**DwBP - 4 - Using JavaScript with Brightcove Player**

[Intro slide]

There are a few JavaScript concepts we need to understand before we jump into development. These are key tools from JavaScript that are used frequently in Brightcove Player development. Please know that if you have an understanding of callback functions and event driven development framework, you really do not need to watch this video and move onto the next one.

[Event driven slide]

The Brightcove Player API is an event driven framework. As we can see by the definition, this means that the program behavior is driven by the production, detection, and consumption of events. In simpler terms, everything that happens with Brightcove Player takes place because an event is dispatched, handled, and associated event handler code is then executed.

In the demo we saw the ready event, when the player is ready to communicate with, and the loadedmetadata event, when the player is ready to play, used. When these events dispatched code was executed.

Program execution behavior is not as we see it on the bottom left. Where line 1, line 2, line 3, et cetera are executed in order.

Actual program execution is shown on the bottom right. Where an event, like when the player is ready or the video starts playing, is dispatched. And then an event handler function is called and code is executed.

[Callback functions slide]

Another thing that we have to understand are callback functions. This is a standard JavaScript technique and not something particular to Brightcove.

In the demo, a callback function was used with the on() method to wait for the loadedmetadata event to be dispatched. A callback function was used, and the code to play the video was not executed until the function finished its job. It may be instructive to know that event handler functions are a type of callback function.

It's a really a good thing we have this callback functionality the player engineers gave us to use in JavaScript. And here's why.

Consider the method called getVideo(). I could ask the question, “how long is it going to take to get the requested video back to your computer device after the method is called?” This can be a confusing question because there is not a single answer. The answer is, of course, it depends. And that is exactly why we have to have a callback function.

So what happens when we do call getVideo()? The method sends a request to a Brightcove server for a video object. At some point, the data will be returned. ONLY at this point, when the video object is returned, will the callback function be called.

By the way, there is a default timeout of 45 seconds before an error is displayed if the video is not returned.

[Function implementation slide]

I don't want to go too deep into the messy details, but callback functions need to be thoroughly understood as you will use them often when developing with Brightcove player. We need to decide how to write callback functions. There are three options. As an anonymous function, as a function declaration, and as a function expression. The slide shows us the look of each implementation. You will find you use mostly anonymous functions in Brightcove Player development.

As a quick review of the most important concepts from this section of the course, remember:

* Brightcove Player API is event driven
* Callback function’s argument (function in parentheses) is not called until the callback function’s job is finished

Now that we've built a good foundation on some key JavaScript concepts, let's do some more coding.