**DwBP - 8 - Using the iframe Player Implementation**

In the last two tasks, we've written code to meet a couple of use cases where we use the advanced, also called in-page embed, player implementation. This is where the video-js tag and the associated script are together on the HTML page. In this lesson, we're going to use the Standard, or iframe, player implementation. We're going to see what it looks like and discover the advantages and disadvantages of using the iframe.

Our use case this time is very similar to something we've done earlier with the Advanced code, and that is we're going to write code to change the video that's playing in the player based upon user interaction based on a button click.

Let’s look at the solution so we know exactly what we are trying to accomplish in this task. The change video button will work whether the first video is started or not. In this case the videos are NOT autoplay. The functionality is the same as task 2, but this time using a different player implementation.

Let's go into the Media module and see how we can get the Standard/ iframe player implementation code. In the media module pick a video to publish by clicking on it. Now click Publish and Embed and Web Player. Pick a player to use, then scroll down to the code section.

The Standard code is shown by default. Change any settings, and when we click on the code it is automatically copied to the clipboard. That’s what is involved getting the Standard code.

In task 4 solution we see the Standard player code and a button.

Let’s take a close look at the script block. It is important to now that really this is a JavaScript exercise, not really a Brightcove Player exercise, which can happen when doing Brightcove Player development.

The high-level solution to the problem is we are going to programmatically **replace** the query string that contains the video ID that is in the player. When we do this, the browser will automatically reload the player with the new video in the player.

The code does this:  
- Gets a handle on the iframe element  
- Creates a variable that contains the new query string, which contains the new video ID  
- Extracts the value of the src attribute from the iframe and stores it in a variable named theSrc  
- Uses the JavaScript substring() method to remove the old query string from the source URL  
- Use string concatenation to place the new query string on the source URL  
- Assign the new source back to the iframe’s src attribute

We’ve already seen this works correctly.

Let's take a couple minutes now that we're somewhat familiar with the iframe implementation to talk about some good and maybe not so good things about the iframe and a few other details, and also some review.

First of all, the advantages of iframe player. Number one, there will be no collisions with existing JavaScript and/or CSS. It's possible if we haven't handled our JavaScript in an elegant manner we could have conflict simply by using the same variable name in two different ways. That cannot happen when we're using an iframe player.

It's nearly automatically responsive. We just have to go in and make sure that our width and height are set to percentages.

Another nice thing about the iframe implementations is that it travels nicely to social media apps. So, if we want to share a video like in Facebook or on Twitter, it will use the iframe implementation to do that.

There are some times when we have to be careful about using the iframe implementation. One of those is code in the containing page needs to listen for or act on player events. For instance, before we had some code in the player and we wanted something to happen, like display some text in the HTML page, based upon the loadstart event. That we can't do with an iframe implementation.

Also, if we want to pull styles in and affect our player based on styles from the containing page, we won't be able to do that.

Lastly, there may be times where it simply won't fit our application logic. For instance, let's say at the end of a video or at a call to action we want to change the page we're on. Well if we have that logic in the iframe, the page will change, but all that will change is in the iframe. And so we'll be in the new page just in the iframe, and all the surrounding HTML page will be the same.

So there's good things and bad things about using the iframe player.

So when should we use Standard (iframe) versus Advanced (in-page) implementation? The general rule of thumb is that if we are going to do development around the player, like we are doing this course, we should use the Advanced implementation. If we are not going to alter the player with code, and/or using the player in social media, then we should use the Standard implementation.

It may seem like there's no way to communicate between the HTML page and the iframe based on when we have seen here. Actually, we can, but it may be a bit difficult based on our HTML and JavaScript background. This sample exists that shows how to play the video from the iframe parent. The technique the sample uses is not part of the Brightcove player API, but the JavaScript postmessage method. This is how we can send message, in the format of a JavaScript object, into the iframe. The iframe listens for an event that comes in, and then can access the data in the object passed in.

Great, we have accomplished another task. What we've done is changed the video in an iframe player implementation.

In the next video, we're going to use a great feature of Brightcove player call plugins. This provides a best practice way to add functionality to our player.

Hope to see you there. Thank for watching!!