**DwBP - 5 - Using the Player Catalog**

[TONE]

In the last video, we built a great foundation on which to continue our development with Bright Cove Player. In this video, we're going to use the Player Catalog to change the video that is currently playing in the Player. So based upon a user interaction-- in this case, a user clicking on a button-- we're going to use a couple of methods to get a video and then load that video into the Player. Let's go ahead and get started.

Here you see the code that we left with at the end of the last video. I have done a couple things. I have a new video playing, which is fine. And I've also removed the time update, because that was an example to show you how time update worked. Not necessary for this.

One of the things we're going to need to do is access the Player throughout all of our JavaScript code. So the first thing is I have to change how our myPlayer variable is defined. So I'm going to come in here and say var myPlayer. Won't assign it anything there, because I want to assign it down here. So I'm going to take the var off. So now myPlayer is going to have meaning throughout our whole script block here.

I want this to happen on user interaction, the change of the video. So I'm going to put in a button. And you don't need to see me type HTML, so I'm just going to insert the button. And here you see we're defining a button. The clickable text on the button's going to say changeVideo. And it's going to call an Event Handler called changeVideo. So we're going to click, and we need to have the changeVideo method called.

So that's what we're going to do. And the heart of the code in this session is we're going to go through, and we're going to create this function.

So let's say function changeVideo. And what we want to do in here is we want to use the Player Catalog and call the getVideo method. So we know that myPlayer now has meaning throughout the whole script block. And we're going to use the Catalog. WE just say catalog., and the method is getVideo.

Now, getVideo has two arguments. The first ones is the video ID we're going to get, and the second one is a callback function. So let's put in an ID. And I've just copied an ID here. And this is the ID of a video from the proper account. And then we're going to use an anonymous function again. So I'm going to say function.

And two parameters are passed to this function automatically. The first one is an error, which I'm hoping will just see that it's null. And the second is videoObject. So there's our two arguments that are passed in.

And now what we want to do in this callback function-- and of course, the callback function is only called once the video is returned to the client. We want to go through, and we're going to display the error to the console. And hopefully, again, that'll be null. Then we're going to load that videoObject that's returned into the Player, and then we'll play the videos.

All right. So let's do a log. And we're going to just have the string error, and then will display the errorObject. And again, myPlayer.catalog. And now we're going to use the Load method. And what we want to Load is that videoObject that was returned from the getVideo method. And finally, we're going to Play the video that was returned.

I'm going to save that, and then we're going to go see if it works.

We'll come out here. We're working on Task2.html.

So here's one video playing. I'm going to click the Change Video button. And there is the new video. OK. Pause that.

All right. Nice and quick. We have seen how to use the Video Catalog. And what we wanted to do is change the video that is playing in the video based on the user interaction. In this case, we clicked the Change Video button. The user clicked the Change Video button.

Now I want to take a moment to review what we did with a couple of slides. And here we see the explanation of the Player Catalog. It's basically a helper library for making requests to the Video Cloud Catalog.

And you see at the bottom there, we've used two of those methods. We used getVideo. And notice, uses an ID and a callback function. And we used the Load method. And as you remember, we passed in the videoObject that was returned by getVideo.

Also, one method that's part of the Catalog that we did not use is the getPlaylist. You can also pass in a playlist ID, and then you'll have a callback function to act upon, do whatever you need, when that playlist is returned.

Here you see the object that's actually returned from the getVideo. We didn't look at this in the browser. But if you look down, you'll see the response and the response text. That's actually the information about the video that's brought in. And you can see the actual type is an XML HTTP request object that is returned.

We've now completed our second task. And what we did is we used the Catalog object to be able to dynamically load and play a video that was not originally in the Player.

In our next video, we're going to use metadata-- information about the video-- in our HTML page . So let's get started. See you there.

[TONE]