

# Sonettic-Cinema

## **Advanced JavaScript API**

© 2009 Sonettic

[www.sonettic-cinema.com](http://www.sonettic-cinema.com)

# Advanced JavaScript API Design

Advanced JavaScript API design based on Event model (Observer design pattern). Player and JavaScript communicate through events and all events either from Player or from JavaScript may contain arguments. List of all available events and its arguments in [Player Events document](#).

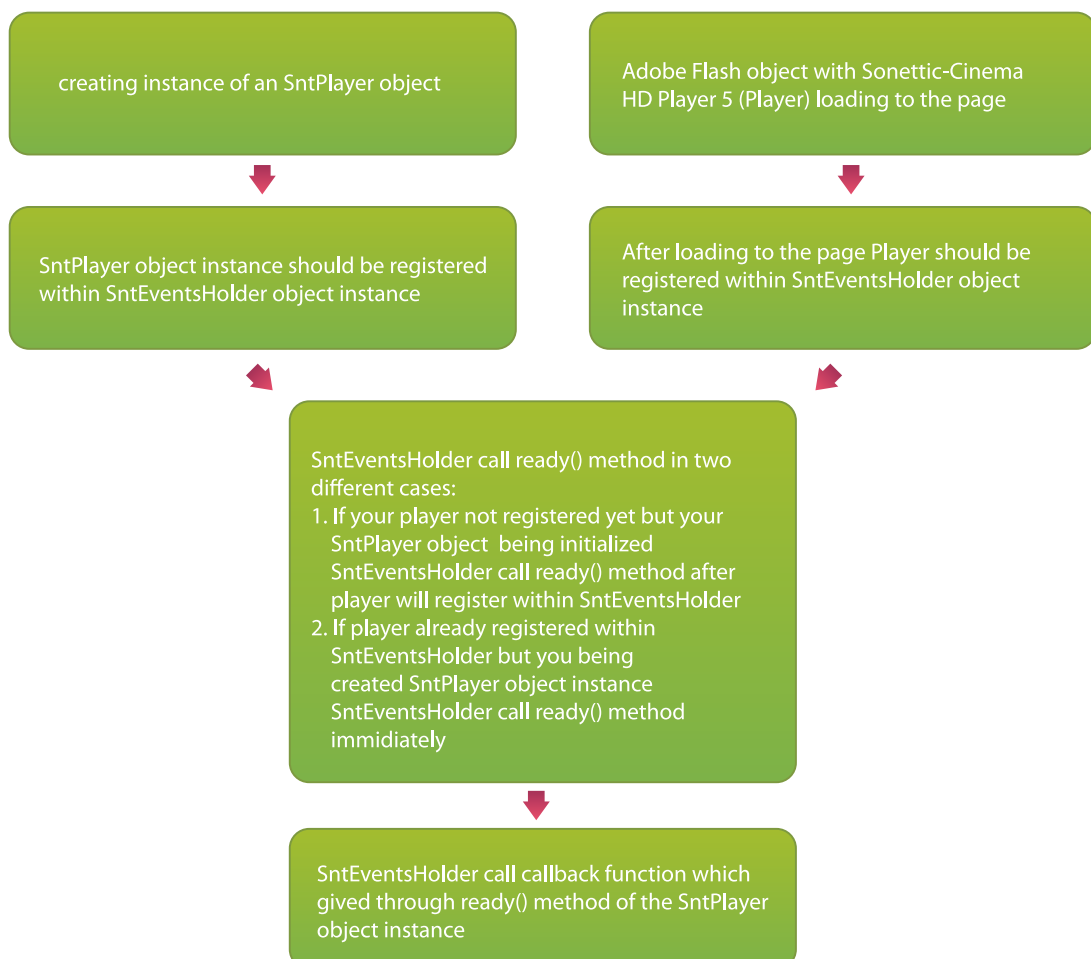
## Events Holder

Player and Javascript communicate through SntEventsHolder object instance. This object is internal and it's not necessary to know how it work in details.

Logic of the Player workflow is very simple. To receive events from Player Flash Object with Player invoke anonymous function **SntReceiveMessage**. To send event to the Player JavaScript invoke a **receiveMessage** function of the Player Flash Object. All of events identify through PlayerID. The instance of SntEventsHolder object create during webpage loading.

## Ready callback function

Method **ready** of **SntPlayer** object with necessary argument **callback function** will be invoked after initialization of Flash Object with Player on the webpage. Adding of new **event listeners** or sending **events** to the Player have to be **inside** of callback function. If you add new events listeners or send it inside callback function then it will be guarantee that your events will be sent to the player and will be received from it.



## New Player's JavaScript holder

To create new JavaScript holder for the Player it is necessary to add flashvar playerId to the player with Flash Object ID.

In this document I will use helper JavaScript function to get DOM Element from DOM Tree:

```
// helper function
function $$(id){ return document.getElementById(id) || null; }
```

## Example of HTML and JavaScript

HTML sample code:

```
<object
  classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,40,0"
  width="470" height="264" id="player2videoholder">
  <param name="movie" value="cinemaplayerdev.swf" />
  <param name="quality" value="high" />
  <param name="bgcolor" value="transparent" />
  <param name="wmode" value="transparent" />
  <param name="allowFullScreen" value="true" />
  <param name="allowScriptAccess" value="sameDomain" />
  <param name="flashvars"
    value="playerID=player2videoholder&content=video.flv&slavemode=true" />

  <embed src="cinemaplayerdev.swf" quality="high"
    bgcolor="transparent"
    flashvars="playerID=player2videoholder&content=video.flv&slavemode=true"
    width="470" height="264" name="player2videoholder"
    align="center" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer"
    allowScriptAccess="sameDomain"
    allowFullScreen="true"
  ></embed>
</object>
```

Initialize JavaScript Player instance:

```
<script type="text/javascript">

// Flash Object ID
var videoHolderName = 'player2holder';

// create SntPlayer object instance
var player = new SntPlayer(videoHolderName);

// here we set player callback function
player.ready(function(r){
  // write your code here...
});
</script>
```

Ok, our player object have been initialized. Now we can send and receive events.

## Send events to the player

Ok, here we add one control button and one event inside of **player.ready** callback function.

HTML Code:

```
<input type="button" name="play" id="play-button" value="play" />
```

JavaScript code:

```
// play event - send play event to the player
var playEvent = function(){
    player.sendMessage('uiPlayClick');
};
```

This code send one event to player – it's play a video.

## Receive events from the player

To receive messages from the Player we have to add event listener and write a function to handle event.

Lets try to create play/pause control with same button.

HTML Code from previous sample:

```
<input type="button" name="play" id="play-button" value="play" />
```

JavaScript code:

```
// get control object
var play_button = $$('play-button');

// play event - send play event to the player
var playEvent = function(){
    player.sendMessage('uiPlayClick');
};

// pause event - send pause event to the player
var pauseEvent = function(){
    player.sendMessage('uiPauseClick');
};

// play event listener
var onPlayEvent = function(){
    // change text on the control
    play_button.value = 'pause';

    // change click handler
    play_button.onclick = pauseEvent;
};

// pause event listener
var onPauseEvent = function(){
    // change text on the control
    play_button.value = 'play';

    // change click handler
    play_button.onclick = playEvent;
};

// add event listeners to the player
```

```
player.addEventListener('mediaPlay', onPlayEvent);
player.addEventListener(['mediaPause', 'mediaStopStartPosition'], onPauseEvent);

// add event handler to our controls
play_button.onclick = playEvent;
```

Here we have been added few event listeners – **mediaPlay**, **mediaPause** and **mediaStopStartPosition**. With **mediaPlay** and **mediaPause** it is clear. Event **mediaStopStartPosition** will be send by Player after it is initialized. It's for creating preview of the video.

For all of the events we create handler functions and have added it with **player.addEventListener**. It work very simple – you add callback to event listener and when event is sent by player **SntEventsHolder** invoke your callback function.

## Receive and send events with arguments

In this chapter we using sample html and javascript code from previous chapter and supplement it by new JavaScript and HTML code.

Add text field to HTML Code:

```
<input type="text" size="6" name="time" id="play-time" value="00:00" />
```

JavaScript code:

```
var play_time = $$('play-time');

// media state listener - current playing position
var onStateChange = function(s){
    // round position
    var position = Math.ceil(s.position);

    // show video position in text field
    play_time.value = position;
};

// add event to receive video status every second
player.addEventListener('mediaStateChange', onStateChange);
```