

04 | 如何在前端调用语音合成和视觉模型

月影 · 跟月影学前端智能体开发



你好，我是月影。

我们知道，AI 大模型应用的输入、输出通常是多模态的，不论大模型本身是否支持不同格式的输入输出，在业务上，我们都能够通过调用不同的服务整合来做到这一点。

在这节课里，我们就来体验一下火山引擎和月之暗面的服务，通过具体实践了解如何使用语音合成和视觉模型。

使用火山引擎语音合成

首先我们注册火山引擎账号，然后进入控制台，搜索并选择“语音技术”。



进入语音技术操作面板后，点击右侧创建应用按钮创建应用。



选择服务勾选“大模型语音合成”和“语音合成”。

创建应用

* 应用名称

TTS_Demo

* 应用简介

大模型语音合成测试

* 选择服务

语音合成大模型

☒ 大模型语音合成

声音复刻大模型

☐ 字符版 ☐ 并发版 (暂不支持试用)

流式语音识别大模型

☐ 小时版 ☐ 并发版

录音文件识别大模型

☐ 大模型录音文件识别

语音合成

☒ 语音合成

精品长文本语音合成

☐ 普通版 ☐ 情感预测版

创建完毕后，左侧菜单切换到“API 服务中心 > 音频生成大模型 > 语音合成大模型”，右侧可以看到“服务详情”、“音色详情”、“服务接口认证信息”等内容。注意我们要将服务接口认证信息里的 APP ID 和 Access Token 保存下来，后续调用需要用到。

广西远舟	--	趣味方言	中文	zh_male_guangxiyuanzhou_moon_big tts	-
妹坨洁儿	--	趣味方言	中文	zh_female_meituojieer_moon_big tts	-
豫州子轩	--	趣味方言	中文	zh_male_yuzhouzixuan_moon_big tts	-
邻家女孩	--	通用场景	中文	zh_female_linjianvai_moon_big tts	-
高冷御姐	--	角色扮演	中文	zh_female_gaolengyujie_moon_big tts	-
渊博小叔	--	通用场景	中文	zh_male_yuanboxiaoshu_moon_big tts	-
阳光青年	--	通用场景	中文	zh_male_yangguangqingnian_moon_big tts	-
傲娇霸总	--	角色扮演	中文	zh_male_aojiaobazong_moon_big tts	-
京腔侃爷/Harmony	--	趣味方言	中/英	zh_male_jingqiangkanye_moon_big tts	-
爽快思思/Skye	--	通用场景	中/英	zh_female_shuangkuaisisi_moon_big tts	-

共 122 条 < 1 2 3 4 5 ... 13 > 10 条/页 前往 页

服务接口认证信息

APP ID

Access Token

Secret Key

5934290469 5***** *****

接入方式

API

SDK

[接入文档](#)[接入文档](#)

至此火山引擎注册和开通服务部分已经完成，接下来我们就可以创建项目了。

还是在 Trae 中创建一个 Vue 项目并添加 .env.local，配置 AppID、AccessToken 和 ClusterID，以备后续使用。

复制代码

```
1 VITE_APP_ID=5934290469
2 VITE_ACCESS_TOKEN=c-*****Ln4N
3 VITE_CLUSTER_ID=volcano_tts
```

接着我们修改 App.vue，实现具体功能和 UI：

复制代码

```
1 <script setup lang="ts">
2 import { ref } from 'vue';
3
4 const prompt = ref('您好，请问有什么可以帮您？');
5 const status = ref('ready');
```

```
6  const audioEl = ref<HTMLAudioElement>();
7
8  function createBlobURL(base64AudioData: string): string {
9      var byteArrays = [];
10     var byteCharacters = atob(base64AudioData);
11     for (var offset = 0; offset < byteCharacters.length; offset++) {
12         var byteArray = byteCharacters.charCodeAt(offset);
13         byteArrays.push(byteArray);
14     }
15
16     var blob = new Blob([new Uint8Array(byteArrays)], { type: 'audio/mp3' });
17
18     // 创建一个临时 URL 供音频播放
19     return URL.createObjectURL(blob);
20 }
21
22
23 const generateAudio = async () => {
24     const token = import.meta.env.VITE_ACCESS_TOKEN;
25     const appId = import.meta.env.VITE_APP_ID;
26     const clusterId = import.meta.env.VITE_CLUSTER_ID;
27     const voiceName = "zh_female_shuangkuaisisi_moon_big tts";
28
29     const endpoint = '/tts/api/v1/tts';
30     const headers = {
31         'Content-Type': 'application/json',
32         Authorization: `Bearer;${token}`,
33     };
34
35     const payload = {
36         app: {
37             appid: appId,
38             token,
39             cluster: clusterId,
40         },
41         user: {
42             uid: 'bearbobo',
43         },
44         audio: {
45             voice_type: voiceName,
46             encoding: 'ogg_opus',
47             compression_rate: 1,
48             rate: 24000,
49             speed_ratio: 1.0,
50             volume_ratio: 1.0,
51             pitch_ratio: 1.0,
52             emotion: 'happy',
53         },
54         request: {
```

```

55     reqid: Math.random().toString(36).substring(7),
56     text: prompt.value,
57     text_type: 'plain',
58     operation: 'query',
59     silence_duration: '125',
60     with_frontend: '1',
61     frontend_type: 'unitTson',
62     pure_english_opt: '1',
63   },
64 };
65
66 status.value = 'generating';
67 const res = await fetch(endpoint, {
68   method: 'POST',
69   headers,
70   body: JSON.stringify(payload),
71 });
72 const data = await res.json();
73
74 if (!data.data) {
75   throw new Error(JSON.stringify(data));
76 }
77 const url = createBlobURL(data.data);
78
79 audioEl.value && (audioEl.value.src = url)
80 audioEl.value?.play();
81 status.value = 'done';
82 };
83 </script>
84
85 <template>
86   <div class="container">
87     <div>
88       <label>Prompt </label>
89       <button @click="generateAudio">Generate & Play</button>
90       <textarea class="input" type="text" v-model="prompt" />
91     </div>
92     <div class="output">
93       <div>>> {{ status }}</div>
94       <audio ref="audioEl"></audio>
95     </div>
96   </div>
97 </template>
98
99 <style scoped>
100 .input {
101   width: 100%;
102   height: 2rem;
103   font-size: 1rem;

```


```

104 padding: 0.5rem;
105 border: 1px solid #ccc;
106 border-radius: 0.5rem;
107 }
108 .progress {
109 width: 100%;
110 height: 0.1rem;
111 margin: .4rem 0;
112 background: #ccc;
113 }
114 .progress > div {
115 background: #c00;
116 height: 100%;
117 }
118 .container {
119 display: flex;
120 flex-direction: column;
121 align-items: start;
122 justify-content: start;
123 height: 100vh;
124 }
125 .output {
126 display: flex;
127 flex-direction: column;
128 align-items: center;
129 justify-content: center;
130 }
131 .output > div {
132 width: 100%;
133 max-width: 600px;
134 }
135 </style>

```

我们看一下主要的实现代码。

首先，我们读取配置项，并设置语音的音色：

 复制代码


```

1 const token = import.meta.env.VITE_ACCESS_TOKEN;
2 const appId = import.meta.env.VITE_APP_ID;
3 const clusterId = import.meta.env.VITE_CLUSTER_ID;
4 const voiceName = "zh_female_shuangkuaisisi_moon_bigtts";

```


然后我们设置请求的 URL，注意直接调用火山引擎服务会有跨域问题，所以我们仍然要先修改 vite.config.ts 配置，添加 server 代理：

vite.config.ts

 复制代码

```
1  server: {
2    allowedHosts: true,
3    proxy: {
4      '/tts': {
5        target: 'https://openspeech.bytedance.com',
6        changeOrigin: true,
7        rewrite: path => path.replace(/^\/tts/, ''),
8      }
9    },
10  },
```

好了，接着继续，我们设置 headers 和 payload，这个和我们前几节课内容差不多，相信大家已经很熟悉了。

 复制代码

```
1  const endpoint = '/tts/api/v1/tts';
2  const headers = {
3    'Content-Type': 'application/json',
4    Authorization: `Bearer;${token}`,
5  };
6
7  const payload = {
8    app: {
9      appid: appId,
10     token,
11     cluster: clusterId,
12   },
13   user: {
14     uid: 'bearbobo',
15   },
16   audio: {
17     voice_type: voiceName,
18     encoding: 'ogg_opus',
19     compression_rate: 1,
20     rate: 24000,
```



```
21     speed_ratio: 1.0,
22     volume_ratio: 1.0,
23     pitch_ratio: 1.0,
24     emotion: 'happy',
25 },
26 request: {
27     reqid: Math.random().toString(36).substring(7),
28     text: prompt.value,
29     text_type: 'plain',
30     operation: 'query',
31     silence_duration: '125',
32     with_frontend: '1',
33     frontend_type: 'unitTson',
34     pure_english_opt: '1',
35 },
36 };
```

注意，这里有一些必要的语音参数：

voice_type：音色，火山引擎支持数十种不同的音色，我们的例子用的是“爽快思思”，你也可以换成其他的音色。注意，当你希望转换的内容包含非中文内容时，通常应当选择支持多语言的音色或者当前语种的音色，具体你可以根据需求在音色列表中选择。

encoding：语音格式，这里我们可以选择使用 mp3、ogg 等多种格式，注意如果你的应用希望兼容多个终端，应当考虑适配最广的模式，例如有些设备对于 ogg 格式的文件无法播放，那么我们最好是生成 mp3 格式。

compression_rate：控制音频压缩率，通常这个值影响音频文件的大小和质量。

rate：表示音频的采样率（samples per second），通常影响音频的质量和清晰度。24000 是音频的常用采样率，即每秒钟的采样次数。24000 Hz（赫兹）是一个常见的采样率，它的音质较好。


speed_ratio：表示语速，1.0 表示正常语速。

volume_ratio：表示音量，1.0 是正常音量。

pitch_ratio：表示音调，音调的调整影响语音的高低，1.0 为正常音调。

emotion：表示语音的情感表达，大模型根据它来调整语音的情感色彩，happy 表示欢快，语音可能会更轻松愉快，语气上会有更多的高低起伏。

最后，我们发起请求，拿到 JSON 格式的返回数据，其中 data 字段内是音频的 Base64 编码，我们可以直接使用。

 复制代码

```
1  status.value = 'generating';
2  const res = await fetch(endpoint, {
3    method: 'POST',
4    headers,
5    body: JSON.stringify(payload),
6  });
7  const data = await res.json();
8
9  if (!data.data) {
10    throw new Error(JSON.stringify(data));
11  }
12
13  const url = createBlobURL(data.data);
14
15  audioEl.value && (audioEl.value.src = url)
16  audioEl.value?.play();
17  status.value = 'done';
```

这里我们通过 createBlobURL 函数将 Base64 编码的数据转换成二进制对象，然后生成 URL，将 URL 设置为 audio 标签的 src，再执行 play 方法，就可以直接将语音播放出来了。

 复制代码

```
1  function createBlobURL(base64AudioData: string): string {
2    var byteArrays = [];
3    var byteCharacters = atob(base64AudioData);
4    for (var offset = 0; offset < byteCharacters.length; offset++) {
5      var byteArray = byteCharacters.charCodeAt(offset);
6      byteArrays.push(byteArray);
7    }
8
9    var blob = new Blob([new Uint8Array(byteArrays)], { type: 'audio/mp3' });
10
11    // 创建一个临时 URL 供音频播放
12    return URL.createObjectURL(blob);
13  }
```

到此为止，我们就实现了一个最简单的文字合成语音功能。

在一般情况下，我们的产品功能不会直接将任意文字转语音，通常是将文本模型生成的回答内容转为语音，所以我们会将语音合成结合文本模型使用，组成特定的智能体来实现应用。关于这部分内容，在后续的课程中我们有机会进一步深入探讨。在这一节课里，我们先了解如何通过 API 将文字合成语音就可以了。

我们可以听一下将上面这段文字转语音的效果。

使用 Kimi 视觉模型

在有些 AI 应用中，我们不仅仅让用户提供文字的信息，还可以让用户提供图像信息。这时候，视觉（Vision）模型就起到了分析和理解图片内容的重要作用。

国内也有一些平台支持了视觉模型，这里我们以月之暗面的 Kimi 为例，来学习如何使用视觉模型。


首先，我们在 <https://platform.moonshot.cn/> 完成注册，进入控制台。Kimi 和我们之前了解的 Deepseek 平台差不多，我们先点击控制台的左侧菜单，选择 API Key 管理，新建一个 API Key。

创建了 API Key 之后，我们就可以通过 API 调用 Kimi 大模型了，其中 kimi 的视觉模型包括 moonshot-v1-8k-vision-preview/moonshot-v1-32k-vision-preview/moonshot-v1-128k-vision-preview 等，它们能力基本一样，区别是 token 的数量限制。在这里我们使用 moonshot-v1-8k-vision-preview。


首先依然是使用 Trae 创建一个新的项目 Kimi Vision Demo。

添加.env.local：

```
1 VITE_API_KEY=sk-qi2o*****xbp4
```

 复制代码

修改 App.vue 文件为如下内容：

 复制代码


```
1 <script setup lang="ts">
2 import { ref, computed } from 'vue';
3
4 const content = ref('');
5 const imgBase64Data = ref('');
6 const isValid = computed(() => imgBase64Data.value !== '');
7
8 const updateBase64Data = async (e: Event) => {
9   imgBase64Data.value = '';
10   const file = (e.target as HTMLInputElement).files?.[0];
11   if (!file) {
12     return;
13   }
14
15   const reader = new FileReader();
16   reader.readAsDataURL(file);
17   reader.onload = () => {
18     imgBase64Data.value = reader.result as string;
19   };
20 };
21
22 const update = async () => {
23   if(!imgBase64Data.value) {
24     return;
25   }
26
27   const endpoint = 'https://api.moonshot.cn/v1/chat/completions';
28   const headers = {
29     'Content-Type': 'application/json',
30     Authorization: `Bearer ${import.meta.env.VITE_API_KEY}`
31   };
32
33   content.value = '思考中...'
34   const response = await fetch(endpoint, {
35     method: 'POST',
36     headers: headers,
37     body: JSON.stringify({
38       model: 'moonshot-v1-8k-vision-preview',
39       messages: [
40         { role: 'user',
41           content: [{
42             type: "image_url",
43             image_url: {
44               "url": imgBase64Data.value,
```

```
45         },
46         }, {
47             type: "text",
48             text: "请描述图片的内容。",
49         }]
50     }
51 ],
52     stream: false,
53 })
54 });
55
56 const data = await response.json();
57 content.value = data.choices[0].message.content;
58 };
59 </script>
60
61 <template>
62     <div class="container">
63         <div>
64             <label>文件: </label>
65             <input class="input" type="file"
66                 accept=".jpg, .jpeg, .png, .gif"
67                 @change="updateBase64Data"/>
68             <button @click="update" :disabled="!isValid">提交</button>
69         </div>
70         <div class="output">
71             <div class="preview">
72                 
73             </div>
74             <div>{{ content }}</div>
75         </div>
76     </div>
77 </template>
78
79 <style scoped>
80 .container {
81     display: flex;
82     flex-direction: column;
83     align-items: start;
84     justify-content: start;
85     height: 100vh;
86     font-size: .85rem;
87 }
88 .input {
89     width: 200px;
90 }
91 .output {
92     margin-top: 10px;
93     min-height: 300px;
```

```
94     width: 100%;
95     text-align: left;
96 }
97 .preview img {
98     max-width: 100%;
99 }
100 button {
101     padding: 0 10px;
102     margin-left: 6px;
103 }
104 </style>
```

我们看一下代码的关键部分。其实它和调用前面文本模型的区别很小，只是输入由文本内容换成了图片内容。


注意这里我们使用了 FileReader，直接在浏览器端获取图片的 Base64 数据，然后将它传给 Kimi 的多模态大模型进行处理。

 复制代码

```
1  const updateBase64Data = async (e: Event) => {
2      imgBase64Data.value = '';
3      const file = (e.target as HTMLInputElement).files?.[0];
4      if (!file) {
5          return;
6      }
7
8      const reader = new FileReader();
9      reader.readAsDataURL(file);
10     reader.onload = () => {
11         imgBase64Data.value = reader.result as string;
12     };
13 };
```

reader.readAsDataURL 会将文件的文本或者二进制内容自动解析为 Base64 编码的字符串，并带上格式头，也就是说，对于 png 图片来说，它会生成以 `data:image/png;base64`，开头的字符串，我们可以将这内容直接传给 Kimi 的多模态大模型服务。

在传递参数给大模型时，Kimi 多模态的大模型支持 type 为 image_url 的内容，当 type 设置为 image_url 时，对应的 image_url 字段可以支持图片的 Base64 数据，因此我们只需要按照下面这个数据结构将数据发给 Kimi 的多模态大模型进行处理就可以了。整个代码也非常的简单，和之前的文本大模型的调用，除了参数格式的区别外，几乎没有其他区别。

 复制代码

```
1  const response = await fetch(endpoint, {
2    method: 'POST',
3    headers: headers,
4    body: JSON.stringify({
5      model: 'moonshot-v1-8k-vision-preview',
6      messages: [
7        { role: 'user',
8          content: [{
9            type: "image_url",
10             image_url: {
11               "url": imgBase64Data.value,
12             },
13           }, {
14             type: "text",
15             text: "请描述图片的内容。",
16           }
17         ]
18       },
19       stream: false,
20     })
21   });
```

这样，我们就实现了视觉识别，让我们的应用具有了接受和分析图片内容的能力。



要点总结

在这一节课，我们通过实战，继续学习了其他类型的大模型能力，包括语音合成和视觉模型。

语音合成和视觉模型引入，对于我们实现多模态的 AI 应用非常有帮助。

语音合成的作用是将一段文本文字，通过大模型转换为带有类似真人感情的语音，这样我们就能将语音内容在前端播放出来。

视觉大模型的能力是接受图片输入，然后按照用户的要求，分析图片中的内容，将其中的内容用文字描述出来，或者进行其他处理。

课后练习

1. 火山引擎的语音模型可以支持数十种不同的音色，修改上面的例子，给界面增加音色选择功能，对同样的文本合成不同音色的语音效果，同时配合修改 emotion 情感参数，体验一下它们的区别。
2. 在使用语音合成模型的实践中，我们采用了将 Base64 字符串转换为 Blob（二进制对象）的方式来播放语音，我们为什么这么做，这么做有什么好处？有没有其他可行的办法？请大家思考、自行搜索资料或者询问 AI 来深入学习。
3. 在前面视觉大模型的例子里，我们只输入图片，然后固定要求 AI 描述图片内容。实际上，我们还可以通过指令控制 AI 对图片做其他的处理，比如对物品进行分类，说出主要物品的英文单词、看图说话撰写一篇作文等等。请你修改前面的例子，增加一个用户要求的输入框，让用户可以输入不同的要求，看 AI 是否能够根据用户的要求完成对图片的解读。

欢迎把你的学习心得体会分享到评论区。

AI智能总结

1. 火山引擎注册和开通服务后，可以使用语音合成功能，需要设置请求的URL并处理跨域问题。
2. 通过处理返回的JSON格式数据，可以将语音合成的音频进行播放。
3. 语音合成功能通常与文本模型结合使用，以实现特定的智能体应用。
4. 视觉模型在AI应用中起到了分析和理解图片内容的重要作用。
5. Kimi视觉模型支持多种能力，包括moonshot-v1-8k-vision-preview/moonshot-v1-32k-vision-preview/moonshot-v1-128k-vision-preview等。
6. 通过实战学习了语音合成和视觉模型的能力，对实现多模态的AI应用非常有帮助。
7. 语音合成的作用是将文本文字转换为带有真人感情的语音，可在前端播放出来。
8. 视觉大模型能够接受图片输入，分析图片内容并用文字描述出来，或进行其他处理。
9. 课后练习包括修改界面增加音色选择功能，探索语音合成模型的播放方式，以及修改视觉大模型的例子，增加用户要求的输入框。

© 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。



咕叽咕叽

2025-04-18 来自新加坡

将 Base64 字符串转换为 Blob（二进制对象）的好处：

1. Base64 字符串增加了数据大小，Blob（二进制对象）减少内存占用
2. Base64 字符串嵌入到HTML中，大的字符串影响页面渲染性能
3. audio播放base64字符串，需要先解析字符串，再解码为二进制，再播放。Blob(二进制对象)可直接播放

作者回复: 赞👍



1