Onload Callback Function

This tutorial shows you how to implement a callback function that allows you to execute code as soon as a Player loads. The function can contain several actions that are called when the API of the respective Player loads. The code in this tutorial introduces our recommended approach for dynamically adding a player on load.

> ⚠ Please note that when lazy loading is enabled, the Player will not be accessible through the API until a user clicks play. This is an inherent quality of the lazy loading feature, which prevents the Player from loading until a user clicks play. Find more information on lazy loading here.

The following code shows our recommended approach for adding a player on load. This example will play and mute the video as soon as the Player loads:

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

<div id="player_container"></div>

<script type="text/javascript">

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

        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>
```

> ⚠ Please note that you need to specify necessary IDs and URLs.

Note the addition of the "success" property (line 14). This should define the function you'd like to execute as soon as the Player loads (the callback function). By adding this to your "initPlayer" object and passing it into the second argument of the "addPlayerById" function (line 28), a Player object will be sent to your callback function, which you can then use to make API calls.

> ⓘ For a list of other API functions available, see the Basic Functions Reference.

Adding the Player Script

First take note of the way that the Player script has been added in this example. The code above adds the "embed.js" file by appending it to the "head" tag dynamically with JavaScript (lines 31-35). We have found this to be the best way to add the "embed.js" file because it facilitates correct timing and order of operations by allowing you to use the script's "onload" property to initiate the creation of your Player (line 37).

Creating the Player

Also note the "createPlayer" function (line 12), which is called when the script loads. This function creates an "initPlayer" object to pass into the API's "addPlayerById" function. The "parameters" property should use the attributes from your embed code. This allows you to set up an empty div, "player_container" as a placeholder and add these properties dynamically when the player is created.