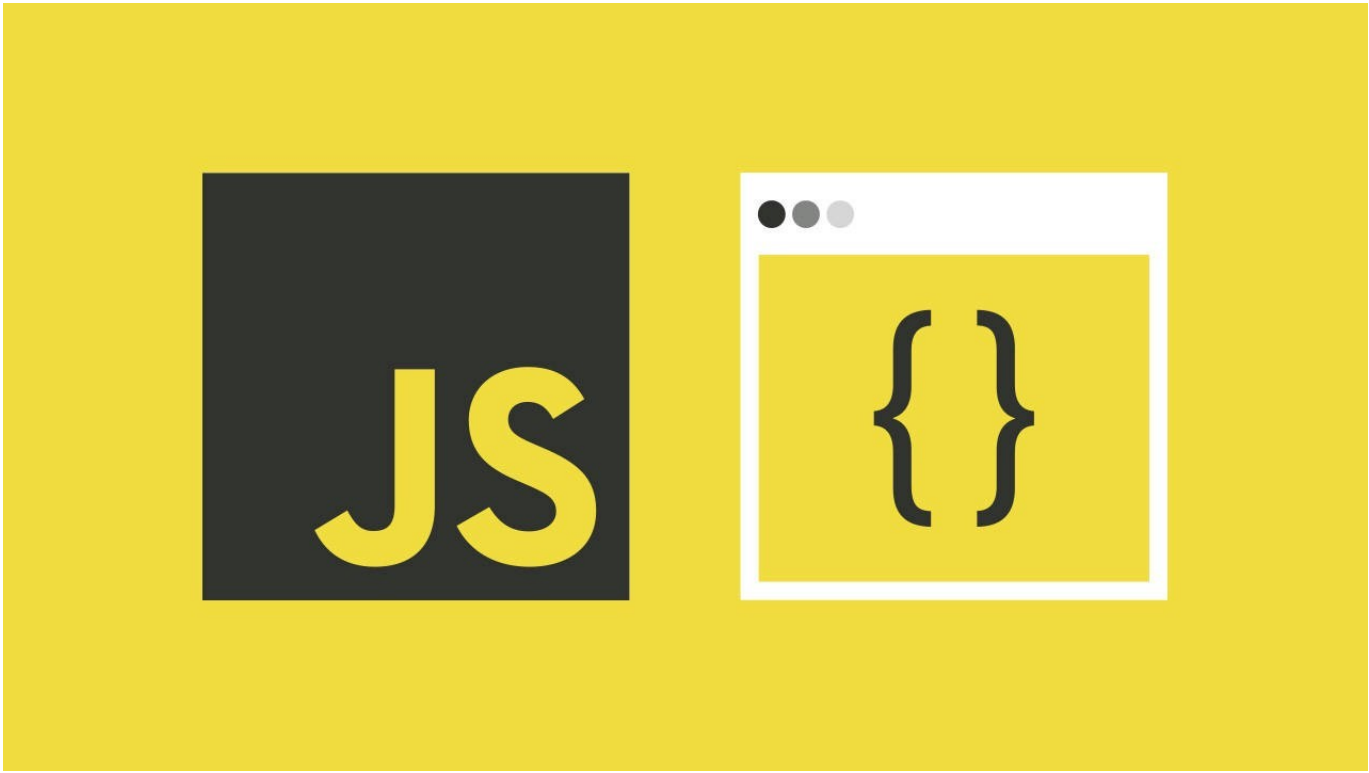
Langage web et programmation



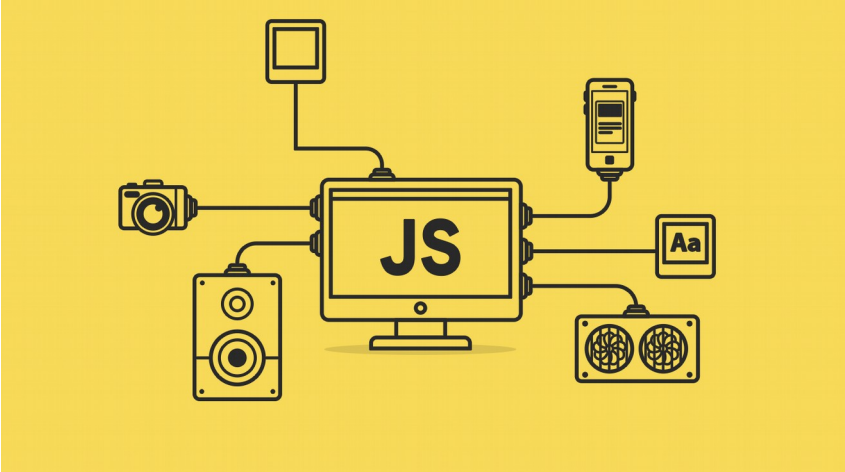
4TTr : Informatique

Table des matières

1 Introduction :

Le langage Javascript est un langage de script développé par la société Netscape. Contrairement à ce que pourrait laisser penser son nom, Javascript n'est pas apparenté avec le langage Java.

Du fait de sa capacité à s'exécuter dans virtuellement n'importe quel navigateur Web, Javascript est devenu au cours du temps le standard pour créer des interfaces utilisateur dynamiques pour le Web.

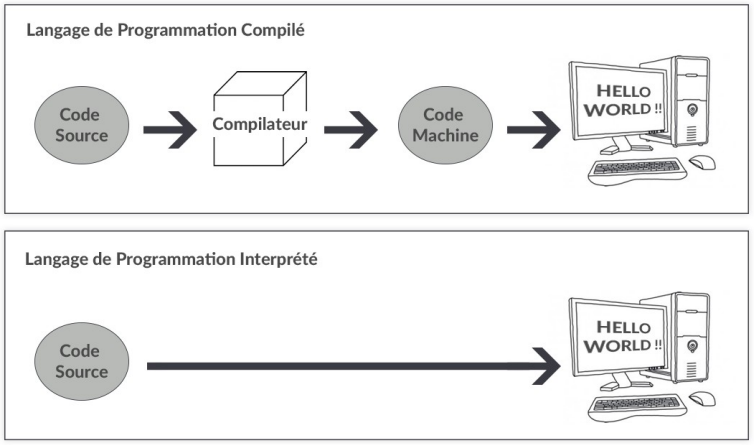


1.1 Javascript est un langage interprété.

On distingue habituellement deux modes d'exécution des programmes. Dans le premier mode, dit **compilé**, le code écrit par le programmeur (on dit aussi programme source) est traduit dans un autre format, directement compréhensible par la machine. La phase de traduction s'appelle la compilation et donne un fichier exécutable (ou binaire). Les avantages de ce mode sont :

- la rapidité d'exécution. Avec un fichier exécutable, les instructions sont directement exécutées par la machine.
- la validité syntaxique du code source. Pour que le compilateur puisse traduire le code source, il faut que les instructions soient syntaxiquement correctes.
- D'autre part, un avantage commercial est de pouvoir donner à une personne tierce, uniquement le fichier exécutable sans que cette personne puisse voir le code source. Elle sera donc liée au programmeur pour tout souhait de modification. L'inconvénient de cette méthode est que le fichier exécutable dépend à la fois de la machine et du système d'exploitation, et il est par conséquent plus difficile de le distribuer.

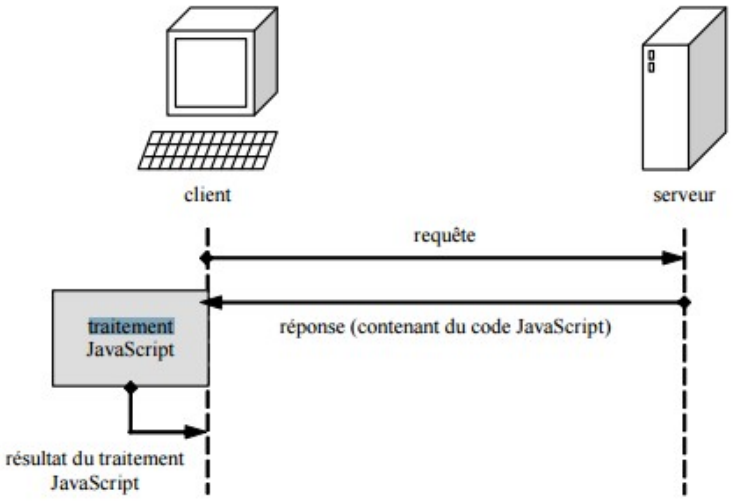
Le deuxième mode est dit **interprété** : un programme interprété lit le programme source, et essaye d'exécuter les instructions les unes après les autres. A la différence du mode compilé, il n'y a pas de fichier exécutable. L'avantage de ce mode est que toute personne ayant le programme interprété sur son ordinateur, peut exécuter le programme. L'inconvénient est la lenteur due à l'exécution et de l'interprète et du programme.



JavaScript est un langage ***interprété***. Il faut donc un programme interprète pour exécuter des programmes écrits dans ce langage. La grande majorité des navigateurs à l'heure actuelle sur le marché incorporent de tels interprètes.

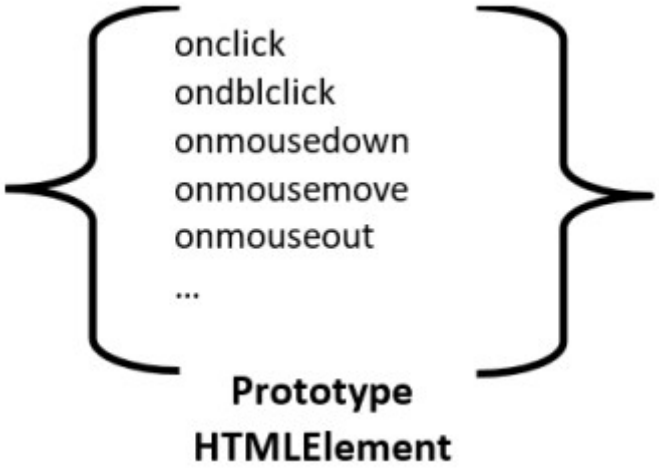
1.2 Javascript est un langage client.

Le javascript est un langage client, c'est à dire que l'exécution du code n'a pas besoin d'être sur un serveur (comme pour le php par exemple), elle s'exécute directement dans le navigateur. En d'autres termes, le serveur qui héberge le site internet n'interprète pas le code javascript, il est envoyé tel quel. C'est le navigateur lors de la réception qui l'interprétera, de la même manière qu'il interprétera l'HTML.



1.3 Javascript est un langage événementiel.

La programmation événementielle définit une programmation où les composants de l'application interagissent entre eux et avec l'environnement. Ils communiquent en réponse à des événements. Ces événements peuvent correspondre à une action de l'utilisateur : un clic sur un bouton de commande, une écriture dans une zone de texte, un choix dans une case d'option ou une case à cocher, le déplacement d'un objet, ... Ils peuvent aussi être déclenchés par le système : chargement d'une feuille, un top déclenché par l'horloge, ... On dira qu'une programmation événementielle fonctionne sur base de déclencheurs (triggers) et d'auditeurs (listeners).



2 Premiers pas avec le javascript

2.1 Les logiciels nécessaires :

Javascript est un langage facile à mettre en œuvre. De même que pour créer une page html, il vous suffit

- d'un éditeur de texte. *Bloc-notes* est suffisant mais d'autres logiciels offrent plus d'options, comme la coloration syntaxique qui est très appréciable. Des logiciels tels que *Notepad++* ou *SublimeText* seront à privilégier.
- d'un navigateur et de sa console de développement. Pour chrome, vous pourrez utiliser la console native [F12]. Pour Firefox, vous pourrez télécharger un plugin de développement tel que *Web Developer* ou encore *Firebug*.

  En outre de votre navigateur préféré, afin de vérifier leur compatibilité, il est nécessaire de tester vos scripts sur d'autres navigateurs. Internet explorer, Firefox, Opera, Chrome et Safari sont, à l'heure actuelle, les plus répandus.

2.2 Comment insérer le code javascript :

Il y a deux manières d'ajouter du code JavaScript dans une page :

1 : Lorsque les scripts sont relativement petits, il est possible de les intégrer directement dans le fichier html concerné en ajoutant le code JavaScript suivant:

```
<script type="text/javascript">
  // Mon code Javascript
  ...
</script>
```

Traditionnellement il est d'usage de placer la balise <script> entre les tags <head> et </head>, cependant dans certains cas, il est recommandé de la placer en fin de document juste avant </body> pour ne pas bloquer le chargement de la page, et exécuter les scripts uniquement lorsque le DOM est prêt.

Attention, la balise "script" n'est pas auto-fermante, c'est-à-dire que l'on ne peut pas écrire

```
<script type="text/javascript" ... />
```

2 : En liant depuis la page HTML un fichier externe, dans lequel sont placées les instructions JavaScript :

```
<script type="text/javascript" src="monscript.js"></script>
```

2.3 Les commentaires :

Pour rappel, les commentaires sont les phrases que l'on utilise pour commenter le code. Ces phrases n'ont aucune influence sur le code. L'ajout de commentaires explicatifs permet d'avoir un code clair et compréhensible pour toute personne qui le lirait.

Les commentaires en HTML s'écrivent de cette manière :

```
< !-- Je suis un commentaire -->
```

Vous pouvez donc y écrire plus ou moins ce que vous voulez. Evitez cependant les caractères spéciaux et plus particulièrement les crochets < et >, le dernier pouvant être interprété par le navigateur comme la fin du commentaire.

En javascript, c'est tout à fait différent, il existe deux manières d'écrire un commentaire.

- Les commentaires sur une seule ligne, ils se placent sur la fin de la ligne après deux slashs //.
- Les commentaires sur plusieurs lignes. Exactement comme en css, ils se placent entre /* et */ et ne possèdent aucune limitation de longueur. Vous pouvez donc écrire autant que vous voulez.

```
DuCode ;          // commentaire court
codeBis ;         /* autre commentaire court */
suiteDuCode ;
encoreDuCode ;    /* Ici commence une commentaire plus long
Je peux écrire autant que je veux
sur autant de ligne que j'en aurais envie
*/
SuiteDuCode ;
```

3 Les bases du javascript

3.1 Les instructions

Qu'est-ce que c'est ? Une instruction est un « ordre » que l'on donne à l'ordinateur. alert('Hello Wordl') ; est une instruction qui ordonne au script de créer une boîte de dialogue contenant le texte « Hello World ». Autre exemple, un calcul est une instruction.

Quand on donne une instruction à l'ordinateur, il faut également indiquer à l'ordinateur où se situe la fin de cette instruction. Pour cela en javascript, il y a deux solutions.

- Revenir à la ligne (avec la touche Enter) : l'ordinateur comprendra qu'il aura ensuite affaire à une autre instruction.
- Utiliser un point-virgule (;) à la fin de l'instruction.

Dans de nombreux langages de programmation (le javascript étant une exception) le point-virgule est obligatoire à la fin de chaque instruction. Je recommande donc fortement de prendre l'habitude dès maintenant d'utiliser un point-virgule à la fin de chaque instruction.

La séquence d'instruction.

En informatique, une séquence d'instruction est une série d'instruction exécutées les unes à la suite des autres. Quand on veut faire exécuter plusieurs instructions, on écrit donc ces instructions de haut en bas, en insérant un point-virgule entre chaque instruction.

3.2 Les variables

Une variable est une case mémoire à laquelle on a donné un nom. Le contenu de cette case mémoire (sa valeur) peut varier au cours de l'exécution du programme. On peut donc ranger une valeur dans cette case/variable par une instruction d'*affectation*. On peut aussi lire son contenu à tout moment.

La déclaration d'une variable.

```
var compteur ;
```

Ici, var est un mot-clé et compteur le nom choisi par le programmeur pour la variable. Aprèss une

telle déclaration et sans précision supplémentaire, la variable vaut *undefined* ou encore la valeur nommée *NUL*.

L'affectation d'une variable.

Si je veux signifier à un moment du programme que le nombre de points est quatre, j'écris l'instruction d'affectation suivante :

```
compteur = 4;
```

Les noms des variables

Les noms de variables sont de longueur quelconques, peuvent contenir des lettres, chiffres et le caractère underscore (_) , mais doivent commencer par une lettre.

Notons que JavaScript, contrairement à HTML, distingue les majuscules des minuscules, et qu'ainsi MaVariable et mavariable désignent deux variables différentes.

Il est d'usage en javascript d'écrire une variable en minuscules. Si le mot est composé, on écrira alors la variable en ajoutant une majuscule à la première lettre du second mot et des suivants.

```
var compteur ;
var nombreDeParticipant ;
```

Les identifiants en majuscules sont réservés pour les constantes.

Les constantes se déclarent avec le mot réservé const (à la place de var) et sont assignées lors de la déclaration, puis il est par définition impossible de les modifier ultérieurement.

```
var compteur ;
const MAX ;
```

L'identifiant d'une variable ne peut pas utiliser un mot clé déjà utilisé par le langage javascript. Les mots clés pour le javascript sont les suivants.

abstract	float	public
boolean	for	return
break	function	short
byte	goto	static
case	if	super
catch	implements	switch
chat	import	synchronized
class	in	this
const	instanceof	throw
continue	int	throws
default	interface	transient
do	long	true
double	native	try
else	new	var
extends	null	void
false	package	while
final	private	width
finally	protected	

Les types de données

On pourrait se poser la question suivante : peut on mettre n'importe quelle sorte de données dans une variable ? La réponse est non, car il se poserait alors des problèmes de cohérence : que se passerait il si on mettait le message « bonjour » dans compteur et que l'on essayait ensuite de lui ajouter un ? Pour assurer la cohérence des instruction, les langages informatiques définissent habituellement des types de données.

Javascript n'autorise la manipulation de 3 types de données :

- des nombres (entiers et des nombres à virgule flottante)
- des chaînes de caractères (string) : une suite de caractères
- des booléens : des variables à deux états permettant de vérifier une condition :
  - false: lors d'un résultat faux
  - true: si le résultat est vrai

En Javascript, contrairement à des langages évolués tels que le langage C ou Java, il n'y a pas besoin de déclarer le type de variables que l'on utilise, par contre leur nature influencera le comportement des exécutions.

Ainsi si x et y sont des chaînes de caractères :

```
x = '2' ;
y = '4' ;
z = x + y ; // z vaudra alors la chaîne de caractère 24
```

Ainsi si x et y sont des entiers :

```
x = 2 ;
y = 4 ;
z = x + y ; // z vaudra alors 6
```

Tel que vous le verrez dans les exercices, il existe des fonctions qui permettent alors de passer d'un type à un autre. Ainsi la fonction

```
nombreCommeEntier = parseIn(nombreDansUneChaineDeCaractere) ;
```

permettra de transformer une chaîne de caractères en un entier

Exemples de déclaration

Voici, quelques exemples de déclarations :

```
var texte;
var cle;
var compteur = 3;
var Erreur = "Connexion perdue.";
var Erreur2 = 'Pas assez de mémoire';
```

Notons que l'on peut initialiser les variables (y inscrire une valeur initiale) à leur déclaration, comme c'est le cas dans les trois dernières déclarations. On remarque aussi dans l'exemple précédent que l'on peut indifféremment utiliser les guillemets ou les quotes pour délimiter une chaîne de caractères.

3.3 Opérations :

Les opérateurs arithmétiques

Les opérations arithmétiques travaillent avec les nombres. Soit les types entiers et flottants. Ils permettent de modifier mathématiquement la valeur d'une variable

Opérateur	Dénomination	Effet	Exemple	Résultat (avec x valant 7)
+	opérateur d'addition	Ajoute deux valeurs	x+3	10
-	opérateur de soustraction	Soustrait deux valeurs	x-3	4
*	opérateur de multiplication	Multiplie deux valeurs	x*3	21
/	plus: opérateur de division	Divise deux valeurs	x/3	2.3333333
%	opérateur modulo	Retourne le reste de la division entière de l'opérande de gauche par celle de droite	x % 2	1

Les opérateurs de comparaison

Opérateur	Dénomination	Effet	Exemple	Résultat (avec x valant 7)
==	opérateur d'égalité **A ne pas confondre avec le signe d'affectation (=)!!**	Compare deux valeurs et vérifie leur égalité	x==3	Retourne *True* si X est égal à 3, sinon *False*
===	opérateur d'identité	Vérifie l'identité de valeur **et de type** de deux valeurs	a===b	Retourne *True* si a est égal à b et est de même type, sinon *False*
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	x!=3	Retourne 1 si X est différent de 3, sinon 0

!==	opérateur de non identité	Vérifie la non identité de valeur **et de type** de deux valeurs, c'est-à-dire si les deux valeurs n'ont pas la même valeur ou bien sont de types différents.	a!==b	Retourne *True* si a est différent de b ou bien est de type différent, sinon *False*
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	x<3	Retourne *True* si X est inférieur à 3, sinon *False*
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	x<=3	Retourne *True* si X est inférieur ou égale à 3, sinon *False*
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	x>3	Retourne *True* si X est supérieur à 3, sinon *False*
>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	x>=3	Retourne *True* si X est supérieur ou égal à 3, sinon *False*

Les opérateurs logiques (booléens)

Ce type d'opérateur permet de vérifier si plusieurs conditions sont vraies :

Opérateur	Dénomination	Effet	Syntaxe
||	OU logique	Vérifie qu'une des conditions est réalisée	((expression1) || (expression2))
&&	ET logique	Vérifie que toutes les conditions sont réalisées	((expression1) && (expression2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	(!condition)

4 Les structures

4.1 Les structures alternatives / conditionnelles:

Contrairement au traitement séquentiel, La structure alternative ou conditionnelle permet d'exécuter ou non une série d'instruction selon la valeur d'une condition.

Instruction if

L'instruction if est la structure de test la plus basique, on la retrouve dans tous les langages (avec une syntaxe parfois différente d'un langage à l'autre...). Elle permet d'exécuter une série d'instructions (un bloc d'instructions) si jamais une condition est réalisée.

La syntaxe de cette expression est la suivante :

```
if (condition réalisée) {
   liste d'instructions
}
```

Remarques:

- la condition doit être entre des parenthèses
- il est possible de définir plusieurs conditions à remplir avec les opérateurs ET et OU (&& et ||)

L'instruction suivante par exemple teste si les deux conditions sont réalisées:

```
if ((condition1)&&(condition2)){
      instructions
}
```

L'instruction suivante par contre exécutera les instructions si l'une ou l'autre des deux conditions est réalisée :

```
if ((condition1)||(condition2)){
      instructions
}
```

REM : S'il n'y a qu'une instruction dans le bloc d'instruction, les accolades ne sont pas nécessaires...

```
if (x==2) instructionUnique;
```

Instruction if ( ) { ... } else { ... }

L'instruction *if* dans sa forme basique ne permet de tester qu'une condition, or la plupart du temps on aimerait pouvoir choisir les instructions à exécuter **en cas de non réalisation de la condition...**

L'expression *if ... else* permet d'exécuter une autre série d'instruction en cas de non-réalisation de la condition. La syntaxe de cette expression est la suivante :

```
if (condition réalisée) {
   //liste d'instructions
}
else {
  //autre série d'instructions
}
```

Les structures conditionnelles peuvent être imbriquées, il est donc nécessaire d'indenter le code pour plus de lisibilité, c'est-à-dire de décaler à l'aide d'une tabulation chaque bloc d'instruction pour pouvoir rapidement visualiser l'imbrication des structures !

Instruction if ( ) { ... } else if ( ) { ... } else { ... }

L'instruction *if* peut également contenir plusieurs conditions (et éventuellement le else pour les cas de non réalisation des premières conditions). Dans ce cas les différentes conditions sont testées dans l'ordre de haut en bas. Seront exécutées **uniquement** les instructions correspondant à la première condition testées à vrai. Dès lors l'ordre des instructions doit être donné du plus général vers le plus précis. La syntaxe de cette expression est la suivante :

```
if (première condition réalisée) {
   //liste d'instructions
} else if (seconde condition réalisée) {
  //autre série d'instructions
} else {
   // ...
}
```

Opérateur ternaire ()?:

Si pour une condition, il n'y a qu'une seule instruction à réaliser dans le cas Vrai ou dans le cas Faux, il est possible de faire un test simple avec une structure beaucoup moins lourde grâce à la structure suivante :

```
(condition) ? instruction si vrai : instruction si faux ;
```

Remarques:

- la condition doit être entre des parenthèses
- Lorsque la condition est vraie, l'instruction de gauche est exécutée
- Lorsque la condition est fausse, l'instruction de droite est exécutée

4.2 Les structures itératives / les boucles

Les boucles sont des structures qui permettent d'exécuter plusieurs fois la même série d'instructions tant qu'une condition est réalisée. On appelle cette condition, condition de continuation. Son inverse est appelée la condition de terminaison ou encore condition de sortie.

Boucle while

La boucle *while*, francisée en boucle *tant que*, est une structure qui permet d'exécuter un ensemble d'instructions de façon répétée sur la base d'une condition booléenne (VRAI ou FAUX).

Une boucle while est constituée d'un bloc de code source et d'une condition. À l'exécution, la condition est d'abord évaluée, et si elle est vraie, le bloc de code source est évalué. L'exécution du code est ensuite répétée jusqu'à ce que la condition soit fausse.

La syntaxe de cette expression est la suivante :

```
while (condition réalisée) {
  liste d'instructions
}
```

Cette instruction exécute la liste d'instructions tant que (while) la condition est vraie.

Exemple d'utilisation :

```
var input = 999;
while (input>=10) {
  input = prompt('Entrez un chiffre
          plus petit que 10');
}
alert (input);
```

Ce code, demande à l'utilisateur d'insérer un nombre dans une boite de dialogue. Tant que le nombre inséré sera supérieur ou égal à 10, il recommencera sa demande. Lorsque le nombre inséré est strictement inférieur à 10 (soit lorsqu'il ne répond plus à la condition de continuation), on sort de la boucle et la suite du programme est exécuté : Le dernier nombre inséré est affiché.

Remarques :

Attention, lors de l'écriture de votre code, veillez scrupuleusement aux points suivant :

- Pour entrer la première fois dans la boucle, votre condition de continuation doit être vraie.
- Pour vous assurer de sortir de la boucle, à un moment, vous devez d'une manière ou d'une autre rompre la condition de continuation. Dans le cas inverse, vous créez une boucle infinie, ce qui aura comme conséquence de surcharger, voir monopoliser tout votre processeur et de ne jamais effectuer la suite de votre programme.

La boucle For

En théorie, la boucle while permet de réaliser toutes les boucles que l'on veut. Toutefois, il est dans certains cas utile d'avoir un autre système de boucle plus « condensé », plus rapide à écrire.

L'instruction for permet d'exécuter plusieurs fois la même série d'instructions sur base d'un compteur qui sera itéré à chaque passage de boucle.

Dans sa syntaxe, il suffit de préciser

- le nom de la variable qui sert de compteur ainsi que sa valeur de départ. **→ ex : i = 0**
- la condition sur la variable pour laquelle la boucle s'arrête lorsque la condition n'est plus respectée. De manière basique une condition qui teste si la valeur du compteur dépasse une limite. **→ ex : i<10 (lisez tant que i<10)**
- et enfin une instruction qui incrémente (ou décrémente) le compteur. **→ ex : i++ ;**

La syntaxe de cette expression est la suivante :

```
for (compteur; condition; modification du compteur) {
   liste d'instructions
}
```

Par exemple :

<table>
<tr>
<td>

```
for (i=1; i<6; i++) {

   alert(i) ;

}
```

</td>
<td>Cette boucle affiche 5 fois la valeur de i, c'est-à-dire
1 2 3 4 5

• Elle commence à i=1,
• vérifie que i est bien inférieur à 6, etc... jusqu'à atteindre la valeur i=6, pour laquelle la condition ne sera plus réalisée,
• la boucle s'interrompra et le programme continuera son cours.</td>
</tr>
</table>

Remarques :

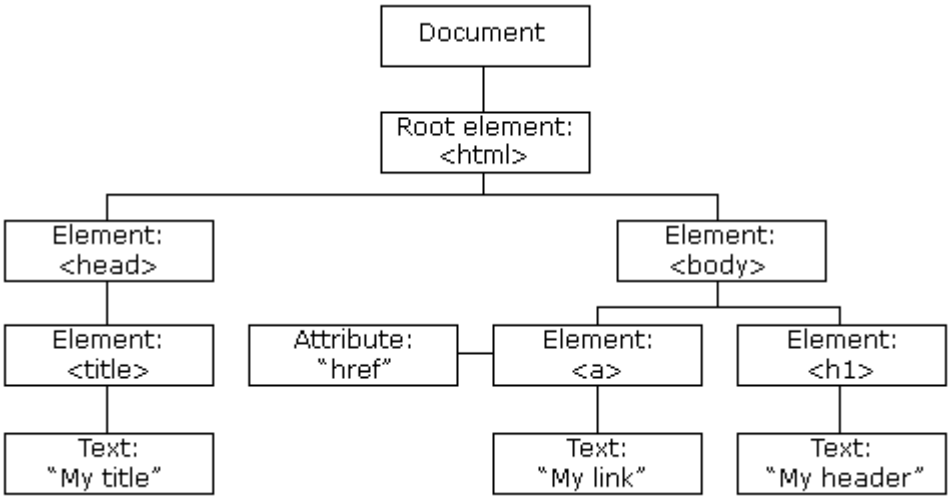
- Il faudra toujours vérifier que la boucle a bien une condition de sortie. Par exemple que le compteur s'incrémente correctement.
- Une instruction *Alert(i);* dans votre boucle est un bon moyen pour vérifier la valeur du compteur pas à pas.
- Il faut bien compter le nombre de fois que l'on veut faire exécuter la boucle:

```
for(i=0;i<10;i++) //exécute 10 fois la boucle (i de 0 à 9)

for(i=0;i<=10;i++)// exécute 11 fois la boucle (i de 0 à 10)

for(i=1;i<10;i++) //exécute 9 fois la boucle (i de 1 à 9)

for(i=1;i<=10;i++) //exécute 10 fois la boucle (i de 1 à 10)
```

5 Le document Object Model

Le HTML DOM (Document Object Model) est un modèle objet standard et une interface de programmation normalisée par le W3C. Il permet aux programmeurs d'accéder et de modifier l'entièreté d'un document html via des scripts. Les objets DOM peuvent représenter une fenêtre, un document, une phrase, une div, un style, …

Quand une page est loadée, le navigateur crée un « Document Object Model » de la page. Par DOM la composition d'un document HTML (ou XML) est représentée sous forme d'un jeu d'objets reliés selon une structure en arbre. A l'aide de DOM un script peut modifier le document présent dans le navigateur en ajoutant ou en supprimant des nœuds de l'arbre.



Grâce au modèle object, Javascript a à sa disposition tout le nécessaire pour créer un HTML dynamique.

- Javascript peut changer tous les éléments HTML présents dans la page.
- Javascript peut changer tous les attributs HTML dans la page.
- Javascript peut changer tous les styles HTML dans la page.
- Javascript peut effacer les éléments et attributs existants dans la page.
- Javascript peut créer de nouveaux événements dans la page.

L'objet « document » représente votre page web. Si vous souhaitez accéder à un des éléments de votre page HTML, vous commencez toujours par accéder à l'objet « document ». Ci dessous, vous trouverez quelques exemples de comment utiliser l'objet objet afin d'accéder et de manipuler l'HTML.

Trouver des éléments HTML :

Méthode	Description
document.getElementById(*id*)	Trouve un élément via son id
document.getElementByTagName(*balise*)	Trouve une série d'éléments via leur balise
document.getElementByClassName(*classe*)	Trouve une série d'éléments via leur classe

Changer des éléments HTML :

Méthode	Description
element.innerHTML = *nouveau contenu HTML*	Change l'intérieur du code HTML d'un élément
element.attribute = nouvelle valeur	Change la valeur de l'attribut d'un élément HTML
element.setAttribut(attribut, valeur)	Change ou initialise la valeur de l'attribut d'un élément HTML
element.style.property = *nouveau style*	Change le style d'un élément HTML

Ajouter ou effacer des éléments HTML

Méthode	Description
document.createElement(*element*)	Crée un élément HTML
document.removeChild(*element*)	Efface un élément HTML
document.appenChild(element)	Ajoute un élément HTML
document.replaceChild(element)	Remplace un élément HTML
document.write(texte)	Écrit dans le flux de sortie du HTML

Ajouter des gestionnaires d'évènements

Méthode	Description
document.getElementById(*id*).onClick = function { *code* }	Assigne un évènement onClick à un élément HTML

6 Les événements / déclencheurs Javascripts

Les événements également appelés déclencheurs sont des actions qui se produisent sur des éléments HTML. Ces actions peuvent être automatisées ou provenir de l'utilisateur. Lorsque le Javascript est utilisé sur la page HTML, le Javascript peut réagir à ces événements. Voici quelques exemples :

- une page HTML, a fini d'être chargée
- Un bouton a été cliqué
- le contenu d'un champ à changé.

Dans l'extrait de code ci-dessous, JS exécutera la fonction *dispayDate()* lorsque le bouton sera cliqué.

```
<button onclick="displayDate()">The time is?</button>
```

Voici quelque exemples d'événements :

[*vous trouverez la liste complète sur* ***http://www.w3schools.com/jsref/dom_obj_event.asp***]

Événements liés à la souris

Méthode	Description
Onclick	L'événement se produit lorsque l'utilisateur clique sur un élément
Oncontextmenu	L'événement se produit lorsque l'utilisateur clique avec le bouton droit de la souris sur un élément pour ouvrir un menu contextuel
Ondblclick	L'événement se produit lorsque l'utilisateur double-clique sur un élément
Onmousedown	L'événement se produit lorsque l'utilisateur appuie sur un bouton de la souris sur un élément
Onmouseenter	L'événement se produit lorsque le pointeur est déplacé sur un élément
Onmouseleave	L'événement se produit lorsque le pointeur est déplacé hors d'un élément
Onmousemove	L'événement se produit lorsque le pointeur se déplace alors qu'il se trouve sur un élément
Onmouseover	L'événement se produit lorsque le pointeur est déplacé sur un élément, ou sur l'un de ses enfants
Onmouseout	L'événement se produit lorsqu'un utilisateur déplace le pointeur de la souris hors d'un élément ou d'un de ses enfants
Onmouseup	L'événement se produit lorsqu'un utilisateur relâche un bouton de la souris sur un élément

Événements liés à la clavier

Méthode	Description
Onkeypress	L'événement se produit lorsque l'utilisateur appuie sur une touche
Onkeypress	L'événement se produit lorsque l'utilisateur appuie sur une touche
Onkeyup	L'événement se produit lorsque l'utilisateur relâche une touche

Événements liés à la fenêtre

Méthode	Description
Onabort	L'événement se produit lorsque le chargement d'une ressource a été interrompu
Onerror	L'événement se produit lorsqu'une erreur se produit lors du chargement d'un fichier externe
Onload	L'événement se produit lorsqu'un objet a chargé
Onresize	L'événement se produit lorsque la vue du document est redimensionnée
Onscroll	L'événement se produit lorsque la barre de défilement d'un élément est défilée

etc.

7 Les tableaux

Les variables de Javascript ne permettent de stocker qu'une seule donnée à la fois. Or, étant donné qu'il est souvent utile de manipuler de nombreuses données, le concept de variable se révèle parfois insuffisant, car il devient difficile de gérer un grand nombre de variable distinctes.

Pour y remédier Javascript propose une structure de données permettant de stocker l'ensemble de ces données dans une "variable commune" : le tableau.

Un tableau, en Javascript, est donc une variable pouvant contenir plusieurs données indépendantes, indexées par un numéro, appelé **indice**.

> *L'indice d'un tableau est ainsi l'élément permettant d'accéder aux données qui y sont stockées.*

Exemple :

Données :	254	569	658	748	758	694	148	364	569	998	475	586	225
Indices :	0	1	2	3	4	5	6	7	8	9	10	11	12

- **Remarquez que le premier élément d'un tableau porte toujours l'indice 0 !**
- **Dans un tableau à n éléments, le nième élément porte ainsi l'indice n-1.**

Création

Le langage Javascript fournit plusieurs façons de créer un tableau :

- `var MonTableau = [] ;`
- `var MonTableau = new Array();`

(La première méthode est conseillée.)

Il est possible d'initialiser des valeurs à la création

- `var MonTableau = ["donnée 1", "donnée 2"];`
- `var MonTableau = new Array("donnée 1", "donnée 2");`

Accès aux données

L'accès aux éléments du tableau se fait en écrivant

le nom du tableau suivi de crochets contenant l'indice de l'élément.

```
var MonTableau = [ "Eaulive",
    "Asevere",
    "Kalamit",
    "Serge",
    "Chat_Teigne",
    "BmV"];

alert("Le 4ième élément est : " + MonTableau[3]);
```

// Affichera "Le 4ième élément est Kalamit"

Listage des valeurs

Pour lister l'intégralité du tableau, il nous faut utiliser une boucle. Il va nous être utile de connaître la "longueur" du tableau (le nombre d'indices qu'il possède). Pour cela, on fait appel à la méthode **length**. Ainsi, on accède aux valeurs de notre tableau grâce à ses indices comme ceci :

ex1 :

```
for (i = 0; i <= mon_tableau.length-1; i++)
{
      alert(mon_tableau[i]);
}

----------------------------------------
```

ex2 :

```
var listing = "" ;
for (i = 0; i <= mon_tableau.length-1; i++)
{
      listing = listing + " " + mon_tableau[i];
}
var el = document.getElementById("maDiv") ;
el.innerHTML = listing ;
```

Manipulation de tableaux

Le langage Javascript propose l'objet Array, comportant de nombreuses méthodes permettant de manipuler les tableaux, c'est-à-dire d'insérer, supprimer, ou extraire des éléments dans le tableau et également de trier les élément du tableau.

8 Les fonctions

Depuis les premiers exercices réalisé en javascript, vous avez déjà rencontré les fonctions. Par exemples, les premières se présentaient sous la forme suivante : `alert()`, `prompt()`, et `parseInt()`. En les utilisant, vous avez pu constater que chacune de ces fonctions avait pour but de mener à bien une action précise, reconnaissable par un nom explicite. Le principe d'une fonction est d'exécuter un extrait de code répondant à un objectif. Utiliser des fonction simplifie également le code. Vous l'appelez en lui passant éventuellement quelques paramètres, elle va ensuite exécuter le code qu'elle contient puis va renvoyer un résultat en sortie.

Le plus gros avantage d'une fonction est que vous pouvez exécuter un code assez long et complexe juste en appelant la fonction le contenant. Cela réduit considérablement votre code et le simplifie d'autant plus ! Seulement, vous êtes bien limités en utilisant seulement les fonctions natives du JavaScript. C'est pourquoi il vous est possible de créer vos propres fonctions, c'est ce que nous allons étudier tout au long de ce chapitre.

Exemple 01 :

```
function myFunction(p1, p2) {
      return p1 * p2;
   // The function returns the product of p1 and p2
}
```

Une fonction JavaScript est un bloc de code conçu pour effectuer une tâche particulière.

8.1 Définition de la fonction

Avant d'utiliser une fonction, nous devons la définir. La façon la plus courante de définir une fonction dans JavaScript est en utilisant

- le mot-clé « function »
- **suivi d'un nom de fonction unique de votre choix,**

  Les noms de fonctions peuvent contenir des lettres, des chiffres, des caractères de soulignement (underscore) et des signes de dollar (mêmes règles que les variables).
- **d'une liste de paramètres** (qui peut être vide)

  Les noms des paramètres sont séparés par des virgules
- **et d'un bloc d'instructions entouré d'accolades**

Note importante : Une fonction JavaScript n'est exécuté que lorsqu'elle est appelée !

Exemple 02 :

```
function name(parameter1, parameter2, parameter3) {
    code to be executed
}
```

8.2 Invocation de la fonction (Appel de fonction)

Le code à l'intérieur de la fonction sera exécuté lorsque celle-ci est invoquée dans le code. Cela peut provenir :

- Lorsqu'un événement se produit (lorsqu'un utilisateur clique sur un bouton)
- Lorsqu'il est invoqué (appelé) à partir du code JavaScript
- Automatiquement (auto-invoqué)

  REM : Une expression auto-appelante est invoquée (démarrée) automatiquement, sans appel. Les expressions de fonction s'exécutent automatiquement si l'expression est suivie de (). Vous ne pouvez pas auto-invoquer une déclaration de fonction. Vous devez ajouter des parenthèses autour de la fonction pour indiquer qu'il s'agit d'une expression de fonction:

Exemple 03 : Appel de fonction depuis un évènement en HTML

```
<html>
   <head>
      <script type="text/javascript">
         function sayHello(){
            document.write ("Hello there!");
         }
      </script>
   </head>
   <body>
      <p>Click the following button to call the function</p>
      <form>
         <input type="button" onclick="sayHello()" value="Say Hello">
      </form>
      <p>Use different text in write method and then try...</p>
   </body>
</html>
```

Exemple 04 : Appel de fonction depuis un code Javascript

```
function myFunction(a, b) {
  alert(a * b);   // Function display the product of a and b
}

myFunction(4, 3);
```

Exemple 05 : Fonction anonyme (aucun nom n'est déclaré, auto-invoquante dans un code JS)

```
(function () {
  var x = "Hello!!";    // I will invoke myself
})();
```

8.3 Instruction Return

L'instruction return met fin à l'exécution d'une fonction et définit une valeur à renvoyer à la fonction appelante. En d'autre termes, une fonction est capable de retourner une valeur que l'on stocke alors dans une variable.

Attention : Il est important de préciser que dans ce cas, **une fonction ne peut retourner qu'une seule et unique valeur, pas plus !** Il est toutefois possible de contourner légèrement le problème en renvoyant un tableau ou un objet.

Lorsque JavaScript atteint une instruction de retour (« return »), la fonction arrête son exécution et retourne à l'appelant. Aucune autre instruction suivante dans le corps de la fonction ne sera exécutée.

Exemple 06 : Calculez le produit de deux nombres et renvoyez le résultat:

```
function myFunction(a, b) {
    return a * b;                // Function returns the product of a and b
}
var x = myFunction(4, 3);
// Function is called, return value will end up in x
```

Le résultat dans x sera : 12

Dès lors qu'elle possède une valeur de retour, une fonction peut être utilisée de la même manière que l'on utilise une variable, dans tous les types de formules, affectations et calculs.

Exemple 07 : Au lieu d'utiliser une variable qui stocke la valeur de retour de la fonction :

```
function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
var x = toCelsius(77);

var text = "The temperature is " + x + " Celsius";
```

Vous pouvez utiliser directement la fonction comme une valeur:

```
function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
var text = "The temperature is " + toCelsius(77) + " Celsius";
```

8.4 Pourquoi les fonctions?

- Vous pouvez réutiliser le code.
  Définissez le code une fois et utilisez-le plusieurs fois.
- Vous pouvez utiliser le même code plusieurs fois avec des arguments différents, pour produire des résultats différents.

Rappels et notes supplémentaires :

- Il existe des fonctions natives, mais il est aussi possible d'en créer, avec le mot-clé `function`.
- Les variables déclarées avec `var` au sein d'une fonction ne sont accessibles que dans cette fonction. Elles sont appelées variables locales à la fonction.
- Il faut éviter le plus possible d'avoir recours aux variables globales.
- Une fonction peut recevoir un nombre défini ou indéfini de paramètres. Elle peut aussi retourner une valeur ou ne rien retourner du tout.

9 Exercices

1 JS : INTERACTION DES FICHIERS

Les compétences acquises à la fin de cet exercice seront les suivantes :

- Connexion des fichiers.
- Sélection d'un extrait du DOM depuis le fichier .js
- Déclencheur « OnClick »
- L'appel de fonction depuis le DOM
- Comment interagir avec la console depuis le fichier .js

Créez 3 fichiers. Un fichier .html, un fichier .css et un fichier .js. Créez 3 boutons colorés (un vert, un rouge, un bleu). A coté, créez trois div colorées des mêmes couleurs. Avec les notions vues au cours, faites en sorte que lorsque vous cliquez sur un bouton, les texte présent dans la div de la même couleur que le bouton change.

2 JS : MA PREMIÈRE CALCULATRICE

Les compétences acquises à la fin de cet exercice seront les suivantes :

- HTML : les formulaires
- Notions de variables et de constantes
- La déclaration, l'initialisation, l'affectation d'une variable.
- Les suites d'instructions
- Les opérateurs arithmétiques (+ , - , * , / , %)
- L'écriture et la concaténation
- les passages de paramètres (introduction)

Créez 3 fichiers. Un fichier .html, un fichier .css et un fichier .js. Créez 2 champs d'insertions de type number. Créez une div finale qui écrira les calculs et le résultat de ceux-ci. Créez 4 boutons : respectivement : +, -, x, /. Lorsque vous cliquez sur chacun d'eux, les caractères saisis dans les champs sont calculés en fonction de l'opérateur sélectionné et affichés dans la div finale (le calcul ainsi que le résultat seront affichés).

3 JS : RÉCAPITULATIF DES COMPÉTENCES.

A l'aide du code écrit pour les deux premiers exercices, répondez aux questions suivantes :

3.1 Quel est le code qui permet de créer le lien entre le fichier html et le fichier javascript.

3.2 Javascript est un langage de programmation événementiel, que cela signifie-t-il ? Quelle élément de code fait directement référence à cette propriété ?

3.3 Dans le cadre de ces exercices, à quoi sert une fonction ? Coté JS, quelle est la syntaxe de sa définition ?

3.4 Qu'est ce qu'un appel de fonction ? Où les trouvent-on dans un code HTML ?

3.5 Comment code-t-on un champ de type number ? Expliquez la propriété « type » de cette partie de code.

3.6 Que signifie l'identifiant var ?

3.7 Je souhaite faire passer une valeur d'un élément de ma page HTML à travers ma fonction pour pouvoir la réutiliser du coté javascript. Comment m'y prendre ?

3.8 A contrario, je souhaite sélectionner un élément du DOM depuis mon fichier .js. Quel est le code qui me permet d'accomplir cette action ? Exemple je souhaite sélectionner une div dont l'id est « case1 ».

3.9 Il existe un point commun entre la concaténation et l'addition. Lequel ?
Expliquez comment on affecte une addition à une variable.
Expliquez comment on affecte une concaténation à une variable.

3.10 A quoi sert la fonction *parseInt(nb1AsString) ?*
Pourquoi en ai-je besoin ?

3.11 Expliquez le principe d'une variable.

En utilisant les standards d'utilisation. Quel forme puis-je donner à son nom ?

3.12 Quelle instruction me permet de communiquer avec la console ?

Par la suite, à quoi cela peut-il me servir ?

3.13 Expliquez l'extrait de code suivant :

```
var element = document.getElementById("myResponse");
element.innerHTML = "Hey! I said! Hello World!";
```

4 JS : MA PREMIÈRE CALCULATRICE : LES CONSTANTES

A l'inverse d'une variable la déclaration d'une constante se nomme *const NAME = 256 ;*

Ajoutez à votre calculatrice les boutons suivants : Périmètre du cercle, Aire du cercle.

Au déclenchement de ceux-ci, écrivez dans la div de résultat :« Le périmètre / L'aire du cercle d'un rayon de … cm fait … cm ».

Ensuite, améliorez le css de votre calculatrice afin de lui donner un design esthétique au niveau des champs mais également au niveau des boutons. Centrez également votre calculatrice au milieu de la page.

REM : Des balises css permettent par exemple de créer des ombrages.

box-shadow: 0px 0px 8px #eee;
text-shadow: 0px 0px 5px #666666;

5 JS : MA PREMIÈRE CALCULATRICE : COMPARAISON ET PARITÉ

Les compétences acquises à la fin de cet exercice seront les suivantes :

- Les opérateurs de comparaison == , != , < , <= , > , >= , === , !==
- Les opérateurs logiques ||, &&, !
- La structure conditionnelle : if (true) { } else { }

Ajoutez à votre calculatrice un bouton qui compare les deux nombres. La div finale indiquera si le nombre A est plus petit que le nombre B, ou plus grand ou égal.

Bonus : Ajoutez ensuite un bouton qui indique si le nombre du premier champ est pair ou impair.

6 JS : LE JEU DU PLUS OU MOINS

Sur base de fichiers fournis en classe, créez :

un programme qui génère un nombre aléatoire au chargement de la page. Le nombre est généré uniquement au chargement de la page.

l'utilisateur devra ensuite insérer un nombre et cliquer sur le bouton « plus petit ou plus grand ? »

------ les instruction ci-dessus sont fournies dans le fichier -------

Cela déclenchera une fonction qui affichera ensuite à l'utilisateur dans une espace de la page html si le nombre généré aléatoirement est plus grand ou plus petit que le nombre inséré par l'utilisateur.

6.1 Créez la fonction qui indique à l'utilisateur si le nombre aléatoire est plus petit ou plus grand que le nombre final donné. Si le nombre est correct indiquez. « Félicitation, le nombre à trouver était ... ». Insérez ensuite une image de félicitation pour le cas ou l'utilisateur a gagné.

6.2 Vous remarquerez que lorsque le résultat affiché est « plus grand » ou « plus petit » deux fois de suite, on ne détecte pas de ré affichage et cela nuit à la compréhension du jeu.

Ceci est une fonction qui permettra de simuler une réapparition de la div à chaque fois que l'on clique sur le bouton « plus petit ou plus grand ? »

```
// La fonction FadeIn simule une réapparition de la div
function fadeIn(el){

  // el voit sont style : opacité mis à zéro.
  el.style.opacity = 0;
  // el voit sont style : display affecté "block".
  el.style.display = "block";

  // -- Extrait de code simulant une apparition
  (function fade() {
    var val = parseFloat(el.style.opacity);
    val = val + 0.1;
    if ( val < 1) {
      el.style.opacity = val;
      requestAnimationFrame(fade);
    }
  })();
}
```

Placez cette fonction dans votre fichier.

L'appel de fonction se fait avec comme paramètre : l'élément du DOM que vous avez sélectionné (votre div). L'appel doit se faire de manière proche du moment où l'on réécris le contenu de la div. La syntaxe de l'appel est la suivante :

```
fadeIn(elementDuDom*);
```

***Nom à changer**

Placez cet appel dans votre fichier afin d'obtenir le résultat souhaité.

6.3 Dépassement :

OnClick=« .... » est un déclencheur javascript.

Il en existe d'autres, par exemple :

```
onKeyPress=« nomDeLaFonction(parametre1, ...)»
```

Cela signifie que cette balise insérée dans un champ texte (coté html) permettra de déclencher la fonction nommée « nomDeLaFonction » à chaque frappe de clavier dans ce champ.

Transformez votre code afin de pouvoir appeler la fonction du jeu depuis une insertion au clavier dans le champ. Faites en sorte que le nombre ne soit testé que si l'utilisateur appuie sur « enter ».

7 STRUCTURES CONDITIONNELLES : RENFORCEMENT

7.1 INDENTATION : Considérez la séquence d'instructions suivante:

```
var texte ;
if (A>B)
if (A>10)
texte = "premier choix \n"; else if (B<10)
texte = "deuxième choix \n"; else
if (A==B) texte = "troisième choix \n";
else texte = "quatrième choix \n";
```

Dans la colonne de gauche, copiez la séquence d'instructions en utilisant des tabulateurs et des accolades pour marquer les blocs **if – else** appartenant ensemble.

Après avoir répondu aux deux questions ci-dessous, cette première solution vous parait-elle pertinente ? Proposez une deuxième solution dans la colonne de droite.

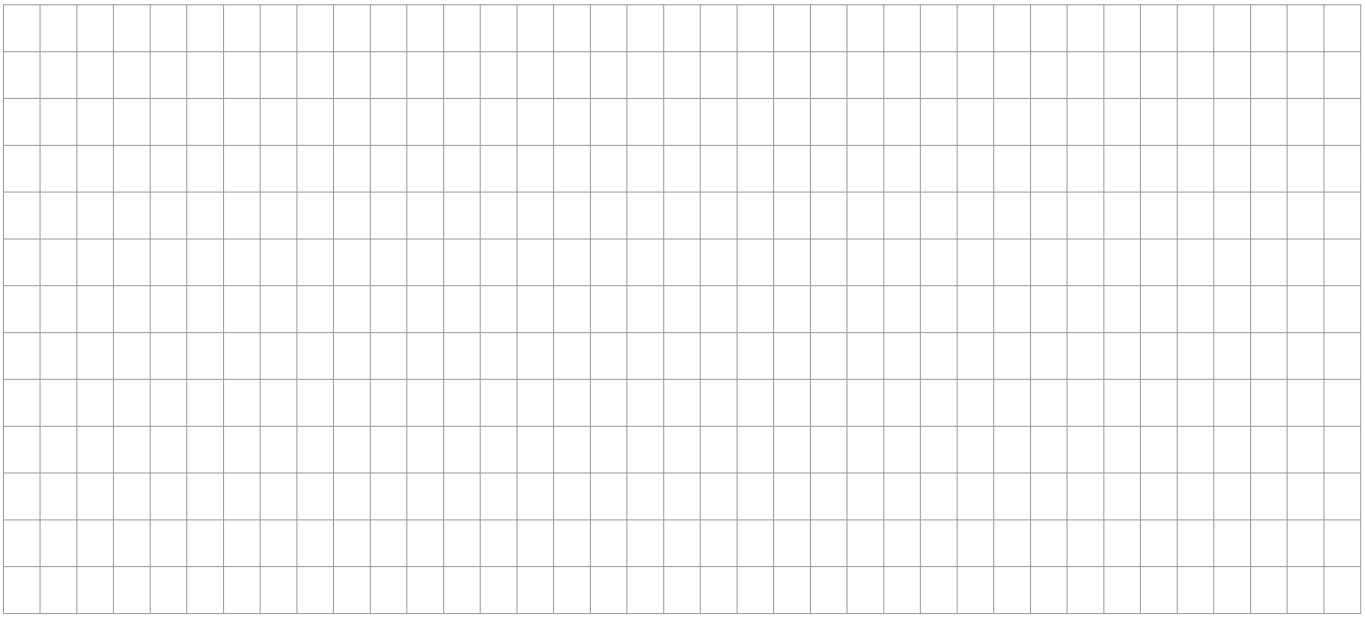
(Dans les deux cas) Pour quelles valeurs de A et B obtient-on les résultats: *premier choix*, *deuxième choix*, ...

(Dans les deux cas) Pour quelles valeurs de A et B n'obtient-on pas de réponse sur l'écran?

7.2 Minimum

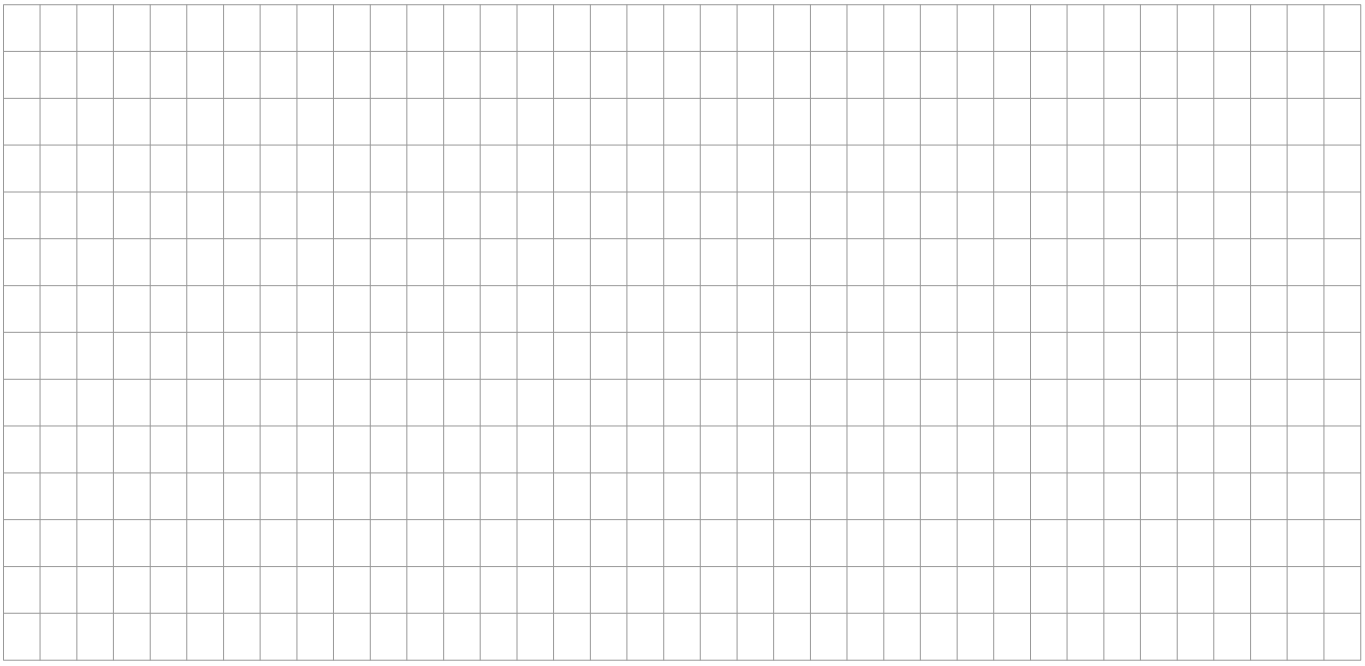
Écrivez un programme qui :

1. initialise 2 variables a et b.
2. déclare une variable *nbmin* également entière
3. à l'aide d'un if, fait en sorte que la variable *nbmin* contienne la valeur minimale de a et b.
4. affiche cette valeur minimale ainsi déterminée.



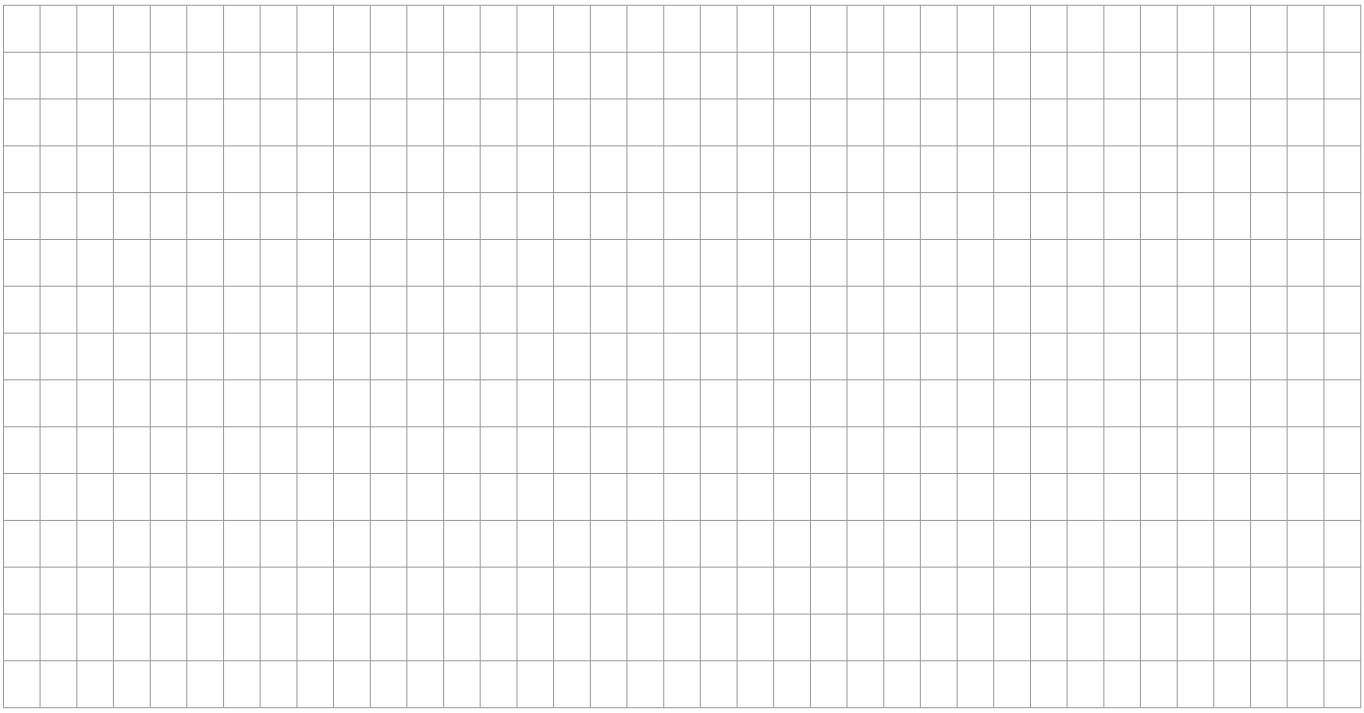
7.3 Minimum (2)

Même exercice mais pour 3 nombres entiers.



7.4 Maximum

Écrire un programme qui calcule le maximum entre 3 entiers a, b et c.



7.5 Signe du produit

Écrivez un programme qui affiche le signe du produit de a et b sans faire la multiplication. Pour cela, vous testerez uniquement si le résultat est inférieur à 0.

7.6 Age

Initialisez une variable entière qui corresponde à l'âge d'une personne, si la personne a au moins 18 ans, alors on affiche "peut voter", sinon, on affiche "ne peut pas voter". Refaire la même chose en le formulant négativement : si la personne n'a pas 18 ans, elle ne peut pas voter, sinon elle peut.

7.7 Mention

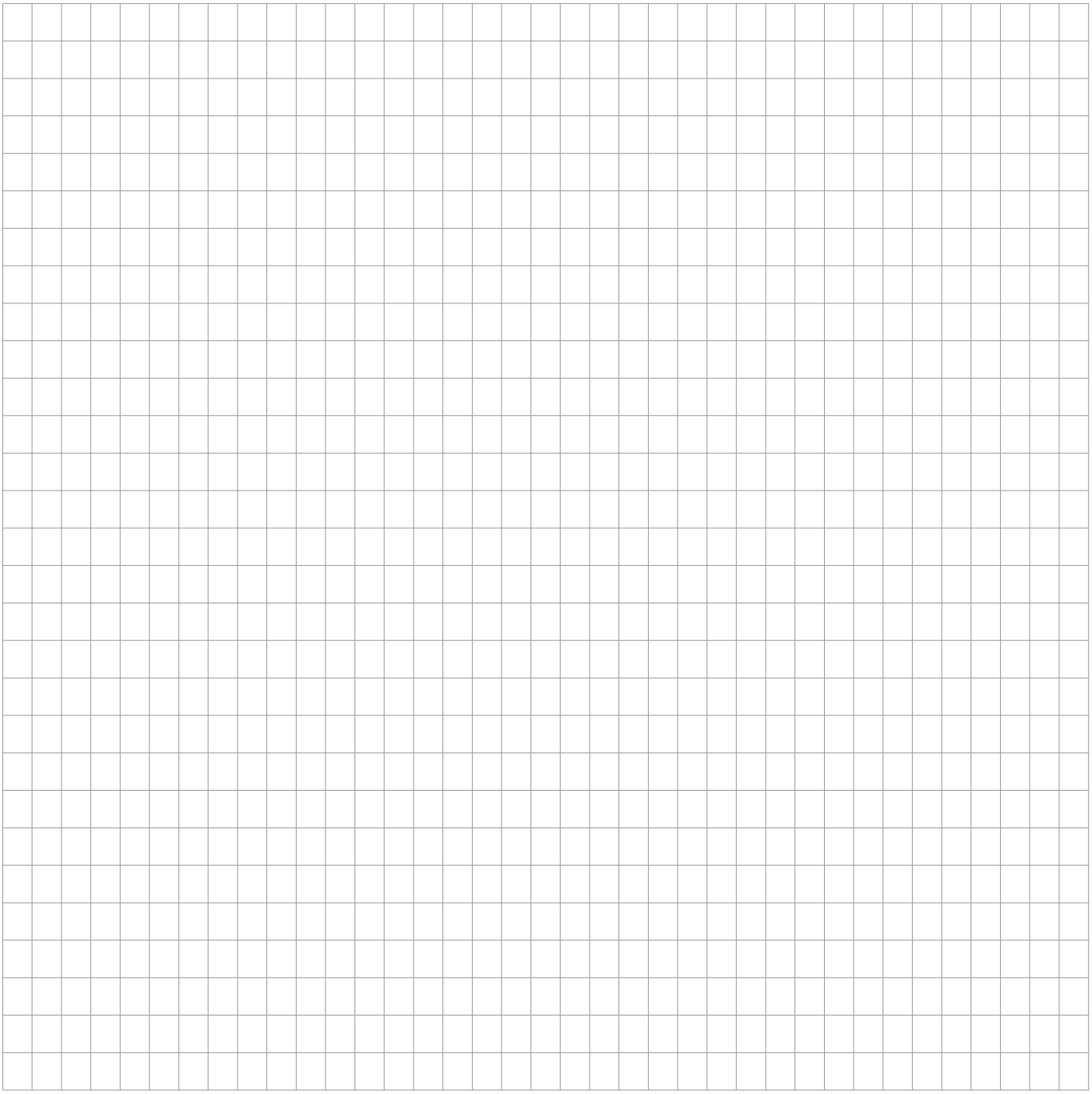
Nous désirons afficher la mention obtenue par un élève en fonction de la moyenne de ses notes. S'il a une moyenne strictement inférieure à 10, il est recalé. S'il a une moyenne entre 10 (inclus) et 12, il obtient la mention passable. S'il a une moyenne entre 12 (inclus) et 14, il obtient la mention assez bien. S'il a une moyenne entre 14 (inclus) et 16, il obtient la mention bien. S'il a une moyenne supérieure à 16 (inclus) il obtient la mention très bien. Écrire les instructions nécessaires.

7.8 Année bissextile

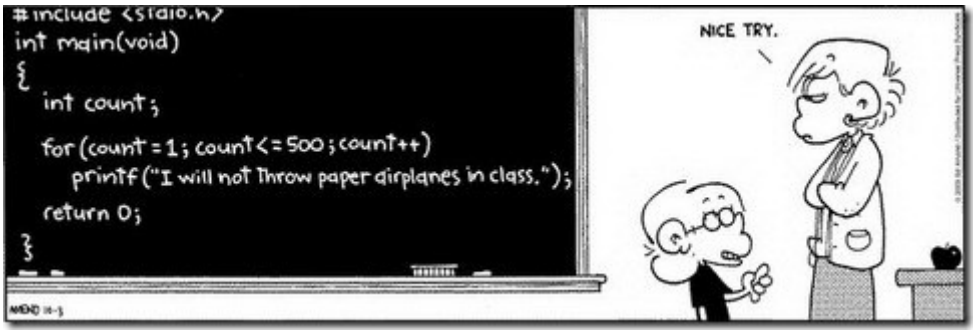
Si l'année A n'est pas divisible par 4, alors elle n'est pas bissextile. Si A est divisible par 4, l'année est bissextile sauf si A est divisible par 100 et pas par 400.

Exemples :

– 1901 n'est pas bissextile car non divisible par 4
– 2004 est bissextile car divisible par 4 et pas par 100
– 2100 n'est pas bissextile car divisible par 4, divisible par 100 mais pas par 400
– 2000 est bissextile car divisible par 4, par 100 et par 400. Écrire un programme qui détermine si une année est bissextile ou non.



8 JS : STRUCTURES ITÉRATIVES : GÉNÉRATEURS DE PUNITIONS



Créez une page HTML qui contient deux champs

- Le premier champ servira à insérer une phrase.
- Le second champ servira à insérer un nombre n.

et une div qui présentera la forme d'un tableau noir.

Un bouton intitulé « copier la punition [FOR] » vous permettra de recopier n fois la phrase sur le tableau noir de votre page HTML. Pour ce faire vous utiliserez la structure itérative *For*. Bien entendu l'écriture du tableau noir et uniquement du tableau noir sera manuscrite. Utilisez une police Google font.

Si la punition est trop grande, le tableau noir s'étendra automatiquement en hauteur.

Une fois l'exercice terminé, créez un second bouton qui aura strictement le même objectif mais utilisera la structure *while*. Le bouton sera intitulé « copier la punition [WHILE] ».

9 STRUCTURES ITÉRATIVE

Corrigez les extraits de code suivants :

```
function copierViaWHILE(phrase, nombre){
     var el = document.getElementById("blackboard");
     var i;

     while (i=0; i<nombre; i++){
          el.innerHTML= el.innerHTML + "<br>" + phrase
     }
}
```

```
function copierViaFOR(mph,mnbr){
     var el = document.getElementById("phrase");
     el.innerHTML = mph + " " + mnbr;
     var i;
     }
     for(i = 0; i < mnbr; i++){
         el.innerHTML = el.innerHTML + "<br>" + mph;
     }
```

```
function copierViaWHILE(nbre, Mot){
     var x;
     var el = document.getElementById('tableau')
     while(x != nbre ){
          el.innertHTML = el.innertHTML + "<'br'>" + Mot;
          var x = x + 1;
     }
}
```

10 DRILL : STRUCTURES ITÉRATIVES

- Dans une page HTML, via un input, demandez à l'utilisateur un nombre n, affichez ensuite la suite des nombres de 1 à n.
- Dans une page HTML, via un input, demandez à l'utilisateur un nombre n entre 1 et 10, affichez ensuite la table de multiplication de ce nombre.
- Dans une page HTML, via un input, demandez à l'utilisateur un mot de passe. Tant que le mot de passe est incorrect, affichez « Veuillez réessayer ». Dès que le mot de passe est correct, affichez « Vous avez accès aux données ».
- Créez un programme qui demande à l'utilisateur de taper un mot de passe. Celui-ci a trois essais. Si le caractère tapé est 'indbg305', on affiche "gagné", et on arrête. Si au bout de 3 essais, l'utilisateur n'a pas toujours tapé de "0", on sort en affichant "perdu". → Pour réaliser ce programmer, utilisez les fonctions prompt() et alert().

11 JAVASCRIPT : EXERCICE D'INTÉGRATION : EXERCICE LIFE

12 JAVASCRIPT : EXERCICE SUR LES TABLEAUX

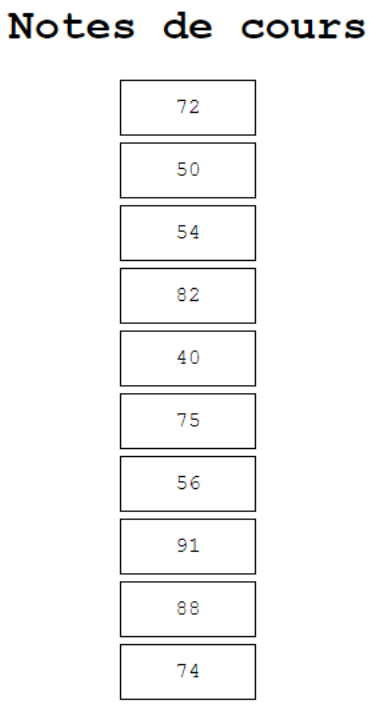
12.1 Créez une page HTML , liée à une page script.

12.2 Déclarez un tableau en javascript nommé « notes » et contenant les valeurs suivantes :

72	50	54	82	40	75	56	91	88	74

12.3 Affichez ce tableau dans la page HTML dès le chargement de la page. Chaque cellule du tableau doit être représentée par une div dont la classe serait « case » et dont l'identifiant serait « noteX », X étant le numéro de l'indice.

Votre résultat doit s'approcher de l'interface suivante :



12.4 Ajoutez un bouton qui calcule le total des notes du tableau. Le bouton et le résultat se situeront à droite du tableau.

12.5 Ajoutez un bouton qui calcule la moyenne des notes du tableau. Le bouton et le résultat se situeront à droite du tableau. Si la moyenne est inférieure à 50, la moyenne se présentera sur un fond rouge. Si la moyenne est entre 50 et 65 inclus la moyenne se présentera sur un fond orange. Si la moyenne se situe au dessus de 65, elle se présentera sur un fond vert. Le nombre affiché sera arrondi à deux chiffres après la virgule.

12.6 Ajoutez un bouton qui extrait la note maximale. Le bouton et le résultat se situeront à droite du tableau.

12.7 Ajoutez un bouton qui extrait la note minimale. Le bouton et le résultat se situeront à droite du tableau.

12.8 Ajoutez un champ sous le tableau et un bouton qui vous permette d'ajouter une note au tableau. Bien entendu le tableau affiché en HTML sera également mis à jour. L'instruction pour ajouter un élément au tableau est la suivante.

```
NomDuTableau.push(nbrAInserer) ;
```

! Veillez à insérer un entier dans le tableau, sinon les différents calculs seront faussés.

12.9 Veillez à mettre en page votre interface de la manière suivantes

13 JAVASCRIPT : EXERCICE SUR LES TABLEAUX (2)

13.1 Créez une page HTML , liée à une page script.

13.2 Déclarez un tableau en javascript nommé « notes » et contenant les valeurs suivantes :

15	20	5	12	10	7	0	20	18	14

13.3 Affichez ce tableau dans la page HTML dès le chargement de la page. Chaque cellule du tableau doit être représentée par une div dont l'identifiant serait « noteX »,
X étant le numéro de l'indice.

13.4 Les notes au dessus ou égales à 12 seront présentées sur un fond bleu, les notes en dessous de 12 seront présentées sur un fond rouge.

13.5 Construisez un bouton, qui vous indique le nombre de notes supérieures ou égales à 12.

13.6 Ajoutez un champ sous le tableau et un bouton qui vous permette d'ajouter une note au tableau. Bien entendu le tableau affiché en HTML sera également mis à jour. L'instruction pour ajouter un élément au tableau est la suivante.

```
NomDuTableau.push(nbrAInserer) ;
```

! Veillez à insérer un entier dans le tableau, sinon les différents calculs seront faussés.

14 JS : LES FONCTIONS

L'objectif de l'exercice est de faire varier l'interface HTML en fonction d'une thématique choisie.

Exercice 1 :

Dans l'interface html pour chaque élément <li> de la liste, ajoutez le déclencheur onClick.
Sous chacun de ces déclencheurs, faites un appel de fonction vers la fonction "selectTheme".
Chaque appel de fonction possède en paramètre une chaine de caractères (entourée de simple guillemets).
Le bouton pinceau envoie en paramètre la chaine "peinture";
Le bouton clé de sol envoie en paramètre la chaine "musique";
Le bouton bulle envoie en paramètre la chaine "bandeDessinee";
Le bouton cinema envoie en paramètre la chaine "cinema";
Le bouton livre envoie en paramètre la chaine "poesie";

Ces différentes chaînes, sont les clés qui permettront la recherche dans le tableau

Exercice 2.

La fonction selectTheme possède 3 appels de fonctions :
une variable "txt" recoit le résultat de la fonction "rechercheTxt"
une variable "pix" recoit le résultat de la fonction "recherchePicture"
le dernier appel envoie ces deux variables en paramètre à la fonction "afficherRslt"

Les détails des fonctions sont donnés ci-dessous.

Exercice 3.

Le tableau "donnees" est global et contient une liste d'objets.
Ces objets sont composés des attributs :

- theme
- txt
- image

La fonction rechercheTXt possède le paramètre key.
La fonction rechercheTxt fait une recherche sur le tableau "donnees" et teste chaque cellule du tableau.
Lors d'un test, si l'attibut « theme » du tableau est égal au paramètre « key », la fonction retourne l'attribut "txt" de cette case.

Exercice 4.

La fonction recherchePicture possède le paramètre key.

La fonction recherchePicture fait une recherche sur le tableau "donnees" et teste chaque cellule du tableau.

Lors d'un test, si l'attibut « theme » du tableau est égal au paramètre « key », la fonction retourne l'attribut "image" de cette case.

Exercice 5.

La fonction afficherRslt possède deux paramètres : txt et pix.

La fonction exécute les instructions suivantes.

dans une première variable : sélectionnez la div dont l'id est "rslt"

remplacez sont contenu html par le paramètre txt

dans une seconde variable : sélectionnez la div dont l'id est "decoration"

remplacez son attribut css afin que l'image en background recoive la valeur du paramètre pix

remplacez sont attribut css display recoive la valeur "block"

Exercice 6

Améliorez au maximum :)

15 LES FONCTIONS (2)

L'objectif de l'exercice est de créer un convertisseur CMJN → RVB en utilisant au maximum les fonctions. Pour commencer, téléchargez le fichier de base sur google Class Room. Modifiez ce code en suivant les instructions ci-dessous afin d'obtenir un ensemble fonctionnel.

1. Appel depuis le HTML

Depuis le fichier html, ajoutez sous un évènement onClick. l'appel de fonction vers la fonction principale « convert ». La fonction nécessite comme paramètres les 4 valeurs des champs Cyan, Magenta, Jaune, Noir.

2. Contenu de la fonction « convert »

Cette fonction est la fonction principale au convertisseur. Cette fonction exécute les actions suivantes :

Vérifie si le champ cyan possède bien une valeur entre 0 et 100 inclus
(cfr sous-programme « check »)

Vérifie si le champ magenta possède bien une valeur entre 0 et 100 inclus
(cfr sous-programme « check »)

Vérifie si le champ jaune possède bien une valeur entre 0 et 100 inclus
(cfr sous-programme « check »)

Vérifie si le champ noir possède bien une valeur entre 0 et 100 inclus
(cfr sous-programme « check »)

Attribue à une variable locale nommée « valeurRVB » le résultat de la conversion RVB
(cfr sous-programme «toRVB»)

Affiche le résultat obtenu (cfr sous-programme « displayRslt»).

Vous l'aurez compris, la fonction principale ne possède que des appels de fonction vers des sous-programmes annexes.

3. Les sous programmes.

function check(){} ;

Le sous-programme check. La fonction check(). Possède deux paramètres :

val : la valeur du champ,

nameDiv : le nom de la div à sélectionner.

Le sous-programme « check » exécute les actions suivantes :

Si la valeur du champ n'est pas entre 0 et 100

sélectionner l'input dont l'identifiant est nameDiv, mettre son border en rouge.

function toRVB(){} ;

La fonction « toRVB » possède 4 paramètres : c, m, j, n

La fonction « toRVB » a comme objectif de calculer le code hexadécimal d'un ensemble de données CMJN.

Par exemple un ensemble CMJN (55,55,5,0) renverra 7373f2.

Voici le code nécessaire à la conversion :

```
var rouge = Math.round(255 * (1-C/100) *(1-n/100));
var vert = Math.round(255 * (1-m/100) * (1-n/100));
var bleu = Math.round(255 * (1-j/100) * (1-n/100));
var formattedRed = ("0" + rouge.toString(16)).slice(-2);
var formattedGreen = ("0" + vert.toString(16)).slice(-2);
var formattedBlue = ("0" + bleu.toString(16)).slice(-2);
var rvbHexa = formattedRed + formattedGreen + formattedBlue;
```

La fonction doit renvoyer au programme appelant le résultat de la conversion.

function displayRslt(){} ;

Le sous-programme possède un unique paramètre qui contiendra le code hexadécimal obtenu lors de la conversion.

Le sous-programme effectue les actions suivantes :

sélection de la div « rslt » dans une variable. Change son contenu HTML par le code hexadécimal.

sélection de la div « ColoredSquare » dans une seconde variable. Change la couleur du background par la couleur en hexadécimal.