**DwBP - 7 - Displaying Video Information From the mediainfo Property**

In the last video, we retrieved the video object from the catalog and used it to replace the video that was in the player with the new one that we retrieved. In this video, what we're going to do is retrieve metadata about the video that is currently in the player and then use that somewhere in the HTML page. In this case, we're going to dynamically build some HTML and injected into and HTML div element.

The video metadata is utilized in a number of ways. From our samples page you can see metadata used as:

\*Dynamic content in an end screen  
\*Dynamic URL in a call to action  
\*Dynamic video explanations in an overlay

The great thing about the video metadata is that, of course, it changes for each video. So by using it you get customized, dynamic information to utilize in JavaScript.

Let’s look at the solution so we know exactly what we are trying to accomplish in this task. The video loads, and below the player we see information from the mediainfo, properly labeled. That’s our goal here.

In the first couple of tasks I’ve been entering code nearly line-by-line. Now that the basics of developing with Brightcove Player are understood, rather than me typing all the code, we're just going to go through and look at the solution code, and we can gain knowledge of the concepts from that.

Looking at the solution code for task 3 we see inside the ready() method two variables are created.

Next an event handler is setup. Here the on method is used to create an event handler for the loadstart event. The loadstart event tells us when the video begins to load into the player. When that happens, the mediainfo property is populated, and we have access to the data. Again, we couldn't do this just on ready, have to wait for the loadstart event to be dispatched.

Here we log to the console what this mediainfo object is. Let's go look at that. In the browser with developer tools turned on we see the mediainfo object. Click the link to see all the properties in the mediainfo object, including:

- description  
- name  
- poster, which is the image that's used before the video starts to play  
- duration  
- tags  
- sources, in this case the rendition profile created 14 different renditions

Let's go back to our code and dynamically build the HTML that appears under the player. We're going to generate what is displayed with a combination of HTML and also some variables from the mediainfo property combined using string concatenation, in JavaScript implemented by the + sign. Here you see we have a paragraph which contains the video title label, then the name from the mediainfo, given the strong element.

We then add to that HTML the next line, using this plus equals. Add to that the description myplayer.mediainfo.description.

The HTML is now built, and next it needs to be places on the page. We get a reference to the div using the getElementById method. Next, assign the dynamically built HTML to the div using the innerHTML method. Back to the browser, and we’ve already seen it functions correctly. There's the video title label, there's the actual title. And here's a description label, and there's the actual description.

Let's do a quick review now of the coding concepts we just used. Here we have the mediainfo property. And as it says, it contains information on the media currently in the player. The property is created and populate after loadstart is dispatched.

Here's a screenshot of the data that comes back with the media info. We actually looked at it in real time with our console.

Here's highlighted in red is the information from mediainfo we used to build the dynamic HTML You can see how we can mix the static HTML with the variables from the mediainfo property to build dynamic HTML, and inject into the page.

That completes task three.

In the next video, we're going to use a different implementation of the player. So far what we've done is we've copied the actual video tag the script tag and put it onto the page. Now, in the next video, we're going to use the I-frame implementation, and we'll see both good things and bad things about using that. Hope to see you there. Thank you.