Projet

Chasse au Trésor

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

Date	Révision
07/04/2022	Ajout de la partie héritage et attributs privés

Sommaire

A Objectifs

Cette activité doit permettre le développement d'un jeu utilisant les langages HTML/CSS, JavaScript et PHP.

B Présentation

Le jeu est très simple : un trésor est caché sur une île et il faut le trouver.

L'île sera représentée par une grille de 10 × 10 cases, et le trésor occupera une seule case, qui sera choisie au hasard en début de partie. L'île aura également des événements – positifs ou négatifs – qui pourront avantager ou désavantager le joueur.

Ce dernier devra trouver le trésor le plus rapidement possible et pourra enregistrer son score (date, nom, score) dans une base commune. Les résultats seront lisibles sur une page score.

Le projet sera créé en plusieurs étapes :

- Créer automatiquement une carte et des cases cliquables
- Placer un item dans une coordonnée de la carte
- Créer des items positifs ou négatifs (pourcentage de présence) et placement
    - On place le trésor en dernier pour éviter un recouvrement
- Créer un score (comptable, etc)
- Créer un formulaire d'enregistrement de score et récupération en PHP pour placement dans une base de données.

C Partie JavaScript

Un certain nombre de fonctions seront nécessaires en JavaScript, pour créer la table, pour recevoir les événements (clic sur les cases), pour interagir, etc.

JavaScript sera au centre de notre projet. Le code total ne sera pas donné mais des morceaux de codes plus complexes sont proposés ci-dessous.

C.1 Tableau à deux dimensions

Avant d'afficher un tableau à l'écran, il faut créer un tableau dans la mémoire de l'ordinateur. Pour cela, il faut utiliser une variable qui contiendra un tableau en deux dimensions. Cependant, il n'est pas possible de déclarer et d'utiliser simplement un tableau à 2 dimensions finies.

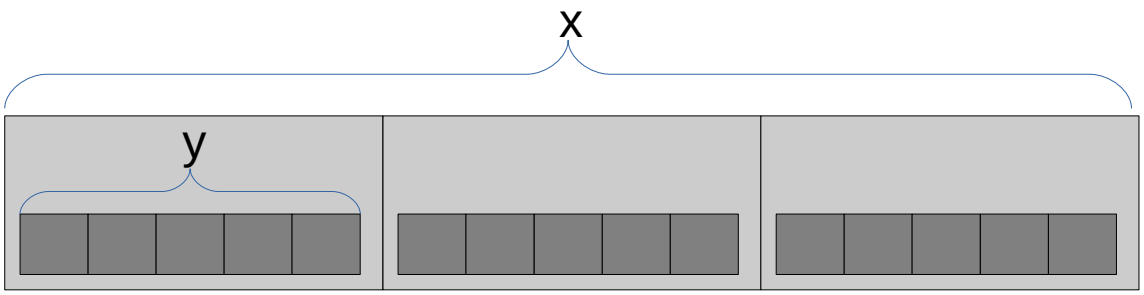
La fonction suivante permet de créer un tableau en 2 dimensions et de le renvoyer :

On créé un premier tableau

```
function Tableau2D(x, y) {
    var array2D = new Array(x);
    for (var i = 0; i < array2D.length; i++) {
        array2D[i] = new Array(y);
    }
    return array2D;
}
```

A l'intérieur de chaque case, on créé un nouveau tableau

Le tableau retourné ressemble à ceci : il s'agit d'un tableau à une dimension qui contient un tableau à une dimension.



Pour utiliser la fonction, if suffit de placer le résultat dans une variable :

```
let monTableau = new Tableau2D(10, 10);
```

L'intérêt de créer un tableau 2D, est de pouvoir cacher d'autres surprises que le trésor dans les cases : un bonus, une potion ou au contraire, du poison ou un malus.

Un autre avantage est de pouvoir y placer une valeur si la case à déjà été visitée.

L'écriture dans le tableau est simple :

```
monTableau[5][2] = "Tresor" ;
```

ou encore

```
let coordonneeX = 5 ;
let coordonneeY = 2 ;
//...
monTableau[coordonneeX][coordonneeY] = "Tresor" ;
```

C.2 Nombre aléatoire

Pour créer un nombre aléatoire, il faut utiliser la fonction **Math.random()**.

Cette fonction renvoie un nombre décimal entre 0 et 0,999999. Pour simuler un dé (qui va de 1 à 6) il faudrait écrire :

```
let nombreFace = 6
let De6 = Math.floor(Math.random() * nombreFace + 1) ;
```

En effet, puisque la fonction Math.floor() arrondi à l'entier inférieur et que Math.random() ne sera jamais égal à 1, écrire Math.random() * 6 serait au maximum égal à 5,99999 qui serait arrondi à 5. à l'inverse, si Math.random() renvoie 0, la multiplication avec 6 reste toujours égale à 0. Il faut donc ajouter 1 pour avoir un résultat compris entre 1 et 6.

C.3 Dessiner un tableau en HTML

Le nombre important de cases à dessiner implique un script JavaScript qui va générer un bloc en HTML.

Pour rappel, voici comment JavaScript peut écrire un simple paragraphe en HTML dans une page existante :

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <div id="ici"></div>
</body>
</html>
```

Le code JavaScript sera :

```
let unTexte = "<p>Code <strong>HTML</strong> généré via JavaScript.</p>"
document.getElementById("ici").value = unTexte
```

L'objectif est de générer le code HTML grâce à Javascript, puis de l'écrire à l'emplacement que vous aurez prévu pour ce tableau.

Rappel : un tableau d'une ligne et deux colonnes en HTML c'est :

<table> <tr> <td></td> <td></td> </tr> </table>

Donc il faut deux boucles JavaScript : une pour écrire les lignes et une autre à l'intérieur de la première boucle, pour écrire les colonnes (<td></td>).

```
function drawTableau2D(x, y, emplacement) {
    let idString = ""
    let texte = "<table>"
    for (let xx=0; xx < x; xx++) {
        texte += `<tr>`
        for (let yy=0; yy < y; yy++) {
            idString = xx+"-"+yy
            texte += `<td id="${idString}"></td>`
        }
        texte += `</tr>`
    }
    texte += `</table>`
    document.getElementById(emplacement).innerHTML = texte
}
```

Cette fonction traitant du code HTML, elle ne renvoie pas de variable de retour, mais elle écrit directement le tableau de taille x par y, dans l'emplacement transmis en paramètre :

```
drawTableau2D(10,10,"ici")
```

Notez que le script écrit un ID pour chaque case, de la forme X-Y (par exemple 2-5 pour la deuxième ligne et la case à la 5^{e} colonne).

Ainsi, chaque case a une coordonnée unique.

C.4 Emplacement d'un élément (lors d'un clic)

Cette dernière fonction permet (dans le code HTML) de placer un appel à une fonction JavaScript en lui transmettant l'ID de l'élément cliqué.

```
<td onclick="choix(this.id);" id="5-2"></td>
```

La fonction JavaScript choix(variable) recevra l'ID de l'endroit où il se trouve, ici : 5-2

Cette ligne peut être générée par du code JavaScript : il suffit d'écrire dans une variable JavaScript l'ensemble du code HTML, puis d'utiliser l'écriture à un emplacement de JavaScript.

Ainsi, un exemple d'utilisation serait le suivant :

Mettre les coordonnées dans une chaine de caractères

Utiliser la fonction d'appel JavaScript

```
let idCase = String(y) + "-" + String(x);
corpsTable = corpsTable + '<td onclick="choix(this.id);" id="' + idCase +
'"></td>';
document.getElementById("emplacementTable").innerHTML = enteteTable +
corpsTable + finTable;
```

Écrire les coordonnées dans le code HTML

Modifier le document HTML avec nos variables

C.5 Changer un attribut

JavaScript peut modifier les attributs d'une balise, d'une classe, d'un ID. Voici comment changer la classe d'un élément :

```
caseTable = document.getElementById(idCellule);
caseTable.setAttribute('class', 'good');
```

C.6 Tableau mémoire et tableau graphique

Nous avons créé deux tableaux : un tableau contient les informations sur le trésor, l'autre tableau permet de cliquer sur les cases pour déterminer la position que demande le joueur, puis on peut y afficher une couleur de case en fonction des données dans le vrai tableau JavaScript.

Tableau JS

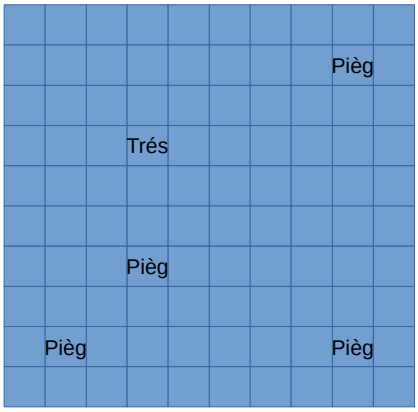


Tableau HTML

Clic sur 4-4
Lire tableau JS :
Si case[4][4] contient trésor c'est gagné !
Sinon si case [4][4] contient piège : enlever point
Sinon colorier "déjà fait"

D Partie HTML

Elle sera réduite à sa plus simple partie : le code standard et une balise ayant comme ID « emplacementTable » et appartenant à la classe « cadre » (nous utiliserons du code CSS sur cette classe).

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <title>L'ile au trésor</title>
    <script type="text/javascript" src="SI6-IleAuTresor.js"></script>
    <link rel="stylesheet" type="text/css" href="SI6-IleAuTresor.css">
</head>

<body>
    <H1>L'ile au trésor</H1>

    <p>Nombre de coup(s) : <span id="compte">0</span></p>
    <hr>
        <div id="emplacementTable" class="cadre">
            Tableau
        </div>
        <div id="emplacementCommentaires" class="cadre">
            Vous vous apprêtez à creuser avec votre perroquet sur votre
chapeau...<br>
        </div>
        <hr>
</body>
</html>
```

Permet à JavaScript de retrouver où dessiner la table

Permet à CSS dd'appliquer un style

Le code sera amélioré par la suite, pour permettre à l'utilisateur de savoir s'il est sur la bonne colonne ou la bonne ligne… mais nous pouvons ajouter une simple légende pour la couleur de la bonne ou mauvaise case :

```
    <table>
        <tr>
            <td class="bad"></td>
            <td>Mauvaise case</td>
            <td class="good"></td>
            <td>Bonne case</td>
        </tr>
    </table>
```

Notez l'utilisation d'une classe pour définir la couleur de la case !

Au passage, ce code rappelle le fonctionnement d'un tableau en HTML : Table, ligne puis colonne (ou disons plutôt, « cellules »).

E Partie CSS

Ce fichier va permettre de placer les éléments et rendre l'aspect visuel plus attractif :

E.1 Définir le tableau

Ici, l'idéal étant de ne modifier que le tableau généré par JavaScript, il est recommandé d'utiliser un ID :

Un petit commentaire...

l'id du tableau

```
/* format tableau */
#fondtable {
    background: url(carte.png);
    border-width: 1px;
    border-color: black;
    border-style: solid;
    border-collapse: collapse;
}
```

Image de fond pour le tableau

Limite l'espace entre les cellules

E.2 Définir les cellules

Cette section montre comment créer des cases carrées dans un tableau :

```
/* Largeur et hauteur des cases de la grille */
td {
    width:40px;
    height:40px;
    border-width: 1px;
    border-color: black;
    border-style: solid;
}
```

E.3 Définir un bloc

Pour le on favorise un bloc dont on fixe la largeur et qui accepte d'autres blocs à côté de lui.

```
/* Gestion des blocs de classe "cadre" */
.cadre {
    display: inline-block;
    width: 450px;
    /*margin: 1em;*/
    font-size: 18px;
}
```

Inline : les blocs peuvent être côte-à-côte
Block : les blocs seront dans la même hauteur

E.4 Utiliser les couleurs

L'idée, lorsque le joueur clique sur une case, est de changer la couleur de celle-ci. Comme JavaScript peut modifier tous les attributs du DOM, il pourra changer la classe d'une case. Il faut donc définir les deux classes de jeu : celle d'une mauvaise case, en rouge ; celle de la case gagnante, en cyan ;

Nous allons cependant utiliser le système rgba() pour gérer l'opacité de la case.

```
/* Couleurs des cases, avec opacité de 0.9 */
.bad {
    background:rgba(200,0,0,0.9);
}
.good {
    background:cyan;
}
```

Opacité : 0,9

Bleu : 0

Vert : 0

Rouge : 200

Nous aurions pu utiliser le système hexadécimal habituel : #C80000E5

F PHP

Lorsque la partie est terminée, il faut envoyer le score via un formulaire : le plus simple sera de cacher le formulaire jusqu'au moment où la partie se termine. Un seul champ sera visible (demande de nom) et l'autre sera invisible et contiendra le score.

F.1 Partie HTML

Le code est le suivant :

```
<form id="appelScore" method="post" action="page.php" style="display:none;">
  <input type="text" id="nom" name="nom" required/>
  <input type="hidden" id="score" name="score" required/>
</form>
```

F.2 Partie JavaScript

Dans cette partie de code, on crée une fonction qui valide le formulaire automatiquement. La saisie du joueur se fera avec une boite de dialogue prompt() :

```
let nomGagnant = window.prompt("Indiquez votre nom pour le score","anonyme")
document.getElementById('nom').value = nomGagnant
document.getElementById('score').value = nobreCoup   // variable de comptage des coups joués
document.getElementById('appelScore').submit()
```

F.3 Partie PHP

Dans le fichier PHP sur un serveur il suffira de récupérer les valeurs et d'effectuer une requête SQL d'insertion dans la base.

```
<?php
$name = $_POST['nom'];
$scoreGame = $_POST['score'];

// base en ligne, accessible depuis Internet
$server = "db4free.net"
$dbname = "db_bts_sio_test";
$username="adm_sio_berges";
$password = "AristideBerges38!";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "INSERT INTO tableScore (nom, score)
  VALUES ($name, $scoreGame)";
  // use exec() because no results are returned
  $conn->exec($sql);
  echo "Enregistrement du score avec succès";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

G Structure du code

Désormais, nous avons tous les éléments de codes pour créer notre jeu. Il faut cependant définir le fonctionnement.

Voici les fonctions JavaScript, à vous de les remplir :

```
// Tableau2D renvoit un objet tableau en 2D
function Tableau2D(x, y) {
}

// choix() récupère l'ID de la case cliquée et traite le résultat
function choix(that) {
}

// AfficherCompteur() permet d'afficher la variable compteur à l'emplacement
voulu.
//                    affichera plus tard des commentaires.
function AfficherCompteur() {
}

// onload vérifie que la page soit complètement chargée pour lancer la fonction
window.onload = function() { initTab(); }

//
------------------------------------------------------------------------------
// initTab() affiche le tableau et choisit les coordonnées du trésor
function initTab() {
}
```

H Résultat

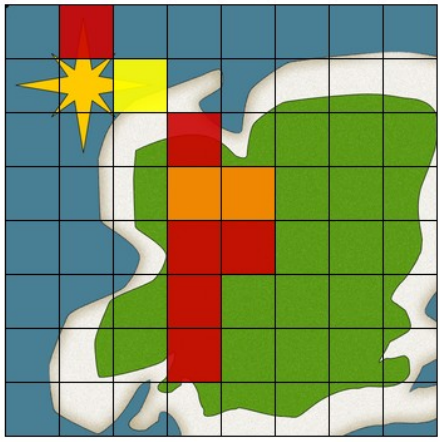
Voici un exemple simple de chasse au trésor : vous pouvez faire bien mieux

L'ile au trésor

Pour jouer, il suffit de cliquer sur une des cases du tableau. La couleur indique le résultat suivant :

	Mauvaise case		Bonne case		Bonne colonne		Bonne ligne

Nombre de coup(s) : 9



Vous vous aprêtez à creuser avec votre perroquet sur votre chapeau...
Coco : Hey Capt'ain, bonne chance !
Coco : Hey Capt'ain, ça creuse ?
Coco : Hey Capt'ain, t'es sûr de ce que tu fais ?
Coco : Hey Capt'ain, t'es pas un vrai boucanier, n'est-ce pas ?

I Partie Programmation Orientée Objet

I.1 Classes et usages

JavaScript permet également de rédiger des classes et utiliser des objets.

```
class Surprise {
    // Déclaration des attributs
    x = 0
    y = 0
    desc = ""

    // Déclaration des constructeurs
    constructor(desc, x, y) {
        this.description = desc
        this.x = x
        this.y = y
    }
    constructor(desc) {
        // c'est le mode automatique, on décide de tout
        this.x = Hasard(0, size);
        this.y = Hasard(0, size);
        this.desc = desc;
    }

    // Déclaration des méthodes
    setLog(texte) {
        this.descLog = texte
    }
    isGood() {
        return fin & 0x01
    }
    isEnd() {
        return fin>>1 & 0x1 // décale les bits vers la droite
    }
    setBadEvent() {
        let bad
    }
    Melange(Good, End) {
        // cette fonction place le bit Good et décale puis place le bit End
        return Good + End<<1
    }
```

Pour utiliser cette classe, il suffit de l'instancier dans une variable :

```
// création de la liste (collection)
let listeSurprise = []

// création d'un objet trésor à remplir manuellement
// méthode 1 : créer l'objet puis remplir les éléments
let tresor = new Surprise()
tresor.X = Surprise.hasard(10)
tresor.Y = Surprise.hasard(10)
```

I.2 Héritage et attributs privés

JavaScript permet également la gestion de l'héritage :

```
// création de la classe mère Vehicule
class Vehicule {
    nom = ""
    km = 0

    constructor(nom, km) {
        this.nom = nom      // attribut public
        this.#km = km       // attribut privé
    }

    getKm() { return this.#km}
}

// création de la classe fille Voiture
class Voiture extends Vehicule {
    carburant=""

    constructor(nom, km, carburant) {
        super(nom, km)      // on appelle le constructeur de la classe mère
        this.carburant = carburant
    }
}


// création d'un objet
let F1 = new Voiture("Formule 1", 5430, "Essence")
console.log("Fonctionne : ",F1.nom, F1.getKm(), F1.carburant)
console.log("Indéfini   : ",F1.km)
```

Il est donc recommandé d'utiliser ce concept pour programmer des événements dans notre jeu.