



# Vue3仿YesPlayMusic

## 项目介绍

使用Vue3全家桶仿YesPlayMusic

---

## 数据来源

样式参考[YesPlayMusic](#)

API来源[NeteaseCloudMusicApi](#)

播放器 [APlayer](#)

---

## 项目阶段

### 2023/3/9 项目启动

技术栈：vue3+vite+scss+axios+pinia+vue-router

git地址：<https://github.com/Younglina/vue3-yesplay>

---

### 2023/3/10 开始首页开发

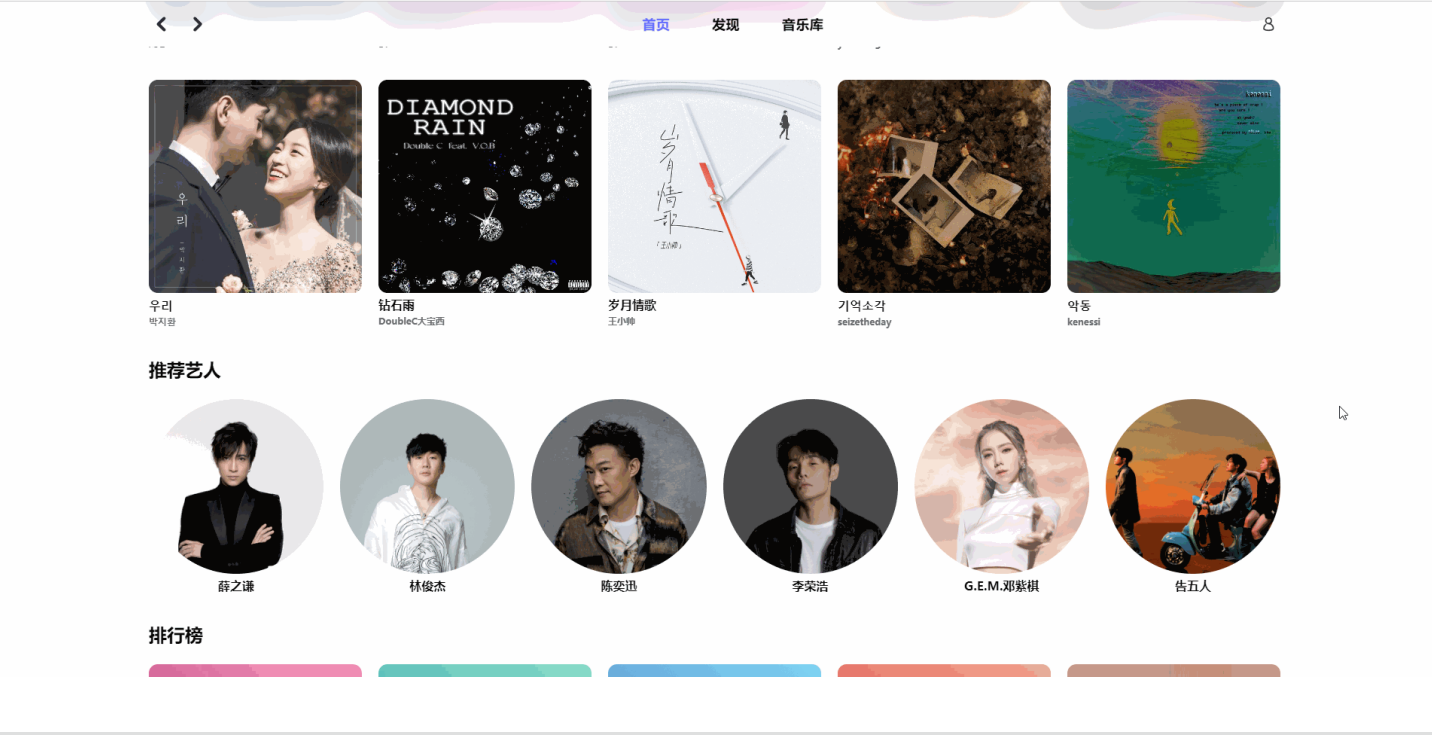
需拆分为上中下三个模块

上：主要是一些页面跳转，包括前进后退；首页、发现、音乐库；我的；  
中：包括推荐歌单、每日推荐、推荐艺人、新专速递、排行榜几个卡片列表  
下：音乐播放相关、歌词页面控制

已完成内容

- 头部静态模块
- 推荐歌单、新专速递、推荐艺人、排行榜接口静态数据展示

1.0



2023/3/11 开始发现页开发

包含歌单分类模块、歌单列表模块；

已完成内容

- 歌单分类模块
- 静态歌单列表模块

1.0

localhost:5173/#/explore?category=推荐歌单&apiType=personalized


<>

首页发现音乐库


8

发现


全部推荐歌单精品歌单官方排行榜流行影视原声华语怀旧摇滚...




写作业必备歌单 ✨【百听不厌】




国语古风中国风神曲，细腻地抚爱你的耳膜




用音乐保持你每天的嘴角上扬





欧美万评优质女声，萦绕耳畔忆于心间

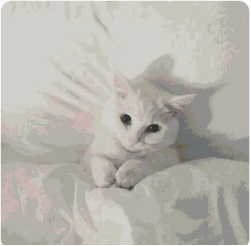



『曲作精选』细数古风圈原创作曲人 ●











2023/3/14 开始详情页开发

已完成内容


- 静态页开发 3/14~3/17

1.0

<>

首页发现音乐库

8



【民谣盛宴】100首必听欧美民谣


Playlist by pure日月  
最后更新于2017年11月03日 · 100首歌

民谣是什么？民谣是一壶酒，我有一壶酒，足以慰风尘。民谣是什么？民谣是一朵花，你若是盛开，清风自会来。喜欢民谣，民谣像是一位说书人，不紧不慢的讲述曾经动人的故事。现在，就让我们来倾听欧美的民谣故事吧！


▶ 播放

♡


...




Going Home  
Sophie Zelmani




Sutter's Mill  
Dan Fogelberg



Paradise  
Bruce Springsteen



Sleep, Baby, Sleep  
BROODS



Five Hundred Miles (五百英里)

Decade of dreams 1995-2005

High Country Snows

The Rising

BROODS EP

Inside Llewyn Davis: Original Soundtrack Recording

4:10

6:33

5:34

3:00

3:27

# 2023/3/20 开始专辑页开发

抽离了列表组件

## 已完成内容

- 静态页开发 3/19~3/20

### 1.0

<

>

首页发现音乐库8

摇滚北京1

Album by 群星

1993 · 11首歌

▶ 播放

♡

...

▶ 烽火扬州路4:40♡

2 自己的天堂5:46

3 安魂进行曲6:50

4 不要匆忙5:10

5 请走人行道4:59

6 臆潜5:32

7 新世界4:09

8 希望之光6:06

9 祖先的阴影5:36

10 玩世不恭3:26

11 这一刻我是真心的4:59

发行于 1993年03月01日

More by 群星

2023热歌合集

Album · 2023

治愈旋律 极致欧美金曲

Album · 2023

律动音乐盒 欧美童歌

Album · 2023

青少年耳机里的欧美歌单

Album · 2023

寂寞夜晚

Album · 2023

# 2023/3/21 开始歌手页开发

## 已完成内容

- 静态页开发 3/21

### 1.0



## 张学友

艺人

1719首歌 · 129张专辑 · 358张专辑

张学友 (Jacky Cheung)，1961年7月10日出生于中国香港，祖籍天津市，中国香港男歌手、演员、作曲家，毕业于香港英文书院。1984年因获得首届香港十八区业余歌唱大赛冠军而出道。1985年发行个人首张专辑《Smile》。1986年凭借歌曲《遥远的她》在香港乐坛获得关注。1993年发行的专辑《吻别》打破华语唱片在台湾的销量纪录。1995年起连续两年获得世界音乐大奖全球销量最高华人歌手奖。1997年参与策划并主演音乐剧《月亮湾》，1998年凭借专辑《想和你去吹吹风》获得第3届台湾金曲奖最佳...

▶ 播放



### 热门歌曲

如果这都不算爱  
李友 热烦恼歌  
在你身边她来听我的演唱会  
友情歌又十年  
又十年慢慢  
忘记你我做不到等你等到我心痛  
等你等到我心痛 张学友精选遥远的她  
遥远的她 Amour饿狼传说  
饿狼传说情书  
友情歌吻别  
吻别只想一生跟你走  
一生跟你走 年度代表作品辑还是觉得你最好  
等你等到我心痛 张学友精选

### 专辑



## 2023/3/23 开始播放器开发

使用APlayer

### 已完成内容

- 覆盖样式，只展示歌词
- 监听歌词点击可跳转

### 1.0

不能在一起游来游去  
能不能让你清醒  
爱是快乐的事情  
我只有真心而已  
世界末日我都不会离去  
需要你 我是一只鱼  
水里的空气  
是你小心眼和坏脾气  
没有你 像离开水的鱼  
快要活不下去  
不能在一起游来游去  
我是一只站在岸上的鱼  
如何能忘记曾经活在海里  
曾经我活在你的生命  
哦 耶 需要你 我是一只鱼  
水里的空气

### 1.1 新增功能

- 上一首、下一首、播放、暂停控制
- 播放事件跟随显示

- 滚动条手动滑动播放跟随变化



## 2023/3/23 开始歌词页开发

### 已完成内容

- 覆盖APlayer样式展示歌词
- 监听APlayer timeupdate事件实现歌词滚动
- 点击歌词歌曲跟着跳转

1.0

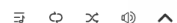


## 2023/4/1 开始底部栏开发

### 已完成内容

- 上一首、下一首、播放、暂停控制
- 显示歌词切换控制

1.0



## 开发点记录

2023/3/9

### 配置自动导入、svg组件、全局sass

自动导入：<https://github.com/antfu/unplugin-vue-components>

svg组件：<https://github.com/vbenjs/vite-plugin-svg-icons>

### 安装插件

```
1 // 自动导入
2 pnpm i unplugin-vue-components/vite -D
3
```



```
4 // svg组件
5 pnpm i vite-plugin-svg-icons -D
6
7 // sass
8 pnpm i sass
```

## 配置插件

修改 vite.config.js

```
1 import { defineConfig } from 'vite'
2 import vue from '@vitejs/plugin-vue'
3 import { join, resolve } from 'path'
4 // 自动导入vue3的api; 不用import组件直接使用
5 import Components from 'unplugin-vue-components/vite'
6 // svg组件
7 import { createSvgIconsPlugin } from 'vite-plugin-svg-icons'
8
9 export default defineConfig({
10   plugins: [
11     vue(),
12     Components({}),
13     createSvgIconsPlugin({
14       // 指定需要缓存的svg文件夹
15       iconDirs: [resolve(__dirname, 'src/assets/svg')],
16       // 指定symbolId格式
17       symbolId: 'icon-[name]',
18     })
19   ],
20   resolve: {
21     alias: {
22       '@': join(__dirname, '/src')
23     }
24   },
25   css: {
26     preprocessorOptions: {
27       scss: {
28         additionalData: `@import "@/mixin.scss";`,
29       },
30     },
31   },
32 })
33
```

## 使用插件



## 使用svg

创建 `src/components/SvgIcon.vue`

```
1 <template>
2   <svg aria-hidden="true" class="svg-icon">
3     <use :href="symbolId" :fill="color" />
4   </svg>
5 </template>
6
7 <script>
8 import { defineComponent, computed } from 'vue'
9
10 export default defineComponent({
11   name: 'SvgIcon',
12   props: {
13     prefix: {
14       type: String,
15       default: 'icon',
16     },
17     name: {
18       type: String,
19       required: true,
20     },
21     color: {
22       type: String,
23       default: '#333',
24     },
25   },
26   setup(props) {
27     const symbolId = computed(() => `#${props.prefix}-${props.name}`)
28     return { symbolId }
29   },
30 })
31 </script>
32 <style lang="scss" scoped>
33 .svg-icon{
34   width: 1.4em;
35   height: 1.4em;
36 }
37 </style>
```

在main.js中引入

```
1 import 'virtual:svg-icons-register'
```

```
2 import svgIcon from './components/SvgIcon.vue'
3
4 const app = createApp(App)
5 app.component('SvgIcon',svgIcon)
```

## 组件使用

```
1 // name就是svg文件的名字
2 <SvgIcon name="back"></SvgIcon>
```

## 使用全局sass

创建 `/src/mixin.scss`，保存一些常用的样式

```
1 @mixin flex-center{
2   display: flex;
3   align-items: center;
4   justify-content: center;
5 }
```

## 任意地方可使用

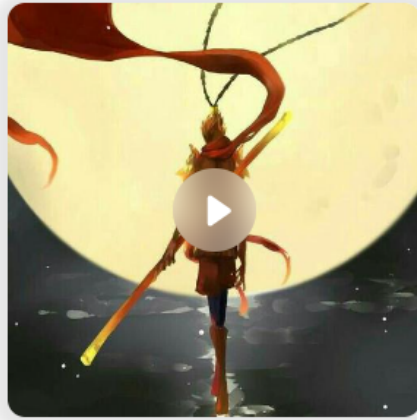
```
1 div{
2   @include flex-center
3 }
```

---

2023/3/10

## 新增卡片组件

整体卡片形式都以这个为准，hover显示播放按钮、box-shadow；



国语古风中国风神曲，细腻地抚摸你的耳膜

## 新增链接组件

整体链接跳转都以这个为准，通过类型和id跳转对应页面



国语古风中国风神曲，细腻地抚摸你的耳膜

2023/3/11

## 歌单分类模块



通过点击歌单分类，监听修改route的query，完成接口的调用和样式的修改

```
1 // immediate是为了一进入页面就执行一次接口调用和样式修改
2 watch(() => route.query, ({ category, apiType }) => {
```

```
3      ...
4    }, { immediate: true })
5
6    const changeCat = ({ name, apiType, icon }) => {
7      ...
8      router.push(`/explore?category=${name}${apiType ? '&apiType=' + apiType : ''}`)
9      ...
10 }
```

新增loading组件



2023/3/12

所有歌单分类

点击“更多”展示，下面的分类功能与上面一样

发现

全部 推荐歌单 精品歌单 官方 排行榜 流行 影视原声 华语 怀旧 摇滚 ...

语种	欧美	日语	韩语	粤语									
风格	民谣	电子	舞曲	说唱	轻音乐	爵士	乡村	R&B/Soul	古典	民族	英伦	金属	朋克
	蓝调	雷鬼	世界音乐	拉丁	New Age	古风	后摇	Bossa Nova					
场景	清晨	夜晚	学习	工作	午休	下午茶	地铁	驾车	运动	旅行	散步	酒吧	
情感	清新	浪漫	伤感	治愈	放松	孤独	感动	兴奋	快乐	安静	思念		
主题	ACG	儿童	校园	游戏	70后	80后	90后	网络歌曲	KTV	经典	翻唱	吉他	钢琴
	翻乐	榜单	00后										

2023/3/13

message组件

show success Message

show error Message

show warning Message

show info Message

### 用Transition组件包裹动画

`<Transition>` 是一个Vue3的内置组件，这意味着它在任意别的组件中都可以被使用，无需注册。它可以将进入和离开动画应用到通过默认插槽传递给它的元素或组件上。进入或离开可以由以下的条件之一触发：

- 由 `v-if` 所触发的切换
- 由 `v-show` 所触发的切换
- 由特殊元素 `<component>` 切换的动态组件

### CSS

```
1 .message {
2   position: fixed;
3   left: 50%;
4   transform: translateX(-50%);
5   display: flex;
6   align-items: center;
7   padding: 12px 18px;
8   border-radius: 6px;
9   transition: opacity 0.3s, transform 0.3s, top 0.3s;
10 }
11
12 .success{
13   color: #67c23a;
14   background-color: #f0f9eb;
15   border-color: #e1f3d8;
16 }
17 .info{
18   color: #909399;
19   background-color: #f4f4f5;
20   border-color: #e9e9eb;
```

```

21 }
22 .error{
23   color: #f56c6c;
24   background-color: #fef0f0;
25   border-color: #fde2e2;
26 }
27 .warning{
28   color: #e6a23c;
29   background-color: #fdf6ec;
30   border-color: #faecd8;
31 }
32
33 // v-enter-from: 进入动画的起始状态。
34 // v-leave-to 离开动画的结束状态。
35 // 意思是当组件显示时,
36 // 从opacity: 0;transform: translate(-50%, -100%);开始,
37 // 到opacity:1;top: 60px;transform: translateX(-50%);
38 // 是当组件隐藏时回到opacity: 0;transform: translate(-50%, -100%);
39 .v-enter-from,
40 .v-leave-to {
41   opacity: 0;
42   transform: translate(-50%, -100%);
43 }

```

## template

```

1 <template>
2 <!-- 监听transition组件的钩子, 离开前可以触发自定义的close方法, 离开后把整个组件卸载 -->
3 <transition @before-leave="onClose" @after-leave="$emit('destroy')">
4   <div v-show="visible" :class="[type, 'message']" :style="customStyle" ref="m
5     @mouseleave="startTimer">
6     <span>{{ message }}</span>
7   </div>
8 </transition>
9 </template>

```

## Script

```

1 <script setup>
2 import { ref, computed, onMounted, nextTick } from "vue";
3 import { getOffsetOrSpace, getLastOffset } from './instance.js'
4
5 const props = defineProps({

```

```

6   id: { type: String, require: true },
7   type: { type: String, default: 'info' },
8   message: { type: String, default: "message" },
9   duration: { type: Number, default: 3000 },
10  onClose: { type: Function },
11  offset: { type: Number, default: 12 },
12  zIndex: { type: Number, default: 2000 }
13 })
14
15 const emits = defineEmits({ destroy: () => true }) // 当组件消失时抛出卸载钩子
16
17 const visible = ref(false);
18 const durationTime = ref(null) // 控制关闭组件的定时器
19 const messageRef = ref() // 组件ref
20 const height = ref(0) // 组件整体高度，通过上面的ref获取
21
22 function startTimer() {
23   if (props.duration !== 0) {
24     durationTime.value = setTimeout(() => {
25       close()
26     }, props.duration)
27   }
28 }
29 function close() {
30   visible.value = false
31 }
32 function clearTimer() {
33   clearTimeout(durationTime.value)
34   durationTime.value = null
35 }
36 // 可能会同时存在多个message，所以需要控制它们之间的offset
37 const lastOffset = computed(() => getLastOffset(props.id)) // 获取当前组件前一个me
38 const offset = computed(
39   () => getOffsetOrSpace(props.id) + lastOffset.value
40 )
41 const bottom = computed(() => height.value + offset.value)
42 const customStyle = computed(() => ({
43   top: `${offset.value}px`,
44   zIndex: props.zIndex,
45 })))
46
47 // 用于声明当组件实例被父组件通过模板引用访问时暴露的公共属性。
48 // 可以直接通过实例的.vm.exposed.bottom.value获取
49 defineExpose({
50   visible,
51   close,
52   bottom

```



```

53 })
54 onMounted(async ()=>{
55   startTimer()
56   visible.value = true
57   await nextTick()
58   height.value = messageRef.value ? messageRef.value.getBoundingClientRect().hei
59 })
60 </script>

```

## message实例

所有message的集合，挂载message，挂载可直接调用的success、info、error、warning方法。

被调用的真正使用的组件

```

1 import { h, render, shallowReactive } from 'vue'
2 import MessageContext from './message.vue'
3
4 let seed = 1
5 const instances = shallowReactive([])
6
7 const closeMessage = (instance) => {
8   const idx = instances.indexOf(instance)
9   if (idx === -1) return
10  instances.splice(idx, 1)
11  const { handler } = instance
12  handler.close()
13 }
14
15 export const getInstance = (id) => {
16   const idx = instances.findIndex((instance) => instance.id === id)
17   const current = instances[idx]
18   let prev
19   if (idx > 0) {
20     prev = instances[idx - 1]
21   }
22   return { current, prev }
23 }
24
25 export const getLastOffset = (id) => {
26   const { prev } = getInstance(id)
27   if (!prev) return 0
28   return prev.vm.exposed.bottom.value
29 }
30
31 export const getOffsetOrSpace = (id, offset) => {

```

```
32   const idx = instances.findIndex((instance) => instance.id === id)
33   return idx > 0 ? 16 : 60
34 }
35
36 const createMessage = (
37   options,
38   context
39 ) => {
40   const id = `message_${seed++}`
41   const userOnClose = options.onClose
42
43   const container = document.createElement('div')
44   const appendTo = document.body
45   const props = {
46     ...options,
47     id,
48     onClose: () => {
49       userOnClose?.()
50       closeMessage(instance)
51     },
52     onDestroy: () => {
53       render(null, container)
54     },
55   }
56   const vnode = h(
57     MessageContext,
58     props
59   )
60   vnode.appContext = context || message._context
61
62   render(vnode, container)
63   appendTo.appendChild(container.firstElementChild)
64
65   const vm = vnode.component
66
67   const handler = {
68     close: () => {
69       vm.exposed.visible.value = false
70     },
71   }
72
73   const instance = {
74     id,
75     vnode,
76     vm,
77     handler,
78     props: props,
```

```

79   }
80
81   return instance
82 }
83
84 const mergeOptions = (options) => {
85   return typeof options === 'string' ? { message: options } : options
86 }
87
88 const message = (options, context) => {
89   let config = mergeOptions(options)
90   const instance = createMessage(config, context)
91   instances.push(instance)
92   return instance.handler
93 }
94 const messageTypes = ['success', 'warning', 'info', 'error']
95 messageTypes.forEach((type) => {
96   message[type] = (options = {}) => {
97     let config = mergeOptions(options)
98     return message({ ...config, type }, null)
99   }
100 })
101 export default message

```

## 使用

```

1 <script setup>
2 import message from '@components/Message'
3
4 message('message')
5 message.success('success message')
6 message({
7   message: 'warning message',
8   type: 'warning',
9   duration: 1000
10 })
11 </script>

```

2023/3/19

右键菜单

主要逻辑为，首先监听标签的右键点击事件，阻止系统的默认行为，然后创建一个自定义的右键菜单，并将其设置为默认聚焦状态。接着，监听菜单失焦事件以关闭菜单。

## 监听右键点击事件

在标签上监听 `@click.right.native` 事件;

views/test.vue

```
1 <script setup>
2 const showContextMenu = (e) => {
3   e.preventDefault()
4   console.log('监听右键点击')
5 }
6 </script>
7 <template>
8   <div class="context-menu" @click.right.native="showContextMenu($event)">
9     展示右键菜单
10  </div>
11 </template>
12 <style scoped lang="scss">
13 </style>
14
```

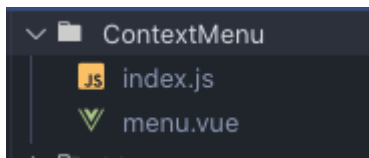
效果



## 菜单模板

现在，我们已经知道如何触发右键点击事件，并做一些相关的操作了，接下来展示我们的自定义菜单模板。

文件格式如下：



index.js: 主要功能是创建菜单、位置计算

```
1 // 先创建一个简单index.js
2 import MenuContext from './menu.vue'
3 const contextMenu = (e, data) => {
4   console.log(e, data)
5 }
6 export default contextMenu
```

menu.vue: 显示的菜单模板

```
1 // 先创建一个简单menu组件
2 <template>
3   <div class="context-menu">
4     显示的自定义右键菜单
5   </div>
6 </template>
7 <style>
8   .context-menu{
9     position: fixed;
10  }
11 </style>
```

## 菜单显示

右键点击时，应该计算菜单显示的位置，以避免出现临界情况。例如，如果我们在屏幕的右边界点击右键，如果菜单仍在鼠标的右侧显示，那么我们可能无法看到菜单。因此，在这种情况下，菜单应该在鼠标的左侧显示。

## 主要逻辑

前置知识：vue的 `h()` 和 `render()`；

官网链接：<https://cn.vuejs.org/api/render-function.html#h>

`h` 就是 `createVnode`，是Vue 提供了一个函数用于创建虚拟节点。

完整参数签名

```
1 // 第一个参数既可以是一个字符串（用于原生元素）也可以是一个 Vue 组件定义。
2 // 第二个参数是要传递的 prop，
```

```
3 // 第三个参数是子节点。
4 function h(
5   type: string | Component,
6   props?: object | null,
7   children?: Children | Slot | Slots
8 )
```

render：用于程式化地创建组件虚拟 DOM 树的函数。

需提供至少两个参数，第一个为需要渲染的虚拟节点，第二个为需要渲染到的容器节点

```
1 import MenuContext from './ContextMenu'
2 import { h, render } from 'vue'
3
4 const contextMenu = (e, data) => {
5   // 创建一个临时的div，用于挂载我们的菜单
6   const container = document.createElement('div')
7   // 获取body标签，用于挂载整个菜单
8   const appendTo = document.body
9   // 传给组件的props
10  const props = {
11    ...
12  }
13  // 渲染虚拟节点
14  const vnode = h(
15    MenuContext,
16    props
17  )
18  // vnode为需要渲染的虚拟节点，container为渲染的容器
19  render(vnode, container)
20 }
```

可以输出看看，渲染前后各个节点的情况。

vnode：就是我们 `menu.vue` 组件的相关信息

vnode: [index.js?t=1679279069702:24](#)  
[index.js?t=1679279069702:25](#)

```

▼ Object
  anchor: null
  appContext: null
  children: null
  ▶ component: {uid: 78, vnode: {...}, type: {...}, parent: null, appContext: {...}, ...}
  ctx: null
  dirs: null
  dynamicChildren: null
  dynamicProps: null
  ▶ el: div.context-menu
  key: null
  patchFlag: 0
  ▶ props: {data: Proxy(Object), onClose: f}
  ref: null
  scopeId: null
  shapeFlag: 4
  slotScopeIds: null
  ssContent: null
  ssFallback: null
  staticCount: 0
  suspense: null
  target: null
  targetAnchor: null
  transition: null
  ▶ type: {__name: 'menu', props: {...}, __hmrId: '90952d98', setup: f, render: f, ...}
  __v_isVNode: true
  __v_skip: true
  ▶ [[Prototype]]: Object

```

渲染前的container：就是一个空的div标签

```

container:
  firstChild: null
  firstElementChild: null
  hidden: false
  id: ""
  inert: false
  innerHTML: ""
  innerText: ""
  inputMode: ""
  isConnected: false
  isContentEditable: false
  lang: ""
  lastChild: null
  lastElementChild: null

```

渲染后的container：已经挂载了vnode，并能从container上获取相关信息

```

container:
  ▶ firstChild: div.context-menu
  ▶ firstElementChild: div.context-menu
  hidden: false
  id: ""
  inert: false
  innerHTML: "<div data-v-90952d98=\"\" class=\"context-menu\" tabindex=\"-1\"><div
  innerText: "Going HomeSophie Zelmani播放添加到队列添加到我喜欢的音乐添加到歌单复制链接"
  inputMode: ""
  isConnected: false
  isContentEditable: false
  lang: ""
  ▶ lastChild: div.context-menu
  ▶ lastElementChild: div.context-menu

```



在 `render` 函数执行完以后，我们就已经能获取到虚拟的dom节点了，这时候只要把它挂载到我们想要显示的位置上就好了。

## 位置计算

我们需要获取的数据有：渲染出来的菜单高度和宽度、当前可视区域的高度和宽度、当前点击时相对于浏览器的坐标。

```
1 // 首先需要先把菜单真正渲染到页面，才能拿到它的宽度和高度
2 appendTo.appendChild(container.firstChild)
3 // 当前真正的菜单节点，上面输出的vnode中可以看到，el就是我们的菜单节点
4 const curMenu = vnode.el
5 // 获取curMenu的高度和宽度，用于临界的计算
6 const { offsetWidth, offsetHeight } = curMenu
7 // 获取body的可视区域的宽度
8 const { clientWidth } = appendTo
9 // 取出右键点击时的坐标，clientX是距离左侧的位置，clientY是距离顶部的位置
10 const { clientX, clientY } = e
11
12 // 当前可视区域的宽度 - 当前鼠标距离浏览器左边的距离
13 // 如果 大于菜单的宽度，说明正常设置菜单距离左边界的距离,即设置style.left
14 // 否则菜单需要在鼠标左侧展示，即需要设置style.right组件距离可视区域右侧的距离
15 const leftOrRight = clientWidth - clientX > offsetWidth ? "left" : "right"
16
17 // 当前浏览器的高度(不包含滚动条) - 当前鼠标距离浏览器上边的距离
18 // 如果 大于菜单的高度，说明可以正常设置菜单距离上边界的距离,即设置style.top
19 // 否则需要设置菜单距离底部边界的位置，即style.bottom
20 const topOrBottom = window.innerHeight - clientY > offsetHeight ? "top" : "bottom"
21
22 // 设置top或者bottom的style
23 curMenu.style[leftOrRight] = leftOrRight === "left" ? `${clientX + 20}px` : `2px`
24
25 // 设置left或者right的style
26 curMenu.style[topOrBottom] = topOrBottom === "bottom" ? '2px' : `${clientY}px`
```

## 结果展示



## 关闭菜单

可以发现上面的结果中，我们展示了三个菜单，实际情况中我们应该在下一次右键时，关闭并卸载上一个菜单。

这里我们可以维护一个菜单实例，在创建菜单前，判断实例是否已经存在了，如果已经存在，先把它卸载。

index.js的完整代码：

```
1 import MenuContext from './menu.vue'
2 import { h, render } from 'vue'
3
4 // 维护一个菜单实例
5 let curInstance = null
6 let seed = 1
7 const contextMenu = (e, data) => {
8   if (curInstance) {
```

```

9     curInstance.destroy()
10 }
11 curInstance = null
12 let id = seed++
13 // 创建一个临时的div, 用于挂载我们的菜单
14 const container = document.createElement('div')
15 // 获取body标签, 用于挂载整个菜单
16 const appendTo = document.body
17 // 传给menu组件的props
18 const props = {
19   data,
20   onClose: () => {
21     curInstance.destroy()
22   },
23 }
24 // 渲染虚拟节点
25 const vnode = h(
26   MenuContext,
27   props
28 )
29 // vnode为需要渲染的虚拟节点, container为渲染的容器
30 render(vnode, container)
31 // 首先需要先把菜单真正渲染到页面, 才能拿到它的宽度和高度
32 appendTo.appendChild(container.firstElementChild)
33 // 当前真正的菜单节点, 上面输出的vnode中可以看到, el就是我们的菜单节点
34 const curMenu = vnode.el
35 // 获取curMenu的高度和宽度, 用于临界的计算
36 const { offsetWidth, offsetHeight } = curMenu
37 // 获取body的可视区域的宽度
38 const { clientWidth } = appendTo
39 // 取出右键点击时的坐标, clientX是距离左侧的位置, clientY是距离顶部的位置
40 const { clientX, clientY } = e
41
42 // 当前可视区域的宽度 - 当前鼠标距离浏览器左边的距离
43 // 如果 大于菜单的宽度, 说明正常设置菜单距离左边界的距离, 即设置style.left
44 // 否则菜单需要在鼠标左侧展示, 即需要设置style.right组件距离可视区域右侧的距离
45 const leftOrRight = clientWidth - clientX > offsetWidth ? "left" : "right"
46
47 // 当前浏览器的高度(不包含滚动条) - 当前鼠标距离浏览器上边的距离
48 // 如果 大于菜单的高度, 说明可以正常设置菜单距离上边界的距离, 即设置style.top
49 // 否则需要设置菜单距离底部边界的位置, 即style.bottom
50 const topOrBottom = window.innerHeight - clientY > offsetHeight ? "top" : "bottom"
51 const offsetLeft = Math.abs(clientWidth - clientX)
52 // 设置left或者right的style
53 curMenu.style[leftOrRight] = leftOrRight === "left" ? `${clientX + 20}px` : `-${offsetLeft}px`
54 // 设置top或者bottom的style
55 curMenu.style[topOrBottom] = topOrBottom === "bottom" ? '2px' : `-${clientY}px`

```

```
56
57   const instance = {
58     id,
59     destroy: () => {
60       render(null, container)
61     },
62   }
63   curInstance = instance
64   return instance
65
66 }
67
68 export default contextMenu
```

效果：



效果实现了，但是这时又发现另一个问题，当切换页面时，菜单没有正常关闭。

## focus和blur

为了解决上面的问题，我们可以在菜单正在挂载以后，使其聚焦，即主动触发 `focus` 事件，然后监听它的失焦事件 `blur`，当失焦时触发props传进来的 `destroy` 方法。

需要注意的是，要给 `div` 标签设置 `tabindex` 属性，否则无法触发 `focus` 事件。

menu.vue完整代码：

```
1 <script setup>
2 import { computed, onMounted, ref, nextTick } from "vue";
3 const props = defineProps({
```

```
4   data: { default: null },
5   onClose: { type: Function, default: () => {} },
6 });
7 const showData = computed(() => {
8   let data = {},
9   pd = props.data;
10  if (pd) {
11    data.name = pd.name;
12    data.id = pd.id;
13    data.subname = pd.ar[0].name;
14    data.picUrl = pd.al.picUrl;
15  }
16  return data;
17 });
18
19 // 创建组件ref
20 const contextMenu = ref(null);
21 onMounted(async () => {
22   // 确保组件已经渲染
23   await nextTick();
24   // 触发组件focus
25   contextMenu.value.focus();
26 });
27
28 const clickFunc = (type) => {
29   props.onClose();
30 };
31 const actions = [
32   "",
33   {
34     label: "播放",
35     type: "play",
36   },
37   {
38     label: "添加到队列",
39     type: "添加到队列",
40   },
41   "",
42   {
43     label: "添加到我喜欢的音乐",
44     type: "添加到我喜欢的音乐",
45   },
46   {
47     label: "添加到歌单",
48     type: "添加到歌单",
49   },
50   {
```

```
51     label: "复制链接",
52     type: "复制链接",
53   },
54 ];
55 </script>
56 <template>
57   <div
58     v-if="showData.name"
59     class="context-menu"
60     ref="contextMenu"
61     @blur="onClose"
62     tabindex="-1"
63   >
64     <div class="context-menu__info">
65       
66       <div>
67         <div class="context-menu__name">
68           {{ showData.name }}
69         </div>
70         <div class="context-menu__subname">
71           {{ showData.subname }}
72         </div>
73       </div>
74     </div>
75     <div v-for="(item, idx) in actions" :key="idx">
76       <div class="divide" v-if="!item" />
77       <div v-else class="context-menu__item" @click="clickFunc(item.type)">
78         {{ item.label }}
79       </div>
80     </div>
81   </div>
82 </template>
83 <style scoped lang="scss">
84   .context-menu {
85     position: fixed;
86     padding: 12px 4px;
87     border-radius: 6px;
88     border: 1px solid rgba(222, 222, 222, 0.5);
89     background-color: #ffffff;
90     font-size: 14px;
91     font-weight: 500;
92     user-select: none;
93     &:focus {
94       outline: none;
95     }
96
97     .divide {
```



```
98     height: 1px;
99     background-color: rgba(222, 222, 222, 0.5);
100     margin: 8px auto;
101     width: calc(100% - 12px);
102 }
103
104 &__item {
105     padding: 8px 12px;
106     cursor: pointer;
107
108     &:hover {
109         color: #646cff;
110         background-color: rgba(100, 108, 255, 0.1);
111         border-radius: 6px;
112     }
113 }
114
115 &__info {
116     display: flex;
117     padding: 0 12px;
118 }
119
120 &__img {
121     width: 36px;
122     height: 36px;
123     margin-right: 6px;
124 }
125
126 &__name {
127     font-size: 16px;
128 }
129
130 &__subname {
131     color: #666;
132     font-size: 12px;
133 }
134 }
135 </style>
136
```

## test.vue完整代码

```
1 <script setup>
2 import axios from "axios";
3 import ContextMenu from "../components/ContextMenu";
```

```
4 const showContextMenu = (e) => {
5   e.preventDefault();
6   axios
7     .get(
8       "https://www.fastmock.site/mock/6b16c722604e6f9b79e16f7ec3a768d4/vue3vite/"
9     )
10    .then((res) => {
11      ContextMenu(e, res.data.playlist.tracks[0]);
12    });
13 };
14 </script>
15 <template>
16   <div @click.right.native="showContextMenu($event)">
17     展示右键菜单
18   </div>
19   <div style="text-align: right;" @click.right.native="showContextMenu($event)">
20     展示右侧边界菜单
21   </div>
22   <div style="position: absolute;bottom: 0px" @click.right.native="showContextMe
23     展示底部边界菜单
24   </div>
25 </template>
26 <style scoped lang="scss"></style>
27
```

## 最终效果



2023/3/21

简单骨架屏 vue-content-loader

安装

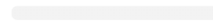
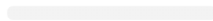
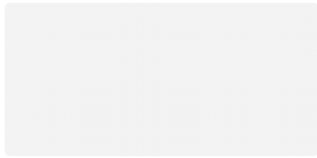
```
1 pnpm i vue-content-loader
```

使用

- `x` : 矩形左上角的 x 坐标。
- `y` : 矩形左上角的 y 坐标。
- `rx` : 矩形的圆角 x 方向半径。
- `ry` : 矩形的圆角 y 方向半径。
- `width` : 矩形的宽度。
- `height` : 矩形的高度。

```
1 <script setup>
2 import { ContentLoader } from 'vue-content-loader'
3 </script>
4 <template>
5 <content-loader
6   width="100%"
7   height="80vh"
8   primaryColor="#f3f3f3"
9   secondaryColor="#cccccc"
10 >
11   <rect x="0" y="0" rx="8" ry="8" width="30%" height="30%" />
12   <rect x="35%" y="8" rx="8" ry="8" width="60%" height="28px" />
13   <rect x="35%" y="68" rx="8" ry="8" width="40%" height="28px" />
14   <rect x="35%" y="108" rx="8" ry="8" width="55%" height="28px" />
15   <rect x="35%" y="158" rx="8" ry="8" width="58px" height="38px" />
16   <rect x="43%" y="158" rx="8" ry="8" width="58px" height="38px" />
17   <rect x="51%" y="158" rx="8" ry="8" width="58px" height="38px" />
18
19   <rect x="0" y="45%" rx="8" ry="8" width="10%" height="10%" />
20   <rect x="12%" y="46%" rx="8" ry="8" width="15%" height="20px" />
21   <rect x="12%" y="50%" rx="8" ry="8" width="10%" height="20px" />
22   <rect x="42%" y="48%" rx="8" ry="8" width="20%" height="20px" />
23   <rect x="75%" y="48%" rx="8" ry="8" width="20%" height="20px" />
24 </content-loader>
25 </template>
```

效果



2023/4/2

## 滚动条 vue-slider-component

[官网](#)

### 安装

```
1 pnpm install vue-slider-component@next --save
```

### 使用

```
1 <script setup>
2 import { ref } from 'vue'
3 import VueSlider from 'vue-slider-component'
4 import 'vue-slider-component/theme/antd.css'
5
6 const timeCtrl = ref(0)
7 const beginTime = ref(0)
8 const handleDragEnd = () => {
9   beginTime.value = timeCtrl.value
10   wyPlayer.seek(timeCtrl.value)
11 }
12 </script>
13
14 <template>
15   <span>{{ beginTime }}</span>
16   <VueSlider
17     v-model="timeCtrl"
```

```
18         :min="0"
19         :max="~~(currentPlaying.dt / 1000)"
20         :interval="1"
21         :drag-on-click="true"
22         :duration="0"
23         :dot-size="12"
24         :height="2"
25         :tooltip-formatter="secondToTime"
26         :lazy="true"
27         :silent="true"
28         @drag-end="hanldeDragEnd"
29     />
30 </template>
```