

16 | 直播中心搭建（二）：如何通过VideoJs配置直播中心？

2023-05-29 Barry 来自北京

《Python实战·从0到1搭建直播视频平台》



你好，我是 Barry。

上节课我们根据游客的操作动线，完成了直播中心游客需求功能的设计和开发工作，相信你对直播中心的整体功能已经有了新的认识。

作为一个视频平台，直播功能是平台的刚需，用户通过直播可以进行授课、个人展示、带货等等，这也是当下非常主流的媒体宣传方式。

这节课我们继续推进直播模块的功能开发，把焦点放在直播功能和页面的开发实现上。这次我们先从主播的视角出发，从需求分析开始，一步步设计开发平台的直播功能。

直播功能的需求有哪些

先来梳理一下直播模块的功能需求。

我们需要从两个维度综合分析。从平台的维度来看，平台需要保证直播内容是健康、绿色、安全的，所以一定要对主播进行实名认证，这就需要我们实现认证功能；另外，从主播的维度来看，在完成实名认证之后，这时候需要提交直播相关信息并发起直播。

以上我们对功能需求就梳理完了，接下来我们就要实现每个模块的功能。

页面设计

根据前面的需求分析，我们把直播模块整体切割成两个模块。

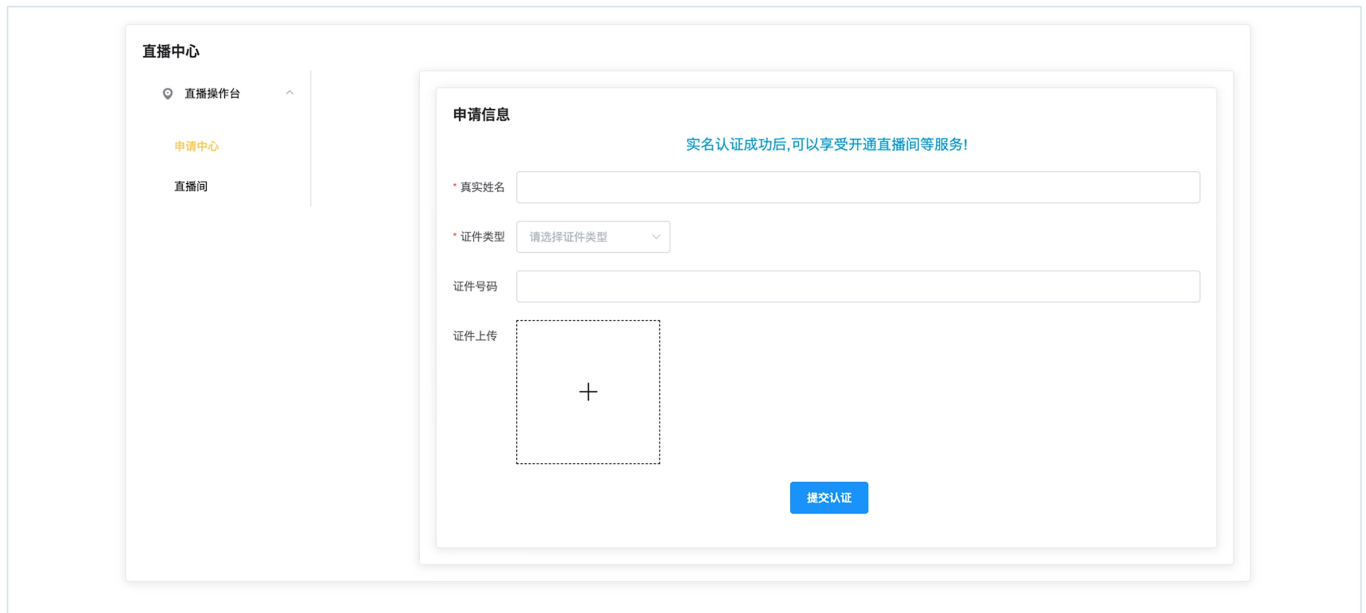
一个模块是申请中心，用户可以在该模块完成个人信息的实名认证。你可以参考后面这张表格，来了解用户需要提交的信息具体涉及的主要字段。当然，你也可以灵活添加调整，但是一定保证数据库中新增了字段。

字段	含义	类型
real_name	真实姓名	string
id_type	证件类型	string
id_number	证件号码	string
postFile	证件照片	string



明确了需要哪些用户信息之后，接下来就是设计实现界面样式，这一部分我们通过折叠菜单的形式来展示。

结合界面效果截图我们可以看到，这里有两个选择项，分别是申请中心和直播间。点击不同的菜单，用户就可以看到对应的功能区，用户无需跳转界面，这样反而能更高效地解决问题。



页面代码实战

菜单组件我们依然使用 Element 的组件，应用组件的方法和我们前面讲过的一样，相信经过前面的练习现在你已经可以应对自如了。你需要重点关注的是组件的 API 以及组件本身的一些属性方法。如果想全面学习了解菜单组件，你还可以直接查看 [🔗 menu 组件的官网链接](#)。

好，我们继续来看直播功能里页面菜单怎么实现。我们先用栅格把菜单组件做一个定位，然后开始做菜单组件的开发，具体实现代码是后面这样。

复制代码

```
1 <el-col :span="4">
2   <el-menu
3     default-active="1,1"
4     class="el-menu-vertical-demo"
5     @open="handleOpen"
6     @close="handleClose"
7     :unique-opened="true"
8     text-color="black"
9     active-text-color="#ffd04b"
10    menu-trigger="click"
11    @select="handleSelect"
12  >
13    <el-submenu index="1">
14      <template slot="title">
15        <i class="el-icon-location"></i>
16        <span>直播操作台</span>
```


```

17     </template>
18     <el-menu-item-group>
19       <el-menu-item index="1-1">申请中心</el-menu-item>
20       <el-menu-item index="1-2">直播间</el-menu-item>
21     </el-menu-item-group>
22   </el-submenu>
23 </el-menu>
24 </el-col>

```

这段代码的重点是 `<el-menu>` 的 **select 方法**，每一级菜单都有一个 `index` 值用来表示对应的层级，因此在用户点击菜单选项的时候，我们需要通过 `index` 值的判断来更换右侧展示的内容。

接下来就到了申请中心的页面实现，我们还是边看代码边分析。

 复制代码

```

1 <el-card v-if="status === '1-1'" class="card">
2   <p class="data-title">申请信息</p>
3   <p v-show="!applyStatus" class="applyTitle">您已认证成功，赶快去直播吧！</p>
4   <p v-show="applyStatus" class="applyTitle">实名认证成功后，可以享受开通直播间等服务！</p>
5   <el-form
6     ref="form"
7     :model="form"
8     :rules="formRule"
9     label-position="left"
10    label-width="80px"
11    v-show="applyStatus"
12  >
13     <el-form-item label="真实姓名" prop="name">
14       <el-input v-model="form.name"></el-input>
15     </el-form-item>
16     <el-form-item label="证件类型" prop="type">
17       <el-select v-model="form.type" placeholder="请选择证件类型">
18         <el-option label="身份证" value="1"></el-option>
19         <el-option label="护照（中国签发）" value="2"></el-option>
20         <el-option label="其他国家或地区证明" value="3"></el-option>
21       </el-select>
22     </el-form-item>
23     <el-form-item label="证件号码" prop="number">
24       <el-input v-model.number="form.number"></el-input>
25     </el-form-item>
26     <el-form-item label="证件上传">
27       <el-upload

```


```

28     class="avatar-uploader"
29     :action="存储文件的服务器地址"
30     :show-file-list="false"
31     :on-success="handleAvatarSuccess"
32     :before-upload="beforeAvatarUpload"
33   >
34     
35     <i v-else class="el-icon-plus avatar-uploader-icon"></i>
36   </el-upload>
37 </el-form-item>
38 <el-form-item>
39   <el-button
40     class="buttonC"
41     type="primary"
42     @click="submitForm('form')"
43   >提交认证</el-button>
44 </el-form-item>
45 </el-form>
46 </el-card>

```

这里就是简单的表单应用，让用户能够填写和提交相关的信息即可。唯一需要注意的就是 v-if 的应用，我们通过对 status 的值判断最终呈现申请信息内容。其实这个 status 就是我在上面提到的每一个菜单的 index 值。

紧接着我们看看如何呈现页面，具体的实现代码是后面这样。

 复制代码

```

1 handleSelect(index, indexPath) {
2   console.log(indexPath); //['1', '1-2']
3   console.log(index); //1-2
4   this.status = index;
5 },

```

可以看到，这里我们用到了 @select 方法 handleSelect，它主要包含两个参数——key 和 keypath，分别代表选中菜单的 index 和选中菜单的 indexPath。在点击触发 handleSelect 方法的时候，就会给 status 值进行赋值，这样展示的页面内容就可以根据 status 来判断了。

在 status 值改变之后，对应的页面 card 内容也会随之变化，这样我们就实现了申请中心的功能。菜单中的“直播间”主要是用来配置直播相关的信息，整体的实现方式与申请中心一致，

区别就是申请中心是以主播信息为核心，而直播间配置更加关注的则是给更多用户展示每一场直播的相关信息。

到这里，信息配置模块的页面开发告一段落。直播前的准备已经就绪，我们下一步就来搞定直播里的视频播放器，这是我们直播中最关键的功能。

认识一下 Video-Player

平台的直播应用方面，我们选取 Video-Player 充当视频播放器。

选择 Video-Player 有两个原因：第一个就是除了前面学过的 DPlayer 播放器以外，我想带你多接触一些不同的播放器，丰富一下你个人的技术库。第二是因为相比 DPlayer，Video-Player 更适合做平台的直播。为什么说它更适合呢？我们接着往下看。

Video-Player 是由俄罗斯的开发团队 2007 年开始开发的。它最初是在俄罗斯的一个社交网站上开始流行的，之后在 2010 年被翻译成英语并在全球范围内推广。在经历了一些版本的更新和改进之后，Video-Player 在 2014 年被开源，并成为了广受欢迎的知名开源项目。

接下来，我们从应用场景和相关技术支持的角度比较一下 DPlayer 与 video-player，你就明白我们为什么选择后者了。

Video-Player 适合在各种平台上播放多种格式的视频，包括网络视频平台、本地播放器、移动设备等。它支持的编解码器和流媒体格式非常丰富，可以轻松的支持各种主流的流媒体服务，如 YouTube、Facebook Live 等。

shikey.com 转载分享

另外，Video-Player 采用了先进的 H.265 视频编码技术，可以提供更加清晰、流畅和低延迟的视频播放体验。它还支持多种音频格式，可以为用户带来更加逼真的音频和视频体验，这个是它非常大的一个优势。

shikey.com 转载分享

Dplayer 则更加注重原始素材的加工和处理，提供的素材管理和调度工具更为丰富，支持多种视频编辑和转换工具，用户制作原生视频内容会更加方便。此外，Dplayer 还支持实时流媒体服务，可以实时转码并推流到不同的平台上。

综合来看，如果你需要在网络直播平台上使用视频播放器，Video-Player 更适合。但如果你只需要在本地或移动设备上播放视频，Dplayer 更适合。

如果你有兴趣，课后可以看看 Video-Player 的 [官网链接](#)，进入官网你会发现是 Video.js 官网，不过不用疑惑，Video.js 是一个流行的 JavaScript 视频播放器库，而 Video-Player 是 Video.js 的一个分支。两者在功能和支持的编解码器方面有一定的重叠，但 Video-Player 对流行视频平台和格式的原生支持更多一些，自定义选项也更加丰富。

播放器 Video-Player 实战

全面了解了 Video-Player 之后，我们进入实战环节。在 Vue 中使用第三方应用的流程，相信你已经非常熟练了，下面请跟上我的节奏，先从安装环节入手。


我们先要安装 Video-Player，在对应的项目路径下执行下面的命令即可。

```
1 cnpm install vue-video-player --save
```

 复制代码

然后我们需要在 main.js 中引入 VideoPlayer。

```
1 import VideoPlayer from 'vue-video-player'
2 Vue.use(VideoPlayer)
```


 复制代码

shikey.com转载分享

与此同时，我们还要引入播放器需要的 CSS 内容，代码是后面这样。

shikey.com转载分享

```
1 require('video.js/dist/video-js.css')
2 require('vue-video-player/src/custom-theme.css')
```

 复制代码

因为我们的直播流采用的是 m3u8 文件，所以我们下一步需要导入 HLS。HLS 是什么呢？HLS 是一种流媒体传输协议，它允许流媒体服务器向客户端推送实时流媒体流，Video.js 可以通过 JavaScript 来处理 HLS 流媒体传输。

Video.js 提供了一个简单易用的 API，可以轻松的处理 HLS 流媒体传输，并支持多种流行的视频平台和格式。


接下来我们就在项目中安装一下 HLS，执行命令我写在了后面。

```
1 cnpm install videojs-contrib-hls --save
```

 复制代码

同样的道理，我们依旧需要在 main.js 文件中引入，具体代码如下。

```
1 在main.js中引入
2 import videojsC from 'videojs-contrib-hls'
3 Vue.use(videojsC)
```


 复制代码

这样我们就完成了整个安装的过程。

那么如何在页面中使用 Video-Player 呢？第一步，我们需要在直播的页面中引入 Video-Player，这样在页面中才可以使用它。

shikey.com转载分享


```
1 import { videoPlayer } from "vue-video-player";
```

 复制代码

shikey.com转载分享

第二步就是实现 Video-Player 组件。你可以对照后面的具体代码听我讲解。

```
1 <video-player
2   class="video-player vjs-custom-skin"
```

 复制代码



```

3   ref="videoPlayer"
4   :playsinline="false"
5   :options="playerOptions"
6   @play="onPlayerPlay($event)"
7   @pause="onPlayerPause($event)"
8 ></video-player>

```

对照代码可以看到，这里面放了两种方法，分别是 **play() 播放方法**和 **pause() 暂停方法**。当然 video-player API 有很多方法，这个我稍后再给你梳理。

接下来我们需要重点关注的是 **options 的参数 playerOptions**。对照后面的代码和详细的代码注释，你很容易就能明白每个参数的含义和作用。


 复制代码

```

1  playerOptions: {
2    playbackRates: [0.7, 1.0, 1.5, 2.0], //播放速度
3    autoplay: false, //如果true,浏览器准备好时开始回放。
4    muted: false, // 默认情况下将会消除任何音频。
5    loop: false, // 导致视频一结束就重新开始。
6    preload: "auto", // 建议浏览器在<video>加载元素后是否应该开始下载视频数据。auto浏览器选择
7    language: "zh-CN",
8    aspectRatio: "16:9", // 将播放器置于流畅模式，并在计算播放器的动态大小时使用该值。值应该代
9    fluid: true, // 当true时，播放器将按比例缩放以适应其容器。
10   sources: [
11     {
12       type: "application/x-mpegURL",
13       src: "你的直播m3u8地址（必填），当然在开发工程中，这里需要设置为动态值"
14     }
15   ],
16   poster: "poster.jpg", //你的封面地址
17   width: document.documentElement.clientWidth,
18   notSupportedMessage: "此视频暂无法播放，请稍后再试", //允许覆盖Video.js无法播放媒体源时！
19   controlBar: {
20     timeDivider: true, //当前时间和持续时间的分隔符
21     durationDisplay: true, //表示当前时间和持续时间的显示格式
22     remainingTimeDisplay: true, //表示当前时间还剩余的时间的显示格式
23     fullscreenToggle: true //全屏按钮
24   }
25 }


```

前面我们提到的两个方法——播放和暂停的方法。我们在方法里可以放置要执行的相关代码逻辑，你可以结合后面的例子体会一下用法。

 复制代码

```
1  methods: {
2    onPlayerPlay(player) {
3      console.log("play");
4      //TODO:开始播放需要执行的相关业务操作
5      this.$refs.videoPlayer.player.play();
6    },
7    onPlayerPause(player) {
8      //TODO: 暂停需要执行的相关业务操作
9      console.log("pause");
10   }
11 }
```

我还帮你整理了 Video-Player 相关操作的常用方法，在之后的开发中你可以直接应用这些方法，解决你在播放器功能上的需求。

 复制代码

```
1  //整体的写法@后为固定方法名 =后自定义，和播放、暂停的写法一致
2  @play="onPlayerPlay($event)"
3  @pause="onPlayerPause($event)"
4  @ended="onPlayerEnded($event)"
5  @waiting="onPlayerWaiting($event)"
6  @playing="onPlayerPlaying($event)"
7  @loadeddata="onPlayerLoadeddata($event)"
8  @timeupdate="onPlayerTimeupdate($event)"
9
10 //具体的用法和含义写在下面
11
12 // 播放回调
13 onPlayerPlay(player) {
14   console.log('player play!', player);
15 },
16
17 // 暂停回调
18 onPlayerPause(player) {
19   console.log('player pause!', player);
20 },
21
22 // 视频播完回调
23 onPlayerEnded($event) {
```

```
24 console.log(player)
25 },
26
27 // 当视频播放时出现暂停、停止或重新加载等状态变化时，调用该方法
28 onPlayerWaiting($event) {
29 console.log(player)
30 },
31
32 // 当用户开始播放视频时，调用该方法
33 onPlayerPlaying($event) {
34 console.log(player)
35 },
36
37 // 当播放器在当前播放位置下载数据时触发
38 onPlayerLoadeddata($event) {
39 console.log(player)
40 },
41
42 // 当视频播放时出现时间更新时，调用该方法
43 onPlayerTimeupdate($event) {
44 console.log(player)
45 }
```

完成前面的步骤，我们还要通过后端接口请求获取到 sources 中的 src 地址，这样就能使用 Video-Player 在平台里实现直播的功能了。因为整体实现过程的步骤比较多，建议课后你再自己把整个流程回顾一下。

总结

不知不觉又到了课程尾声，我们一起来回顾总结一下。

shikey.com转载分享

这节课我们主要实现了用户在平台的直播功能。在此之前我们要先设计好用户的直播申请功能，这一部分，你需要重点关注两点：一是**新组件 <el-menu> 的应用**，二是如何通过 v-show 的指令控制实现菜单切换的功能。

shikey.com转载分享

在直播功能的开发实践环节，我们使用了 Video-Player 播放器，它有强大的综合功能，我们发现 Video-Player 更适合应用在平台类的直播。当然如果你只需要在本地或移动设备上播放视频，Dplayer 更适合，你可以灵活选择。

Video-Player 整个安装配置的过程是这节课的重点，也是我建议你多花心思的部分，只要你细心，按部就班跟着课程讲解来实现，一定可以拿下。完成安装之后，你就可以参照我提供的实践代码，自己先把播放器应用起来，通过现成的第三方直播链接测试一下效果。当然，在学完课程的后端直播模块之后，你就可以打通前后端，在平台内实现自己的直播应用了。

思考题

直播中心应用中我们一起实现了申请中心，你可以尝试模拟实现一下直播间信息的配置。请你思考一下需要用户提交哪些信息呢？页面又该怎样实现呢？

期待你在留言区和我交流互动。如果这节课的内容让你感觉实用有趣的话，别忘了推荐给你的伙伴，让我们一起学习进步。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (1)



peter

2023-05-29 来自北京

Q1：菜单“直播间”，观众和主播点进去以后的界面是一样的吗？

Q2：VideoPlayer能用于安卓吗？

作者回复：1、直播的主播界面是不一样的 和观众看的，直播的主播看的是要展示给观众什么内容，例如我们投屏操作等等。

2、VideoPlayer可以用于安卓，这个是没有问题的。

 shikey.com 转载分享 

shikey.com 转载分享