**INSFP OULED FAYET**

**Spécialité :** Développeur web et mobile / S2

**Module :** Les interfaces utilisateurs interactives

# Chapitre 1 : Introduction au langage JavaScript

**Introduction**: Aujourd’hui, le développement web est en croissance continue, car, nous pouvons créer des sites web ou des applications qui marchent sur des ordinateurs, tablettes ou Smartphones.

**1- Définition :** Un **langage informatique** est un ensemble de caractères, symboles et de règles à appliquer pour la conception et la mise en œuvre d'un [système d'information](https://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27information).

On distingue plusieurs types, comme les l**angages de conception (**UML, Merise..), de **programmation** (Java, PHP..), de **description** (HTML, CSS..), de **requête** (SQL, OQL..), etc.

**2- Classification des langages de programmation :** La classification se fait selon plusieurs **critères** dont on site :

**2.1- Langages interprétés / compilés :** Un **script** écrit avec un **langage interprété** a besoin d'un interpréteur pour traduire au fur et à mesure les instructions du script. Si on a des instructions erronées, la partie erronée ne va pas s’exécutée, mais la partie correcte s’exécute de façon normale.

**Exemples** : HTML, CSS, JavaScript, Python …etc.

**-** Un **programme** écrit avec un **langage compilé** va être traduit une fois pour toute si on n’a pas d’erreurs par un compilateur, afin de générer un nouveau fichier exécutable. Il a comme avantages de garantir la sécurité du code source et la rapidité. **Exemples** : Pascal, Delphi, C++, Python …etc.

**- Interpréteur/ compilateur :** C’est un logiciel permettant **d’interpréter/ compiler** un programme en vérifiant si il n’y a pas d’erreurs (traduire le code source en code machine).

### Remarques :

### a- Certains langages appartiennent aux deux catégories interprété/compilé (LISP, Java, Python, ..) car dans certaines cas, on a besoin d’une phase de compilation pour créer certaines applications, et dans d’autres cas, on a besoin d’un interpréteur comme le cas du web.

**b**- On appelle un **code**, un **programme** (langage C, Pascal) ou bien un **script** (HTML, JavaScript).

**c-** On **exécute** un code par la machine (moteur), soit c’est un programme ou bien un script.

**2.2- Langages fonctionnels/ Orientés Objet :** On distingue deux familles de langages de **programmation**, selon la manière dont les instructions sont traitées :

### - Langage fonctionnel : Parfois appelé langage procédural ou déclaratif, est un langage dans lequel le programme est construit de fonctions/procédures prédéfinies ou déclarées, dont on fait appel au besoin.

### Exemples : Pascal, C…etc.

**- Langages Orientés Objet :** La programmation orientée objet est bien dirigée à l’usage de la notion d’objet et de classe. Une classe est une entité qui possède un nom, des attributs et des méthodes, or que l’objet est une instance de la classe. (On peut créer un objet sans passer par la classe).

**Exemples :** Delphi, C#, C++, JavaScript, PHP, Java…etc.

**3- Les types de développeurs :**

**I- Développeur front-end :** Il est censé de travailler dans les champs de compétences suivants :

**a- Design** (web designer). **b-Développement** HTML**,** CSS, JavaScript (web master).

Permet d’améliorer **l’aspect visuel (**[**front-end :** façade du site**)**](https://fr.wikipedia.org/wiki/D%C3%A9veloppement_web_frontal)sur le navigateur **(client side :**côté client**)**.

**II- Développeur back-end :** Censé de travailler dans les champs de compétences suivants :

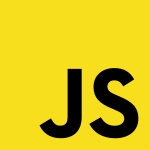
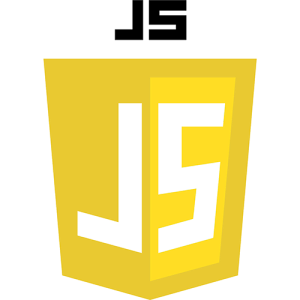
**a- Hébergement** du site web. **b-Administration** de la base de données du site web.

**c- Développement** Java, C++, Python, PHP, JavaScript, etc.

Permet de gérer le site **(**[**back-end**](https://fr.wikipedia.org/wiki/Backend)**:** partie invisible du site**)** sur le serveur **(server side :** côté serveur**)**.

**III- Développeur full-stack :** Capable d’intervenir aussi sur le front-end que le back-end.

**4- JavaScript (JS) :** JavaScript est un langage de programmation créé en 1995 par [Brendan Eich](https://fr.wikipedia.org/wiki/Brendan_Eich) pour le compte de la [Netscape Communications Corporation](https://fr.wikipedia.org/wiki/Netscape_Communications_Corporation), et standardisé sous les noms ECMA-262 et ISO/CEI 16262.

Une grande majorité des [sites web](https://fr.wikipedia.org/wiki/Site_web) l'utilisent, et la majorité des [navigateurs web](https://fr.wikipedia.org/wiki/Navigateur_web) disposent d'un [moteur JavaScript](https://fr.wikipedia.org/wiki/Moteur_JavaScript) dédié pour l'interpréter. La version actuelle de JavaScript est la version 13 édition - ECMA-262, publiée en juin 2022).

**Logo de JavaScript**

JavaScript support la programmation fonctionnelle et orientée objet.

JavaScript été utilisé à l’origine pour rendre dynamiques les pages web. Là où le HTML5 et le CSS3 permettent respectivement l’affichage et la mise en forme de vos éléments sur votre page web, le JavaScript rendra cette page interactive.

**Rôle de JavaScript dans le web :** Nous avons quelques exemples comme suite :

- Proposer un diaporama (changement d'image toute les x secondes, boutons pour mettre en pause, aller

à l'image précédente / suivante).

- Créer des applications interactives (par ex. tests et quiz).

- Vérification de formulaires, etc.

**5- Emplacement de JavaScript :** Nous avons deux possibilités pour écrire un code JavaScript.

**a-** A l’intérieur d’un document HTML.

|  |  |
| --- | --- |
|  |  |

**b-** Dans un fichier séparé portant l’extension **.js**.

**Remarque :** Il est recommandé d’utiliser un fichier JavaScript séparé pour des raisons de performances.

**a-A l’intérieur d’un document HTML :** Nous avons un exemple d’un code JavaScript permettant d’afficher un message dans la page web.

**<body>**

**<script language= "JavaScript" type="text/javascript">  
 document.write("Ceci est le message 1")   
</script>**

**</body>**

**b- Dans un fichier JavaScript appart :** Ici, nous allons créer un fichier dans notre éditeur de texte et l’enregistrer au format «**.js** », en plus, on doit lier ce fichier avec le fichier HTML.

- Code dans le fichier HTML :

**<script src="fichier.js">**

**</script>**

- Code dans le fichier JavaScript :

**document.write("Ceci est le message 2 ")**

**6- Notions de base sur JavaScript :** Nous avons les notions suivantes :

**a/** Un **identificateur** est le nom d’une variable, d’une fonction, ou d’une propriété.

**b/** Les minuscules et les majuscules ne sont pas équivalentes (JS est sensible à la casse ‘case sensitive’).

**Exemple** : **document.write("Bonjour");** n’est pas l’équivalent de : **Document.write("Bonjour");**

**c/** Pour déclarer une **variable**, on doit respecter les contraintes suivantes :

- Ne contient pas d’espace.

- Commence par une lettre, \_ ou $.

- Ne doit pas être un mot réservé comme par exemple : break, do, case, else, new, var, return, void, etc.

Les types de variables peuvent être :

**- Réels ou entiers**: 3.14 ou 200.

**- Chaînes de caractères** : "Cours de Programmation Web".

**- Logiques ou booléennes** : true/ false (vrai ou faux).

- **Null** : Qui signifie pas de valeur ou d'objet.

**- …etc.**

**d/ Les opérateurs :**

**- Opérateur d’affectation** : (**A = B**) mettre le contenu de B dans A.

1. **- Opérateurs arithmétiques** : **+, -, \*, /, %:** Modulo **++:** Incrémentation **--:** Décrémentation.

n = n + 1 ⇔n++ ⇔++n ⇔n += 1 (Incrémentation). Même principe pour la décrémentation.

a = a + b ⇔a += b, a = a - b ⇔a -= b, a = 2 \* a ⇔ a \*= 2 c=a% b.

**- Opérateurs de comparaison** : **== , !=, <, <=, >, >= ,** (=== : comparer l’égalité du type et du contenu entre deux variables), ( !== : comparer la différence du type et du contenu entre deux variables).

**Remarque** : Il faut s’assurer de la compatibilité des types lors des comparaisons.

**Opérateurs logiques** :(&& : et logique), (|| : ou logique), (! : négation).

**Opérateur de chaîne de caractères (+)** : Permet de **concaténer** deux chaînes de caractères.

**Exemple**: "Java" + "Script" donnera "JavaScript".

**Remarque :** La priorité des opérateurs est comme suite : (1) les parenthèses (2) négation (3) \*,/ ,modulo ,et logique (4) +,-,ou logique (5)> ,< , <=, >= , != , ==.

Pour deux opérateurs de même priorité, on commence par le plus à gauche.

**Exemple** : (8+ (5-3)\*8)/4 \* 3. Le résultat est 18.

**e/** Pour écrire un commentaire, vous pouvez utiliser le **double slash ( //)** au cours d’une ligne, ou bien utiliser /\*…\*/ sur plusieurs lignes.

**Exemple** :   
<script language= "JavaScript">

//voici un commentaire sur une ligne  
/\* Voici un commentaire

sur plusieurs lignes \*/  
</script>

**Remarques** :

**1-** Pour l’HTML : **<! - -** Ceci est un commentaire en HTML sur une ligne -->

**<! - -**Ceci est un commentaire en HTML

sur plusieurs lignes…………..-->

**2-** Le raccourci clavier **Ctrl+/** permet de créer ou d’annuler un commentaire sur une ligne, pour les langages HTML et JavaScript.

**f/** Pour sauter la ligne en JavaScript, on peut utiliser la balise <br/> dans la chaîne de caractère.

**document.write("Bonjour <br/> tout le monde");**

**g/** On peut utiliser une variable non déclarée.

**Exemple** : **var x = 10;**

**y=7 ;**

**document.write(x, y);**

**INSFP OULED FAYET**

**Spécialité :** Développeur web et mobile / S2

**Module :** Les interfaces utilisateurs interactives

**TP n° 1**

Ecrire en JavaScript, les scripts permettant de :

**1-** Faire la multiplication de deux nombres entiers.

**2-** Donner le triple d’un nombre entier.

**3-** Permuter le contenue de deux variables réels.

**4-** Renvoyer la racine carrée d’un entier.

**5-** Donner la moyenne de deux nombres entiers, cette moyenne doit être arrondie.

**6-** Donner la partie entière de la moyenne de deux nombres entiers.

**Solution** :

**1-** Multiplication de deux nombres entiers.

var a=3, b=7;

c=a\*b;

document.write(c);

**2-** Triple d’un nombre entier.

var a=10;

document.write(a\*3);

**3-** Permutation de nombre réels;

var a=3.2, b=7.5;

c=a;

a=b;

b=c;

document.write(a+'<br> '+b) ;

**4-** Racine carrée d’un entier.

var a=27;

b=Math.sqrt(a);

document.write(b);

**5-** Moyenne arrondie de 2 nombres entiers.

var a=8, b=13;

c=Math.round((a+b)/2);

document.write(c);

**6-** Partie entière de la moyenne de 2 entiers.

var a=8, b=13;

c=Math.trunc((a+b)/2);

document.write(c);

**INSFP OULED FAYET**

**Spécialité :** Développeur web et mobile / S2

**Module :** Les interfaces utilisateurs interactives

**Chapitre 2 : Les instructions conditionnelles et répétitives**

JavaScript définit plusieurs structures de contrôle afin de réaliser des conditions et des boucles.

**1- Les conditions :** Il existe deux types de structures de conditions.

**1.1 L’instruction if** : Son rôle est de vérifier une condition et de réaliser des traitements en cas de validité. La syntaxe est la suivante :

**if** (condition)

{

Bloc **1** d’instructions

}

**else**

{

Bloc **2** d’instructions

}

**Remarques** :

**a**- Il est possible d’utiliser if sans else.

**b-** Il est possible d’utiliser autant de if else imbriqués que nécessaire.

**c-** L’**opérateur conditionnel** peut remplacer if-else, selon la syntaxe de l’exemple suivant :

**x = (y > z) ? 5 : 3 ;** Si (**y** > **z**) alors **x**=**5**, sinon **x**=**3**.

**Exemple 1 :** Ecrire un script en JS qui détermine si un nombre entier est supérieur à 10.

**var** x =17;

**if** (x>10)

{

document.write(x+ ' est supérieur à dix' );

}

**Exemple 2 :** Ecrire un script en JS qui détermine si un nombre entier est supérieur ou inférieur à 10.

**var** x =17;

**if** (x>10)

{

document.write (x+ ' est supérieur à dix' );

}

**else**

**{**

document.write (x+ ' est inférieur à dix' );

}

**Exemple 3 :** Ecrire un script en JS qui détermine si un nombre entier est supérieur, inférieur ou égal à 10.

**var** x =17;

**if** (x>10)

**{**

document.write (x+ ' est supérieur à dix' );

**}**

**else**

**{**

**if** (x==10)

**{**

document.write (x+ ' est égal à dix' );

**}**

**else**

{

document.write (x+ ' est inférieur à dix' );

}

**}**

**1.2 Les instructions switch, case et default** : Cela permet de faire des traitements en fonction de la valeur d’une variable (variable = valeur) avec la syntaxe suivante :

**switch** (variable)

{

**case** valeur 1:

Bloc **1** d’instructions

**break**;

**case** valeur 2:

Bloc **2** d’instructions

**break**;

**default**:

Bloc **3** d’instructions

}

**Exemple :** Ecrire un script en JS qui renvoie le semestre de formation d’un stagiaire.

var s=2;

**switch** (s)

{

**case** 1:

document.write ('Vous êtes en semestre '+s);

**break**;

**case** 2:

document.write ('Vous êtes en semestre '+s);

**break**;

**default**:

document.write ('Vous êtes en semestre '+s);

}

**2-Les instructions répétitives :**

**2.1-L’instruction for :** Cette instruction permet d’incrémenter une variable à partir d'une valeur initiale jusqu'à une valeur finale et d'exécuter des instructions entre chaque incrémentation. La syntaxe est la suivante :

**for** (condition de début ; condition de fin ; incrémentation/décrémentation)

**{**

Blocd’instructions

**}**

**Exemple :** Ecrire un script en JS qui calcule le factoriel d’un entier. (Exp : factoriel (3)= 3\*2\*1=6).

**var** n=3; f=1;

**for** ( var i=1; i<=n; i++ )

**{**

f=f\*i;

**}**

**document.write**("Le factoriel de "+n+" est "+f );

**2.2-L’instruction while :** Ce type de boucle, contrairement à la précédente, évalue une condition avant d'exécuter des instructions comme suite :

**while** (condition)

**{**

Blocd’instructions

**}**

**Exemple :** Reprendre l’exercice du factoriel avec la boucle while.

**var** n=3; f=1; i=1;

**while** ( i<=n )

{

f=f\*i;

i=i+1;

}

**document.write**("Le factoriel de "+n+" est "+f);

**2.3-L’instruction do :** Cette boucle effectue les instructions placées entre deux accolades et évalue à chaque fois une condition de type booléenne avec while avant de continuer la boucle pour décider l'arrêt.

La syntaxe est comme suite :

**do**

{

Blocd’instructions

}

**while** (condition)

**Exemple :** Reprendre l’exercice du factoriel avec la boucle do**.**

**var** n=3; f=1; i=1;

do

{

f=f\*i;

i=i+1;

}

**while**(i<=n)

**document.write** ("Le factoriel de "+n+" est "+f);

**Remarques :**

**1-** Pour la boucle while, les instructions peuvent ne pas être exécutées si la condition n’est pas remplit, contrairement à la boucle do où au moins une instruction est exécutée.

**2-** Il est possible d’imbriquer les instructions conditionnelles et répétitives selon le besoin.

**3-** Pour la boucle for, il est possible d’utiliser plusieurs compteurs en même temps.

**Exemple**: **for (i=1, j=3 ; i<=n, j<=m ; i++, j++)**

**4-** Il est possible d’utiliser **break** avec la boucle **for** pour arrêter immédiatement la boucle avec une certaine condition comme le montre l’exemple ci-dessous.

**Exemple**: Afficher les tous les multiples de 3 qui sont inférieurs à 10.

**var i;**

**for (i=1; i<20; i++)**

**{**

**if (3\*i > 10)**

**break;**

**document.write(3\*i);**

**}**

**5-** Il est possible d’utiliser **continue** avec la boucle **for** pour pouvoir continuer la boucle dans certains cas exceptionnels, comme le montre l’exemple ci-dessous.

**Exemple**: Afficher l’inverse des nombres dans l’intervalle [-3, 3] (1 divisé par 0 n’est pas possible).

**var i;**

**for(i=-3; i<=3; i++)**

**{**

**if (i == 0)**

**continue;**

**document.write(1/i+);**

**}**

**NB : NaN** signifie (Not a Number) pas un nombre, message qui peut être affiché en divisant un nombre sur une chaine de caractères.

**INSFP OULED FAYET**

**Spécialité :** Développeur web et mobile / S2

**Module :** Les interfaces utilisateurs interactives

**TP n° 2**

Ecrire en JavaScript, les scripts permettant de :

**1-** Vérifier si une année donnée est bissextile ou non, en utilisant if-else.

L’année bissextile est une année spéciale contenant un jour supplémentaire dans une année(366).

Une année est considérée comme bissextile si elle est divisible par 4 mais non divisible par 100.

**2-** Trouver un maximum entre trois nombres en utilisant if-else imbriquée.

**3-** En utilisant l’instruction switch, afficher en lettre un nombre donné en chiffre de 1 jusqu’à 3 et afficher le message « supérieur à trois » pour les nombres supérieur à 3.

**4**- Calculer la somme de N premier nombre entier avec les trois types de boucles.

**5-** En utilisant la boucle for, calculez la puissance n d’un entier x (**xn**).

**6-** Afficher les nombres impairs compris entre 1 et N, en utilisant la boucle for.

**7-** Afficher les nombres pairs compris entre 1 et N, en utilisant la boucle while.

**Solution** :

**1-** Vérification des années bissextiles.

var annee=2020;

if (annee%4==0 && annee%100!=0)

{

document.write (annee+" est une

année bissextile");

}

else

{

document.write (annee+" est une année

non bissextile");

}

**2-** Maximum entre trois nombres.

var max, x = 15 ; y =9 ; z = 7 ;

if (x > y && x > z )

{

max=x;

}

if (y > x && y > z )

{

max=y;

}

if (z > y && z > x )

{

max=z

}

document.write (max);

**3-** Affichage en lettre d’un nombre donné

en chiffre.

var n=3;

switch (n)

{

case 1:

document.write ('Vous avez saisi le

nombre un');

break;

case 2:

document.write ('Vous avez saisi le

nombre deux');

break;

case 3:

document.write ('Vous avez saisi

le nombre trois');

break;

default:

document.write ('Vous avez saisi un

nombre supérieur à trois');

}

**4**- **a/** Calcul de somme de N premier nombre

entier avec la boucle for.

var n=3; s=0;

for ( var i=1; i<=n; i++ )

{

s=s+i;

}

document.write ("La somme est "+s);

**4**- **b/** Calcul de somme de N premier nombre

entier avec la boucle while.

var n=3; s=0; i=1;

while ( i<=n )

{

s=s+i;

i=i+1;

}

document.write("La somme est "+s);

**4**- **c/** Calcul de somme de N premier nombre

entier avec la boucle do.

var n=3; s=0; i=1;

do

{

s=s+i;

i=i+1;

}

while(i<=n)

document.write ("La somme est "+s);

**5-** Calcul de puissance n d’un entier x.

var x=5; n=3; p=1;

for ( i=1 ; i<=n ; i++ )

{

p=p\*x;

}

document.write (x+" puissance "+n+"

est "+p);

**6-** Affichage des nombres impairs entre 1 et N.

var n =10;

for ( var i=1; i<=n; i=i+2 )

{

document.write("<br>"+i);

}

**7-** Affichage des nombres pairs entre 1 et N.

var n=10; i=1;

while ( i<=n )

{

if (i%2==0)

{

document.write("<br>"+i);

}

i=i+1;

}

**INSFP OULED FAYET**

**Spécialité :** Développeur web et mobile / S2

**Module :** Les interfaces utilisateurs interactives

**Chapitre 3 : Les fonctions**

**1-Définition** : Une fonction est un morceau de code qui permet de réaliser une tâche bien définie.

Une fonction a 3 éléments comme suite :

- **Nom** de la fonction, avec lequel on fait appel à la fonction.

- Liste et types d’**arguments** (paramètres d’entrée).

- Valeur à renvoyé par la fonction (paramètre de sortie).

La syntaxe **globale** pour **déclarer** une fonction est comme suite :

**function** nom (v1,…,vn) **//** v1,…,vn : variables (Arguments).

**{**

Bloc d’instructions

**return** **R** ; // **R** : résultat (valeur) à renvoyer.

**}**

- Le nom d'une fonction doit être différent de celui d'un des mots réservés au langage JS.

- Une fonction peut être appelée par une autre fonction ou par le script principal.

- Une fonction peut dépendre d'une structure de boucle, de condition, etc.

**Remarque :** En programmation, on distingue entre les procédures et les fonctions.

Une procédure ne doit jamais renvoyer une valeur (modification de variable, etc).

Une fonction doit toujours renvoyer une valeur (renvoyer le factoriel d’un nombre, etc).

En JavaScript, on utilise le même mot clé (**function**) pour les procédures et les fonctions.

**2-Structure des fonctions**:Une fonction peut avoir 4 formes comme suite :

**2.1-** Fonction **avec** arguments et **avec** valeur à renvoyée.

**Exemple :** Ecrire un script JS qui appelle une fonction permettant de calculer la somme de 2 nombres.

**function** somme (a,b)

**{**

c=a+b;

**return** c ;

**}**

**document.write** (somme(5, 8));

**2.2-** Fonction **avec** arguments et **sans** valeur à renvoyée.

**Exemple :** Ecrire un script JS qui appelle une fonction qui permute le contenu de 2 variables.

**var** x=5, y=8 ;

**function** permute (a,b)

**{**

c=a;

a=b;

b=c;

**document.write** (a+'<br>'+b);

}

**permute**(x,y);

**Remarque**: Une fonction retourne une valeur avec (**return**) et pas avec un affichage.

**2.3-** Fonction **sans** arguments et **avec** valeur à renvoyée.

**Exemple :** Ecrire un script JS qui appelle une fonction qui renvoie la multiplication de 2 nombres.

**var x=5, y=8 ;**

**function mult ()**

**{**

**c=x\*y;**

**return c ;**

**}**

**document.write (mult());**

**2.4-** Fonction **sans** arguments et **sans** valeur à renvoyée.

**Exemple 1 :** La fonction **onClick**() qui déclenche un événement en cliquant sur la souris.

**Exemple 2 :** Ecrire un script JS qui appelle une fonction qui affiche un message.

**function hello**()

{

**document.write** ('Bonjour tout le monde');

}

**hello**();

**3- Visibilité des Variables** : On appelle **visibilité** des variables la possibilité d’utiliser des variables depuis diverses parties du script.

Une Variable est dite **locale** si elle est déclarée avec le mot clé **var** et à **l’intérieur** d’une fonction. Alors elle n’est connue qu’à l’intérieur de cette fonction.

**Exemple** : Utiliser une fonction avec une **variable locale x** pour afficher un message.

**function** message()

{

**var** x = 'Bonjour';

**document.write** (x);

}

y=x; // x n’est pas visible, alors ne peut pas être affectée à y.

**document.write** (y); // y ne s’affiche pas.

Une variable est dite **globale** si :

**a/** Elle est déclarée **sans** le mot clé **var** et à l’intérieur d’une fonction qui aura été **exécutée**.

**Exemple** : Utiliser une fonction avec une **variable globale x** pour afficher un message.

**function** message()

{

x = 'Bonjour';

**document.write** (x);

}

message() ;

y=x; **//** x est visible, alors peut être affectée à y.

**document.write** (y);

**b/** Elle est déclarée tout au début du script en dehors et avant toute fonction, soit avec ou sans var. elle peut être utilisée dans toute la page HTML et à l’intérieur des fonctions éventuelles.

**Exemple**: Ecrire une fonction qui calcule le périmètre d'un carré et une autre fonction qui calcul la

surface en utilisant la même variable **globale**.

**c=5;**

**function** périmètre ()

{

p=c\*4;

return (p) ;

}

**function** surface ()

{

s=c\*c;

return(s) ;

}

**document.write**(périmètre (c)+ '<br>');

**document.write**(surface(c));

**4- La récursivité** : La récursivité est une technique de programmation qui remplace les instructions de boucle (while, for, etc.) par des appels de fonction.

**Exemple**: Ecrire une fonction récursive permettant de calculer les N premiers nombres d’un entier.

**function** som\_n\_nombre(n)

{

**return** (n==1?1: n+som\_n\_nombre(n-1)) ; // Si n vaut 1, la somme vaut 1

} // Sinon vaut n + n-1

**document.write**(som\_n\_nombre(3));

**5- Les fonctions prédéfinies**: Il existe en JavaScript des fonctions prédéfinies, voici des exemples :

###### **Fonctions pour entrer et afficher des données :**

* **prompt** () : Entrer une valeur qui pourra être récupérée dans une variable qui sera traitée ailleurs.
* **alert**() : Affiche la sortie sur une boîte de dialogue du navigateur.
* **confirm**() : Ouvre une boîte de dialogue oui/non et renvoie true/false en fonction du clic.
* **console**.**log**() : Affiche la sortie sur la console du navigateur, utile pour le débogage.
* **document**.**write**() : Ecrit directement dans le document HTML.

###### **Fonctions globales :**

* **isNaN**() : Détermine si une valeur est NaN((Not a Number) ou non.
* **Number**() : Convertit l’argument en nombre représentant la valeur de l’objet. Si la valeur ne peut pas être convertie en nombre, NaN est renvoyé.
* **eval**() : Évalue le code JavaScript représenté sous forme de chaîne.
* **encodeURI**() : Encode un URI en UTF-8.
* **decodeURI**() : Décode un URI créé par encodeURI() ou d’autres fonctions.
* **decodeURIComponent**() : Décode un composant URI.
* **parseInt**() : convertit une chaîne de caractères en entier.
* **parseFloat**() : convertit une chaîne de caractères en nombre à virgule flottante.
* **isFinite**() : Détermine si la valeur transmise est un nombre fini.

###### **Fonctions mathématiques :**

* **Math.random**() : Retourne un nombre aléatoire compris entre 0 et 1.
* **Math.round**(x) : Arrondie la valeur de x à son entier le plus proche.
* **Math.pow**(x, y) : X à la puissance de y.
* **Math.exp**(x) : Exponentielle de x.
* **Math.log**(x) : Le logarithme naturel (base E) de x.
* **Math.max**(x, y, z, …, n) : Retourne le maximum.
* **Math.min**(x, y, z, …, n) : Retourne le minimum.
* **Math.abs**(x) : Retourne la valeur absolue de x.
* **Math.sqrt**(x) : La racine carrée de x.
* **Math.cos**(x) : Le cosinus de x.
* **Math.sin**(x) : Le sinus de x.
* **Math.tan**(x) : La tangente de x.
* **Math.ceil**(x) : Arrondie la valeur de x à l’entier le plus proche.

**Fonctions de l’objet String**

* **match**() : Récupère les correspondances d’une chaîne par rapport à un motif de recherche
* **search**() : Recherche dans une chaîne de caractères la valeur spécifiée et renvoie la position de la correspondance.

* **toLowerCase**() : Convertit une chaîne de caractères en minuscules.
* **toUpperCase**() : Convertit une chaîne de caractères en majuscules.
* **indexOf**() : Fournit la position du première occurrence d’un texte spécifié dans une chaîne.
* **charAt**() : Retourne le caractère à la position spécifiée dans la chaîne.
* **concat**() : Concatène deux ou plusieurs chaînes en une seule.
* **charCodeAt**() : Retourne la valeur unicode d’un caractère à la position spécifiée.
* **indexOf**() : Fournit la position du première occurrence d’un texte spécifié dans une chaîne.
* **lastIndexOf**() : Fournit la position du dernière occurrence d’un texte spécifié dans une chaîne.
* **slice**() : Extrait une partie d’une chaîne et la renvoie sous forme de nouvelle chaîne.
* **split**() : Découpe une chaîne de caractère en un tableau de sous-chaînes en fonction d’un séparateur.
* **substr**() : Comme slice() mais extrait une sous-chaîne en fonction du nombre de caractères spécifié.
* **substring**() : Comme slice() mais ne peut accepter les index négatifs.
* **replace**() : Recherche et remplace un texte spécifique dans une chaîne.

###### **Fonctions de gestion des tableaux (Array)**

* **pop**() : Supprime le dernier élément d’un tableau.
* **delete**.nom-Tableau[indice] : Supprimer un élément selon l’indice.
* **push**() : Ajoute un nouvel élément à la fin du tableau.
* **sort**() : Trie les éléments par ordre alphabétique (string).
* **reverse**() : Trier les éléments dans un ordre décroissant (indice).
* **shift**() : Supprime le premier élément d’un tableau.
* **toString**() : Convertit des éléments en chaînes de caractères.
* **unshift**() : Ajoute un nouvel élément au début d’un tableau.
* **concat**() : Fusionne plusieurs tableaux en un seul.
* **join**() : Combine les éléments d’un tableau en une seule chaîne de caractères et renvoie la chaîne.
* **slice**() : Copie une partie d’un tableau dans un nouveau tableau.
* **splice**() : Est utilisé pour modifier le contenu d’un tableau en supprimant les éléments existants et/ou en ajoutant de nouveaux éléments.
* **valueOf**() : Retourne la valeur primitive de l’objet spécifié.
* **indexOf**() : Retourne le premier index auquel l’élément donné peut être trouvé dans un tableau, ou s’il n’est pas présent dans un tableau.
* **lastIndexOf**() : Retourne le dernier index auquel l’élément donné peut être trouvé dans un tableau, ou -1 s’il n’est pas présent dans un tableau.

###### **Fonctions pour la manipulation des dates/heures en Javascript**

* **getDate**() : Récupère le jour du mois sous forme de nombre (1-31).
* **getDay**() : Le jour de la semaine sous forme de nombre (0-6).
* **getFullYear**() : Année sous la forme d’un nombre à quatre chiffres (aaaa).
* **getHours**() : Retourne l’heure (0-23).
* **getMilliseconds**() : Retourne les millisecondes (0-999).
* **getMinutes**() : Retourne les minutes (0-59).
* **getMonth**() : Retourne le mois sous forme de nombre (0-11).
* **getSeconds**() : Récupère les secondes (0-59).
* **getTime**() : Récupère les millisecondes depuis le 1er janvier 1970.
* **getUTCDate**() : Le jour du mois à la date spécifiée selon l’heure universelle.
* **setDate**() : Définit le jour en tant que nombre (1-31).
* **setFullYear**() : Définit l’année.
* **setHours**() : Définit l’heure (0-23).
* **setMilliseconds**() : Définit les millisecondes (0 à 999).
* **setMinutes**() : Définit les minutes (0-59).
* **setMonth**() : Définit le mois (0-11).
* **setSeconds**() : Définit les secondes (0-59).
* **setTime**() : Fixe l’heure (en millisecondes depuis le 1er janvier 1970).
* **setUTCDate**() : Définit le jour du mois pour une date spécifiée en fonction du temps universel.

**6- Les boîtes de dialogue** : JavaScript nous offre 3 types de boîtes de dialogue :

**6.1- La boite de message** : Alert () ouvre une petite fenêtre avec un texte et un bouton OK.

La syntaxe est : **alert** ('Texte à afficher')

Cette boîte de dialogue permet de faire apparaitre les valeurs des variables au cours d’exécution du code.

**Exemple** : Afficher le produit de 2 nombres avec la fonction **alert.**

**var** a=5, b=7;

c=a\*b;

**alert ('**Le produit de **' +a+ '** et **' +b+ '** est **: ' +c)**

**6.2- La boite de saisie** : prompt () permet de saisir un texte et contient 2 boutons, OK et Annuler.

La syntaxe est comme suite : **prompt** ("message" **,** "texte par défaut")

Le texte par défaut est optionnel (pas obligatoire).

La valeur entrée par l’utilisateur pourra être récupérée dans une variable qui sera traitée ailleurs.

**Exemple 1** : Reprendre l’exemple précédent avec la fonction **prompt**.

**var** a = **prompt** ("Entrez la valeur de a");

**var** b = **prompt** ("Entrez la valeur de b");

c=a\*b;

**alert** **('**Le produit de **' +a+ '** et **' +b+ '** est **: ' +c)**

**Exemple 2** : Vérifier si une personne possède le permis de conduire avec la fonction **prompt.**

**var** permis = prompt ('Possédez-vous le Permis de conduire ?','oui ou non');

**if** (permis=='oui')

**document.write**('Conduite autorisée')

**else**

**document.write**('Conduite non autorisée')

**6.3- La boite de message** : confirm () affiche le message entre les parenthèses, elle comporte 2 Boutons, OK et Annuler.

Cette méthode retourne vrai (true) si l'utilisateur clique sur OK et faux (false) s'il clique sur Annuler.    
    
La syntaxe est : **confirm** ("message")

Cette instruction appelle la méthode confirm en affichant la question. Si l'utilisateur clique sur OK. Alors on retourne la valeur true sinon la valeur false.

**Exemple** :

<**center**> <**input** **type**=button onClick="**if** (**confirm**('voulez-vous vraiment quitter?'))

document.write(**'Vous avez quitté'**) ; **else** document.write(**'Vous n’avez pas quitté'**);"

**value**="Quitter"></**center**>

**7- Débogage** : Le débogage est le processus de recherche et de correction des erreurs (syntaxique, lexicale ou sémantique) dans un script.

Tous les navigateurs modernes prennent en charge les outils de débogage.

Il est possible d’afficher la sortie sur la console du navigateur avec la fonction **console**.**log**().

**Exemple** : Afficher une variable x initialisée avec un nombre entier et une variable y non initialisé avec la fonction **console**.**log**().

**var** x=7, y;

**console.log**(x)

**console.log**(y) // La console affiche (undefined) pour y.

**8- Les fonctions anonymes** : Une fonction anonyme est déclarée sans non de fonction, pour faire appel à cette fonction on utilise une variable.

**Exemple** : Ecrire une fonction anonyme qui calcule le produit de deux nombres.

**var** p = (a,b)=> a\*b ;

**document.write** (**p**(3,5)) ; // Appel à la fonction avec la variable **p** et les arguments 3 et 5

**Remarque** : Il existe deux manières pour séparer les instructions en JS :

**1-** Utiliser un point-virgule (;) à la fin de l'instruction.

**2-** Revenir à la ligne avec la touche Entrée.

**INSFP OULED FAYET**

**Spécialité :** Développeur web et mobile / S2

**Module :** Les interfaces utilisateurs interactives

**TP n° 3**

Ecrire en JavaScript, les scripts permettant de :

**1-** Créer une fonction qui prend un nombre entier de minutes et le convertit en secondes.

**2-** Calculez le nombre de combinaison de P parmi N avec une fonction qui calcule le factoriel (**fact**).

C**Pn**= **fact**(n) **/** (**fact**(n-p)\***fact**(p)).

**3-** Calculez le nombre de combinaison de P parmi N en utilisant une fonction récursive (**facr**).

C**Pn**= **facr**(n) **/** (**facr**(n-p)\***facr**(p)).

**4-** Créer 2 fonctions qui calculent le périmètre et la surface d’un cercle.

**5-** Créer un script qui affiche une boîte de dialogue pour saisir le message "Entrez l’année à vérifier".

Le script testera ensuite si l’année saisie est bissextile et retourne année bissextile ou non bissextile

avec une boite de dialogue.

**Solution** :

**1-** Conversion des minutes en secondes.

**var** m = 7;

**function** min\_sec(x)

{

s=x\*60;

return s;

}

**document.write** (min\_sec(m));

**2-** Calcul du nombre de combinaison de P parmi N avec fonction (**fact**).

**var** n=5, p=3;

**function** fact(x)

{

**var** f=1;

**for**(i=1;i<=x;i++)

{

f=f\*i;

}

**return** f ;

}

c = fact (n) / (fact (n-p) \* fact (p));

**document.write**(c);

**3-** Calcul du nombre de combinaison de P parmi N en utilisant la récursivité.

**var** n=5, p=3;

**function** facr(x)

{

**if** (x==1)

**return** 1;

**else**

**return** x\*facr(x-1);

}

**c** = facr (n) / (facr (n-p) \* facr (p));

**document.write**(c);

**4-** Fonctions qui calculent le périmètre et la surface d’un cercle.

**var** r=5;

**function** périmètre ()

{

p=r\*2\*Math.PI;

**return** p ;

}

**function** surface ()

{

s=r\*r\*Math.PI;

**return** s ;

}

**document.write** (périmètre ()+ '<br>');

**document.write** (surface());

**5-** Renvoyer si l’année est bissextile ou non en

utilisant les boites de dialogues.

**var** a= **prompt** ("Entrez l’année à vérifier");

if (a%4==0 && a%100!=0)

{

**alert (a+**"est une année bissextile");

}

else

{

**alert (a+**"est une année non bissextile");

**INSFP OULED FAYET**

**Spécialité :** Développeur web et mobile / S2

**Module :** Les interfaces utilisateurs interactives

**Chapitre 4 : Les tableaux et les matrices**

**I- Les tableaux :**

**Introduction** : On suppose que nous avons 100 étudiants dont on veut représenter leurs noms, alors on doit déclarer 100 variables et la solution serait très longe. Pour remédier à ce problème, on peut déclarer une structure de données qui s’appelle **tableau**.

**1-Définition** : Un **tableau** est une structure de données contenant des cases, ayant une taille, il permet de stocker en mémoire plusieurs valeurs de même ou de différents types.

Ces valeurs sont appelées les éléments du tableau.

On peut parcourir un tableau en utilisant des indices.

**Exemple** : Nous avons le tableau T suivant de type entier et de taille n=5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 5 | 10 | 9 | 7 |

Le premier élément du tableau est T [0]=3. T :

Le dernier élément du tableau est T [4]=7. 0 1 2 3 4

Le nombre i allant de 0 à 4 est appelé l’**indice** du tableau (0 est le premier indice par défaut).

**2-Ecriture d’un tableau** : Un tableau en JS est déclaré come suite :

**var** nom-tableau = **new** **Array**();

**Exemple** : Créer un tableau qui contient les noms de trois stagiaires.

**Première méthode :**

**var** Tab = **new Array** ("Ali", "Réda", "Tahar");

**Deuxième méthode :**

**var** Tab **= new Array**(3);

**for** (i=1;i<=3;i++)

**{**

**Tab[i]=prompt** ("Donnez le nom du stagiaire "+i);

**}**

**3- Lecture d’un** **tableau** : Pour pouvoir afficher un tableau, il suffit de lire ses éléments un par un.

**Première méthode :**

**var** Tab = **new Array** ("Ali", "Réda", "Tahar");

**for** (i=0;i<3;i++)

**{**

**document.write** (Tab[i]+ "<br>");

**}**

**Deuxième méthode :**

**var** Tab **= new Array**(3);

**for** (i=1;i<=3;i++)

**{**

**Tab[i]=prompt** ("Donnez le nom du stagiaire "+i);

**}**

**for** (i=1;i<=3;i++)

**{**

**document.write** (Tab[i]+ "<br>");

**}**

**Troisième méthode :**

**var** Tab = **new Array** ("Ali", "Réda", "Tahar");

**for** (**var** indice **in** Tab)

**{**

**document.write** (Tab[indice]+ "<br>");

**}**

**Remarques** :

**a/** L’indice utilisé avec for (**i** **in** Tab) détecte automatiquement le premier indice (0, 1…etc).

**b/** Il est possible d’afficher un tableau dans la console avec **console.table** (nom\_tabeau);

**4- Les** **tableaux et les types** : Un tableau peut contenir des valeurs de même type ou de différents types.

**Exemple** : Créer et afficher un tableau avec des nombres entiers, réels, chaîne de caractères et booléens.

**Première méthode :**

**var** Tab = **new** Array (5, "Bonjour", 290, 1.57, false, true) ;

Tab ["RDV"] = "Réserver un rendez-vous";

Tab ["pi"] = Math.PI;

**for** (i=0;i<6;i++)

{

**document.write** (Tab [i]+ "<br>");

}

**document.write** (Tab ["RDV"]+ "<br>");

**document.write** (Tab ["pi"]);

**Remarque** : L’indice d’un tableau peut être un entier ou un autre type.

**Deuxième méthode :**

**var** Tab = **new** Array (5, "Bonjour", 290, 1.57, false, true) ;

Tab ["RDV"] = "Réserver un rendez-vous";

Tab ["pi"] = Math.PI;

**for** (i **in** Tab)

{

**document.write** (Tab [i]+ "<br>");

}

**Remarque** : L’indice i dans for (i in Tab) permet de parcourir tous les éléments du tableau, soit l’indice est un nombre, chaîne de caractères, etc).

**5- Manipulations dans les tableaux** : Il existe plusieurs traitements applicables sur les tableaux comme la recherche, l’ajout et la suppression d’un élément, le tri du tableau, fusionner plusieurs tableaux…etc.

Voici quelques exemples de fonctions pour manipuler les tableaux :

**a/** La taille du tableau est renvoyé avec (**nom\_tableau.length**).

**Remarque** : La taille avec **.length** prend en considération uniquement les indices qui sont des nombres.

**b/** Pour trier un tableau, on fait appel à la fonction **sort()** comme suite : **nom\_tableau.sort()**.

Le tri se fait par ordre **alphabétique** avec des chaînes de caractères.

Pour trier les **nombre** en ordre **croissant**, on procède comme suite :

**const** x = (a,b)=> **a - b**;

**document.write** (**nom\_tableau.sort** (x));

Pour trier les **nombre** en ordre **décroissant**, on procède comme suite :

**const** x = (a,b)=> **b - a**;

**document.write** (**nom\_tableau.sort** (x));

**Remarque** : Le tri est irréversible, une fois trié, il est impossible de récupérer le tableau initial.

**c/** La fonction **join**("**;**") combine les éléments d’un tableau en une seule chaîne de caractères séparée avec des **;** (par défaut, la séparation se fait avec des virgules).

**d/** La fonction **reverse**() permet de trier les éléments dans un ordre décroissant (selon l’indice).

**II- Les matrices :**

**1-Définition** : Une matrice est un tableau à deux dimensions dont les éléments sont de même ou de différents types, ces éléments sont représentés selon 2 indices, pour les lignes et pour les colonnes.

**2-Ecriture et lecture d’une matrice** : La syntaxe ci-dessous permet de créer et d’afficher une matrice.

var nom\_matrice= new Array(); // Création des lignes de la matrice.

for(var i=0; i<n; i++) // n est le nombre de lignes.

{

nom\_matrice [i] = new Array(); // dans chaque ligne, on crée un tableau (colonnes).

}

for(var i=0; i<n; i++)

{

for(var j=0; j<m; j++) // m est le nombre de colonnes.

{

nom\_matrice [i][j] = v; // Saisie des valeurs de la matrice.

}

}

for(var i=0; i<n; i++)

{

for(var j=0; j<m; j++)

{

document.write(Mat[i][j]); // Affichage des valeurs de la matrice.

}

}

**Exemple** : Créer et afficher une matrice Mat avec 3 lignes et 2 colonnes comme c’est montré ci-dessous.

0 1

|  |  |
| --- | --- |
| 1 | 2 |
| 3 | 4 |
| 5 | 6 |

0

1

2

**var** Mat = **new Array** (3);

**for** (i=0;i<3;i++)

{

**Mat [i]** = new Array(2);

}

**for** (i=0;i<3;i++)

{

**for** (j=0;j<2;j++)

{

**Mat[i][j]** = prompt ("Donnez la valeur de la case " +i+ " , " +j);

}

}

**for** (i=0;i<3;i++)

{

**for** (j=0;j<2;j++)

{

**document.writeln** (Mat[i][j]);

}

**document.write** ("<br>");

}

**3-Les opérations sur les matrices**: On peut effectuer sur les éléments d’une matrice la lecture, l’écriture, la recherche d’une valeur, l’addition de deux matrices, le produit de deux matrices…etc.

**INSFP OULED FAYET**

**Spécialité :** Développeur web et mobile / S2

**Module :** Les interfaces utilisateurs interactives

**TP n° 4**

Ecrire en JavaScript, les scripts permettant de :

**1-** Faire la recherche d’un entier x dans un tableau et afficher sa position.

**2-** Supprimer un élément x, s’il existe dans un tableau d’entiers.

**3-** Renvoyer la valeur minimale des éléments d’un tableau d’entier.

**4-** Insérer un élément dans un tableau d’entier à une position donnée.

**5-** Calculez la somme de deux matrices d’entiers de dimensions (x\*y).

**6-** Créer une matrice qui contient les noms et les notes des stagiaires (comme c’est montré ci-dessous).

Calculez pour chaque stagiaire la moyenne des deux modules et faites l’affichage dans la console.

Math Physique Moyenne

|  |  |  |
| --- | --- | --- |
| 12 | 13 | 12,5 |
| 10 | 13 | 11,5 |
| 15 | 11 | 13 |

Ali

Réda

Tahar

**Solution** :

**1-** Recherche d’un entier x dans un tableau

et affichage de sa position.

**var** Tab = **new Array** ();

**for** (i=1;i<=5;i++)

**{**

**Tab[i]=prompt** ("Donnez la valeur de la case "+i);

**}**

**var** x=**prompt**("Entrez la valeur à rechercher");

**var** y=**true**;

**for** (i=0;i<5;i++)

{

**if** (Tab[i]==x)

{

**document.write** (x+" existe à la position :

"+i +"<br>");

y=**false**;

}

}

**if** (y==true)

**document.write** (x+" n'existe pas dans le

tableau");

**2-** Suppression d’un élément x, s’il existe dans un tableau d’entiers

**var** Tab = new Array (3, 8, 14, 10, 20);

**var** y=-1;

**var** x=prompt ("Entrez la valeur x à supprimer");

**for** (i=0;i<Tab.length;i++)

{

**document.writeln** (Tab[i]);

}

**document.writeln** ("<br>");

**for** (i=0; i<Tab.length; i++)

{

**if** (Tab[i]==x)

y=i;

}

**if** (y==-1)

{

**document.write** (x+" n'existe pas dans

le tableau");

}

**else**

{

**for** (i=y;i<Tab.length-1;i++)

{

**Tab**[i]=**Tab**[i+1];

}

**delete** **Tab**[Tab.length-1]; // Supprimer // le dernier élément après avoir fait le décalage.

**for** (i=0;i<Tab.length-1;i++)

{

**document.writeln** (Tab[i]);

}

}

**3-** Renvoi de la valeur minimale des éléments d’un tableau d’entier.

**var** Tab = **new Array** (3, 8, 1, 10, 0);

**var** min=**Tab**[0];

**for** (i=1;i< Tab.length;i++)

{

**if** (min>Tab[i])

min=Tab[i]

}

**document.write** ("La valeur minimale est :

"+min);

**4-** Insertion d’un élément dans un tableau d’entier à une position donnée.

**var** Tab = **new Array** ();

**var** a=**prompt** ("Entrez la taille du tableau");

**for** (i=1;i<=a;i++)

{

**Tab**[i]=**prompt** ("Entrez l'élément de la

case "+i);

}

**var** x=**prompt** ("Entrez la valeur x à insérer");

**var** y=**prompt** ("Entrez la valeur de la position

où doit se faire l’insertion");

**for** (i=1;i<=a;i++)

{

**document.write** (Tab[i]);

}

**document.write** ("<br>");

**for** (i=a; i>=y; i--)

{

**Tab**[i+1]=Tab[i]

}

**Tab**[y]=x;

**for** (i in Tab)

{

**document.write** (Tab[i]);

}

**5-** Calcul de la somme de deux matrices d’entiers de dimensions (x\*y).

**var** Mat1 = **new Array**();

**var** Mat2 = **new Array**();

**var** Mat3 = **new Array**();

**var** x=**prompt**("Entrez le nombre de lignes de

la matrice");

**var** y=**prompt**("Entrez le nombre de colonnes de la matrice");

**for** (i=1;i<=x;i++)

{

**Mat1** [i]=**new Array**();

}

**for** (i=1;i<=x;i++)

{

**for** (j=1;j<=y;j++)

{

**Mat1**[i][j]=**prompt** ("Pour la première matrice,

donnez la valeur de la case " +i+ " , " +j);

}

}

**for** (i=1;i<=x;i++)

{

**for** (j=1;j<=y;j++)

{

**document.writeln**(Mat1[i][j]);

}

**document.write** ("<br>");

}

**document.write** ("<br>");

**for** (i=1;i<=x;i++)

{

**Mat2** [i]=**new Array**();

}

**for** (i=1;i<=x;i++)

{

**for** (j=1;j<=y;j++)

{

**Mat2**[i][j]=**prompt** ("Pour la deuxième matrice, donnez la valeur de la case " +i+ " , " +j);

}

}

**for** (i=1;i<=x;i++)

{

**for** (j=1;j<=y;j++)

{

**document.writeln** (Mat2[i][j]);

}

**document.write** ("<br>");

}

**document.write** ("<br>");

**for** (i=1;i<=x;i++)

{

**Mat3** [i]=**new Array**();

}

**for** (i=1;i<=x;i++)

{

**for** (j=1;j<=y;j++)

{

**Mat3**[i][j]=**Math.abs**(**Mat1**[i][j])+

**Math.ab**s(**Mat2**[i][j]);

**document.writeln**(Mat3[i][j]);

}

**document.write** ("<br>");

}

**6-** Calcul des moyennes des stagiaires selon les modules.

**var** Mat= **new Array**();

**var** n=**prompt**("Entrez le nombre de lignes");

**var** m=**prompt**("Entrez le nombre de

colonnes");

**for** (i=0; i<n;i++)

{

**Mat**[i]=**new Array**();

}

**for** (i=1; i<n;i++)

{

**Mat** [i][0]=**prompt**("Entrez le nom du

stagiaire "+i+" : ");

}

**for** (j=1; j<m-1;j++)

{

**Mat** [0][j]=**prompt**("Entrez le nom du

module "+j+" : ");

}

**Mat** [0][m-1]="Moyenne";

**for** (i=1; i<n;i++)

{

**for** (j=1; j<m-1;j++)

{

**Mat** [i][j]=**prompt**("Entrez la note de

"+**Mat**[i][0]+" dans le module "+**Mat**[0][j]);

}

}

for (i=1; i<n;i++)

{

s=0;

for (j=1; j<m-1;j++)

{

s=s+Math.abs(Mat[i][j])

}

Mat[i][j]=s/(m-2);

}

Mat[0][0]=" ";

**console.table**(**Mat**); //Affichage dans la

console.

**INSFP OULED FAYET**

**Spécialité :** Développeur web et mobile / S2

**Module :** Les interfaces utilisateurs interactives

**Chapitre 5 : Les objets et les DOM**

**I- Notion d’objet :** JavaScript est un langage qui repose sur les objets.

Un objet se caractérise par ses propriétés et ses méthodes (fonctions).

Une propriété est une paire (**clé** : **valeur**) « **key** : **value** », dont la clé est une chaîne de caractères.

**Exemple :** Age : 25, (**Age** est la **clé**, **25** est la **valeur**).

Il existe des objets prédéfinis dans JavaScript, comme il est possible de définir ses propres objets.

**1- Création d’objet :** Il existe 4 façons de créer des objets en JavaScript comme suite :

**a/** Déclaration **littérale**.

**b/** Utilisation de constructeur d'objet avec le mot clé **new object.**

**c/** Utilisation d’une **fonction** **constructeur.**

**d/ Instanciation** de classe.

### 1.1- Déclaration littérale : La déclaration littérale consiste à utiliser des accolades :

**Exemple :** Créer un objet « utilisateur » avec 3 propriétés (prénom, nom, age) et une méthode pour afficher ces propriétés.

**var** utilisateur = {

**prénom** : "Ali",

**nom** : "Mimoune",

**age** : 22,

**afficher**: function() {

console.log (**utilisateur**.prénom, **utilisateur**.nom, **utilisateur**.age);

}

}

**utilisateur**.afficher ();

**Remarques :**

**1-**Le mot-clé **this** permet d’accéder aux propriétés et aux méthodes de l’objet à l’intérieur d’une méthode.

**2-**Le mot clé function n’est pas obligatoire. **afficher**: function() ⬄ **afficher**()

Pour l’exemple précédent, on peut écrire la méthode « afficher » comme suite :

**afficher**() {

console.log (**this**.prénom, **this**.nom, **this**.age);

}

**1.2- Utilisation de constructeur d’objet :** En utilisant le mot clé **new object.**

**Exemple** : Reprendre l’exemple précédent en utilisant un constructeur d’objet.

**var** utilisateur = **new** Object(); // d’abord créer un objet vide

**utilisateur**.prenom ="Ali",

**utilisateur**['nom'] ="Mimoune",

**utilisateur**.age = 22,

**utilisateur**.afficher = function() // Déclaration de la méthode

{

**console**.log(**utilisateur**.prenom, **utilisateur**.nom, **utilisateur**.age)

}

**utilisateur**.afficher(); // Appel à la méthode

**Remarques :**

**1-** La syntaxe (**var** obj = **new** Object (); est l’équivalence de ( **var** obj = {}; ).

**2-** La syntaxe (**utilisateur.nom**) est l’équivalence de (**utilisateur['nom']**).

**1.3 -** Utilisation d’une **fonction** **constructeur :** C’est une fonction permettant de créer un objet.

Par convention ces fonctions commencent par une majuscule.

**Exemple** : Reprendre l’exemple précédent en utilisant une fonction constructeur.

**function** Utilisateur() {

**this**.prénom = "Ali";

**this**.nom = "Mimoune";

**this**.age = 22;

**this**.afficher = function() {

**console**.log (**this**.nom, **this**.prénom, **this**.age);

}

}

**var** u = **new** Utilisateur; // Pour créer un objet, il faut utiliser le mot-clé **new**.

**u**.afficher ();

Un des intérêts d’un constructeur est d’utiliser des paramètres pour initialiser l’état de l’objet comme suite :

### function Utilisateur (prénom, nom, age)

### {

### this.p = prénom;

### this.n = nom;

### this.a = age;

### this.afficher = function() {

### console.log(this.n, this.p, this.a);

### }

### }

### var u = new Utilisateur ("Ali","Mimoune", 22 );

### u.afficher ();

**Remarque :** En utilisant le principe de la fermeture (***closure***) sur les fonctions, il est possible de créer des propriétés privées à un objet :

### function Utilisateur (Prénom, Nom, Age)

### {

### P = Prénom;

### N = Nom;

### A = Age;

### this.afficher = function() {

### console.log(P, N, A);

### }

### }

### var u = new Utilisateur ("Ali","Mimoune", 22 );

### u.afficher ();

**document.write**(u.A); // Pas d’affichage, parce que A est une variable locale (privées à l’objet u)

Dans l’exemple précédent, les variables P, N et A sont locales à la fonction **Utilisateur** et ne seront pas visibles à travers la variable u (non affichées avec **document.write**(u.A)).

Par contre, en utilisant les 2 premières méthodes (déclaration littérale et utilisation de constructeur avec new object), il est possible d’afficher l’âge avec document.write(u.A), car, avec les 2 premières méthodes, la propriété n’est pas privée.

## 1.4- Instanciation de classe : Créer un objet à partir d’une classe.

**Exemple** : Créer un objet à partir d’une classe permettant de renvoyer le prix et la référence d’un article.

**class** Article {

**constructor** (Produit, Reference, Prix) {

**this**.produit = Produit;

**this**.reference = Reference;

**this**.prix = Prix;

}

**calculerPrix**() {

return this.prix \* 1.2; // Pour chaque produit, on a un bénéfice de 20%

}

}

**var** article1 = **new** Article ('Sac à dos', 520, 2500);

**var** article2 = **new** Article ('Chaussure',625,3200);

**console.log** ('Le prix de '+**article1.produit**+' avec la référence '+**article1.reference**+' est : '+**article1.calculerPrix**());

**console.log** ('Le prix de '+**article2.produit**+' avec la référence '+**article2.reference**+' est : '+**article2.calculerPrix**());

**2- Héritage avec les classes :** On peut créer une classe enfant à partir d’une classe parent en utilisant les mots clés **extends** et **super**, comme on peut apporter des modifications dans la classe enfant.

**Exemple** : Créer une classe Promo à partir de la classe Article créée précédemment (Promo hérite les propriétés et les méthodes de Article) et ajouter une méthode qui calcule le nouveau prix de promotion.

**class** Article {

**constructor** (produit, reference, prix)

{

**this**.produit = produit;

**this**.reference = reference;

**this**.prix = prix;

}

**calculerPrix**() {

**return** this.prix \* 1.2;

}

}

**class** Promo **extends** Article // Promo est la classe enfant de la classe parent Article

{

**constructor** (produit, reference, prix, remise) // Appel de constructeur parent

{

**super** (produit, reference, prix); // super détermine les propriétés de la clase parent

**this**.r = remise ; // Créer une nouvelle propriété pour le pourcentage de remise

}

**calculerPrixPromo**()

{

**return** **super**.calculerPrix()\* **this**.r ; // Calculer le nouveau prix promotionnel

}

}

**var** promo1 = **new** Promo('Sac à dos', 520, 2500, 0.7 ); // La remise est à 30% (prix \*0.7)

**console.log** ("Le prix avec promotion de " + **promo1.produit** + " est : "**+ promo1.calculerPrixPromo**() + " au lieu de " + **promo1.calculerPrix**());

**3- Prototype :** Un prototype permet de créer l’héritage avec les **objets** (sans passer par les classes).

**Exemple** : En utilisant un prototype, créer un objet personne qui contient le nom, le prénom et l’âge, puis un autre objet fils qui hérite les propriétés de personne. Modifier les valeurs (nom et âge) de fils.

**function** Personne() {

**this**.nom = "Benyahia";

**this**.prénom = "Tahar";

**this**.age=50 ;

}

**function** Fils() {

}

**Fils**.**prototype** = new Personne;

**e** = **new** Fils;

**Fils.prototype**.prénom = "Ahmed"; // changer la valeur de la propriété prénom

**Fils.prototype**.age=20; // changer la valeur de la propriété age

**console.log** (**e**.nom, **e**.prénom, e.age); // Affiche les valeurs des propriétés de fils

**4- Méthode static** : Une méthode avec le mot-clé **static** désigne une fonction qui est ajoutée à la classe et non à l’instance (fonction accessible avec le nom de la classe mais pas avec le nom de l’objet).

**Exemple** : A l’intérieur d’une classe, créer une méthode qui calcule le triple d’un nombre avec le mot-clé **static**. Créer un objet à partir de la classe et l’appel de la méthode doit se faire avec le nom de classe.

**class** Triple {

**static** calcultriple(a) {

**return** a\*3;

}

}

**T** = **new** Triple; // Création d’objet T

**console.log** (**Triple**.calcultriple(**5**)); // Affichage de résultat égal à 15

// console.log(T.calcultriple(5)); => Erreur (la méthode doit être appelée par la classe et pas par l’objet).

**II Notion de DOM (**Document Object Model)**:**

**DOM** pour (modèle objet de document) est une interface de programmation pour documents HTML. Normalisée par le [W3C](https://fr.wikipedia.org/wiki/W3C), il fournit une représentation structurée sous forme d'**arbre** composé de nœuds.

Chaque nœud représente un objet possédant des propriétés et des méthodes pour manipuler des **éléments** HTML ou CSS.

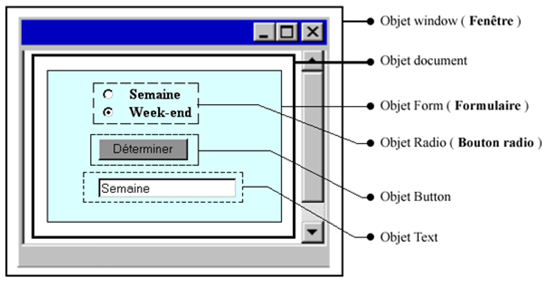
Un DOM fournit une représentation structurée et la manière dont un script peut accéder à un objet.



**Exemple de représentation d’un document HTML sous forme d’arbre**

**Exemple** : Dans le schéma ci-dessous, pour accéder à l’objet Buttom, on suit la hiérarchie suivante :

**window.document.Form.Button**



**Exemple d'imbrication des objets**

L'objet **window** est au sommet de la hiérarchie et chaque objet peut posséder des sous-objets.

**Exemples d’objets prédéfinis :**

**1- Objet Window** : Les trois propriétés suivantes sont des objets enfant de l'objet window.

- **document** : Contient la page web.

- **location** : Stocke l'adresse du site visité (URL)

- **history** : Conserve la liste des sites visités par le navigateur.

Autres propriétés de l'objet window :

**- closed** : Indique si une fenêtre est ouverte ou pas.

**- defaultStatus** : Indique le message par défaut affiché dans la barre d'état.

Quelques méthodes de l'objet window :

**- Open** : Permet d'ouvrir une nouvelle fenêtre.

**- close** : Referme une fenêtre ouverte par la méthode open.

**- print** : Permet d'imprimer la page courante.

**alert , confirm, prompt** etc.

**2- Objet document**: L'objet document est l'objet enfant de l'objet window.

Il représente le contenu de la page Web affichée.

Quelques propriétés de l'objet document :

- **form**[] : C'est un tableau qui referme tous les formulaires du document.

- **title** : Donne le titre du document.

Quelques méthodes de l'objet document :

- **write** : Permet d'écrire sur le document courant.

- **writeln** : Il écrit la chaîne donnée avec un espace à la fin.

|  |  |
| --- | --- |
| **- getElementsByName**() | : Il renvoie tous les éléments ayant la valeur de nom donnée. |

**3- Objet history** : C’est le fils de l'objet window.

**- length** : Propriété indiquant le nombre de sites visités.

Quelques méthodes de l'objet history :

**- history.back()** : Retourne à l'adresse précédente.

**- history.forward()** : Aller à l'adresse suivante sur la liste.

**- history.go()** : Conduit à l'adresse indiquée par son numéro positif (suivante) ou négatif (précédente).

**4- Objet Form** : Représente un formulaire HTML.

Quelques propriétés de l'objet form :

**- name** : Nom du formulaire.

**- length** : Donne le nombre de formulaires dans le document HTML.

**- action** : Définie l’action de traitement du formulaire.

Quelques méthodes de l'objet form :

**- submit** : Envoi du formulaire à la destination indiquée dans l'attribut action.

**- reset** : Restore toutes les valeurs des champs du formulaire à leur valeurs initiale.

Autres Objets : L'objet **link**, **Date**, **String**, **Math**, etc.

**INSFP OULED FAYET**

**Spécialité :** Développeur web et mobile / S2

**Module :** Les interfaces utilisateurs interactives

**TP n° 5**

Ecrire en JavaScript, les scripts permettant de :

**1-** Ecrire une fonction qui vérifie si un objet est vide ou non.

**2-** Manipuler un objet comme suite :

**a/** Créer un objet vide « **étudiant** ».

**b/** Ajoutez la propriété « **nom** » avec la valeur « **Tahar** ».

**c/** Ajoutez la propriété « **age** » avec la valeur « **22** ».

**d/** Ajoutez la propriété « **spécialité** » avec la valeur « **informatique** ».

**e/** Afficher les valeurs de ces trois propriétés dans la console.

**f/** Changer la valeur «**informatique**»  pour «**maths**».

**g/** Supprimez la propriété « **age** » de l’objet.

**h/** Afficher les valeurs des deux propriétés dans la console.

**3-** Calculer la somme des salaires de trois employés dans l’objet employé.

**4-** Créer l’objet livre avec titre, nom-auteur, année et ISBN. Afficher le résultat.

**5-** Déclarer une classe voiture avec marque, référence et année et créer une autre classe enfant identique à la classe parent, puis créer une instance d’objet et afficher ces propriétés.