**DwBP - 4 - Getting Started with Brightcove Player Development**

Up to now, we have been laying the foundation on which to build our development with Brightcove Player knowledge. We are now going to start coding. 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 loaded metadata event as the place to write code to make the video begin playing. So let's get started.

I'm browsing the Video Cloud Studio. In the Players module, I'm going to create a new player and name it. I can 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 on our new player. I will go to the Media module, select the video, then click Publish. It is easy to forget the player with which you want to publish the video, so choose the correct player. I'm going to copy the advanced code, which is the non-iframe implementation, then go to my editor.

First, we should notice the student files you can download if you wish. 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 tag and also the script tag. The video tags contains attributes for the video to play, the video account we are using, the player we're using. I'm going to jump to the class, and we can use this when styling the player. The controls acts as a toggle to show the controls in the player.

I skipped over the data-embed. This has to do with the Brightcove Player feature called Embed, which lets you create parent-child relationships with players. This feature is not covered in the course. There are numerous docs on this subject if you're interested.

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. I'm going to add an id attribute to the video tag. Later in JavaScript, we'll need an ID so we can get a reference to the video element. I'm going to use "myPlayerID" for the value.

Now, what we want to do is call the play() method to play the video when the page loads. Let's get started by first inserting a script block. And the script block will use a videojs method to change the generic video element into a Brightcove player. By the way, the videojs method is part of the imported JavaScript file.

Before we use the play() method, we have to ensure the video was loaded and ready to play. Again, remember we are working in an event-driven environment, so we will wait for the loaded metadata event. After this event is dispatched, we know we can play the video. We'll use the on method to associate a dispatched event with an anonymous callback function that is the event handler.

We will want to interact with the player in the event handler, so first we'll get a reference to the player by creating a variable called myPlayer. We will assign that player the value of this, which is a reference to the video object. At this point, we want to play the video. Let's use a variable created as a reference to the player and 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. The video is quite small. It defaults to 300 pixels by 150 pixels. So let's go back to our code and style the player to be larger.

In the head section of the HTML page, we'll insert a style block utilizing the class that is automatically part of the player tag, video-js, and we'll set a new width and height. Great. Another test now to be sure it worked. All right. That's looking good, and we have successfully coded to complete our first use case.

Before we move on, let's use another event for more practice-- in this case, the time update event. This event is dispatched every 15 to 250 milliseconds when the video is playing, depending on the playback technology in use. In other words, it dispatches really 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. Part of the code shown here we have seen, and that is creating a variable to hold a reference to the player. We are using a different event and slightly different syntax to code that variable assignment. It is so common to need to wait for the player to be ready to interact with. Special syntax was written just for that. That special syntax uses the ready() method, which under the hood could also be thought of as a special kind of event. An anonymous function is used to house the code for the event handler.

You might be thinking, why didn't we use the ready() method instead of waiting for the loaded metadata of event? This is a very crucial point to understand. The reason is ready waits for the player to be interact with, whereas loaded metadata means the video in the player is ready to interact with. Interestingly, if you use ready() and try the play() method, it might actually work, depending on how long it took the video object to download and also what browser you're using. But of course, we can't write code that might work under special circumstances.

As we've seen, the play() method is used to play video in the player. We're already experts on handling events. We've done it twice already. But here's a bit more information on setting up event handling. We use the on() method in the code we've written. This handles the event every time we [INAUDIBLE] [? dispatch ?] 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.

Well, 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.