Register Callback Event

Events are registered with an event name, a callback function, and an optional priority. The higher the priority, the earlier the callback will be called.

> ⚠ Please note that the following functions can be performed only after you have successfully undertaken the first steps (see the "Getting started" chapter).

After you have connected to the player, you can register or remove the callback using the following functions:

Register Callback Event:

```
Player.registerEventListener(eventName, function(event) {...}, priority = 0);
```

Remove Callback Event:

```
Player.unregisterEventListener(eventName, functionName, priority = 0);
```

Parameter	Description
eventName	Name of the event. Currently all known HTML5 media events are supported (for a list of all events, for example, see the following link: Media-Events). We also provide a couple of additional events (see the "Supported Events" chapter).
callback	A function that should be called when the event is triggered. This can be either the name of the function or the function definition itself. Set up the callback function to receive an event object, this may provide useful data specific to the type of event used. ⓘ Note that if you intend to use "unregisterEventListener", the function must be externally defined (as opposed to an anonymous function as shown in the "registerEventListener" example above).
priority	An optional argument that sets the priority of the event (defaults to 0). The higher the value, the earlier the callback will be called.

The example code below shows you how to register callback events that can be used to trigger actions in the JavaScript API. It will use the "playing" and "pause" events to show an element only while the video is playing.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div id="player_container"></div>
<div id="while_playing" style="display: none">video is playing!</div>

<script type="text/javascript">

    function playingCallback(event) {
            var whilePlaying = document.getElementById('while_playing');
        whilePlaying.style.display = "block";
    }

    function pauseCallback(event) {
            var whilePlaying = document.getElementById('while_playing');
        whilePlaying.style.display = "none";
    }

    //create a new player instance (vmpro)
    function createPlayer() {
        var initPlayer = {
            success: function (playerApi) {
                playerApi.registerEventListener('playing', playingCallback);
                playerApi.registerEventListener('pause', pauseCallback);
            },
            parameters: {
                configType: 'vmpro',
                playerId: '<PlayerID>',
                videoId: '<VideoID>',
                apiUrl: '//d.video-cdn.net/play',
                flashPath: '//e.video-cdn.net/v2/'
            }
        };

        VideoPlayer.Collection.addPlayerById('player_container', initPlayer);
    }

    //add the player
    var head = document.getElementsByTagName('head')[0];
    var script = document.createElement('script');
    script.type = 'text/javascript';
    script.src = '//e.video-cdn.net/v2/embed.js';
    head.appendChild(script);

    script.onload = (function () {
        createPlayer();
    });

</script>

</body>
</html>
```

The first thing to notice here is the addition of the two "registerEventListener" functions on lines 26 and 27. These let the JavaScript API know that as soon as the "playing" and "pause" events occur, the indicated functions should be called. These functions find the element ("while_playing") and show or hide it depending on the event (lines 12-20).