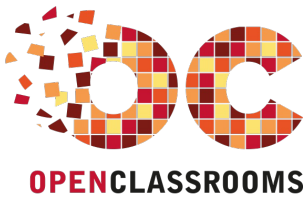
Un lecteur MP3 en flash

Par Gora



www.openclassrooms.com

Sommaire

Sommaire 2
Un lecteur MP3 en flash 3
Théoriquement 3
Mise en place des boutons de lecture 3
L'ActionScript 4
1 - Introduction des fonctions sur les boutons 4
3 - Déclaration des principales variables 4
3 - Déclaration des fonctions des différents boutons du lecteur 5
4 - Le code final 6
Finalisation 7
Annexes 8
L'affichage 8
Partager 9

Un lecteur MP3 en flash

Par  Gora

Mise à jour : 01/01/1970

Bonjour,

Dans ce tutoriel, je vais vous apprendre à créer votre propre lecteur MP3 exportable en flash.
Vous connaissez sans doute les mini-lecteurs flash souvent utilisés sur les blogs, comme celui-ci par exemple.

Si vous y arrivez (ceci est au conditionnel), vous pourrez enfin frimer avec votre joli petit lecteur !

Let's go !
Sommaire du tutoriel :

- Théoriquement
- Mise en place des boutons de lecture
- L'ActionScript
- Finalisation
- Annexes

Théoriquement

Théoriquement, notre lecteur mp3 doit être exportable pour pouvoir être utilisé sur diverses pages web ; nous utiliserons donc le code suivant pour le placer :

Code : HTML

```
<object type="application/x-shockwave-flash" data="lecteur.swf?
fichier=mamusique.mp3" width="LARGEUR" height="LONGUEUR">
  <param name="movie" value="lecteur.swf?fichier=mamusique.mp3" />
</object>
```

Pour commencer, nous n'allons utiliser que les boutons Play, Pause et Stop, nous verrons ensuite les possibilités annexes.

Mise en place des boutons de lecture

Commençons. Tout d'abord, pour permettre une lecture sonore plus agréable, il nous faut des boutons Play, Pause, et Stop. Si vous savez faire par vous-mêmes vos boutons, faites-le ; vous aurez un lecteur personnalisé. Si vous ne savez pas les faire, utilisez les **bibliothèques communes**. Ou alors vous pouvez prendre les boutons du fichier source disponible à la fin de ce tuto !

C'est bien, mais c'est quoi, les bibliothèques communes ?

Les bibliothèques communes (Fenêtre -> Bibliothèques communes) se divisent en trois catégories : les boutons, les classes et les interactions de formations. Ce sont des composants livrés avec le logiciel que vous possédez et parfois très utiles.
Dans notre cas, nous allons nous intéresser aux boutons.

Choisissez les boutons que vous souhaitez. Sur le Calque 1 que vous renommerez **lecteur**, vous placerez ces boutons comme vous le voulez !

Avant de passer à l'étape suivante, nous allons nommer une **occurrence** pour chacun de ces boutons qui permettra au code ActionScript de les reconnaître pour les actionner (et c'est le cas de le dire).

Donc, nous allons nommer respectivement l'occurrence de nos boutons Play, Stop et Pause : **`btnLecture`**, **`btnStop`**, et **`btnPause`**.

Vous êtes prêts ? Maintenant, nous allons passer à la plus grosse partie de ce tutoriel : le code (il ne manque plus que la musique de Rocky pour nous mettre dans l'ambiance).

L'ActionScript

Maintenant, nous allons placer l'ActionScript.

Rappel : l'ActionScript est un langage de programmation utilisé au sein des applications Flash d'Adobe. Ce langage nous permet d'ajouter de l'interactivité aux animations flash en répondant aux actions de l'utilisateur.
L'ActionScript ce place dans "le panneau de programmation ActionScript" (F9).

1 - Introduction des fonctions sur les boutons

Tout d'abord, nous allons appeler les **fonctions** du méga-code-final sur nos boutons en modifiant leurs comportements ; sélectionnez un bouton et ouvrez le panneau de programmation : si tout est bon, vous verrez un onglet en bas à gauche du panneau avec le nom de l'occurrence du bouton.

Trêve de bavardages, voici le code :

Code : Autre

```
on (release) //Si l'utilisateur a cliqué sur le bouton, on continue
{
  _root.lecture(); //Ici, on appelle la fonction
}
```

Bien sûr, vous n'allez pas laisser **`_root.lecture();`** sur chaque bouton ; par exemple, pour Stop, on va utiliser **`_root.stop();`** et pour Pause **`_root.pause();`**.

Ensuite, nous allons créer un nouveau calque que l'on nommera **actions**. Je vais fournir bout à bout le code ainsi que les explications pour que ce soit plus clair.

Je vous déconseille fortement d'utiliser plusieurs images-clefs. Tout au long de ce tuto, nous resterons sur la première image de l'animation.

Ouvrez le panneau de programmation ActionScript sur le calque **actions**.

3 - Déclaration des principales variables

Code : Autre

```
//Tout d'abord, nous déclarons les principales variables

var decalage = 0;
var charge = false;
var temps, si;

//Les propriétés du son joué

chanson = new Sound();
chanson.onSoundComplete = function() {
  _root.stop();
};
```

Ici, nous avons commencé par déclarer les principales variables. **`decalage`** et **`temps`** nous serviront principalement pour le bouton Pause, **`charge`** nous servira à boucler la musique quand elle sera terminée.

3 - Déclaration des fonctions des différents boutons du lecteur

Code : Autre

```
//On déclare la fonction stop

function stop()
{
  clearInterval(si);
  chanson.stop();
  decalage = 0;
}
```

Voici la fonction pour le bouton Stop, très simple ; on stoppe la chanson puis on la reprend du début au prochain démarrage.

Code : Autre

```
//On déclare la fonction pause

function pause()
{
  clearInterval(si);
  chanson.stop();
}

//Puis la fonction memoire liée à celle du dessus

function memoire()
{
  decalage = chanson.position/1000;
}
```

Ça se complique ; ici, nous déclarons la fonction **`pause`** quasiment comme au-dessus, puis, grâce à la fonction **`memoire`**, le lecteur va se souvenir où l'utilisateur a stoppé la musique via **`chanson.position`**, et enfin **`/1000`** signifie que la chanson sera reprise 1 seconde avant qu'elle soit stoppée.

Code : Autre

```
//On déclare la fonction lecture

function lecture()
{
  if (charge == false) //Si la chanson n'est pas chargée
  {
    chanson.onLoad = function(ok) //On la charge
        {
      if (ok=true) //Si elle est chargée
         {
        charge = true;
        temps = Math.round(chanson.duration/1000);
        chanson.start(); //On lit la chanson s'il y a plus d'une seconde d'écart
        si = setInterval(memoire, 1000);
      }
    };
    chanson.loadSound(fichier, true); //Dans le cas contraire, on charge la char
  }
  else
  {
    chanson.start(decalage); //Si la chanson est stoppée par l'utilisateur via l
    si = setInterval(memoire, 1000); //De même pour memoire
  }
}
```

Et enfin, le gros bloc de code pour conclure.
On déclare la fonction **`lecture`**, la plus complexe ; si la chanson n'est pas chargée, on la charge ; si elle est chargée et qu'il y a plus d'une seconde d'écart entre le chargement et l'état de la lecture, on la lit. Dans le cas contraire, on la stoppe jusqu'à ce que ça se produise.

4 - Le code final

Code : Autre

```
//Tout d'abord, nous déclarons les principales variables

var decalage = 0;
var charge = false;
var temps, si;

//Les propriétés du son joué

chanson = new Sound();
chanson.onSoundComplete = function() {
  _root.stop();
};

//On déclare la fonction stop

function stop()
{
  clearInterval(si);
  chanson.stop();
  decalage = 0;
}

//On déclare la fonction pause

function pause()
{
  clearInterval(si);
  chanson.stop();
```

```
}

//Puis la fonction memoire liée à celle du dessus

function memoire()
{
  decalage = chanson.position/1000;
}

//On déclare la fonction lecture

function lecture()
{
  if (charge == false) //Si la chanson n'est pas chargée
  {
    chanson.onLoad = function(ok) //On la charge
        {
      if (ok=true) //Si elle est chargée
         {
        charge = true;
        temps = Math.round(chanson.duration/1000);
        chanson.start(); //On lit la chanson s'il y a plus d'une seconde d'écart
        si = setInterval(memoire, 1000);
      }
    };
    chanson.loadSound(fichier, true); //Dans le cas contraire, on charge la char
  }
  else
  {
    chanson.start(decalage); //Si la chanson est stoppée par l'utilisateur via l
    si = setInterval(memoire, 1000); //De même pour memoire
  }
}
```

Finalisation

Votre lecteur est fin prêt !
Vous pouvez bien sûr le personnaliser à votre goût, ou bien commencer à faire une web-radio.

Je vous rappelle quand même le code HTML à insérer pour pouvoir profiter de votre lecteur (si la flemme s'empare de vous et que votre souris refuse de remonter en tête de page) :

Code : HTML

```
<object type="application/x-shockwave-flash" data="lecteur.swf?
fichier=mamusique.mp3" width="LARGEUR" height="LONGUEUR">
  <param name="movie" value="lecteur.swf?fichier=mamusique.mp3" />
</object>
```

Pourquoi la chanson est-elle chargée par la variable **`fichier`** (**`fichier=`**) ?

Très bonne question : vous souvenez-vous de ce bout de code lorsqu'on déclare la variable **`lecture`** ?

Code : Autre

```
};
    chanson.loadSound(fichier, true); //Dans le cas contraire, on charge la chans
  }
```

Eh bien c'est ceci qui nous permet de définir que la variable **fichier** contenue dans le code HTML contiendra le .mp3 à jouer !

Et puis, vu que je suis généreux, voici l'archive .rar avec le .fla, le .swf, la page HTML puis une musique libre de droit de Jenny Gillespie en prime.

Télécharger l'archive

Annexes

Faire un lecteur c'est bien, mais l'améliorer c'est mieux !
Oui je sais, ce n'est pas la meilleure des devises mais il suffit d'en comprendre le principe. Maintenant, vous savez faire un lecteur mp3 en flash avec trois fonctionnalités : la lecture, la pause et l'arrêt.
Pourquoi ne pas en découvrir de nouvelles ?

Dans cette partie du tutoriel, je ne vais pas vous dire exactement comment faire, mais je vais vous montrer le chemin à suivre.

L'affichage

Si nous affichions quelque chose sur notre lecteur pour le rendre plus interactif, par exemple un **champ de texte dynamique** avec des données ou bien des informations ?

Imaginons que nous souhaitons afficher un texte pour dire au visiteur ce qu'il fait ou bien ce qu'il doit faire. Par exemple au début, quand nous déclarons nos variables, vous pouvez ajouter :

Code : Autre

```
txt.affichage = "Cliquez sur le bouton Play pour commencer !";
```

Et ceci dans chaque fonction, avec un texte différent et le concernant (par exemple pour **stop** : "Vous avez stoppé la musique") et puis, pourquoi pas ? le nom de la musique .

De plus, pourquoi ne pas afficher une barre d'état, c'est-à-dire un **movie clip** qui indiquera l'avancement de la musique ?

Secret (cliquez pour afficher)

Code : Autre

```
barredetat._width = 0;
```

Ceci sera à placer au début, bien sûr, lors de la déclaration des variables ; ensuite, via des calculs, vous pourrez indiquer précisément où en est la musique.

Code : Autre

```
//Bien sûr, les bouts de code donnés ci-dessous sont à placer dans les fonctions

barredetat._width = 0;

function stop()
{
   barredetat._width = 0; //La taille de la barre d'état s'initialise à zéro
}

function memoire()
```

```
    {
        etat = Math.round((chanson.position*100)/chanson.duration); //Ici, on calc
        barredetat._width = etat; //On définit la taille de la barre d'état
    }
```

Nous voilà arrivés à la fin de ce tutoriel !

Peut-être vous demandez-vous pourquoi j'ai rédigé une partie *Annexes* sur ce que l'on pouvait rajouter en gros dans le lecteur ? Eh bien, question de logique : personnellement je trouve qu'un tutoriel est fait pour apprendre, et non pour copier bêtement le code que l'on vous donne ; il faut, avant tout, si l'on souhaite apprendre n'importe quel langage de programmation, comprendre et analyser ce que l'on fait !

Bref, j'espère que ce tuto vous aura servi, et si vous avez des questions n'hésitez pas, les forums sont là pour ça.

À bientôt.

Ce tutoriel a été corrigé par les zCorrecteurs.