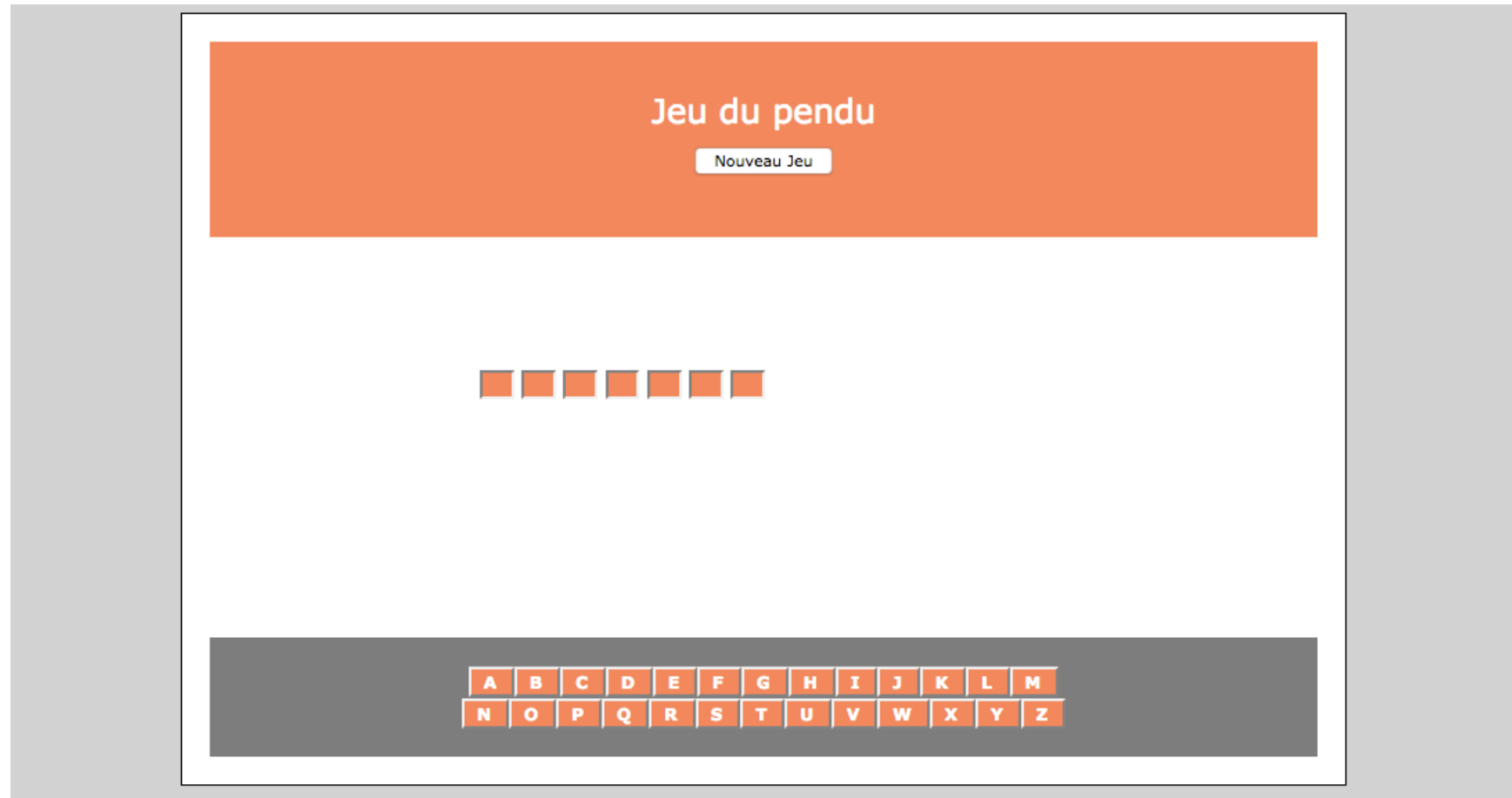


Jeu du pendu

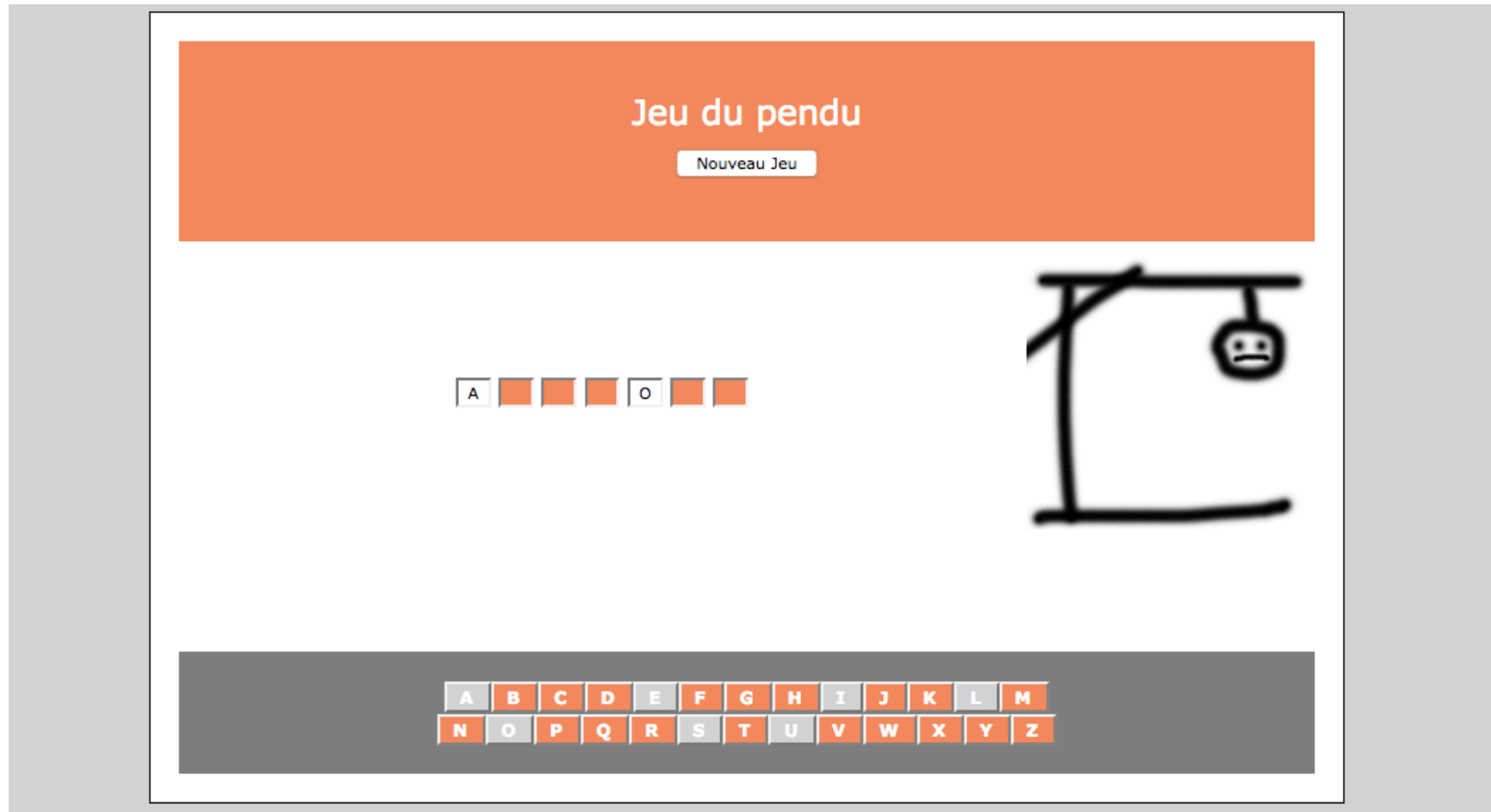
discussions



Situation 1



Situation 2



Situation 3



Situation 4



Interface (html, css, js)

HTML

Utilisation de :

- div
- table (<table>, <tr>, <td>)
- boutons (<button>)
- champs (<input>)
- paragraphe

CSS

- Styles des balises (avec, pour certaines, id et/ou classe)

JAVASCRIPT

Changements dynamiques :

- Etat des champs (<input>)
- Etat des boutons (<button>)
- Etat de la partie

Quels fichiers?

index.html, qui contient un lien sur les fichiers suivants:

- **styles.css** (définition des styles)
- **scripts.js** (code javascript)
- **dico.js** (tableaux de lettres et de prénoms)

fichier index.html

header contenant du texte dans un `<p>` et un bouton qui appelle la fonction `init()`; cette fonction est aussi appelée lors du chargement de la page.

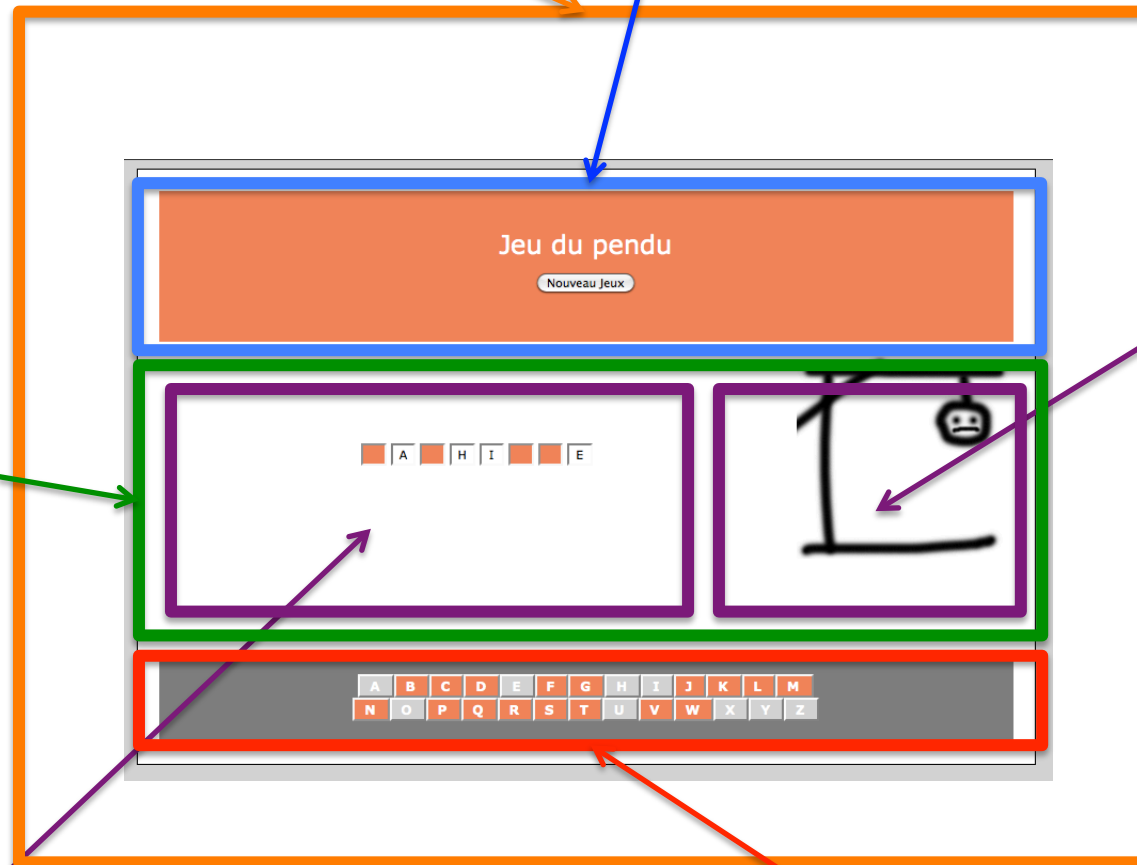
section d'id "corps"

section
contenant
un tableau
de deux cellules
(un rang, deux
colonnes)

cellule de tableau
d'id "tdPendu"
contenant
une image d'id
"imgPendu"

cellule de tableau
d'id "tdMot"

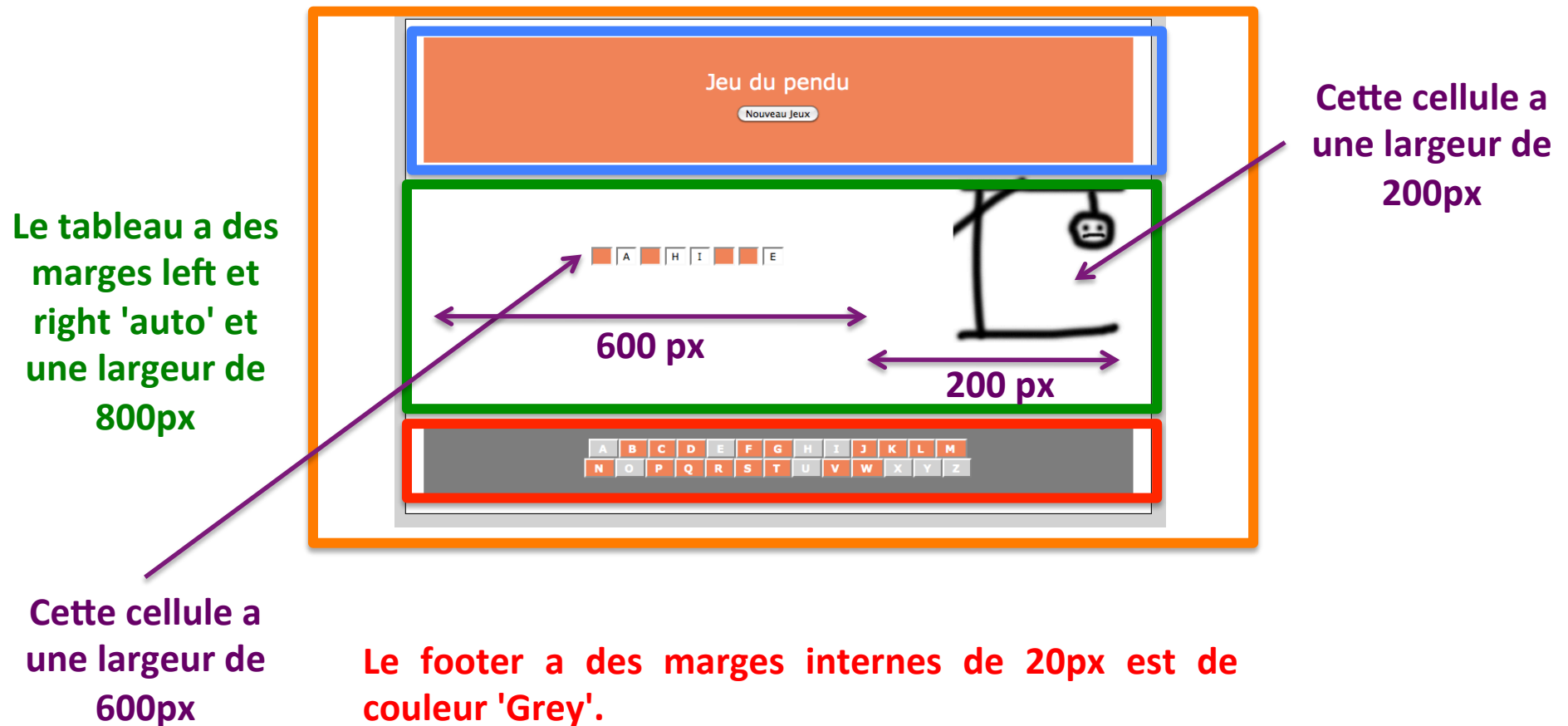
footer d'id "pied"



fichier styles.css

La section principale a une largeur 800px, une bordure noire de 1px, des marges internes de 20px est de couleur 'White'.

Le header a une marge interne de 10px au-dessus et de 20 px en-dessous; il est de couleur 'Coral'.



fichier styles.css

La couleur de fond de la page est 'LightGrey'.

Toutes les cellules des tableau ont une hauteur de 200px et le texte est positionné au milieu de la cellule (verticalement).

Tous les textes (sauf ceux de classe 'btn') sont alignés au centre (horizontalement).

Tous les textes sont écrits en Verdana.

Tous les paragraphes sont écrits en 1.5 em et sont de couleur blanche.

Toutes les balises de type div (attention! Elles ont des noms spécifiques en html5!...) ont une marge auto.

fichier styles.css

Chaque case est une input de classe "champ" et de couleur 'Coral'. (à introduire de manière dynamique via javascript)

Chaque case est une input de type 'button', de classe "btn" (à introduire de manière dynamique via javascript), dans laquelle la taille du texte est de 0.75em. Les lettres sont des petites majuscules (font-variant:small-caps;), écrites en gras (font-weight:bold;) et le texte est justifié (text-align:justify;).

De plus, on spécifie les propriétés suivantes:

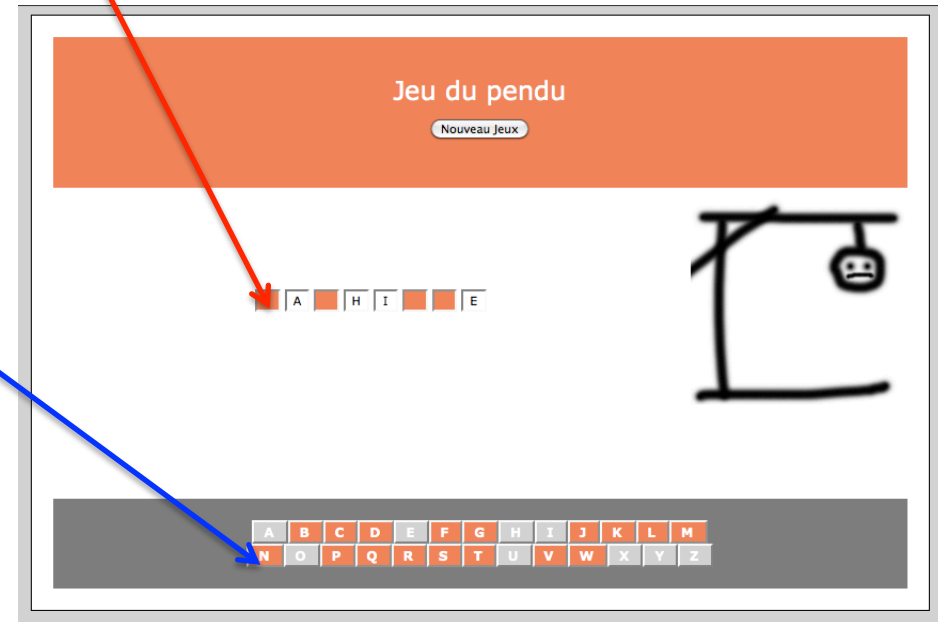
`text-transform:capitalize;`

`text-indent:0px;`

`letter-spacing:2px;`

`color:White;`

`background-color:Coral;`

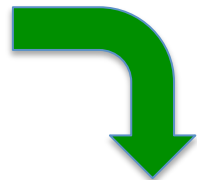


fichier dico.js

Ce fichier javascript contient deux tableaux, tDico (qui contient 200 prénoms) et tLettres (qui contient les 26 lettres de l'alphabet). On doit donc avoir accès initialement à:

- une **liste de prénoms**
 - une **liste de lettres (A -> Z)**
- données dans EXCEL,
à transférer dans un
fichier .js

	A	
1	ADAM	
2	ADELE	
3	ADRIEN	
4	AGATHE	
5	ALAN	
6	ALEX	
7	ALEXANDRA	
8	ALEXANDRE	
9	ALEXIA	
10	ALEXIS	
11	ALICE	
12	ALICIA	
13	AMANDINE	
14	AMBRE	



```
var tDico = new
Array('ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX','ALEXANDRA','ALEXANDRE','ALEXIA','
ALEXIS','ALICE','ALICIA','AMANDINE','AMBRE','AMELIE','AMINE','ANAELLE','ANAIIS','ANDREA','
ANNA','ANTHONY','ANTOINE','ANTONIN','ARNAUD','ARTHUR','AUDREY','AURELIE','AURELIEN','AXEL'
,'AXELLE','BAPTISTE','BASTIEN','BENJAMIN','BRYAN','CAMILLE','CANDICE','CARLA','CASSANDRA',
'CELIA','CHARLES','CHARLOTTE','CHLOE','CLAIRE','CLARA','CLARISSE','CLEMENCE','CLEMENT',')
```

REMARQUE

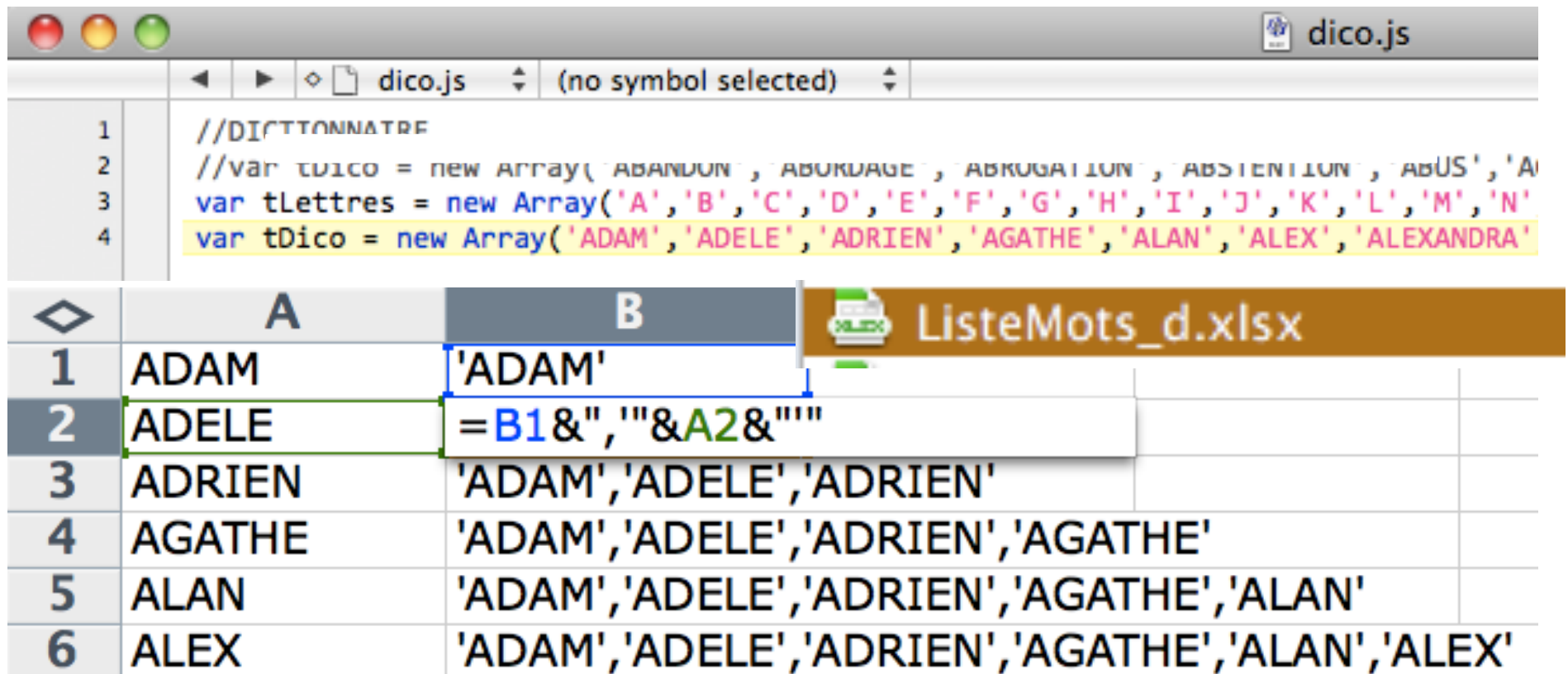
Si on devait gérer un volume de données beaucoup plus grand (tout un dictionnaire, par exemple) ou que ces données aient des relations spécifiques entre elles, il faudrait que qu'elles soient stockées dans une base de données mysql.

Dans notre cas, vu le nombre restreint de données et les relations "simples" entre elles, on peut se contenter de les stocker dans des tableaux déclarés en javascript, ce qui simplifie clairement l'ensemble du code.

```
var tLettres = new
Array('A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U',
'V','W','X','Y','Z')
var tDico = new
Array('ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX','ALEXANDRA','ALEXANDRE','ALEXIA','
ALEXIS','ALICE','ALICIA','AMANDINE','AMBRE','AMELIE','AMINE','ANAELLE','ANAIIS','ANDREA','
ANNA','ANTHONY','ANTOINE','ANTONIN','ARNAUD','ARTHUR','AUDREY','AURELIE','AURELIEN','AXEL',
'AXELLE','BAPTISTE','BASTIEN','BENJAMIN','BRYAN','CAMILLE','CANDICE','CARLA','CASSANDRA',
```

fichier dico.js

Pour transférer les données depuis excel dans javascript, il faut pouvoir écrire des données se trouvant dans une colonne d'un tableau excel dans la syntaxe correspondant à un tableau de strings. Pour réaliser cela, on utilise la concaténation dans excel (cf marche à suivre sur la diapositive suivante).



The image shows a code editor window titled 'dico.js' and an Excel spreadsheet titled 'ListeMots_d.xlsx'.

Code Editor (dico.js):

```
1 //DICTIONNAIRE
2 //var tDico = new Array( 'ABANDON', 'ABORDAGE', 'ABROGATION', 'ABSTENTION', 'ABUS', 'A
3 var tLettres = new Array('A','B','C','D','E','F','G','H','I','J','K','L','M','N'
4 var tDico = new Array('ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX','ALEXANDRA'
```

Excel Spreadsheet (ListeMots_d.xlsx):

	A	B
1	ADAM	'ADAM'
2	ADELE	=B1&"',"&A2&"'"
3	ADRIEN	'ADAM','ADELE','ADRIEN'
4	AGATHE	'ADAM','ADELE','ADRIEN','AGATHE'
5	ALAN	'ADAM','ADELE','ADRIEN','AGATHE','ALAN'
6	ALEX	'ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX'

fichier dico.js

1. Introduire l'instruction suivante dans la case **B1**, du fichier *ListePrenoms_d.xlsx*:
`=""&A1&""`
 (qui correspond à la valeur de la case **A1** entre apostrophes, donc '**ADAM**')
2. Dans la case **B2**, introduire:
`=B1&","&A2&""`
 (qui correspond à la valeur de la case **B1** ('**ADAM**'), concaténée avec une virgule et la valeur de la case **A2** entre apostrophes, donc avec '**ADELE**')
3. Recopier le contenu de la case **B2** dans toute la colonne B.
4. Copier la valeur de la case **B200** et l'introduire dans le fichier dico.js.

	A	B	C	D
1	ADAM	'ADAM'		
2	ADELE			

	A	B	C	D
1	ADAM	'ADAM'		
2	ADELE	'ADAM','ADELE'		
3	ADRIEN			

	A	B	C	D	E	F
1	ADAM	'ADAM'				
2	ADELE	'ADAM','ADELE'				
3	ADRIEN	'ADAM','ADELE','ADRIEN'				
4	AGATHE	'ADAM','ADELE','ADRIEN','AGATHE'				
5	ALAN	'ADAM','ADELE','ADRIEN','AGATHE','ALAN'				
6	ALEX	'ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX'				
7	ALEXANDRA	'ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX','ALEXANDRA'				
8	ALEXANDRE	'ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX','ALEXANDRA','ALEXANDRE'				
9	ALEXIA	'ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX','ALEXANDRA','ALEXANDRE'				
10	ALEXIS	'ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX','ALEXANDRA','ALEXANDRE'				
11	ALICE	'ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX','ALEXANDRA','ALEXANDRE'				
12	ALICIA	'ADAM','ADELE','ADRIEN','AGATHE','ALAN','ALEX','ALEXANDRA','ALEXANDRE'				

Rappel: pour copier une relation dans plusieurs cellules dans excel, il faut sélectionner la poignée de recopie de la cellule et la faire glisser sur toutes les cellules dans lesquelles on veut copier la relation.

VARIABLES GLOBALES

fichier scripts.js

- une variable conservant en mémoire le prénom déterminé aléatoirement (de type string): `var motChoisi = ''`;

(par exemple, on obtiendra aléatoirement: motChoisi = 'AGATHE')

- un tableau contenant les caractères utilisés pour former le prénom choisi (de type tableau): `var tChar`;

(avec l'exemple ci-dessus, on aurait tChar = ['A','G','A','T','H','E'])

Valeurs affectées au début
de la partie, non modifiées
pendant la partie

- une variable comptabilisant le nombre d'erreurs et gérant donc l'état du pendu (de type integer): `var erreur = 0`;
- un tableau de même taille que tChar et initialement vide, contenant les caractères affichés dans les *input* de classe 'champ' (de type tableau):

`var tAffiche`;

(dans cet exemple, on aurait tAffiche = [, 'E', , 'I', , 'A'])



Valeurs modifiées au cours
de la partie en fonction
des entrées de l'utilisateur

2. Traitements (js)

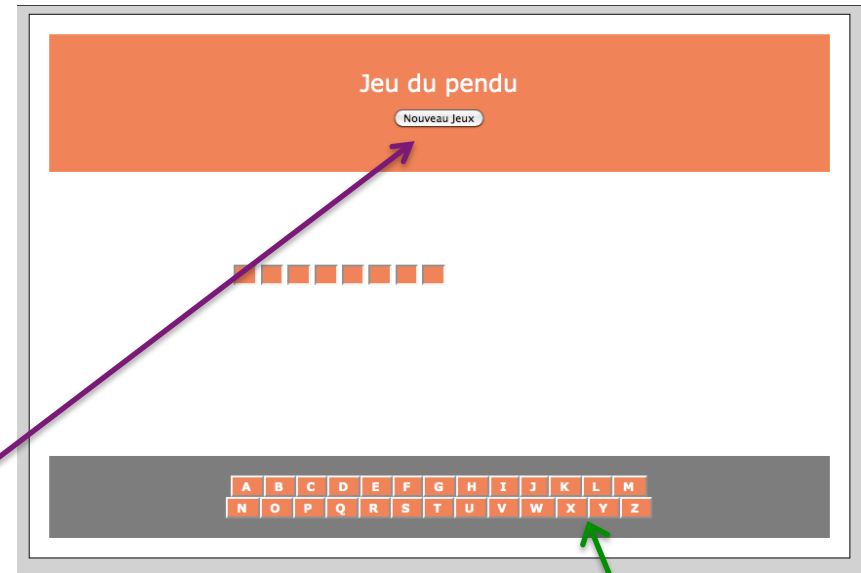
- Au chargement de la page (*onload*)
Initialisation du jeu

- Gestion d'événements

-> soit clic sur le bouton "Nouveau Jeu" → *ré-initialisation du jeu*

-> soit clic sur une lettre →

- Récupération de la lettre proposée par l'utilisateur
- Test de la lettre choisie
 - Appartenance de la lettre au prénom à déterminer ?
- Information sur le choix de la lettre... et conséquences
- Information sur l'état de la partie
 - En cours
 - Gagnée ou perdue



fichier scripts.js

FONCTIONS UTILITAIRES

```
function $(id) {
```

```
    retourner document.getElementById(id);
```

```
}
```

```
function insererHTML(id,txt) {
```

```
    insérer txt dans le innerHTML de la balise d'id='id';
```

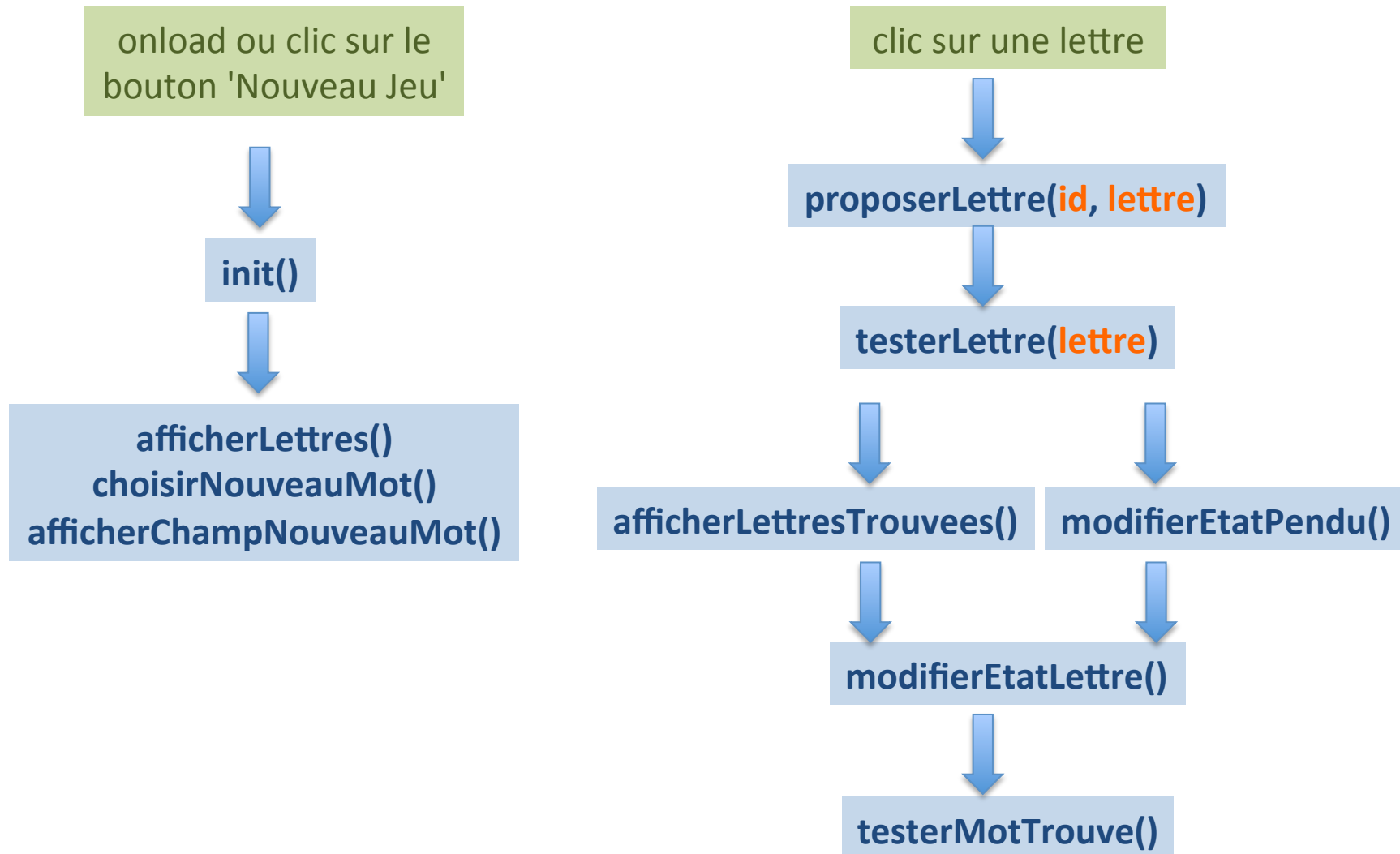
```
}
```

```
function nAlea (inf,sup) {
```

```
    retourner un nombre entier généré aléatoirement  
    et compris dans l'intervalle [ inf , sup [;
```

```
}
```

SCHÉMA GLOBAL



INITIALISATION

fonction init() {

//à appeler au chargement de la page
//et au clic sur le bouton « Nouvelle Partie »

initialiser **erreur** à la valeur 0;
initialiser **l'image** du pendu avec l'image vide (blanche);
afficher **lettres** dans le pied de page: appel de **afficherLettres()**
choisir un nouveau **mot**: appel de **choisirNouveauMot()**
afficher les **champs** représentant les lettres du mot choisi:
appel de **afficherChampNouveauMot()**

}

Cette fonction est appelée lors du chargement de la page ou lorsque l'utilisateur clique sur le bouton « Nouveau jeu ».

AFFICHAGE DES LETTRES

fonction afficherLettres() {

//construit le code HTML à injecter

//dans le pied de la page (lettres dans la div d'id 'pied'

déclarer et *initialiser* une variable locale **txt** (string vide);

répéter pour chaque lettre:

txt += genererNouveauBouton dont la valeur (value) soit la lettre (ex: 'A' pour A), le **id** soit la **lettre**, la classe soit 'btn' et l'action lors d'un onclick soit un appel à la fonction proposerLettre(**id**,**lettre**);

insérer **txt** dans le HTML de la div d'id 'pied';

}

RAPPEL

***RAPPEL:** Si « this » apparaît dans la balise d'id="id_balise", il est équivalent à document.getElementById("id_balise").*

Si l'appel est effectué depuis une balise d'id="id_balise":

this ≡ document.getElementById("id_balise")

this.id ≡ document.getElementById("id_balise").id

this.value ≡ document.getElementById("id_balise").value

CHOIX D'UN NOUVEAU PRÉNOM

fonction choisirNouveauMot() {

//permet de sélectionner un nouveau mot à l'aide d'un nombre
//aléatoire (qui donne la position du mot dans le tableau tDico

déclarer et *initialiser* une variable locale **i** (nbre aléatoire
correspondant à un indice du tableau tDico);

affecter la string situé à l'indice **i** de **tDico** à **motChoisi**;

transformer la string **motChoisi** en un tableau **tChar** //utiliser split('')

affecter la longueur de **tChar** au tableau **tAffiche**;

}

La fonction suivante définie pour les strings renvoie un tableau dont les éléments
correspondent aux caractères de la string txt: txt.split('')

AFFICHAGE DE | | | | | | | |--|---|--|---|--|---| | | E | | I | | A | |--|---|--|---|--|---|

fonction afficherChampNouveauMot() {

//construit le code HTML à injecter

//dans le tableau, 1^{ère} colonne pour afficher les champs

déclarer et *initialiser* une variable locale **txt** (string vide);

parcourir le tableau **tChar**:

txt += genererNouveauChamp + id = 'ch' + **i**;

insérer **txt** dans le HTML de la div d'id 'tdMot'

}

LORSQUE L'UTILISATEUR CLIQUE SUR UNE LETTRE...

Clic sur un bouton lettre



fonction proposerLettre (**lettre**)

//action lancée lorsque l'utilisateur clique sur un bouton lettre

{

si **lettre** pas trouvée alors *modifier* l'état du pendu:

appel de **modifierEtatPendु()** ;

sinon *afficher* la lettre trouvée dans le(s) champ(s) concerné(s):

appel de **afficherLettresTrouvees()** ;

modifier l'état de la lettre et la désactiver:

appel de **modifierEtatLettre(lettre)** ;

tester si le mot a été trouvé: appel de **testerMotTrouve()** ;

}

Remarque: Il est possible aussi de déclarer cette fonction avec un seul paramètre, (la string lettre), qui sera utilisé tour à tour comme id et comme valeur de l'input.

LORSQUE LA LETTRE PROPOSÉE PAR L'UTILISATEUR NE SE TROUVE PAS DANS LE PRÉNOM...

```
fonction modifierEtatPendou() {
```

```
//modifie l'image du pendu et incrémente le décompte des  
//erreurs
```

```
    incrémenter erreur;  
    modifier le src de l'img d'id 'imgPendou' pour faire  
    apparaître un trait supplémentaire;
```

```
}
```

POUR MODIFIER L'ÉTAT D'UNE LETTRE (LORSQU'ELLE A ÉTÉ PROPOSÉE PAR L'UTILISATEUR)

```
fonction modifierEtatLettre(id) {
```

```
//rend inactif le bouton correspondant à la lettre et le colore en  
//gris
```

```
    modifier l'attribut disabled de la balise d'id 'id' en lui  
    attribuant la valeur 'disabled';
```

```
    modifier l'attribut de style background-color de la balise  
    d'id 'id' en lui attribuant la valeur 'LightGrey';
```

```
}
```

POUR AFFICHER L'ÉTAT ACTUEL DU PRÉNOM
(QUI DÉPEND DES LETTRES DEVINÉES PAR L'UTILISATEUR)...

fonction afficherLettresTrouvees() {

//affiche les lettres déjà trouvées par le joueur dans les input d'id
// 'ch'+i situées à l'intérieur de la cellule d'id 'tdMot'

parcourir le tableau **tAffiche**:

afficher les lettres figurant dans le tableau **tAffiche** dans
les inputs adéquats

modifier la couleur de ces inputs (Coral -> White)

}

POUR TESTER SI UNE LETTRE SE TROUVE OU NON DANS LE PRÉNOM

Clic sur un bouton lettre



```
fonction testerLettre (lettre) {
```

```
//test si la lettre proposée est présente dans le mot; si c'est le cas,  
//l'introduire à la bonne (aux bonnes) position(s)
```

```
    déclarer et initialiser une variable booléenne locale trouve à  
    false;
```

```
    parcourir le tableau tChar:
```

```
        si lettre == lettre à la position i de tChar
```

```
            alors lettre est introduite à la position i de tAffiche et
```

```
                trouve = true;
```

```
    retourner trouve;
```

```
}
```

GESTION DE L'ÉTAT DE LA PARTIE

Clic sur un bouton lettre



fonction testerMotTrouve() {

//test si le mot a été trouvé et si le jeu est terminé

déclarer et *initialiser* une variable integer locale **compte** à 0;
parcourir le tableau **tAffiche**:

tester si une lettre est présente à la position i de **tAffiche**
si c'est le cas, *incrémenter* **compte** ;

si **compte** == nombreCaractèreDuMot

insérer 'PARTIE GAGNEE!' dans la balise d'id 'pied';

sinon si **erreur** == 8

insérer ' PARTIE PERDUE! Le mot était ...' dans la balise d'id
'pied';

}