Algorithmique JavaScript

"La plus grande attention doit être portée à la compréhension du problème, faute de quoi l'algorithme n'a aucune chance d'être correct". Denis Lapoire

"C'est toujours l'impatience de gagner qui fait perdre", Louis XIV cité dans "L'immortel" de FOG.

J'écoute et j'oublie.
Je lis et je retiens.
Je fais et j'apprends.
(Proverbe chinois)

TABLE DES MATIÈRES

1.1 - PRÉMISSES

1.1.1 - Objectif de ce document et ses limites

Ce document est un support de cours. Donc difficilement praticable sans un formateur !
Ce n'est pas un support de cours JavaScript. C'est un support de cours d'algorithmique avec JavaScript.

Il est le parallèle du support de cours Algorithmique.odt, ou plutôt une éventuelle continuation.

Seules sont présentées les « instructions » nécessaires pour réaliser des algorithmes basiques.
JavaScript est utilisé dans un contexte « client ». Donc en association avec des pages HTML. Donc une connaissance basique mais solide de HTML est requise (Assimilation des notions d'élément, de balise, d'identification, …).

1.1.2 - Utilisation de ce document

Pour un cours d'algorithmique pour des débutants en informatique, seuls les chapitres suivants sont abordés :

structure séquentielle,
fonction et procédure,
structure conditionnelle,
structure itérative : le FOR,
tableaux ordinaux,
structure itérative : le WHILE,
chaînes de caractères.
Table de hachage !

1.1.3 - Trois mots sur JavaScript

https://developer.mozilla.org/fr/docs/Web/JavaScript

1.1.3.1 - Généralités

Trois mots sur JavaScript peuvent être utiles, ne serait-ce que pour situer le contexte.

JavaScript « client » est un langage interprété par la plupart des navigateurs (par tous les navigateurs de « large » diffusion mondiale (Chrome, Firefox, Edge, IE, Safari, Opera, ...)).

JavaScript est aussi un langage serveur (avec Node.js par exemple).

JavaScript est sensible à la casse.

Le code JavaScript est écrit soit entre les balises <script> et </script> dans un page HTML.
Soit dans un fichier JavaScript d'extension .js et référencé dans une page HTML (ce qui est fortement recommandé).

De nombreuses bibliothèques (jQuery, …) et de nombreux frameworks (AngularJS, Angular 2, ReactJS, React, ...) sont basés sur JavaScript.

JavaScript est un langage qui implémente les paradigmes **objet, impératif, événementiel**.

C'est-à-dire que – côté client - les éléments HTML sont transformés en OBJETS avec des attributs et des méthodes, que l'utilisateur interagit avec l'interface (IHM : Interface Homme Machine) et que le développeur peut gérer ces interactions via des événements (onclick, …). Enfin on retrouve les structures classiques de la programmation impératives (séquence, conditionnelle, itérative).

1.1.3.2 - Paradigme impératif ou procédural

Un programme procédural – un script procédural - est un programme qui met en place les structures de la programmation procédurale.
Les instructions peuvent se suivre en séquence (exécutées en séquence les unes après les autres toujours dans le même ordre).
Les instructions peuvent être conditionnées à la validé d'une condition (VRAI ou FAUX).
Les instructions peuvent être répétées n fois.

1.1.3.3 - Paradigme objet

Un objet est une représentation informatique d'un objet concret (une personne) ou abstrait (un module de cours) du monde réel.

Un objet est caractérisé par des attributs (éléments statiques, des attributs ou des variables) et des méthodes (éléments dynamiques, des méthodes ou des fonctions).
Par exemple les attributs d'une page HTML sont les éléments du BODY.
Et les attributs d'un élément HTML sont des attributs des objets.

Un objet – une classe - est représenté par un rectangle à 3 compartiments (nom de l'objet, attributs, méthodes).
Le premier rectangle est « syntaxique », générique.
Les 3 autres rectangles présente de façon non exhaustive 3 objets JS (document, input, label) qui correspondent à 3 éléments HTML.

Classe
attributs
methodes()

Document
characterSet contentType
getElementById(id)

Input
type value id
onclick() onfocus() onblur()

Label
innerHTML
onclick()

1.1.3.4 - Paradigme événementiel

L'exécution d'un code événementiel (ensemble d'instructions) est conditionné par l'interaction de l'utilisateur. Quand un utilisateur clique sur un bouton, le système reçoit un message. Ce message peut être « réceptionné » par du code JS et traité.

1.2 - Les instructions de base de JavaScript

1.2.1 - Notions de JavaScript

Si le code JS est situé dans une page HTML il est écrit à l'intérieur des balises **<script></script>**.
Mais il est déconseillé de mettre du code JS dans une page HTML.

L'inclusion de code JS externe est réalisée avec cette balise **<script src="../js/fichier.js"></script>**.
Le code JS dans un fichier JS est hors de toute balise.

Les commentaires sont :
entre /* et */ : commentaire de bloc,
ou après // : commentaire de ligne,
ou entre /** */ : commentaire de documentation.

Toute instruction JS est terminée par un ; (point-virgule).

Les noms des variables, constantes, fonctions sont **sensibles à la casse**.

Les variables, les constantes et les fonctions ne sont pas typées.

Une variable est déclarée après le mot **var ou let**.
Elle peut être initialisée lors de la déclaration.
Une variable globale (déclarée hors d'une fonction avec le qualificateur **var**) est utilisable hors et dans une fonction.
Une variable locale (déclarée dans une fonction avec le qualificateur **let**) n'est utilisable que dans la fonction où elle est déclarée.

Une fonction ou une procédure est déclarée avec le mot **function**.

Le mot **return** permet de renvoyer une valeur ou une expression dans une fonction.
Sans instruction return une fonction JS est "void" donc une procédure.

Un bloc d'instructions commence par une accolade ouvrante **{** et se termine par une accolade fermante **}**.

if est utilisé pour les conditionnelles. La condition est entre parenthèses **()**.

for et **while** sont utilisés pour les boucles. Comme pour le if, les parenthèses ouvrent et ferment la condition.

La méthode **document.getElementById**("element") est utilisée pour pointer vers un élément HTML identifié. Elle renvoie un objet qui correspond à l'élément HTML dont l'id est passé en argument.

element.**onclick** = nomDeFonction; est utilisé pour associer une fonction sans paramètre et l'événement clic.

L'attribut element.**value** est utilisé pour affecter ou récupérer la valeur d'un élément HTML qui possède l'attribut value (des éléments de formulaire : input, select).

L'attribut element.**innerHTML** est utilisé pour affecter ou récupérer la valeur d'un élément HTML (div, p, label, ...) qui ne possède pas l'attribut value.

console.log(variable); Cette commande est utilisée pour afficher le contenu d'une variable à la console du navigateur.
Dans ce cours nous l'utiliserons très souvent. Vous affichez la console du navigateur avec F12.

alert(variable); Cette commande est utilisée pour afficher le contenu d'une variable dans une boîte de dialogue (déconseillée mais parfois utile pour aller vite !).

Quelques autres syntaxes de base seront vues au cours des présentations et exercices.

1.2.2 - Les variables

Une variable est un espace mémoire.
Une variable est un triplet : nom, type, valeur à un instant.
La valeur d'une variable peut changer !
Dans un fragment de code une variable est initialisée, son contenu est affiché dans une boîte de dialogue, modifié, puis affiché à nouveau.

```
var i = 10;
alert(i);
i++;
alert(i);
```


1.2.3 - Quelques opérateurs JavaScript

Catégories	Opérateurs
Affectation	=
Arithmétiques	+, -, *, /
Incrémentation, décrémentation, ...	++, --, +=, -=, *=, /= (i++ signifierait i = i + 1, etc)
Comparaisons	==, >, >=, <, <=, !=, ===, !== (Égalité, supérieur, supérieur ou égal, inférieur, inférieur ou égal, différent, identique, pas identique)
Logiques	&&, ||, ! (ET, OU, NON)
Concaténation	+

Remarques :
L'opérateur + est autant l'opérateur de l'addition que l'opérateur de la concaténation. Or lorsque l'on récupère une saisie d'un input type text, c'est une chaîne de caractères. Si c'est un chiffre ou un nombre il faut alors le transformer en nombre soit avec parseFloat("valeur"), parseInt("valeur").

1.3 - LES FONCTIONS ET LES PROCÉDURES

1.3.1 - Les fonctions

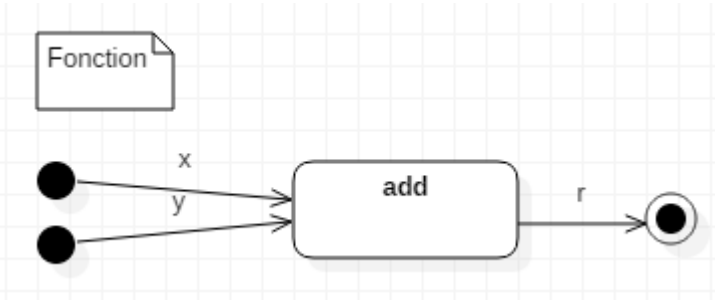
Définition : une fonction est un bloc de code qui se termine par un return. Même si techniquement il peut y avoir plusieurs « return » dans une fonction cela est déconseillé, cela dé-structure le code.
Elle a des arguments ou pas (parenthèses vides).
Le code de la fonction est exécuté lorsqu'on l'appelle – par son nom – dans une expression (affectation, condition, condition d'itération. Lors de l'appel l'ordre des paramètres doit respecté l'ordre des paramètres.
Les noms des fonctions, des paramètres, des variables locales sont camélisées et commencent par une minuscule. Ces noms doivent être significatifs !!!

Définition

```
function nomDeFonction([parametre1[, parametre2]]) {
    var variableLocaleDeResultat = valeur;

    // Calcul

    return variableLocaleDeResultat;
}
```

Appel

```
var x = nomDeFonction([parametre1[, parametre2]]);
```




```
// La fonction
function add(x,y)
{
    var r = 0;
    r = x + y;
    return r;
} /// add


// L'appel de la fonction
var somme = add(3,5);

// Affichage du contenu de la variable somme dans une boîte de dialogue
alert(somme);
```

1.3.2 - Les procédures

Définition : une procédure est un bloc de code sans return, plutôt réservée à de l'affichage.
Elle a des arguments ou pas (parenthèses vides).
Le code de la procédure est exécuté lorsqu'on l'appelle – par son nom - .
L'ordre des paramètres doit respecté l'ordre des arguments.
Plus rare d'utilisation que les fonctions.

```
// La procédure
function add(x,y)
{
    var r = x + y;
    alert(r);
} /// add



// L'appel de la procédure
add(3,5);
```

1.4 - STRUCTURE SÉQUENTIELLE

Une code est de structure séquentielle lorsque toutes les instructions sont exécutées et toujours dans le même ordre.

Algorithme	Description
Algo1	Afficher un texte suite à un clic sur un bouton. init(), afficher(), document.getElementById(), ...
Algo2	Afficher un texte suite à un clic sur un bouton. Fonction anonyme
Addition	Addition de 2 saisies
FonctionAddition	Même chose avec une fonction
Exercices	
Permutation	

1.4.1 - Premier exemple

Figure : la page web avec l'affichage d'un texte suite à un clic de l'utilisateur.


```
<!DOCTYPE html>
<html>
    <head>
        <title>Algo1.html</title>

        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
    </head>

    <body>
        <div>
            <h1>Algo1</h1>
            <input type="button" id="btOK" value="Go" />
            <label id="lblMessage"></label>
        </div>

        <script src="../js/Algo1.js"></script>
    </body>
</html>
```

```
/*
* Algo1.js
*/

// La procédure
function affichage() {
   document.getElementById("lblAffichage").innerHTML =
document.getElementById("itSaisie").value;
} /// affichage

document.getElementById("btValider").onclick = affichage;
```

1.4.2 - Deuxième exemple : une fonction anonyme

Figure : la page web avec l'affichage d'un texte au chargement de la page.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Algo2.html</title>

        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
    </head>

    <body>
        <div>
            <h1>Algo2</h1>
            <input type="button" id="btOK" value="Go" />
            <label id="lblMessage"></label>
        </div>

        <script src="../js/Algo2.js"></script>
    </body>
</html>
```

```
/*
* Algo2.js
*/

window.onload = function() {
    document.getElementById("btOK").onclick = function() {
        document.getElementById("lblMessage").innerHTML = "Hello JS";
    };
};
```


1.4.3 - Troisième exemple : une addition

X 3 + Y 5 = 8

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

        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
    </head>

    <body>
        <div>
            <label>X </label>
            <input type="text" id="itX" value="3" />
            <label> + Y </label>
            <input type="text" id="itY" value="5" />

            <input type="button" id="btAddition" value="=" />

            <label id="lblResultat">0</label>
        </div>

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

```
/*
 * Addition.js
 */
var itX;
var itY;
var lblResultat;
var btAddition;

// ------------
function init() {
    itX = document.getElementById("itX");
    itY = document.getElementById("itY");
    lblResultat = document.getElementById("lblResultat");
    btAddition = document.getElementById("btAddition");

    btAddition.onclick = function() {
        lblResultat.innerHTML = Number(itX.value) + Number(itY.value);
    };
} /// init

window.onload = init;
```


1.4.4 - Quatrième exemple : une addition avec une fonction

X 3 + Y 5 = 8

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

        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
    </head>

    <body>
        <div>
            <label>X </label>
            <input type="text" id="itX" value="3" />
            <label> + Y </label>
            <input type="text" id="itY" value="5" />

            <input type="button" id="btAddition" value="=" />

            <label id="lblResultat">0</label>
        </div>

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

```
/*
 * FonctionAddition.js
 */
var itX;
var itY;
var lblResultat;
var btAddition;

// ------------
function init() {
    itX = document.getElementById("itX");
    itY = document.getElementById("itY");
    lblResultat = document.getElementById("lblResultat");
    btAddition = document.getElementById("btAddition");
    btAddition.onclick = function() {
        lblResultat.innerHTML = additionner(parseInt(itX.value),
Number(itY.value));
    };
} /// init

window.onload = init;

// -----------------
function additionner(x, y) {
//                  var r;
//                  r = x + y;
//                  return r;
    return x + y;
} /// additionner

window.onload = init;
```

1.4.5 - Exercice : la permutation

a : 3

b : 5

Permutation

inter : 3

a : 5

b : 3

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Permutation</title>
        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
        <link href="../../css/jsAlgo.css" rel="stylesheet" />
    </head>

    <body>
        <div>
            <p>
                a : <input type="text" id="a" value="3" />
            </p>
            <p>
                b : <input type="text" id="b" value="5" />
            </p>
            <input type="button" value="Permutation" id="btPermutation" />
            <hr>
            <p>
                inter : <label id="lblInter" class="fondGris"></label>
            </p>
            <p>
                a : <label id="lblA" class="fondRouge"></label>
            </p>
            <p>
                b : <label id="lblB" class="fondVert"></label>
            </p>

            <p id="pResultats"></p>
        </div>
        <script src="../js/Permutation.js"></script>
    </body>
</html>
```

```
A vous !
```

1.5 - Structure conditionnelle : le IF

1.5.1 - Objectif

Structure conditionnelle : le IF. Lorsqu'un code, une partie d'un code n'est exécuté que quand une condition est remplie alors il s'agit d'une structure conditionnelle.

Note : en JavaScript il existe d'autres instructions pour gérer une ou plusieurs conditions ; le switch, l'opérateur (ternaire) conditionnel (ceci sera vu dans le module JavaScript).

1.5.2 - Syntaxe

```
if (condition)
{
  Action a1;
  Action a2;
}
[else
{
  Action b1;
  Action b2; …
}]
```


1.5.3 - Un exemple

IF

Votre âge ? 18 OK Majeur

Affiche Majeur ou Mineur en fonction de l'âge saisi.

```
<!DOCTYPE html>
<!-- Un IF simple : SI la valeur saisie est >= valeur alors ... SINON ... -->
<html>
    <head>
        <title>IF.html</title>

        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
    </head>

    <body>
        <div>
            <label>Votre âge ?</label>
            <input type="text"   id="itAge" value="18" />
            <input type="button" id="btOK"  value="OK" />
            <label id="lblMessage"></label>
        </div>

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

```
/*
 * IF.js
 */

var itAge;
var btOK;
var lblMessage;

// ------------
function init() {
    itAge = document.getElementById("itAge");
    btOK = document.getElementById("btOK");
    lblMessage = document.getElementById("lblMessage");

    btOK.onclick = afficher;
} /// init



// --------------
function afficher() {
    /// Variables
    var age;
    var lsMessage = "";
```

```
    // IN
    age = itAge.value;
    // Traitement
    if (age >= 18) {
        lsMessage = "Majeur";
    }
    else {
        lsMessage = "Mineur";
    }
    // OUT
    lblMessage.innerHTML = lsMessage;
} /// afficher

window.onload = init;
```

cf page suivante une version améliorée

1.5.4 - Version sans variables globales

Cette version n'utilise pas de variables globales, on passe un argument à la fonction. Donc on utilise une fonction anonyme.

```
/*
 * IF.js
 */

// ------------
function init() {
    document.getElementById("btOK").onclick = function() {
        document.getElementById("lblMessage").innerHTML =
calculer(document.getElementById("itAge").value);
    };
} /// init



// --------------
function calculer(age) {
    var etat = "";

    if (age >= 18) {
        etat = "Majeur";
    }
    else {
        etat = "Mineur";
    }

    return etat;
} /// calculer



/// ---------
window.onload = init;
```

1.5.5 - Exercices sur le IF

Algorithme	Description
If else imbriqués	Contrôles de la saisie de l'âge On n'accepte que les « Tintin » donc les personnes âgées de 7 à 77 ans.
	Authentification
	Une année est-elle bissextile ?
If et condition complexe	Calculette

IFs

Votre âge ? 188 Valider Pas un peu trop âgé(e)!!!

IF - Authentification

UT : pb MDP : mdp GO
Message

IF - ControleSaisieCodeCarteBancaire

Votre code ? 1111 OK KO - Encore 2 essai(s)

IF - ControleSaisieCodeCarteBancaire

Votre code ? 1111 OK Votre carte est avalée!!!

1.6 - FOR

1.6.1 - Objectif

Structure itérative : le FOR.

Cette instruction permet de faire une boucle.
La variable de contrôle doit être un numérique de type ENTIER.

1.6.2 - Syntaxe

```
for (initialisation du compteur; condition de sortie;
incrémentation/décrémentation du compteur)
{
  instruction 1;
  instruction 2; …
}
```

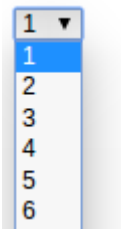
1.6.3 - Un exemple

Affiche 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-

```
<!DOCTYPE html>
<html>
    <head>
        <title>FOR.html</title>

        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
    </head>

    <body>
        <label id="lblResultat"></label>

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

```
/*
 * FOR.js
 */

// Déclaration des variables
var i; // Un compteur
var sTexte = ""; // Chaîne initialisée

// Boucle de 1 à 31
for (i = 1; i <= 31; i++) {
    sTexte += i + "-"; // Concaténation de la valeur du compteur et d'un -
}

// Affichage
document.getElementById("lblResultat").innerHTML = sTexte;
```


1.6.4 - Remplissage d'une liste déroulante (sans utiliser le DOM)




```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>ListeDeroulante.html</title>
    </head>

    <body>
        <select id="liste"></select>

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

```
/*
 * ListeDeroulante.js
 */

var options = "";

for (var i = 1; i <= 12; i++) {
    options += "<option>" + i + "</option>";
}
document.getElementById("liste").innerHTML = options;
```

1.6.5 - Exercices sur le FOR

Algorithme	Description
For-	Les nombres de 31 à 1
ForPasDe2	Les chiffres impairs
FonctionMultiplication	Multiplication sans *
Fonction Modulo	Réécrire la fonction Modulo
La liste des années	Afficher dans une liste déroulante les années de 1900 à l'année en cours

1.7 - TABLEAUX ORDINAUX

Les tableaux permettent d'associer plusieurs valeurs à un seul nom de variable, à la différence d'une variable scalaire qui associe une seule valeur à une variable.
Les tableaux JavaScript sont dynamiques ie la taille du tableau peut changer dynamiquement.

1.7.1 - Définition

Un tableau ordinal est un tableau à indices. C'est un index – non stocké – qui permet de gérer ses éléments. Le premier élément d'un tableau est d'indice 0, le dernier élément d'un tableau est d'indice nombre d'éléments – 1 (length – 1).

1.7.2 - Représentation

Seule la « colonne valeur » est stockée en mémoire.

Index	Valeur
0	Lundi
1	Mardi
2	Mercredi
3	Jeudi
4	Vendredi
5	Samedi
6	Dimanche

1.7.3 - Syntaxes

Initialisation d'un tableau

```
var tableau = new Array(1,3,5,7,9,11);
```

ou

```
var tableau = [1,3,5,7,9,11];
```

Nombre d'éléments d'un tableau

```
var nb = tableau.length;
```

Valeur d'un élément (récupération ou affectation)

```
var valeur = tableau[indice];
ou
tableau[indice] = valeur;
```

Boucle sur un tableau

```
for (var i = 0; i < tableau.length; i++) {
    console.log(tableau[i]);
}
```

Ajout d'un élément à la fin d'un tableau

```
tableau.push(valeur);
```

```
tableau[tableau.length] = valeur;
```

Note : pour plus de détails cf le support JavaScript.

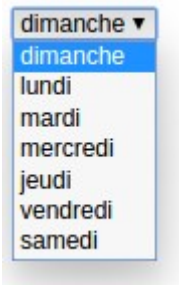
1.7.4 - Un exemple

Affiche le contenu du tableau, le nombre d'éléments et le 2ème élément.

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Tableau_0</title>

        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
    </head>

    <body>
        <div>
            <p id="pResultats"></p>
        </div>

        <script src="../js/Tableau_0.js"></script>
    </body>
</html>
```

```
/*
 * Tableau_0.js
 */

var pResultats = document.getElementById("pResultats");
var t = new Array(1, 3, 5, 7, 9, 11);

pResultats.innerHTML = "Contenu du tableau : " + t + "<br>";
pResultats.innerHTML += "Nombre d'éléments : " + t.length + "<br>";
pResultats.innerHTML += "2ème élément : " + t[1];
```


1.7.5 - La liste des jours




```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>ListeDeroulanteJours.html</title>
    </head>

    <body>
        <select id="liste">
        </select>

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

```
/*
 * ListeDeroulanteJours.js
 */
// Déclaration d'une variable de type String
var options = "";
// Déclaration d'un tableau de String
var tJours = new Array("dimanche", "lundi", "mardi", "mercredi", "jeudi",
"vendredi", "samedi");
// Boucle dans le tableau et concaténation par accumulation
for (var i = 0; i < tJours.length; i++) {
    options += "<option>" + tJours[i] + "</option>";
}
// Affichage
document.getElementById("liste").innerHTML = options;
```


1.7.6 - Exercices sur les tableaux

Algorithme	Description
Somme	Affichage de la somme des valeurs d'un tableau.
Moyenne	Affichage de la moyenne des valeurs d'un tableau.
Min	Affichage de la valeur min d'un tableau. **Les valeurs sont uniques**.
Max	Affichage de la valeur max d'un tableau. Les valeurs sont uniques.
PositionMin et PositionMax	Affichage de la position de la valeur min d'un tableau. Les valeurs sont uniques. Affichage de la position de la valeur max d'un tableau. Les valeurs sont uniques.
Rechercher une valeur	Saisie d'une valeur et affichage de sa position dans le tableau à valeurs uniques; si non trouvée affichage de -1.
Rechercher une valeur (version optimisée)	Saisie d'une valeur et affichage de sa position dans le tableau à valeurs uniques; si non trouvée affichage de -1.
Rechercher une valeur	Saisie d'une valeur et affichage de **ses** positions dans le tableau à **valeurs non uniques**; si non trouvée affichage de -1.
Recherche multiple	Rechercher des valeurs dans un tableau à **valeurs uniques** en fonction des valeurs présentes dans un autre tableau. Par exemple [3,5] dans [1,2,3,4,5,6,7,8,9,10].
Tableau2D : somme	
Tableau 2D : Rechercher une valeur	
Tableau2D : somme par ligne	
Tableau2D : somme par colonne	
Calendriers	Des tableaux 2D.

Note :

Création d'un tableau 2D

```
var dataArray = [["matin", 13], ["midi", 21], ["soir", 17]];
```

1.8 - WHILE

1.8.1 - Objectif

Structure itérative : le WHILE. « Tant que » en français.

Cette instruction permet de faire une boucle.
Tant que la condition est VRAIE on boucle.
Si la condition est à FAUX avant l'entrée dans la boucle on n'entre pas dans la bouche.
Dans la boucle quand la condition passe à FAUX on sort de la boucle.
La variable de contrôle, à la différence du FOR, peut être de n'importe quel type (entier, booléen, chaîne, ...).

1.8.2 - Syntaxe

```
initialisation pour que la condition passe (éventuellement) à TRUE;

while (condition)
{
  action 1;
  action 2;
  action qui modifie la condition; // Pour la faire passer à FALSE
}
```


1.8.3 - Un exemple

Affiche 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-

Donc à comparer à l'exemple du FOR.

```
<!DOCTYPE html>
<html>
    <head>
        <title>While_1</title>

        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
    </head>

    <body>
        <label id="lblResultat"></label>

        <script src="../js/While_1.js"></script>
    </body>
</html>
```

```
/*
 * While_1.js
 */

// Déclaration des variables
// Chaîne initialisée à chaîne vide
var sTexte = "";

// Un compteur; cette valeur permet d'entrer dans la boucle
var i = 1;

// Boucle de 1 à 31
while (i <= 31) {
    // Concaténation de la valeur du compteur et d'un -
    sTexte += i + "-";
    // i va passer de 1 à 2 puis de 2 à 3 etc
    // Cela permettra de sortir de la boucle
    i++;
}

// Affichage
document.getElementById("lblResultat").innerHTML = sTexte;
```


1.8.4 - Exercices sur le WHILE

Algorithme	Description
Élévation à la puissance	
Factorielle	
Recherche optimisée dans un tableau ordinal	
Cf aussi les exercices sur les chaînes de caractères	
Fonction multiplication	Multiplication sans le symbole *
Fonction Division	Division entière sans le symbole /

1.9 - Chaînes de caractères

Une chaîne de caractères est un tableau de caractères en lecture seule (ce qui veut dire qu'il est impossible de ré-affecter un caractère via un indice comme pour un tableau).

1.9.1 - Syntaxes

Déclaration d'une chaîne de caractères et affectation d'une valeur.

```
var chaine = "";
```

Création d'une chaîne de caractères (instanciation avec affectation d'une chaîne vide).

```
var chaine = new String("");
```

Calcul de la longueur de la chaîne de caractères.

```
var longueur = chaine.length;
```

Récupération d'un caractère comme pour la récupération d'un élément d'un tableau. L'indice du premier caractère est 0.

```
var car = chaine[indice];
```

Concaténation (+)

```
var phrase = "Bonjour " + " monsieur Dupont";
```

Concaténation « par accumulation » (+=)

```
phrase += " et madame Tuche";
```

Note : pour plus de détails cf le support JavaScript.

1.9.2 - Un exemple

Affiche la chaîne de caractères, le nombre de caractères de cette chaîne, le 2ème caractère.

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>chaine_1</title>

        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
    </head>

    <body>
        <div>
            <p id="pResultats"></p>
        </div>

        <script src="../js/chaine_1.js"></script>
    </body>
</html>
```

```
/*
 * chaine_1.js
 */

var pResultats = document.getElementById("pResultats");
var chaine = "azerty";

pResultats.innerHTML = "Contenu de la chaine : " + chaine + "<br>";
pResultats.innerHTML += "Longueur de la chaîne : " + chaine.length + "<br>";
pResultats.innerHTML += "2ème caractère : " + chaine[1];
```


1.9.3 - Autre exemple

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>chaine_2</title>

        <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0, maximum-scale=1.0" />
        <meta charset="utf-8" />
    </head>

    <body>
        <h1>Ouvrez la console !!!</h1>
        <div>
            A la verticale !
            <p id="pResultats"></p>
        </div>

        <script src="../js/chaine_2.js"></script>
    </body>
</html>
```

```
/*
 * chaine_2.js
 */
var chaine = "azerty";

/*
 * Chaîne à la verticale
 */
var resultat = "";
for (var i = 0; i < chaine.length; i++) {
    resultat += chaine[i] + "<br>";
}

document.getElementById("pResultats").innerHTML = resultat;
```

1.9.4 - Quelques méthodes (fonctions) sur les chaînes de caractères

Propriété/Méthode	Description
Ch.length	Renvoie la longueur de la chaîne. Attention pas de parenthèses.
Ch.substr(départ[, longueur])	Renvoie une sous-chaîne. Par exemple "azerty".substr(0, 3) renvoie "aze". Par exemple "azerty".substr(2) renvoie "erty". Si départ est négatif JS commence par la fin, par exemple "azerty".substr(-3, 3) renvoie "rty".
Ch.toUpperCase()	Renvoie la chaîne en majuscules.
Ch.toLowerCase()	Renvoie la chaîne en minuscules.
Ch.charAt(indice)	Renvoie le caractère à la position indice.
Ch.split("séparateur")	Explose une chaîne de caractères dans un tableau.
String.fromCharCode(code)	Renvoie le caractère correspondant au code. String.fromCharCode(65) → 'A'
String.charCodeAt(caractère)	Renvoie le code ASCII d'un caractère. "A".charCodeAt(0) → 65

La méthode split()

Dans cet exemple chaque mot de la phrase (donc séparé par un espace) va être placé dans un élément du tableau.

```
<!DOCTYPE html>
<html>
    <head>
        <title>explode.html</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>
            <h1>explode.html</h1>
            <h3>Ouvrez la console (F12)</h3>
        </div>

        <script>
            var chaine = "Il fait très chaud aujourd'hui";
            var tableau = chaine.split(" ");
            for (var i = 0; i < tableau.length; i++) {
                console.log(tableau[i]);
            }
        </script>
    </body>
</html>
```

Le résultat du split() :

Il
fait
très
chaud
aujourd'hui

1.9.5 - Exercices sur les Chaînes

Algorithme	Description	Niveau
Implode	Tableau ordinal → Chaîne	1
MajPremier	Met la première lettre en majuscule et les autres en minuscule (UCFirst)	1
RechercherCaractere	Rechercher la position d'un caractère dans une chaîne de caractères 2 variantes : le dernier trouvé, le premier trouvé (indexOf() ou lastIndexOf())	1
RemplacerCaractere	Remplacer un caractère par un autre dans toute la chaîne	2
LTrim	Élimine les espaces à gauche	2
Rtrim	Élimine les espaces à droite	2
Trim	Élimine les espaces à gauche et à droite	2
Underscore2camel	Camélise : nom_du_client → nomDuClient. Split() autorisé	2
Extraction	Extrait une sous-chaîne d'une chaîne (réécrire substr)	3
Explode	Chaîne → Tableau ordinal	3
Camel2underscore	NomDuClient → nom_du_client	3
NomPropre	Met en majuscule la première lettre de tous les mots d'un nom (UCWords ou presque) par exemple Louis-Napoléon Bonaparte, Valéry Giscard d'Estaing	3
Trim Interne	Élimine les espaces superflus dans la chaîne	3
Fréquence de chaque caractère Alpha dans un texte	Résultats dans un tableau à clés (donc cf tableau à clés sur internet)	4
RechercherMot		4
Nombre de mots dans un texte		5
RemplacerMot		5
Fréquence de chaque mot dans un texte	Résultats dans un tableau à clés (donc cf tableau à clés sur internet)	5

1.10 - Annexe : De la console pure et dure !

Vous pouvez taper directement du code à la console et l'exécuter !!!

```
> var valeur = 3;
  var multiplicateur = 5;
  var r = 0;

  var i = 1;
  while (i <= multiplicateur) {
      r += 3;
      i++;
  }

  console.log("R&sultat de la multiplication de " + valeur + " par " + multiplicateur + " est de : " + r);
  R&sultat de la multiplication de 3 par 5 est de : 15
```

Le code :

```
var valeur = 3;
var multiplicateur = 5;
var r = 0;

var i = 1;
while (i <= multiplicateur) {
    r += 3;
    i++;
}

console.log("R&sultat de la multiplication de " + valeur + " par " +
multiplicateur + " est de : " + r);
```

