Developing with the Brightcove Player

Matt Boles

mboles@brightcove.com

Introducing the Course

What: Brightcove Player

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

- Three core elements:
  - Video embed code - Places a video into a website using the HTML5 <video> element falling back to Flash automatically
  - 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

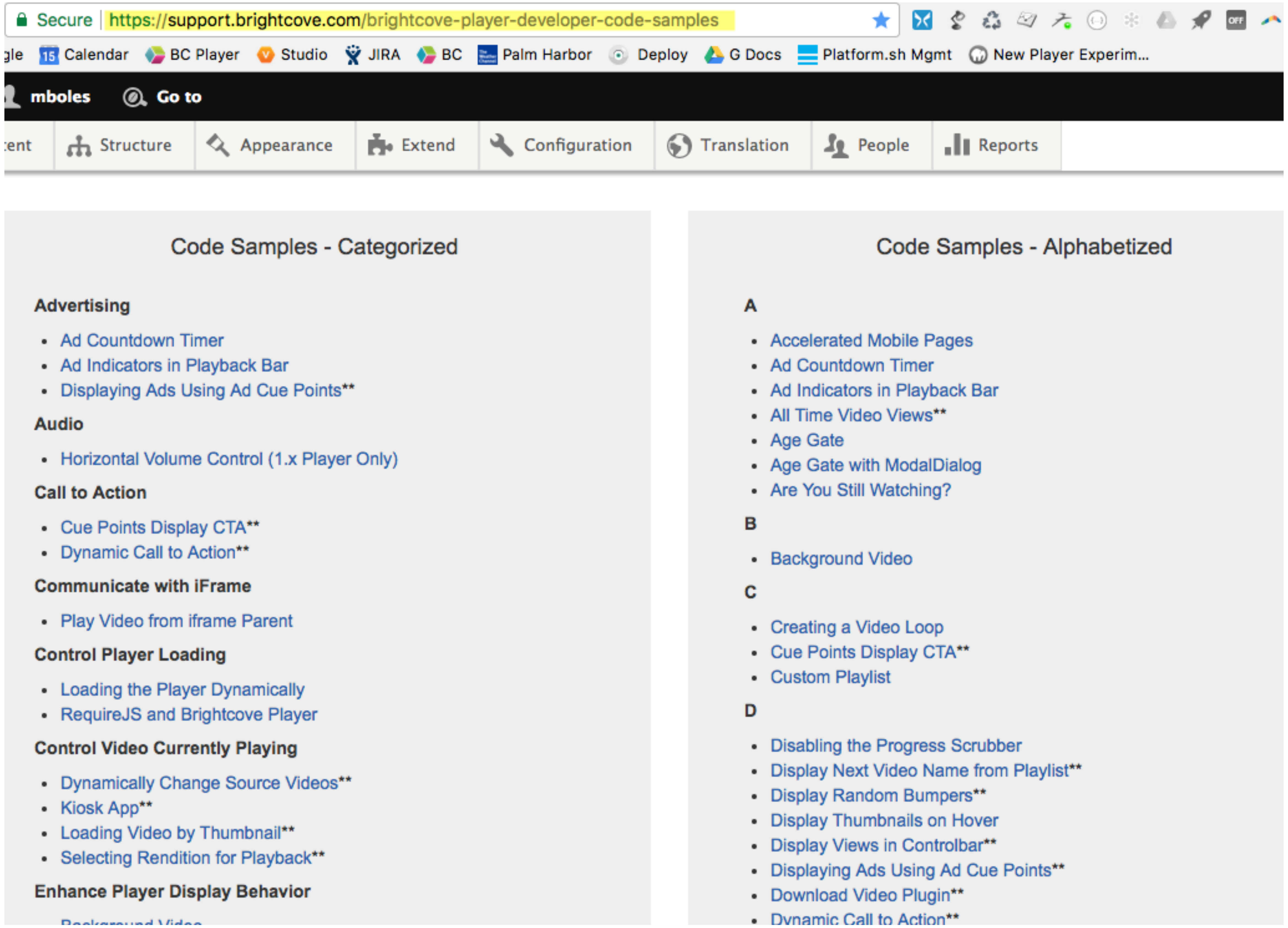
- Used to customize, integrate with, or add functionality to, your players

- Uses HTML5, CSS, JavaScript and the Player API

HTML CSS JAVASCRIPT

Cross-platform standards
Developer-friendly
technologies

Why: Code Samples



Secure | https://support.brightcove.com/brightcove-player-developer-code-samples

Code Samples - Categorized

Advertising

- Ad Countdown Timer
- Ad Indicators in Playback Bar
- Displaying Ads Using Ad Cue Points**

Audio

- Horizontal Volume Control (1.x Player Only)

Call to Action

- Cue Points Display CTA**
- Dynamic Call to Action**

Communicate with iFrame

- Play Video from iframe Parent

Control Player Loading

- Loading the Player Dynamically
- RequireJS and Brightcove Player

Control Video Currently Playing

- Dynamically Change Source Videos**
- Kiosk App**
- Loading Video by Thumbnail**
- Selecting Rendition for Playback**

Enhance Player Display Behavior

Code Samples - Alphabetized

A

- Accelerated Mobile Pages
- Ad Countdown Timer
- Ad Indicators in Playback Bar
- All Time Video Views**
- Age Gate
- Age Gate with ModalDialog
- Are You Still Watching?

B

- Background Video

C

- Creating a Video Loop
- Cue Points Display CTA**
- Custom Playlist

D

- Disabling the Progress Scrubber
- Display Next Video Name from Playlist**
- Display Random Bumpers**
- Display Thumbnails on Hover
- Display Views in Controlbar**
- Displaying Ads Using Ad Cue Points**
- Download Video Plugin**

How: Agenda

- Introducing the Course
- Setting Up to Develop with Brightcove Player
- Using JavaScript with Brightcove Player
- Getting Started with Brightcove Player Development
- Task1: Using the API to Play a Video
- Using the Player Catalog
- Task 2: Dynamically Loading and Playing a Video
- Using the mediainfo Property
- Task 3: Displaying Video Information in the HTML Page
- Using the iframe Player Implementation
- Task 4: Changing the Video in an iframe Player Implementation

How: Agenda (cont)

- Adding a Brightcove Plugin to a Player
- Task5: Adding the Overlay Plugin to a Player
- Task 6: Using the IMA Plugin to Play VAST Ads
- Using Playlists
- Task 7: Associate a Playlist with a Player

Prerequisites

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

Setting Up to Develop with Brightcove Player

Setup

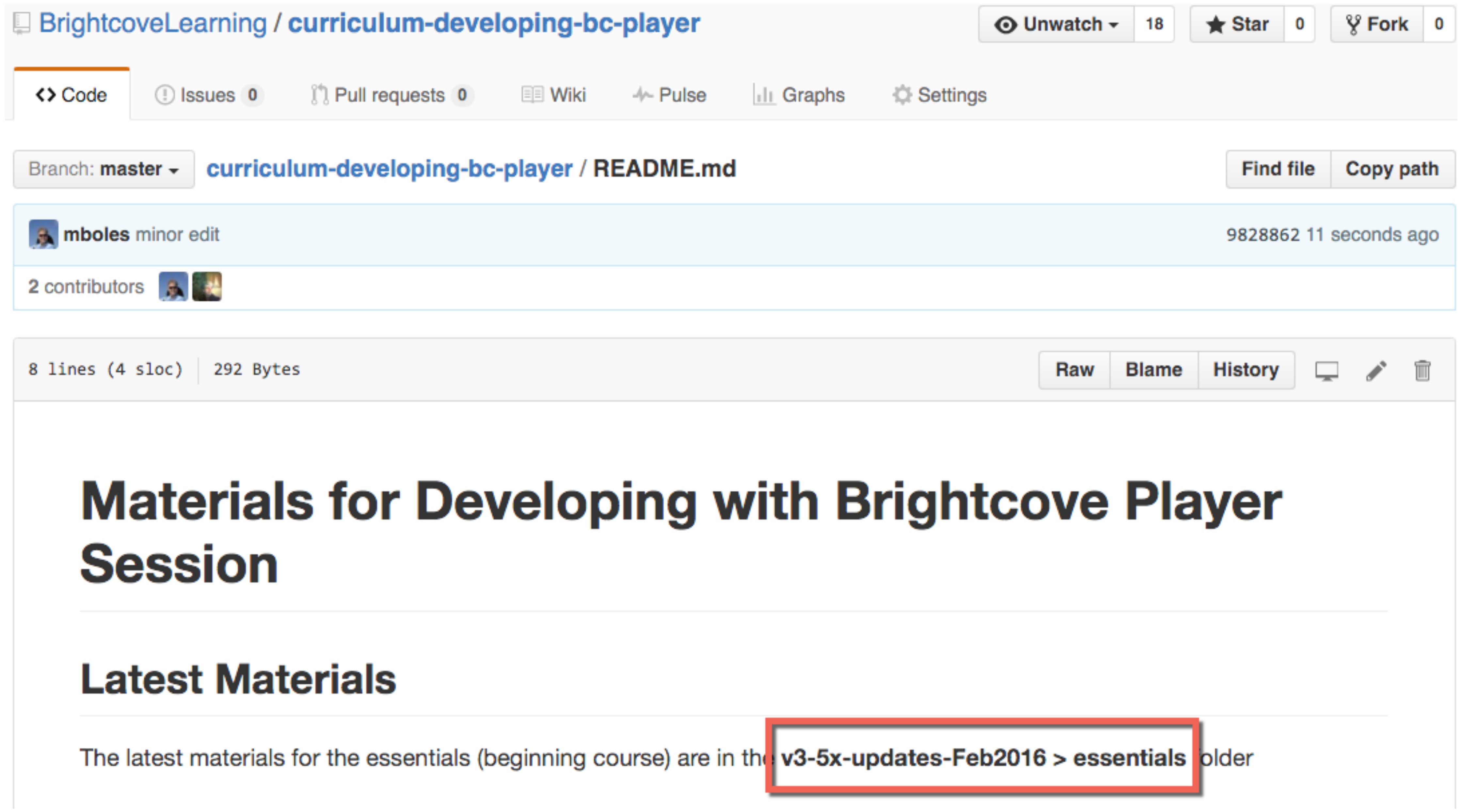
- Video Cloud Account

- You will also need an editor for HTML/JavaScript
  - Any plain text editor will work
  - An editor such as 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
  - http://bit.ly/1EDWaCA



curriculum-developing-bc-player / README.md

mboles minor edit 9828862 11 seconds ago

2 contributors

8 lines (4 sloc) 292 Bytes

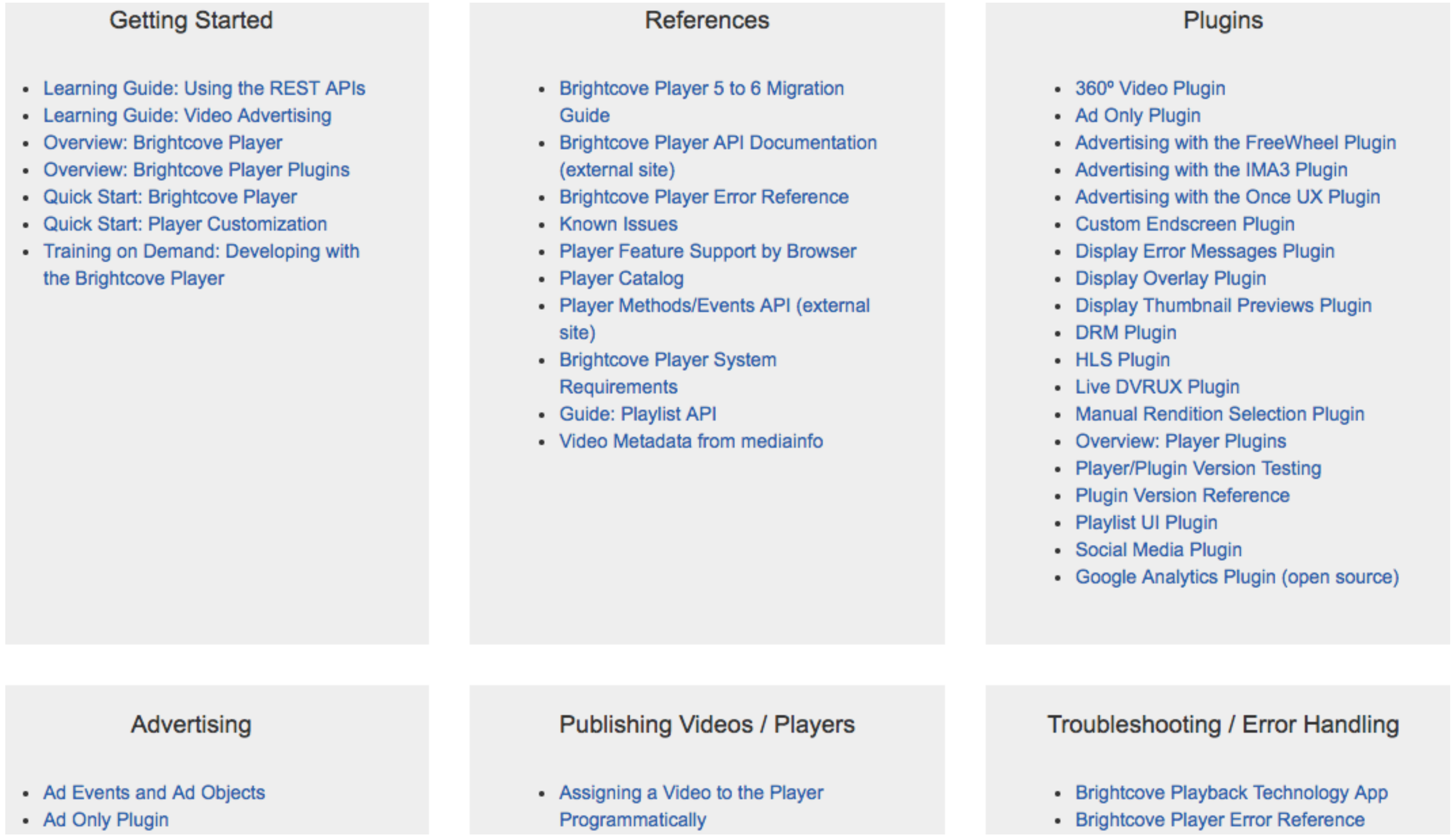
Materials for Developing with Brightcove Player Session

Latest Materials

The latest materials for the essentials (beginning course) are in the **v3-5x-updates-Feb2016 > essentials** folder

Brightcove Player Documentation

- https://support.brightcove.com/brightcove-player-developer



Getting Started

- Learning Guide: Using the REST APIs
- Learning Guide: Video Advertising
- Overview: Brightcove Player
- Overview: Brightcove Player Plugins
- Quick Start: Brightcove Player
- Quick Start: Player Customization
- Training on Demand: Developing with the Brightcove Player

References

- Brightcove Player 5 to 6 Migration Guide
- Brightcove Player API Documentation (external site)
- Brightcove Player Error Reference
- Known Issues
- Player Feature Support by Browser
- Player Catalog
- Player Methods/Events API (external site)
- Brightcove Player System Requirements
- Guide: Playlist API
- Video Metadata from mediainfo

Plugins

- 360° Video Plugin
- Ad Only Plugin
- Advertising with the FreeWheel Plugin
- Advertising with the IMA3 Plugin
- Advertising with the Once UX Plugin
- Custom Endscreen Plugin
- Display Error Messages Plugin
- Display Overlay Plugin
- Display Thumbnail Previews Plugin
- DRM Plugin
- HLS Plugin
- Live DVRUX Plugin
- Manual Rendition Selection Plugin
- Overview: Player Plugins
- Player/Plugin Version Testing
- Plugin Version Reference
- Playlist UI Plugin
- Social Media Plugin
- Google Analytics Plugin (open source)

Advertising

- Ad Events and Ad Objects
- Ad Only Plugin

Publishing Videos / Players

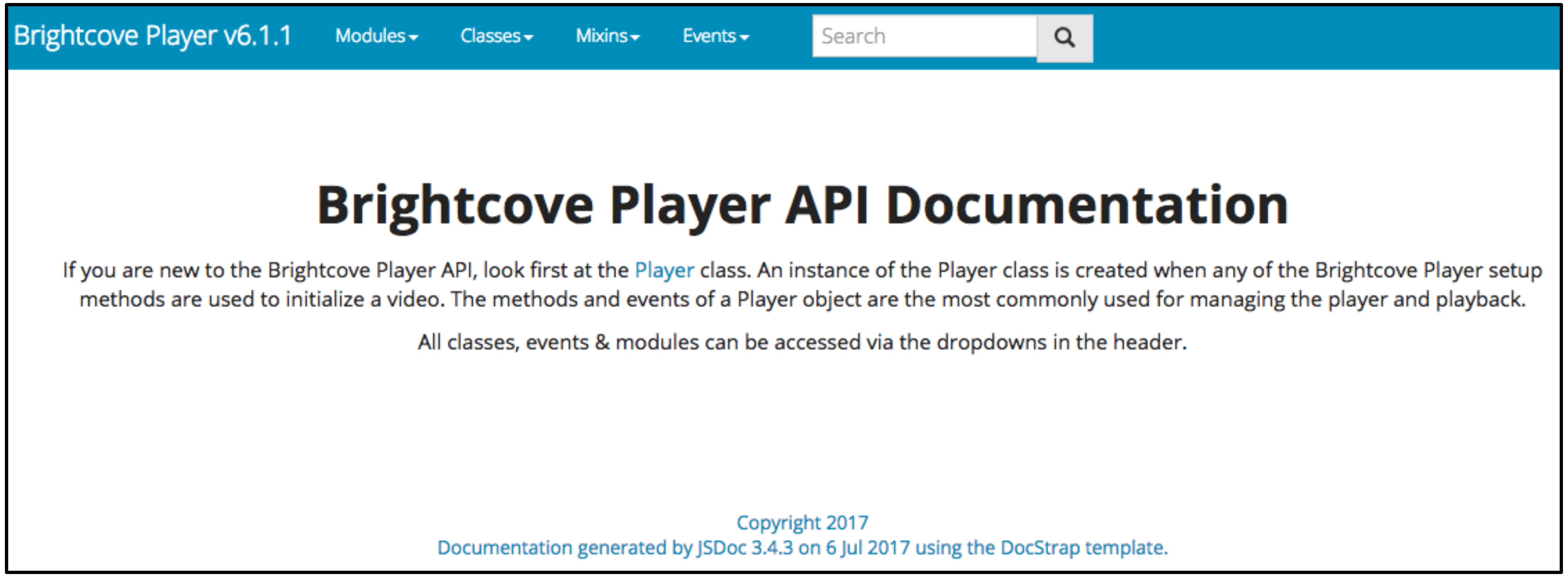
- Assigning a Video to the Player Programmatically

Troubleshooting / Error Handling

- Brightcove Playback Technology App
- Brightcove Player Error Reference

Brightcove Player API Documentation

- https://brightcovelearning.github.io/Brightcove-API-References/brightcove-player/current-release/index.html



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 dropdowns in the header.

Copyright 2017
Documentation generated by JSDoc 3.4.3 on 6 Jul 2017 using the DocStrap template.

Using JavaScript with Brightcove Player

JavaScript Code Dilemma

- Purpose of this session is to teach Brightcove Player API code
  - Decided it is not appropriate to suggest too many best practices in JavaScript

- Good pattern to use is a basic version of the Module pattern
  - Keeps variables out of the global name space to avoid collisions with other scripts used in the page
  - All variable initialized at the top to make it easier to find them
  - Allows you to have both public and private data/functions
  - Used in numerous document solutions

API Is Event Driven

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

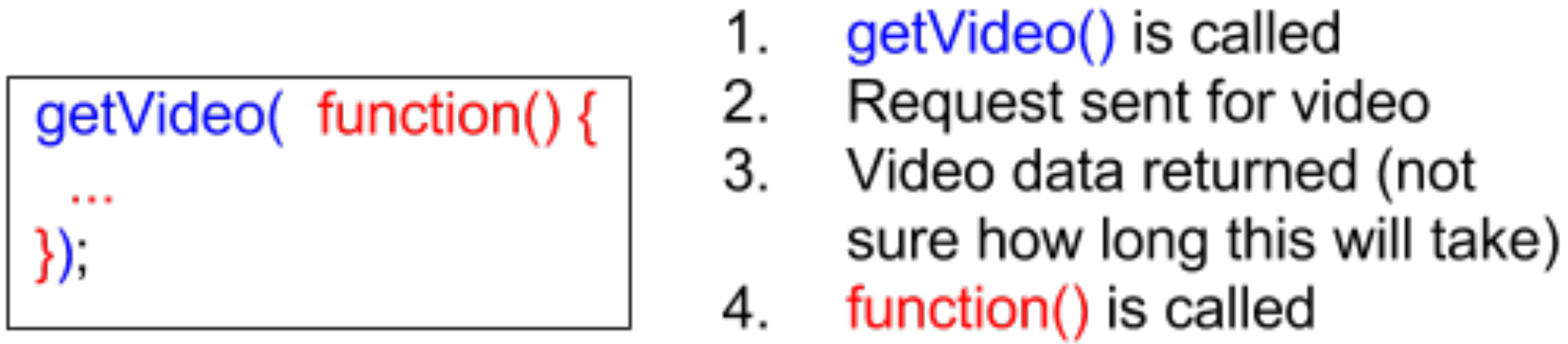
```
function foo() {
  player = this;
  player.loadVideo(123);
  player.play();
}
```

```
videojs("video").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

```
getVideo( function(){ … })
```

- Function declaration (“normal way”)
  - Loads before any code is executed

```
function foo() { … }
```

- Function expression
  - Loads only when the interpreter reaches that line of code

```
var foo = function() { … }
```

Callback Function Implementations

Anonymous Function – function definition is the argument of the callback function

```
videoPlayer.getVideo(function(videoDTO) {
 document.getElementById("displayName").
       innerHTML = videoDTO.displayName;
});
```

Callback Function Implementations (cont)

Function Declaration

```
videoPlayer.getVideo(onGetVideo);

function onGetVideo(videoDTO) {
 document.getElementById("displayName").
       innerHTML = videoDTO.displayName;
};
```

Callback Function Implementations (cont)

Function Expression

```
var onGetVideo = function(videoDTO) {
 document.getElementById("displayName").
       innerHTML = videoDTO.displayName;
};

videoPlayer.getVideo(onGetVideo);
```

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("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: //brightcovelearning.github.io/Brightcove-API-References/brightcove-player/current-release/Player.html#toc6__anchor

- Method example

```
myPlayer.play();
```

Player Events

- Docs: //brightcovelearning.github.io/Brightcove-API-References/brightcove-player/current-release/Player.html#toc120__anchor
- 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 initialize the player

```
videojs("myPlayerID").on('loadedmetadata',function(){
        var myPlayer = this;
        myPlayer.play();
});
```

**Most likely NOT necessary to do this as you could use autoplay to immediately play video
**The need to use the event for player initialization is browser dependent

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

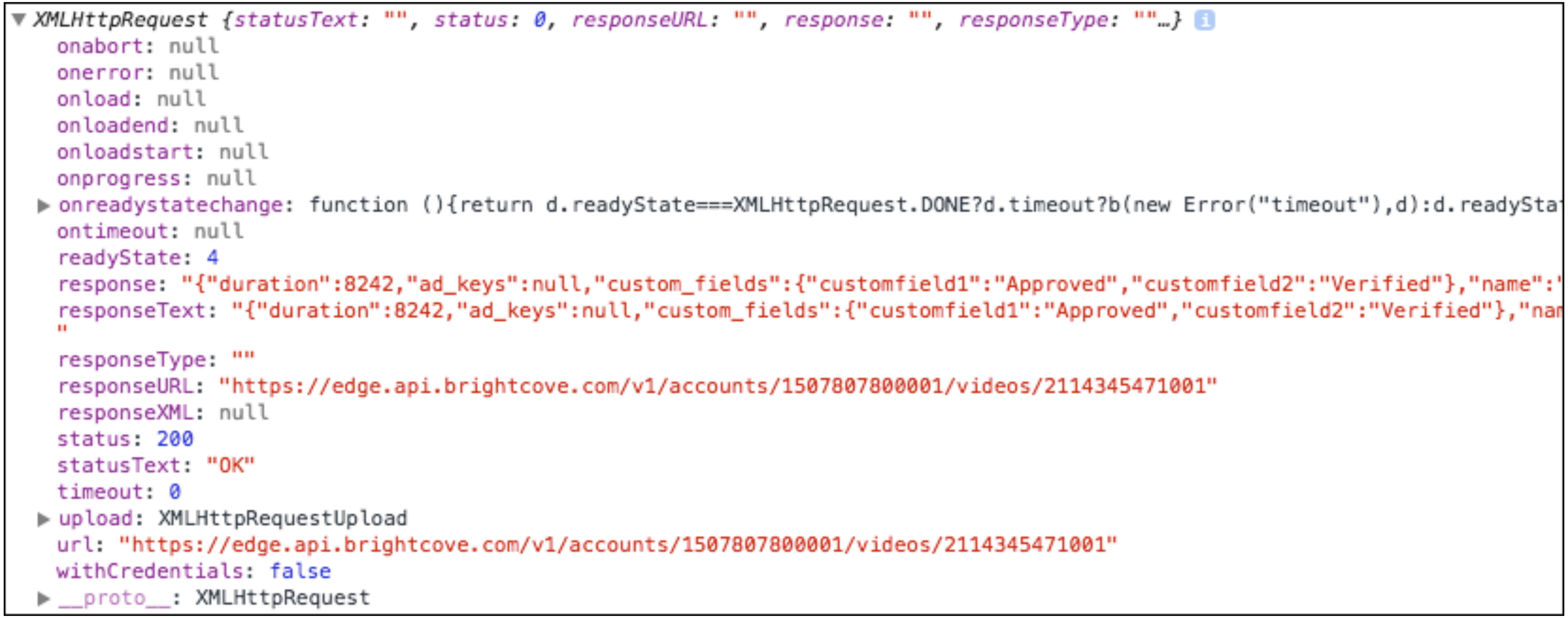
- Player Catalog is a helper library for making requests to the Video Cloud catalog
  - The catalog makes it easy to get information on Video Cloud media and loads them into a player

- Currently three methods

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

Returned Object from getVideo()

- Catalog returns an object of type XMLHttpRequest



```
▼ XMLHttpRequest {statusText: "", status: 0, responseURL: "", response: "", responseType: ""…}
    onabort: null
    onerror: null
    onload: null
    onloadend: null
    onloadstart: null
    onprogress: null
  ▶ onreadystatechange: function (){return d.readyState===XMLHttpRequest.DONE?d.timeout?b(new Error("timeout"),d):d.readySta
    ontimeout: null
    readyState: 4
    response: "{"duration":8242,"ad_keys":null,"custom_fields":{"customfield1":"Approved","customfield2":"Verified"},"name":'
    responseText: "{"duration":8242,"ad_keys":null,"custom_fields":{"customfield1":"Approved","customfield2":"Verified"},"nar
    "
    responseType: ""
    responseURL: "https://edge.api.brightcove.com/v1/accounts/1507807800001/videos/2114345471001"
    responseXML: null
    status: 200
    statusText: "OK"
    timeout: 0
  ▶ upload: XMLHttpRequestUpload
    url: "https://edge.api.brightcove.com/v1/accounts/1507807800001/videos/2114345471001"
    withCredentials: false
  ▶ __proto__: XMLHttpRequest
```

Task 2: Dynamically Loading and Playing a Video

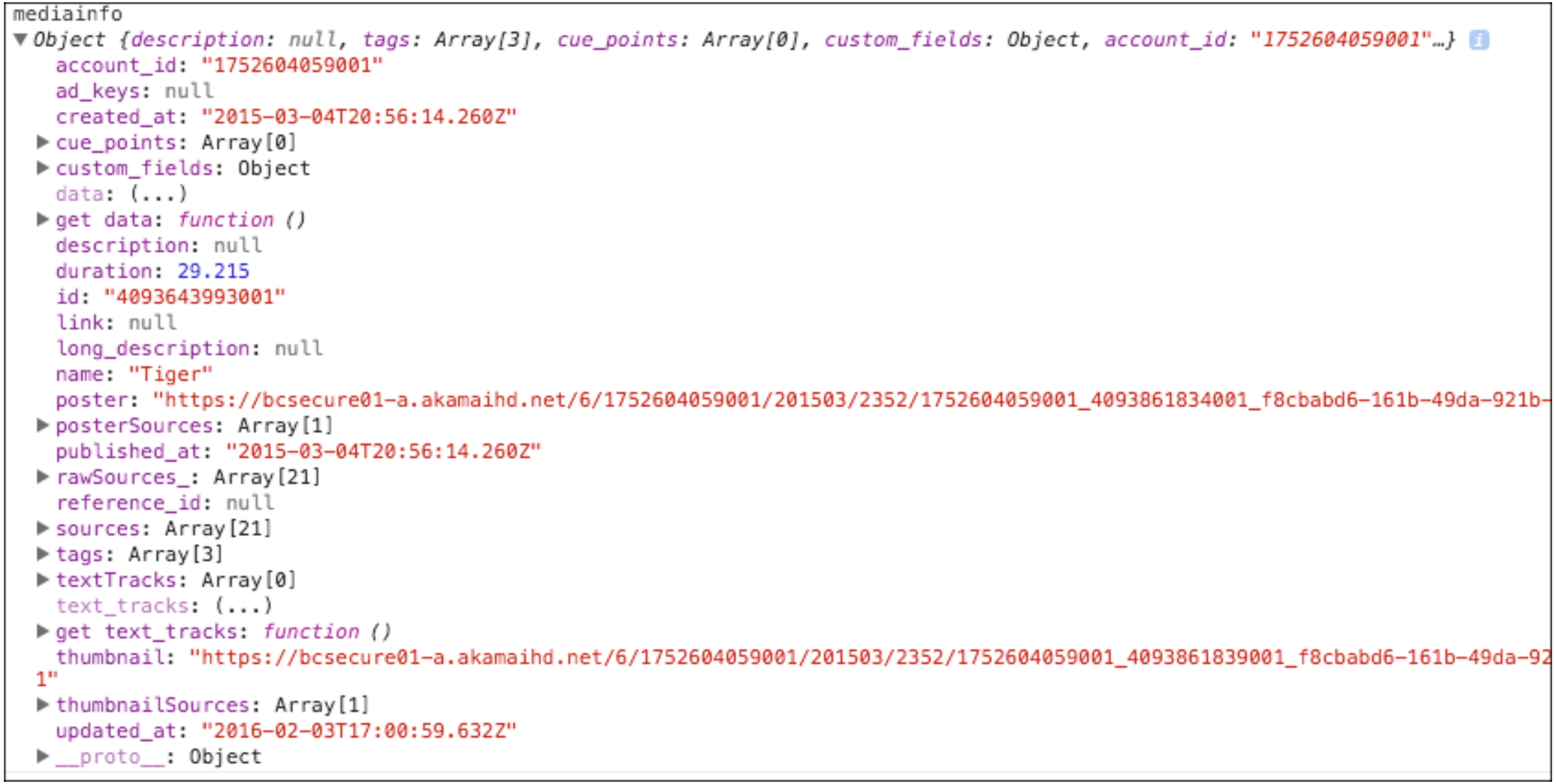
Using the mediainfo Property

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

mediainfo Property

- The mediainfo property is an object which contains information on the current media in the player
- The property 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;
```

Task 3: Display Video Information in the HTML Page

**Uses the ready() event/method

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

Using the iframe Player Implementation

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

Advantages of 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='//players.brightcove.net/921483702001/a5f0f07c-ce3b-48a4-af02-f5f6c38546ac_default/index.html
?videoId=4341341161001' ...></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 Video from iframe Parent*
  - //docs.brightcove.com/en/player/brightcove-player/samples/listen-for-play-button.html
- Example doc: *Implementing Playlists Programmatically: Passing video ID on URL page request for iframe*
  - //support.brightcove.com/implementing-playlists-programmatically#Set_initial_video

Task 4: Changing the Video in an iframe Player Implementation

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

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

- Brightcove has released, and continues to release, plugins
  - 360 Video
  - Ad Only Plugin
  - Advertising with FreeWheel (beta)
  - Advertising with IMA3
  - Advertising with OnceUX
  - Custom Endscreens
  - Display Errors
  - Display Overlay
  - DRM
  - HLS
  - Live DVRUX
  - Playlist UI
  - Quality Selection
  - Social Media

Brightcove Plugins Loaded by Default

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

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

Implementing Plugins Using curl Statements

- Can configure the player, and associated plugins, using the Player Management API
- Details on using curl not part of this course

```
curl --header "Content-Type: application/json" --user $EMAIL --request PATCH \
  --data '{
    "stylesheets": ["http://…/plugin-dev.css"
    ],
    "scripts": ["http://…/plugin-dev.js"
    ],
    "plugins": [{ "name": "pluginDev", "options": {"overlayText": "This …"}
    }]
  }' \
  https://players.api.brightcove.com/v1/accounts/$ACCOUNT_ID/players
     /$PLAYER_ID/configuration
```

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

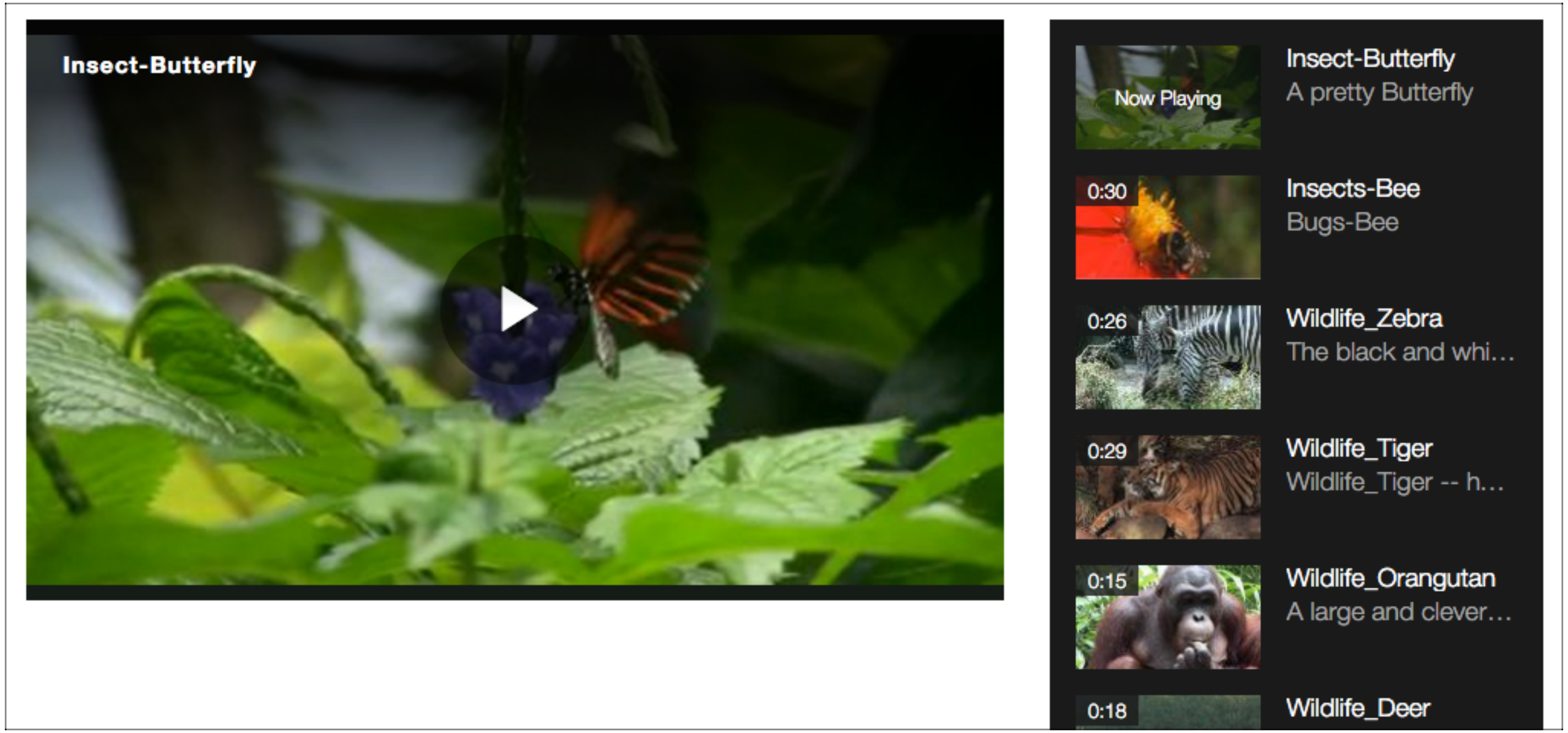
Task 6 CodePen: http://codepen.io/team/bcls/pen/PNEWQJ

Using Playlists

Use Case: Allow users to select a video to watch from a playlist

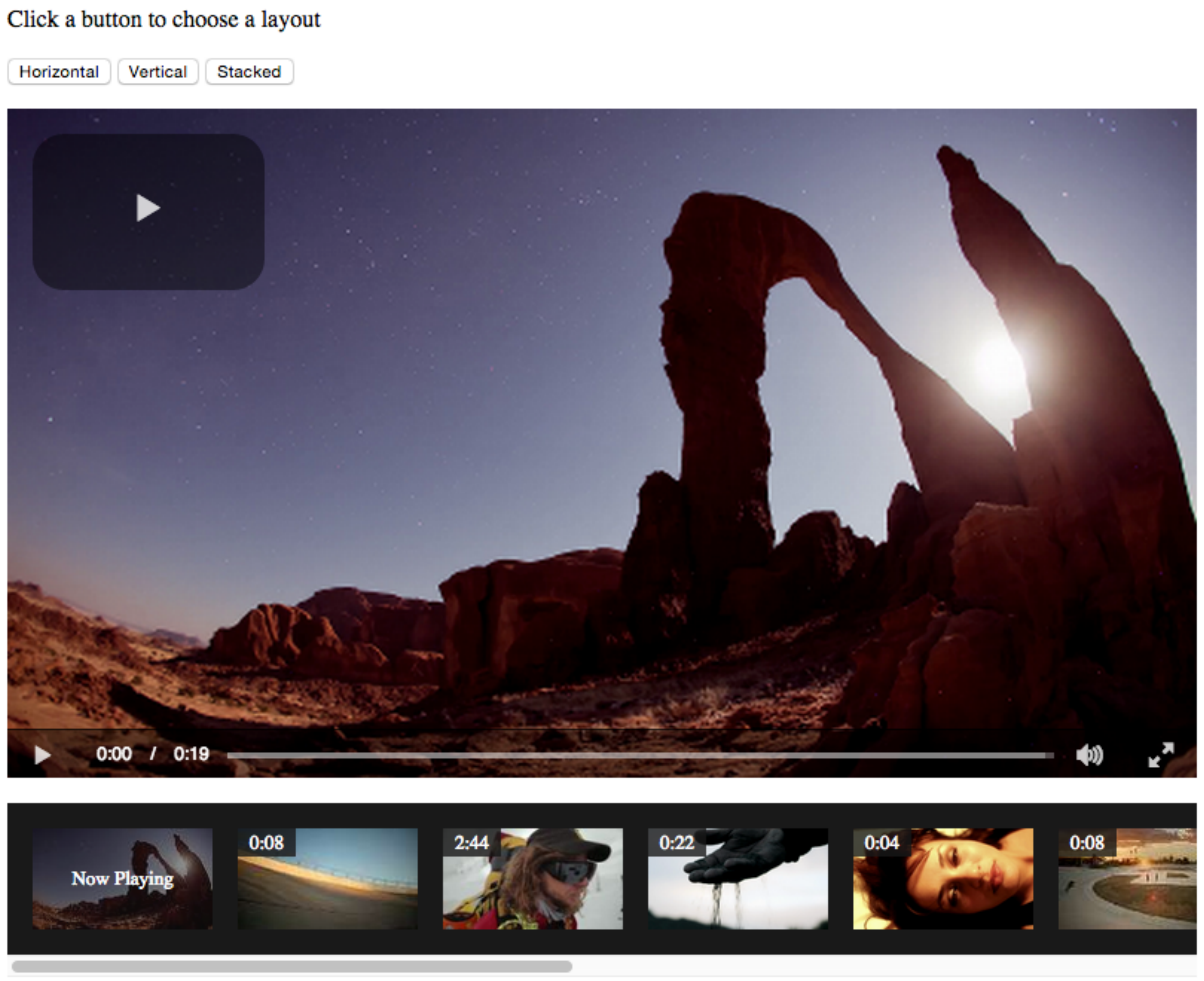
Playlists

- Create playlists in Studio's **Media** module
- Default playlist appearance



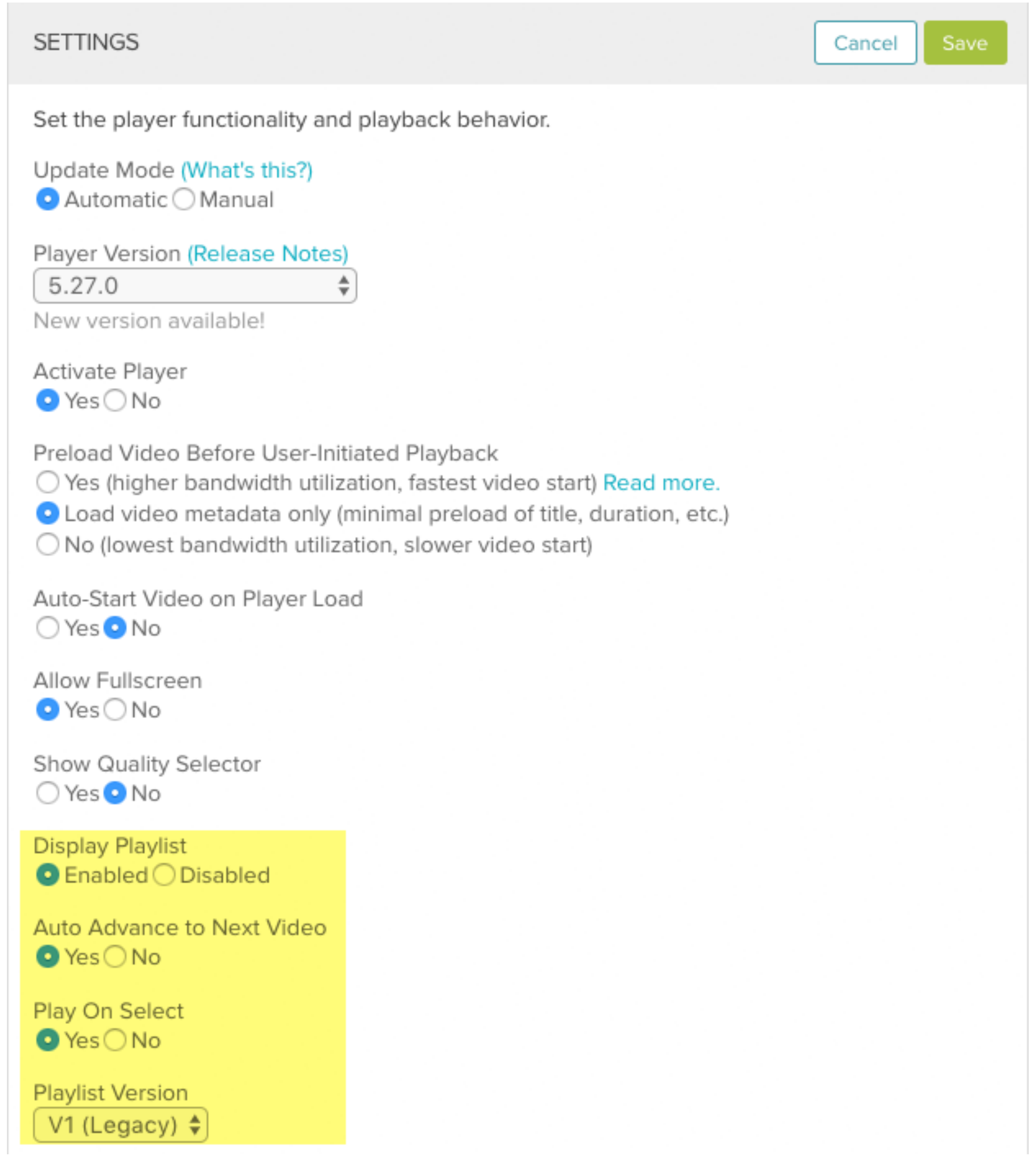
Other Layouts Possible

- http://solutions.brightcove.com/pcosta/showcase/playlist-changer.html



Enable Playlists in Studio

- **Players** module > **Settings** section



SETTINGS

Cancel Save

Set the player functionality and playback behavior.

Update Mode (What's this?)
[✓ Automatic] Manual

Player Version (Release Notes)
5.27.0
New version available!

Activate Player
[✓ Yes] No

Preload Video Before User-Initiated Playback
Yes (higher bandwidth utilization, fastest video start) Read more.
[✓ Load video metadata only (minimal preload of title, duration, etc.)]
No (lowest bandwidth utilization, slower video start)

Auto-Start Video on Player Load
Yes [✓ No]

Allow Fullscreen
[✓ Yes] No

Show Quality Selector
Yes [✓ No]

Display Playlist
[✓ Enabled] Disabled

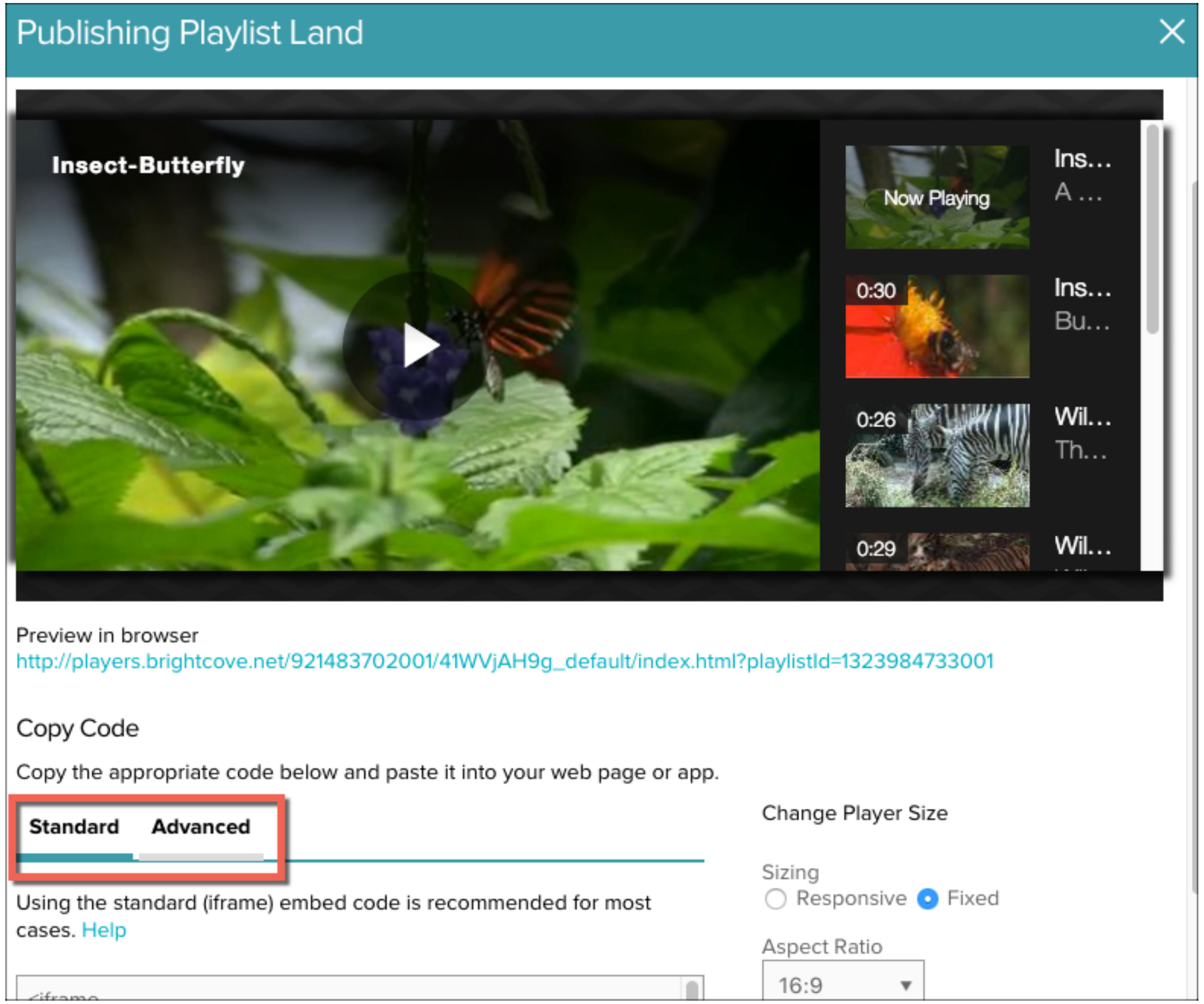
Auto Advance to Next Video
[✓ Yes] No

Play On Select
[✓ Yes] No

Playlist Version
V1 (Legacy)

Associate Playlist with Enabled Player

- Select playlist in Media module
- Select an enabled player
- Use desired code implementation



Preview in browser
http://players.brightcove.net/921483702001/41WVjAH9g_default/index.html?playlistId=1323984733001

Copy Code

Copy the appropriate code below and paste it into your web page or app.

Using the standard (iframe) embed code is recommended for most cases. Help

Change Player Size

Sizing
○ Responsive [● Fixed]

Aspect Ratio
16:9

In-Page Code Needs <ol>

- If using in-page code you must
  - Place the HTML ordered list where you want the playlist to appear
    - Must use vjs-playlist as class

```
<ol class="vjs-playlist"></ol>
```

  - Style player and playlist as you choose

```
.video-js {
  float: left; margin: 15px; width: 640px; height: 380px;
}
.vjs-playlist {
  width: 250px; float: left; margin: 15px;
}
```

iframe Code Needs Styling

```
 <style>
    iframe {
      width: 88%;
      min-height: 360px;
    }
  </style>

  <iframe src='//players.brightcove.net/921483702001/c517d6aa-
c198-469b-92c4-6944d80dd143_default/index.html?
playlistId=1323984733001' allowfullscreen webkitallowfullscreen
mozallowfullscreen></iframe>
```

Task 7: Display a Playlist

Task 7 CodePen: http://codepen.io/team/bcls/pen/oxpaaO brightcove

Thank You!

Matt Boles

mboles@brightcove.com