

Module 8 : interactions html-javascript

Pré-requis :

- Langage HTML
- Feuilles de styles CSS
- Bases de Javascript :
 - Définition de variables
 - Définition de fonctions
 - Appel de fonctions
 - Utilisation de fonctions mathématiques (ex : `Math.random()`)
 - Les boucles (for, while, switch)
 - Les expressions booléennes
 - Les chaînes de caractères
 - Les tableaux et opérations sur les tableaux
 - Les appels différés ou périodiques (`SetTimeout()` et `SetInterval()`)
 - Les interactions simples html/JS : fonctions `alert()` et `prompt()`

Objectifs :

- Maîtriser les principales instructions permettant de rendre une page html dynamique en utilisant les interactions entre le langage javascript et les balises html
- Comprendre les concepts généraux de ces interactions (c-à-d être capable d'avoir une "vue d'ensemble" du code final).

Conception du cours :

L'apprentissage sera réalisé sur la base de séries de petits exercices, chaque série permettant de découvrir un ou deux nouveaux concepts (A :<button>, B : innerHTML et `document.getElementById(id)`, etc.). Il est indispensable de maîtriser les exercices noirs pour passer au concept suivant, mais les exercices gris ne sont pas facultatifs : ils doivent être maîtrisés pour la leçon suivante !

La maîtrise de tous les exercices (noirs et gris) permettra de réaliser l'exercice final (jeu du pingouin).

Les concepts théoriques seront présentés par l'enseignant au fur et à mesure de leur apparition dans les exercices.

Exercice :

Pour chaque série, ouvrez un nouveau fichier.html contenant le code de base ci-dessous (balises html de base et balise <script> permettant d'introduire du code en javascript):

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">

    <title>modele_html_js</title>

    <script type="text/javascript">

    </script>
  </head>
  <body>

  </body>
</html>
```

Chaque exercice (1), 2), 3)... à l'intérieur d'une série donnée est résolu dans un même fichier (éventuellement lié à une feuille de style externe) en le testant dans Safari et en le modifiant au fur et à mesure.

Les exercices en noir sont résolus en classe. Les élèves qui ont de l'avance commencent les exercices en gris, qui seront donnés en devoirs à la maison.

A. Appel d'une fonction avec *onload* et `<button>`

- A.1) A l'aide d'une seule ligne de code dans la partie `<script>`, faites apparaître une *fenêtre* « *alert* » annonçant : "Bravo! Votre pingouin a réussi son ascension!"
- A.2) Modifiez la partie `<script>` en introduisant l'instruction « *alert* » dans une *fonction* nommée `resultatAscension()` qui est appelée au-dessous de sa déclaration.
- A.3) Modifiez le programme pour que la *fonction* `resultatAscension()` soit appelée depuis la *balise* `<body>` au moment où la page html est chargée.
- A.4) Modifiez le programme pour que la *fonction* `resultatAscension()` soit appelée depuis la *balise* `<html>` au moment où vous cliquez avec la souris dans la page html.
- A.5) Faites apparaître un *bouton* (*balise* `<button>`) "Afficher le résultat de l'ascension" sur la page html. Modifiez le programme pour que la *fonction* `resultatAscension()` soit appelée au moment où vous cliquez sur ce bouton.
- A.6) Créez une feuille de style externe (nommée `styles_A.css`) dans laquelle vous attribuez à `<button>` une grandeur (*height*, *width*), une couleur (*background-color*) et un style de texte (*font-family*, *font-size*) qui vous paraissent adéquats.
- A.7) Copiez quatre fois le bouton "Afficher le résultat de l'ascension". Remplacez dans chacun des boutons l'attribut *onclick* par *onmousedown*, *onmousemove*, *onmouseup* ou *ondblclick* en indiquant ce choix dans le texte du bouton (par ex., "Afficher le résultat de l'ascension (ondblclick)"). Testez les effets respectifs de ces attributs.
- A.8) Créez un bouton "Début (instant t_0)" et un bouton "Lecture du chronomètre" qui appellent les fonctions `demarrerChrono()` et `lectureChrono()` respectivement. Un clic sur le premier bouton lance un chronomètre interne qui démarre à l'instant t_0 ; un clic sur le second bouton mesure l'instant t_1 et affiche, dans une fenêtre « *alert* », le temps dt qui s'est écoulé entre les instants t_0 et t_1 . Les instructions suivantes vous seront utiles pour créer le chronomètre :

```
var laDate = new Date();  
var t=laDate.getTime();
```

exemple tiré de <http://www.w3schools.com>:

Return the number of milliseconds since 1970/01/01:

```
var d = new Date();  
var n = d.getTime();
```

The result of n could be:

```
1485520524804
```

B. Affichage dans la page html : *innerHTML* et *document.getElementById(id)*

B.1) Créez un bouton nommé "Afficher les résultats du jeu" qui appelle la fonction *resJeu()*; cette fonction affiche une fenêtre « alert » contenant le texte suivant: "Vous avez gagné !".

B.2) Créez une *<div>* ayant un *id* nommé "prenom_res" dans la partie *<body>* de votre programme. (Pour que la *<div>* ne disparaisse pas quand elle est vide, remplissez-la initialement avec la valeur " ", qui signifie "non breakable space") et modifiez la fonction *resJeu()* pour qu'elle affiche le texte "Vous avez gagné !" non pas dans une fenêtre « alert » mais dans le *innerHTML* de la *<div>*. Pour y parvenir, utilisez l'instruction suivante qui donne accès au contenu de la balise d'*id="prenom_res"*:

```
document.getElementById("prenom_res").innerHTML;
```

B.3) Modifiez la fonction *resJeu()* pour que le texte soit affiché dans la *<div>* en format titre *<h3>*.

B.4) Modifiez la fonction *resJeu()* pour que le texte affiché dans la *<div>* en format titre *<h3>* soit aléatoirement "Bravo ! Vous avez gagné !", "Essayez encore ! Vous avez une réponse fausse !" ou "Révissez vos classiques ! Vous avez plusieurs réponses fausses !".

B.5) Faites suivre les textes "Essayez encore ! Vous avez une réponse fausse !" et "Révissez vos classiques ! Vous avez plusieurs réponses fausses !" d'un bouton "Nouveau résultat" qui appelle la fonction *resJeu()*.

C. Collecte d'information : <input>

C.1) Dans la partie <body> : Créez une <div> ayant un id nommé "prenom_res" et contenant initialement le texte "Entrez votre prénom pour commencer la première partie: " et un bouton "Entrer le prénom" qui appelle une fonction nommée premierePartie(). Cette fonction ouvre une fenêtre « prompt » qui vous demande votre prénom (→ variable prenom), puis affiche dans la <div id="prenom_res"> : "prenom, vous pouvez commencer la partie."

C.2) Placez une balise <input> entre le texte "Entrez votre prénom pour commencer la première partie:" et le bouton "Entrer le prénom". Cette <input> possède les attributs suivants : un id="prenom" et une maxlength="30".

Modifiez la fonction premierePartie() pour qu'elle aille lire la valeur de la variable prenom dans la fenêtre de l'<input> (au lieu de la lire dans la fenêtre « prompt » qui disparaît du programme) en utilisant l'instruction suivante :

```
document.getElementById("prenom").value;
```

C.3) Faites disparaître le bouton "Entrer le prénom" et appelez la fonction premierePartie() directement avec la touche « entrée (retour à la ligne) » dans la fenêtre d'<input> en introduisant l'événement suivant:

```
onkeypress='if(event.key=="Enter") premierePartie()';
```

C.4) Placez une deuxième <input> avec un id="nom" et une maxlength="30" à la suite de la précédente et modifiez le texte d'introduction pour que l'utilisateur doive entrer son prénom et son nom puis appuyer sur « entrée (retour à la ligne) » pour appeler la fonction premierePartie(). Modifiez cette fonction pour qu'elle lise le prénom et le nom et qu'elle affiche dans la <div> : "prenom nom, vous pouvez commencer la partie.". Si l'un au moins des deux champs d'input est vide, un message d'erreur apparaît via une fenêtre « alert ».

C.5) Introduisez des valeurs initiales dans vos <input> (par exemple vos propres prénom et nom) en précisant value="..." dans la balise d'<input>.

C.6) Créez un bouton "Initialiser" qui appelle une fonction `init()` ; cette dernière choisit aléatoirement un prénom (resp. un nom) dans une liste contenant quatre prénoms (resp. noms) et vient les placer dans les `<input>` "prenom" et "nom".

C.7) Créez un fichier nommé *fcts_utilitaires.js* contenant les fonctions utilitaires suivantes :

```
function $(id) {  
    return document.getElementById(id);  
}  
function remplirHTML(id,txt) {  
    $(id).innerHTML = txt;  
}
```

Appelez ce fichier en introduisant dans la partie `<head>` de votre document html :

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

et modifiez le code de C.6) en utilisant ces fonctions.

D. Modification des attributs d'une image

D.1) Faites apparaître l'image `pingouin_grand.png` :

```
<img src='images/pingouin_grand.png' id='pingouin'>
```

et créez une feuille de style externe (nommée `styles_DEF.css`) qui sera appelée dans les codes des exercices des séries D, E et F et qui contient les instructions suivantes :

```
img{
    position:absolute;
    left:100px;
    top:50px;
    width:290px;
    height:421px;
}

img#pingouin{
    visibility:visible;
}

img#pingouin_rouge,img#pingouin_vert,img#pingouin_jaune{
    visibility:hidden;
}

table{
    position:absolute;
    left:380px;
    top:450px;
}

td{
    width:40px;
    height:100px;
    background-color:black;
    border-width:1px;
    border-style:inset;
}
```

Créez un bouton "Droite" qui appelle la fonction `pingouinD()` qui déplace le pingouin de 10 [px] vers la droite.

Remarque : Afin de simplifier le code, on introduit ici une variable `x` qui gère la position horizontale du pingouin et qui est initialisée à 100 px, comme la propriété `left` de l'image en

css. Evidemment, cette manière de procéder n'est pas idéale, puisqu'il faut modifier une valeur à deux endroits du code si l'on souhaite changer la position initiale du pingouin. Pour écrire le code correctement, il faudrait déclarer `x` (sans forcément l'initialiser) et introduire une fonction `initialiserParamètres()`, appelée depuis le `<body>` lors du chargement de la page, qui contienne l'instruction suivante¹ :

```
x = document.getElementById("pingouin").style.left
```

Tout en étant conscient des imperfections de ce choix, on procèdera de même avec les autres variables dans le cadre de cet exercice, ceci dans un souci évident de simplicité.

- D.2) Créez de même les boutons "Gauche", "Bas" et "Haut" et déclarez leurs fonctions respectives (nommées `pingouinG()`, `pingouinB()` et `pingouinH()` qui permettent de déplacer le pingouin de 10 [px] vers la gauche, le bas et le haut respectivement).

Conservez une copie de la solution D.2) qui constituera le fichier de départ de la série d'exercices E.

- D.3) Créez un bouton "+" (resp. "-") appelant une fonction `pingouinPlus()` (resp. `pingouinMoins()`) qui agrandit (resp. diminue) la taille (*width et height*) du pingouin de 10 %.
- D.4) Appelez la fonction `pingouinD()` par un *onclick* directement depuis la balise d'image du pingouin.
- D.5) Supprimez l'appel ajouté en D.4). Réduisez votre code en faisant en sorte que les boutons "Droite" et "Gauche" appellent la même fonction `pingouinX(...)`, les boutons "Bas" et "Haut" la même fonction `pingouinY(...)` et les boutons "+" et "-" la même fonction `pingouinZoom(...)`,
- D.6) Créez un bouton "Changer de pingouin" qui appelle la fonction `changerPingouin()` permettant de remplacer le pingouin initial par le pingouin rouge et vice-versa en modifiant l'attribut *src* de la balise ``.
- D.7) Faites apparaître l'image `pingouin_rouge_grand.png` avec les mêmes attributs qu'en D.1. Modifiez la fonction `changerPingouin()` en utilisant l'attribut de style *visibility*

¹ Concrètement, cette écriture ne fonctionnera que si le style est écrit en interne directement à l'intérieur de la balise 'pingouin'. Si le style est écrit dans un fichier css extérieur, l'expression devient légèrement plus compliquée... :

```
x = parseInt(getComputedStyle(document.getElementById("pingouin")).getPropertyValue("left"))
```

(valeurs : *"visible"*, *"hidden"*) de la balise ** (et non plus l'attribut *src*). Les fonctions *pingouinX()*, *pingouinY()* et *pingouinZoom()* déplacent/modifient les deux pingouins à la fois. Le résultat final est identique à celui de D.6))

D.8) Idem qu'en D.7) mais en ne déplaçant/modifiant que le pingouin visible à ce moment-là.

E. Utilisation de setTimeout() et setInterval()

Résoudre cet exercice dans un fichier qui soit une copie de la solution obtenue en D.2).

- E.1) Modifiez l'appel à la fonction pinguinD() depuis le bouton "Droite" pour que la fonction soit appelée périodiquement toutes les 100 [ms].
- E.2) Supprimez tous les boutons précédents et créez un bouton "Voyage aléatoire du pinguin" qui appelle périodiquement une fonction choixDirection(); cette fonction appelle aléatoirement la fonction pinguinD(), pinguinG(), pinguinB() ou pinguinH() toutes les 100 [ms].
- E.3) On souhaite pouvoir stopper le pinguin, c-à-d arrêter l'animation, donc annuler l'appel récurrent à la fonction choixDirection(). Pour réaliser cela, il faut:
- > déclarer une variable globale `t` qui va devenir un objet gérant l'animation
 - > déclarer une fonction nommée `voyageP()`, appelée depuis le bouton "Voyage aléatoire du pinguin", qui contient l'instruction suivante :
`t = setInterval(choixDirection,100);`
 - > déclarer une fonction nommée `stopP()`, appelée depuis le bouton "Arrêt du pinguin", qui contient l'instruction suivante : `clearInterval(t);`
- E.4) Faites apparaître les images pinguin_rouge_grand.png, pinguin_vert_grand.png et pinguin_jaune_grand.png. Modifiez votre code en déclarant les fonctions adéquates pour que le pinguin arrêté clignote en rouge-vert-jaune (changement de couleur toutes les 0.5[s]) lorsqu'il est arrêté. Lorsque le pinguin reprend son voyage aléatoire, il retrouve sa couleur initiale.

F. Utilisation de l'objet "event"

Rappel : Les exercices de cette série contiennent un lien avec la feuille de style externe styles_DEF.css

- F.1) Affichez dans une **fenêtre « alert »** les coordonnées de la souris (**variables globales sourisX, sourisY**) lors d'un clic dans la page ; une manière de réaliser cela est d'appeler via l'événement *onclick* de la *balise* `<html>` la **fonction `afficherCoord(event)`** qui contient un appel à la fonction utilitaire suivante :

```
var sourisX;  
var sourisY;  
function bouge(coord) {  
    sourisX= coord.clientX;  
    sourisY= coord.clientY;  
}
```

- F.2) Faites apparaître **l'image `pingouin_grand.png`**:

```
<img src='images/pingouin_grand.png' id='pingouin'>
```

Cliquez au milieu du pingouin et relevez les coordonnées de la souris (en utilisant la fonction écrite en F.1)).

- F.3) Effacez la **fonction `afficherCoord()`** et son appel. En tenant compte du décalage entre la position initiale du pingouin et les coordonnées de la souris (valeurs récoltées en F.2)), écrivez le code de la **fonction `adapterCoord()`** qui introduit dans les **variables globales pingouinX et pingouinY** les valeurs des positions *left* et *top* du pingouin quand la souris est placée au milieu du pingouin. Ecrivez le code de la **fonction `bougerPingouin(souris)`** (appelée depuis l'événement *onmousemove* de l'`` du pingouin avec l'objet *event* en argument effectif) qui permet de déplacer le pingouin avec la souris tant que le curseur de la souris se trouve sur l'image du pingouin.

- F.4) Modifiez votre code pour qu'une fonction `controlePingouin(souris)` soit appelée depuis l'événement *onclick* de l'`` du pingouin avec l'objet *event* en argument effectif. Cette fonction teste si le milieu du pingouin se trouve dans une zone carrée centrée autour des coordonnées $x=400\pm 20$ [px], $y=500\pm 50$ [px]. Si c'est le cas, une fenêtre « *alert* » affichant "Bravo ! Votre pingouin a réussi son expédition!" apparaît.

Si ce n'est pas le cas, la fenêtre « *alert* » affiche : "Votre pingouin n'a pas encore atteint le but de son expédition. Essayez encore, vous pouvez y arriver."

Pour visualiser la zone, entourez-la d'une cellule de tableau vide.

G. Éléments actifs (*input*)

G.1) Dans la partie `<body>`, introduisez l'extrait de code suivant (qui permet d'afficher sept boutons radio associés chacun au nom d'une montagne):

```
<input type="radio" name="radio_noms" value="l'Eiger">l'Eiger <br>
<input type="radio" name="radio_noms" value="la Dent Blanche">la Dent Blanche <br>
<input type="radio" name="radio_noms" value="le Chasseral">le Chasseral <br>
<input type="radio" name="radio_noms" value="le Cochet">le Cochet <br>
<input type="radio" name="radio_noms" value="le Zinalrothorn">le Zinalrothorn <br>
<input type="radio" name="radio_noms" value="la Dent d'Hérens">la Dent d'Hérens <br>
<input type="radio" name="radio_noms" value="le Chasseron">le Chasseron <br>
```

Créez un bouton « Contrôler le nom de la montagne (boutons radio) » qui appelle une fonction `controleNom_radio()`, ainsi qu'une `<div>` initialement vide d'`id= "nom_radio"`.

Remarque : Afin de simplifier la lecture du code, on introduit ici le code html directement dans la partie `<body>`, ce qui nous oblige à introduire des répétitions regrettables : les valeurs de attributs `value` de chaque bouton radio et le texte qui suit chaque balise est le même ! Pour éviter ces répétitions, il faudrait introduire le code de manière dynamique (via javascript) en allant chercher les noms des montagnes dans un tableau pour les introduire aux endroits adéquats. D'autres occasions dans le cours (en particulier dans les jeux du verger et du pendu) vous permettront de vous exercer à introduire le code de manière dynamique.

G.2) La fonction `controleNom_radio()` passe en revue tous les boutons et affiche dans le `innerHTML` de la `<div id= "nom_radio">` le nom de la montagne sélectionnée : "La montagne sélectionnée est `nom_de_la_montagne`" (ou "Aucune montagne n'est sélectionnée." si c'est le cas) . Pour déterminer si un bouton est sélectionné, on utilise l'instruction suivante (qui est vraie si le bouton radio n°i (i=0,1,2...) est sélectionné et fausse sinon):

```
document.getElementsByName(name)[i].checked;
```

G.3) Modifiez le programme pour que la fonction `controleNom_radio()` soit appelée chaque fois qu'un bouton radio est sélectionné.

G.4) Ajoutez un nouveau set de balises `<input>` qui soient de type "checkbox" :

```
<input type="checkbox" name="checkBox_noms" value=" nom de la montagne n°i "> nom de la montagne n°i <br />
```

Associez une checkbox à chaque montagne et testez les cases cochées avec la fonction `controleNom_checkbox()` appelée depuis le bouton "Contrôler le nom de la montagne (checkbox)".

Attention, il est possible de cocher plusieurs checkboxes en même temps ! Il faut donc différencier les cas suivants :

- 1) "Aucune montagne n'est sélectionnée."
- 2) "La montagne sélectionnée est *nom_de_la_montagne*."
- 3) "Les montagnes sélectionnées sont *nom_de_la_montagne1* et *nom_de_la_montagne2*."
"Les montagnes sélectionnées sont *nom_de_la_montagne1*,
nom_de_la_montagne2 et *nom_de_la_montagne3*."
etc.

L'instruction ci-dessous est vraie si la checkbox n°1 est sélectionnée ; elle est fausse sinon.

```
document.getElementsByName(name)[i].checked;
```

G.5) Idem qu'en G.4), mais en ajoutant un menu déroulant:

```
<select id="select1" onclick="controleNom_select();">
  <option value="L'Eiger">L'Eiger</option>
  <option value="la Dent Blanche">la Dent Blanche</option>
  ...
</select>
```

L'instruction ci-dessous renvoie la valeur de l'option sélectionnée:

```
document.getElementById('select_noms').value;
```

H. Modification des attributs d'une cellule de tableau (<td>)

H.1) Créez un tableau 2x2 contenant des cellules initialement vides en introduisant l'extrait de code suivant dans la partie <body> :

```
<table>
  <tr>
    <td id="1_1">&nbsp;</td>
    <td id="1_2">&nbsp;</td>
  </tr>
  <tr>
    <td id="1_3">&nbsp;</td>
    <td id="1_4">&nbsp;</td>
  </tr>
</table>
```

et créez une feuille de style externe (nommée styles_HI.css) qui sera appelée dans les codes des exercices des séries H et I et qui contient les instructions suivantes :

```
td{
    height:180px;
    width:180px;
    text-align:center;
    background-color:white;
    border-width:1px;
    border-style:inset;
}

table{
    position:absolute;
    left:0px;
    top:0px;
}

div.bouton{
    position:absolute;
    left:0px;
    top:400px;
    height:110px;
    width:800px
}
```


Conservez une copie de la solution H.1) qui constituera le fichier de départ de la série d'exercices I.

H.2) Créez un bouton "Changer la couleur" placé dans une <div> ayant l'attribut class="bouton":

Ce bouton colore la cellule d'id= "1_3" en gris en appelant une fonction changerCouleur() qui contient l'instruction suivante :

```
document.getElementById("1_3").style.backgroundColor="#BEBEBE";
```

H.3) Modifiez la fonction changerCouleur() pour qu'un deuxième clic sur le bouton colore la cellule en blanc, qu'un troisième clic la colore en gris, etc.

H.4) Appelez la fonction changerCouleur() directement depuis la balise de la cellule d'id= "1_3" pour qu'elle change de couleur lorsque l'on clique dessus.

H.5) Supprimez l'appel de la fonction depuis le bouton, mais conservez-le. Supprimez la possibilité de passer du blanc au gris au blanc etc. Modifiez votre programme pour qu'un clic sur les autres cellules du tableau les colore aussi en gris (en appelant chaque fois la même fonction changerCouleur(...)).

H.6) Créez un bouton "Afficher les images" qui appelle une fonction afficheImages(); cette fonction affiche les quatre images de la Dent d'Hérens (5_1.jpg, 5_2.jpg, 5_3.jpg et 5_4.jpg) dans les quatre cellules du tableau dans le bon ordre (image 5_1.jpg dans la case d'id= "1_1", etc.)

H.7) Créez une fonction choixNo() qui détermine aléatoirement un numéro entre 1 et 7 que l'on attribuera à une variable globale nommée no_montagne.

Créez une fonction choixOrdre() qui choisit aléatoirement un ordre d'apparition des images parmi six mélanges possibles ; pour y parvenir, déterminez aléatoirement un nombre entre 1 et 6 puis, à l'aide d'un « switch », attribuez à une variable globale ordre=new Array() l'une des valeurs suivantes :

```
ordre=[2,3,1,4]
ordre=[2,4,3,1]
ordre=[3,1,2,4]
ordre=[3,1,4,2]
ordre=[4,1,3,2]
ordre=[4,3,2,1]
```

H.8) Créez une fonction `init()` qui est appelée depuis le bouton "Afficher les images". Cette fonction appelle les fonctions `choixNo()`, `choixOrdre()` et `afficheImages()` ; modifiez cette dernière fonction de manière à ce qu'elle affiche dans le tableau les images de la montagne sélectionnée par `choixNo()` dans l'ordre sélectionné par `choixOrdre()`.

I. Utilisation de « this »

Résoudre cet exercice dans un fichier qui soit une copie de la solution obtenue en H.1).

I.1) Créez une fonction `memoriser(id)` qui est appelée par l'événement `onclick` depuis chaque cellule du tableau. Cette fonction vient inscrire dans la cellule sélectionnée du tableau la valeur de l'id de cette cellule en format titre `<h1>`.

I.2) Pour appeler la fonction `mémoriser(id)` depuis une cellule, tester l'instruction suivante :

```
<td id="1_1" onclick="memoriser(this.id);">&nbsp;</td>
```

Le symbole « this » est équivalent, dans cet exemple, à « `document.getElementById("1_1")` ». Introduisez cette instruction dans les quatre cellules et observez le résultat.

I.3) Remplacez l'appel à la fonction par l'instruction ci-dessous et modifiez la fonction `mémoriser(...)` en conséquence.

```
<td id="1_1" onclick="memoriser(this);">&nbsp;</td>
```

I.4) Modifiez la fonction `memoriser(...)` pour qu'elle dépose dans une variable globale nommée `selection` le n° de la cellule (par exemple, `selection=3` pour la cellule d'id="1_3") et qu'elle affiche cette valeur dans la cellule.

J. Ecriture du code complet du jeu

Travaillez dans un fichier dont le code source initial est celui du fichier intitulé "modele_J.html" qui comprend la partie `<body>`, les fonctions utilitaires, les variables globales avec leur initialisation, les noms des fonctions à compléter dans la partie `<script>` et un lien à la feuille de style externe styles_J.css (que vous pouvez utiliser telle quelle).

Plusieurs des fonctions à compléter ainsi que de nombreuses variables ont déjà été utilisées dans les séries précédentes de cet exercice ; leur nom est conservé dans le fichier.

Les images utilisées sont celles des montagnes (déjà importées dans la série H.), une image vide pour l'initialisation des tableaux et celles des petits pingouins gris, rouge, vert et jaune (pingouin.png, pingouin_rouge.png, pingouin_vert.png et pingouin_jaune.png).

En vous aidant des extraits de code déjà rédigés dans les séries A.-I. et en étudiant les diverses fonctionnalités du jeu (sans regarder le code source...), complétez toutes les fonctions pour obtenir un résultat du même type que le jeu proposé. Si vous le désirez, vous pouvez remplacer les images des montagnes (180[px] x 180[px] chacune) et celles des petits pingouins (25[px] x 36[px]) par d'autres images personnelles et modifier les textes de boutons radio et ceux affichés dans les `<div>` ou les fenêtres « *alert* » en conséquence.

Quelques remarques :


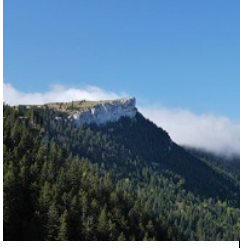
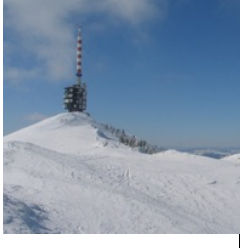
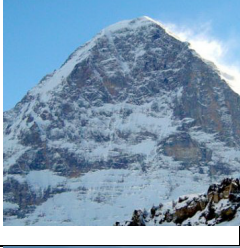
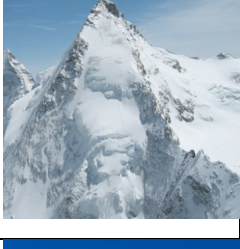

- Pour que qu'une image devienne semi-transparente lorsqu'elle est « cliquée », il faut introduire dans la balise `` l'attribut suivant:


```
onclick='this.style.opacity=0.4;this.filters.alpha.opacity=40'
```

- La variable globale *controle* est un tableau de six valeurs permettant de vérifier les six réponses de l'utilisateur (une pour chacune des quatre images, une pour le nom de la montagne et une pour son altitude). Chaque réponse vaut 0 si elle est fausse et 1 si elle est juste.

- La variable globale *puzzle* est un tableau de quatre valeurs permettant de vérifier l'emplacement des pièces du puzzle. Elle vaut [1,2,3,4] si les images ont été positionnées correctement dans le deuxième tableau.
- La variable globale *coordPics* comprend les coordonnées des sommets des montagnes lorsque les images des sommets sont correctement positionnées dans le deuxième tableau. Les deux premiers éléments correspondent aux coordonnées x et y (en [px]) de la montagne n°1, les deux valeurs suivantes aux coordonnées x et y (en [px]) de la montagne n°2 et ainsi de suite.
- La variable globale *depot* permet de rendre interactives (*depot=true*) ou non (*depot=false*) les cellules du 2^{ème} tableau (dans lequel sont déposées les images des montagnes). Elle permet de « figer » les images au moment où la vérification des images donne 6 réponses justes.
- La variable globale *pingouinEnMvt* ne devient *true* que lorsque toutes les réponses sont justes et que le pingouin peut se déplacer. Elle gère (avec un if...) l'action ou l'inaction de la fonction *bougerPingouin()* appelé depuis <body>. Lorsque le pingouin a atteint le sommet et/ou lorsque l'utilisateur recommence une partie, elle est à nouveau *false*.
- La variable globale *pingouinClignotant* permet d'empêcher l'utilisateur de faire apparaître un deuxième pingouin (lorsque le pingouin clignote et ne se déplace plus avec la souris) en cliquant sur le bouton « Vérifier les réponses ». Elle gère (avec un if...) l'action ou l'inaction de la fonction *controleReponses()*. Lorsque l'utilisateur recommence une partie, elle est à nouveau *false*.
- Les variables globales *tPingouins* et *compteurCouleur* permettent de gérer les changements de couleur du pingouin lorsqu'il se trouve au sommet de la montagne.

- Voici les réponses et les dimensions des images:

montagne n°	nom	altitude [m]	coord. xm [px]	coord. ym [px]	image
1	Le Cochet	1483	554	230	
2	Le Chasseron	1607	581	158	
3	Le Chasseral	1607.4	488	193	
4	L'Eiger	3970	599	61	
5	La Dent d'Hérens	4171	532	44	
6	Le Zinalrothorn	4221	613	54	

7	La Dent Blanche	4356	585	83	
---	--------------------	------	-----	----	---