Developing with Brightcove Player

Matt Boles

mboles@brightcove.com

BRIGHTCOVE®

Introducing the Course

What: Brightcove Player

- The Brightcove Player is based on the open source Video.js Player

- Three core elements:
  - Video embed code - Places a video into a website using the <video-js> element
  - JavaScript library - Makes the player work across browsers, their various versions and around device / platform bugs
  - Pure HTML/CSS skin - Creates a uniform look across HTML5 browsers and easy custom skinning for a branded look

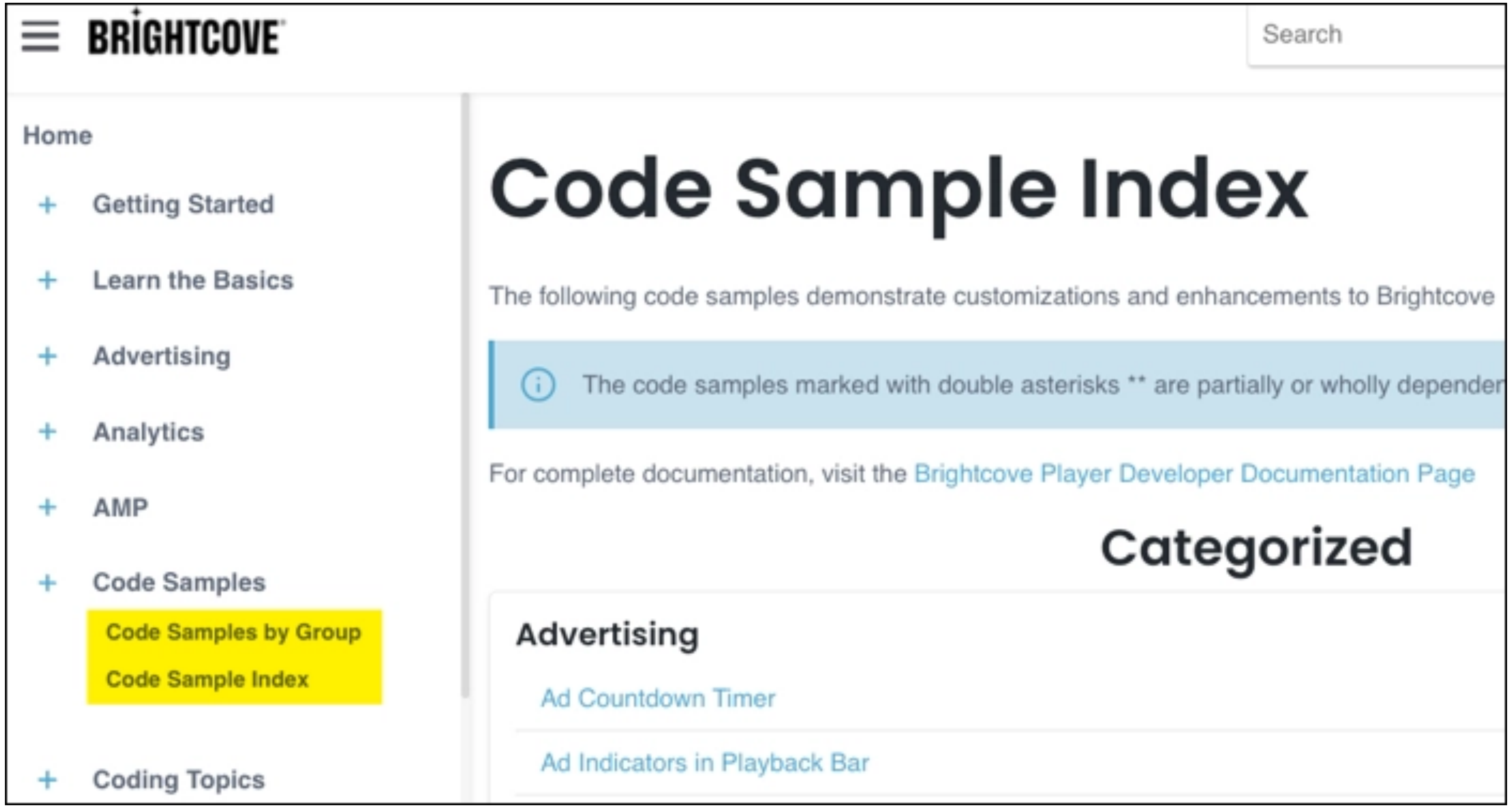
What: Brightcove Player Development

- Used to customize, integrate with, or add functionality to, your players
- Uses HTML5, CSS, JavaScript and the Player API

Cross-platform standards
Developer-friendly technologies

Why: Code Samples

- https://player.support.brightcove.com/code-samples/index.html



Search

Home

+ Getting Started

+ Learn the Basics

+ Advertising

+ Analytics

+ AMP

+ Code Samples

Code Samples by Group

Code Sample Index

+ Coding Topics

Code Sample Index

The following code samples demonstrate customizations and enhancements to Brightcove

The code samples marked with double asterisks ** are partially or wholly dependen

For complete documentation, visit the Brightcove Player Developer Documentation Page

Categorized

Advertising

Ad Countdown Timer

Ad Indicators in Playback Bar

How: Agenda

- Introducing the Course
- Setting Up to Develop with Brightcove Player
- Using JavaScript with Brightcove Player
- Getting Started with Brightcove Player Development
- Task 1: Using the API to Play a Video
- Using the Player Catalog
- Task 2: Dynamically Loading and Playing a Video
- Using the mediainfo Object
- Task 3: Displaying Video Information in the HTML Page
- Using the Standard (iframe) Player Implementation
- Task 4: Changing the Video in an iframe Player Implementation

How: Agenda (cont)

- Adding a Brightcove Plugin to a Player
- Task 5: Adding the Overlay Plugin to a Player
- Task 6: Using the IMA Plugin to Play VAST Ads

Review poll questions also asked periodically

Prerequisites

- The session is designed for developers with basic HTML and JavaScript experience

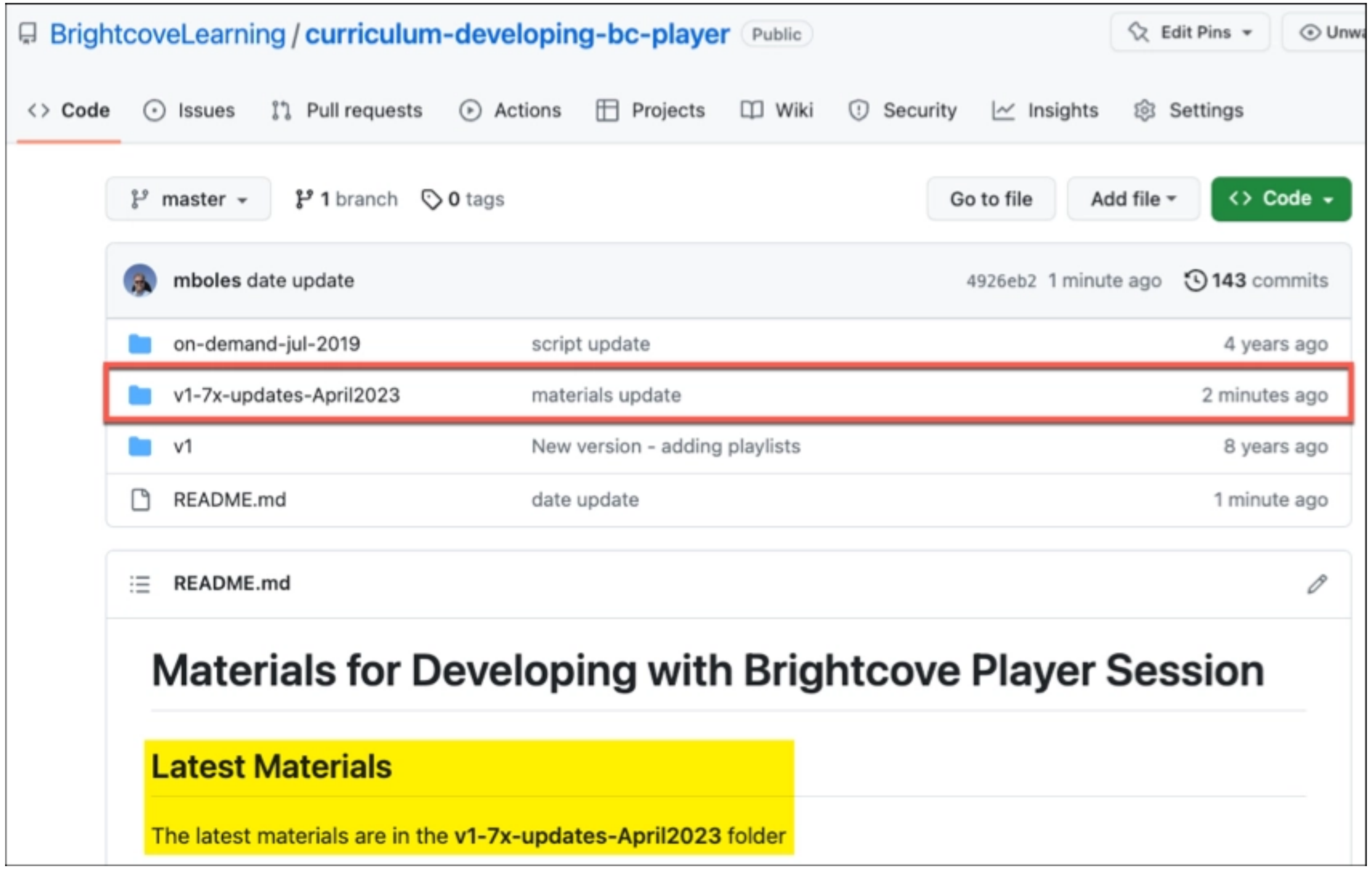
Setting Up to Develop with Brightcove Player

Setup

- Video Cloud Account
- You will also need an editor for HTML/JavaScript
  - Any plain text editor will work
  - An editor such as Visual Studio Code, Atom, Chocolat, Sublime Text, Dreamweaver, BBEdit, or CoffeeCup, that provides code-hinting and syntax highlighting is recommended
- For iframe player implementation examples a web server is needed
  - XAMPP and WAMP free options

Getting Session Materials - GitHub

- Student files and slides
  - https://github.com/BrightcoveLearning/curriculum-developing-bc-player


BrightcoveLearning / curriculum-developing-bc-player Public

mboles date update 4926eb2 1 minute ago 143 commits

on-demand-jul-2019	script update	4 years ago
v1-7x-updates-April2023	materials update	2 minutes ago
v1	New version - adding playlists	8 years ago
README.md	date update	1 minute ago

README.md

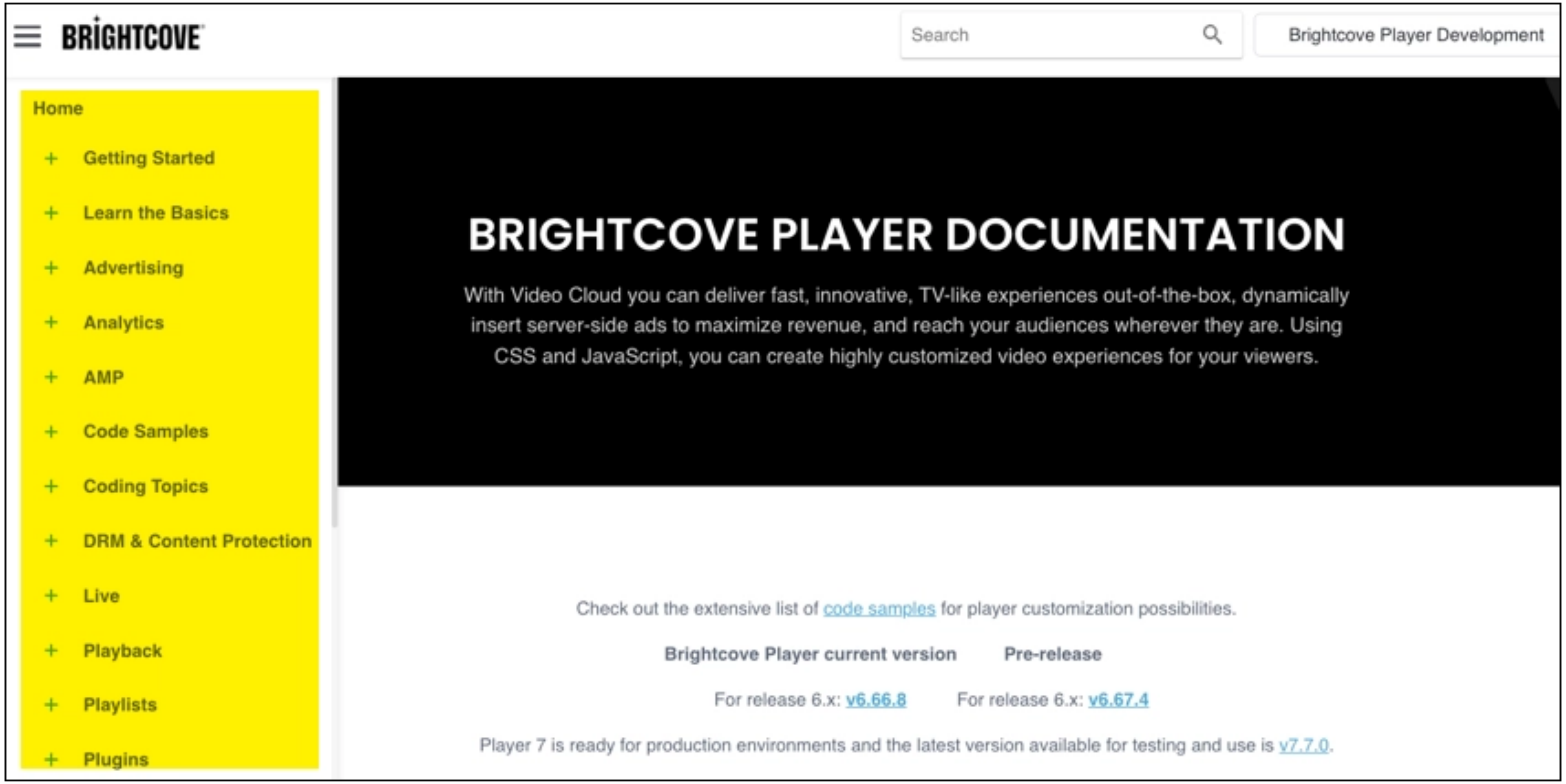
Materials for Developing with Brightcove Player Session

Latest Materials

The latest materials are in the **v1-7x-updates-April2023** folder

Brightcove Player Documentation

- https://player.support.brightcove.com/index.html



Search

Brightcove Player Development

Home

- Getting Started
- Learn the Basics
- Advertising
- Analytics
- AMP
- Code Samples
- Coding Topics
- DRM & Content Protection
- Live
- Playback
- Playlists
- Plugins

BRIGHTCOVE PLAYER DOCUMENTATION

With Video Cloud you can deliver fast, innovative, TV-like experiences out-of-the-box, dynamically insert server-side ads to maximize revenue, and reach your audiences wherever they are. Using CSS and JavaScript, you can create highly customized video experiences for your viewers.

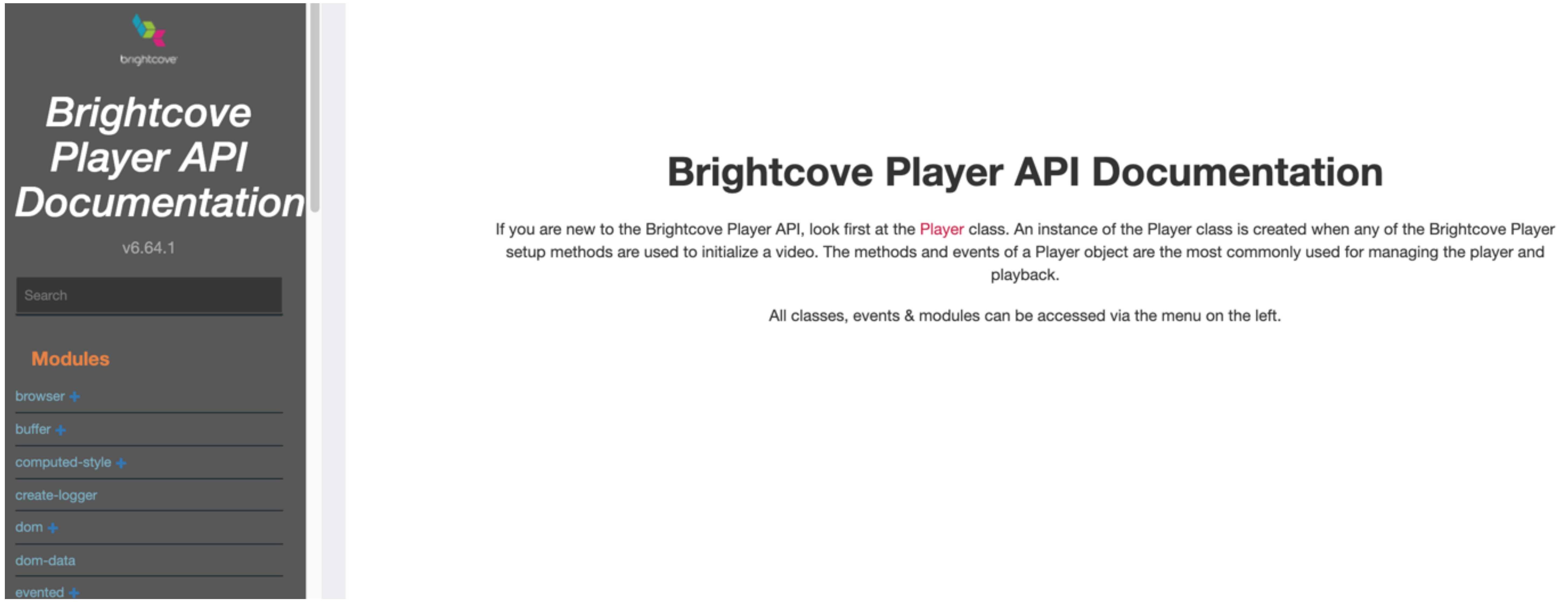
Check out the extensive list of code samples for player customization possibilities.

Brightcove Player current version	Pre-release
For release 6.x: v6.66.8	For release 6.x: v6.67.4

Player 7 is ready for production environments and the latest version available for testing and use is v7.7.0.

Brightcove Player API Documentation

- https://player.support.brightcove.com/brightcove-player/current-release/index.html



Brightcove Player API Documentation

v6.64.1

Search

Modules

- browser +
- buffer +
- computed-style +
- create-logger
- dom +
- dom-data
- evented +

Brightcove Player API Documentation

If you are new to the Brightcove Player API, look first at the Player class. An instance of the Player class is created when any of the Brightcove Player setup methods are used to initialize a video. The methods and events of a Player object are the most commonly used for managing the player and playback.

All classes, events & modules can be accessed via the menu on the left.

Demo: Programmatically Play a Video

Quick look at the process of using the API

(a “Spiral Learning” event)

Using JavaScript with Brightcove Player

API Is Event Driven

- Event driven framework: Behaviors driven by the production, detection and consumption of events

```
function foo() {
  player = this;
  player.loadVideo(123);
  player.play();
}
```

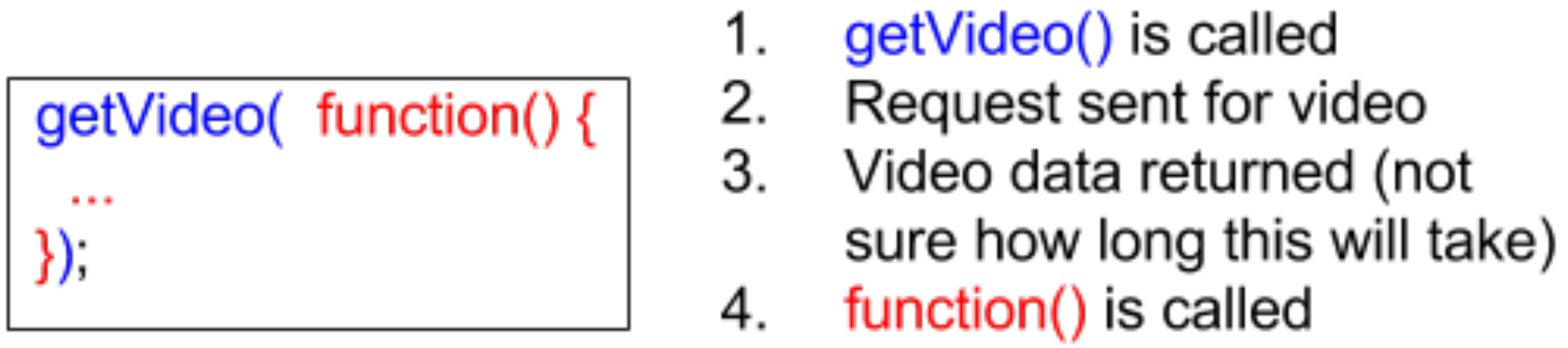
```
videojs.getPlayer('myPlayerID')
  .ready(function(){
  var myPlayer = this;
});

otherComponent.on('play', function(){
  //Video is playing
});
```

Callback Functions

- A function passed to another function to be called at a later time

- Example: `getVideo()` called, then the callback function called when video data returned, which is a variable amount of time



```
getVideo(  function() {
  ...
});
```

1. getVideo() is called
2. Request sent for video
3. Video data returned (not sure how long this will take)
4. function() is called

Callback Function Implementations

- Anonymous functions: The function definition is the argument of the function
  - Function not named, hence anonymous
  - Called immediately after `getVideo` function has done its job

```
getVideo( function(){ … })
```

- Function declaration ("normal way")
  - Loads before any code is executed, then called from different location

```
function foo() { … }
```

- Function expression
  - Loads only when the interpreter reaches that line of code, then called from a different location

Conceptual Blockbusters!!

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

BRIGHTCOVE®

Quick Review Poll

DwBP1

Quick Review Poll

DwBP2

Getting Started with Brightcove Player Development

Use Case: Play the video programmatically

Get Reference to Player

1. Create a `<script>` block
2. Use the `ready` method
3. Create variable that holds reference to the player instance

```
videojs.getPlayer('myPlayerID').ready(function(){
  var myPlayer = this;
});
```

Get Reference to Player - cont

- Note that using `ready()` functions correctly if you wish to interact with the player, for instance programmatically to change player behavior

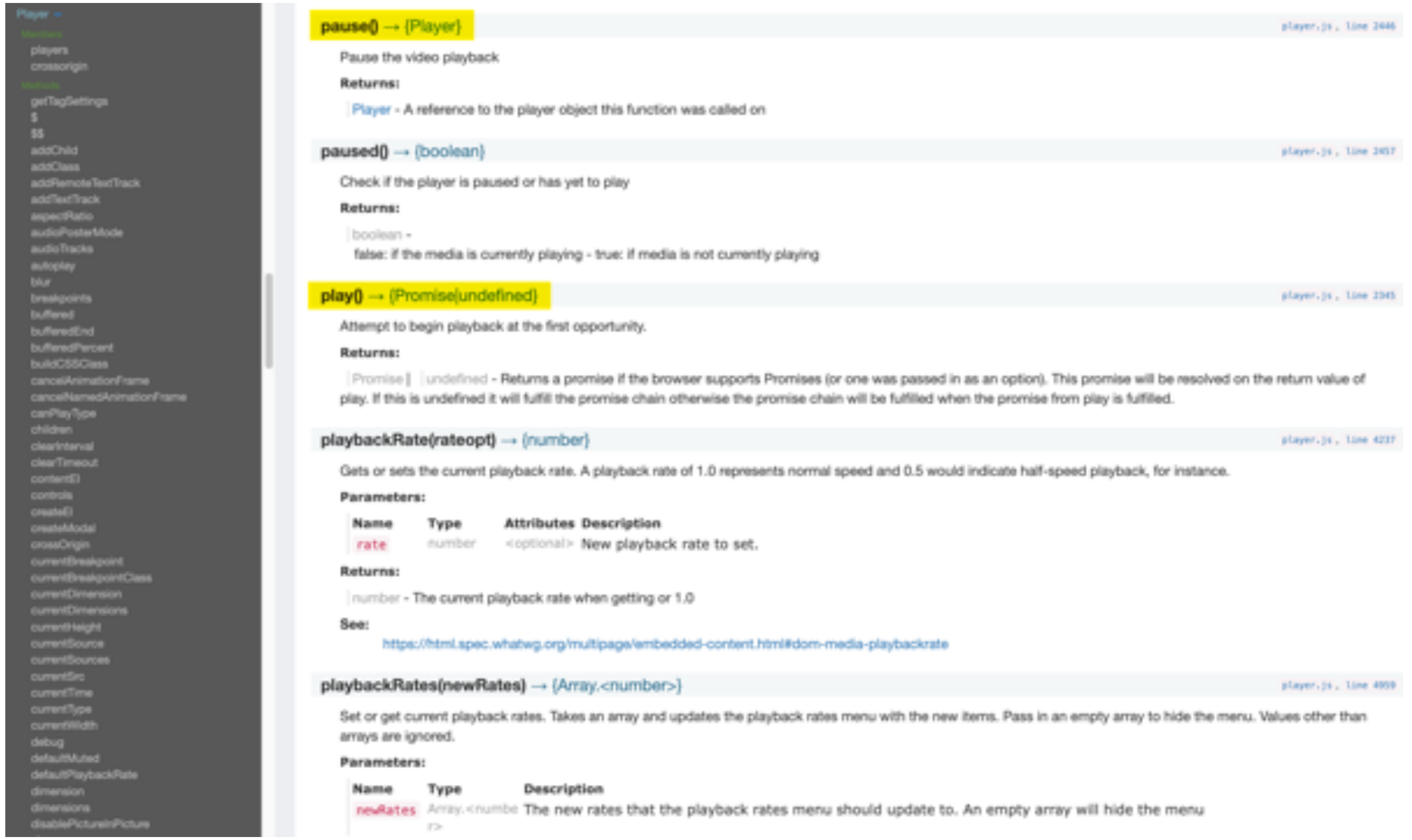
- If you wish to immediately interact with the video, for instance use `play()`, another approach must be used
  - Detailed in the coming **Events** section

Player Methods

- Docs: https://player.support.brightcove.com/brightcove-player/current-release/Player.html

- Method example

```
myPlayer.play();
 myPlayer.muted(true);
```



Player

Members

- players
- crossorigin

Methods

- getTagSettings
- $
- $$
- addChild
- addClass
- addRemoteTextTrack
- addTextTrack
- aspectRatio
- audioPosterMode
- audioTracks
- autoplay
- blur
- breakpoints
- buffered
- bufferedEnd
- bufferedPercent
- buildCSSClass
- cancelAnimationFrame
- cancelNamedAnimationFrame
- canPlayType
- children
- clearInterval
- clearTimeout
- contentEl
- controls
- createEl
- createModal
- crossOrigin
- currentBreakpoint
- currentBreakpointClass
- currentDimension
- currentDimensions
- currentHeight
- currentSource
- currentSources
- currentSrc
- currentTime
- currentType
- currentWidth
- debug
- defaultMuted
- defaultPlaybackRate
- dimension
- dimensions
- disablePictureInPicture

pause() → {Player} player.js, line 2446

Pause the video playback

Returns:

Player - A reference to the player object this function was called on

paused() → {boolean} player.js, line 2457

Check if the player is paused or has yet to play

Returns:

boolean -
false: if the media is currently playing - true: if media is not currently playing

play() → {Promise|undefined} player.js, line 2345

Attempt to begin playback at the first opportunity.

Returns:

Promise | undefined - Returns a promise if the browser supports Promises (or one was passed in as an option). This promise will be resolved on the return value of play. If this is undefined it will fulfill the promise chain otherwise the promise chain will be fulfilled when the promise from play is fulfilled.

playbackRate(rateopt) → {number} player.js, line 4237

Gets or sets the current playback rate. A playback rate of 1.0 represents normal speed and 0.5 would indicate half-speed playback, for instance.

Parameters:

Name	Type	Attributes	Description
rate	number	<optional>	New playback rate to set.

Returns:

number - The current playback rate when getting or 1.0

See:

https://html.spec.whatwg.org/multipage/embedded-content.html#dom-media-playbackrate

playbackRates(newRates) → {Array.<number>} player.js, line 4959

Set or get current playback rates. Takes an array and updates the playback rates menu with the new items. Pass in an empty array to hide the menu. Values other than arrays are ignored.

Parameters:

Name	Type	Description
newRates	Array.<number>	The new rates that the playback rates menu should update to. An empty array will hide the menu

Player Events

- Docs: https://docs.brightcove.com/brightcove-player/current-release/Player.html#event:ended
- Use `on()`, `one()` and `off()` methods to add and remove event listeners

- Event example

```
myPlayer.on("timeupdate", showUpdate);
```

Player Events - cont

- If you wish to immediately interact with the video, for instance use `play()`, you should use the `loadedmetadata` event to be sure the VIDEO is loaded in the PLAYER

```
videojs.getPlayer('myPlayerID').ready(function(){
    var myPlayer = this;
    myPlayer.muted(true);
        myPlayer.on('loadedmetadata', function(){
             myPlayer.play();
        });
 });
```

Considerations for autoplay

- Using the `muted()` getter/setter method to avoid the issue in this session
- Document available with details
  - Autoplay Considerations
  - https://player.support.brightcove.com/playback/autoplay-considerations.html

- Sample "solution"
  - Brightcove Player Sample: Autoplay with Unmute Button for iOS/Safari/Chrome
  - https://player.support.brightcove.com/code-samples/brightcove-player-sample-autoplay-unmute-button.html

Conceptual Blockbuster!!

- When playing a video in the Video Cloud environment, TWO entities are involved
  - Player
  - Video

BRIGHTCOVE®

Task 1: Using the API to Play a Video and Display Event Object

Using the Player Catalog

Use Case: Change the video on user interaction

Player Catalog

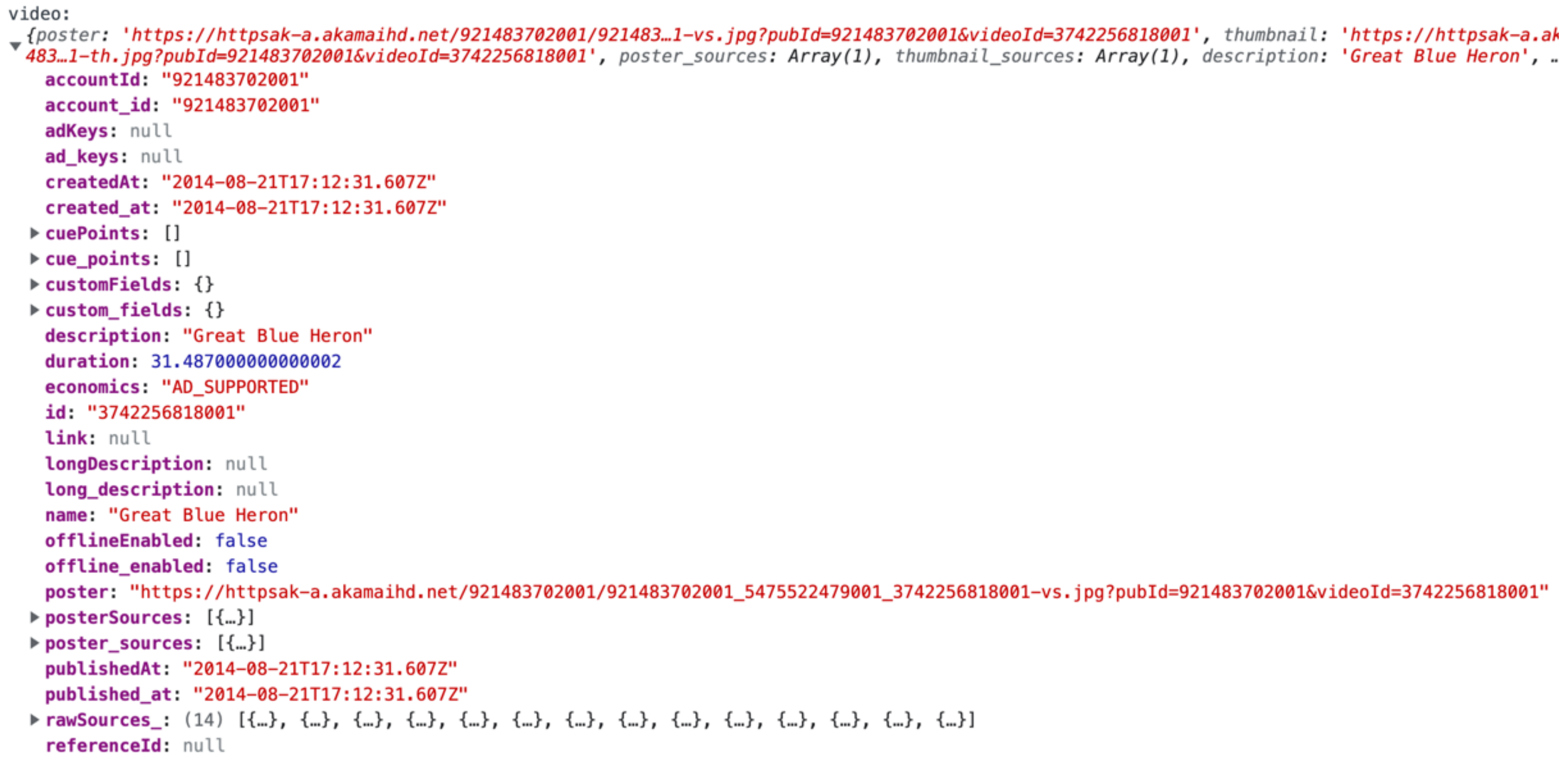
- Player Catalog is a helper library for making requests to the Video Cloud catalog
  - The catalog makes it easy to get information on Video Cloud media/playlists and use

- Numerous methods available, but in this session will focus on

```
myPlayer.catalog.getVideo(videoID,callback)
myPlayer.catalog.getPlaylist(playlistID,callback)
myPlayer.catalog.load(videoObject)
```

Returned Object from getVideo()

- Example **video** object



```
video:
▼{poster: 'https://httpsak-a.akamaihd.net/921483702001/921483…1-vs.jpg?pubId=921483702001&videoId=3742256818001', thumbnail: 'https://httpsak-a.ak
 483…1-th.jpg?pubId=921483702001&videoId=3742256818001', poster_sources: Array(1), thumbnail_sources: Array(1), description: 'Great Blue Heron', …
   accountId: "921483702001"
   account_id: "921483702001"
   adKeys: null
   ad_keys: null
   createdAt: "2014-08-21T17:12:31.607Z"
   created_at: "2014-08-21T17:12:31.607Z"
 ▶ cuePoints: []
 ▶ cue_points: []
 ▶ customFields: {}
 ▶ custom_fields: {}
   description: "Great Blue Heron"
   duration: 31.487000000000002
   economics: "AD_SUPPORTED"
   id: "3742256818001"
   link: null
   longDescription: null
   long_description: null
   name: "Great Blue Heron"
   offlineEnabled: false
   offline_enabled: false
   poster: "https://httpsak-a.akamaihd.net/921483702001/921483702001_5475522479001_3742256818001-vs.jpg?pubId=921483702001&videoId=3742256818001"
 ▶ posterSources: [{…}]
 ▶ poster_sources: [{…}]
   publishedAt: "2014-08-21T17:12:31.607Z"
   published_at: "2014-08-21T17:12:31.607Z"
 ▶ rawSources_: (14) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
   referenceId: null
```

Task 2: Dynamically Loading and Playing a Video

Quick Review Poll

DwBP3

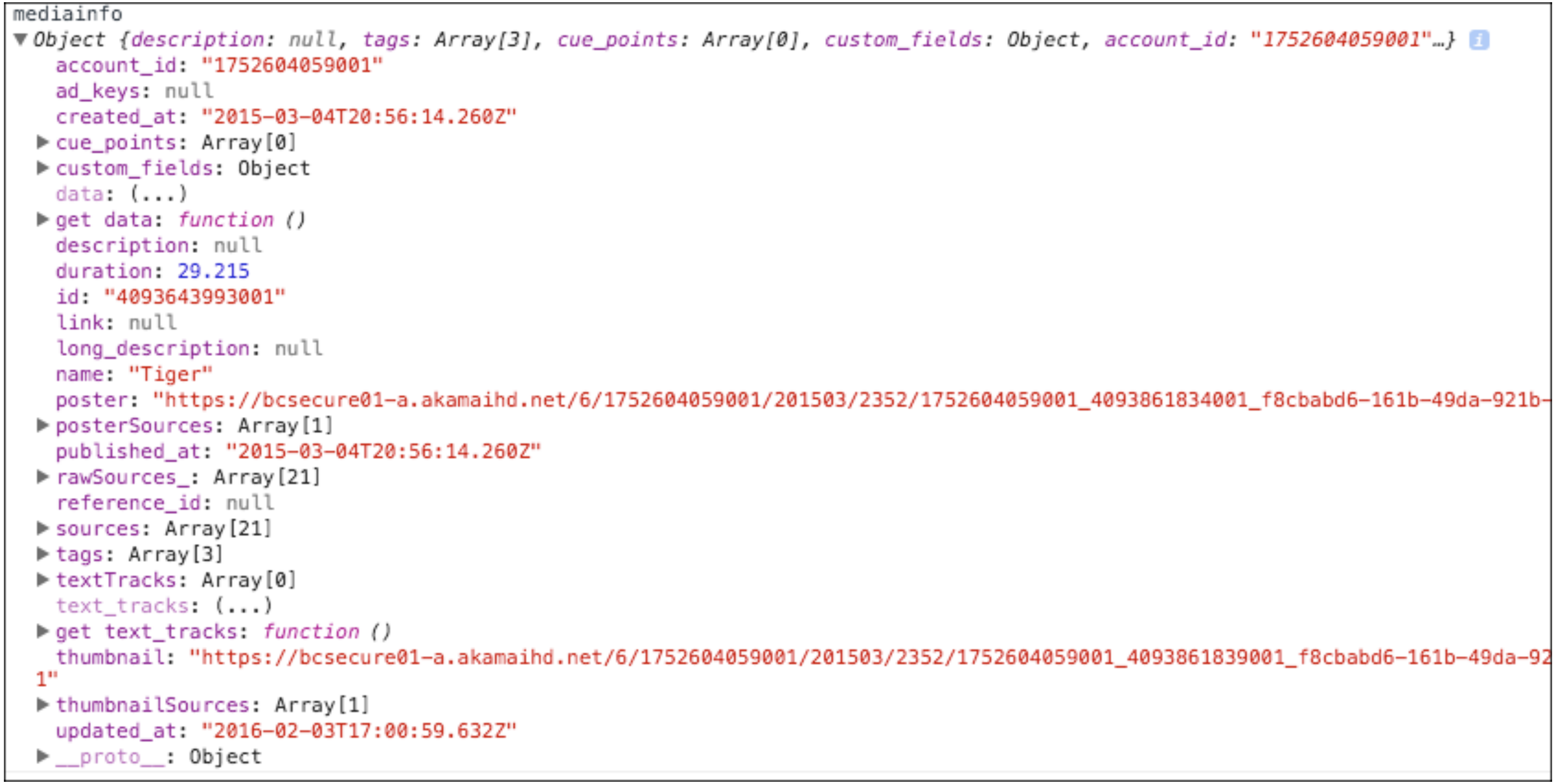
Using the mediainfo Object

Use Case: Display information about the video on the HTML page

mediainfo Object

- The `mediainfo` object is an object which contains information on the current media in the player
- The object is created and populated after the `loadstart` event is dispatched
- After the mediainfo object is populated, use it for convenient data retrieval when wishing to display video information, like the video name or description

Data in mediainfo



```
mediainfo
▼ Object {description: null, tags: Array[3], cue_points: Array[0], custom_fields: Object, account_id: "1752604059001"…}
    account_id: "1752604059001"
    ad_keys: null
    created_at: "2015-03-04T20:56:14.260Z"
  ▶ cue_points: Array[0]
  ▶ custom_fields: Object
    data: (...)
  ▶ get data: function ()
    description: null
    duration: 29.215
    id: "4093643993001"
    link: null
    long_description: null
    name: "Tiger"
    poster: "https://bcsecure01-a.akamaihd.net/6/1752604059001/201503/2352/1752604059001_4093861834001_f8cbabd6-161b-49da-921b-
  ▶ posterSources: Array[1]
    published_at: "2015-03-04T20:56:14.260Z"
  ▶ rawSources_: Array[21]
    reference_id: null
  ▶ sources: Array[21]
  ▶ tags: Array[3]
  ▶ textTracks: Array[0]
    text_tracks: (...)
  ▶ get text_tracks: function ()
    thumbnail: "https://bcsecure01-a.akamaihd.net/6/1752604059001/201503/2352/1752604059001_4093861839001_f8cbabd6-161b-49da-92
  1"
  ▶ thumbnailSources: Array[1]
    updated_at: "2016-02-03T17:00:59.632Z"
  ▶ __proto__: Object
```

Access mediainfo Data

- Access the data in the mediainfo object by simple `object.property` notation

```
dynamicHTML = "<p>Video Title: <strong>" +
 myPlayer.mediainfo.name + "</strong></p>";

dynamicHTML += "<p>Description: <strong>" +
 myPlayer.mediainfo.description + "</strong></p>";

document.getElementById("textTarget").innerHTML =
 dynamicHTML;
```

Conceptual Blockbuster!!

- You cannot access the `mediainfo` object until the `loadstart` event is dispatched

Task 3: Display Video Information in the HTML Page

CodePen: http://codepen.io/team/bcls/pen/KzyoNG

Using the Standard (iframe) Player Implementation

Use Case: Utilize the iframe implementation of the player and change the video on user interaction

Advantages of Standard (iframe) Player Implementation

- No collisions with existing JavaScript and/or CSS
- Automatically responsive (nearly)
- The iframe eases use in social media apps (or whenever the video will need to "travel" into other apps)

When You Cannot Use iframe Implementation

- Code in the containing page needs to listen for and act on player events
- The player uses styles from the containing page
- The iframe will cause app logic to fail, like a redirect from the containing page

Dynamically Change Video in iframe

- To dynamically change video in an iframe change the query string's the `src` property

```
<iframe src="https://players.brightcove.net/921483702001/MCQjvqXXF_default/
    index.html?videoId=5831704295001"
    allowfullscreen=""
    allow="encrypted-media"
    width="960" height="540"></iframe>
```

- Need to remove the existing query string then add a new one

Dynamically Change Video in iframe (cont)

- Plan of action
  1. Get a handle on the `<iframe>` tag
  2. Create a variable with the new query string (new video ID)
  3. Assign the `src` property of the `<iframe>` to a variable
  4. Remove the existing query string from the source
  5. Add the new query string to the source
  6. Assign the new source to the `<iframe>`

Dynamically Change Video in iframe (cont)

```
<function changeVideo() {
 var iframeTag = document.getElementsByTagName("iframe")[0],
  newVideo = "?videoId=3742256815001",
  theSrc = iframeTag.src,
  srcWithoutVideo = theSrc.substring( 0,  theSrc.indexOf( "?" ) ),
  newSrc = srcWithoutVideo + newVideo;
 iframeTag.src = newSrc;
}
```

- JavaScript's `theString.substring()` extracts characters from the first parameter to the second

Communicate Between HTML Page and iframe

- It is possible to communicate between the parent page and the iframe
  - Uses HTML `postMessage()`

- Example doc: *Play/Pause Video from iframe Parent*
  - https://player.support.brightcove.com/code-samples/brightcove-player-sample-playpause-video-iframe-parent.html

Task 4: Changing the Video in an iframe Player Implementation

CodePen: http://codepen.io/team/bcls/pen/WwXVNm

Quick Review Poll

DwBP4

Adding a Brightcove Plugin to a Player

Use Case 1: Play IMA3 ads

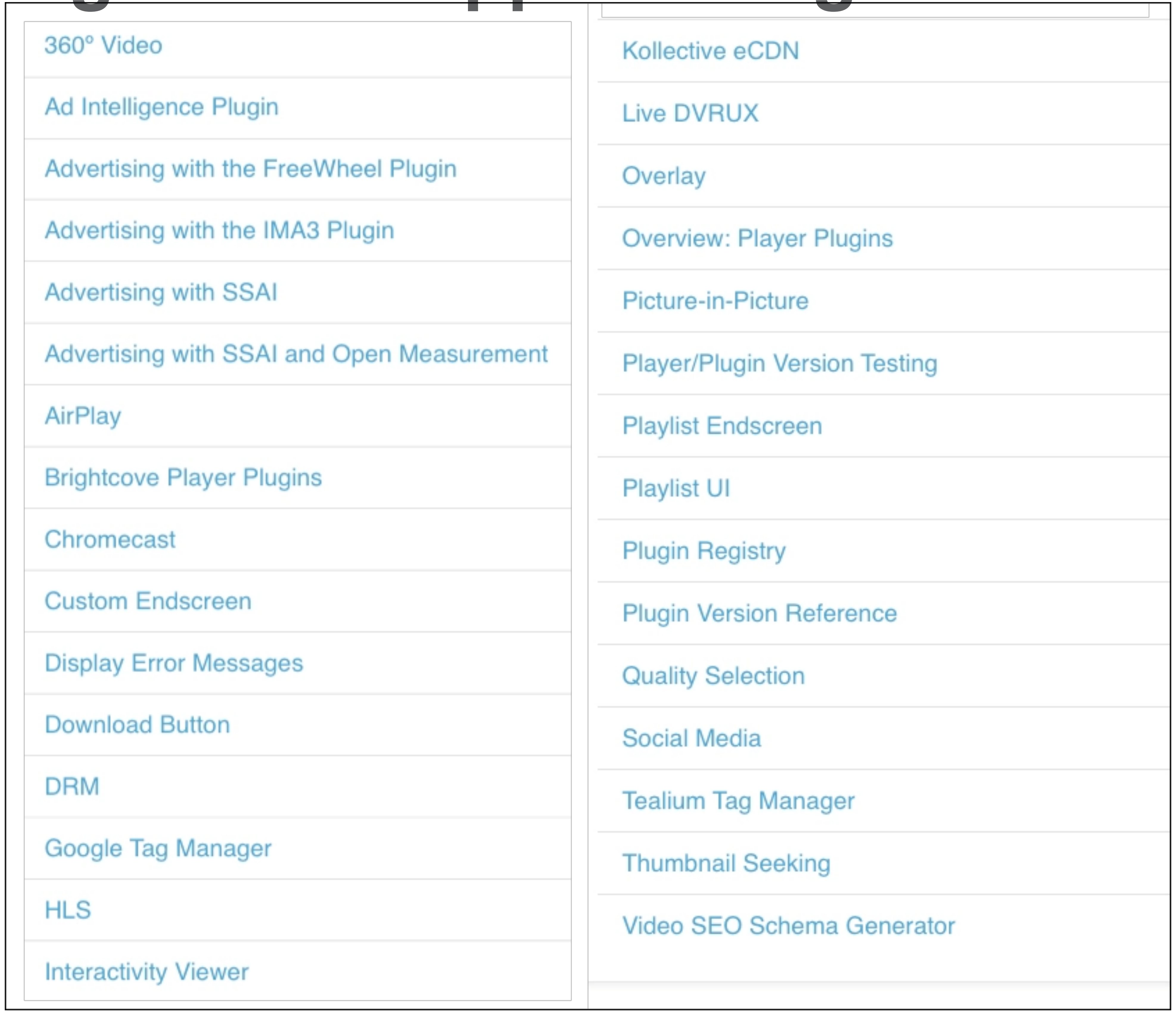
Use Case 2: Display an overlay that uses data from the mediainfo object

Plugins for Brightcove Player

- A plugin for the Brightcove player uses a combination of HTML, JavaScript and/or CSS to somehow customize the player
  - In other words, anything you can do in a web page, you can do in a plugin

- Broadly, plugins can be developed to
  - Modify default behavior
  - Add functionality
  - Customize appearance

Brightcove Supplied Plugins



- 360° Video
- Ad Intelligence Plugin
- Advertising with the FreeWheel Plugin
- Advertising with the IMA3 Plugin
- Advertising with SSAI
- Advertising with SSAI and Open Measurement
- AirPlay
- Brightcove Player Plugins
- Chromecast
- Custom Endscreen
- Display Error Messages
- Download Button
- DRM
- Google Tag Manager
- HLS
- Interactivity Viewer
- Kollective eCDN
- Live DVRUX
- Overlay
- Overview: Player Plugins
- Picture-in-Picture
- Player/Plugin Version Testing
- Playlist Endscreen
- Playlist UI
- Plugin Registry
- Plugin Version Reference
- Quality Selection
- Social Media
- Tealium Tag Manager
- Thumbnail Seeking
- Video SEO Schema Generator

Brightcove Plugins Loaded by Default

- The following are plugins loaded by default
  - Errors
  - HLS
  - DRM

Add Plugin In Page or In Studio

- In Page: Plugin only affects that player instance
- Studio: All instances of that player will have plugin functionality

Implementing Plugins Using Studio UI

- One of three ways to use a plugin

- Use the Studio UI to supply the plugin's
  - JavaScript
  - Name
  - Options (if needed)
  - CSS (if needed)

- Plugin associated with ALL instances of the player

Implementing Plugins Using Custom Code

- Second way use a plugin
  - Use a `<script>` tag to manually include the plugin's JavaScript
  - Use a `<link>` tag to manually include the plugin's CSS (if needed)
  - Call the plugin as a method, supplying required options

```
myPlayer.overlay({
     …
 });
```

- Plugin associated ONLY with the instance of the player on the page

- Provides flexibility, such as dynamically supplying options

Task 5: Play IMA3 Ads (Studio based task)
AND/OR
Task 6: Display an Overlay that Uses mediainfo Data

Task 6 CodePen: http://codepen.io/team/bcls/pen/PNEWQJ

BRIGHTCOVE®

FULL STREAM AHEAD