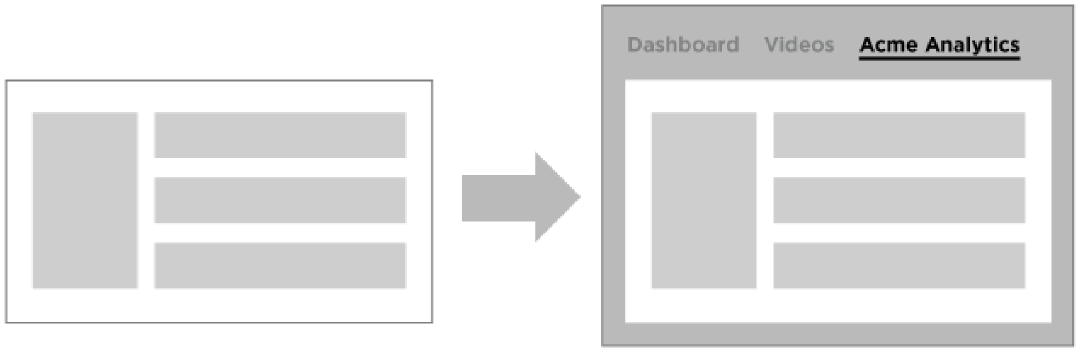
Creating Brightcove Studio Modules

Brightcove Studio Modules allow developers to easily add functionality to Brightcove Studio. Using a simple Javascript API, any web application can be fully integrated into the Brightcove end-user experience.

For example, consider "Acme Analytics," a fictional video analytics service. As a studio module, Acme Analytics can expose some or all of its features to Brightcove users in a seamless manner without additional login or registration steps.



Authentication

Brightcove uses OAuth [1] and other standard web technologies to incorporate modules into the Brightcove Studio experience. As a module developer, you are responsible for implementing core business logic and workflows, while the Studio Module SDK is responsible for user authentication, global navigation, and global configuration.

Registering a module

Studio Modules must be registered with Brightcove. To register a module, please provide the following information:

1. The name of your module. This value will appear in the navigation bar inside Brightcove Studio. Example: Acme Analytics
2. A description of your module. This value may be used for marketing purposes. Example: Acme Analytics for Brightcove Studio helps you track video engagement across leading social media platforms.
3. The landing page of your module. This must be a secure URL. Example:

https://bc.acme.com/index.html

After registering, Brightcove will issue you a Module ID.

The Studio Module SDK

The Studio Module SDK enables your standalone web application to function as a Brightcove Studio Module. It consists of a JavaScript file and a CSS file.

studio-module.js

This JavaScript file manages OAuth 2.0 implicit flow authentication, exposes user and account information, and provides an interface to various Brightcove APIs.

studio-module.css

This CSS file, based on Twitter Bootstrap [2], allows modules to adopt a standard look and feel. It also provides CSS rules for enabling responsive display on phones and tablets.

Creating a module

Your module can be a single-page web application [3] or a traditional, multi-page web site. Brightcove recommends using a single-page architecture for a faster and more seamless user experience.

Below is a boilerplate HTML document demonstrating how to import the necessary JavaScript and CSS files and instantiate your module. This example loads JavaScript files synchronously at the end of the <body> tag so as not to block rendering of the UI, however you are free to implement other loading strategies, including use of the async [4] attribute. Note all assets should be loaded via HTTPS.

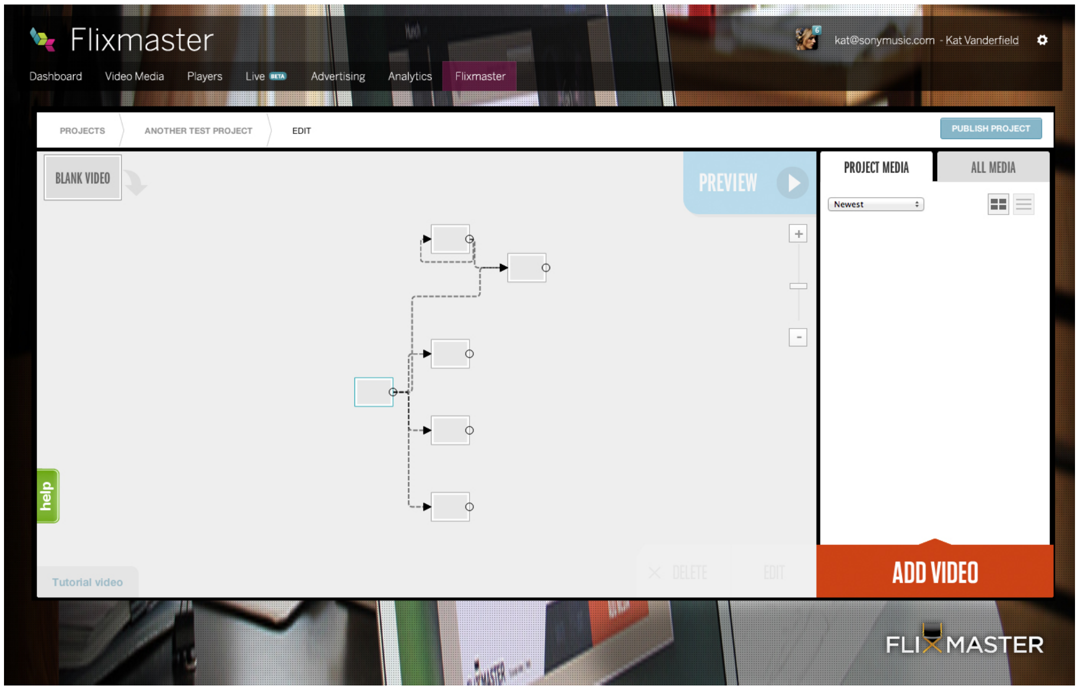
```
<!DOCTYPE html>
<html>
<head>
  <meta content="text/html; charset=utf-8" http-equiv="Content-Type">
  <title>Acme Module</title>

  <!-- Import Brightcove CSS -->
  <link rel="stylesheet"
    href="https://xcloud.brightcove.com/public/assets/css/studio-module.css" />

  <!-- Import your supplemental CSS -->
  <link rel="stylesheet" href="https://acme.com/bc/css/acme-module.css" />
</head>

<body>
  <div id="bc-nav">
    <!-- Injected by SDK -->
  </div>

  <div>Hi from Acme Module!</div>

  <!-- Import your JS -->
  <script src="https://acme.com/bc/js/acme-module.js"></script>

  <!-- Import Brightcove JS -->
  <script src="https://xcloud.brightcove.com/public/assets/js/studio-module.js"></script>

  <!-- Instantiate the module -->
  <script>
  // This function runs as soon as the SDK is loaded. It receives a StudioModule object
  window.onStudioModuleLoad = function(studioModule) {

    // The "init" event indicates the StudioModule object is ready to receive commands
    studioModule.on("init", function(data) {
      // data contains a "user" object and an "error" object (or null)
    });

    // The "auth" event fires whenever new credentials are issued
    studioModule.on("auth", function(data) {
      // data contains a "token" object and an "error" object (or null)
    });

    // The "init" command initializes your module with the given options
  studioModule.init({
   moduleId: "b67fde96-52af-4899-a44f-567e94167d3c"
  });

 };
 </script>
</body>

</html>
```

Global Navigation

The first element inside the <body> tag of your document should be an empty <div> with the ID “bc-nav”:

```
 <div id="bc-nav"></div>
```

The Studio Module SDK will inject a global navigation bar into this <div>. Everything else is up to you! The following screenshot illustrates the effect:



Notice how background styles can show through the global header.

The StudioModule object

The StudioModule object is the primary interface to Brightcove. You can obtain a reference to the StudioModule object via the global function onStudioModuleLoad, which is automatically called by studio-module.js:

```
window.onStudioModuleLoad = function(studioModule) {
};
```

StudioModule Events

“init ”

The “init” event fires on the StudioModule object after user data is loaded (but before an authorization token is issued). The callback function takes a single argument, data, which contains a user object. In the event of an error, data will contain an error object:

```
studioModule.on("init", function(data) {
  // handle an error
  if (data.error) {          myApp.handleError(data.error.code);
      return;
  }

  // otherwise handle the user data
  myApp.loadProfile(data.user);
});
```

The user object contains both user and account information:

```
  {
      email_address: "john.doe@example.com",
      first_name: "John",
      last_name: "Doe",
      id: "99475571628",
    locale: "en_US",
    timezone: "AET",
    accounts: [
      {
        id: "830395163001",
        name: "John Doe Enterprises",
        trial_account: false
      },
      {...}
    ],
    currentAccount: {...}
  }
```

The error object contains an error code and a human-readable message. For example:

```
{
    code: "NOT_LOGGED_IN",
    message: "The user is not logged in."
  }
```

Possible error codes for the "init" event are "NOT_LOGGED_IN", "NO_VALID_ACCOUNT", and "SYSTEM_ERROR".

"auth "

The "auth" event fires on the StudioModule object each time a new access token is granted to the module. The callback function takes a single argument, data, which contains a token object. In the event of an error, data will contain an error object.

This event is useful only if you intend to handle access tokens yourself. Otherwise, use the api method of the StudioModule object.

```
studioModule.on("auth", function(data) {
    // handle an error
    if (data.error) {          myApp.handleError(data.error.code);
```

```
            return;
          }
          // otherwise handle the access token
          myApp.handleAuthToken(data.token.access_token);
        });
```

The token object contains properties of the OAuth credential:

```
      {
          access_token: "AIqq7znVV4Wsqiqhdw2iKdkjiz5w-..."
          expires_in: 300
          token_type: "Bearer"
        }
```

The error object contains an error code and a human-readable message. For example:

```
      {
          code: "TOKEN_REQUEST_ERROR",
          message: "Access denied."
        }
```

Possible error codes for the “auth” event are “TOKEN_REQUEST_ERROR”.

Brightcove uses the OAuth 2.0 implicit flow for Studio Modules, which means the “auth” event may fire as often as every five minutes. If you only want to handle this event one time, call module.once instead of module.on.

“home ”

The “home” event fires when the user selects the current module in the global navigation bar. The HTML document is not reloaded; rather, you should use this event to change the program state.

```
studioModule.on("home", function() {        myApp.goHome();    });
```

“logout ”

The “logout” event fires when the user’s session expires (after 24 hours), or if the user signs out in another window or tab.

```
studioModule.on("logout", function() {        myApp.handleSessionEnd();    });
```

StudioModule Methods

init(config)

The init method attempts to initialize your module. It accepts a configuration object with a moduleId property:

```
studioModule.init({
    moduleId: "b67fde96-52af-4899-a44f-567e94167d3c"
  });
```

The StudioModule object fires an “init” event after this method is called.

api ()

The api method returns a reference to the BCApi object. This object can be used to make calls to Brightcove APIs, and also implements the OAuth 2.0 implicit flow.

```
var bcapi = studioModule.api();
  bcapi.call(...);
```

getUser ()

The getUser method returns a reference to the user object or null if no user exists. See the “init” event, above, for more information about the user object.

```
var user = studioModule.getUser();
  myApp.setUser(user);
```

This methods returns null if you call it before the “init” event has fired on the StudioModule object.

login ()

The login method redirects the user to the Brightcove login screen.

```
studioModule.login();
```

The BCApi object

The BCApi object, obtained by calling the api method of StudioModule, hides some of the complexity of calling Brightcove APIs.

BCApi Methods

call(parameters, onSuccess, onError)

The call method calls a Brightcove API service using the OAuth token issued to the module.

```
var params = {
    path: "https://api.brightcove.com/services/library",
    method: "search_videos"
  };
  var handleData = function (data) {
    console.log(data);
  };
  var handleError = function (error) {
    console.error(error);
```

```
          };
          var bcapi = studioModule.api();
          bcapi.call(params, handleData, handleError);
```

getToken ()

The getToken method returns an object with the most recent OAuth token or null if no token exists. See the "auth" event, above, for a description of this object.

```
        var token = studioModule.api().getToken();
```

This method returns null if you call it before the "auth" event has fired on the StudioModule object.

Design considerations

Although your module runs as a standalone web application, it should appear to be a section within Brightcove Studio. To this end, the Brightcove Studio Module SDK injects a global banner and navigation bar into the first element of the <body> tag of your document. This element should be identified as "bc-nav":

```
<body>
    <div id="bc-nav">
      <!-- Injected by SDK -->
    </div>
    <div>
      <!-- Everything else -->
    <div>
  </body>
```

Be careful not to override styles inside the "bc-nav" element.

Using Brightcove styles

The file studio-module.css includes a Brightcove-themed version of Twitter Bootstrap [5]. In addition to standard Bootstrap classes like “btn” and “btn-group,” it includes some Brightcove-specific components and styles, listed here. To roll your own theme, see Using your own styles, below.

body.bc-background

Applies a Brightcove theme to the body background. Usage:

```
<body class="bc-background">
    <div id="bc-nav"></div>      <div> ... </div>
  </body>
```

.bc-canvas

Applies padding to the main content area. Usage:

```
<body class="bc-background">
    <div id="bc-nav"></div>      <div class="bc-canvas">
       <div class="row-fluid"> … </div>
    </div>    </body>
```

.bc-panel, .bc-panel-head, .bc-panel-content

Creates a Brightcove-style panel with an arbitrary width. Usage :

```
<div class="span4">
    <div class="bc-panel">
      <div class="bc-panel-head">
        <h2>Videos</h2>
      </div>        <div class="bc-panel-content">
        ...
        </div>
      </div>
    </div>
```

.bc-panel-controls

Floats panel controls to the right of a panel headline.

```
  <div class="bc-panel-head">
      <div class="bc-panel-controls"> ... </div>
      <h2>Videos</h2>
    </div>
```

.bc-panel-inset

Creates a white, padded box inside bc-panel-content.

```
  <div class="span4">
      <div class="bc-panel">
        <div class="bc-panel-head">
          <h2>Videos</h2>
        </div>        <div class="bc-panel-content">
          <div class="bc-panel-inset"> ... </div>
          <div class="bc-panel-inset"> ... </div>
        </div>
      </div>
    </div>
```

.bc-img

Adds a Polariod-style border to an image or video element.

```
<img class="bc-img" src="..." />
```

input.bc-checkbox

Creates a Brightcove-style checkbox field. Note <label> is required.

```
<input class="bc-checkbox" type="checkbox" name="fruit" value="apple" id="apple"
    checked="checked"><label for="apple">Apple</label>
```

input.bc-radio

Creates a Brightcove-style radio field. Note <label> is required.

```
<input class="bc-radio" type="radio" name="fruit" value="apple" id="apple"
    checked="checked"><label for="apple">Apple</label>
```

input.bc-search

Adds a search icon to a text field.

```
<input class="bc-search" type="text" name="query" value="Search ..."/>
```

input.bc-yes

Adds a green success icon to a text field.

```
<input class="bc-yes" type="text" name="email"/>
```

input.bc-no

Adds a red error icon to a text field.

```
<input class="bc-no" type="text" name="email"/>
```

button.bc-toggle-on, button.bc-toggle-off

Creates a Brightcove-style toggle switch.

```
<button class="bc-toggle-on"/>
  <button class="bc-toggle-off"/>
```

button.bc-toggle-yes, button.bc-toggle-no

Creates a Brightcove-style toggle switch with red and green icons.

```
<button class="bc-toggle-yes"></button>
  <button class="bc-toggle-no"></button>
```

.bc-tag

Creates a Brightcove-style tag (e.g. for a tag cloud).

```
<span class="bc-tag">Apples</span>
```

ul.bc-list

Creates a Brightcove-style list.

```
<ul class="bc-list">
    <li>Moe</li>
    <li>Larry</li>
    <li>Curly</li>
  </ul>
```

.bc-pager, .bc-pager-alternate

Creates a Brightcove-style page flipper.

```
<!-- green -->
  <div class="bc-pager">
    <a href="page1.html">Previous</a>
    <a href="page3.html">Next</a>
  </div>
  <!-- black -->
  <div class="bc-pager-alternate">
    <a href="page1.html">Previous</a>
    <a href="page3.html">Next</a>
  </div>
```

Using your own styles

If you prefer to roll your own theme, you can import a “light” version of the Brightcove stylesheet:

```
<link rel="stylesheet"
    href="https://xcloud.brightcove.com/public/assets/css/studio-module-light.css" />
```

This version includes only styles required to render the banner and navigation bar.

1. http://oauth.net/
2. http://twitter.github.com/bootstrap/
3. http://en.wikipedia.org/wiki/Single-page_application
4. http://davidwalsh.name/html5-async
5. http://twitter.github.com/bootstrap/