ULIZA Player (HTML5)

インテグレーションガイドv1.29.0

はじめに

本書では、ULIZA Player (HTML5)のプレイヤーの組み込み方法を説明します。

参考資料

- ULIZA Player (HTML5)ユーザーガイド
- ULIZA Player (HTML5) APIリファレンス
- ULIZA Google Castレシーバー
- ULIZA Video Analytics (Cloud)ユーザーガイド
- ULIZA Player (HTML5) WebViewインテグレーションガイド
- ULIZA Player (HTML5)プレイリストユーザーガイド

> **補足**
>
> 本書内では、ULIZA Player (HTML5)ユーザーガイドは「ユーザーガイド」、ULIZA Player (HTML5) APIリファレンスは「APIリファレンス」と表記します。

実装方法

プレイヤーを埋め込むWEBページの文字コードはUTF-8を推奨します。UTF-8以外の場合、プレイヤーが正しく動作しない場合があります。本章の実装例は、ページにプレイヤーを埋め込むの実装がされている前提とします。

ページにプレイヤーを埋め込む

以下の手順を実行します。

1. プレイヤーをページに読み込みます。

```html
<link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
<script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
```

2. ページにvideo要素またはaudio要素を記述し、以下の属性を指定します。

   - id属性： プレイヤー識別子
   - class属性：「ulizahtml5」文字列

   > **注意**
   >
   > source要素でメディアソース（コンテンツのURLおよびコンテンツのMIME-type）を指定する場合、プレイヤーが正しく動作しない場合があります。次に示す例のように、プレイヤーオプションでメディアソースを指定します。

3. ページのviewportを設定します。

   ULIZA Player (HTML5)推奨のviewport設定は以下の通りです。

```html
<meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
```

MP4コンテンツを再生する

以下のプレイヤーオプションを指定します。

- sources： メディアソースのObjectの配列

実装例

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
    <link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
  </head>
  <body>
    <video id="video1" class="ulizahtml5"></video>
    <script type="text/javascript">
      ulizahtml5('video1', {
        sources: [{
          src: 'https://host/path/content.mp4',
          type: 'video/mp4'
        }]
      });
    </script>
```

```
  </body>
</html>
```

HLSコンテンツを再生する

以下のプレイヤーオプションを指定します。

- sources： メディアソースのObjectの配列
- html5.hlsjsConfig.authDomain： 認証情報送受信許可ドメイン

必要に応じ、認証情報の送受信を許可するドメインを指定してください。詳細については、「APIリファレンス」のauthDomainの説明を参照してください。

実装例

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
    <link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
  </head>
  <body>
    <video id="video1" class="ulizahtml5"></video>
    <script type="text/javascript">
      ulizahtml5('video1', {
        sources: [{
          src: 'https://host/path/testhls.m3u8',
          type: 'application/x-mpegURL'
        }],
        html5: {
          hlsjsConfig: {
            authDomain: ['www.example1.com', 'www.example2.com']
          }
        }
      });
    </script>
  </body>
</html>
```

DASHコンテンツを再生する

以下の手順を実行します。

1. ulizahtml5-dash1プラグインをページに読み込みます。ulizahtml5-dash1プラグインの読み込みは、プレイヤーの読み込み後にします。

   ```html
   <script src="https://host/path/ulizahtml5-dash1.min.js" type="text/javascript"></script>
   ```

2. 以下のプレイヤーオプションを指定します。

   - sources： メディアソースのObjectの配列
   - html5.keySystem.['com.widevine.alpha']： DASH (Widevine)のライセンス要求URL
   - html5.keySystem.['com.microsoft.playready']： DASH (PlayReady)のライセンス要求URL

   DASH (Widevine)のライセンス要求URLとDASH (PlayReady)のライセンス要求URLの両方を指定する場合、プレイヤーは再生環境に応じ適切なライセンス要求URLを判断し選択します。また、いずれかのライセンス要求URLだけを指定することもできます。

   必要に応じライセンス要求URLには予めパラメータをセットしてください。詳細については、「APIリファレンス」のcom.widevine.alpha, com.microsoft.playreadyの説明を参照してください。

実装例

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
    <link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
    <script src="https://host/path/ulizahtml5-dash1.min.js" type="text/javascript"></script>
  </head>
  <body>
    <video id="video1" class="ulizahtml5"></video>
    <script type="text/javascript">
      ulizahtml5('video1', {
        sources: [{
          src: 'https://host/path/dash.mpd',
          type: 'application/dash+xml'
        }],
        html5: {
          keySystem: {
            'com.widevine.alpha': 'https://host/path/getWvLisence',
            'com.microsoft.playready': 'https://host/path/getPrLisence'
          }
        }
      });
    </script>
  </body>
</html>
```

オーディオを再生する

オーディオを再生するには、ページにプレイヤーを埋め込むをaudio要素で実行します。

実装例

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
    <link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
  </head>
  <body>
    <audio id="audio1" class="ulizahtml5"></audio>
    <script type="text/javascript">
      ulizahtml5('audio1', {
        sources: [{
          src: 'https://host/path/content.mp4',
          type: 'video/mp4'
        }]
      });
    </script>
  </body>
</html>
```

自動再生する

以下のプレイヤーオプションを指定します。

- autoplay： 自動再生の有効／無効

以下のプレイヤーオプションは必要に応じ指定します。

- controller.unmuteButtonVisible： ミュート解除ボタンの表示／非表示

実装例

```js
ulizahtml5('video1', {
  autoplay: true,
  controller: {
    unmuteButtonVisible: true
  }
  // 他のプレイヤーオプションは省略。
});
```

ミュートで再生を開始する

以下のプレイヤーオプションを指定します。

- muted： ボリュームのミュート／非ミュート

実装例

```js
ulizahtml5('video1', {
  muted: true
  // 他のプレイヤーオプションは省略。
});
```

ULIZA Video Analytics (Cloud)視聴分析を使用する

以下のプレイヤーオプションを指定します。

- videoAnalytics.enable： ULIZA Video Analytics (Cloud)連携の有効／無効
- videoAnalytics.trackingId： ビーコンの送信先識別子
- videoAnalytics.contentName: コンテンツ識別子（ID）
- videoAnalytics.contentTitle: コンテンツタイトル

以下のプレイヤーオプションは必要に応じ指定します。

- videoAnalytics.userId： ユーザーID
- videoAnalytics.contentCategory： コンテンツカテゴリ
- videoAnalytics.defaultTrackingOptout： トラッキングオプトアウトの初期状態
- videoAnalytics.trackingOptoutButtonVisible： トラッキングオプトアウトボタンの表示／非表示
- videoAnalytics.takeOverSession : Cast接続時のセッション引き継ぎの有効／無効

実装例

```js
ulizahtml5('video1', {
  videoAnalytics: {
    enable: true,
    trackingId: 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX',
    userId: 'user_id',
    contentName: 'content_name',
    contentTitle: 'content_title',
    contentCategory: 'content_category',
    defaultTrackingOptout: false,
    trackingOptoutButtonVisible: false,
    takeOverSession: true
  }
  // 他のプレイヤーオプションは省略。
});
```

ULIZA Video Analytics (Cloud)リアクションを使用する

ULIZA Video Analytics (Cloud)視聴分析を使用するに加え、以下のプレイヤーオプションを指定します。

- videoAnalytics.enableReaction： リアクションの有効／無効
- videoAnalytics.reactionItem： リアクションの情報
- videoAnalytics.reactionItem[n].buttonId： リアクションID

以下のプレイヤーオプションは必要に応じ指定します。

- videoAnalytics.reactionItem[n].buttonVisible： リアクションボタンの表示／非表示
- videoAnalytics.reactionItem[n].iconId： ボタンアイコン
- videoAnalytics.reactionItem[n].label： リアクションのラベル
- videoAnalytics.reactionItem[n].graphVisible： リアクショングラフの表示／非表示

実装例

```js
ulizahtml5('video1', {
  videoAnalytics: {
    enable: true,
    trackingId: 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX',
    enableReaction: true,
    reactionItem: [
      {
        buttonId: 'reaction1',
        buttonVisible: true,
        iconId: 'uliza-icon-thumbs-up',
        label: 'いいね',
        graphVisible: true,
      }
    ]
  }
  // 他のプレイヤーオプションは省略。
});
```

独自のボタンでリアクション情報をカウントアップする

プレイヤーのAPIを使用して、リアクション情報をカウントアップできます。

- reactionCountUp

リアクション情報をカウントアップします。詳細は「APIリファレンス」を参照してください。

実装例

ULIZA Video Analytics (Cloud)リアクションを使用するが実装されている前提とします。

```html
<button type="button" onclick="onClickCustomButton()">カスタマイズボタン</button>
<script type="text/javascript">
  var player = ulizahtml5(省略);

  // ボタン押下時にリアクションの回数をカウントアップします。
  function onClickCustomButton() {
    player.reactionCountUp(0);
  }
</script>
```

独自のグラフを作成する

リアクショングラフはプレイヤーのAPIを使用して独自に実装できます。詳細は「APIリファレンス」を参照してください。

- getReactionGraphData

リアクショングラフ作成用のデータを取得します。詳細は「APIリファレンス」を参照してください。

リアクショングラフ作成用のデータの例は以下の通りです。

```js
[
  {
    reaction_point: 0,
    count: 0
  },
  {
    reaction_point: 1,
    count: 3
  },
  {
    reaction_point: 2,
    count: 10
  },

  // 中略

  {
    reaction_point: 99,
    count: 5
  },
  {
    reaction_point: 100,
    count: 6
  }
]
```

リアクショングラフを独自に実装するには、以下の手順を実行します。

1. グラフ領域を指定します。
2. プレイヤーがコールバックするreactionDataReceived契機にAPI getReactionGraphDataを実行します。
3. 取得したデータを元にグラフを作成します。

実装例

ULIZA Video Analytics (Cloud)リアクションを使用するが実装されている前提とします。

```html
<!-- 1. グラフ領域を指定します。 -->
<div id="reactionGraph" style="width: 640px; height: 70px;"></div>

<script type="text/javascript">
  var player = ulizahtml5( /* 省略 */ );

  // 2. プレイヤーがコールバックするreactionDataReceived契機にgetReactionGraphDataを実行します。
  player.on('reactionDataReceived', function() {
    var reactionGraphData = player.getReactionGraphData(0);
    // 3. 取得したデータを元にグラフを作成します。
  });
</script>
```

ビーコンを送信する

以下のプレイヤーオプションを指定します。

- beacon.url： ビーコン送信URL

以下のプレイヤーオプションは必要に応じ指定します。

- beacon.requestHeaders： 追加HTTPヘッダ

実装例

```js
ulizahtml5('video1', {
  beacon: {
    enable: true,
    url: 'https://host/path/beacon.gif',
    requestHeaders: {
      'X-AddHeader1': 'addHeaderValue1',
      'X-AddHeader2': 'addHeaderValue2'
    }
  }
  // 他のプレイヤーオプションは省略。
});
```

リニア広告を表示する

以下のプレイヤーオプションを指定します。両方を指定する場合はVASTを優先します。

- advertising.vastUrl： VASTの取得URL
- advertising.vmapUrl： VMAPの取得URL

実装例

```js
ulizahtml5('video1', {
  advertising: {
    enable: true,
    vastUrl: 'https://host/path/vast.xml'
  }
  // 他のプレイヤーオプションは省略。
});
```

なお、LIVEコンテンツでミッドロールのみを表示する場合は、以下のように設定します。ライブミッドロールを挿入するには、別途弊社製品で操作します。

実装例

```js
ulizahtml5('video1', {
  advertising: {
    enable: true,
  }
  // 他のプレイヤーオプションは省略。
});
```

リニア広告を小窓で表示する

リニア広告を表示するに加え、以下のプレイヤーオプションを指定します。

- advertising.smallWindowMode： 小窓の表示モード
- advertising.smallWindowPosition： 小窓の表示位置

実装例

```js
ulizahtml5('video1', {
  advertising: {
    enable: true,
    smallWindowMode: 'ad_reduce',
    smallWindowPosition: 'topleft'
  }
  // 他のプレイヤーオプションは省略。
});
```

コンパニオン広告を表示する

リニア広告を表示するに加え、以下の手順を実行します。

1. コンパニオン広告の要素を、ページに埋め込みます。コンパニオン広告の要素の詳しい情報は「ユーザーガイド」を参照してください。

```html
<div id="ulizabanner_01" style="width: 300px; height: 250px;"></div>
```

2. ulizahtml5()を実行します。広告に指定のコンパニオン広告の要素と同じサイズのコンパニオン広告が存在する場合、コンパニオン広告を表示します。

実装例

```html
<video id="video1" class="ulizahtml5"></video>
<div id="ulizabanner_01" style="width: 300px; height: 250px;"></div>
<script type="text/javascript">
  ulizahtml5('video1', {
    sources: [{
      src: 'https://host/path/content.mp4',
      type: 'video/mp4'
    }],
    advertising: {
      enable: true,
      vastUrl: 'https://host/path/vast.xml'
    }
  });
</script>
```

ポスター画像／スライドショーを使用する

以下のプレイヤーオプションを指定します。

- poster： ポスター画像のURL
- posterSlideShow.enable： スライドショーの有効／無効
- posterSlideShow.posters： 画像のURLのリスト
- posterSlideShow.interval： 画像を切り替える間隔

実装例

```js
ulizahtml5('video1', {
  poster: 'https://host/path/poster.png',
  posterSlideShow: {
    enable: true,
    posters: [
      'https://host/path/poster1.png',
      'https://host/path/poster2.png',
      'https://host/path/poster3.png'
    ],
    interval: 1000
  }
  // 他のプレイヤーオプションは省略。
});
```

倍速再生を使用する

以下のプレイヤーオプションを指定します。

- playbackRates.enable： 倍速再生の有効／無効
- playbackRates.values： 再生速度のリスト

以下のプレイヤーオプションは必要に応じ指定します。

- playbackRates.defaultIndex： 再生速度インデックスの初期値
- playbackRates.method： 再生速度の切り替え方法
- playbackRates.menuButtonVisible： 再生速度ボタンの表示／非表示
- playbackRates.saveLocalStorage： 選択中の再生速度の保存の有効／無効

実装例

```js
ulizahtml5('video1', {
  playbackRates: {
    enable: true,
    defaultIndex: 3,
    values: [2.0, 1.6, 1.4, 1.0, 0.5],
    method: 'list',
    menuButtonVisible: true,
    saveLocalStorage: true
  }
  // 他のプレイヤーオプションは省略。
});
```

シークプレビュー画像を表示する

以下のプレイヤーオプションを指定します。

- seekpreview.enable： シークプレビューの有効／無効
- seekpreview.url： シークプレビュー画像の取得URL

実装例

```js
ulizahtml5('video1', {
  seekpreview: {
    enable: true,
    url: 'https://host/path/preview.jpg'
  }
  // 他のプレイヤーオプションは省略。
});
```

ボリュームボタンを表示する

以下のプレイヤーオプションを指定します。

- controller.volumeControlVisible： ボリュームボタンの表示／非表示

実装例

```js
ulizahtml5('video1', {
  controller: {
    volumeControlVisible: true
  }
  // 他のプレイヤーオプションは省略。
});
```

パノラマ動画を再生する

HLSコンテンツを再生するに加え、以下の手順を実行します。

1. ulizahtml5-panoramaプラグインをページに読み込みます。ulizahtml5-panoramaプラグインの読み込みは、プレイヤーの読み込み後にします。

```html
<link href="https://host/path/ulizahtml5-panorama.min.css" rel="stylesheet">
<script src="https://host/path/ulizahtml5-panorama.min.js" type="text/javascript"></script>
```

2. videoエレメントのcrossorigin属性を指定します。

```html
<video id="video1" class="ulizahtml5" crossorigin="anonymous"></video>
```

3. 以下のプレイヤーオプションを指定します。

   - panorama.enable： パノラマ動画再生の有効／無効

   以下のプレイヤーオプションは必要に応じ指定します。

   - panorama.videoType： パノラマ動画の映像形式
   - panorama.inputType： パノラマ動画の映像の構成
   - panorama.cardboardButtonVisible： Cardboardボタンの表示／非表示
   - panorama.autoMobileOrientation： 端末の向きの変更による視点移動の有効／無効

実装例

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
    <link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
    <link href="https://host/path/ulizahtml5-panorama.min.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5-panorama.min.js" type="text/javascript"></script>
  </head>
  <body>
    <video id="video1" class="ulizahtml5" crossorigin="anonymous"></video>
    <script type="text/javascript">
      ulizahtml5('video1', {
        sources: [{
          src: 'https://host/path/testhls.m3u8',
          type: 'application/x-mpegURL'
        }],
        html5: {
          hlsjsConfig: {
            authDomain: ['www.example1.com', 'www.example2.com']
          }
        },
        panorama: {
          enable: true,
          videoType: 'equirectangular',
          inputType: 'mono',
          cardboardButtonVisible: true,
          autoMobileOrientation: true
        }
      });
    </script>
  </body>
</html>
```

オーバーレイボタンを追加する

オーバーレイボタンを追加するには、API addButtonを実行しボタンを追加します。

実装例

以下の例では追加するボタンをプレイヤー表示領域の右上に表示し、ボタン押下時にAPI playを実行します。

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
    <link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
    <style>
      .button1 {
        border: none;
        color: white;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        line-height: 46px;
        border-radius: 4px;
        background-color: green;
        width: 46px;
        height: 46px;
        top: 10px;
        right: 20px;
      }
      .button1:hover {
        background-color: blue;
      }
      .button1:active {
        background-color: purple;
      }
    </style>
  </head>
  <body>
    <video id="video1" class="ulizahtml5"></video>
    <script type="text/javascript">
      var player = ulizahtml5('video1', {
        // プレイヤーオプションは省略
      });
      var button1 = player.addButton({});
      button1.id = 'button1';
      button1.className += ' button1';
      button1.addEventListener('click', function(e) {
        player.play();
      });
    </script>
  </body>
</html>
```

コントロールバーボタンを追加する

以下のプレイヤーオプションを指定します。

- controlBarButton.enable： コントロールバーボタン追加の有効／無効
- controlBarButton.item： コントロールバーボタン情報
- controlBarButton.item[n].id： コントロールバーボタンの識別子
- controlBarButton.item[n].state[n]： ボタン状態

以下のプレイヤーオプションを指定することでコントロールバーボタンにアイコン、またはラベルを指定できます。両方を指定する場合はアイコンを優先します。

- controlBarButton.item[n].state[n].iconfont： ボタンアイコン
- controlBarButton.item[n].state[n].textJa： ボタンラベル

コントロールバーボタンはラベル、またはアイコンを複数指定でき、ボタン押下でラベル、またはアイコンを更新します。ただし、複数指定する場合はラベルまたはアイコンのどちらかに統一してください。

また、以下のプレイヤーオプションを指定することでコントロールバーボタン押下時の動作をカスタマイズできます。詳細は「APIリファレンス」を参照してください。

- controlBarButton.item[n].onClickListener： コントロールバーボタン押下コールバック関数

以下のプレイヤーオプションは必要に応じ指定します。

- controlBarButton.item[n].state[n].tooltipJa： ボタン状態ごとのツールチップラベル（日本語）
- controlBarButton.item[n].state[n].tooltip： ボタン状態ごとのツールチップラベル（日本語以外）
- controlBarButton.item[n].defaultStateIndex： ボタン状態インデックスの初期値
- controlBarButton.item[n].fontFamily： フォント名
- controlBarButton.item[n].tooltipJa： コントロールバーボタンのツールチップラベル（日本語）
- controlBarButton.item[n].tooltip： コントロールバーボタンのツールチップラベル（日本語以外）

実装例

```js
ulizahtml5('video1', {
  controlBarButton: {
    enable: true,
    item: [
      {
        id: 'ContorolBarButtonId1',
        state: [
          {
            textJa: 'オン',
            tooltipJa: 'ボタン1_オン',
            tooltip: 'Button1ON'
          },
          {
            textJa: 'オフ',
            tooltipJa: 'ボタン1_オフ',
            tooltip: 'Button1OFF'
          }
        ],
        defaultStateIndex: 1,
        fontFamily: 'fontFamily',
        onClickListener: function(currentStateIndex, nextStateIndex, player, element) {
          player.play();
          return nextStateIndex;
        }
      },
      {
        id: 'ContorolBarButtonId2',
        state: [
          {
            iconfont: 'iconfontClassName'
          }
        ],
        tooltipJa: 'ボタン2',
        tooltip: 'Button2'
      }
    ]
  }
  // 他のプレイヤーオプションは省略。
});
```

センターコントローラーをカスタマイズする

センターコントローラーをカスタマイズするには、プレイヤーの初期化後にAPIでボタンや行を編集します。

実装例

以下の例は以下の通りにカスタマイズする想定です。

- 再生開始前： 既存のボタンを削除し独自のボタンを追加する。
- 再生開始後： 既存の行を削除し再構築する。

```js
var player = ulizahtml5('video1', {
  skipForwardAndBackward: {
    enable: true
  }
  // 他のプレイヤーオプションは省略。
});

// 再生開始前は既存のボタンを削除し独自のボタンを追加する。
var button = document.createElement('div');
button.style.width = '50px';
button.style.height = '50px';
button.style.backgroundColor = 'red';
button.style.visibility = 'visible';
button.style.pointerEvents = 'auto';
button.onclick = function() {
  player.play();
};
var lineLayout = player.getCenterControllerLine(0);
lineLayout.removeChild(lineLayout.firstChild);
lineLayout.appendChild(button);

// 再生開始後は既存の行を削除し再構築する。
player.one('play', function() {
  player.removeCenterControllerLine(0);
  var lineIndex = player.appendCenterControllerLine();
  var lineLayout = player.getCenterControllerLine(lineIndex);
  var skipBackward30 = player.createCenterSkipBackwardButton(30);
  var skipBackward5 = player.createCenterSkipBackwardButton(5);
  var playPause = player.createCenterPlayPauseButton();
  var skipForward5 = player.createCenterSkipForwardButton(5);
  var skipForward30 = player.createCenterSkipForwardButton(30);
  lineLayout.appendChild(skipBackward30);
  lineLayout.appendChild(skipBackward5);
  lineLayout.appendChild(playPause);
  lineLayout.appendChild(skipForward5);
  lineLayout.appendChild(skipForward30);
});
```

字幕を使用する

1. 以下のスタイルシートをページに読み込みます。スタイルシートの読み込みは、プレイヤーの読み込み後にします。

   ```html
   <link href="https://host/path/ulizahtml5-subtitle-list.css" rel="stylesheet">
   ```

2. 以下のプレイヤーオプションを指定します。字幕リストのデザインは、ulizahtml5-subtitle-list.cssを変更することでカスタマイズできます。字幕リストのデザインをカスタマイズするを参照してください。

- subtitles.enable： 字幕の有効／無効
- subtitles.src： 字幕情報

以下のプレイヤーオプションは必要に応じ指定します。

- subtitles.defaultIndex： 字幕インデックスの初期値
- subtitles.horizontalAlign： 整列方向
- subtitles.method： 字幕の切り替え方法
- subtitles.menuButtonVisible： 字幕ボタンの表示／非表示
- subtitles.parentElementId： 字幕リストのdiv要素のid
- subtitles.saveLocalStorage： 選択中の字幕の保存の有効／無効

実装例

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
    <link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
    <link href="https://host/path/ulizahtml5-subtitle-list.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
  </head>
  <body>
    <video id="video1" class="ulizahtml5"></video>
    <div id="subtitle-list-area"></div>
    <script type="text/javascript">
      ulizahtml5('video1', {
        sources: [{
          src: 'https://host/path/content.mp4',
          type: 'video/mp4'
        }],
        subtitles: {
          enable: true,
          src: [
            {
              label: '日本語',
              url: 'https://host/path/subtitle_ja.vtt'
            },
            {
              label: '英語',
              url: 'https://host/path/subtitle_en.vtt'
            }
          ],
          defaultIndex: 1,
          horizontalAlign: 'middle',
          method: 'list',
          menuButtonVisible: true,
          parentElementId: 'subtitle-list-area',
          saveLocalStorage: true
        }
      });
    </script>
  </body>
</html>
```

Google Cast/AirPlayを使用する

以下の手順を実行します。

1. 以下のプレイヤーオプションを指定します。

   - googlecast.enable： Google Cast Senderの有効／無効
   - googlecast.appId： ReceiverアプリケーションID ※AirPlayのみを有効にする場合でも指定します。
   - airplay.enable： AirPlayの有効／無効

   以下のプレイヤーオプションは必要に応じ指定します。

   - googlecast.targetReceiverVer： Receiverアプリケーションのメジャーバージョン
   - googlecast.castReceiverConfig： Receiver設定情報
   - googlecast.castMediaInfo： Cast情報

実装例

```js
ulizahtml5('video1', {
  googlecast: {
    enable: true,
```

```
    appId: 'XXXXXXX',
    targetReceiverVer: 2,
    castReceiverConfig: {
      ui: {
        splashImage: 'https://host/path/firstscreen.png',
        slideshowImages: [
          'https://host/path/idleScreen1.png',
          'https://host/path/idleScreen2.png',
          'https://host/path/idleScreen3.png',
          'https://host/path/idleScreen4.png'
        ]
      }
    },
    castMediaInfo: {
      title: 'Cast Demo',
      url: 'https://host/path/cast_media.m3u8',
      streamType: 'vod',
      mediaProtocol: 'hls',
      thumb: 'https://host/path/cast_media_thumb.png',
      artwork: [
        'https://host/path/cast_media_artwork.png'
      ]
    }
  },
  airplay: {
    enable: true
  }
  // 他のプレイヤーオプションは省略。
});
```

画質を切り替える

以下の手順を実行します。

1. ビデオトラック情報のリストを取得します。

```html
<script type="text/javascript">
  var videoTrack = player.getVideoTrackArray();
</script>
```

ビデオトラック情報のリストは、画質の高い順にソートされています。ビデオトラック情報のリストの例は以下の通りです。

```js
[
  {
    bandwidth: 2000000,
    height: 1080,
    label: '高画質',
    width: 1920
  },
  {
    bandwidth: 1000000,
    height: 720,
    label: '通常画質',
    width: 1280
  },
  {
    bandwidth: 260000,
    height: 234,
    label: '低画質',
    width: 416
  }
]
```

2. 切り替えるビデオトラックを指定します。ビデオトラックの指定には、ビデオトラックインデックスを使用します。-1を指定すると自動選択になります。

```html
<script type="text/javascript">
  player.setCurrentVideoTrack(1);
</script>
```

画質メニューをカスタマイズする

以下のプレイヤーオプションを指定します。

- videoTrack.enable： 画質切り替えの有効／無効
- videoTrack.menuLabels： 画質メニューの各項目に表示する文言

画質メニューの項目を、上から順に指定の文字列に置き換えます。先頭の項目は 'Auto' を置き換えます。

- videoTrack.defaultIndex： ビデオトラックインデックスの初期値
- videoTrack.menuButtonVisible： 画質ボタンの表示／非表示
- videoTrack.method： 画質の切り替え方法
- videoTrack.saveLocalStorage： 選択中の画質の保存の有効／無効

実装例

```js
ulizahtml5('video1', {
  videoTrack: {
    enable: true,
    defaultIndex: 1,
    menuLabels: ['自動', '高画質', '通常画質', '低画質'],
    menuButtonVisible: true,
    method: 'toggle',
    saveLocalStorage: true
  }
  // 他のプレイヤーオプションは省略。
});
```

スキンを切り替える

以下の手順を実行します。

1. 以下のプレイヤーオプションを指定します。
   - skin： スキン名

実装例

```js
ulizahtml5('video1', {
  skin: 'skin2'
  // 他のプレイヤーオプションは省略。
});
```

ホットキーを使用する

以下のプレイヤーオプションを指定します。

- hotkey.enable： ホットキーの有効／無効

以下のプレイヤーオプションを指定することでホットキーをカスタマイズできます。詳細は「APIリファレンス」を参照してください。

- hotkey.eventListener： キー押下コールバック関数

実装例

```js
ulizahtml5('video1', {
  hotkey: {
    enable: true
    eventListener: function(player, event) {
      // Mキーが押下された時
      if (event.key === 'm' || event.key === 'M') {
        // 戻り値をfalseにするとデフォルトのホットキー挙動を行わない。
        return false;
      }
      return true;
    }
  }
  // 他のプレイヤーオプションは省略。
});
```

プレイヤーの初期化の完了時に任意の処理を行う

プレイヤーの初期化の完了時に任意の処理を行うには、ターゲットのイベントリスナーリストにulizaplayerreadyイベントを追加します。

実装例

```js
var player = ulizahtml5('video1', {
  // プレイヤーオプションは省略
});
window.addEventListener('ulizaplayerready', function(e) {
  console.log(e.type);
});
```

レジューム再生する

以下のプレイヤーオプションを指定します。

- resumePlayback.enable： レジューム再生の有効／無効

前回の再生位置から再生を開始するか、先頭から再生を開始するかをユーザーに選択させる等のカスタマイズができます。プレイヤーは、再生を開始する時に再生開始位置が指定、または保存されている場合にレジューム確認コールバック関数を呼び出し、再生処理を中断します（レジューム確認待ち状態）。API startResumeが実行されると、再生処理を再開します。カスタマイズする場合は、以下のプレイヤーオプションを指定します。この関数が未指定の場合、再生開始位置から再生を開始します。詳細は「APIリファレンス」を参照してください

- resumePlayback.resumeConfirmFunc： レジューム確認コールバック関数

実装例

```html
<div id="resumebutton" style="display: none;">
  <input type="button" value="top playback" onclick="selectResume(false);">
  <input type="button" value="resume playback" onclick="selectResume(true);">
</div>

<script type="text/javascript">
  var player = ulizahtml5('video1', {
    resumePlayback: {
      enable: true,
      resumeConfirmFunc: resumeCallback
    }
    // 他のプレイヤーオプションは省略。
  });

  var resumePosition;
```

```
  // レジューム確認コールバック関数に指定した関数
  function resumeCallback(resumePos) {
    resumePosition = resumePos;
    document.getElementById('resumebutton').style.display = 'block';
  }

  // ユーザーのレジューム再生要否(true/false)が渡される関数
  function selectResume(isResume) {
    if (isResume === false) {
      resumePosition = 0;
    }
    player.startResume(resumePosition);
  }
</script>
```

拡大表示を使用する

以下のプレイヤーオプションを指定します。

- enlargement.enable： 拡大表示の有効／無効
- enlargement.maxRate： 最大拡大率

実装例

```js
ulizahtml5('video1', {
  enlargement: {
    enable: true,
    maxRate: 500
  }
  // 他のプレイヤーオプションは省略。
});
```

ウォーターマークを表示する

以下のプレイヤーオプションを指定します。

- watermark.enable： ウォーターマークの有効／無効
- watermark.url： ウォーターマーク画像の取得URL
- watermark.position： ウォーターマーク表示位置

実装例

```js
ulizahtml5('video1', {
  watermark: {
    enable: true,
    url: 'https://host/path/watermark.png',
    position: 'topright'
  }
  // 他のプレイヤーオプションは省略。
});
```

進む／戻るを使用する

以下のプレイヤーオプションを指定します。

- skipForwardAndBackward.enable： 進む／戻るの有効／無効

以下のプレイヤーオプションは必要に応じ指定します。

- skipForwardAndBackward.centerForwardAndBackwardVisible： センターコントローラー上の進む／戻るボタン表示／非表示

- skipForwardAndBackward.forwardAndBackwardVisible： コントロールバー上の進む／戻るボタン表示／非表示
- skipForwardAndBackward.skipForwardSeconds： 前方シークする秒数
- skipForwardAndBackward.skipBackwardSeconds： 後方シークする秒数
- skipForwardAndBackward.multiTapAreaWidth： モバイルブラウザでプレイヤー表示領域の左右部をダブルタップしてシークする場合の、反応領域の幅です。

実装例

```js
ulizahtml5('video1', {
  skipForwardAndBackward: {
    enable: true,
    centerForwardAndBackwardVisible: true,
    forwardAndBackwardVisible: true,
    skipForwardSeconds: 10,
    skipBackwardSeconds: 10,
    multiTapAreaWidth: '30%'
  }
  // 他のプレイヤーオプションは省略。
});
```

ジェスチャーをカスタマイズする

以下のプレイヤーオプションを必要に応じ指定します。

- gestureControl.flickInterval： フリックを判断する期間
- gestureControl.multiPressInterval： ダブルクリック／ダブルタップを判断する期間
- gestureControl.enablePageScroll： ページスクロールの有効／無効

実装例

```js
ulizahtml5('video1', {
  gestureControl: {
    flickInterval: 100,
    multiPressInterval: 300,
    enablePageScroll: true
  }
  // 他のプレイヤーオプションは省略。
});
```

再生中のプレイヤーオプションを変更する

以下の手順を実行します。変更できるプレイヤーオプションについては、「ユーザーガイド」を参照してください。

プレイヤーオプション変更（コンテンツ切り替え）

1. 再生するコンテンツを指定します。また、変更したいプレイヤーオプションを指定します。
2. API initを呼び出し、1. の設定を反映します。

実装例

以下の例ではウォーターマークを指定します。

```js
var param = {
  sources: [{
    src: 'https://host/path/content.mp4',
    type: 'video/mp4'
  }],
  watermark: {
    enable: true,
    url: './path/watermark.png',
```

```
    width: 120,
    height: 250
  }
};
player.setOptions(param);
player.init();
```

プレイヤーオプション変更（パラメータ切り替え）

変更したいプレイヤーオプションを指定します。

実装例

以下の例ではウォーターマークを指定します。

```js
var param = {
  watermark: {
    enable: true,
    url: './path/watermark.png',
    width: 120,
    height: 250
  }
};
player.setOptions(param);
```

リピートする

以下のプレイヤーオプションを指定します。

- repeat.enable： リピートの有効／無効

以下のプレイヤーオプションは必要に応じ指定します。

- repeat.buttonVisible： リピートボタンの表示／非表示
- repeat.defaultStatus： リピートの初期値
- repeat.saveLocalStorage： 選択中のリピートの有効／無効の保存の有効／無効

実装例

```js
ulizahtml5('video1', {
  repeat: {
    enable: true,
    buttonVisible: true,
    defaultStatus: true,
    saveLocalStorage: true
  }
  // 他のプレイヤーオプションは省略。
});
```

設定メニューボタンを表示する

以下のプレイヤーオプションを指定します。

- settingMenu.enable： 設定メニューの有効／無効

以下のプレイヤーオプションは必要に応じ指定します。

- settingMenu.menuButtonVisible： 設定メニューボタンの表示／非表示

実装例

```js
ulizahtml5('video1', {
  settingMenu: {
    enable: true,
    menuButtonVisible: true
  }
  // 他のプレイヤーオプションは省略。
});
```

自動サイズ調整を使用する

以下のプレイヤーオプションを指定します。

- width： プレイヤーの幅
- height： プレイヤーの高さ
- aspectRatio： プレイヤーのアスペクト比

実装例

```js
ulizahtml5('video1', {
  width: '100%',
  aspectRatio: '16:9',
  // 他のプレイヤーオプションは省略。
});
```

自動リロードする

以下のプレイヤーオプションを指定します。

- autoReload.enable： 自動リロードの有効／無効
- autoReload.loadingTimeout： 再生を継続できない状態になってからリロードするまでの時間

実装例

```js
ulizahtml5('video1', {
  autoReload: {
    enable: true,
    loadingTimeout: 10000
  }
  // 他のプレイヤーオプションは省略。
});
```

アイコンフォントを追加する

以下の手順を実行します。

1. アイコンフォントを追加します。
2. アイコンフォントを表示するためのclass属性を該当のボタンに付与します。

実装例

以下の例ではアイコンフォントをリアクションボタンに指定します。

```html
<!-- 1. アイコンフォントを追加します。 -->
<link rel="stylesheet" href="https://host/path/iconfont.css">

<script type="text/javascript">
  ulizahtml5('video1', {
    // 該当するパラメータ以外は省略します。
    videoAnalytics: {
```

```
      reactionItem: [
        {
          // 2. アイコンフォントを表示するためのclass属性を該当のボタンに付与します。
          iconId: 'IconfontClass'
        }
      ]
    }
  });
</script>
```

画面ロックを使用する

以下のプレイヤーオプションを指定します。

- screenLock.enable： 画面ロックの有効／無効

以下のプレイヤーオプションは必要に応じ指定します。

- screenLock.unlockType： 画面ロックの解除方法
- screenLock.unlockTime： 画面ロックの解除方法が 'longPress' の場合の長押しの秒数
- screenLock.buttonVisible： 画面ロックボタンの表示／非表示

実装例

```js
ulizahtml5('video1', {
  screenLock: {
    enable: true,
    unlockType: 'longPress',
    unlockTime: 5000,
    buttonVisible: true
  }
  // 他のプレイヤーオプションは省略。
});
```

切り出し再生する

以下のプレイヤーオプションを指定します。

- playbackRange.enable： 切り出し再生の有効／無効
- playbackRange.startSeconds： 切り出し開始位置
- playbackRange.endSeconds： 切り出し終了位置

実装例

```js
ulizahtml5('video1', {
  playbackRange: {
    enable: true,
    startSeconds: 30,
    endSeconds: 'end'
  }
  // 他のプレイヤーオプションは省略。
});
```

パーツサイズを変更する

以下のプレイヤーオプションを指定します。

- partsSize.enable： パーツサイズの有効／無効
- partsSize.values： パーツサイズの倍率のリスト

以下のプレイヤーオプションは必要に応じ指定します。

- partsSize.defaultIndex： パーツサイズインデックスの初期値
- partsSize.menuButtonVisible： パーツサイズボタンの表示／非表示
- partsSize.menuLabels： パーツサイズメニューの各項目に表示する文言
- partsSize.method： パーツサイズの切り替え方法
- partsSize.saveLocalStorage： 選択中のパーツサイズの保存の有効／無効

実装例

```js
ulizahtml5('video1', {
  partsSize: {
    enable: true,
    values: [1, 1.5, 2],
    defaultIndex: 0,
    menuButtonVisible: true,
    menuLabels: ['中', '大', '特大'],
    method: 'toggle',
    saveLocalStorage: true
  }
  // 他のプレイヤーオプションは省略。
});
```

チャプターを使用する

以下の手順を実行します。

1. 以下のスタイルシートをページに読み込みます。スタイルシートの読み込みは、プレイヤーの読み込み後にします。

   ```html
   <link href="https://host/path/ulizahtml5-chapter.css" rel="stylesheet">
   ```

2. 以下のプレイヤーオプションを指定します。チャプター領域のデザインは、ulizahtml5-chapter.cssを変更することでカスタマイズできます。チャプター領域のデザインをカスタマイズするを参照してください。

   - chapter.enable： チャプターの有効／無効
   - chapter.prevButtonVisible： 前チャプターボタンの表示／非表示
   - chapter.nextButtonVisible： 次チャプターボタンの表示／非表示
   - chapter.item： チャプター情報のリスト
   - chapter.item[n].startTime： 再生開始位置

   以下のプレイヤーオプションは必要に応じ指定します。

   - chapter.parentElementId： チャプター領域のdiv要素のid
   - chapter.item[n].title： タイトル
   - chapter.item[n].thumbnailUrl： サムネイル画像の取得URL
   - chapter.item[n].data.page： プレゼンテーションのページ番号

プレゼンテーションについては、「プレゼンテーションを表示する」を参照してください。

実装例

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
    <link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
    <link href="https://host/path/ulizahtml5-chapter.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
  </head>
  <body>
    <video id="video1" class="ulizahtml5"></video>
```

```
    <div id="chapter-area"></div>
    <script type="text/javascript">
      ulizahtml5('video1', {
        sources: [{
          src: 'https://host/path/content.mp4',
          type: 'video/mp4'
        }],
        chapter: {
          enable: true,
          parentElementId: 'chapter-area',
          prevButtonVisible: true,
          nextButtonVisible: true,
          item: [
            {
              startTime: 0,
              title: 'プロローグ',
              thumbnailUrl: './thumbnail_1.png',
              data: { page: 1 }
            },
            {
              startTime: 20,
              title: '本編',
              thumbnailUrl: './thumbnail_2.png',
              data: { page: 2 }
            },
            {
              startTime: 130,
              title: 'エンドロール',
              thumbnailUrl: './thumbnail_3.png',
              data: { page: 3 }
            }
          ]
        }
      });
    </script>
  </body>
</html>
```

プレゼンテーションを表示する

以下の手順を実行します。

1. 以下のスタイルシートをページに読み込みます。スタイルシートの読み込みは、プレイヤーの読み込み後にします。

```
<link href="https://host/path/ulizahtml5-presentation.css" rel="stylesheet">
```
html

2. 以下のプレイヤーオプションを指定します。プレゼンテーション領域のデザインは、ulizahtml5-presentation.cssを変更することでカスタマイズできます。プレゼンテーション領域のデザインをカスタマイズするを参照してください。

   - presentation.enable： プレゼンテーションの有効／無効
   - presentation.documentUrl 資料URL
   - presentation.parentElementId： プレゼンテーション領域のdiv要素のid

   以下のプレイヤーオプションは必要に応じ指定します。

   - presentation.displayInPlayer： フルスクリーン再生時に、プレイヤー領域に資料を表示するか否か
   - presentation.positionChangeButtonVisible： 資料表示位置変更ボタンの表示／非表示
   - presentation.fullscreenViewChangeButtonVisible： 資料表示状態変更ボタンの表示／非表示
   - presentation.defaultFullscreenView： プレイヤー領域に資料を表示する時の、資料の初期表示状態
   - presentation.saveFullscreenViewToLocalStorage： フルスクリーン再生時の資料表示状態の保存の有効／無効

実装例

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
    <link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
    <link href="https://host/path/ulizahtml5-presentation.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
  </head>
  <body>
    <video id="video1" class="ulizahtml5"></video>
    <div id="presentation-area"></div>
    <script type="text/javascript">
      ulizahtml5('video1', {
        sources: [{
          src: 'https://host/path/content.mp4',
          type: 'video/mp4'
        }],
        chapter: {
          enable: true,
          prevButtonVisible: true,
          nextButtonVisible: true,
          item: [
            {
              startTime: 0,
              title: 'プロローグ',
              thumbnailUrl: './thumbnail_1.png',
              data: { page: 1 }
            },
            {
              startTime: 20,
              title: '本編',
              thumbnailUrl: './thumbnail_2.png',
              data: { page: 2 }
            },
            {
              startTime: 130,
              title: 'エンドロール',
              thumbnailUrl: './thumbnail_3.png',
              data: { page: 3 }
            }
          ]
        },
        presentation: {
          enable: true,
          parentElementId: 'presentation-area',
          documentUrl: 'https://host/path/sample.pdf',
          displayInPlayer: true,
          positionChangeButtonVisible: true,
          fullscreenViewChangeButtonVisible: true,
          defaultFullscreenView: 'side-by-side',
          saveFullscreenViewToLocalStorage: true,
        }
      });
    </script>
  </body>
</html>
```

シーク範囲制御を使用する

以下のプレイヤーオプションを指定します。

- seekableRange.enable： シーク範囲制御の有効／無効
- seekableRange.start： シーク有効範囲の開始位置
- seekableRange.end： シーク有効範囲の終了位置

実装例

```js
ulizahtml5('video1', {
  seekableRange: {
    enable: true,
    start: 0,
    end: 'watched'
  }
  // 他のプレイヤーオプションは省略。
});
```

ヘルプを表示する

以下のプレイヤーオプションを指定します。

- help.enable： ヘルプの有効／無効

以下のプレイヤーオプションは必要に応じ指定します。

- help.buttonVisible： ヘルプボタンの表示／非表示

実装例

```js
ulizahtml5('video1', {
  help: {
    enable: true,
    buttonVisible: true
  }
  // 他のプレイヤーオプションは省略。
});
```

開始位置ボタンを表示する

以下のプレイヤーオプションを指定します。

- controller.beginningButtonVisible： 開始位置ボタンの表示／非表示

実装例

```js
ulizahtml5('video1', {
  controller: {
    beginningButtonVisible : true
  }
  // 他のプレイヤーオプションは省略。
});
```

次／前コンテンツボタンを表示する

プレイリストの組み込みに加え、以下のプレイヤーオプションを指定します。

- nextPrevContent.enable： 次／前コンテンツの有効／無効

以下のプレイヤーオプションは必要に応じ指定します。

- nextPrevContent.nextButtonVisible： 次コンテンツボタンの表示／非表示
- nextPrevContent.prevButtonVisible： 前コンテンツボタンの表示／非表示

プレイリストについては、「ULIZA Player (HTML5)プレイリスト機能ユーザーガイド」を参照してください。

実装例

```js
var playlistOptions = {
  parentElementId: 'playlist1',
  mediaElementId: 'video1',
  repeatType: 'all',
  defaultIndex: 0,
  items: [
    {
      thumbUrl: 'https://host/path/thumbnail1.png',
      description: '項目1',
      playerOptions: {
        width: 800,
        height: 450,
        sources: [{
          src: 'https://host/path/item1.mp4',
          type: 'video/mp4'
        }],
        nextPrevContent: {
          enable: true,
          nextButtonVisible: true,
          prevButtonVisible: true
        }
      }
    },
    {
      thumbUrl: 'https://host/path/thumbnail2.png',
      description: '項目2',
      playerOptions: 'https://host/path/player_options1.json'
    }
  ]
};
var playlist = new UlizaPlaylist(playlistOptions);
```

外部出力／録画を抑止する

以下のプレイヤーオプションを指定します。

- disableExternalDisplayType： 外部出力を抑止する出力方式
- disableScreenCapture： 録画抑止の有効／無効

実装例

```js
ulizahtml5('video1', {
  disableExternalDisplayType: ['airplay', 'googlecast', 'miracast'],
  disableScreenCapture: true
  // 他のプレイヤーオプションは省略。
});
```

オーディオを切り替える

以下の手順を実行します。

1. オーディオトラック情報のリストを取得します。

   ```html
   <script type="text/javascript">
     var audioTrack = player.getAudioTrackArray();
   </script>
   ```

   オーディオトラック情報のリストは、Audioタグの記載順です。オーディオトラック情報のリストの例は以下の通りです。

   ```js
   [
     {
   ```

```
    id: 0,
    lang: 'ja',
    name: '主音声',
    // ...
  },
  {
    id: 1,
    lang: 'eng',
    name: '副音声',
    // ...
  }
]
```

2. 切り替えるオーディオトラックを指定します。オーディオトラックの指定には、オーディオトラックインデックスを使用します。-1を指定するとデフォルトのオーディオになります。

```html
<script type="text/javascript">
  player.setCurrentAudioTrack(1);
</script>
```

音声メニューをカスタマイズする

以下のプレイヤーオプションを指定します。

- audioTrack.enable： オーディオ切り替えの有効／無効

以下のプレイヤーオプションは必要に応じ指定します。

- audioTrack.defaultIndex： オーディオトラックインデックスの初期値
- audioTrack.menuButtonVisible： 音声ボタンの表示／非表示
- audioTrack.menuLabels： 音声メニューの各項目に表示する文言
- audioTrack.method： オーディオの切り替え方法
- audioTrack.saveLocalStorage： 選択中のオーディオの保存の有効／無効

実装例

```js
ulizahtml5('video1', {
  audioTrack: {
    enable: true,
    defaultIndex: 1,
    menuButtonVisible: true,
    menuLabels: ['主音声', '副音声'],
    method: 'list',
    saveLocalStorage: true
  }
  // 他のプレイヤーオプションは省略。
});
```

カスタムレイヤーを表示する

以下のプレイヤーオプションを指定します。

- customLayer.enable： カスタムレイヤーの有効／無効

以下のプレイヤーオプションは必要に応じ指定します。

- customLayer.className： カスタムレイヤーに追加するクラス属性

実装例

```js
var player = ulizahtml5('video1', {
  customLayer: {
    enable: true,
    className: 'class-name1 class-name2'
  }
  // 他のプレイヤーオプションは省略。
});

window.addEventListener('ulizaplayerready', function(e) {
  // 子孫要素をカスタマイズします。
  var customLayerEl = player.getCustomLayerElement();
  customLayerEl.innerHTML = '<iframe width="40%" height="100%" frameborder="0"marginheight="0" marginwidth="0"
allowtransparency="true" scrolling="no" src="https://host/path/inline_page.html"></iframe>';

  // コントロールバーにカスタムレイヤーの表示／非表示ボタンを追加します。
  var itemOption = {
    id: 'customLayerVisibleButton',
    type: 'button',
    tooltip: 'Custom Layer',
    tooltipJa: 'カスタムレイヤー',
    state: [
      {
        textJa: '表示'
      },
      {
        textJa: '非表示'
      }
    ],
    onClickListener: function(currentStateIndex, nextStateIndex, player, element) {
      if (nextStateIndex === 1) {
        player.setCustomLayerVisible(false);
      } else {
        player.setCustomLayerVisible(true);
      }
      return nextStateIndex;
    }
  };
  player.addControlBarButton(itemOption);
});
```

リモートシークを抑止する

以下のプレイヤーオプションを指定します。

- disableRemoteSeek： リモートシーク制御の有効／無効

実装例

```js
ulizahtml5('video1', {
  disableRemoteSeek: true
  // 他のプレイヤーオプションは省略。
});
```

ピクチャーインピクチャーを使用する

以下のプレイヤーオプションを指定します。

- controller.pictureInPictureButtonVisible： ピクチャーインピクチャーボタンの表示／非表示

実装例

```js
ulizahtml5('video1', {
  controller: {
```

```
    pictureInPictureButtonVisible: true
  }
  // 他のプレイヤーオプションは省略。
});
```

タイトルを表示する

以下のプレイヤーオプションを指定します。

- title.enable：タイトル表示の有効／無効
- title.visibleForMobile：モバイルブラウザでタイトル表示／非表示
- title.textJa：タイトル（日本語）
- title.text：タイトル（日本語以外）

実装例

```js
ulizahtml5('video1', {
  title: {
    enable: true,
    visibleForMobile: false,
    text: 'English title',
    textJa: '日本語タイトル',
  }
  // 他のプレイヤーオプションは省略。
});
```

再生開始ボタンを装飾する

以下のプレイヤーオプションを指定します。

- controller.centerPlayStartVisible : 再生開始ボタン表示／非表示
- controller.playStartButtonEffect：再生開始ボタンアニメーション

実装例

```js
ulizahtml5('video1', {
  controller: {
    centerPlayStartVisible: true,
    playStartButtonEffect: 'reflection'
  }
  // 他のプレイヤーオプションは省略。
});
```

マルチアングル動画を再生する

以下の手順を実行します。

1. 以下のスタイルシートをページに読み込みます。スタイルシートの読み込みは、プレイヤーの読み込み後にします。

   ```html
   <link href="https://host/path/ulizahtml5-multi-angle.css" rel="stylesheet">
   ```

2. 以下のプレイヤーオプションを指定します。マルチアングルリストのデザインは、ulizahtml5-multi-angle.cssを変更することでカスタマイズできます。マルチアングルリストのデザインをカスタマイズするを参照してください。

- multiAngle.enable： マルチアングル動画の有効／無効
- multiAngle.angle： アングル数

以下のプレイヤーオプションは必要に応じ指定します。

- multiAngle.angleLabels： マルチアングルリストのラベルに表示する文言
- multiAngle.defaultIndex： 再生を開始する時に選択するアングルのインデックス
- multiAngle.parentElementId： マルチアングルリストのdiv要素のid
- multiAngle.saveLocalStorage： 選択中のアングルの保存の有効／無効

実装例

```html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,viewport-fit=cover">
    <link href="https://host/path/ulizahtml5.min.css" rel="stylesheet">
    <link href="https://host/path/ulizahtml5-multi-angle.css" rel="stylesheet">
    <script src="https://host/path/ulizahtml5.min.js" type="text/javascript"></script>
  </head>
  <body>
    <video id="video1" class="ulizahtml5"></video>
    <div id="multi-angle-list-area" style="width: 160px; height: 360px"></div>
    <script type="text/javascript">
      ulizahtml5('video1', {
        sources: [{
          src: 'https://host/path/content.mp4',
          type: 'video/mp4'
        }],
        multiAngle: {
          enable: true,
          angle: 4,
          angleLabels: ['メイン', 'カメラ1', 'カメラ2', 'カメラ3'],
          defaultIndex: 0,
          parentElementId: 'multi-angle-list-area',
          saveLocalStorage: true
        }
      });
    </script>
  </body>
</html>
```

ビデオビューアを抑止する

以下のプレイヤーオプションを指定します。

- disableInlineFullscreen： ビデオビューア制御の有効／無効

実装例

```js
ulizahtml5('video1', {
  disableInlineFullscreen: true
  // 他のプレイヤーオプションは省略。
});
```

付録

チャプター領域のデザインをカスタマイズする

チャプターの構造は以下の通りです。

```html
<div id="..."><!-- チャプター領域のdiv要素 -->
  <div class="uliza-chapter">
    <div class="item">
      <div class="thumbnail"></div>
      <div class="description" data-start-time="start-time"></div>
    </div>
    <!-- <div class="item"> が並ぶ -->
  </div>
</div>
```

要素を表すclassは以下の通りです。

uliza-chapter	チャプター領域の中に表示するチャプターの実体です。
item	チャプターの項目です。
thumbnail	項目内のサムネイルです。
description	項目内のテキストです。
title	項目内のタイトルです。
start-time	項目内の再生開始位置です。

状態を表すclassは以下の通りです。

has-started	プレイヤーが再生中の状態です。class「uliza-chapter」と同じ要素に付きます。
linearad-has-started	プレイヤーが広告再生中の状態です。class「uliza-chapter」と同じ要素に付きます。
lock	プレイヤーが画面ロックをしている状態です。class「uliza-chapter」と同じ要素に付きます。
selected	選択中の項目です。class「item」と同じ要素に付きます。
hover	マウスカーソルがホバー中の項目です。class「item」と同じ要素に付きます。

「ユーザーガイド」のチャプター領域の表示例に示すCSSはソースファイルulizahtml5-chapter.cssをご確認ください。カスタマイズする場合は十分な動作検証をしてください。

プレゼンテーション領域のデザインをカスタマイズする

プレゼンテーションの構造は以下の通りです。

```html
<div id="..."><!-- プレゼンテーション領域のdiv要素 -->
  <div class="uliza-presentation">
    <div class="toolbar">
      <div class="prev-chapter toolbar-button"></div>
      <div class="next-chapter toolbar-button"></div>
      <div class="page-number"></div>
      <div class="spacer"></div>
      <div class="shrink-page toolbar-button"></div>
      <div class="page-enlargement"></div>
      <div class="enlarge-page toolbar-button"></div>
    </div>
    <div class="document-area">
      <canvas class="document-layer"></canvas>
```

```
      <div class="annotation-layer"></div>
    </div>
  </div>
</div>
```

要素を表すclassは以下の通りです。

uliza-presentation	プレゼンテーション領域の中に表示するプレゼンテーションの実体です。
toolbar	ツールバーです。
toolbar-button	ツールバーのボタン類です。
prev-chapter	前チャプターボタンです。
next-chapter	次チャプターボタンです。
page-number	現在のページ番号です。
spacer	現在のページ番号と縮小ボタンのスペースです。
shrink-page	縮小ボタンです。
page-enlargement	現在の拡大率です。
enlarge-page	拡大ボタンです。
document-area	資料を表示する領域です。余白を含みます。
document-layer	資料を表示する領域です。
annotation-layer	資料のリンク用の領域です。

状態を表すclassは以下の通りです。

disabled	無効中のツールバーのボタンです。class「toolbar-button」と同じ要素に付きます。
fit	資料のサイズが資料を表示する領域のサイズと同じ状態です。class「document-area」と同じ要素に付きます。
has-started	プレイヤーが再生中の状態です。class「uliza-presentation」と同じ要素に付きます。
letter	資料の幅が資料を表示する領域の幅より大きい状態です。class「document-area」と同じ要素に付きます。
linearad-has-started	プレイヤーが広告再生中の状態です。class「uliza-presentation」と同じ要素に付きます。
lock	プレイヤーが画面ロックをしている状態です。class「uliza-presentation」と同じ要素に付きます。
hover	マウスカーソルがホバー中のツールバーのボタンです。class「toolbar-button」と同じ要素に付きます。
over	資料のサイズが資料を表示する領域のサイズより大きい状態です。class「document-area」と同じ要素に付きます。
pillar	資料の高さが資料を表示する領域の高さより大きい状態です。class「document-area」と同じ要素に付きます。

「ユーザーガイド」のプレゼンテーション領域の表示例に示すCSSはソースファイルulizahtml5-presentation.cssをご確認ください。カスタマイズする場合は十分な動作検証をしてください。

字幕リストのデザインをカスタマイズする

字幕リストの構造は以下の通りです。

```html
<div id="..."><!-- 字幕リストのdiv要素 -->
  <div class="uliza-subtitle-list">
    <div class="header">
      <div class="search-area uliza-icon-search">
        <input class="search-input" placeholder="search"></input>
      </div>
      <div class="track-area uliza-icon-subtitles">
```

```
        <span class="select-track"></span>
        <div class="track-list">
          <div class="track-item"></div>
          <!-- <div class="track-item"> が並ぶ -->
        </div>
      </div>
    </div>
    <div class="cue-list-area">
      <div class="cue-list">
        <div class="cue" data-start-time="start-time"></div>
        <!-- <div class="cue" data-start-time="start-time"></div> が並ぶ -->
      </div>
    </div>
    <div class="auto-scroll-button"></div>
  </div>
</div>
```

要素を表すclassは以下の通りです。

uliza-subtitle-list	字幕リストの実体です。
header	字幕リストのヘッダー領域の項目です。
search-area	ヘッダーの検索窓の領域です。
search-input	検索窓の入力領域です。
search	入力領域のプレースホルダーの文言です。
track-area	ヘッダーの字幕切り替えの領域です。
select-track	現在選択されている言語のラベルです。
track-list	切り替え可能な言語のラベルのリストです。
track-item	切り替え可能な言語のラベルです。
cue-list-area	字幕のリストの領域です。
cue-list	字幕のリストです。
cue	字幕です。
auto-scroll-button	自動スクロールボタンです。

状態を表すclassは以下の通りです。

has-started	プレイヤーが再生中の状態です。class「uliza-subtitle-list」と同じ要素に付きます。
linearad-has-started	プレイヤーが広告再生中の状態です。class「uliza-subtitle-list」と同じ要素に付きます。
lock	プレイヤーが画面ロックをしている状態です。class「uliza-subtitle-list」と同じ要素に付きます。
auto-scroll	自動スクロールする状態です。class「uliza-subtitle-list」と同じ要素に付きます。
open	切り替え可能な言語ラベルのリストが表示されている状態です。class「header」と同じ要素に付きます。
active	表示中の字幕です。class「cue」と同じ要素に付きます。
hover	マウスカーソルがホバー中の項目です。class「track-item」とclass「cue」と同じ要素に付きます。
non-hit	検索の条件に一致しなかった項目です。class「cue」と同じ要素に付きます。

「ユーザーガイド」の字幕リストの表示例に示すCSSはソースファイルulizahtml5-subtitle-list.cssをご確認ください。カスタマイズする場合は十分な動作検証をしてください。

マルチアングルリストのデザインをカスタマイズする

マルチアングルリストの構造は以下の通りです。

```html
<div id="..."><!-- マルチアングルリストのdiv要素 -->

  <div class="uliza-multi-angle-container">
    <div class="wrapper">
      <div class="uliza-multi-angle-list">
        <div class="item">
          <canvas class="angle"></canvas>
          <div class="poster"></div>
          <div class="cover"></div>
          <span class="label"></span>
        </div>
        <!-- <div class="item"> が並ぶ -->
      </div>
      <div class="uliza-multi-angle-cover">
        <span class="message"></span>
      </div>
    </div>
  </div>
</div>
```

要素を表すclassは以下の通りです。

uliza-multi-angle-container	マルチアングルリストの実体です。
uliza-multi-angle-list	項目の一覧です。
item	アングルの項目です。
angle	映像です。
poster	ポスター画像です。
cover	ホバー時のカバーです。また、ラベルの背景です。
label	各項目のラベルです。また、選択時の枠です。
uliza-multi-angle-cover	操作不可時のカバーです。
message	操作不可時の文言です。

状態を表すclassは以下の通りです。

has-started	プレイヤーが再生中の状態です。class「uliza-multi-angle-container」と同じ要素に付きます。
cannot-operate	マルチアングルリストを操作できない状態です。class「uliza-multi-angle-container」と同じ要素に付きます。
selected	選択中の項目です。class「item」と同じ要素に付きます。
hover	マウスカーソルがホバー中の項目です。class「item」と同じ要素に付きます。

「ユーザーガイド」のマルチアングルリストの表示例に示すCSSはソースファイルulizahtml5-multi-angle.cssをご確認ください。カスタマイズする場合は十分な動作検証をしてください。

改版履歴

v1.29.0 2024/10/8

- 【実装方法 > ビデオビューアを抑止する】追加しました。

v1.28.0 2024/7/10

- 【参考資料】変更しました。
- 【実装方法 > チャプターを使用する】変更しました。
- 【実装方法 > プレゼンテーションを表示する】追加しました。
- 【付録 > プレゼンテーション領域のデザインをカスタマイズする】追加しました。

v1.26.0 2024/3/5

- 【実装方法 > アイコンフォントを追加する】変更しました。

v1.25.0 2024/10/4

- 【実装方法 > ULIZA Video Analytics (Basic)視聴分析を使用する】削除しました。
- 【実装方法 > ULIZA Video Analytics (Basic)リアクションを使用する】削除しました。
- 以下の節からULIZA Video Analytics (Basic)連携に関する記載を削除しました。
  - 【参考資料】変更しました。

v1.23.0 2023/3/28

- 【実装方法 > ULIZA Video Analytics (Cloud)視聴分析を使用する】変更しました。
- 【実装方法 > スキンを切り替える】変更しました。
- 【実装方法 > 倍速再生を使用する】変更しました。

v1.22.0 2023/1/31

- 【実装方法 > ジェスチャーをカスタマイズする】変更しました。
- 【実装方法 > 色を設定する】削除しました。
- 【実装方法 > 次／前コンテンツボタンを表示する】変更しました。
- 【実装方法 > 字幕を使用する】変更しました。
- 【実装方法 > タイトルを表示する】変更しました。
- 【実装方法 > ULIZA Video Analytics (Cloud)リアクションを使用する】変更しました。
- 【実装方法 > ウォーターマークを表示する】変更しました。
- 【付録 > 字幕リストのデザインをカスタマイズする】変更しました。

v1.21.0 2022/11/15

- 【参考資料】変更しました。
- 【実装方法 > ULIZA Video Analytics (Cloud)視聴分析を使用する】変更しました。
- 【実装方法 > ULIZA Video Analytics (Cloud)リアクションを使用する】変更しました。
- 【実装方法 > Google Cast Senderを使用する】と【実装方法 > AirPlayを使用する】を統合しました。
- 【実装方法 > 再生開始ボタンを装飾する】変更しました。

v1.20.0 2022/7/26

- 【実装方法 > 色を設定する】変更しました。
- 【実装方法 > ULIZA Video Analytics (Cloud)リアクションを使用する】変更しました。
- 【実装方法 > 進む／戻るを使用する】変更しました。

- 【付録 > 字幕リストのデザインをカスタマイズする】変更しました。

v1.19.1 2022/6/28

- 【参考資料】変更しました。
- 【ULIZA Video Analytics (Cloud)視聴分析を使用する】追加しました。
- 【ULIZA Video Analytics (Cloud)リアクションを使用する】追加しました。

v1.19.0 2022/6/14

- 【任意のデバイス識別子を設定する】削除しました。
- 【シーク範囲制御を使用する】変更しました。

v1.18.1 2022/3/16

- 【マルチアングル動画を再生する】追加しました。
- 【マルチアングルリストのデザインをカスタマイズする】追加しました。

v1.18.0 2022/1/31

- 【自動再生する】変更しました。
- 【字幕を使用する】変更しました。
- 【画質メニューをカスタマイズする】変更しました。
- 【パーツサイズを変更する】変更しました。
- 【音声メニューをカスタマイズする】変更しました。
- 【再生開始ボタンを装飾する】追加しました。
- 【チャプター領域のデザインをカスタマイズする】変更しました。
- 【字幕リストのデザインをカスタマイズする】追加しました。
- 【ジェスチャーをカスタマイズする】追加しました。

v1.17.1 2021/10/29

- 【次／前コンテンツボタンを表示する】変更しました。

v1.17.0 2021/9/30

- 【MP4コンテンツを再生する】変更しました。
- 【HLSコンテンツを再生する】変更しました。
- 【DASHコンテンツを再生する】変更しました。
- 【オーディオを再生する】変更しました。
- 【ULIZA Video Analytics (Basic)連携を使用する】変更しました。
- 【パノラマ動画を再生する】変更しました。
- 【Google Cast Senderを使用する】変更しました。
- 【AirPlayを使用する】変更しました。
- 【スキンを切り替える】変更しました。
- 【ジェスチャーをカスタマイズする】変更しました。
- 【チャプターを使用する】変更しました。
- 【タイトルを表示する】追加しました。

v1.16.0 2021/6/30

- 「2.1. ページにプレイヤーを埋め込む」を更新
- 「2.1.1. MP4コンテンツを再生する」を更新
- 「2.1.2. HLSコンテンツを再生する」を更新
- 「2.1.3. DASHコンテンツを再生する」を更新
- 「2.1.4. オーディオを再生する」を更新
- 「2.6.1. リニア広告を小窓で表示する」を更新

- 「2.10. 倍速再生を使用する」を更新
- 「2.13. パノラマ動画を再生する」を更新
- 「2.18. 字幕を使用する」を更新
- 「2.19. Google Cast Senderを使用する」を更新
- 「2.20. AirPlayを使用する」を更新
- 「2.22. 画質リストをカスタマイズする」を更新
- 「2.23. スキンを切り替える」を更新
- 「2.31. ジェスチャーをカスタマイズする」を更新
- 「2.40. パーツサイズを変更する」を更新
- 「2.41. チャプターを使用する」を更新
- 「2.48. 音声リストをカスタマイズする」を更新
- 「2.51. ピクチャーインピクチャーを使用する」を追加

v1.15.4 2021/4/19

- 「3.1. チャプター領域のデザインをカスタマイズする」を更新

v1.15.1 2021/3/3

- 一部文言を統一

v1.15.0 2021/1/25

- 全体的な書式を更新
- 「2.6. リニア広告を表示する」を更新
- 「2.8. 再生前画像を表示する」を削除
- 「2.8. 再生前画像／再生前スライドショーを使用する」を更新
- 「2.16. コントロールバーボタンを追加する」を更新
- 「2.24. ホットキーを使用する」を更新
- 「2.33. リピートする」を更新
- 「2.40. パーツサイズを変更する」を更新
- 「2.47. 音声を切り替える」を追加
- 「2.48. 音声メニューをカスタマイズする」を追加
- 「2.49. カスタムレイヤーを表示する」を追加
- 「2.50. リモートシークを抑止する」を追加
- 「3.1. チャプター領域のデザインをカスタマイズする」を更新

v1.14.1 2020/10/2

- 「2.49. 外部出力／録画制御を使用する」を追加

v1.14.0 2020/9/17

- 「1.2. 参考資料」を更新
- 「2.19. コントロールバーボタンを追加する」を更新
- 「2.34. ジェスチャーを使用する」を更新
- 「2.41. 画面ロックを使用する」を追加
- 「2.42. 切り出し再生を使用する」を追加
- 「2.43. パーツサイズを変更する」を追加
- 「2.44. チャプターを使用する」を追加
- 「2.45. シーク範囲制御を使用する」を追加
- 「2.46. ヘルプを表示する」を追加
- 「2.47. 開始位置シークを使用する」を追加
- 「2.48. 次／前コンテンツを使用する」を追加
- 「3. 付録」を追加

v1.13.3 2020/7/3

- 「2.20. センターコントローラーをカスタマイズする」を追加
- 「2.39. 自動リロードを使用する」を追加

v1.13.0 2020/4/30

- 「2.6. ULIZA Video Analytics (Basic)連携を使用する」を更新
- 「2.13. 倍速再生を使用する」を更新
- 「2.15. ボリュームボタンの表示／非表示を設定する」を更新
- 「2.16. パノラマ動画を再生する」を更新
- 「2.18. 色を指定する」を更新
- 「2.18.1コントロールバーに背景色を指定する」を削除
- 「2.20. 字幕を使用する」を更新
- 「2.24. 画質メニューをカスタマイズする」を更新
- 「2.35. リピートを使用する」を追加
- 「2.36. 設定メニューを使用する」を追加
- 「2.37. 自動サイズ調整を使用する」を追加
- 以下の章で組み込む必要がなくなったjsについての記述を削除
  - 「2.5. HLSコンテンツを再生する」
  - 「2.6. ULIZA Video Analytics (Basic)連携を使用する」
  - 「2.8. リニア広告を表示する」
  - 「2.8.1. リニア広告をPicture in Pictureで表示する」
  - 「2.9. コンパニオン広告を表示する
  - 「2.16. パノラマ動画を再生する」
  - 「2.21. Google Cast Senderを使用する」
  - 「2.22. AirPlayを使用する」
  - 「2.24. 画質メニューをカスタマイズする」
  - 「2.32. リアクションを使用する」

v1.12.1 2020/1/31

- 「2.1.2. autoplay（自動再生）の利用について」を削除
- 「2.1.3. 再生開始時のミュートについて」を削除
- 「2.2. 自動再生する」を追加
- 「2.3. ミュート状態で再生開始する」を追加
- 「2.5. HLSコンテンツを再生する」を更新
- 「2.6. Windows7のIE11でHLSコンテンツを再生する」を削除
- 「2.23. 画質切り替えを使用する」を更新
- 「2.24. 画質メニューをカスタマイズする」を追加

v1.12.0 2019/9/13

- API and User Guideの3章を独立させて本ドキュメントとした
- deprecatedとなったAPI名、および初期化オプション名を修正
- 「2.1. ページにプレイヤーを埋め込む」を更新
- 「2.1.2. autoplay（自動再生）の利用について」を更新
- 「2.1.3. 再生開始時のミュートについて」を更新
- 「2.1.4. Android向けのレイアウトについて」を更新
- 「2.4. ULIZA Video Analytics (Lite/Professional)連携を使用する」を削除
- 「2.8. 再生前画像を表示する」を更新
- 「2.29. リアクションを使用する」を追加
- 「2.30. ジェスチャーカスタマイズを使用する」を追加
- 「2.31. 再生中の初期化オプションを変更する」を追加
- 「2.32. フォントを追加する」を追加