LUDISCAPE - SCRIPT

Fonctionnement

Ludiscape affiche ces différents éléments à l'aide du **DOM**.

Le Document Object Model (ou DOM) est un standard du W3C (World Wide Web Consortium) qui décrit une interface indépendante de tout langage de programmation et de toute plate-forme, permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu.

Le DOM permet de construire une arborescence de la structure d'un document et de ses éléments.

Dans Ludiscape l'écran principal est représenté par un objet DOM de type DIV qui a l'identifiant **main**.

A la différence d'une page HTML statique, les objets DOM du moteur Ludiscape sont créés de manière dynamique à l'aide du langage JavaScript.

Exemple : Lorsque je suis sur l'écran N°1 Ludiscape ne va charger que les objets nécessaires à l'écran 1

De plus Ludiscape dimensionne les objets en fonction de la taille de l'écran.

Par défauts les objets sont considérés en pixels avec un ratio de 1

Exemple : Mon écran d'ordinateur me permet d'afficher un écran avec une largeur de 960 pixels je garde le ratio 1. Par contre si l'écran ne permet qu'une largeur de 800 pixels alors mon ratio sera de 800/960 soit **0.83**. Donc une image de 200px par 200px sera affichée avec les valeurs 166 px par 166px.

Ce ratio est calculé automatiquement par Ludiscape et stocké dans la Variable globale **zoom**.

jQuery

Ludiscape repose en grande partie sur le langage JavaScript et la librairie JQuery

La puissance de jQuery repose sur l'une de ses facettes : la sélection d'éléments. Qui permet de sélectionner, modifier et obtenir la valeur de la plupart des éléments qui peuvent être rencontrés dans le DOM de votre application pédagogique.

jQuery repose sur une seule et unique fonction : **jQuery(),** ou son alias, **$().** Cette fonction accepte un ou plusieurs paramètres et retourne un objet que nous appellerons **« objet jQuery ».**

Les paramètres peuvent être d'un des types suivants :

- Un sélecteur CSS : l'élément (ou les éléments) qui correspond au sélecteur est retourné.
- Un attribut de l'objet DOM, un document un texte ou l'objet window : un objet jQuery correspondant à cet élément est retourné.

```
//Changer le fond d'un objet
$(‘.element').css(‘background-color','red') ;
```

Dans cet exemple nous sélectionnons l'objet (ou les objets) Ludiscape avec un identifiant **element** et nous lui appliquons un fond rouge.

Le point fait référence à l'attribut **class** d'une balise.

Supposons que vous ayez défini la balise ** .** Le sélecteur **.rouge** sélectionne cette balise.

Pour des raisons pratiques (plusieurs objets peuvent avoir le même identifiant) Ludiscape permet de définir un ID qui est appliqué comme une classe.

Les objets HTML peuvent contenir un ou plusieurs attributs. Par exemple, la balise **<img src="data/chat.jpg" width="40" height="30">** contient trois attributs : src, width et height. Pour sélectionner toutes les balises qui contiennent un attribut src, vous utiliserez le sélecteur [src].

```
//Changer l'image d'un objet
$('.element').attr('src,'data/chien.jpg') ;
```

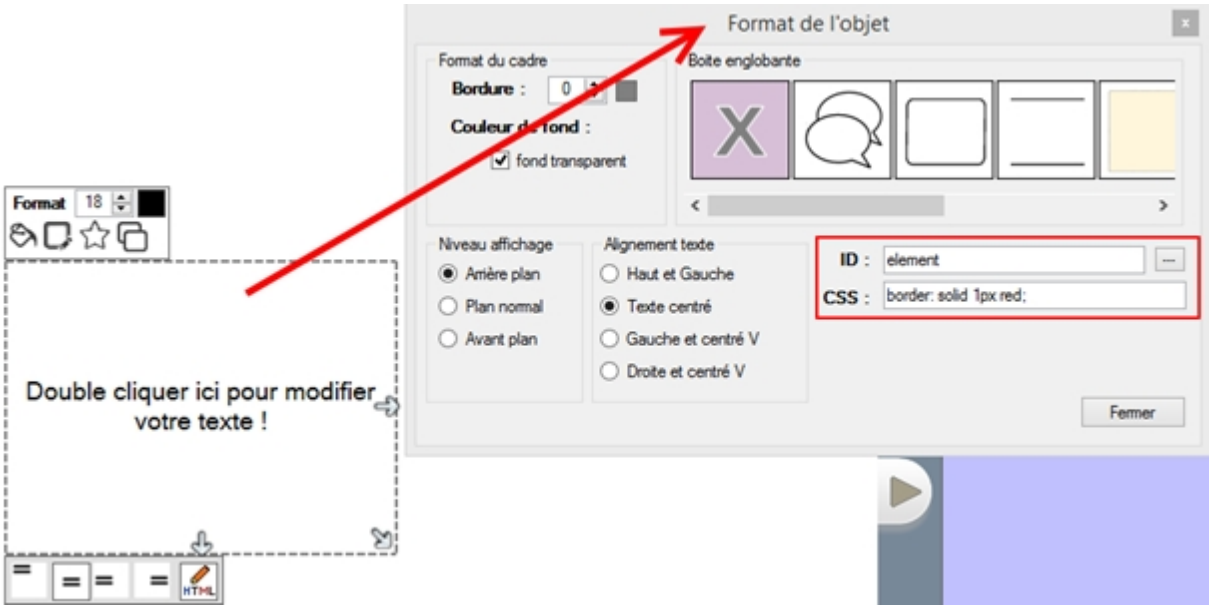
Note : *Le DOM doit être disponible et créé pour être manipulable, les objets de chaque écran étant créés à la demande le script sur un bouton action de l'écran 1 ne peut pas influer sur un objet de l'écran 2.*

Les attributs des objets

Pour éditer l'identifiant d'un objet dans l'éditeur Ludiscape faire clique-droit sur l'objet et sélectionnez **Format de l'objet** :

Deux zones permettent de :

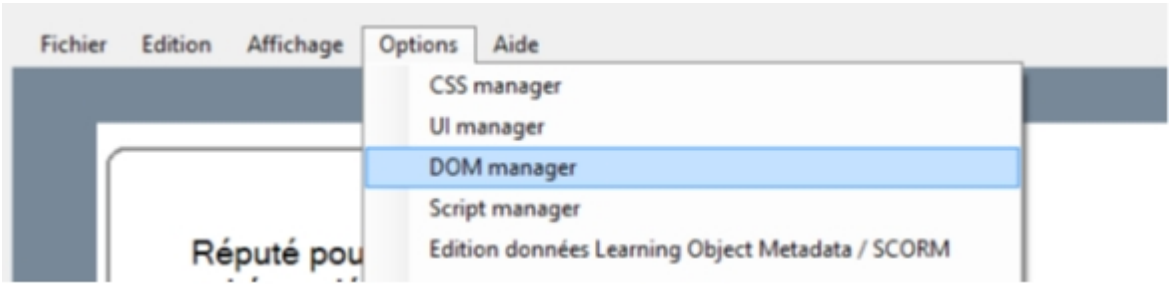
- *Saisir un Identifiant*
- *Saisir des propriétés CSS rapidement.*



DOM Manager

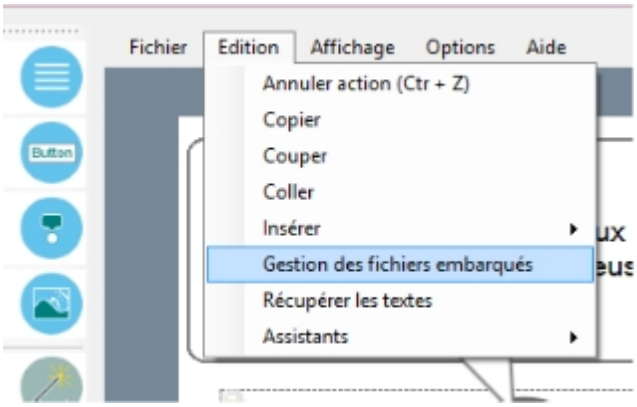
La fonction DOM manager permet d'insérer sur tous les écrans des éléments de DOM automatiquement dans

chaque écran.



Ajout de fichiers supplémentaires

La fonction Gestion des fichiers embarqués permet de visualiser et d'ajouter des fichiers supplémentaires à votre projet.



Note importante : Les fichiers de type JS et CSS sont automatiquement insérés comme bibliothèque et chargés en début de projet sauf si son paramètre contient **« no ».**

Un fichier **ZIP** peut être automatiquement décompressé au moment de la publication si son paramètre contient **« extract »**

Communiquer avec le moteur ludiscape

Fonctions ludiscape

Ludiscape possède une bibliothèque de fonctions simples pour intéragir avec les objets ou le moteru Ludiscape.

Vous trouverez sur cette page la description des fonctions

Fonctions de navigation :

`LUDI.nextPage();`	Permet de charger la page suivante	
`LUDI.nextPageAnd1();`	Permet de charger la page suivante + 1	

`LUDI.nextPageAnd2();`	Permet de charger la page suivante + 2	
`LUDI.prevPage();`	Permet de charger la page précédente	
`LUDI.goPage(3);`	Permet de charger une page	Exemple : chargement e la page d'accueil : `LUDI.goPage(0);`
`LUDI.displayLastPage();`	Permet de charger la dernière page affichée	

Fonctions d'animation :

`LUDI.fadeIn('ID');`	Fait apparaître un objet	
`LUDI.fadeOut('ID');`	Fait disparaître un objet	
`LUDI.locationXY('ID',100,100);`	Permet de redéfinir la position en X et Y	Exemple: je souhaite que l'objet image avec l'identifiant **boule** se positionne en x:10 et y:20 `LUDI.locationXY('boule',10,20);`
`LUDI.translateXY('ID',100,100);`	Permet de redéfinir la position en X et Y avec une glissade	Exemple: je souhaite que l'objet image avec l'identifiant **boule** glisse vers x:10 et y:20 `LUDI.translateXY('boule',10,20);`
`LUDI.mapTo('ID','TO');`	Permet de faire correspondre un objet a un autre en X Y et Width et Height.	Exemple : recouvrir une image avec une autre (jeu du mémory)
`LUDI.rotateAngle('ID',45);`	Permet de définir une rotation en degrées	Exemple: je souhaite que l'objet image avec l'identifiant **boule** se positionne avec une rotation de 20° `LUDI.rotateAngle('boule',20);`
`LUDI.wait(2000);`	Permet de définir une pause entre deux animations en millisecondes	

Fonctions pour agir sur les variables du contenu ludiscape :

`LUDI.updateNote(-5);`	Ajoute ou supprime des points sur le total apprenant	
`LUDI.updateScore(5);`	Ajoute ou supprime des points sur le total score de l'apprenant	
`LUDI.deleteLife();`	Permet de supprimer une vie, entraine une animation si l'objet life est présent dans la page	
`LUDI.updateNoteExam();`	Fonction qui donne une pénalitée de 1 point par écran non validé, fonction spéciale pour le générateur d'examen	

Fonctions pour récupérer et définir des valeurs textuelles :

`LUDI.setValueTxt('ID', 'texte');`	Définit la valeur d'un texte	

`LUDI.getValueInput('ID');`	Récupere la valeur d'un texte	`Exemple :` `var txt = LUDI.getValueInput('ID');`
`LUDI.getValueQcm('ID');`	Récupere les réponses sous la forme d'un texte	Exemple: `var txt = LUDI.getValueQcm('ID');`

Variables globales

//Fonctions utiles :

helperDateActu(); //Renvoi la date
helperHourActu(); // Renvoi l'heure

getXmlInteractions();//renvoi un XML avec toutes les intéractions apprenants

Les variables suivantes sont accecible et modifiables a tout moment via du script

//Variables utiles :

SatisfationScore; // Satisfaction de l'apprenant
N_T; // Nombre total de points au total
N_F; // Nombre total de points de l'apprenant
remarques; // ensembles des remarques depuis le départ

domainesN_T; // Tableau avec la totalité des points pour chaque domaine
domainesN_F; // Tableau avec la totalité des points de l'apprenant pour chaque domaine
domainesPour;// Tableau avec la pourcentage de reussite pour chaque domaine
domainesRemarques;// Tableau avec les remarques pour chaque domaine

LUDIscore; // Définit le score du joueur
LUDIlife; // Définit le nombre de vies du joueur

Ensemble de variables en utilisation libre :

Variable1 = ""; // La variable N°1 remplace automatiquement le tag {Variable1} dans une zone de texte

Variable2 = "";
Variable3 = "";
Variable4 = "";
Variable5 = "";
Variable6 = "";
Variable7 = "";
Variable8 = "";
Variable9 = "";
Variable10 = "";

Hooks

Qu'est-ce qu'un HOOK ?

Un **hook** (littéralement « crochet » ou « hameçon ») permet à l'utilisateur d'un logiciel de personnaliser le fonctionnement de ce dernier, en lui faisant réaliser des actions supplémentaires à des moments déterminés. Le concepteur du logiciel prévoit des **hooks** au long du fonctionnement de son programme, qui sont des points d'entrée vers des listes d'actions. Par défaut, le hook est généralement vide et seules les fonctionnalités de base de l'application sont exécutées.

Cependant, l'utilisateur peut « accrocher » des morceaux de programme à ces hooks pour personnaliser le logiciel. *Source wikipédia*

Dans Ludiscape il existe des hooks

Ces hooks ne sont appelés par le moteur que **si** ils été déclarés à un endroit (par exemple Script Manager, ou dans un fichier JS embarqué)

Dans ce TP vous devez afficher dans une boite de message (fonction alert) les différents paramètres de chaque hook.

Puis dans un deuxième temps :

Vous allez faire une copie du fichier ludiscape réalisé dans le **TP11** et afficher la progression et les remarques dans la zone de messages de manière automatique.

Ajouter plusieurs écrans QCM de facon à rendre le test plus réaliste.

```
//A la fin du contenu (appelé par l'objet fin de TP);
function ludiscapeFinishActivity(login,activityId,title,success,duration,score_raw){

}

//A la fin du contenu
function ludiscapeFinishAll(login,activityId,title,duration,score_raw){

}

//A la fin d'un écran (duration = milliseconds)
function ludiscapeFinishScreen(login,activityId,pageId,title,duration,score,scoreMax,annotations,xmlData){

}
```

Les interactions et leurs apercus

Avec les intéractions SCORM il est possible de visualiser les objectifs atteints ou non en fonction des identifiants questions.

Mais que faire lorsque que les interactions sont trop complexes pour être représentés par du texte.

La fonction getXmlInteractions() permet de récupérer les données XML de l'ensemble des interactions à un moment donné.

En rechargeant ce fichier dans votre contenu Ludiscape, vous pouvez à tout moment revoir tous les écrans et leurs ré ponses au pixel près.

Methode :

Dans le SCRIPT Manager veuillez rentrer le hook suivant qui permet de sauvegarder vos données xml vers une base ou une base de données :

HOOK JavaScript :

```
//A la fin du contenu
function ludiscapeFinishAll(login,activityId,title,duration,score_raw){
        //Si nous ne sommes pas en mode apercu alors
        if(ViewerAfterBilan==false){
                //Enregistrement des intéractions et des écrans
                $.ajax({
                  type: "POST",
                  url: "http://127.0.0.1:81/InteractionsLink/index.php",
                  data: { d: getXmlInteractions() },
                  success: function(data){},
                  error: function(){}
                });
        }
}
```

Puis lorsque vous souhaiter relancer l'aperçu ultérieurement :

http://monsomaine.com/scormModule/index.html?resume=interactions.xml

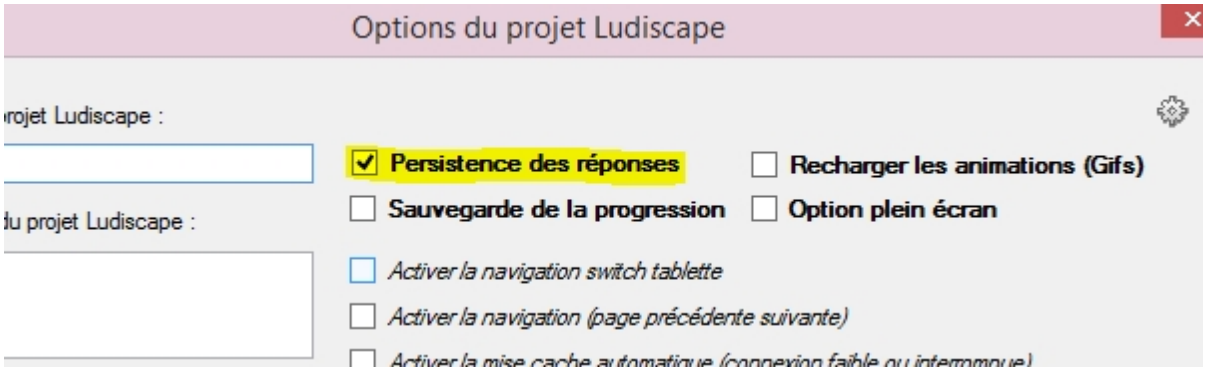
Votre contenu ludiscape va se relancer en mode bilan et apercu des réponses donnés

Changer la note par script

La note est représentée par deux variables :

N_T; // Nombre total de points au total
N_F; // Nombre total de points de l'apprenant

Cependant ces variables sont continuellement recalculées lorsque l'option persistance est cochée :



Pour agir par script sur la note il faut donc créer un objet question virtuel :

// exemple
LUDI.processNoteMemory('ma question',1,3,0,'remarque');

LUDI.processNoteMemory({1},{2},{3},{4},{5});

{1}	Nom de la question
{2}	Total de points obtenus

{3}	Total des points de la question
{4}	Numéro du domaine (domaine numéro 1 = 0; domaine numéro 2 = 1)
{5}	Remarque éventuelle pour le bilan

Les plugins

La Bibliothèque Ludiscape possède des objets Pédagogiques très puissants et qui suffisent pour la plupart des projets. Cependant pour optimiser vos créations pédagogiques il peut être utile de créer vos propres objets.

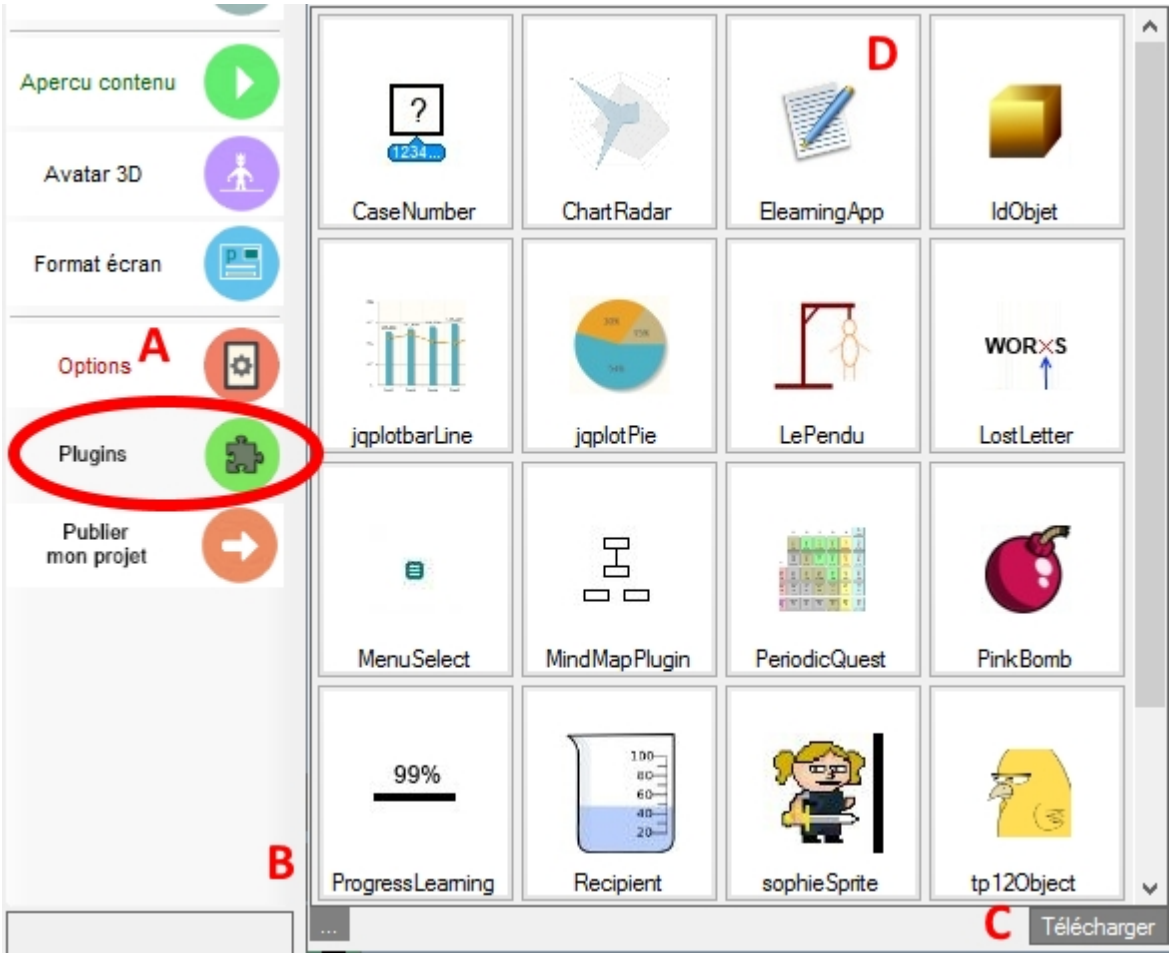
Ludiscape vous offre la possibilité de vous créer vos propres objets grâces aux plugins.

Les prérequis :

Tout d'abord, vérifier si le plug-in existe dans notre base (he oui pas la peine de créer un plug-in s'il existe déjà), s'il n'existe pas : vérifier si un plug-in se rapproche du vôtre, de façon à limiter votre travail.
Ensuite, il vous faudra connaître le Javascript et le CSS. L'avoir pratiqué en ayant créé des programmes ou applications est un plus, car vous saurez comment organiser votre code.
Si vous ne connaissez pas le Javascript, vous pouvez trouver des tutoriels sur le site developpez.com , ils sont globalement bien, mais ils ne vous apprennent pas tous (les énumérations, par exemple).
Par ailleurs, je vous conseille l'environnement de travail Notepad ++. Je vous le recommande véritablement, car il est très simple à utiliser.

Installer un plugin

Le menu Plugin se trouve en bas à gauche de votre liste d'outils.



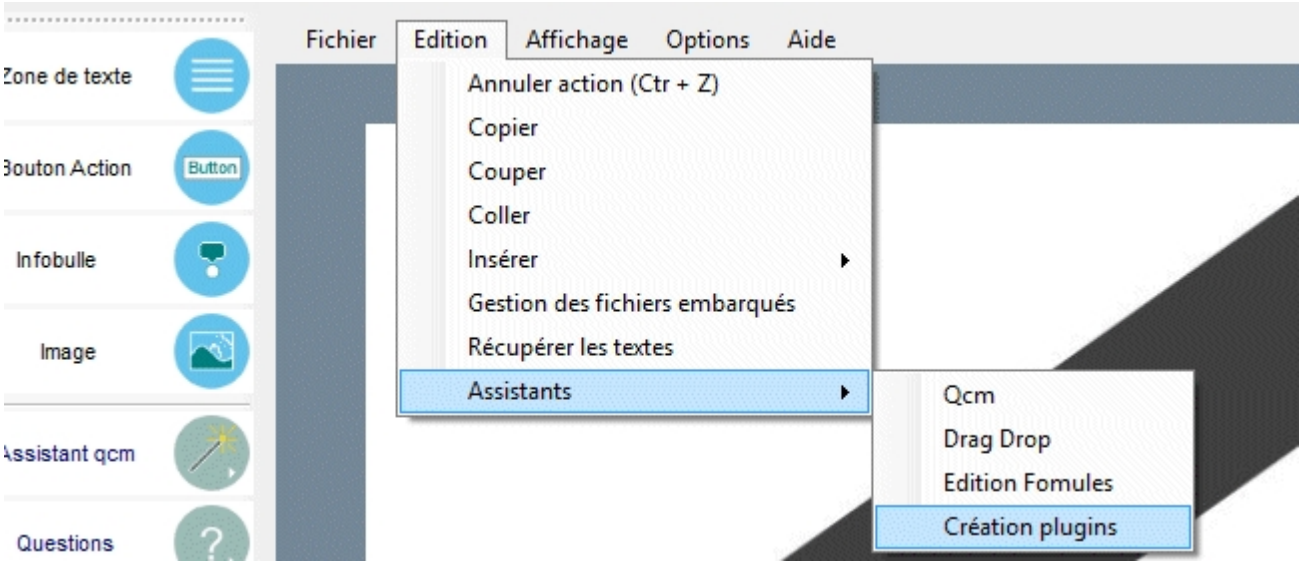
Etapes pour insérer un plugin :

1. Appuyer sur **A**
2. Utiliser le bouton de chargement **B** en bas à gauche pour charger votre plugin (sous la forme d'un fichier ZIP)
3. Si vous n'avez pas de fichier plugin vous pouvez en télécharger quelques uns sur le site www.ludiscape.com en **C**
4. Un fois le plugin charger il deveint disponible sous la forme d'un rectangle avec son icone et son nom associé. **D**

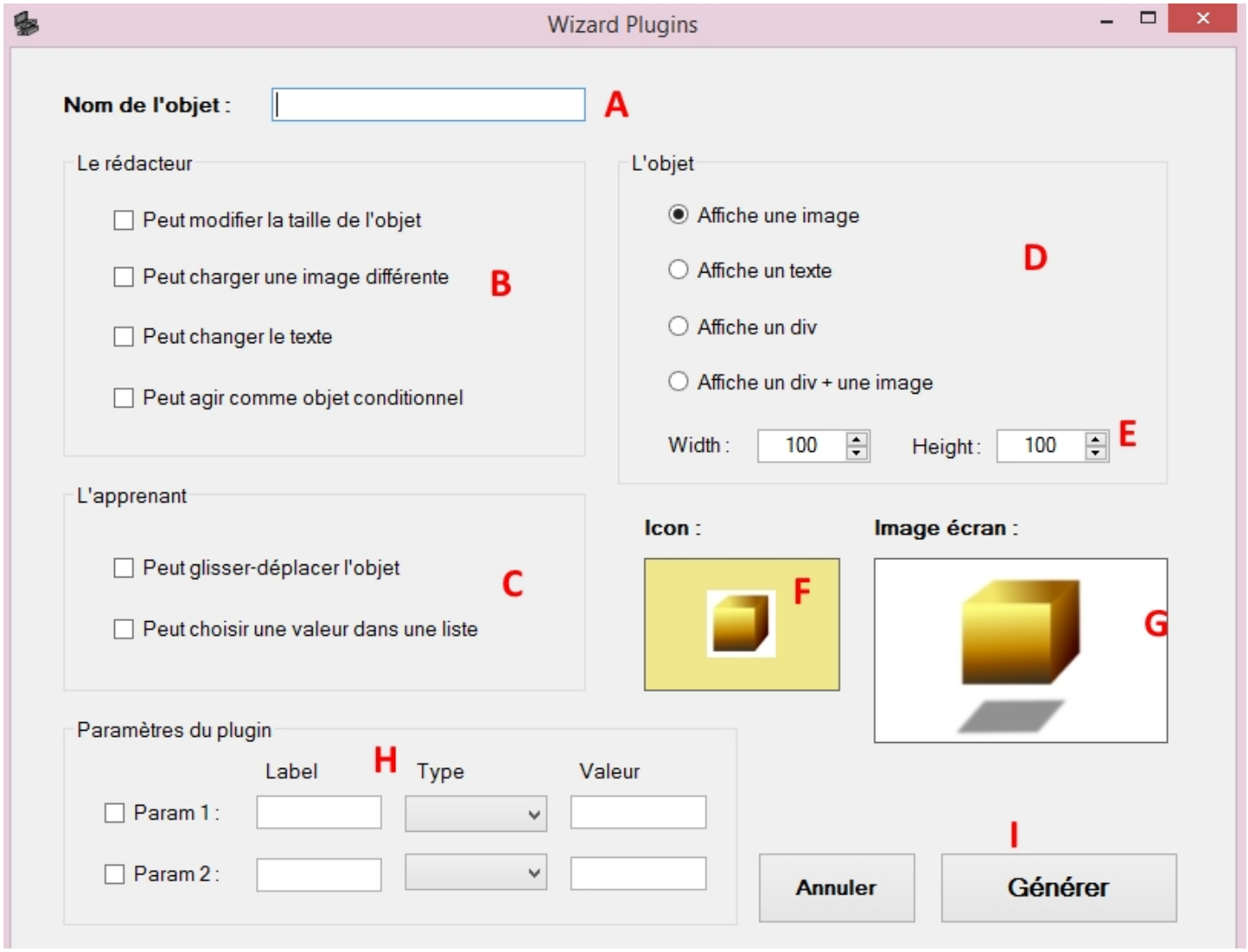
Assistant de création de plugins

Ludiscape possède un outil qui permet l'initialisation radide de vos plugins.

Vous trouverez cet assistant dans le menu **Assistants**>> **Création plugins**



Cet assistant lance le fenêtre ci-dessous :



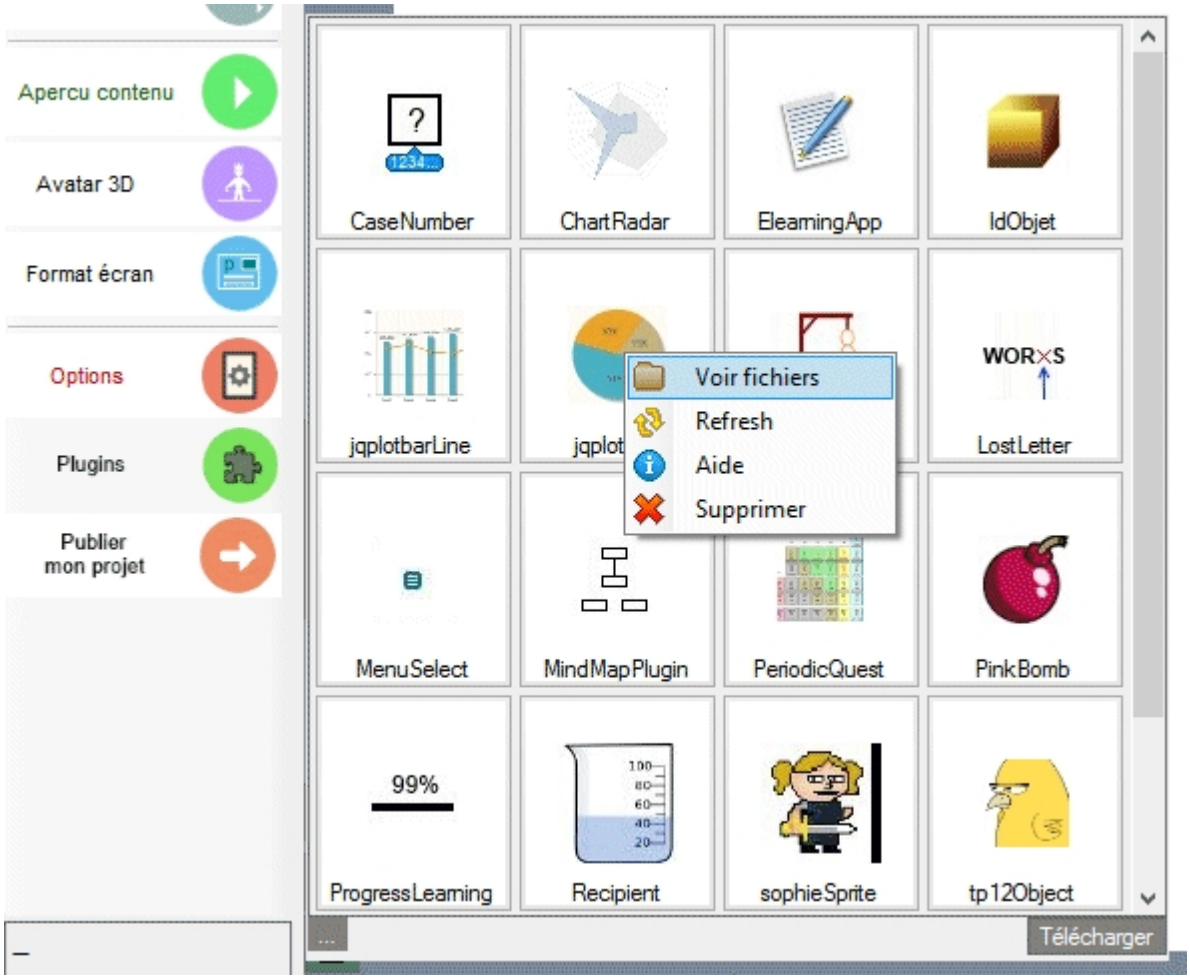
A = Définit le nom du plugin
B = Ce que le redacteur pédagogique pourra faire une fois l'objet inséré dans un écran.
C = Ce qu l'apprenant pourra faire
D = Le type d'objet générer par defaut
E = La taille de l'objet en pixels (attention cette valeur change automatiquement a chaque chargement d'image en G)
F = Chargement d'une icone (l'image est redimensionner automatiquement a la taille 60px par 60px)
G = Chargement d'une icone
H = Parametres du plugins à remplir par le rédacteur (2 dans le générateur 4 maximum en réel)
I = Bouton qui permet de générer le dossier plugin

Modifier un plugin

Pour modifier un plugin dans le navigateur de plugins faire : Clique-droit pour faire apparaître un menu contextuel

Dans ce menu vous avez la possibilité de :

- Voir fichiers (permets d'ouvrir le dossier et d'éditer les fichiers)
- Refresh (recharge tous les fichiers et les plugins avec leurs données, idéal après une modification)
- Aide (permets de consulter la doc sur les scripts et la création de plugins)
- Supprimer (permet de supprimer votre plugin et son dossier, attention irréversible)



Le fichier config.xml

La structure d'un plugin :

Un plug-in est composé de 4 fichiers :

- **Le fichier icon.jpg**

Le fichier icon.jpg est l'image affichée dans Ludiscape il faut de préférence une image de taille 50px par 50px.

- **Le fichier config.xml**

Ce fichier contient les informations de paramétrage du plugin et la définition des fichiers qui vont être intégrés.

Descriptif des balises :

<?xml version="1.0" encoding="UTF-8"?>
<properties>
<data>
<type>**Type objet**</type>
<title>**Titre du plugin**</title>
<screenImage>**Image affiché sur l'écran d'édition**</screenImage>
<files>**Fichiers supplémentaires lié au plugin**</files>
<embeddedFiles>**Fichiers Bibliothèque javascript supplémentaires automatiquement intégrés**</embeddedFiles>

<defautText>**Texte par défaut**</defautText>
<width>**Largeur par défaut en pixels**</width>
<height>**Hauteur par défaut en pixels**</height>

<options>
<resize>**Objet redimensionnable à l'écran**</resize>
<changeImage>**Possibilité de changer l'image**</changeImage>
<changeText>**Possibilité de changer le texte**</changeText>
<conditionalObject>**Objet conditionnel**</conditionalObject>
</options>

<fields>
<label1>**Libellé champs N°1**</label1>
<field1>**Type de données ou texte par défaut**</field1>
</fields>

</data>
</properties>

- **Le fichier run.css**

CSS supplémentaire

- **Le fichier run.js**

Le fichier **run.js** est le moteur principal du plug-in il possède trois fonctions qui permettent d'interagir avec le moteur**JavaScript**.

Les méthodes onPaint () et onZoom () sont appelés respectivement à l' exécution du plugin
onPaint permet de créer le DOM.
onZoom permet de redimensionner les éléments en fonction de la variable zoom écran.

La méthode **isOK**

Dans le cas où il s' agit d' un objet conditionnel, cette fonction renvoi true ou false pour le rendre valide ou non. (voir « page suivante si OK »).

```
function onPaint(obj){

var html = '';
html = '<div id="bloc' + obj.id + '" ';
html = html + ' style="border: solid 1px red;position:absolute;" ';
html = html + ' class="bloc' + obj.id + '" >';
html = html + '</div>';
return html ;
```

```
}
```

Note : *La classe* ***bloc + obj.id*** *est une classe reconnue automatiquement qui redimensionne l'objet proportionnellement au zoom de l'écran.*

```
function onZoom(obj){

var largw = parseInt(parseInt(obj.w) * zoom);
var largh = parseInt(parseInt(obj.h) * zoom);
$(".votreid" + obj.id) .css("height",largh + "px");
$(".votreid" + obj.id) .css("width",largw + "px");

}

function isOK(obj)(


)
```

Exemples de plugins

Bilan personnalisé

Dans certains cas l'objet bilan proposé par ludiscape ne suffit pas a répondre a vos exigences :

Voyons comment on peut personnaliser un bilan à l'aide d'un plug-in :

Pour le graphique nous utilisons la librairie vis.js

Pour le rapport nous utilisons ici la variable BilanXML

Télécharger le plug-in de bilan personnalisé (compatible avec la version 2.84 ou >)

Scorm Ludiscape

La communication SCORM :

Sharable Content Object Reference Model (SCORM) est une spécification de codage permettant de créer des objets pédagogiques structurés.

Visant à répondre à des exigences d'accessibilité, d'adaptabilité, de durabilité, d'interopérabilité et de ré-utilisabilité, les normes du modèle informatique SCORM cherchent à faciliter les échanges entre plates-formes de formation en ligne en maîtrisant l'agrégation de contenu, l'environnement d'exécution et la navigation Internet.

La couche SCORM via Ludiscape :

La couche de communication Ludiscape SCORM à été pensé pour s'adapter de manière agile aux fonctions SCOME disponibles sur le lecteur intégrer dans le LMS ou sur votre intranet.

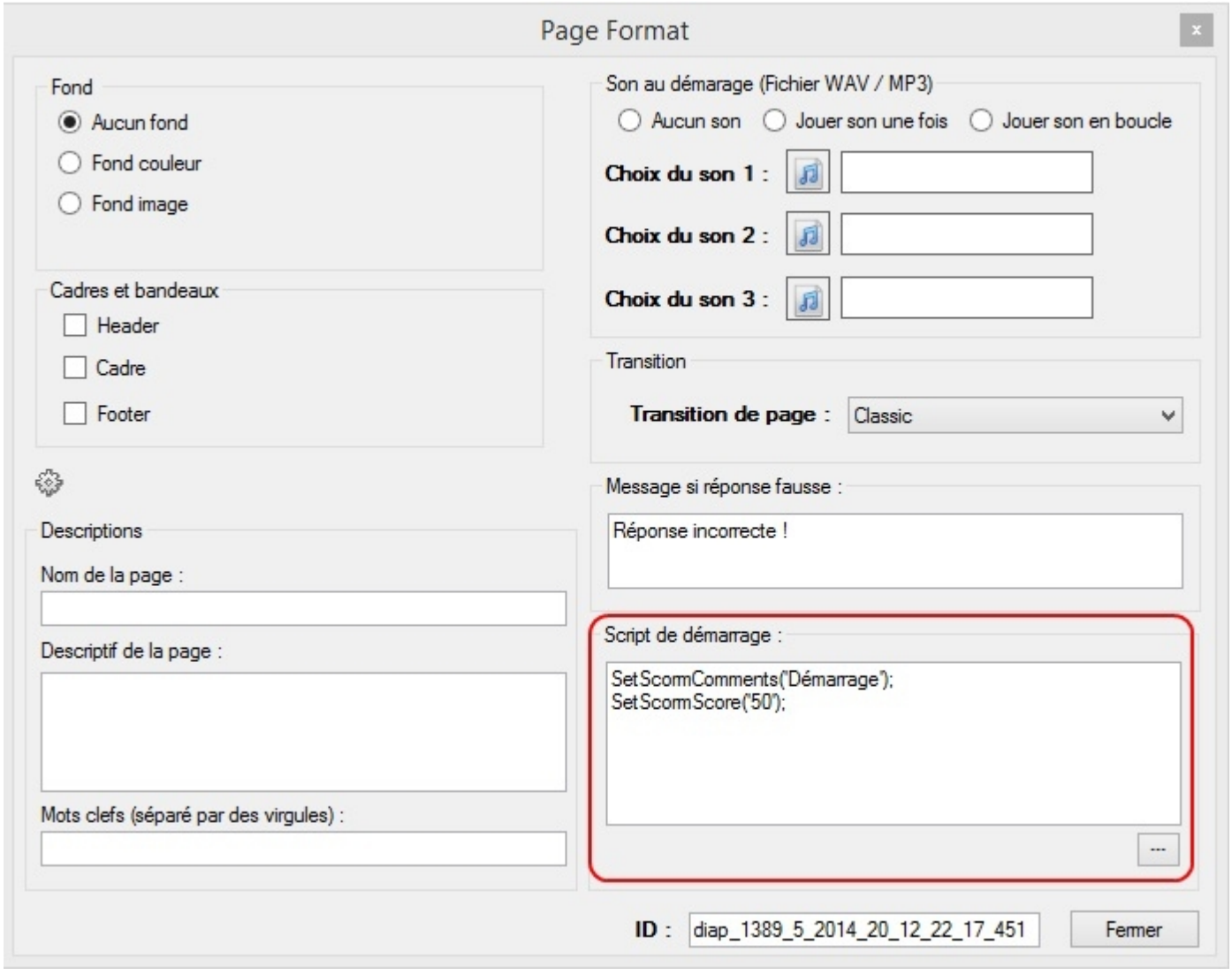
Il vérifie que les fonctions SCORM 1.2 ou 2004 sont implementés dans le lecteur et en fonction du résultat envoi les données.

Le deploiement d'un contenu SCORM se veux donc compatible avec tous le lecteurs SCORM d'aujourd'hui quelque soit la norme utilisée.

Activité partielle d'un apprenant :

Pour détecter l'activité partielle d'un apprenant, il peut être utile d'envoyer regulièrement des informations sur le parcours parcouru par l'utilisateur.

Pour cela vous pouvez appeler les fonctions SCORM ou vos propres fonctions via le script présent au démarrage de la page :



Autres liens :

Créer votre lecteur SCORM :
http://www.batisseurs-numeriques.fr/ludiscapeguide/3816-creer-votre-propre-couche-scorm.html

Le fichier scorm.js :

```
var API = null; /* SCORM API 1.2 & 2004 */

//Log Console
function logconsole(msg){

	if (typeof console === "undefined" || typeof console.log === "undefined"){

	}else{
		console.log(msg)
	}

}

/* Check SCORM API or AlterScorm */
function findAPI(win)
{

	try{

		if (typeof(win.API_1484_11) != "undefined") {
			if(win.API_1484_11!=null){
				API = win.API_1484_11;
				logconsole("FIND win.API_1484_11");
				return true;
			}
		}

		while ((win.API_1484_11 == null) && (win.parent != null) && (win.parent != win))
		{

			var alterwin = win.parent;

			if (typeof(alterwin.API_1484_11) != "undefined") {
				if(alterwin.API_1484_11!=null){
					API = alterwin.API_1484_11;
					logconsole("FIND win.API_1484_11");
					return true;
				}
			}

		}

		while ((win.API == null) && (win.parent != null) && (win.parent != win))
		{
			win = win.parent;
			logconsole("win = win.parent");
		}

		API = win.API;
```

```
	}catch(exception){

		logconsole("findAPI error");
		return false;

	}

}

/* initialize the SCORM API */
function initAPI(win)
{

	logconsole("initAPI start");

	try{

		/* look for the SCORM API up in the frameset */
		findAPI(win);

		/* if we still have not found the API, look at the opener and its frameset */
		if ((API == null) && (win.opener != null))
		{
			findAPI(win.opener);
		}

		logconsole("initAPI end");

	}catch(exception){

		logconsole("findAPI error");
		return false;

	}

}

var ScormSubmitted = false; //use this to check whether LMSFinish has been called later.

function ScormStartCom(){

	initAPI(window);

	if (API != null){

		//SCOMR 1.2
		if (typeof(API.LMSInitialize) != "undefined") {
			API.LMSInitialize('');
			API.LMSSetValue('cmi.core.lesson_status', 'browsed');
			API.LMSSetValue('cmi.core.score.min', 0);
			API.LMSSetValue('cmi.core.score.max', 100);
			API.LMSCommit('');
		}
		//SCORM 2004
		if (typeof(API.Initialize) != "undefined") {
			var r = API.Initialize('');
			if(r==true||r=='true'){
				API.SetValue('cmi.core.lesson_status', 'browsed');
				API.SetValue('cmi.core.score.min', 0);
				API.SetValue('cmi.core.score.max', 100);
				API.Commit('');
				API.SetValue('cmi.lesson_status', 'browsed');
				API.SetValue('cmi.score.min', 0);
				API.SetValue('cmi.score.max', 100);
```

```
                        API.Commit('');
                        logconsole("Initialize ScormStartCom");
                    }else{
                        logconsole("Initialize Error");
                    }

                }
            }

}

function CheckLMSFinish(){
        if (API != null){
            if (ScormSubmitted == false){
                //SCORM 1.2
                if (typeof(API.LMSCommit) != "undefined") {
                    API.LMSCommit('');
                    API.LMSFinish('');

                }
                //SCORM 2004
                if (typeof(API.Terminate) != "undefined") {
                    API.Commit('');
                    API.Terminate('');
                    logconsole("Terminate CheckLMSFinish");
                }
                ScormSubmitted = true;
            }
        }
}

function CheckLMSLearnerName(){

        if (API != null){
            //SCORM 2004
            if (typeof(API.data.learner_name) != "undefined") {
                return API.data.learner_name;
            }
        }
        return '';
}


function SetScormIncomplete(){
        if (ScormSubmitted == true){
            return;
        }
        SetScormScore();
        if (API != null){
            //SCORM 1.2
            if (typeof(API.LMSSetValue) != "undefined") {
                API.LMSSetValue('cmi.core.lesson_status', 'incomplete');
                API.LMSSetValue('cmi.core.session_time', MillisecondsToTime((new Date()).getTime() -
ScormStartTime));
                API.LMSCommit('');
            }
            //SCORM 2004
            if (typeof(API.Terminate) != "undefined") {
                API.SetValue('cmi.core.lesson_status', 'incomplete');
                API.SetValue('cmi.core.session_time', MillisecondsToTime((new Date()).getTime() -
ScormStartTime));
                API.SetValue('cmi.lesson_status', 'incomplete');
                API.SetValue('cmi.session_time', MillisecondsToTime((new Date()).getTime() - ScormStartTime));
```

```
                        API.Commit('');
                }
        }
}

var isScormFinish = false;

function SetScormComplete(){

        if(isScormFinish==false){

                if (API != null){

                        //SCORM 1.2
                        if (typeof(API.LMSSetValue) != "undefined") {
                                API.LMSSetValue('cmi.core.session_time', MillisecondsToTime((new Date()).getTime() -
ScormStartTime));
                                API.LMSSetValue('cmi.core.lesson_status', 'completed');
                                SetScormScore();
                                API.LMSCommit('');
                                API.LMSFinish('');
                                isScormFinish = true;
                        }

                        //SCORM 2004
                        if (typeof(API.Terminate) != "undefined") {
                                var timefull = MillisecondsToTime2004((new Date()).getTime() - ScormStartTime);
                                API.SetValue('cmi.core.session_time', timefull);
                                API.SetValue('cmi.session_time', timefull);
                                logconsole("session_time = " + timefull);
                                API.SetValue('cmi.core.lesson_status', 'completed');
                                API.SetValue('cmi.lesson_status', 'completed');
                                API.SetValue('cmi.completion_status', 'completed');
                                SetScormScore();
                                API.Commit('');
                                API.Terminate('');
                                logconsole("Terminate SetScormComplete");
                                isScormFinish = true;
                        }
                        ScormSubmitted = true;

                }

        }

}

var ScormStartTime = (new Date()).getTime();
var SuspendData = '';

function SetScormTimedOut(){
        if (API != null){
                if (ScormSubmitted == false){

                        //SCORM 1.2
                        if (typeof(API.LMSSetValue) != "undefined") {
                                SetScormScore();
                                API.LMSSetValue('cmi.core.exit', 'time-out');
                                API.LMSCommit('');
                                CheckLMSFinish();
                        }
                        //SCORM 2004
                        if (typeof(API.Terminate) != "undefined") {
                                SetScormScore();
                                API.SetValue('cmi.core.exit', 'time-out');
```

```
                                API.SetValue('cmi.exit', 'time-out');
                                API.Commit('');
                                API.Terminate('');
                        }

                }
        }
}

function SetScormComments(m){
        if (API != null){
                if (ScormSubmitted == false){

                        //SCORM 1.2
                        if (typeof(API.LMSSetValue) != "undefined") {
                                API.LMSSetValue('cmi.comments', m);
                                API.LMSCommit('');
                        }
                        //SCORM 2004
                        if (typeof(API.Terminate) != "undefined") {
                                API.SetValue('cmi.comments', m);
                                API.Commit('');
                        }

                }
        }
}

//TIME RENDERING FUNCTION
function MillisecondsToTime(Seconds){
        Seconds = Math.round(Seconds/1000);
        var S = Seconds % 60;
        Seconds -= S;
        if (S < 10){S = '0' + S;}
        var M = (Seconds / 60) % 60;
        if (M < 10){M = '0' + M;}
        var H = Math.floor(Seconds / 3600);
        if (H < 10){H = '0' + H;}
        return H + ':' + M + ':' + S;
}

//TIME RENDERING FUNCTION
function MillisecondsToTime2004(Seconds){
        Seconds = Math.round(Seconds/1000);
        var S = Seconds % 60;
        Seconds -= S;
        if (S < 10){S = '0' + S;}
        var M = (Seconds / 60) % 60;
        if (M < 10){M = '0' + M;}
        var H = Math.floor(Seconds / 3600);
        if (H < 10){H = '0' + H;}
        return 'PT' + H + 'H' + M + 'M' + S + 'S';
}

//SetScormScore
function SetScormScore(score){
        if (score != null){
                if (API != null){
                        //SCORM 1.2
                        if (typeof(API.LMSSetValue) != "undefined") {
                                API.LMSSetValue('cmi.core.score.raw', score);
                        }
                        //SCORM 2004
                        if (typeof(API.Terminate) != "undefined") {
                                API.SetValue('cmi.core.score.raw', score);
```

```
                                API.SetValue('cmi.score.raw', score);
                                API.SetValue('cmi.score.scaled', score/100);
                        }
                }
                logconsole("SetScormScore " + score);
        }
}
```
