官方网站: <https://docs.videojs.com/index.html>

下载最新版videojs: <https://github.com/videojs/video.js/releases>

源码地址：<https://github.com/videojs/video.js>

官方文档：<https://github.com/videojs/video.js/blob/main/docs/guides/options.md>

查看DEMO: <http://liuyandeng.gitee.io/cgfy-html-ui/>

DEMO下载: <https://gitee.com/liuyandeng/cgfy-html-ui>

# 入门使用

**1. 引入video.js和video-js.css**

|  |
| --- |
| <link href="https://cdnjs.cloudflare.com/ajax/libs/video.js/7.3.0/video-js.min.css"  rel="stylesheet">  <script src="https://cdnjs.cloudflare.com/ajax/libs/video.js/7.3.0/video.min.js">  </script> |

**2.使用video标签就像下面这样：**

|  |
| --- |
| <video id="example\_video\_1"  class="video-js vjs-default-skin"  controls  preload="none"  width="640"  height="264"  poster="http://vjs.zencdn.net/v/oceans.png">  <source src="http://vjs.zencdn.net/v/oceans.mp4" type="video/mp4">  </video> |

**3.videojs使用方式就是以类似的方式开始的，不过由于我们借助videojs对视频进行一些控制或制定**

|  |
| --- |
| var player = videojs('example\_video\_1',{  muted: true,  controls : true/false,  height:300,  width:300,  loop : true,  // 更多配置.....  }); |

# 常用事件

1. 播放 this.play()

2. 停止 – video没有stop方法，可以用pause 暂停获得同样的效果

3. 暂停 this.pause()

4. 销毁 this.dispose()

5. 监听 this.on(‘click‘,fn)

6. 触发事件this.trigger(‘dispose‘)

|  |
| --- |
| var options = {};    var player = videojs(‘example\_video\_1‘, options, function onPlayerReady() {  videojs.log(‘播放器已经准备好了!‘);    // In this context, `this` is the player that was created by Video.js.<br> // 注意，这个地方的上下文， `this` 指向的是Video.js的实例对像player  this.play();    // How about an event listener?<br> // 如何使用事件监听？  this.on(‘ended‘, function() {  videojs.log(‘播放结束了!‘);  });  }); |

# 常用选项

autoplay : true/false 播放器准备好之后，是否自动播放 【默认false】

controls : true/false 是否拥有控制条 【默认true】,如果设为false ,那么只能通过api进行控制了。也就是说界面上不会出现任何控制按钮

height: 视频容器的高度，字符串或数字 单位像素 比如： height:300 or height:‘300px‘

width: 视频容器的宽度, 字符串或数字 单位像素

loop : true/false 视频播放结束后，是否循环播放

muted : true/false 是否静音

poster: 播放前显示的视频画面，播放开始之后自动移除。通常传入一个URL

preload:预加载

‘auto‘ 自动

’metadata‘ 元数据信息 ，比如视频长度，尺寸等

‘none‘ 不预加载任何数据，直到用户开始播放才开始下载

children: Array | Object 可选子组件 从基础的Component组件继承而来的子组件，数组中的顺序将影响组件的创建顺序哦。

# options 选项

## 标准元素选项

这些选项中的每一个也可用作标准元素属性 ; 因此，可以使用设置指南中列出的所有三种方式定义它们。通常，未列出默认值，因为这是留给浏览器供应商的。

## autoplay

类型： boolean|string

注意1:在这一点上，自动播放属性和选项并不能保证你的视频将自动播放。

注意2:如果在媒体元素上有一个属性，该选项将被忽略。

注意3:不能在属性中传递字符串值，必须在videojs选项中传递它

您应该向videojs函数传递一个自动播放选项，而不是使用autoplay属性。以下值有效:

布尔值false:与视频元素没有属性相同，不会自动播放

布尔值true:与在视频元素上拥有属性相同，将使用浏览器自动播放

字符串值'mute ':将静音视频元素，然后在loadstart上手动调用play()。这可能会奏效。

字符串值'play':将在loadstart上调用play()，类似于浏览器的自动播放

字符串值'any':将在loadstart上调用play()，如果promise被拒绝，它将静音视频元素，然后调用play()。

## controls

类型： boolean

确定播放器是否具有用户可以与之交互的控件。没有控件，启动视频播放的唯一方法是使用autoplay属性或通过Player API。

## height

类型： string|number

设置视频播放器的显示高度（以像素为单位）。

## loop

类型： boolean

使视频一结束就重新开始。

## muted

类型： boolean

默认情况下会静音任何音频。

## poster

类型： string

在视频开始播放之前显示的图像的URL。这通常是视频的框架或自定义标题屏幕。一旦用户点击“播放”，图像就会消失。

## preload

类型： string

建议浏览器是否应在加载元素后立即开始下载视频数据。支持的值是：

**‘auto’**

立即开始加载视频（如果浏览器支持）。某些移动设备不会预加载视频，以保护用户的带宽/数据使用。这就是为什么这个价值被称为’汽车’，而不是更具决定性的东西’true’。

这往往是最常见和推荐的值，因为它允许浏览器选择最佳行为。

**‘metadata’**

仅加载视频的元数据，其中包括视频的持续时间和尺寸等信息。有时，元数据将通过下载几帧视频来加载。

**‘none’**

不要预加载任何数据。浏览器将等待用户点击“播放”开始下载。

## src

类型： string

要嵌入的视频源的源URL。

## width

类型： string|number

设置视频播放器的显示宽度（以像素为单位）。

# Video.js特定的选项

undefined除非另有说明，否则默认情况下每个选项

## aspectRatio

类型： string

将播放器置于流体模式，并在计算播放器的动态大小时使用该值。该值应表示比率 - 由冒号（例如"16:9"或"4:3"）分隔的两个数字。

## autoSetup

类型： boolean

阻止播放器为具有data-setup属性的媒体元素运行autoSetup 。

注意：必须在与videojs.options.autoSetup = falsevideojs源加载生效的同一时刻全局设置。

## breakpoints

当与响应选项一起使用时，设置断点，将配置类名如何在播放器上切换，以根据播放器的维度调整UI。

## children

类型： Array|Object

此选项继承自基Component类。

## fluid

类型： boolean

何时true，Video.js播放器将具有流畅的大小。换句话说，它将扩展以适应其容器。

此外，如果元素具有"vjs-fluid"，则此选项自动设置为true。

## inactivityTimeout

类型： number

Video.js表示用户通过"vjs-user-active"和"vjs-user-inactive"类以及"useractive"事件与播放器进行交互。

在inactivityTimeout决定了不活动的许多毫秒声明用户闲置之前是必需的。值为0表示没有inactivityTimeout，用户永远不会被视为非活动状态。

## language

键入：string，默认值：浏览器默认值或’en’

与播放器中的一种可用语言匹配的语言代码。这为播放器设置了初始语言，但始终可以更改。

在Video.js中了解有关语言的更多信息。

## languages

类型： Object

自定义播放器中可用的语言。此对象的键将是语言代码，值将是具有英语键和翻译值的对象。

在Video.js中了解有关语言的更多信息

注意：通常，不需要此选项，最好将自定义语言传递给videojs.addLanguage()所有播放器！

## liveui

类型:boolean默认值:false

允许播放器使用新的实时ui，包括:

在活动窗口中寻找的进度条

一个按钮，点击它可以寻求到活动的边缘，并使用圆圈指示是否处于活动的边缘。

如果没有这个选项，进度条将被隐藏，在它的位置将是指示实时回放的文本。没有进度控制，你将不能点击文本寻求到实时边缘。在未来的版本中，liveui将默认为true !

## nativeControlsForTouch

类型： boolean

明确设置关联技术选项的默认值。

## notSupportedMessage

类型： string

允许覆盖Video.js无法播放媒体源时显示的默认消息。

## fullscreen

默认值:{options: {navigationUI: 'hide'}

全屏。选项可以设置为传递特定的全屏选项。在某种程度上，它将被添加元素和处理程序以获得更多的功能。

选项

类型:对象默认值:{navigationUI: 'hide'}

更多细节请参阅[Fullscreen AP](https://fullscreen.spec.whatwg.org/" \l "dictdef-fullscreenoptions)I规范。

## playbackRates

类型： Array

严格大于0的数字数组，其中1表示常速（100％），0.5表示半速（50％），2表示双速（200％）等。如果指定，Video.js显示控件（类vjs-playback-rate）允许用户从选择数组中选择播放速度。选项以从下到上的指定顺序显示。

例如：

|  |
| --- |
| videojs('my-player', {  playbackRates: [0.5, 1, 1.5, 2]  }); |

## plugins

类型： Object

这支持在初始化播放器时使用自定义选项自动初始化插件 - 而不是要求您手动初始化它们。

|  |
| --- |
| videojs('my-player', {  plugins: {  foo: {bar: true},  boo: {baz: false}  }  }); |

以上大致相当于：

|  |
| --- |
| var player = videojs('my-player');  player.foo({bar: true});  player.boo({baz: false}); |

虽然，由于plugins选项是对象，因此无法保证初始化顺序！

有关Video.js插件的更多信息，请参阅插件指南。

## responsive

响应

类型:boolean，默认值:false

将此选项设置为true将导致玩家基于响应性断点自定义(参见:breakpoints选项)。

当该选项为false(默认值)时，响应性断点将被忽略。

## sources

类型： Array

一组对象，它们反映了本机元素具有一系列子元素的能力。这应该是带有src和type属性的对象数组。例如：

|  |
| --- |
| videojs('my-player', {  sources: [{  src: '//path/to/video.mp4',  type: 'video/mp4'  }, {  src: '//path/to/video.webm',  type: 'video/webm'  }]  }); |

使用元素将具有相同的效果：

|  |
| --- |
| <video ...>  <source src="//path/to/video.mp4" type="video/mp4">  <source src="//path/to/video.webm" type="video/webm">  </video> |

## suppressNotSupportedError

类型:布尔

如果设置为true，则no compatible source错误不会立即触发，而是会在第一次用户交互时发生。这对谷歌的“mobile friendly”测试工具很有用，它不能播放视频，但你可能不想看到一个错误显示。

## techCanOverridePoster

类型： boolean

使技术人员有可能覆盖播放器的海报并融入播放器的海报生命周期。当使用多个技术时，这可能很有用，每个技术都必须在播放新源时设置自己的海报。

## techOrder

输入：Array，默认值：[‘html5’]

定义Video.js技术首选的顺序。默认情况下，这意味着Html5首选技术。其他注册的技术将在此技术之后按其注册顺序添加。

## userActions

类型:对象

### userActions.doubleClick

类型:布尔|函数

控制如何双击玩家/技术操作。如果设置为false，则禁用双击。如果未定义或设置为true，双击将启用并切换全屏模式。要覆盖默认的双击处理，设置userActions。doubleClick函数接受dblclick事件:

|  |
| --- |
| function myDoubleClickHandler(事件)= {  // `this` is the player in this context  this.pause ();  };  videojs(“my-player”,{  userActions: {  doubleClick:myDoubleClickHandler  }  }); |

### userActions.hotkeys

类型:布尔函数| |对象

控制玩家范围内热键的操作方式。如果设置为false或未定义，热键将被禁用。如果设置为true或一个对象(允许下面的fullscreenKey等定义)，热键将被启用，如下所述。若要覆盖默认热键处理，请设置userActions。将热键转换为接受按键事件的函数:

|  |
| --- |
| var player = videojs('my-player', {  userActions: {  hotkeys: function(event) {  // `this` is the player in this context  // `x` key = pause  if (event.which === 88) {  this.pause();  }  // `y` key = play  if (event.which === 89) {  this.play();  }  }  }  }); |

### userActions.hotkeys.fullscreenKey

类型:功能

覆盖全屏键定义。如果设置了这个参数，函数将接收keydown事件;如果函数返回true，则执行全屏切换操作。

var player = videojs('my-player'， {

userActions: {

热键:{

muteKey:函数(事件){

//禁用静音键

},

fullscreenKey:函数(事件){

//当按下v键时覆盖全屏触发

返回(事件。= = = 86);

}

}

}

});

### userActions.hotkeys.muteKey

类型:功能

覆盖静音键定义。如果设置了这个参数，函数将接收keydown事件;如果函数返回true，则执行静音切换操作。

### userActions.hotkeys.playPauseKey

类型:功能

覆盖播放/暂停键的定义。如果设置了这个参数，函数将接收keydown事件;如果函数返回true，则执行播放/暂停切换操作。

## vtt.js

类型： string

允许覆盖vtt.js的默认URL，该URL可以异步加载到polyfill支持WebVTT。

此选项将用于Video.js（即video.novtt.js）的“novtt”版本中。否则，vtt.js与Video.js捆绑在一起。

# 组件选项

Video.js播放器是一个组件。与所有组件一样，您可以定义它包含的子项，它们出现的顺序以及传递给它们的选项。

这是一个快速参考; 因此，有关Video.js中组件的更多详细信息，请查看组件指南。

## children

类型： Array|Object

如果Array- 这是默认值 - 这用于确定哪些子节点（按组件名称）以及在播放器（或其他组件）上创建它们的顺序：

|  |
| --- |
| // The following code creates a player with ONLY bigPlayButton and  // controlBar child components.  videojs('my-player', {  children: [  'bigPlayButton',  'controlBar'  ]  }); |

该children选项还可以作为传递Object。在这种情况下，它用于提供options任何/所有孩子，包括禁用它们false：

|  |
| --- |
| // This player's ONLY child will be the controlBar. Clearly, this is not the  // ideal method for disabling a grandchild!  videojs('my-player', {  children: {  controlBar: {  fullscreenToggle: false  }  }  }); |

## ${componentName}

类型： Object

可以通过组件名称的低驼峰案例变体（例如controlBarfor ControlBar）为组件提供自定义选项。这些可以嵌套在孙子关系的表示中。例如，要禁用全屏控件：

|  |
| --- |
| videojs('my-player', {  controlBar: {  fullscreenToggle: false  }  }); |

# 技术选择

## ${techName}

类型： Object

Video.js回放技术（即“技术”）可以作为传递给该videojs功能的选项的一部分给予自定义选项。它们应该在技术名称的小写变体下传递（例如"flash"或"html5"）。

**flash**

swf

指定Video.js SWF文件在Flash技术位置的位置：

|  |
| --- |
| videojs('my-player', {  flash: {  swf: '//path/to/videojs.swf'  }  }); |

但是，更改全局默认值通常更合适：

videojs.options.flash.swf = ‘//path/to/videojs.swf’

## html5

### nativeControlsForTouch

类型： boolean

只有技术支持Html5，此选项可以设置true为强制触摸设备的本机控件。

### nativeAudioTracks

类型： boolean

可以设置为false禁用本机音轨支持。最常用于videojs-contrib-hls。

### nativeTextTracks

类型： boolean

可以设置为false强制模拟文本轨道而不是本机支持。该nativeCaptions选项也存在，但只是一个别名nativeTextTracks。

### nativeVideoTracks

类型： boolean

可以设置为false禁用本机视频轨道支持。最常用于videojs-contrib-hls。

### preloadTextTracks

类型:布尔

可以设置为false，以延迟加载非活动文本轨道直到使用。这可能会在切换字幕时造成短暂的延迟，在此期间可能会缺少字幕。

默认行为是预加载所有文本轨道。