Leçon 8.2 – HTML5 – Méthodologie

Jeu des balles - Développement pas-à-pas

J.-Y. Plantec

Ce document a pour objectif de vous aider à réaliser le développement du jeu des balles. Le code complet du jeu n'est pas donné : il vous faudra écrire et tester plusieurs parties en vous aidant du travail de conception réalisé précédemment et en suivant scrupuleusement les étapes proposées !

De loin en loin, vous trouverez des copies d'écran qui vous permettront de vérifier votre progression.

1 – Code HTML

On commence par le fichier HTML que l'on construit au sens du cahier des charges fonctionnel et de la partie consacrée à la liste des écrans et à leur contenu potentiel. Cette partie de code est donnée.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8"/>
<title>Les balles</title>
<link rel="stylesheet" media="screen" href="./css/styles.css" type="text/css"/>
<script src="./js/jquery.js"></script>
<script src="./js/code.js"></script>
</head>
<body>

<div id="accueil">
    <div id="titre"></div>
    <div id="image"></div>
    <div id="texte"></div>
    <div id="boutonJeu"></div>
</div>

<div id="jeu">
    <div id="consigne"></div>
    <div id="animation"></div>
    <div id="boutonQuitter"></div>
</div>

<div id="bilan">
    <div id="recap"></div>
    <div id="boutonAccueil"></div>
    <div id="boutonRejouer"></div>
</div>

<footer></footer>

</body>
</html>
```

2 – Code CSS

Concernant le code CSS, le fichier s'appelle styles.js. Pour commencer à écrire ce fichier CSS, on s'appuie sur les éléments généraux des spécifications techniques détaillées. On définit ainsi :

- le fond de page gris clair #ccc
- le style du pied de page : Arial, noir, 12 pts
- le texte de base des écrans : Arial, noir, 12 pts
- pour l'écran d'accueil
    - le titre : Arial, rouge, 16 pts
    - le texte expliquant les règles :

Vous mettrez à jour ce fichier CSS lorsque vous positionnerez les boutons et définirez le canvas.

3 – Code JavaScript

Concernant le code JavaScript

- ici, on a choisi de mettre en oeuvre jQuery, mais ce n'est pas une obligation ;
- le fichier js s'appelle code.js

Structure générale

On commence à écrire la structure générale du code :

```
$(function() {
    init();
});

function init(){

    // STRUCTURE

    // DONNEES

    // VARIABLES

    // GESTIONNAIRES

    // REGLES

    // LANCEMENT

}
```

Pour ceux qui n'auraient pas choisi jQuery, il faut bien entendu remplacer $(function(){...}) ; par document.ready...

À ce stade, on peut tester ce travail pour voir si aucune erreur n'est levée.

Puis, on construit ce code étape par étape comme ci-dessous et en suivant les spécifications techniques détaillées.

Étape 1 : structure

Nous créons dynamiquement les éléments de chaque écran.
Dans l'écran d'accueil, on définit :

- le texte du titre (une chaîne de caractères),
- le code de l'image du jeu, même si nous n'avons évidemment pas encore d'image du jeu,
- le texte expliquant les règles,
- le code du bouton #boutonJeu.

Dans l'écran de jeu, on définit :

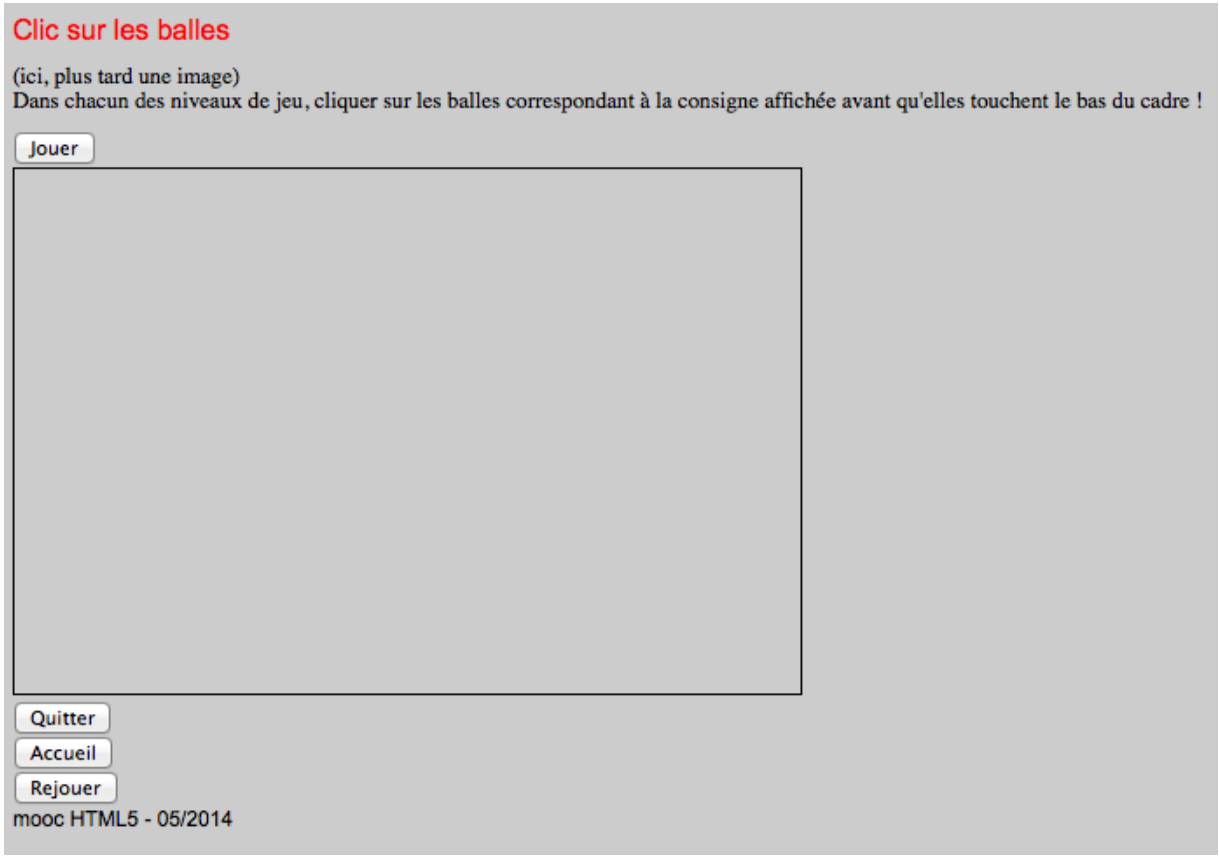
- le texte pour la consigne,
- le code du canvas,
- le code du bouton #boutonQuitter.

Dans l'écran de bilan :

- on laisse vide pour l'instant l'élément où s'affichera le récapitulatif de ce qu'à fait le joueur ;
- on définit le code des boutons #boutonAccueil et #boutonRejeu.

On définit le texte du footer.

On teste. Tous les éléments sont visibles.



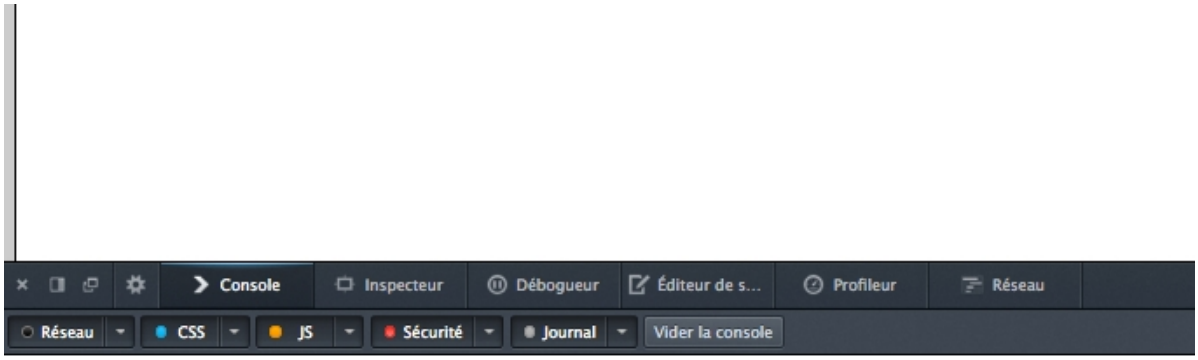
Étape 2 : données

On définit ici les 4 tableaux de données :

- listeNiveaux,
- listeBalles,
- listeCouleurs,
- listeTailles.

On n'oublie pas toutes les colonnes qui ont été ajoutées au moment des spécifications détaillées.
Un conseil : commencez avec un jeu de données simple, par exemple deux niveaux et deux balles à chaque niveau !

On teste avec la console ouverte.



Étape 3 : variables

On définit et on initialise les variables :

- tempsJeu,
- niveauCourant,
- ecranCourant.

On teste avec la console ouverte.

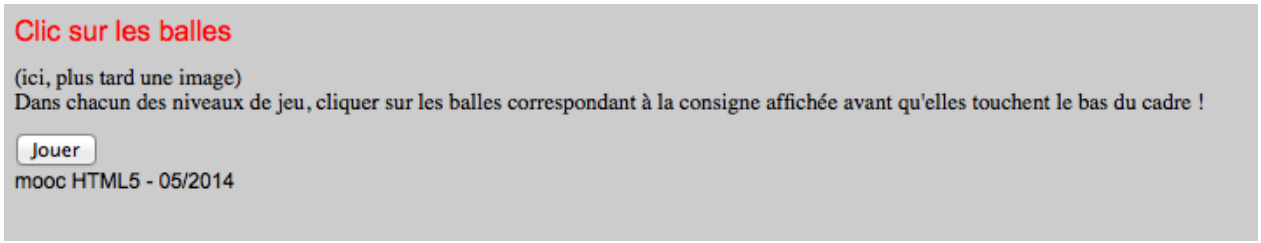
Étape 4 : lancement

Au chargement du jeu

L'écran d'accueil s'affiche au chargement (et masque les autres, bien entendu). Les instructions sont écrites dans la partie LANCEMENT de la fonction init(). On peut créer une fonction afficheAccueil() qui

- met à jour la variable ecranCourant (ecranCourant="accueil";),
- affiche l'écran #accueil,
- masque les autres écrans.

On teste. Seul #accueil est visible.



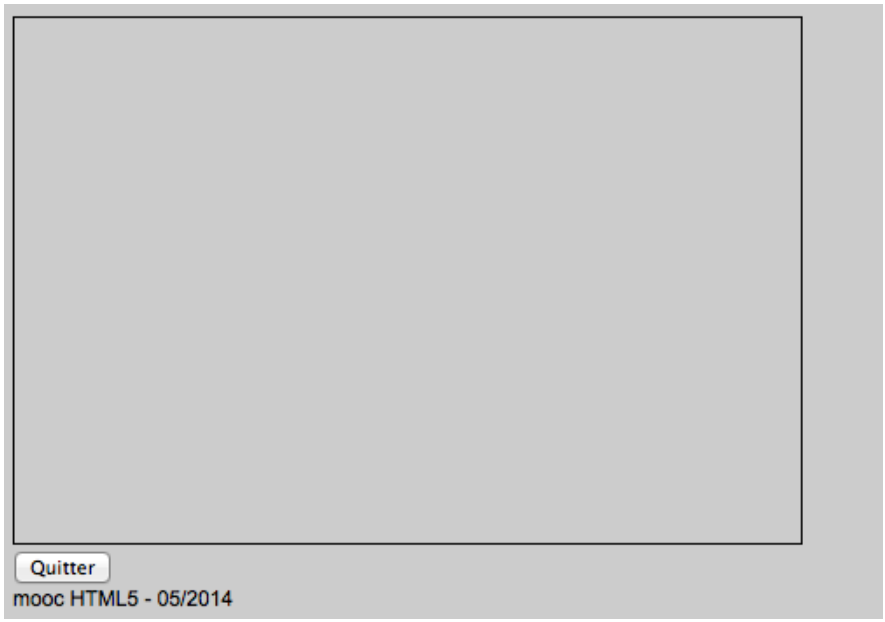
Étape 5 : gestionnaires et règles

Écran d'accueil

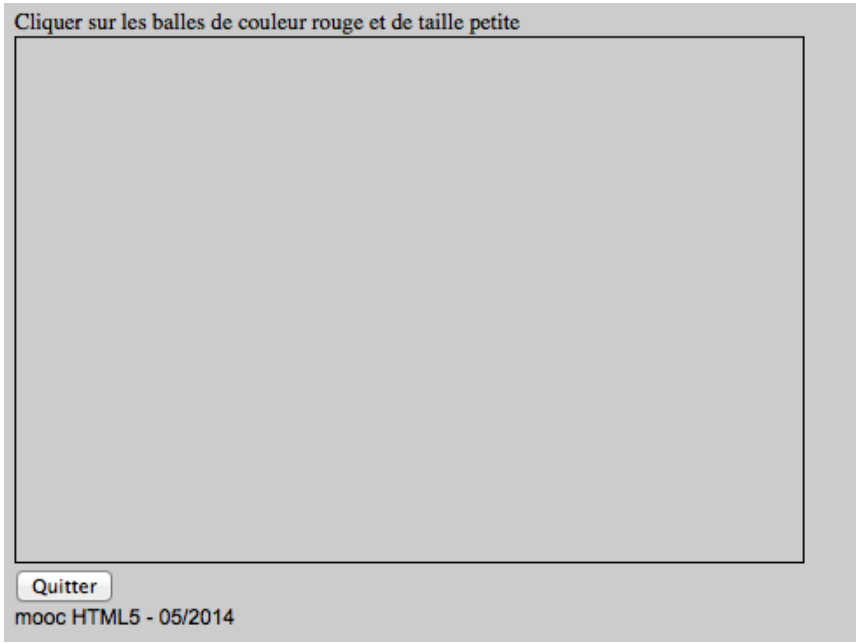
On s'occupe des règles et fonctionnalités de l'écran d'accueil.
Seule fonctionnalité de cet écran : si on clique sur le bouton #boutonJeu alors on passe à l'écran #jeu. On installe un gestionnaire sur le bouton #boutonJeu associé à une fonction afficheJeu() qui

- met à jour la variable ecranCourant,
- affiche l'écran #jeu,
- masque les autres écrans.

On teste :



Écran de jeu
Au départ tous les éléments potentiels s'affichent. La consigne du premier niveau s'affiche. On a vu que l'on construit cette consigne à partir des deux identifiants (un pour la couleur, un pour la taille) donnés dans le tableau des niveaux. Il va être utile de créer une fonction pour l'affichage de cette consigne. On l'appelle afficheConsigne().



On s'occupe des règles de l'écran de jeu.

Première règle très générale : sans condition, les balles du premier niveau descendent.

Selon les spécifications détaillées, il suffit, maintenant que le code HTML du canvas a été créé, de

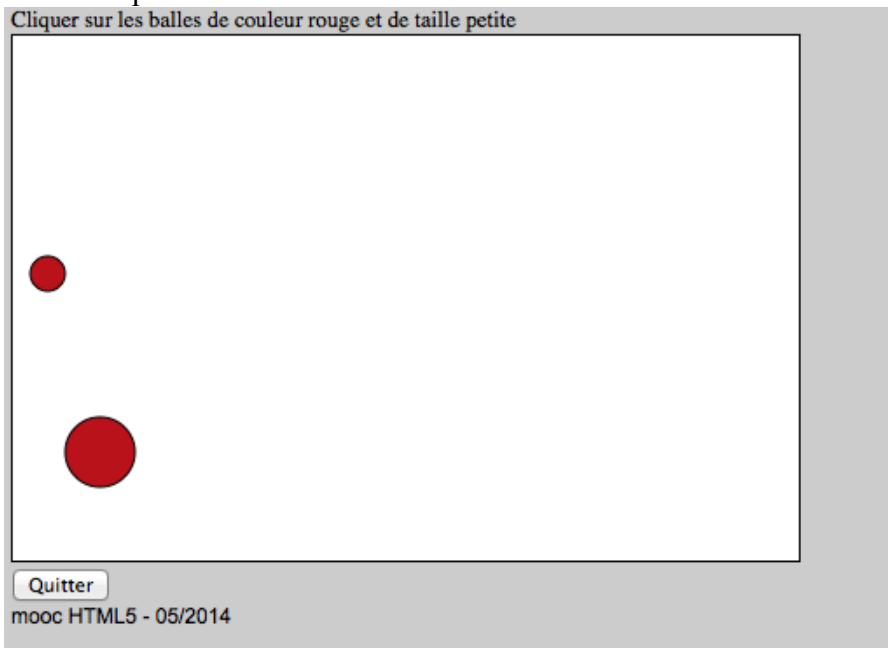
- lancer un setInterval() dans la partie REGLES de la fonction init() ; ce setInterval() exécutera une fonction regles() ;
- définir la fonction regles() qui lancera (à condition que l'écran courant soit l'écran de jeu) une fonction animer().

```
function regles(){
    if (ecranCourant == "jeu"){
        animer();
    }
}
```

Que fait cette fonction animer() ? Elle

- incrémente le temps de jeu,
- efface le canvas,
- boucle sur les balles du niveau courant et dessine ces balles en lançant la fonction dessineBalle().

On teste après avoir défini le code CSS du canvas.



Quelques conseils :

- Soyez vigilants dans dessineBalle() à bien récupérer les bonnes informations pour dessiner la balle : couleur, taille, vitesse, position selon x.
- Soyez vigilants sur la portée des variables utilisées.
- N'hésitez pas à utiliser le débogueur !
- Pour tester, on peut aussi mettre en commentaire le setInterval et lancer une seule fois la fonction Animer() pour vérifier les instructions.

- Vérifiez pour ce premier niveau que les balles tombent de la bonne position x, avec la bonne couleur, la bonne taille et la bonne vitesse.

Règle suivante :

- si le temps de jeu dépasse un certain nombre de secondes alors on passe à l'écran #bilan (et on masque les autres écrans). On peut créer ici une fonction afficheBilan() qui
    - met à jour la variable ecranCourant,
    - affiche l'écran #bilan,
    - masque les autres écrans.

Selon le cahier de spécifications, ce test est fait tout au début de la fonction animer().
On teste. Pour tester cette règle de temps limite, on peut ajuster la valeur maxi de temps à 10 ou 15 secondes (en fonction de la fréquence du setInterval et de la vitesse choisie pour les balles).

Règles suivantes :

- si les balles correspondant à un niveau donné sont toutes passées ou ont toutes disparu parce qu'elles ont été cliquées, alors on passe au niveau suivant.
- si au moment de changer de niveau c'est le dernier niveau qui vient de passer alors on passe à l'écran #bilan (et on masque les autres écrans).

Selon le cahier de spécifications, ce test est fait dans la fonction animer() et l'algorithme est donné.

On se rappelle que l'on doit garder en mémoire les coordonnées selon y de chaque balle. Ceci est fait dans dessineBalle() en mettant à jour la colonne y dans le tableau listeBalles pour la balle courante.

Je vous propose de tester ça.

- Vérifiez que l'on change de niveau !
- Créez si nécessaire d'autres niveaux et d'autres balles !
- N'hésitez pas à donner une grande vitesse de descente aux balles pour accélérer le test !
- Attention à nouveau à la portée des variables !

Si arrivez juste qu'ici, bravo ! L'essentiel est fait. Ce point était le point dur du développement. La suite est plus facile.

Première fonctionnalité de cet écran de jeu :

- si on clique dans l'élément où s'affichent les balles alors cette balle disparaît (même si ce n'est pas une balle à faire disparaître au sens de la consigne).

Selon le cahier de spécifications, nous devons installer un gestionnaire de clic sur le canvas (ceci se fait dans init(), dans la partie GESTIONNAIRE) et nous adaptons la fonction clicCanvas(e) de la semaine 6.
Si le clic se produit bien sur une balle alors on met la colonne visible de cette balle à 0 ; ainsi dans

la fonction Animer() le test qui vérifie que conduit au dessin de la balle courante en vérifiant que la valeur visible de cette balle vaut 1.

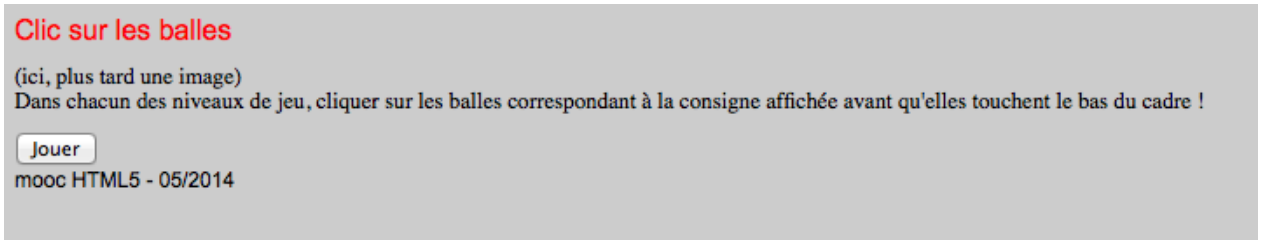
On teste.

Deuxième fonctionnalité : si on clique sur le bouton #boutonQuitter alors on revient à l'écran d'accueil. Pour mettre en oeuvre cette fonctionnalité, on installe un gestionnaire de clic sur le bouton #boutonQuitter. Ce gestionnaire affiche l'écran d'accueil (il lance la fonction afficheAccueil(), mais il ne faut pas oublier de réinitialiser

- le niveau courant,
- la position des balles selon y ainsi que leur visibilité.

Pour ces deux derniers points, on peut créer une fonction reinitialisation().

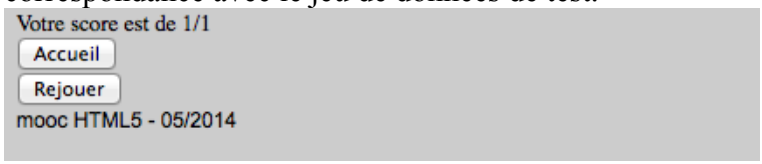
On teste.



Écran de bilan
Au départ tous les éléments potentiels s'affichent : le texte récapitulatif et les deux boutons.

Les spécifications détaillées proposent un l'algorithme qui permet de calculer le nombre de points obtenus ainsi que le nombre de points possible. Cet algorithme sera écrit dans la fonction afficheBilan() ;

On teste avec le débogueur. Conseil : testez de nombreuses fois cet aspect (les utilisateurs n'aiment pas que les scores soient mal calculés, n'est-ce pas ?) en mettant le résultat affiché en correspondance avec le jeu de données de test.



On s'occupe des fonctionnalités de l'écran de bilan :
- si on clique sur le bouton # boutonRejouer alors on repasse à l'écran #ecranJeu ;
- si on clique sur le bouton # boutonAccueil alors on repasse à l'écran #ecranAccueil.

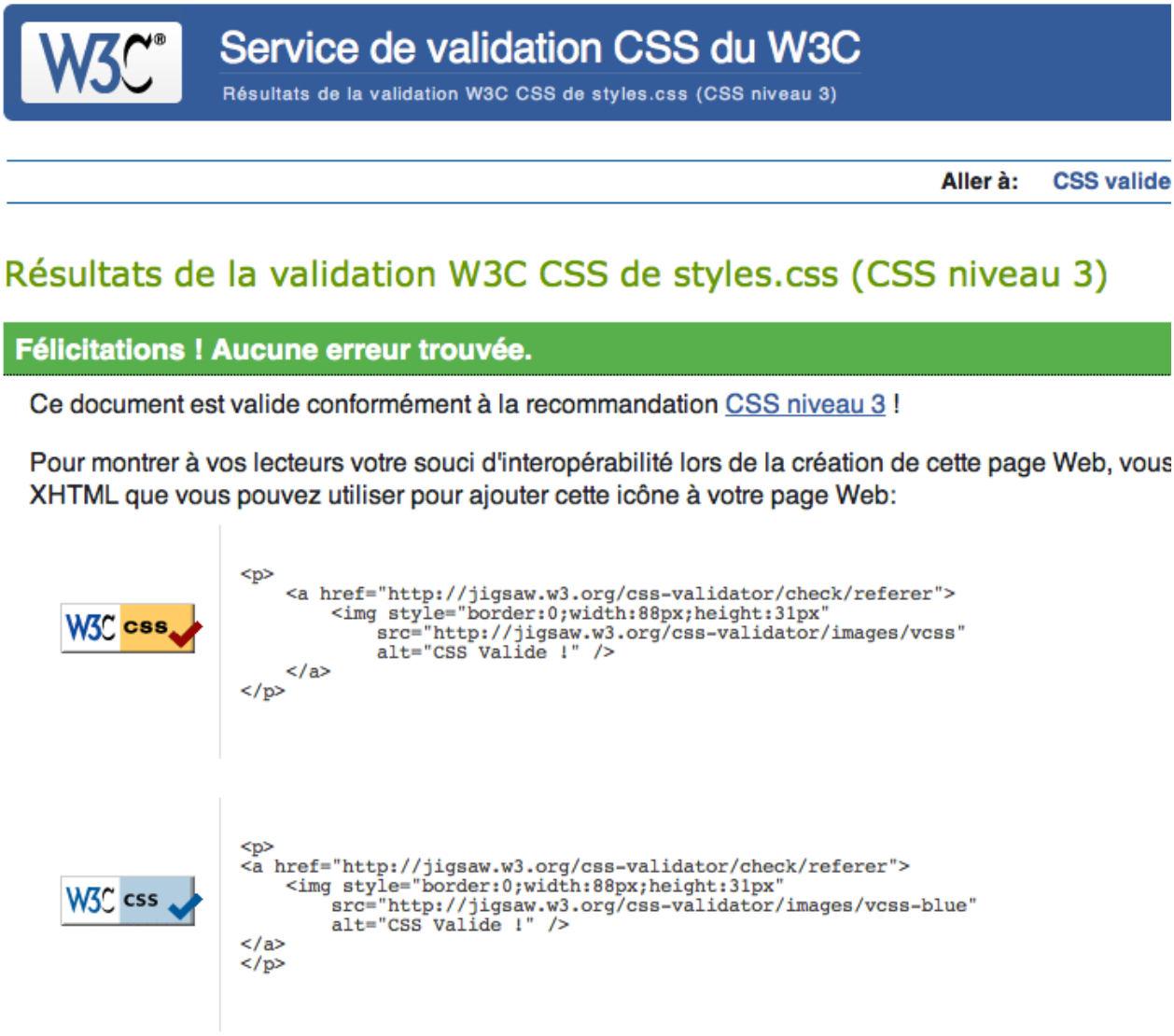
Ces fonctionnalités sont faciles à mettre en oeuvre :
- un gestionnaire sur le bouton #boutonRejouer,
- un autre sur le bouton #boutonAccueil.

Le code à exécuter est décrit dans les spécifications détaillées.

4 – Tests

Voilà. Vous êtes arrivés au bout ! À ce stade, pourquoi ne pas valider le code qui a été écrit ?

- HTML : http://validator.w3.org
- CSS : http://jigsaw.w3.org/css-validator/



Service de validation CSS du W3C

Résultats de la validation W3C CSS de styles.css (CSS niveau 3)

Aller à: CSS valide

Résultats de la validation W3C CSS de styles.css (CSS niveau 3)

Félicitations ! Aucune erreur trouvée.

Ce document est valide conformément à la recommandation CSS niveau 3 !

Pour montrer à vos lecteurs votre souci d'interopérabilité lors de la création de cette page Web, vous
XHTML que vous pouvez utiliser pour ajouter cette icône à votre page Web:

```
<p>
    <a href="http://jigsaw.w3.org/css-validator/check/referer">
        <img style="border:0;width:88px;height:31px"
            src="http://jigsaw.w3.org/css-validator/images/vcss"
            alt="CSS Valide !" />
    </a>
</p>
```

```
<p>
<a href="http://jigsaw.w3.org/css-validator/check/referer">
    <img style="border:0;width:88px;height:31px"
        src="http://jigsaw.w3.org/css-validator/images/vcss-blue"
        alt="CSS Valide !" />
</a>
</p>
```

- JavaScript : http://www.javascriptlint.com/online_lint.php

Tiens au fait ? Que se passe-t-il si des balles se superposent au moment du clic. Voilà un problème que l'on n'avait pas envisagé au moment de la conception et que je vous soumets. Indice, il y a peut-être des choses à vérifier dans la fonction clicCanvas() (à propos, sait-on quelle est la balle qui est par-dessus l'autre ? Est-ce important ?).

5 – Jouabilité

À ce stade le jeu est simplement fonctionnel. Avant de le publier, il faut définir un jeu de données correspondant à un cas réel, c'est-à-dire des tableaux de niveau et de balles tels que le jeu puisse être jouable/intéressant. À vous de définir une succession de niveaux de difficulté croissante, par exemple, en modifiant essentiellement la vitesse des balles.

Il faut également définir un temps limite. Ce temps limite doit a priori être supérieur au temps passé dans chaque écran. Mais, comme le fait de cliquer les bonnes balles plus vite ne modifie pas le gameplay, ce temps limite ne servirait donc à rien ? Je vous laisse échanger sur ce point.