New player type integration.

1. Integrate players with the help of swfobject or with standard ways:

1.1. Example of YouTube player integration with swfobject:

var params = {allowScriptAccess: "always", wmode: "opaque"};

var atts = {id: "ytplayer"}; swfobject.embedSWF("http://www.youtube.com/v/video\_id?enablejsapi=1&controls=0&showinfo=0&playerapiid=**ytplayerDivId**", "**ytapiplayer**", "840", "356", "8", null, null, params, atts);

1.2. Example of JWPlayer integration with standard way:

jwplayer('**jwplayerDivId**').setup({

file: '/path/to//file.mp4',

primary: "flash",

width: '792',

height: '480'

});

JWPlayer also can be integrated with swfobject, the same way as in YouTube example.

2. Setup type and ID of player in parameters of the plugin:

2.1.

jQuery('#**ytapiplayer**').voicebase({

playerId: '**ytplayerDivId**',

playerType: 'youtube',

...

});

2.2.

jQuery('#**jwplayerDivId**).voicebase({

playerId: '**jwplayerDivId**',

playerType: 'jwplayer',

…

});

3.Method VB.init(VB.settings.playerId) is called during the initializing. ID of player is passed in method.

VB.init(VB.settings.playerId);

Instance of new player new VB.Player() is created inside of this method.

create\_instance: function(c, b) {

           this.instance\_index = c,

                   this.player = new VB.Player(this, b);

       }

Type of the player passed in settings is defined there, then interface of the player type is launched:

this.find\_player\_interval = setInterval(function() {

           try {

d.interface = new VB.interface[h](b, d.instance);

           } catch (f) {}

}, 250);

The function is implemented with 0.25 seconds interval, until interface is not instantiated according to the indicated player.

Interfaces are stack of methods to the types of players:

VB.interface = {

       youtube: function(a, b) {...},

       jwplayer: function(a, b) {...}

}

Here are the details for youtube interface:

youtube: function(a, b) {

//s of instances declared

           var c = this;

           c.instance = b;

// Determination and search of player by ID and appropriation of the interface for following usage to local variable

           if (document.getElementById(a) && document.getElementById(a).tagName.toLowerCase() == "iframe") {

               if (typeof YT == "undefined" || typeof YT.Player == "undefined")

                   throw "not\_ready";

               if (!YT.loaded)

                   throw "not\_ready";

               if (c.instance.ytplayerloaded)

                   return !1;

// If you are at the end of this point – it means that the player is found, initialized and instance is being created.

               c.youtube\_player = new YT.Player(a), c.instance.ytplayerloaded = !0

           } else {

// Or playes is seeked by DOM-identifier

               this.youtube\_player = window.document[a];

           }

// Play function of Youtube player

           this.play = function() {

               this.youtube\_player.playVideo();

           },

// Pause function of Youtube player

this.pause = function() {

               this.youtube\_player.pauseVideo();

           },

// Function of getting the current status (playing|paused)

this.play\_state = function() {

               try {

                   var a = this.youtube\_player.getPlayerState()

               } catch (b) {

                   return "PAUSED"

               }

               return parseInt(a) == 1 || parseInt(a) == 5 ? "PLAYING" : "PAUSED"

           },

// Getting of the current position

this.position = function() {

               try {

       // 0.07 – Shift of position. It can be different for each player.

       // Depending on callback of player

                   return this.youtube\_player.getCurrentTime() + 0.07;

               } catch (b) {

                   return 0;

               }

           },

// Getting of uploaded to player media-file duration

// We don’t use this function, we take media-file duration from getFileMetaData method

this.duration = function() {

               return this.youtube\_player ? parseInt(this.youtube\_player.getDuration() + .5) : !1

           },

// Rewind to needed position to 1/10

this.seek = function(a) {

               var a = parseInt(a, 10);

               this.youtube\_player.seekTo(a);

           },

// getting of volume level (0-100)

this.get\_volume = function() {

               return this.youtube\_player ? this.youtube\_player.getVolume() : !1;

           },

// Setup of volume level (0-100)

this.set\_volume = function(a) {

               this.youtube\_player.setVolume(a);

           }

       }

Methods for other players were made the same way.

VB.interface = {

youtube: function(a, b) {...},

jwplayer: function(a, b) {...},

newplayer: function(a, b) {

var c = this;

           c.instance = b;

           …

this.play = function() {

this.newplayer.play();

           }, this.pause = function() {

this.newplayer.pause();

           },

...

}

}

Pay attention to method of wrapping of player block in method VB.api.setMetaData:

if (VB.settings.playerType == 'jwplayer' && $('#' + VB.settings.playerId).is('object')) {

               $('#' + VB.settings.playerId).parent().wrap('<div class="vbs-player-wrapper vbs-' + VB.helper.randId() + '"></div>');

           } else {

               $('#' + VB.settings.playerId).wrap('<div class="vbs-player-wrapper vbs-' + VB.helper.randId() + '"></div>');

           }

Wrapping to new div is required for hiding player with audio-mode and for entering header with video-mode. JWPlayer additionally wraps object element to parental div during flash-initializing. It can cause wrong recognition of player element and wrong wrapping.