**DwBP - 5 - Getting Started with Brightcove Player Development**

We’ve seen the demo, and learned some key JavaScript concepts, so now we are going to start looking at code in detail. The use case we are writing code for is to programmatically play the video. We know there are other ways to start playing the video on page load, like clicking Yes to our autostart in Studio, but this is a great exercise on which to get started. At a bit more technical level, what we're going to do is use an event handler for the loadedmetadata event as the place to write code to make the video begin playing.

In the Players module in Studio, we’ll create a new player and name it. We see details for this player by clicking on the name. We'll be looking at some of these options in later videos.

Let's publish a video in our new player. I will go to the Media module, select the video, then click Publish an Embed. It is easy to forget the player with which you want to publish the video, so choose the correct player. Choose the desired settings for the player, then copy the advanced code and go to the editor.

First, we should notice the structure of the student files you can download. What you'll see in the student download for every task are three files. So here's your starting file, which in this case is just a blank HTML page. I'm going to paste the player code into here. The next file details the steps you need to perform to complete the task on which we are working. The steps are provided in case you would like to redo writing the code on your own. The last file for each task is a solution, which is the finished product after completing the steps. In summary, for each task there's a starting file, the steps to accomplish the task, and then the result from following those steps.

Let's go back to our starting file and examine the code we pasted into our blank HTML page. We see the video-js tag and also the script tag. The following are key attributes in the video-js tag:

* the video account
* the player id/name from Studio
* the attribute to ensure the controls are shown
* the video to play
* video width and height

The other attributes are not vital to us at this point.

The script that is loaded is created as part of the player creation process and is unique to each player. For instance, if you have changed anything in the Player Properties page, those changes would be reflected in the script file, including styling changes. The Brightcove Player API is also part of the minified JavaScript file.

To be able to reference the video-js tag we need to add an ID attribute, and assign it the value myPlayerID.

Let's get started by first inserting a script block. In the script block we’ll wait for the Brightcove Player to be loaded on the page before trying to communicate with it, utilizing the getPlayer() and ready() methods.

We are going to get a reference to the player element using the logical name of myPlayer and assigning it the THIS keyword, which in the context of the getPlayer() and ready() methods is the player itself.

Browsers no longer allow autoplay of a video with an audible audio track, so using the muted() method will allow the video to play without a user interaction.

We are sure the player has loaded, but that does NOT automatically mean the video in the player is ready to play. To wait for that, we will use the on() method to wait for the **loadedmetadata** event.

Finally, inside the event handler for the the **loadedmetadata** event, let’s start the video using the play() method.

Now we're ready to test, so let's save and browse our page. As you can see, the video's playing, so we know our code is successful.

Before we move on, let's use another event for more practice-- in this case, the time update event. This event is dispatched a number of times every second when the video is playing, depending on the playback technology in use. In other words, it dispatches often. Also, we want to practice code using the on() method with a function declaration. Previously, all of our examples use an anonymous function. In the on() method, we'll call an event handler function named showUpdate.

All we'll do in the event handler is log the event object passed as a parameter to the event handler. Note that the function is defined after its actual use, which is fine for function declarations. Let's go watch the new event handler in action.

So I've temporarily commented out the play() method so I can start it manually. And what you want to watch is down here in this console, watch this scrollbar fill up with the events as they appear in the console when I click Play. And there you can see it keeps shrinking and shrinking. That means there's a whole bunch of events down here.

Now we are going to use some slides for a quick review and to look at some other code we'll use later. First the use of the getPlayer() and ready() methods provide the code to wait for the player to be properly loaded before we try to interact with it.

Next are two methods we’ve used. Here are the API docs for those methods.

We're already experts on handling events. We've done it a few times already. But here's a bit more information on setting up event handling. We used the on() method in the code we've written. This handles the event every time the event is dispatched for the life of the application. You might want to use the one() method, which handles an event only once. If, in your application logic, you want to handle an event until some threshold is met, you can use the off() method to remove the event listener.

Here's a summary of what we did and the code we actually wrote for the task. We're using the on() method to listen for the loaded metadata event and using anonymous function to play the video.

We finished our first coding task, and now have the foundational knowledge of how to programmatically interact with Brightcove Player. In the next video, we will use the player catalog to allow a user interaction to change the video that is currently playing in the player. See you there.