

TP : hybride ou native

Doit-on réaliser le projet avec AI2 et du code natif, ou mixer les éléments natifs et les possibilités de la nouvelle version du HTML, assistée de CSS3 et de JS ?

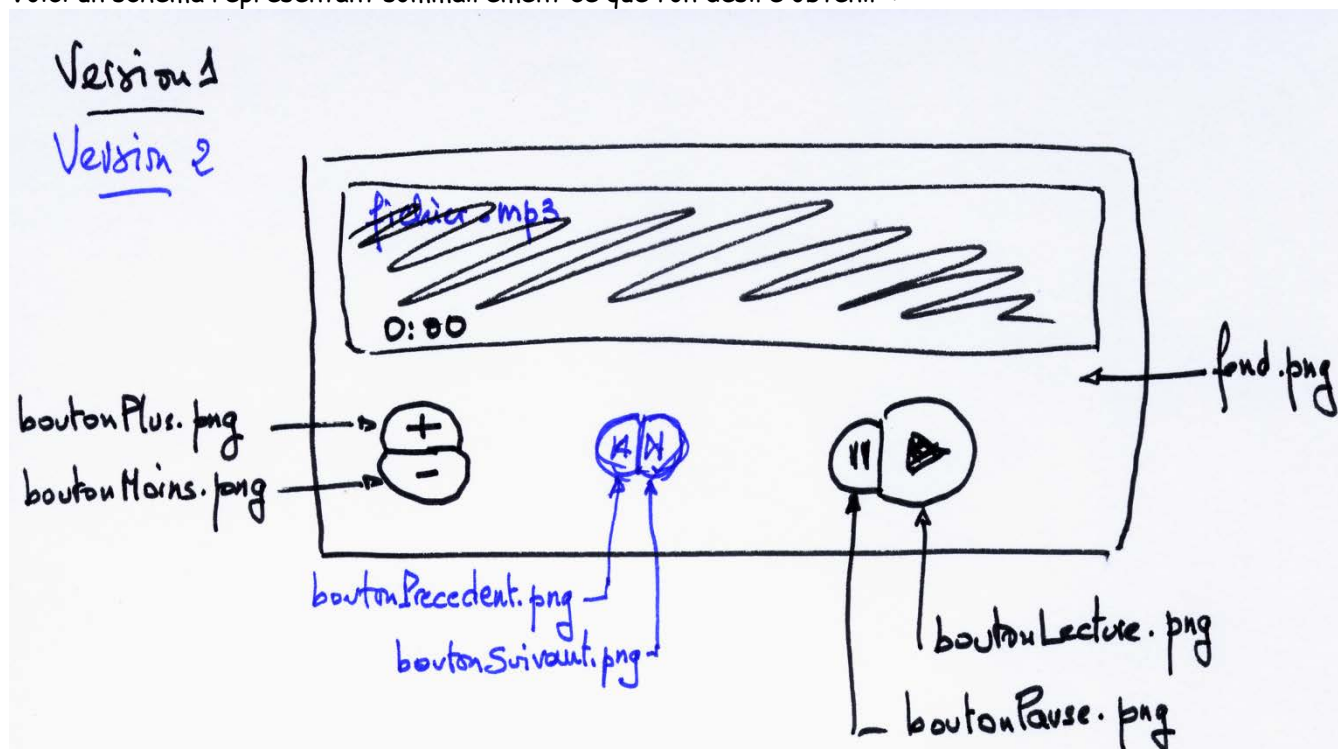
Voici un TP qui va permettre de réaliser les 2 versions, de façon à mieux se rendre compte.

Le but est de réaliser un lecteur de fichiers audio, avec les éléments classiques de l'interface de ce type d'applications.

Habillage

L'interface sera constituée d'éléments déjà dessinés dans un traitement d'images importés dans l'application.

Voici un schéma représentant sommairement ce que l'on désire obtenir :



Le nom des fichiers images est également indiqué. Tous les fichiers sont au format PNG pour pouvoir utiliser la « transparence ».

Les différents éléments prendront exactement la même place dans les deux versions (au pixel près !).

Le bouton « Lecture » permettra de démarrer la lecture du fichier MP3.

Le bouton « Pause » permettra de suspendre la lecture en cours.

Le bouton « + » permettra d'augmenter le niveau sonore.

Le bouton « - » permettra de réduire le niveau sonore.

L'affichage du temps de lecture et de la durée du fichier sera réalisé dans la fenêtre de l'écran.

En guise d'évolution, il serait souhaitable de pouvoir choisir entre plusieurs fichiers sons et de visualiser cette information. Et pourquoi ne pas changer « l'image » affichée sur l'écran ?

Et si on veut choisir soi-même les fichiers à lire ? Et si, et si ...

TP : hybride ou native

1 La version HTML5

HTML a beaucoup évolué, il s'appuie de plus en plus sur des éléments extérieurs, des bibliothèques entières utilisables pour faciliter la programmation en javascript , et CSS 3 pour la mise en forme rapide des éléments de la pages. A tel point qu'il ne reste plus grand-chose en HTML parfois comme nous allons le voir.

1.1 Mise en place du cadre

Il nous faut travailler sur deux fichiers, indissociables, le fichier index.html et le fichier style.css. Pour ces fichiers, un simple éditeur de texte peut faire l'affaire.

Les captures d'écran sont faites avec le Notepad++, mais ce n'est pas le plus important.

Vous devez créer un nouveau fichier, le nommer index.html et lui donner les attributs d'un fichier HTML :

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 </head>
5 <body>
6 </body>
7 <script>
8 </script>
9 </html>

```

à savoir une définition de type : <!DOCTYPE html> et bien évidemment le jeu de balises <html> encadrant les balises <head>, <body> et <script>.

Le fichier style.css est un simple fichier texte, sans élément caractéristique. Enregistrez ces deux fichiers dans votre espace de travail.

La référence au fichier css doit être présente dans le fichier html, dans la balise <head> pour être précis. Et tant qu'à traiter l'entête du fichier html, autant en profiter pour lui donner un titre, et surtout pour lier la bibliothèque jQuery qui va nous servir pour la partie « interactive ».

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Zmp3</title>
5 <link rel="stylesheet" href="style.css" />
6 <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
7 </head>
8 <body>
9 </body>
10 <script>
11 </script>
12 </html>

```

Le lien est fait sur la bibliothèque jQuery, version 1.8.3, mais devrait également fonctionner avec la toute dernière version disponible, la 3.2.1, sur le site de Google.

Il nous faut maintenant créer une « boîte » devant contenir notre lecteur mp3.

Dans la balise <body>, insérez une balise <div> nommée scene.

```

7 </head>
8 <body>
9 <div id="scene">
10 </div>
11 </body>
12 <script>

```

Mais il faut aussi travailler dans le fichier css pour « habiller » cette boîte !

Basculez dans la fenêtre d'édition de votre fichier style.css et complétez la définition de votre « scene » comme indiqué dans la capture d'écran.

```

1 #scene {
2 position : relative;
3 height : 200px;
4 width : 400px;
5 background-image: url('images/fond.png');
6 margin-left:50%;
7 left:-200px;
8 }
9

```

Votre « scene » sera placée au centre (horizontalement) de la fenêtre du navigateur, et l'image fond.png va servir de fond pour la boîte (div) nommée scene. Enregistrez vos deux fichiers et visualisez index.html dans votre navigateur internet préféré pour voir votre travail.

TP : hybride ou native

C'est bon, l'image est bien centrée, quelle que soit la taille de la fenêtre ? Donc on continue ...

1.2 Mise en place des éléments

Maintenant que la première boîte est posée, c'est parti pour le jeu de briques !

Il nous faut une boîte pour ranger les boutons de lecture, pause et arrêt, ainsi qu'une autre pour les contrôles de volume. Dans la fenêtre du fichier html, dans la balise div nommée scene, ajoutez une nouvelle balise div, nommée controleLecture, puis une autre balise div nommée controleSon. Chacune de ces boîtes va recevoir les éléments appropriés :

Dans la balise controleLecture, ajoutez les trois images correspondantes, boutonLecture.png, boutonPause.png et boutonStop.png, ces trois objets recevant respectivement les noms suivants : play, pause et stop.

De même pour la boîte controleSon, il s'agit des images boutonPlus.png, référencée sonP et boutonMoins.png avec l'identificateur sonM.

Ce qui devrait vous donner un code proche de celui de la capture d'écran suivante :

```

9 <div id="scene">
10 <div id="controleLecture">
11     
12     
13     
14 </div>
15 <div id="controleSon">
16     
17     
18 </div>
19 </div>

```

Enregistrez votre fichier puis testez-le dans votre navigateur.
Pas terrible !!! Les images sont bien chargées, mais pas au bon endroit.
Demandons à CSS de s'en charger ...

Dans la fenêtre du fichier style.css, il faut définir la position et l'aspect des différents éléments que nous venons de déposer dans la page HTML.

Ajoutez les définitions de la boîte de controleLecture et de celle de controleSon comme indiquées ci-dessous :

```

7 left: -200px;
8 }
9
10 #controleLecture {
11     position : absolute;
12     height : 52px;
13     width : 81px;
14     left: 226px;
15     top: 130px;
16 }
17
18 #controleSon {
19     position : absolute;
20     height : 50px;
21     width : 34px;
22     left: 38px;
23     top: 129px;
24 }
25
26

```

Enregistrez ce fichier et testez le fichier index.html dans votre navigateur (ou rafraichissez votre affichage si le navigateur est toujours ouvert).
C'est bien mieux, mais ce n'est pas encore ça ! Les boutons sont bien rangés dans leur boîtes, mais un peu en vrac !!!
Il va falloir ranger mieux que ça.
Et toujours dans le code CSS ...

Dans le fichier style.css, à la suite, déclarez les éléments pause, play et stop et affectez-les des éléments correspondants à leur emplacement :

TP : hybride ou native

```

23     top: 129px;
24     }
25
26     #pause{
27     position : absolute;
28     height: 35px;
29     width: 29px;
30     left: 1px;
31     top: 9px;
32     }
33     #play{
34     position : absolute;
35     height: 52px;
36     width: 48px;
37     left: 29px;
38     top: 0px;
39     }
40
41     #stop{
42     position : absolute;
43     height: 35px;
44     width: 29px;
45     left: 1px;
46     top: 9px;
47     }
48

```

Cela positionne les trois boutons correctement dans leur boîte, les dimensions en pixels faisant référence de façon absolue au contenu de la boîte controleLecture. Vous pouvez le vérifier en enregistrant le fichier et en visualisant le fichier index.html dans votre navigateur.

Ca progresse !

Pour les boutons de contrôle du volume, même traitement, dans la boîte controleSon :

```

48
49     #sonP {
50     float:top ;
51     }
52
53     #sonM {
54     position : absolute;
55     height : 28px;
56     width : 34px;
57     left: 0px;
58     top: 27px;
59     }
60

```

La position du boutonPlus est définie juste par un top pour la propriété float. Le bouton sera donc placé avec le haut de l'image contre la limite de la boîte controleSon, plus simple que de définir tous les éléments comme pour le sonM.

Testez votre travail après avoir enregistré votre fichier. Voilà qui est beaucoup mieux.

Mais où est passé le bouton pause ?

Bien rangé dans sa boîte, au même endroit que le bouton stop ! Mais en dessous de celui-ci. Une ligne peut arranger cela, la propriété z-index, qui permet de définir l'ordre de plusieurs éléments.

Ajoutez z-index : 1 pour le stop et z-index : 2 pour le pause.

Enregistrez votre fichier et testez index.html : c'est bon, le bouton pause est de nouveau visible, mais plus le bouton stop.

Si vous modifiez la propriété z-index de pause en lui affectant la valeur 0, après enregistrement et rafraîchissement, c'est le bouton stop qui est de nouveau visible. Il est donc possible d'avoir 2 images superposées (ou plus) et de choisir celle qui sera visible, par modification de la propriété z-index. A retenir !

Et maintenant, jouons ...

1.3 Interactivité ?

Si les boutons sont bien en place, il ne se passe pour l'instant rien, lors d'un clic souris sur ces boutons. Il va falloir maintenant faire intervenir le code JavaScript pour faire « bouger » les choses.

Pour s'assurer du bon fonctionnement de la page, il est plutôt conseillé de ne lancer les éléments du script que lorsque tous les éléments de la page html sont en place. C'est le rôle de l'instruction suivante :

`$(document).ready(function(){})` qui ne va autoriser la suite que lorsque le document (page html) sera prêt (tous les éléments chargés). Bien évidemment, cette instruction doit être placée à l'intérieur de la balise script.

```

19     </div>
20     </body>
21     <script>
22     $(document).ready(function() {
23     }
24     </script>
25     </html>

```

Et pour préparer l'interactivité, il faut encore créer une variable JS qui sera associée à l'objet Audio qui sera joué par notre player de MP3 :

`mplayer=new Audio()` (pour en savoir plus :

<http://www.w3.org/TR/2012/CR-html5-20121217/embedded-content-0.html#the-audio-element>)

TP : hybride ou native

Mais il ne s'agit pas de créer une variable pour un objet audio pour pouvoir le lire. Il faut également indiquer le chemin d'accès au fichier audio à charger. Et là, petit problème : quel format de fichier audio est géré par HTML5. Ce n'est en réalité pas tout à fait le nœud du problème, HTML5 est heureusement capable de gérer plusieurs formats de fichiers audio (MP3, OGG, wave) mais les navigateurs ne sont pas tous capables de lire tous ces formats. Pour des raisons évidentes, le lecteur ne va proposer que la lecture d'un fichier mp3 ou ogg, afin de couvrir l'ensemble des navigateurs et des plateformes. Il va donc falloir effectuer un test pour déterminer quel format de fichier sera utilisé. Cela se réalise avec la méthode `canPlayType` (pour plus d'infos : http://www.w3schools.com/tags/av_met_canplaytype.asp).

```

20 | </body>
21 | <script>
22 | $(document).ready(function() {
23 |     var mplayer = new Audio();
24 |     if (mplayer.canPlayType('audio/mp3') != "") {
25 |         mplayer.src="sons/son1.mp3";
26 |     }
27 |     else {mplayer.src="sons/son1.ogg";}
28 |
29 | })
30 | </script>
31 | </html>

```

Voilà à quoi doit ressembler votre page HTML maintenant.
Attention au point d'exclamation avant le signe égal !
Enregistrez et testez ...

Mais cela ne permet pas encore d'obtenir la lecture du fichier son.

Pour obtenir cette lecture il va falloir écrire la réponse à un événement click sur le l'objet nommé play :

```

21 | <script>
22 | $(document).ready(function() {
23 |     var mplayer = new Audio();
24 |     if (mplayer.canPlayType('audio/mp3') != "") {
25 |         mplayer.src="sons/son1.mp3";
26 |     }
27 |     else {mplayer.src="sons/son1.ogg";}
28 |
29 |     $('#play').click(function() {
30 |         mplayer.play();
31 |         return;
32 |     })
33 | })
34 | </script>
35 | </html>

```

Enregistrez et testez ...
Au clic sur l'image boutonLecture, la lecture du fichier audio se met en route, nickel !
Mais pour l'arrêter ... il faut fermer la page, pour l'instant ...
Programmons le bouton pause !

Comme pour l'objet play, c'est maintenant le tour de l'objet pause :

```

29 |     $('#play').click(function() {
30 |         mplayer.play();
31 |         return;
32 |     })
33 |
34 |     $('#pause').click(function() {
35 |         mplayer.pause();
36 |         return;
37 |     })
38 | })
39 | </script>

```

Une fois la saisie terminée, enregistrez et testez le fichier HTML.
Le bouton de lecture est toujours opérationnel, mais le bouton de pause l'est maintenant lui-aussi.
Encore un petit effort, les boutons de contrôle du volume ne sont pas actifs !
Mais avant de créer les réponses aux événements clic de ces boutons, il serait judicieux de définir le niveau sonore de la

lecture à 50% par exemple, afin de pouvoir l'ajuster par la suite. La propriété `volume` de l'objet audio permet de réaliser ce changement. C'est cette même propriété qui sera modifiée par les boutons plus et moins de contrôle du volume. A insérer à la suite de la déclaration de l'objet `mplayer`.

TP : hybride ou native

```

21 <script>
22 $(document).ready(function(){
23     var mplayer = new Audio();
24     mplayer.volume=0.5;
25
26     if (mplayer.canPlayType('audio/mp3') != "") {

```

Enregistrez et testez votre page ...
Le son est moitié moins fort, trop fort !

En réponse au clic sur un des boutons de réglage, le volume sera modifié de 10%, le volume pouvant varier de 0 à 1. Un test est mis en place de façon à rester dans les limites de la propriété.

```

36     $('#pause').click(function(){
37         mplayer.pause();
38         return;
39     })
40
41     $('#sonM').click(function(){
42         if (mplayer.volume-0.1>=0){
43             mplayer.volume-=0.1;
44         }
45     })
46
47     $('#sonP').click(function(){
48         if (mplayer.volume+0.1<=1){
49             mplayer.volume+=0.1;
50         }
51     })
52 })
53 </script>
54 </html>

```

Enregistrez et testez votre page ...
OK, tout fonctionne !

Mais où est passé le bouton stop ? Et puis, ça fonctionne, mais ce n'est pas encore achevé. Il manque un affichage de la lecture, de la durée, pourquoi pas du titre.

Cela risque d'amener quelques modifications profondes du code.
Enregistrez votre page HTML sous un autre nom, de façon à conserver la version initiale (index1.html fera l'affaire).

1.4 Améliorations

Pourquoi pas un « écran » bleu pour l'affichage des infos, affiché lorsque le lecteur est en fonctionnement. Tout d'abord, il nous faut créer une boîte pour recevoir l'image écranBleu.png, une autre, à l'intérieur, pour contenir du texte qui sera transmis « au lecteur » et une seconde pour recevoir les informations liées au temps.

Ce qui dans le fichier index2.html (pour version 2 de notre lecteur !) nous donne :

```

19 <div id="ecran">
20     
21     <div id="playerInfo">
22     </div>
23     <div id="playerTemps">
24     </div>
25
26 </div>
27

```

Bien évidemment, enregistrez votre fichier et testez-le.
L'écran bleu est bien présent, mais pas au bon endroit, c'est le rôle du fichier style.ccs qui lui aussi doit être modifié.

Dans le style.css, les objets écran, playerInfo et playerTemps doivent être définis ainsi :

```

61 #ecran {
62     position : absolute;
63     height : 95px;
64     width : 389px;
65     left: 3px;
66     top: 3px;
67     z-index:1;
68 }

```

```

70 #playerInfo {
71     position : absolute;
72     height : 26px;
73     width : 126px;
74     left: 10px;
75     top: 10px;
76     color:white;
77     z-index:2;
78 }

```

```

79 #playerTemps {
80     position : absolute;
81     height : 26px;
82     width : 126px;
83     left: 10px;
84     top: 46px;
85     color:white;
86     z-index:3;
87 }

```

C'est bon pour l'écran, mais quid des infos ?

TP : hybride ou native

Cela va se passer dans le fichier index2.html ... Il faut demander un affichage des données, mais quelles données ? Et où ? Dans la balise script, à la suite de la définition de la propriété volume, il suffit de créer 2 nouvelles variables, mplayerEtat, qui sera initialisée à stop et mplayerTemps, initialisée à 0.

Et dans la foulée, il faut demander l'affichage, dans l'objet nommé playerInfo, de la valeur de la variable mplayerEtat. Et même traitement pour la valeur de mplayerTemps qui sera affichée dans la boîte nommée playerTemps. Ce qui en JS nous donne :

```

30 <script>
31 $(document).ready(function() {
32     var mplayer = new Audio();
33     mplayer.volume=0.5;
34     var mplayerEtat="stop";
35     var mplayerTemps="0";
36     $('#playerInfo').text(mplayerEtat);
37     $('#playerTemps').text(mplayerTemps);
38
39     if (mplayer.canPlayType('audio/mp3')!="") {

```

Enregistrez, testez ...

Ecran bleu, un stop et, en dessous, un 0, blancs, OK, plutôt bon signe !

On ne va pas s'arrêter en si bon chemin ?

Il faut maintenant modifier l'état du lecteur lorsque les boutons de lecture sont utilisés et faire défiler la durée de lecture.

Il vous faut compléter le script des événements click pour les deux boutons de la façon suivante :

```

44 $('#play').click(function() {
45     mplayer.play();
46     mplayerEtat="lecture";
47     $('#playerInfo').text(mplayerEtat);
48     return;
49 });
50
51 $('#pause').click(function() {
52     mplayer.pause();
53     mplayerEtat="pause";
54     $('#playerInfo').text(mplayerEtat);
55     return;
56 });
57

```

Pour ce qui est de faire défiler le temps de lecture, il faut faire appel à un événement régulier, or la méthode setInterval semble toute désignée (+ d'infos à <http://www.toutjavascript.com>).

En utilisant un délai de 100 ms, l'affichage sera mis à jour 10 fois par seconde. Mais il faut connaître la durée du fichier son utilisé. La propriété duration de l'objet audio renvoie le nombre de secondes correspondant à cette durée. Une petite remise en forme et la durée peut être exprimée en minutes

et secondes. De même, la propriété currentTime renvoie le nombre de secondes écoulées depuis le début de la lecture. Il serait intéressant de charger 2 fonctions de réaliser les calculs. En voici le script :

```

73 function dureeCalcul() {
74     var minutes = Math.floor(mplayer.duration / 60);
75     var seconds = Math.round(mplayer.duration - minutes * 60);
76     return minutes+' : '+seconds;
77 }
78
79 function timedCount() {
80     var minutes = Math.floor(mplayer.currentTime / 60);
81     var seconds = Math.round(mplayer.currentTime - minutes * 60);
82     $('#playerTemps').text(minutes+' : '+seconds+' // '+dureeCalcul());
83     return;
84 }

```

Bien sûr, pour pouvoir fonctionner, il faut ajouter l'appel à la fonction setInterval en début de script :

```

36 $('#playerInfo').text(mplayerEtat);
37 $('#playerTemps').text(mplayerTemps);
38 var timerAff = setInterval(timedCount,100);
39
40 if (mplayer.canPlayType('audio/mp3')!="") {

```

Enregistrez et testez votre travail.

C'est bon ? On continue ...

Petite remarque, la ligne 37 est rendue obsolète par la ligne 38 (le 0 est

immédiatement remplacé par 0 : 0 // 0 : 0. Elle n'est plus nécessaire. Et la variable mplayerTemps ? Aussi, car il est plus rapide de directement affecter un texte à une boîte que de le stocker dans une variable puis de faire afficher la valeur de la variable. Donc exit la ligne 35 !

TP : hybride ou native

```

30 <script>
31 $(document).ready(function(){
32     var mplayer = new Audio();
33     mplayer.volume=0.5;
34     var mplayerEtat="stop";
35     //var mplayerTemps="0";
36     $('#playerInfo').text(mplayerEtat);
37     $('#playerTemps').text(mplayerTemps);
38     var timerAff = setInterval(timedCount,100);
39
40     if (mplayer.canPlayType('audio/mp3')!=""){

```

C'est pas mal, comme vous pouvez en juger en enregistrant et testant le fichier.
Un truc subsiste : le bouton stop ... Pour l'instant, il ne sert pas !
Pour le faire fonctionner, un événement de type click, comme les autres, et le tout est joué !
Oui, mais que faire faire à ce bouton ?
Un bouton stop doit arrêter la lecture du fichier

et ramener la tête de lecture au point de départ. L'arrêt de la lecture est obtenu avec le bouton pause. Pas de problème, mais pour mettre au départ ?

Deux solutions s'offrent à nous. La première, simple, consiste à recharger le fichier. La seconde, plus élégante, à positionner la tête au début du fichier, en mettant la propriété currentTime de l'objet audio à 0. Optons pour la seconde, ce qui nous donne si on insère le code du bouton stop après celui du bouton pause :

```

58
59 $('#stop').click(function(){
60     mplayer.pause();
61     mplayerEtat="arrêt";
62     $('#playerInfo').text(mplayerEtat);
63     mplayer.currentTime=0;
64     return;
65 })
66
67 $('#sonM').click(function(){

```

Enregistrez et testez votre page ...
Y'a toujours pas de bouton stop visible, il fonctionne (ou pas ...).
Pour faire apparaître le bouton, il faudrait le faire passer au-dessus du bouton pause ! On a déjà fait un truc comme ça !!! Dans le fichier style.css, un réglage de z-index !!! Passez à 3 la valeur de z-index du

bouton stop. Enregistrez et rafraîchissez votre page html. Le bouton stop est visible, et opérationnel !!!
Super, mais on a plus de bouton pause !!! Encore un petit effort ...

Pour que le bouton stop soit accessible, il faut qu'il remplace le bouton pause : lors de la lecture, le bouton pause est visible, si on met en pause, c'est le bouton stop qui doit être visible. Et si on utilise le bouton lecture, c'est le bouton pause qui doit être visible. Logique ! Reste à mettre cela en musique ... en js plutôt !

```

45 $('#play').click(function(){
46     mplayer.play();
47     mplayerEtat="lecture";
48     $('#playerInfo').text(mplayerEtat);
49     document.getElementById("stop").style.zIndex="1";
50     return;
51 })
52
53 $('#pause').click(function(){
54     mplayer.pause();
55     mplayerEtat="pause";
56     $('#playerInfo').text(mplayerEtat);
57     document.getElementById("stop").style.zIndex="3";
58     return;
59 })
60
61 $('#stop').click(function(){
62     mplayer.pause();
63     mplayerEtat="arrêt";
64     $('#playerInfo').text(mplayerEtat);
65     mplayer.currentTime=0;
66     document.getElementById("stop").style.zIndex="1";
67     return;
68 })
69

```

Attention à bien remettre la valeur de z-index à 1 dans style.css pour le bouton stop avant de tester votre page.
Tout fonctionne ? Les boutons jouent à cache-cache ? Bravo, voici votre première version HTML5 d'une application Flash.
Votre page est maintenant visible sur tous les supports et tous les navigateurs.
Eventuellement, proposer à l'utilisateur un choix pour la version à utiliser ...
D'autres améliorations sont envisageables :
Le titre du morceau dans l'écran, la valeur du volume sur l'écran, des effets sur les boutons ...
A vous de voir jusqu'où vous voulez aller !
Des exemples à <http://www.zoliv.fr/HTML5/zmp3.html>

TP : hybride ou native

2 La version hybride

Ouvrez un nouveau projet AI2. Choisissez comme paramètre d'orientation « Paysage », un alignement vertical et horizontal centré.

Insérez un objet WebViewer dans votre page.

Dans les paramètres de votre objet WebViewer donnez l'adresse (publique) de votre page html (attention, la visibilité doit être complète, pas de site local !).

C'est tout ...

Vous pouvez enregistrer, publier et écouter !

Bien évidemment, dans ce mode de conception, une connexion 3/4G est nécessaire pour écouter la musique puisque la totalité des éléments sont transférés à la demande, depuis le serveur web qui héberge les différents éléments. Donc potentiellement une consommation de données ...

3 La version native

Ouvrez un nouveau projet AI2, avec les mêmes paramètres que pour la version hybride.

Pour la construction de l'interface, il va falloir peut-être un peu réfléchir, voir même faire des dessins pour « caser » les différents éléments graphiques de l'interface.

Il existe 3 façons d'insérer des images :

- Comme des images (déco)
- Comme fond de zone (background d'un arrangement)
- Comme image de fond pour un bouton, sans texte c'est mieux !

Dans tous les cas, il faut d'abord importer (upload) les fichiers dans l'application.

Il en va de même pour le (ou les) fichier(s) mp3 que l'on veut utiliser par la suite.

Ces fichiers seront tous rendus disponibles localement dans l'appareil lors de l'installation de l'application.

Ils font partie de la compilation effectuée par AI2. Et, petite astuce, le chemin d'accès à un fichier « installé » est simplement « //nomdufichier » lorsqu'on le définit par codage, cela peut-être très utile.

Pour pointer vers un fichier, il faut au préalable insérer un objet Fichier de l'onglet Stockage dans la page, objet qui sera invisible, donc présent sous la fenêtre de construction.

A vous de jouer, les éléments et leurs rôles ont été déjà détaillés dans la version HTML, il suffit d'essayer de reproduire les différents comportements.