/XHTML (traitement de formulaire, modification de la page)
  - d'effectuer des calculs sans recours au serveur
- **Javascript ≠ scripts Java !**

Javascript
- interprété
- orienté prototypes (héritage de propriétés)
- code intégré (visible) ⇒ à offusquer
- typage faible
- langage à part entière
- débuggage difficile, maintenant facilité par l'arrivée de nouveaux outils

FONCTION EN JS

- Il y a les fonctions/variables natives (déjà existante). On peut aussi en créer.
- Syntaxe

```
function MaFonction(arguments){
    // le mot function est obligatoire
    // ici, le corps de la fonction
}
```

- Exemple

```
function essai(){
    var res = parseInt(prompt('Donner un nombre : '));
    alert(res*2);
}
essai(); // on appelle la fonction
```

VARIABLES LOCALES/GLOBALES

- Toute variable déclaré dans une fonction n'est utilisable que dans cette même fonction : variable locale.
- Déclaré en dehors des fonctions : variables globales.
- Exemple :

```
var mess = 'variable globale';
function showmsg(){
    var mess = 'variable locale';
    alert(mess);
}
showmsg();    // affiche variable locale
alert(mess);  // affiche variable globale
```

- Eviter de déclarer des variables locales et globales qui portent le même nom

LET/VAR

L'instruction **let** permet de déclarer une variable dont la portée est celle du bloc courant, éventuellement en initialisant sa valeur.

```
let x = 1;

if (x === 1) {
  let x = 2;

  console.log(x);
  // expected output: 2
}

console.log(x);
// expected output: 1
```

Le mot-clé let permet de définir des variables au sein d'un bloc et des blocs qu'il contient. var permet quant à lui de définir une variable dont la portée est celle de la fonction englobante.

```
function varTest() {
    var x = 1;
    if (true) {
      var x = 2;  // c'est la même variable !
      console.log(x);  // 2
    }
    console.log(x);  // 2
}

function letTest() {
    let x = 1;
    if (true) {
      let x = 2;  // c'est une variable différente
      console.log(x);  // 2
    }
    console.log(x);  // 1
}
```

```
var a = 5;
var b = 10;

if (a === 5) {
  let a = 4; // La portée est celle du bloc if
  var b = 1; // La portée est celle interne à la fonction

  console.log(a);  // 4
  console.log(b);  // 1
}

console.log(a); // 5
console.log(b); // 1
```

- Let dans for:

```
for (let i = 0; i<10; i++) {
    console.log(i); // 0, 1, 2, 3, 4 ... 9
}

console.log(i); // i n'est pas défini
```

ARGUMENTS

- Exemples avec un/multiples arguments :

```
function fct(arg){
    alert('Votre argument est : ' + arg);
}
fcn('test');
fcn(prompt('Donner votre argument'));
```

```
function fct(arg1, arg2){
    alert('premiere argument : ' + arg1);
    alert('deuxieme argument : ' + arg2);
}

fcn(
    prompt('Donner votre  1er argument :'),
    prompt('Donner votre 2eme argument :')
);
```

- Avec valeur de retour

```
function bjr(){
    return 'salut';
}
alert(bjr());
```

```
function multiply(num1,num2) {
    let result = num1 * num2;
    return result;
}

multiply(4, 7);
multiply(20, 20);
multiply(0.5, 3);
```

LES ERREURS LIES A LET

- Lorsqu'on redéclare une même variable au sein d'une même portée de bloc, cela entraîne une exception SyntaxError.

```
function faire_quelque_chose() {
    console.log(truc); // undefined
    console.log(toto); // ReferenceError
    let toto = 2;
    var truc = 1;
}
```

```
switch (x) {
    case 0:
      let toto;
      break;

    case 1:
      let toto; // SyntaxError pour la redeclaration
      break;
}
```

```
if (x) {
    let toto;
    let toto; // SyntaxError
}
```

```
let x = 1;

if (true) {
  var x = 2; // SyntaxError liée à la redéclaration
}
```

CONSTANTES

- Syntaxe : `const CONSTANT_NAME = value;`

Exemple

```
const K = 0.1;
K = 0.2; // TypeError
```

```
const person = { age: 20 };
person.age = 30; // OK
console.log(person.age); // 30
person = { age: 40 }; // TypeError
```

```
const person = Object.freeze({age: 20});
person.age = 30; // TypeError
```

```
const company = Object.freeze({
    name: 'Jacob',
    address: {
        city: 'Tana',
        state: 'Mada',
        code: 101
    }
});
company.address.country = 'USA'; // OK
```

TAMPLATE LITTERAL

backticks (`)

```
let str = `exemple \n template littéral`;
console.log(str);// ??
console.log(str.length); // 23
console.log(typeof str);// string
```

```
let post = {
    title: 'Template Literals',
    body: 'corps de la page ...',
    tags: ['es6', 'template literals', 'javascript']
};

let {title, body, tags} = post;

let postHtml =
`<header>
    <h1>${title}</h1>
</header>
<section>
    <div>${body}</div>
</section>
<footer>
    <ul>
      ${tags.map(tag => `<li>${tag}</li>`).join('\n    ')}
    </ul>
</footer>`;
```

PARAMETRES PAR DEFAUT

Exemple

```
function say(message='Coucou') {
    console.log(message);
}

say(); // 'Coucou'
say(undefined); // 'Coucou'
say('Bonjour'); // 'Bonjour'
```

Utilisation de l'objet 'arguments'

```
function add(x, y = 1, z = 2) {
    console.log( arguments.length );
    return x + y + z;
}

add(10); // 1
add(10, 20); // 2
add(10, 20, 30); // 3
```

PARAMETRE D'APPUI

Avec l'operateur spread

```
const odd = [1,3,5];
const combined = [2,4,6, ...odd];
console.log(combined); //  [2, 4, 6, 1, 3, 5]
```

Exemple :

```
function f(a, b, ...args) {
    console.log(args);
}
f(1, 2, 3, 4, 5); // [3, 3, 5]
```

```
function sum(...args) {
    let total = 0;
    for (const a of args) {
        total += a;
    }
    return total;
}

sum(1, 2, 3); // 6
```

```
function sum(...x) {
    return x
        .filter(function (e) {
            return typeof e === 'number';
        })
        .reduce(function (prev, curr) {
            return prev + curr;
        });
}

let result = sum(10,'blablabla',null,undefined,20);
console.log(result); // 30
```

DESTRUCTURATION DES TABLEAUX

Exemple

```
function getProfil() {
    return [
        'Mbape',
        'Kylian',
        ['Rouge', 'Vert', 'Bleu']
    ];
}

let [
    firstName,
    lastName,
    [ color1, color2, color3 ]
] = getProfil();

console.log(color1, color2, color3); // Rouge Vert Bleu
```

DESTRUCTURATION D'OBJET

- Exemple :

```
let employee = {
    id: 1001,
    nom: {
        firstName: 'Randrianasoa',
        lastName: 'Iano'
    }
};

let {
    nom: {
        firstName,
        lastName
    },
    nom
} = employee;

console.log(firstName); // Randrianasoa
console.log(lastName); // Iano
console.log(nom); // ??
```

DESTRUCTURATION DES ARGUMENTS D'UNE FONCTION

- Exemple :

```
let display = (person) => console.log(`${person.firstName} ${person.lastName}`);

let person = {
    firstName: 'Randrianasoa',
    lastName: 'Iano'
};

display(person);
```

```
let display = ({firstName, lastName}) => console.log(`${firstName} ${lastName}`);

let person = {
    firstName: 'Randrianasoa',
    lastName: 'Iano'
};

display(person);
```

TABLEAUX ET METHODES

CONSTRUIRE UN TABLEAU

```
function Person(nic,age,sex,classe,work,friend){
    this.nic = nic;
    this.age = age;
    this.sex = sex;
    this.classe = classe;
    this.work = work;
    this.friend = friend;
}
```

- La première lettre de la fonction du constructeur doit-être en Majuscule
- On utilise new pour construire
- On peut ensuite ajouter ou remplacer des variables

```
var Raf = new Person('Raphael',15,'m','3eA','javascripteur',[]);
var Lea = new Person('Léa',18,'f','5eA','webmaster',[]);
alert(Raf.nic);
alert(Lea.nic);
Raf.age = 16;
// On peut ensuite ajouter des méthodes, par exemple un friend dans le tableau
Raf.friend.push(new Person('Jerome',15,'m','3eA','javascripteur',[]));
```

LES METHODES : TOSTRING, JOIN, DELETE

- `toString()` joint tous les éléments du tableau dans une chaîne (séparateur virgule).

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
```

- `join()` joint également tous les éléments du tableau dans une chaîne

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.join(" * ");
```

- `delete` supprimer un élément du tableau

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
delete fruits[0];           // Changes the first element in fruits to undefined
```

LES METHODES : SPLICE

- `splice()` peut être utilisée pour ajouter de nouveaux éléments à un tableau ou supprimer des éléments sans laisser de "trous" dans le tableau

```
<button onclick="myF1()">F1</button> <br />
<button onclick="myF1()">F2</button> <br />
<button onclick="myF2()">F3</button>
<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3"></p>
<script>
    var fruits = ["Banana", "Orange", "Apple", "Mango"];
    document.getElementById("demo1").innerHTML = "Originale :<br>" + fruits;
    function myF1() {
        fruits.splice(2, 0, "Lemon", "Kiwi");
        document.getElementById("demo2").innerHTML = "Nouveau :<br>" + fruits;
    }
    function myF2() {
        fruits.splice(0, 1);
        document.getElementById("dem1").innerHTML = fruits;
    }
    function myF3() {
        var removed = fruits.splice(2, 2, "Lemon", "Kiwi");
        document.getElementById("demo2").innerHTML = "Nouveau :<br>" + fruits;
        document.getElementById("demo3").innerHTML = "Effacés :<br> " + removed;
    }
</script>
```

LES METHODES : CONCAT, SLICE

- `concat()` crée un nouveau tableau en concaténant des tableaux existants

```
var tab = ['test1','test2'].concat(['test3','test4']);

tab.forEach(function(value,index,array){
    alert(
        'Index : ' + index +
        '\n Value : ' + value
    )
})
var arr1 = ["Cecilie", "Lone"];
var arr2 = ["Emil", "Tobias", "Linus"];
var arr3 = ["Robin", "Morgan"];
var myChildren = arr1.concat(arr2, arr3);
```

- `slice()` extraire une partie du tableau

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(1);
// citrus = ["Banana", "Lemon", "Apple", "Mango"]
var citrus = fruits.slice(3);
// citrus = ["Apple", "Mango"]
var citrus = fruits.slice(1, 3);
// citrus = ["Orange", "Lemon"]
```

LES METHODES : SORT, REVERSE

- `sort()` ordonner alphabétiquement et numériquement

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();          // Sorts the elements of fruits
```

```
// par ordre numerique
var tab = [3,2,5,4,7,1,6];

tab.sort(function(a,b){
    if(a<b) {
        return -1;
    }else if(a>b) {
        return 1;
    }else {
        return 0;
    }
})
```

Croissant

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a - b});
```

Décroissant

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return b - a});
```

- `reverse()` inverser un tableau

```
fruits.reverse();
```

LES METHODES : SORT, RANDOM

- `sort()` exemple

```
<body>
    <button onclick="myF1()">Sort Alphabetic</button>
    <button onclick="myF2()">Sort Numeric</button>

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

    <script>
        var points = [40, 100, 1, 5, 25, 10];
        document.getElementById("demo").innerHTML = points;

        function myF1() {
            points.sort();
            document.getElementById("demo").innerHTML = points;
        }
        function myF2() {
            points.sort(function(a, b){return a - b});
            document.getElementById("demo").innerHTML = points;
        }
    </script>
</body>
```

- Ordre random

```
points.sort(function(a, b){return 0.5 - Math.random()});
```

LES METHODES : FOREACH , FILTER

- `forEach()` appelle une fonction (une fonction de rappel) une fois pour chaque élément du tableau

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

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

(La valeur de l'article ,L'index des articles , Le tableau lui-même)

- `filter()` filtre un tableau

```
var numbers = [45, 4, 9, 16, 25];
var over18 = numbers.filter(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

LES METHODES : MAP, EVERY

`map()` crée un nouveau tableau en exécutant une fonction sur chaque élément du tableau.
`map()` n'exécute pas la fonction pour les éléments de tableau sans valeurs.
`map()` ne modifie pas le tableau d'origine.

```
var numbers1 = [45, 4, 9, 16, 25];
var numbers2 = numbers1.map(myFunction);

function myFunction(value) {
  return value * 2;
}
```

`every()` vérifie si toutes les valeurs du tableau réussissent un test

```
var numbers = [45, 4, 9, 16, 25];
var allOver18 = numbers.every(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

LES METHODES : REDUCE, REDUCERIGHT

`reduce()` exécute une fonction sur chaque élément du tableau pour produire (le réduire à) une valeur unique.
`reduce()` fonctionne de gauche à droite dans le tableau.
`reduce()` ne réduit pas le tableau d'origine.

```
var numbers1 = [45, 4, 9, 16, 25];
var sum = numbers1.reduce(myFunction);

function myFunction(total, value, index, array) {
  return total + value;
}
```

total : la valeur initiale / la valeur précédemment renvoyée

Elle peut accepter une valeur initiale

```
var numbers1 = [45, 4, 9, 16, 25];
var sum = numbers1.reduce(myFunction, 100);

function myFunction(total, value) {
  return total + value;
}
```

LES METHODES : SOME, INDEXOF

`some()` vérifie si certaines valeurs du tableau réussissent un test.

```
var numbers = [45, 4, 9, 16, 25];
var someOver18 = numbers.some(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

- `indexOf()` recherche dans un tableau une valeur d'élément et renvoie sa position.
- **Remarque :** Le premier élément a la position 0, le deuxième élément a la position 1, et ainsi de suite

```
var fruits = ["Apple", "Orange", "Apple", "Mango"];
var a = fruits.indexOf("Apple");
```

Syntaxe : *array*.indexOf(*item*, *start*)

item : obligatoire

start : optionnel

LES METHODES : INDEXOF

Valeur de retour
Le premier index de l'élément dans le tableau ou -1 si la valeur n'est pas trouvée.

Exemple : on peut utiliser indexOf afin de trouver l'emplacement d'un élément dans un tableau.

```
var tableau = [2, 9, 9];
tableau.indexOf(2);     // 0
tableau.indexOf(7);     // -1
tableau.indexOf(9, 2);  // 2
tableau.indexOf(2, -1); // -1
tableau.indexOf(2, -3); // 0
```

```
var indices = [];
var tableau = ['a', 'b', 'a', 'c', 'a', 'd'];
var élément = 'a';
var idx = tableau.indexOf(élément);
while (idx != -1) {
  indices.push(idx);
  idx = tableau.indexOf(élément, idx + 1);
}
console.log(indices);
// [0, 2, 4]
```

LES METHODES : FIND, FINDINDEX

`find()` renvoie la valeur du premier élément du tableau qui réussit une fonction de test.

```
var numbers = [4, 9, 16, 25, 29];
var first = numbers.find(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

```
var numbers = [4, 9, 16, 25, 29];
var first = numbers.findIndex(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

GESTION DES CHAINES DE CARACTÈRES

```
var Chaine = "ma chaine"; // chaine primitive
var ChaineReal = new String("ma chaine");

var Tab = []; // Tableau primitif
var TabReal = new Array();

var Objet = {}; // objet primitif
var ObjetReal = new Object();

var Bool = true; // Booleen primitif
var BoolReal = new Boolean("true");

var Nbr = 42; // Nombre primitif
var NbrReal = new Number("42");
```

AUTRES METHODES :

- trim() : supprime les espaces d'une chaine
- split() : coupe une chaine en un tableau

EVENEMENTS FORMULAIRES

LES ÉVÈNEMENTS

`click`	Cliquer (appuyer puis relâcher) sur l'élément
`dblclick`	Double-cliquer sur l'élément
`mouseover`	Faire entrer le curseur sur l'élément
`mouseout`	Faire sortir le curseur de l'élément
`mousedown`	Appuyer (sans relâcher) sur le bouton gauche de la souris sur l'élément
`mouseup`	Relâcher le bouton gauche de la souris sur l'élément
`mousemove`	Faire déplacer le curseur sur l'élément
`keydown`	Appuyer (sans relâcher) sur une touche de clavier sur l'élément
`keyup`	Relâcher une touche de clavier sur l'élément
`keypress`	Frapper (appuyer puis relâcher) une touche de clavier sur l'élément
`focus`	« Cibler » l'élément
`blur`	Annuler le « ciblage » de l'élément
`change`	Changer la valeur d'un élément spécifique aux formulaires (`input`, `checkbox`, etc.)
`select`	Sélectionner le contenu d'un champ de texte (`input`, `textarea`, etc.)

LES ÉVÈNEMENTS

- Les deux événements spécifiques à <form>

`submit`	Envoyer le formulaire
`reset`	Réinitialiser le formulaire

- Utilisation click

```
<span onclick="alert('Voici le contenu de l\'element cliqué : '+this.innerHTML);" >
  clique-moi
</span>
```

- Utilisation focus et blur

```
<input id="input" type="text" size="50" value="cliquez ici"
  onfocus="this.value='Appyuez sur votre touche de tabulation';"
  onblur="this.value='cliquez ici';" />
  <br/><br/>
<a href="#" onfocus="document.getElementById('input').value =
  'vous avez maintenant le focus sur le lien'; " > un lien </a>
```

FORMULAIRES

- Utilisation de la propriété value

```
<input id="text" type="text" value="Pas de focus" />
```

```
var text = document.getElementById('text');
text.addEventListener('focus',function(e) {
  e.target.value= 'Avec focus';
}, true);
text.addEventListener('blur',function(e) {
  e.target.value= 'Pas de focus';
}, true);
```

- Pour désactiver un champ de text

```
var text = document.getElementById('text');
text.disabled = true;
```

FORMULAIRES

- Utilisation checked pour les boutons radio

```
<label> <input type="radio" name="check" value="1" /> case 1</label> <br/>
<label> <input type="radio" name="check" value="2" /> case 2</label> <br/>
<label> <input type="radio" name="check" value="3" /> case 3</label> <br/>
<label> <input type="radio" name="check" value="4" /> case 4</label> <br/>

<input type="button" value="Afficher la case cochée" onclick="check();" />
```

```
function check() {
  var inputs = document.getElementsByTagName('input'),
  inputsLength = inputs.length;
  for(var i=0 ; i<inputsLength ; i++ ){
    if(inputs[i].type == 'radio' && inputs[i].checked){
      alert('La case cochée est la numero '+inputs[i].value);
    }
  }
}
```

FORMULAIRES

- Utilisation selectedIndex pour la liste déroulante

```
<select id="list">
  <Option>Selectionner votre sexe</Option>
  <Option>Homme</Option>
  <Option>Femme</Option>
</select>
```

```
var list = document.getElementById('list');
list.addEventListener('change',function() {
  // On affiche le contenu de <option>
  alert(list.options[list.selectedIndex].innerHTML);
}, true);
```

FORMULAIRES

- Utilisation submit() et reset()

```html
<form id="myF">
  <input type="text" value="Entrer un texte" />
  <input type="submit" value="submit" />
  <input type="reset" value="reset" />
</form>
```

```javascript
var myF = document.getElementById('myF');
myF.addEventListener('submit',function(e) {
  alert('Vous avez envoyé le formulaire');
  e.preventDefault();
}, true);
myF.addEventListener('reset',function(e) {
  alert('Vous avez reinitialisé le formulaire');
}, true);
```

FORMULAIRES

- Utilisation focus() blur() et select()

```
<input id="text" type="text" value="Entrer un texte" /> <br />
<input type="button" value="Donner le focus"
  onclick="document.getElementByID('text').focus();"
/> <br />
<input type="button" value="Retirer le focus"
  onclick="document.getElementByID('text').blur();"
/> <br />
<input type="button" value="Selectionner le focus"
  onclick="document.getElementByID('text').select();"
/>
```

EXERCICES

EXERCICES

- **Exercice 1 :**

Demande d'introduire un nombre dans un champ input

Un clic sur un bouton affiche la valeur binaire de ce nombre

- **Exercice 2 :**

•Soit le tableau suivant : ['Mangue', 'Raisin', 'Figue', 'Kiwi'];

•Écrire un programme qui :

Affiche la liste de fruits disponibles ;

Demande au client quel fruit il désire acheter :

•s'il est présent dans le tableau fruits: le retirer du tableau, et afficher 'ok!',

•sinon, afficher 'indisponible…'.

Affiche à nouveau la liste de fruits disponibles.

- **Exercice 3 :**

Gérer une liste grâce au menu suivant :

1. Ajouter un élément dans la liste
2. Afficher la liste
3. Supprimer un élément de la liste par son indice.
4. Rechercher une chaine dans la liste donnée par l’utilisateur
5. Ordonner par ordre croissant la liste
6. Quitter

NB : Lorsqu'on rajoute un élément, il sera rajouté à la fin de la liste.

Créer un champ input pour que l'utilisateur va introduire son choix et affiche le résultat de son choix

Exercice 4 :

- Ecrire un programme qui demande à l'utilisateur de saisir 5 nombres entiers comprises entre 9 et 99, puis une fois la saisie terminée, les affiche.
- Le programme doit redemander la saisie en cas d'erreur sans compter cette étape
- Le programme affiche les nombres des entiers pairs, le nombre des entiers impairs saisis et le nombre des itérations fausses
- Le programme affiche le tableau des nombres pairs et le tableau des nombres impairs saisis
- Le schéma suivant représente le résultat souhaitait

Nombres pairs	Nombres impairs
14	47
78	45
	47