Video.js 使用教程 - 手把手教你基于 Vue 搭建 HTML 5 视频播放器

作者：HiJiangChuan
原文链接：https://ld246.com/article/1648494657381
来源网站：链滴

本文首发：《Video.js 使用教程 - 手把手教你基于 Vue 搭建 HTML 5 视频播放器》

Video.js 是最强大的网页嵌入式 HTML 5 视频播放器的组件库之一，也是大多数人首选的网页视频播解决方案。复杂的网页视频渲染，在引入 Video.js 后，轻松解决。本教程手把手教你搭建一套基于 V e 的 Video.js 视频播放页。

跟随本教程学习，最终你可以自己搭建一套可以播放本地视频文件及网络流媒体的网页视频播放器。习如何修改 video.js 的默认样式来实现播放按钮自定义形状(圆形)、居中及播放时间的显示与否， 如播放 m3u8 格式，以及如何使用 video 的属性、事件及方法，音量增减，最终实现一个功能齐全的频播放器。

跟随本教程学习，搭建的最终 video.js HTML5 视频播放效果。

另外，这个世界已经悄然发生变化，现在根本无需写任何前端代码，直接使用卡拉云 —— 新一代低码开发工具帮你搭建后台工具，卡拉云可一键接入常见数据库及 API ，无需懂前端，内置完善的各类端组件，无需调试，拖拽即用。原来三天的工作量，现在 1 小时搞定，谁用谁知道，用上早下班，详本文文末。

配置 Vue 环境 - 基础部分

通过 npm 安装 Vue 脚手架 vue-cli

npm install -g @vue/cli

然后创建一个 Vue 项目 kalacloud-vue-video

vue create kalacloud-video

选择合适的选项后，安装完成，通过 cd 命令进入 kalacloud-vue-video 目录(此目录为我们的主开目录)，使用 npm run serve 命令，可以将项目运行起来。

```
App running at:
- Local:   http://localhost:8080/
- Network: http://172.18.51.206:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

在 Vue 中使用 videojs

首先使用 npm 安装 video.js

npm i video.js

安装完毕后，在 main.js 中进行引入

```
import videojs from "video.js";
import "video.js/dist/video-js.css";
Vue.prototype.$video = videojs;
```

为了代码复用性，我们来创建一个 PlayerVideo 组件。在组件中我们分别定义 template 部分和 script 如下

最开始我们选用默认源链接: https://playtv-live.ifeng.com/live/06OLEGEGM4G.m3u8

```
<!-- controls：向用户显示播放按钮控件 -->
<template>
  <video
    ref="video"
    class="video-js vjs-default-skin"
    width="600"
    height="400"
    controls
  >
    <source src="https://playtv-live.ifeng.com/live/06OLEGEGM4G.m3u8" />
  </video>
</template>
export default {
  data() {
    return {
      player: null, // 用来存储当前 video
    };
  },
  mounted() { // 渲染视频
    this.player = this.$video(this.$refs.video);
  },
};
```

然后删除掉初始化 vue 项目默认的 App.vue 内代码，将 PlayerVideo 组件添加到 App 中，并调整放器至中间。

```
<template>
  <div id="app">
```

```
    <div class="video-content">
      <player-video :src="src"></player-video>
    </div>
  </div>
</template>
<script>
  import PlayerVideo from "./components/PlayerVideo.vue";
  export default {
    components: {
      PlayerVideo,
      data() {
        return {
          src: "http://vjs.zencdn.net/v/oceans.mp4",
        };
      },
    },
  };
</script>
<style lang="scss">
  #app {
    font-family: Avenir, Helvetica, Arial, sans-serif;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
    text-align: center;
    color: #2c3e50;

    .video-content {
      display: flex;
      justify-content: center;
      flex-direction: column;
      align-items: center;
    }
  }
</style>
```

在 kalacloud-vue-video 根目录使用 npm 运行下列命令:

```
npm run serve
```

在浏览器打开 http://localhost:8080/ 就可以成功渲染视频。

我们大致的来看一下目前视频播放器拥有的功能:

- 播放与暂停功能(目前播放按钮位于左上角)
- 可以调节音量
- 支持全屏与小屏播放

同样我们也可以发现一些不符合日常习惯的地方:

- 播放按钮通常位于中间
- 播放按钮一般为圆形
- 暂停时会显示播放按钮

下面我们就讲述一些 tips 优化一下播放器。

扩展阅读：《最好用的 6 款 Vue 拖拽组件库推荐》

如何使 Video.js 播放按钮垂直居中

将播放按钮垂直居中非常容易实现，video 官方提供了 vjs-big-play-centered。给 <video> 标签添 vjs-big-play-centered 类名就可以实现播放按钮垂直居中。

```
<video class="video-js vjs-default-skin vjs-big-play-centered"></video>
```

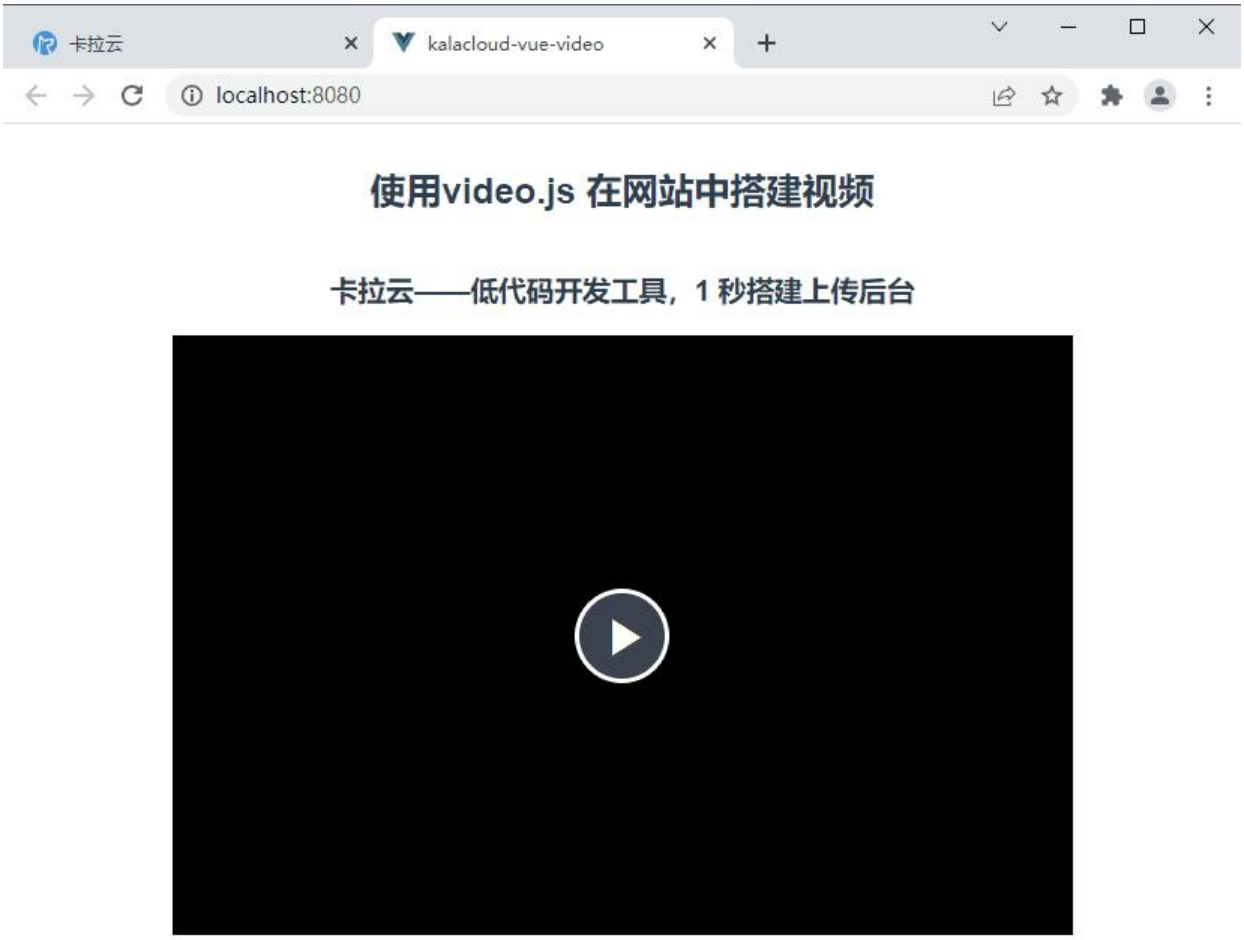
如何修改 Video.js 播放按钮为圆形

修改播放按钮为圆形需要修改对应类名的 CSS 样式。我们在 PlayerVideo 组件的 style 中添加下列式代码。

修改时属性必须设置为 !important ，否则不会生效。

```
.video-js .vjs-big-play-button {
  font-size: 2.5em !important;
  line-height: 2.3em !important;
  height: 2.5em !important;
  width: 2.5em !important;
  -webkit-border-radius: 2.5em !important;
  -moz-border-radius: 2.5em !important;
  border-radius: 2.5em !important;
  background-color: #73859f;
  background-color: rgba(115, 133, 159, 0.5) !important;
  border-width: 0.15em !important;
  margin-top: -1.25em !important;
  margin-left: -1.75em !important;
}
.vjs-big-play-button .vjs-icon-placeholder {
  font-size: 1.63em !important;
```

```
}
```

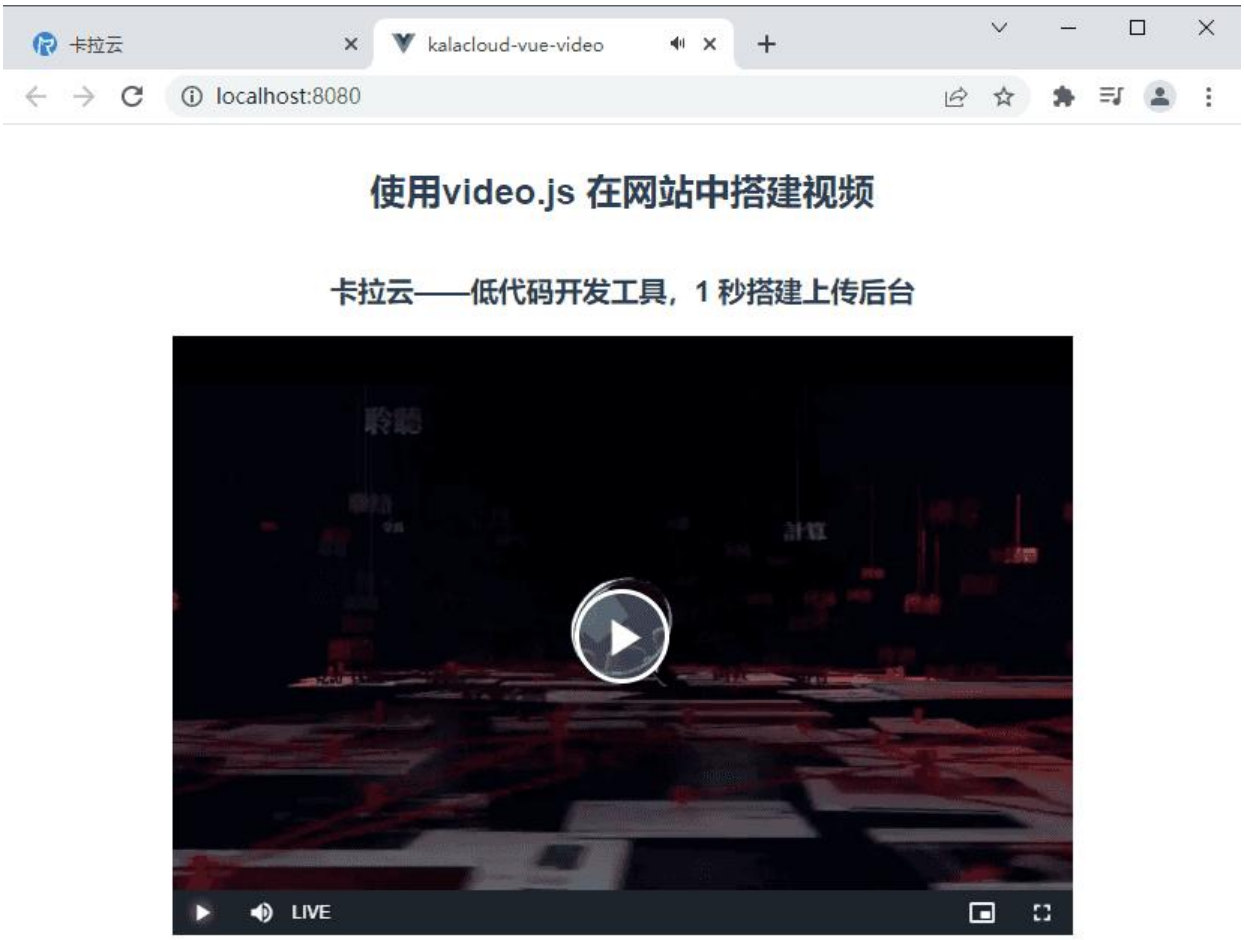


扩展阅读：《最好用的 12 款 Vue Timepicker 时间日期选择器测评推荐》

如何修改 Video.js 暂停时显示播放按钮

这个功能同样可以通过修改 CSS 实现。在 PlayerVideo 组件的 style 中添加下列样式代码。

```
.vjs-paused .vjs-big-play-button,
.vjs-paused.vjs-has-started .vjs-big-play-button {
  display: block !important;
}
```



如何设置 Video.js 显示当前播放时间

通过修改两个类的状态可以实现显示播放时间的功能，在 PlayerVideo 组件中设置下列样式代码:

```
.video-js .vjs-time-control {
  display: block !important;
}
.video-js .vjs-remaining-time {
  display: none !important;
}
```

原文链接：Video.js 使用教程 - 手把手教你基于 Vue 搭建 HTML 5 视频播放器

扩展阅读：《vue.draggable 入门指南 - 手把手教你开发任务看板》

进阶使用部分

基础部分我们使用 this.$video(this.$refs.video) 渲染出播放器，但 this.$video 本质上是 video.js 供的 videojs 函数，videojs 函数共有三个参数，第一个参数是绑定播放器的元素，第二参数为 options 对象，提供播放器的配置项，第三个参数为播放器渲染后的回调函数。

我们给 PlayerVideo 组件的 data 添加 options 对象，并设置 controls 为 false，同时设定一个简单回调函数。

controls 属性是用来控制播放器是否具有与用户交互的控件——播放按钮等。如果设置为 false ，播器将不显示播放控件，那么视频只能通过 Player API 或者 autoplay 控制播放。

```
export default {
  data() {
    return {
      player: null,
      options: { controls: false },
    };
  },
  mounted() {
    this.player = this.$video(this.$refs.video, this.options, () => {
      alert("播放器渲染完成");
    });
```

```
  },
};
```

我们可以发现，播放器渲染完成后，浏览器发出了通知，并且播放器上没有控件出现。

更多的配置项链接: video-options

Video.js 常用事件

video 提供了很多常用事件，我们可以通过监测事件来处理不同的逻辑。

例如监测 play/pause 事件，给用户发送提醒

修改 PlayerVideo 组件中的 mounted 方法:

```
mounted() {
  this.player = this.$video(this.$refs.video, this.options, function () {
    this.on("play", () => {
      alert("视频已经开始播放，祝你有好的体验");
    });
    this.on("pause", () => {
      alert("视频已经暂停");
    });
  });
}
```

扩展阅读：《Vue 实现 PDF 文件在线预览 - 手把手教你写 Vue PDF 预览功能》

Video.js 音量记忆功能

为了让大家更能理解监听事件的用处，我们举一下实际的案例: 音量记忆功能。

为了更好的用户体验，用户每次调整音量后，我们应该帮其记住当前音量。当用户刷新页面或者重新入页面后，无需再次调整音量。

这个功能其实不难实现:

- 监听 volumechange 事件，当用户修改音量时，把此音量存储到 localStorage 中(如果音量功会有多个组件使用，建议同时存放在 Vuex 中)
- 当页面刷新或进入页面后，从 localStorage 中取出音量值，同步设置播放器音量。

我们修改一下 PlayerVideo 组件，使其可以接受属性 volume 音量值，同时添加事件 volumechange 和 play 事件的监听。

当 volumechange 触发时，将当前音量值存储到 localStorage 中；当 play 事件触发时，更新音量。

```
<template>
  <video
    ref="video"
    controls
```

```
    class="video-js vjs-default-skin vjs-big-play-centered"
    width="600"
    height="400"
  >
    <source src="https://playtv-live.ifeng.com/live/06OLEGEGM4G.m3u8" />
  </video>
</template>

<script>
  export default {
    // 接受App传值
    props: ["volume"],
    data() {
      return {
        player: null,
        // volumnVideo 记录音量
        volumeVideo: this.volume,
      };
    },
    mounted() {
      const _this = this;
      this.player = this.$video(this.$refs.video, this.options, function () {
        this.on("volumechange", () => {
          // 存储音量
          _this.volumeVideo = this.volume();
          window.localStorage.volume = this.volume();
        });
        this.on("play", () => {
          this.volume(this.volumeVideo);
        });
      });
    },
  };
</script>
```

同时在 App.vue 中使用全局钩子函数 beforeCreate 获取到 localStorage 中存取的 volume

```
beforeCreate() {
  this.volume = window.localStorage.volume;
},
```

通过上述步骤，就可以成功实现音量记忆功能。

扩展阅读：《顶级好用的 8 款 Vue 弹窗组件测评与推荐》

Video.js 简单视频播放器搭建

下面我带大家实现一下播放器的各种控制方法: 开始、暂停、重新加载、快进、后退、增大音量、降音量以及换台功能。

video.js 对于这些控制方法都对应提供了方法。我们只需对提供的方法略作封装，即可使用。

下面我们就利用 video.js 提供的方法实现一个简单的播放器功能。

我们分别修改 PlayerVideo 组件和 App 组件如下:

PlayerVideo.vue 代码如下:

```
<template>
  <video
    ref="video"
    controls
    class="video-js vjs-default-skin vjs-big-play-centered"
    width="600"
    height="400"
  >
    <source :src="src" />
  </video>
</template>
```

```
<script>
  export default {
    props: ["volume", "src"],
    data() {
      return {
        player: null,
        volumeVideo: this.volume,
      };
    },
    methods: {
      // 封装播放器方法
      play() {
        this.player.src({ src: this.src });
        this.player.load(this.src);
        this.player.play(this.volumeVideo);
      },
      stop() {
        this.player.pause();
      },
      reload() {
        this.stop();
        this.player.load({});
        this.play();
      },
      forward() {
        const currentTime = this.player.currentTime();
        this.player.currentTime(currentTime + 5);
      },
      back() {
        const currentTime = this.player.currentTime();
        this.player.currentTime(currentTime - 5);
      },
      volumeUp() {
        this.player.volume(this.volumeVideo + 0.1);
      },
      volumeDown() {
        this.player.volume(this.volumeVideo - 0.1);
      },
      toggleTv(obj) {
        this.player.src(obj.src);
        this.player.load(obj.load);
        this.player.play(this.volumeVideo);
      },
    },
    mounted() {
      const _this = this;
      this.player = this.$video(this.$refs.video, this.options, function () {
        this.on("volumechange", () => {
          // 存储音量
          _this.volumeVideo = this.volume();
          window.localStorage.volume = this.volume();
        });
        this.on("play", () => {
```

```
        this.volume(this.volumeVideo);
      });
    });
  },
};
</script>

<style>
 .video-js .vjs-time-control {
   display: block !important;
 }
 .video-js .vjs-remaining-time {
   display: none !important;
 }

 .video-js .vjs-big-play-button {
   font-size: 2.5em !important;
   line-height: 2.3em !important;
   height: 2.5em !important;
   width: 2.5em !important;
   -webkit-border-radius: 2.5em !important;
   -moz-border-radius: 2.5em !important;
   border-radius: 2.5em !important;
   background-color: #73859f;
   background-color: rgba(115, 133, 159, 0.5) !important;
   border-width: 0.15em !important;
   margin-top: -1.25em !important;
   margin-left: -1.25em !important;
 }
 .vjs-big-play-button .vjs-icon-placeholder {
   font-size: 1.63em !important;
 }

 .vjs-paused .vjs-big-play-button,
 .vjs-paused.vjs-has-started .vjs-big-play-button {
   display: block !important;
 }
</style>
```

App.vue 代码如下:

```
<template>
 <div id="app">
   <div class="video-content">
     <h2>使用video.js 在网站中搭建视频</h2>
     <h3>卡拉云——低代码开发工具，1 秒搭建上传后台</h3>
     <player-video :volume="volume" ref="video" :src="src"></player-video>
   </div>
   <div class="button-group">
     <el-button class="primary" @click="playVideo">开始视频</el-button>
     <el-button class="primary" @click="stopVideo">暂停视频</el-button>
     <el-button class="primary" @click="reloadVideo">重新加载</el-button>
     <el-button class="primary" @click="forwardVideo">视频快进</el-button>
     <el-button class="primary" @click="backVideo">视频后退</el-button>
```

```
      <el-button class="primary" @click="volumeUpVideo">增大音量</el-button>
      <el-button class="primary" @click="volumeDownVideo">降低音量</el-button>
      <el-button class="primary" @click="toggleToFenghuangwang"
        >凤凰卫视</el-button
      >
      <el-button class="primary" @click="toggleToDefault">默认频道</el-button>
    </div>
  </div>
</template>
<script>
  import PlayerVideo from "./components/PlayerVideo.vue";

  export default {
    components: {
      PlayerVideo,
    },
    data() {
      return {
        volume: 0.5,
        src: "http://vjs.zencdn.net/v/oceans.mp4",
      };
    },
    computed: {
      video() {
        return this.$refs.video;
      },
    },
    methods: {
      // 父类组件调用子组件方法，触发播放器功能
      stopVideo() {
        this.video.stop();
      },
      playVideo() {
        this.video.play();
      },
      reloadVideo() {
        this.video.reload();
      },
      forwardVideo() {
        this.video.forward();
      },
      backVideo() {
        this.video.back();
      },
      fullScreenVideo() {
        this.video.fullScreen();
      },
      screenVideo() {
        this.video.exitScreen();
      },
      volumeUpVideo() {
        this.video.volumeUp();
      },
      volumeDownVideo() {
```

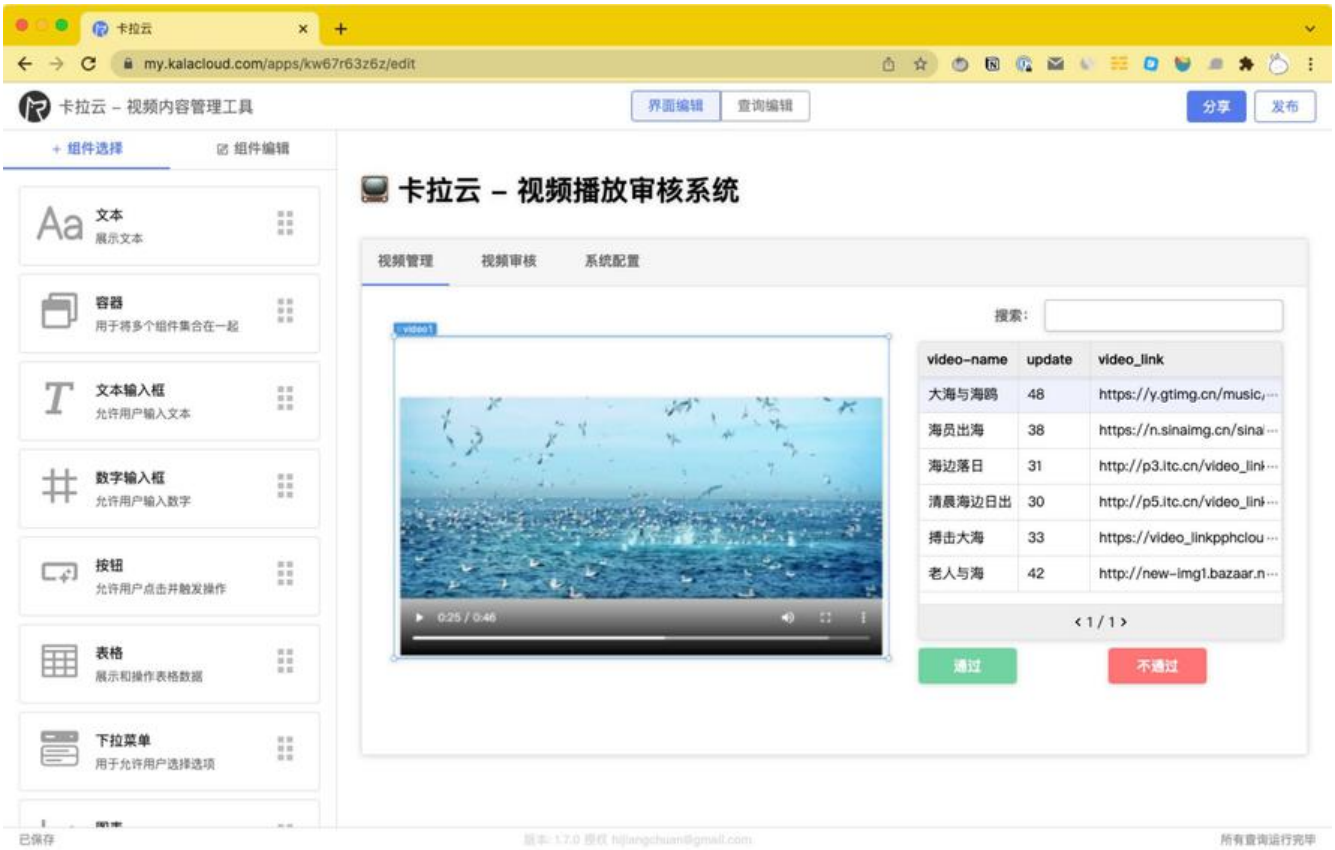
```
      this.video.volumeDown();
    },
    toggleToFenghuangwang() {
      this.video.toggleTv({
        src: {
          type: "application/x-mpegURL",
          src: "https://playtv-live.ifeng.com/live/06OLEGEGM4G.m3u8",
        },
        load: "https://playtv-live.ifeng.com/live/06OLEGEGM4G.m3u8",
      });
    },
    toggleToDefault() {
      this.video.toggleTv({
        src: {
          type: "video/mp4",
          src: "http://vjs.zencdn.net/v/oceans.mp4",
        },
        load: "http://vjs.zencdn.net/v/oceans.mp4",
      });
    },
  },
  beforeCreate() {
    this.volume = window.localStorage.volume;
  },
};
</script>
<style lang="scss">
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  display: flex;
  .video-content {
    display: flex;
    justify-content: center;
    flex-direction: column;
    align-items: center;
    margin-right: 20px;
  }
  .button-group {
    margin-top: 20px;
    display: flex;
    flex: 0 0 100px;
    flex-direction: column;
    justify-content: space-between;
    // align-items: center;
  }
}
</style>
```

搭建完成。

使用 video.js 搭建视频总结

本教程系统的带大家学习如何使用 video.js 在网站中搭建视频播放器，如果你跟着教程走下来，一定完成了和教程中一样的视频播放器。如果你觉得太复杂，不想处理前端问题，推荐使用卡拉云，卡拉是新一代低代码开发工具，内置各类前端组件，无需懂任何前端，仅需拖拽即可快速生成。

下图为使用卡拉云搭建的内部视频审核系统的 Demo 版，仅需拖拽，1小时搞定。



原文链接：Video.js 使用教程 - 手把手教你基于 Vue 搭建 HTML 5 视频播放器

卡拉云是新一代低代码开发工具，免安装部署，可一键接入包括 MySQL 在内的常见数据库及 API。根据自己的工作流，定制开发。无需繁琐的前端开发，只需要简单拖拽，即可快速搭建企业内部工具原来三天的开发工作量，使用卡拉云后可缩减至 1 小时，欢迎免费试用卡拉云。

扩展阅读：

- 最好用的 10 款 MySQL GUI 数据库管理工具横向测评 - 免费和付费到底怎么选?
- 最好用的 7 款 Vue3 admin 后台管理系统框架测评
- 最好的 6 个免费天气预报 API 接口对比测评 - 和风天气、高德天气等 API 接入详解
- 最棒的 7 个 Laravel admin 后台管理系统推荐
- Vue Router 手把手教你搭 Vue3 路由页面跳转
- React Draggable 实现拖拽组件库 - 最详细中文使用教程
- Echarts 折线图完全配置指南