

项目 3：慕云游移动商城

1. 项目目标与准备

1.1 开发技术选型

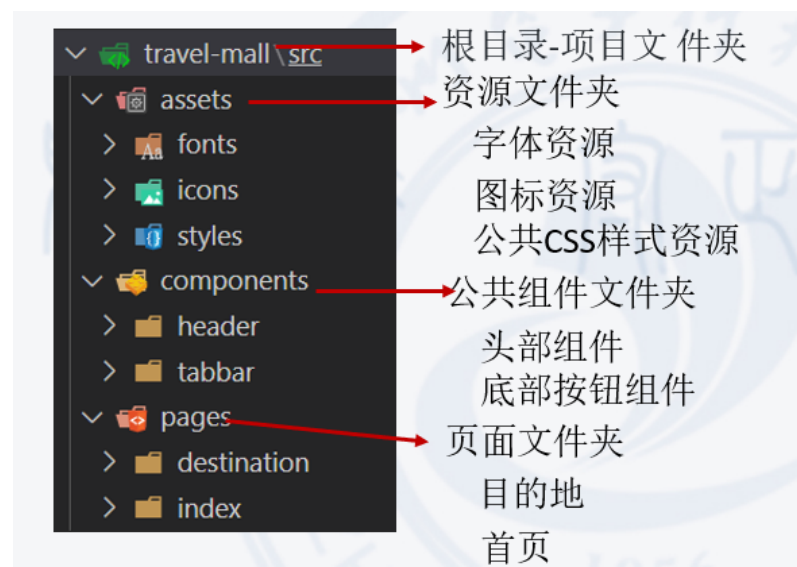
- 开发语言：ES6
- 获取数据：Ajax/Axios/promise/async/await
- 打包工具:Webpack
- 模板引擎：Art-template
- 触摸滑动库：Swiper
- 布局：Flex 布局等

1.2 项目目标

- 移动端屏幕适配
- 模块化与组件化
- 在项目中使用 Ajax 获取服务器端数据

1.3 项目准备

(1) 项目目录结构



(2) 项目配置

Package.json 文件配置。

```
package.json X
12-6 > travel-mall > package.json > {} devDependencies
1  {
2    "name": "travel-mall",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "webpack-dev-server --open"
8    },
9    "author": "",
10   "license": "ISC",
11   "devDependencies": {
12     "art-template-loader": "^1.4.3",
13     "css-loader": "^6.7.2",
14     "file-loader": "^6.2.0",
15     "html-webpack-plugin": "^5.5.0",
16     "style-loader": "^3.3.1",
17     "url-loader": "^4.1.1",
18     "webpack": "^5.75.0",
19     "webpack-cli": "^5.0.0",
20     "webpack-dev-server": "^4.11.1"
21   },
22   "dependencies": {
23     "art-template": "^4.13.2",
24     "swiper": "^6.1.1"
25   }
26 }
```

Webpack.config.js 文件配置:

```
const HtmlWebpackPlugin = require('html-webpack-plugin')
const path = require('path')

module.exports = {
  mode: 'development',
  entry: './src/pages/index/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: '[name].js'
  },
  resolve: {
    preferRelative: true, // 解决 art-template 不能解析上一层地址的问题
    // 自动补全（可以省略）的扩展名
    extensions: ['.js'],
    // 路径别名
    alias: {
```

```
    // api: resolve('src/api'),
    fonts: path.resolve('src/assets/fonts'),
    icons: path.resolve('src/assets/icons'),
    // images: resolve('src/assets/images'),
    styles: path.resolve('src/assets/styles'),
    components: path.resolve('src/components'),
    // pages: resolve('src/pages')
  }
},
module: {
  rules: [
    {
      test: /\.art$/,
      loader: 'art-template-loader'
    },
    {
      test: /\.css$/,
      use: ['style-loader', 'css-loader']
    },
    // 图片
    {
      test: /\.?(png|jpe?g|gif|svg)$/,
      loader: 'url-loader',
      options: {
        // 小于 10K 的图片转成 base64 编码的 dataURL 字符串写到
        limit: 10000,
        // 其他的图片转移到
        name: 'images/[name].[ext]',
        esModule: false
      }
    }
  ],
},
plugins: [
  new HtmlWebpackPlugin({
    template: './src/pages/index/index.art',
    filename: 'index.html'
  })
]
}
```

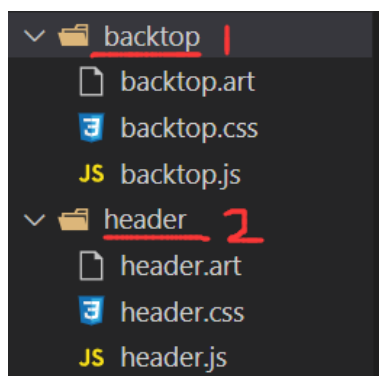
代码中

2. 组件化开发的基本原理

2.1 组件的构成

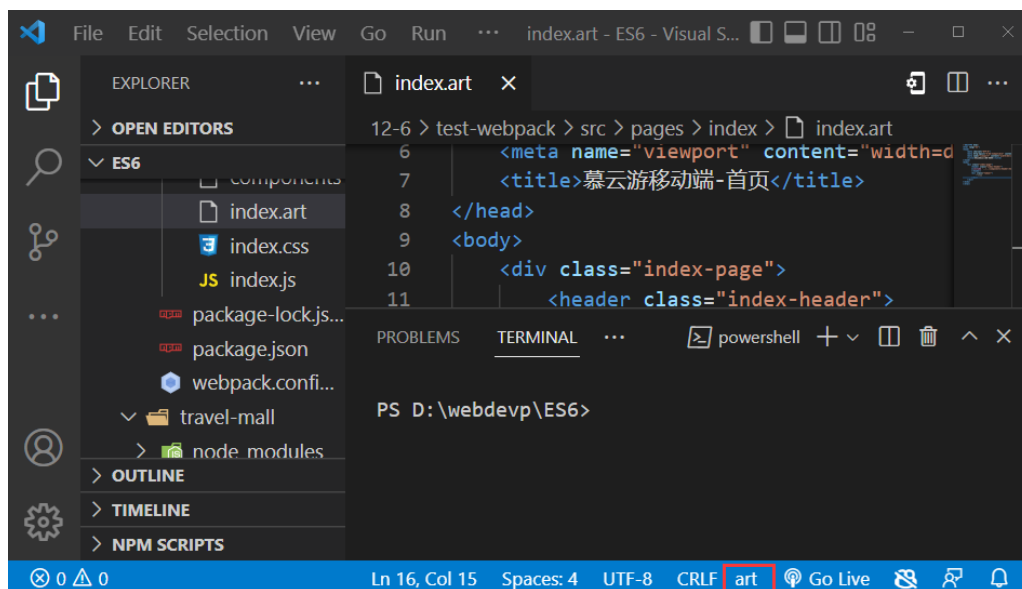
(1) 组件的构成

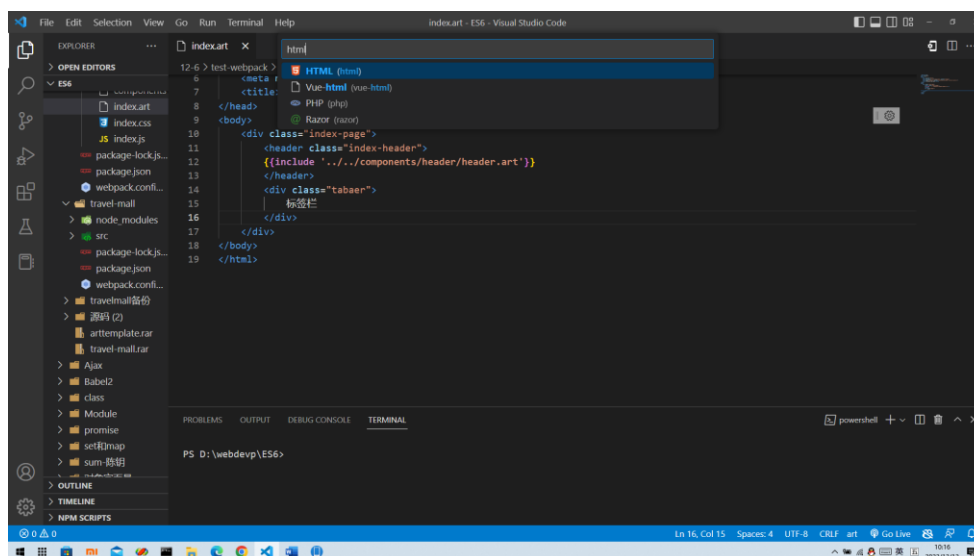
一个组件主要由三个文件构成，如 **backtop** 组件（回到顶部）分别由 **backtop.art**、**backtop.css** 和 **backtop.js** 三个文件构成，**header** 组件由 **header.art**、**header.css** 和 **header.js** 构成。



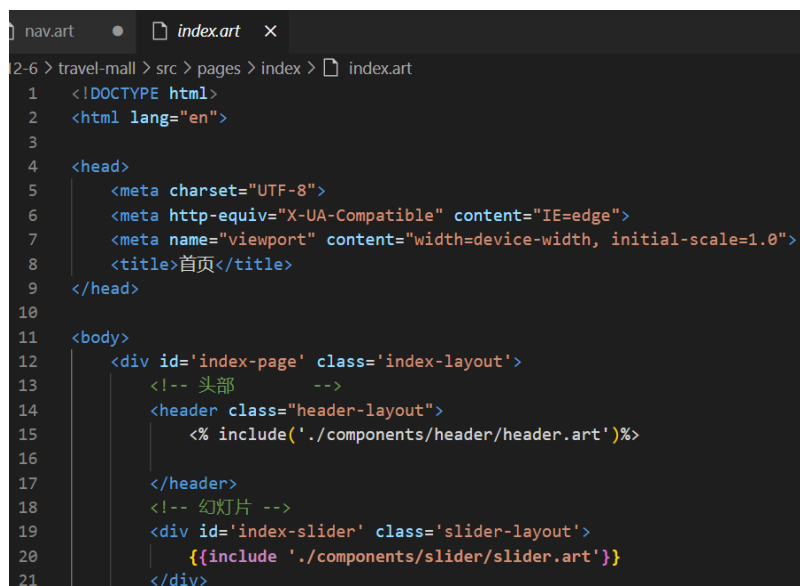
(2) 组件模板

本项目开发采用 **art-template** 模板引擎，模板文件的扩展名为 **.art**。**art** 模板相当于 **html** 页面，在 **vscode** 中需要关联到 **html** 格式才具有 **html** 文件的特征。关联方法：点击下图底部属性栏的 **art**，选择语言模型，在顶部弹出框中输入 **html** 后回车，可将 **art** 文件关联到 **html** 文件。





组件模板的使用通常有两种情况。一是作为首页模板（如 `index.art`）。此时需要有完整 html 结构，包含 `html`、`body`、`header`、`meta`、`title` 等标签，如下图所示。



二是作为功能组件模板。功能组件模板可以引用其它功能组件模板，但最终要嵌入到主页模板才能输出（组件引用方法见下一节组件的通信）。功能组件模板不需要完整的 html 结构标签（`html`、`body`、`header`、`meta`、`title` 等），只需相应功能标签（如 `div`、`p`、`h1`、`ul`、`li` 等）即可，如下图所示。下图为精选折扣功能组件查获板，其中包含 `div`、`h2`、`ul`、`li`、`a` 等标签。

```
nav.art  product.art X
12-6 > travel-mall > src > pages > index > components > product > product.art
1  <div class="product">
2  <h2 class="product-title">精选折扣</h2>
3  <ul id="product-list" class="product-list">
4  |   {{each productdata}}
5  |   <li class="product-item">
6  |   |   <a href="./details.html?id={{value.id}}">
7  |   |   |   <p class="product-pic">
8  |   |   |   
9  |   |   |   </p>
10 |   |   |   <p class="product-name">{{value.title}}</p>
11 |   |   |   <p class="product-price">{{value.price}}元起</p>
12 |   |   </a>
13 |   </li>
14 |   {{/each}}
15 </ul>
16 </div>
```

2.2 组件通信

(1) 组件内的通信。

一个组件内的三个文件之间的信息传递称为组件内通信，包括引用 css 文件，传递参数到 art 模板等。

- 引用组件内 css 样式。若想在 art 模板中使用组件内的.css 文件,需要在组件的.js 文件中引入.css,方法如下。Import 为文件所在的地址,同级目录使用“./”定位到.css 文件。

```
header.js  X
12-6 > travel-mall > src > components > header.js
1  import './header.css'
```

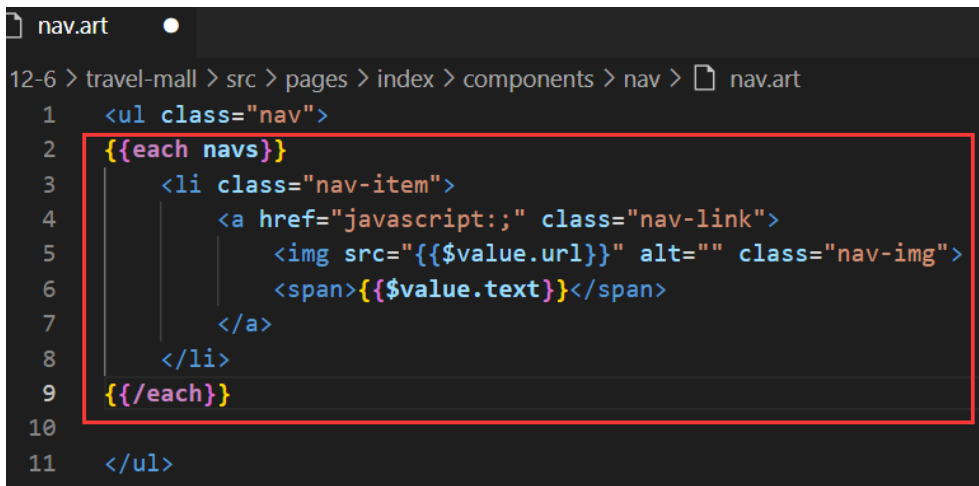
- 将 js 中的数据传入到 art 模板。使用方法分为模板引入、数据输出、数据接收三个步骤。首先,需要在组件的 js 文件中引入 art 模板文件,引入后的模板是一个函数 render (名字可自定)。下图是在导航组件 (nav) 的 js 文件 (nav.js) 中引入 nav.art 模板,并命名为 render。

```
import render from './nav.art'
```

其次是数据传递。获取 art 模板中的 dom 元素,并将数据作为 render 函数的参数传入到 dom 元素的 innerhtml 属性中,如下图所示。其中 data.data 为 Ajax 获取到的数据,navs 为对象属性,其值为 data.data。

```
document.getElementById('index-nav').innerHTML = render({
  navs: data.data
})
```

最后,在 art 模板中接收数据,并遍历,如下图所示。

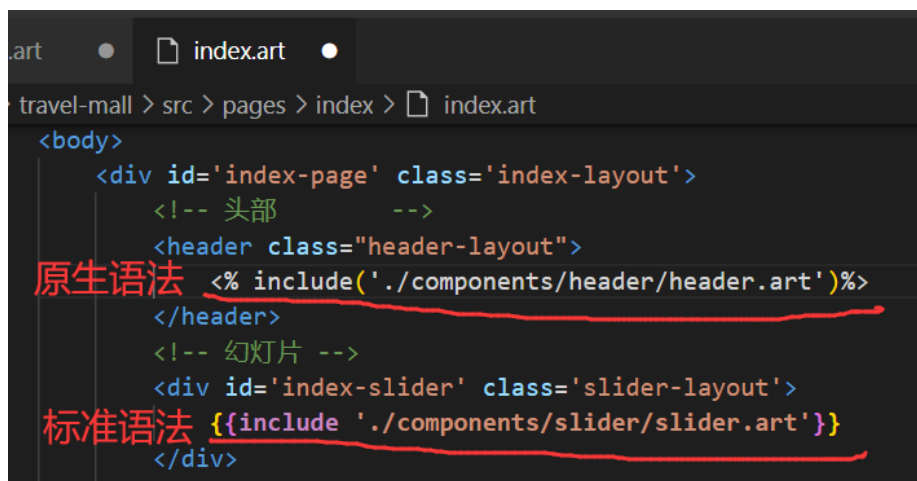


```
nav.art
12-6 > travel-mall > src > pages > index > components > nav > nav.art
1  <ul class="nav">
2  {{each navs}}
3    <li class="nav-item">
4      <a href="javascript:;" class="nav-link">
5        
6        <span>{{value.text}}</span>
7      </a>
8    </li>
9  {{/each}}
10
11 </ul>
```

(2) 组件之间的通信

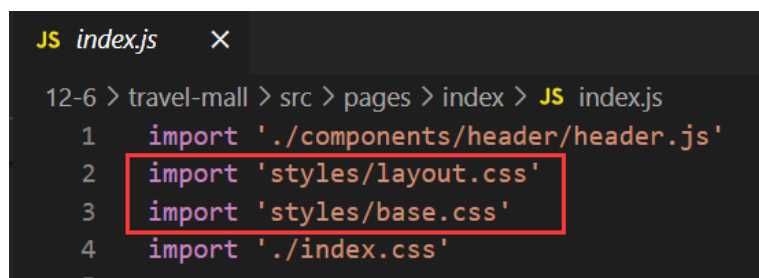
组件之间的通信包括模板引入、CSS 样式引入，整个组件的引入几种情况。

- 模板引入。在 art 模板中引入外部 art 模板。如在首页模板中引入其它功能组件模板，可以采用 art template 的标准语法和原生语法两种方式。标准语法用于不需要向功能组件模板传递参数，而原生语法则是用于需要向功能组件模板传递参数的情况。



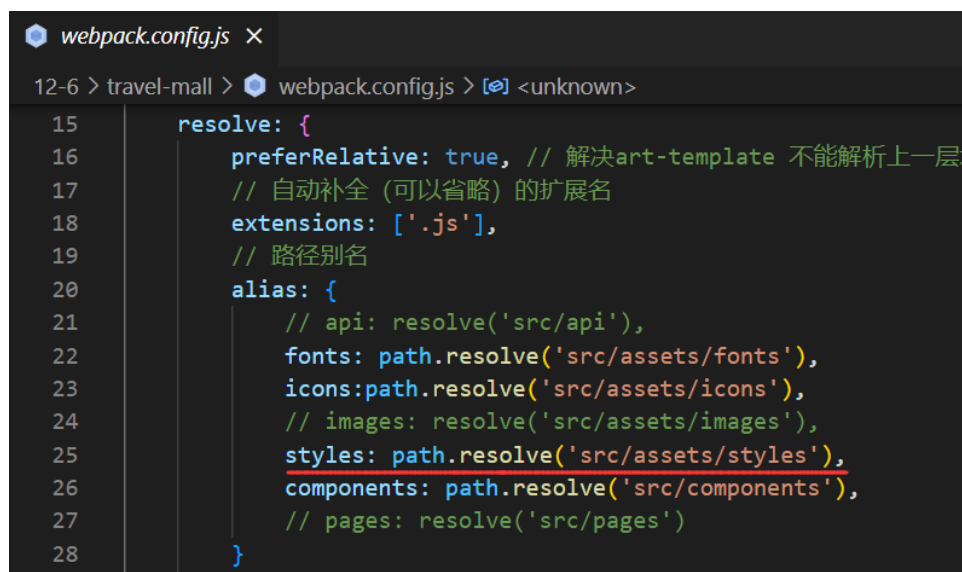
```
index.art
travel-mall > src > pages > index > index.art
<body>
  <div id='index-page' class='index-layout'>
    <!-- 头部 -->
    <header class="header-layout">
      原生语法 <% include('./components/header/header.art')%>
    </header>
    <!-- 幻灯片 -->
    <div id='index-slider' class='slider-layout'>
      标准语法 {{include './components/slider/slider.art'}}
    </div>
```

- 外部组件 CSS 文件引入。如在首页中引入公共组件的 css 文件，或其它功能组件的 CSS 样式。引入的方法是在首页的 js 文件通过 import 命令引入 css 文件，如下图所示。



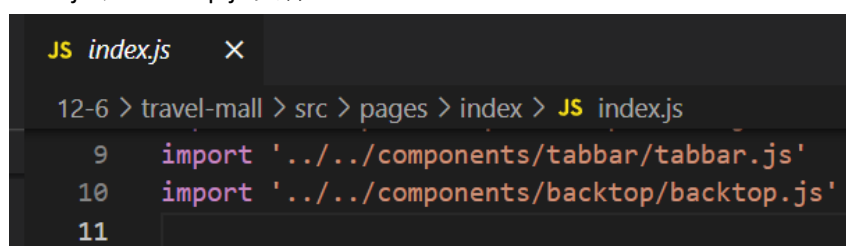
```
JS index.js
12-6 > travel-mall > src > pages > index > JS index.js
1  import './components/header/header.js'
2  import 'styles/layout.css'
3  import 'styles/base.css'
4  import './index.css'
```

需要注意的是上图中的 'styles/base.css' 中 styles 为地址的别名(alias)，需要事先在 webpack.config.js 中设置,如下图所示。'styles/base.css' 表示的地址为: 'src/assets/styles/base.css'。



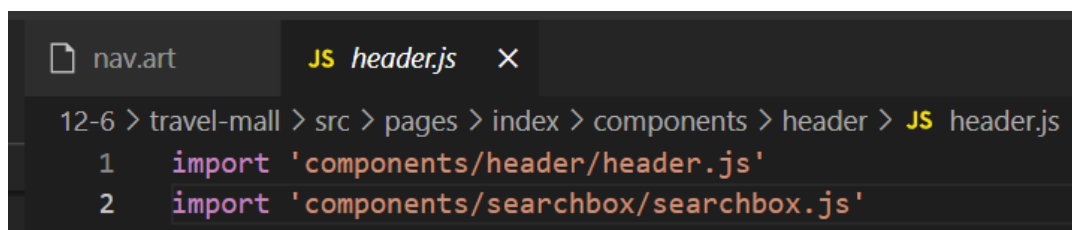
```
15 resolve: {
16   preferRelative: true, // 解决art-template 不能解析上一层
17   // 自动补全 (可以省略) 的扩展名
18   extensions: ['.js'],
19   // 路径别名
20   alias: {
21     // api: resolve('src/api'),
22     fonts: path.resolve('src/assets/fonts'),
23     icons: path.resolve('src/assets/icons'),
24     // images: resolve('src/assets/images'),
25     styles: path.resolve('src/assets/styles'),
26     components: path.resolve('src/components'),
27     // pages: resolve('src/pages')
28   }
29 }
```

如不想设置别名,可以通过`../..../`的方法返回上一层目录,定位到相应的文件,其中一个`../`表示返回一层目录,如有多层,则需要多个`../`拼接,如下图所示。下图为在`index.js`中调用公共组件`tabbar.js`和`backtop.js`文件。



```
9 import '../components/tabbar/tabbar.js'
10 import '../components/backtop/backtop.js'
11
```

- 整个功能组件引入。最后一定要在组件的`js`文档中引入外部组件的`js`文件,为组件的访问提供入口。下图为在页面级组件`header`中引入公共组件`header`和`searchbox`组件。



```
1 import 'components/header/header.js'
2 import 'components/searchbox/searchbox.js'
```

2.3 Ajax 数据获取与使用

(1) 数据获取

以导航组件获取真实数据为例。其`url`地址为:



```
url = 'https://www.imooc.com/api/mall-wepApp/index/nav'
```

获取数据分为 4 步:

- 创建`xhr`对象
- 监听`xhr`状态变化
- 准备发送请求
- 发送请求


```
import render from './nav.art'
const url = 'https://www.imoooc.com/api/mall-wepApp/index/nav'
const xhr = new XMLHttpRequest()
console.log(xhr)
xhr.onreadystatechange = () => {
  console.log(xhr.readyState);
  if (xhr.readyState !== 4) return;
  if ((xhr.status >= 200 && xhr.status <= 300) || xhr.status == 304) {
    const data = JSON.parse(xhr.responseText)
    // console.log(data.data)
    // console.log(typeof data.data)
    document.getElementById('index-nav').innerHTML = render({
      navs: data.data
    })
  }
}
xhr.open('get', url)
xhr.send()
```

接收后的数据保存在 `xhr.responseText` 中，通常需要使用 `json.parse` 转换数据为对象。

(2) 数据输出

首先引入接收数据的 art 模板文件:

```
import render from './nav.art'
```

然后将数据作为 `render` 的参数输出:

```
document.getElementById('index-nav').innerHTML = render({
  navs: data.data
})
```

(3) 数据接收与遍历

使用 `art-templates` 模板的 `each` 方面遍历 `navs` 数据，并将其值 (`url` 和 `text`) 输出到 `img` 和 `span` 标签中。

```
nav.art  x JS nav.js
12-6 > travel-mall > src > pages > index > components > nav > nav.art
1  <ul class="nav">
2  {{each navs}}
3    <li class="nav-item">
4      <a href="javascript:;" class="nav-link">
5        
6        <span>{{value.text}}</span>
7      </a>
8    </li>
9  {{/each}}
10
11 </ul>
```

2.4 事件监听

项目开发中多次要用到监听事件的方法,如在首页中滚动条向上滚动到顶部时出现绿色透明背景条,向下滚动一个页面后出现返回顶部图标等。

(1) 头部组件背景条出现与隐藏的监听

对首页中 index-page 进行监听,事件为 scroll,当 scrollTop 大于 0 时,为 header 组件添加新的样式 header-transition。当 scrollTop 小于等于 0 时,删除样式 header-transition。

```
JS header.js ×
12-6 > travel-mall > src > components > header > JS header.js > [0] INIT_STATE
1  import './header.css'
2  import 'styles/reset.css'
3  import 'styles/base.css'
4
5  //header组件功能, 页面向下滚动后出现header组件, 滚动为0后隐藏组件
6  const scrollContainer = document.getElementById('index-page')
7  const headerEl = document.querySelector('.header')
8  // console.log(scrollContainer)
9  // console.log(headerEl)
10 const INIT_STATE = 'init'
11 const CHANGED_STATE = 'changed'
12 let state = INIT_STATE
13 // console.log(scrollContainer,headerEl)
14 scrollContainer.addEventListener('scroll',() => {
15     if (state !== CHANGED_STATE && scrollContainer.scrollTop > 0) {
16         state = CHANGED_STATE
17         headerEl.classList.add('header-transition')
18     } else if (state !== INIT_STATE && scrollContainer.scrollTop <= 0) {
19         state = INIT_STATE
20         headerEl.classList.remove('header-transition')
21     }
22 }, false)
```

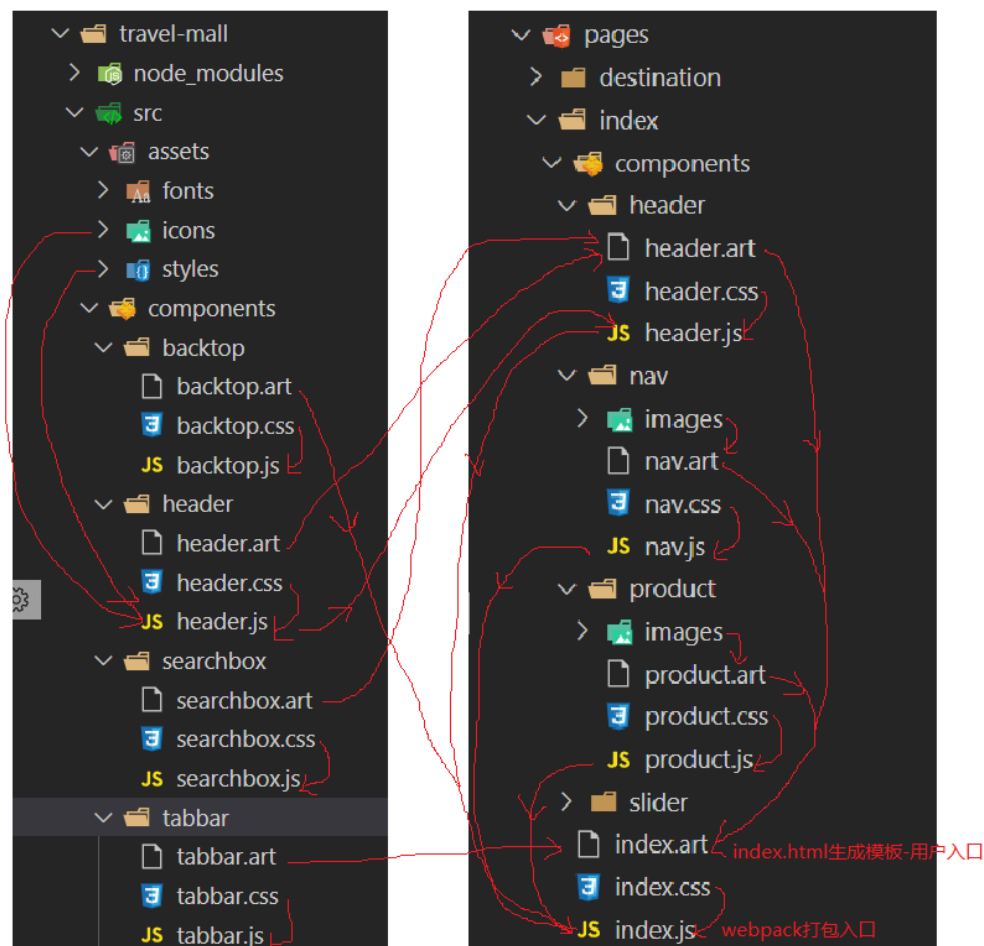
(2) 返回顶部按钮的监听

返回顶部按钮的监听的方法与上述相似。

```
//backtop组件功能, 页面向下滚动滚动一个页面后(window.innerHeight)出现backtop组件, 向上滚动又隐藏组件
const scrollContainer = document.getElementById('index-page')
const backtopEl = document.querySelector('.backtop')
const INIT_STATE = 'init'
const CHANGED_STATE = 'changed'
let state = INIT_STATE
// console.log(scrollContainer,headerEl)
scrollContainer.addEventListener('scroll',() => {
    if (state !== CHANGED_STATE && scrollContainer.scrollTop <= window.innerHeight) {
        state = CHANGED_STATE
        backtopEl.classList.add('backtop-hidden')
    } else if (state !== INIT_STATE && scrollContainer.scrollTop > window.innerHeight) {
        state = INIT_STATE
        backtopEl.classList.remove('backtop-hidden')
    }
}, false)
//点击回到顶部
backtopEl.addEventListener('click',()=>{
    scrollContainer.scrollTo({
        top:0,
        left:0,
        behavior:'smooth'
    })
},false)
```

2.5 项目组件之间的访问逻辑

以首页为例,首页的 `index.js` 文件是 `webpack` 打包的入口, `index.art` 是用于生成 `index.html` (用户入口) 的模板,也是合成其它组件的模板。`CSS` 文件在 `js` 中引入,组件的 `art` 可以拼接模板,子组件的 `js` 文件作为父组件引入的入口。



3. 首页开发

在 `pages/index` 的目录下新建 `index.js`、`index.css`、`index.art` 三个文件。

3.1 首页模板编写

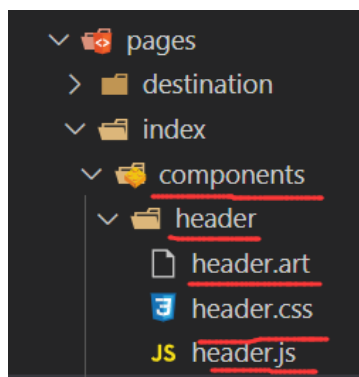
在 `pages/index` 的 `index.art` 中编写首页模板如下:

```
<body>
  <div id='index-page' class='index-layout'>
    <!-- 头部 -->
    <header class="header-layout">头部</header>
    <!-- 幻灯片 -->
    <div id='index-slider' class='slider-layout'>幻灯片</div>
    <!-- 导航 -->
    <div id="index-nav" class="nav-layout">导航</div>
    <!-- 精品折扣 -->
    <div id="index-product" class="product-layout">精品折扣</div>
    <!-- 底部标签栏 -->
    <div id="index-tabbar" class="tabbar-layout">首页底部</div>
    <!-- 回到顶部 -->
    <div id="index-backtop" class="backtop-layout">回到顶部</div>
  </div>
</body>
```

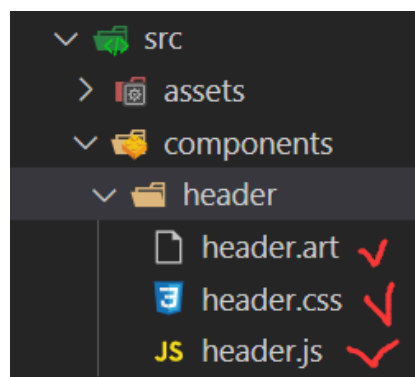
3.2 header 组件创建

(1) header 组件目录及文件创建

在 pages/index 目录下创建 components 文件夹，并在该目录下创建 header 文件夹，然后在 header 在创建 header.js、header.css、header.art 三个文件，如下图所示。在 src/components/header 公共组件目录下创建 header.js、header.css、header.art 三个文件，如下图所示。



header 页面级组件



header 公共组件

(2) 为 header 组件添加模板

在 pages/index/components/header 下的 header.art 中添加以下结构。

```
<div class="header">头部</div>
```

(3) 为 header 组件添加样式

在 src/components/header 下的 header.css 中添加如下 CSS 样式。

```
12-6 > travel-mall > src > components > header > header.css > .header
1  .index-layout{
2      overflow-y: auto;
3      height: 100%;
4  }
5  .header{
6      display: flex;
7      align-items: center;
8      justify-content: center;
9      position: fixed;
10     width: 100%;
11     height: 58px;
12     transition: background-color 0.5s;
13 }
14 }
15 .header-transition{
16     background-color: rgba(25, 196, 138, 0.4);
17 }
18
```

(4) header 公共样式引入

在 src/components/header 下的 header.js 中引入以下样式：

```
import './header.css'
import 'styles/reset.css'
import 'styles/base.css'
```

在 pages/index/components/header 下的 header.js 中引入 src/components/header/ header.js 文件。

(5) header 组件功能编写

在 src/components/header 下的 header.js 中添加以下 Javascript 代码：

```
//header组件功能，页面向下滚动后出现header组件，滚动为0后隐藏组件
const scrollContainer = document.getElementById('index-page')
const headerEl = document.querySelector('.header')
console.log(scrollContainer)
console.log(headerEl)
const INIT_STATE = 'init'
const CHANGED_STATE = 'changed'
let state = INIT_STATE
// console.log(scrollContainer,headerEl)
scrollContainer.addEventListener('scroll',() => {
    if (state !== CHANGED_STATE && scrollContainer.scrollTop > 0) {
        state = CHANGED_STATE
        headerEl.classList.add('header-transition')
    } else if (state !== INIT_STATE && scrollContainer.scrollTop <= 0) {
        state = INIT_STATE
        headerEl.classList.remove('header-transition')
    }
}, false)
```

(6) 在首页中引入 header 组件

在 pages/index 的 index.art 中引入 src/components/header/header.art，方法如下：

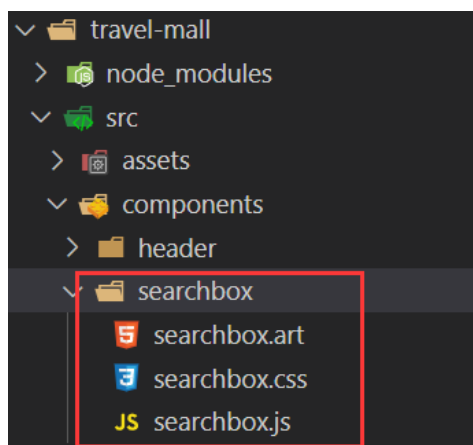
```
<div id='index-page' class='index-layout'>
  <!-- 头部 -->
  <header class="header-layout">
    <% include('./components/header/header.art')%>
  </header>
  <!-- 幻灯片 -->
```

在 pages/index 的 index.js 中引入 pages/index/components/header/ header.js 文件。

3.3 searchbox 组件创建

(1) 创建目录及文件

在 src/components 下创建 searchbox 目录及 art/css/js 三个文件。



(2) searchbox 模板编写

在 searchbox.art 中编写结构模板，如下图所示。

```
searchbox.css  JS searchbox.js  searchbox.art
1-6 > travel-mall > src > components > searchbox > searchbox.art > div.searchbox > form.search
1  <div class="searchbox">
2    <i class="iconfont icon-search"></i>
3    <form class="searchbox-form">
4      <input type="text" class="searchbox-input" name="words"
5        autocomplete="off"
6        placeholder="{{placeholder?placeholder:'搜索目的地/折扣/关键字'}}">
7    </form>
8  </div>
```

(3) 编写样式

在 searchbox.css 中为 searchbox.art 中的各结构编写 CSS 样式。

```
searchbox.css × index.css searchbox.art JS ind
12-6 > travel-mall > src > components > searchbox > searchbox.css >
1  .searchbox{
2    display: flex;
3    width: 100%;
4    height: 30px;
5    padding: 0 20px;
6    background-color: rgba(0,0,0,0.2);
7    align-items: center;
8    border-radius: 27.5px;
9    box-shadow: 0 5px 10px 0 rgba(0,0,0,0.05);
10 }
11 .searchbox .icon-search{
12   color: #fff;
13   font-size: 20px;
14 }
15 }
16 .searchbox-form{
17   flex: 1;
18   margin-left: 6px;
19 }
20 .searchbox-input{
21   width: 100%;
22   background: none;
23   color: #fff;
24   font-size: 14px;
25 }
```

```
/* placeholder颜色修改, 不同浏览器的适配 */
.searchbox-input::-webkit-input-placeholder{
  color: #fff;
}

.searchbox-input:-moz-placeholder{
  color: #fff;
}

.searchbox-input::-moz-placeholder{
  color: #fff;
}

.searchbox-input:-ms-input-placeholder{
  color: #fff;
}
```

在 pages/index 下的 index.css 中编写样式。

```
searchbox.css index.css ×
12-6 > travel-mall > src > pages > index >
1  .header-layout .searchbox{
2    width: 89.33%;
3  }
```

(4) 模板及样式引入

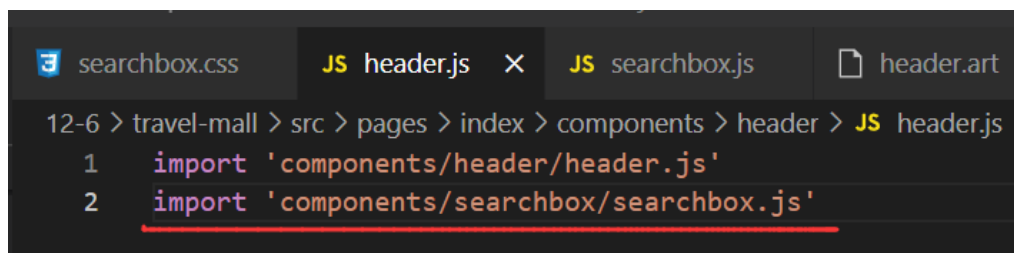
在 searchbox.js 中引入 searchbox.css 和图标样式 iconfont.css 样式。

```
index.css JS searchbox.js × reset.css
12-6 > travel-mall > src > components > searchbox >
1  import './searchbox.css'
2  import 'icons/iconfont.css'
```

在 pages/index/components/header/ header.art 中引入 searchbox.art 模板。

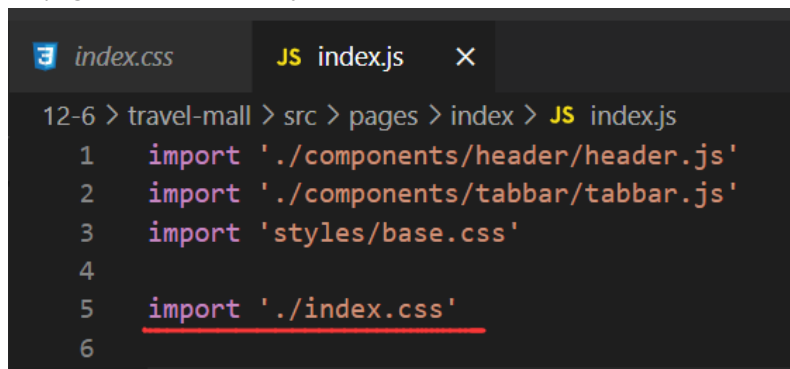
```
searchbox.css header.art × JS index.js JS header.js
12-6 > travel-mall > src > pages > index > components > header > header.art
1  <div class="header ">
2    {{include './../../components/searchbox/searchbox.art'}}
3  </div>
```

在 pages/index/components/header/ header.js 中引入 searchbox.js 样式。



```
12-6 > travel-mall > src > pages > index > components > header > JS header.js
1  import 'components/header/header.js'
2  import 'components/searchbox/searchbox.js'
```

在 pages/index 的 index.js 中引入 index.css 文件。



```
12-6 > travel-mall > src > pages > index > JS index.js
1  import './components/header/header.js'
2  import './components/tabbar/tabbar.js'
3  import 'styles/base.css'
4
5  import './index.css'
6
```

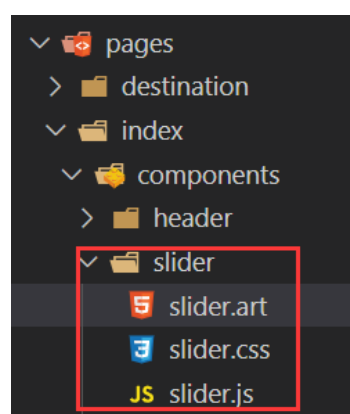
(5) 效果展示



3.4 幻灯片 slider 组件创建

(1) 创建 slider 组件目录及文件

Slider 是一个页面级的组件, 在 pages/index/components/ 下创建 slider 文件夹及 slider.css、slider.js、silder.art 文件。



(2) 搭建 slider 模板

在 silder.art 中搭建 swiper (<https://www.swiper.com.cn/usage/index.html>) 模板。


```
> travel-mall > src > pages > index > components > slider > slider.art
<div class="swiper-container">
  <div class="swiper-wrapper">
    <div class="swiper-slide">
      <a href="javascript:;">
        
      </a>
    </div>
    <div class="swiper-slide">
      <a href="javascript:;">
        
      </a>
    </div>
    <div class="swiper-slide">
      <a href="javascript:;">
        
      </a>
    </div>
  </div>
  <!-- 如果需要分页器 -->
  <div class="swiper-pagination"></div>
  <!-- 如果需要导航按钮 -->
  <!-- 如果需要滚动条 -->
</div>
```

(3) 在首页中引入 slider.art 和 slider.js 文件

在 pages/index 下的 index.art 中引入 slider.art。

```
<!-- 幻灯片 -->
<div id='index-slider' class='slider-layout'>
  {{include './components/slider/slider.art'}}
</div>
```

在 pages/index 下的 index.js 中引入 slider.js。

```
12-6 > travel-mall > src > pages > index > JS index.js
1  import './components/header/header.js'
2  import './components/tabbar/tabbar.js'
3  import 'styles/base.css'
4  import './index.css'
5
6  import './components/slider/slider.js'
```

(4) 添加图片

将 slider.art 中的 slider1, slider2, slider3 文本修改为图片内容，并添加超链接。

```
<div class="swiper">
  <div class="swiper-wrapper">
    <div class="swiper-slide">
      <a href="javascript:;">
        
      </a>
    </div>
    <div class="swiper-slide"> <a href="javascript:;">
      
    </a></div>
    <div class="swiper-slide"> <a href="javascript:;">
      
    </a></div>
  </div>
</div>
```

如果 webpack 不能正常解析图片, 请安装 file-loader 和 url-loader, 并在 webpack.config.js 中的 module>rules 中添加如下配置。

```
// 图片
{
  test: /\. (png|jpe?g|gif|svg)$/,
  loader: 'url-loader',
  options: {
    // 小于 10K 的图片转成 base64 编码的 dataURL 字符串写到代码中
    limit: 10000,
    // 其他的图片转移到
    name: 'images/[name].[ext]',
    esModule: false
  }
},
```

(5) 设置图片大小

在 slider.css 中为图片设置宽高, 可根据实际情况设置。

```
12-6 > travel-mall > src > pages > index > components > sl
1  .swiper-img{
2    width: 100%;
3    height: 180px;
4  }
```

(6) 通过 Ajax 获取真实数据

Url: <https://www.imooc.com/api/mall-wepApp/index/slider>

在 slider.js 中实现 ajax 获取数据。

```
JS slider.js x
12-6 > travel-mall > src > pages > index > components > slider > JS slider.js > onreadystatechange
1
2 import 'swiper/swiper-bundle.min.css'
3 import './slider.css'
4 import Swiper from 'swiper/swiper-bundle.min.js'
5
6 import render from './slider.art'
7 const url = 'https://www.imooc.com/api/mall-wepApp/index/slider'
8 const xhr = new XMLHttpRequest()
9 console.log(xhr)
10 xhr.onreadystatechange = () => {
11   console.log(xhr.readyState);
12   if (xhr.readyState !== 4) return;
13   if ((xhr.status >= 200 && xhr.status <= 300) || xhr.status == 304) {
14     const data = JSON.parse(xhr.responseText)
15     // console.log(data.data)
16     // console.log(typeof data.data)
17     document.getElementById('index-slider').innerHTML = render({
18       imgs: data.data
19     })
20     new Swiper('.swiper-container', {
21       loop: true, // 循环模式选项
22
23       // 如果需要分页器
24       pagination: {
25         el: '.swiper-pagination',
26       },
27
28     })
29   }
30 }
31 }
32 }
33 xhr.open('get', url)
34 xhr.send()
```

注：上图增加了 Swiper 模块的 css 和 js 文析导入，如下图所示。

```
JS slider.js x
12-6 > travel-mall > src > pages > index > components > slider > JS slider.js > ...
1
2 import 'swiper/swiper-bundle.min.css'
3 import './slider.css'
4 import Swiper from 'swiper/swiper-bundle.min.js'
5
```

在 slider.art 中渲染输出数据。

```
JS slider.js slider.art X
12-6 > travel-mall > src > pages > index > components > slider > slider.art > div.swiper-container
1 <div class="swiper-container">
2   <div class="swiper-wrapper">
3     {{each imgs}}
4       <div class="swiper-slide">
5         <a href="javascript:;">
6           
7         </a>
8       </div>
9     {{/each}}
10
11   </div>
12   <!-- 如果需要分页器 -->
13   <div class="swiper-pagination"></div>
14   <!-- 如果需要导航按钮 -->
15   <!-- 如果需要滚动条 -->
16
17 </div>
```

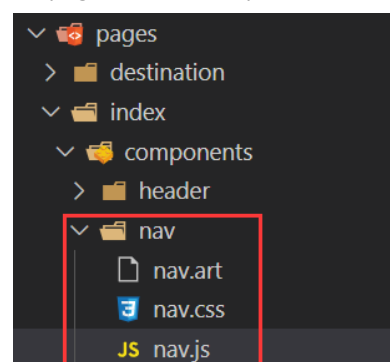
(7) 效果展示



3.5 导航 nav 组件创建

(1) 创建导航组件目录和文件

在 pages/index/components/下创建 nav 文件夹及 nav.css、nav.js、nav.art 文件。



(2) 模板及样式引入

在 nav.js 中引入 nav.css 文件。

```
JS nav.js  X  nav.css  nav.art
12-6 > travel-mall > src > pages > index > components > nav > JS nav.js
1  import './nav.css'
```

在 index.js 中引入 nav.js 文件。

```
JS nav.js  JS index.js  X  nav.css
12-6 > travel-mall > src > pages > index > JS index.js
1  import './components/header/header.js'
2  import './components/tabbar/tabbar.js'
3  import 'styles/base.css'
4  import './index.css'
5
6  import './components/slider/slider.js'
7  import './components/nav/nav.js'
```

在 index.art 中引入 nav.art 文件。

```
<!-- 导航 -->
<div id="index-nav" class="nav-layout">
  {{include './components/nav/nav.art'}}
</div>
```

(3) nav.art 模板编写

为 nav.art 编写模板，共有 8 个列表项，暂时使用假数据。

```
JS nav.js  nav.css  nav.art  X  index.art
12-6 > travel-mall > src > pages > index > components > nav > nav.art > ul.nav > li.nav
1  <ul class="nav">
2    <li class="nav-item">
3      <a href="javascript:;" class="nav-link">
4        
5        <span>自由行</span>
6      </a>
7    </li>
8    <li class="nav-item">
9      <a href="javascript:;" class="nav-link">
10       
11       <span>自由行</span>
12     </a>
13   </li>
```

(4) 编写样式

在 nav.css 中为模板的各部件编写样式。

```
nav.js  nav.css  nav.art
-6 > travel-mall > src > pages > index > components >
1  .nav{
2      display: flex;
3      flex-wrap: wrap;
4      width: 100%;
5      padding-top: 20.5px;
6  }
7  .nav-item{
8      width: 25%;
9      margin-bottom: 20.5px;
10 }
11  .nav-link{
12      display: flex;
13      justify-content: center;
14      align-items: center;
15      flex-direction: column;
16      color: #999;
17      font-size: 12px;
18  }
19  .nav-img{
20      width: 46.9333%;
21      margin-bottom: 5.5px;
22  }
```

(5) Ajax 数据获取

url:<https://www.imooc.com/api/mall-wepApp/index/nav>

```
import render from './nav.art'
const url = 'https://www.imooc.com/api/mall-wepApp/index/nav'
const xhr = new XMLHttpRequest()
console.log(xhr)
xhr.onreadystatechange = () => {
  console.log(xhr.readyState);
  if (xhr.readyState !== 4) return;
  if ((xhr.status >= 200 && xhr.status <= 300) || xhr.status == 304) {
    const data = JSON.parse(xhr.responseText)
    console.log(data.data)
    console.log(typeof data.data)
    document.getElementById('index-nav').innerHTML = render({
      navs: data.data
    })
  }
}
xhr.open('get', url)
xhr.send()
```

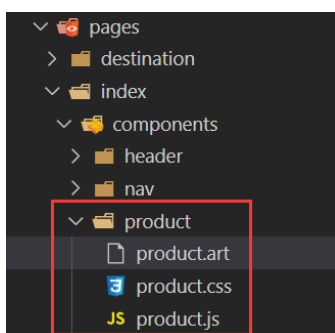
(6) 效果展示



3.6 精选折扣组件创建

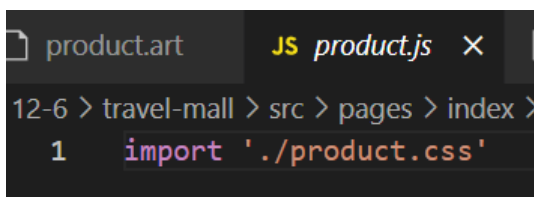
(1) 创建精选组件目录和文件

在 `pages/index/components/` 下创建 `product` 文件夹及 `product.css`、`product.js`、`product.art` 文件。

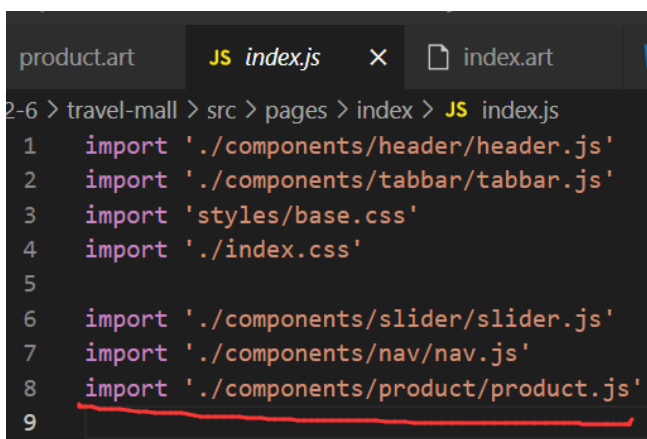


(2) 模板及样式引入

在 `product.js` 中引入 `product.css` 文件。



在 `index.js` 中引入 `product.js` 文件。



在 index.art 中引入 product.art 文件。

```
</div>
<!-- 精品折扣 -->
<div id="index-product" class="product-layout">
  {{include './components/product/product.art'}}
</div>
```

(3) product.art 模板编写

```
product.art X index.art product.css
2-6 > travel-mall > src > pages > index > components > product > product.art > div.product >
1 <div class="product">
2 <h2 class="product-title">精选折扣</h2>
3 <ul id="product-list" class="product-list">
4 <li class="product-item">
5 <a href="./details.html?id=1">
6 <p class="product-pic">
7 
8 </p>
9 <p class="product-name">私人定制 | 巴黎凡尔赛宫+枫丹白露宫 包车1日 ...</p>
10 <p class="product-price">3799元起</p>
11 </a>
12 </li>
13 > <li class="product-item"> ...
21 </li>
22 > <li class="product-item"> ...
30 </li>
31 > <li class="product-item"> ...
39 </li>
40 </ul>
41 </div>
```

(4) 编写样式

在 product.css 中为模板的各部件编写样式。

```
product.css X product.art index.art
c > pages > index > components > product > product.css
1 .product{
2   padding: 20px 20px 0 0;
3 }
4 .product-title{
5   width: 100%;
6   padding: 0 0 10px 20px;
7   color: #191919;
8   font-size: 16px;
9 }
10 .product-list{
11   display: flex;
12   flex-wrap: wrap;
13 }
14 .product-item{
15   width: 50%;
16   padding: 0 0 13px 20px;
17 }
18 .product-pic{
19   margin-bottom: 6px;
20 }
21 .product-img{
22   width: 100%;
23 }
```



```
.product-name{
  height: 40px;
  line-height: 20px;
  margin-bottom: 4px;
  color: #333;
  font-size: 14px;
  /* 多行文字省略 */
  overflow: hidden;
  text-overflow: ellipsis;
  display: -webkit-box;
  -webkit-line-clamp: 2;
  -webkit-box-orient: vertical;
  white-space: normal;
  word-wrap: break-word;
}
```

(5) Ajax 数据获取

url: <https://www.imoooc.com/api/mall-wepApp/index/product?icode=J351519BC12ED655F>

编写 product.js 利用 Ajax 从上述地址获取真实数据，并在 product.art 中输出。

```
index.art  JS product.js  index.css  product.art
12-6 > travel-mall > src > pages > index > components > product > JS product.js > onreadystatechange > productdata
1  import './product.css'
2
3  import render from './product.art'
4  const url = 'https://www.imoooc.com/api/mall-wepApp/index/product?icode=J351519BC12ED655F'
5  const xhr = new XMLHttpRequest()
6  console.log(xhr)
7  xhr.onreadystatechange = () => {
8    console.log(xhr.readyState);
9    if (xhr.readyState !== 4) return;
10   if ((xhr.status >= 200 && xhr.status <= 300) || xhr.status == 304) {
11     const data = JSON.parse(xhr.responseText)
12     // console.log(data.data)
13     // console.log(typeof data.data)
14     console.log(data.data[0].url)
15     document.getElementById('index-product').innerHTML = render({
16       productdata: data.data
17     })
18   }
19 }
20 }
21 xhr.open('get', url)
22 xhr.send()
```

注：上一版的有错误，如上图所示是 index-product，不是 index-nav
在 product.art 中输出数据。

```
product.css  product.art  JS product.js  index.art
2-6 > travel-mall > src > pages > index > components > product > product.art >
1  <div class="product">
2  <h2 class="product-title">精选折扣</h2>
3  <ul id="product-list" class="product-list">
4  |   {{each productdata}}
5  <li class="product-item">
6  |   <a href="./details.html?id={{value.id}}">
7  |     <p class="product-pic">
8  |       
9  |     </p>
10 |     <p class="product-name">{{value.title}}</p>
11 |     <p class="product-price">{{value.price}}元起</p>
12 |   </a>
13 | </li>
14 | {{/each}}
15 </ul>
16 </div>
```

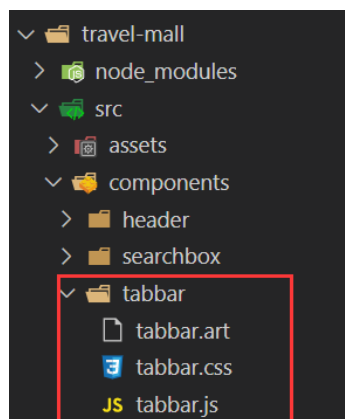
(6) 效果展示



3.7 底部标签栏组件创建

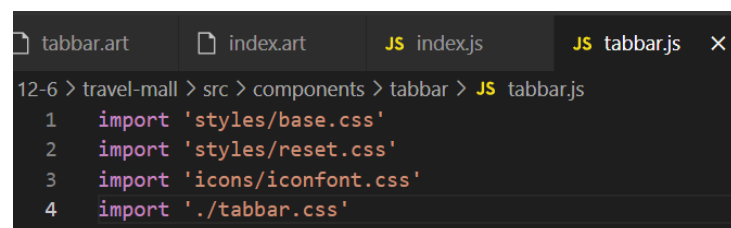
(1) 创建标签栏组件目录和文件

在 src/components/下创建 tabbar 目录及 tabbar.art、tabbar.css、tabbar.js 三个文件。

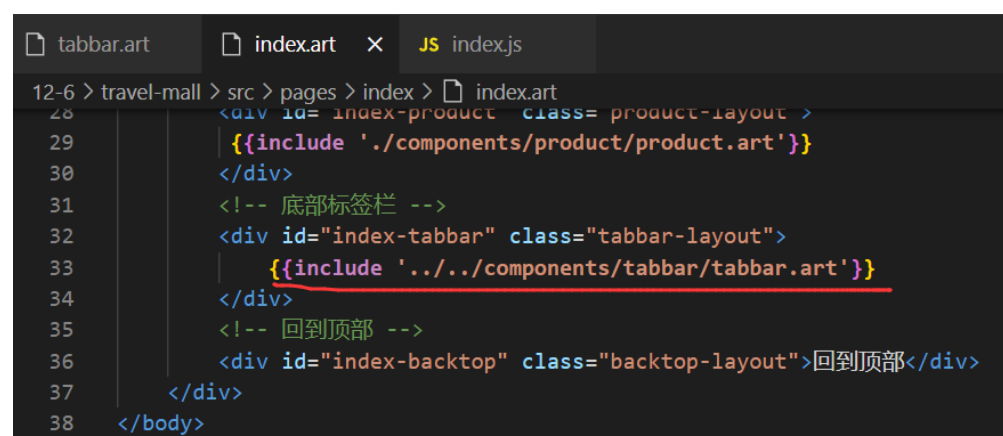


(2) 模板及样式引入

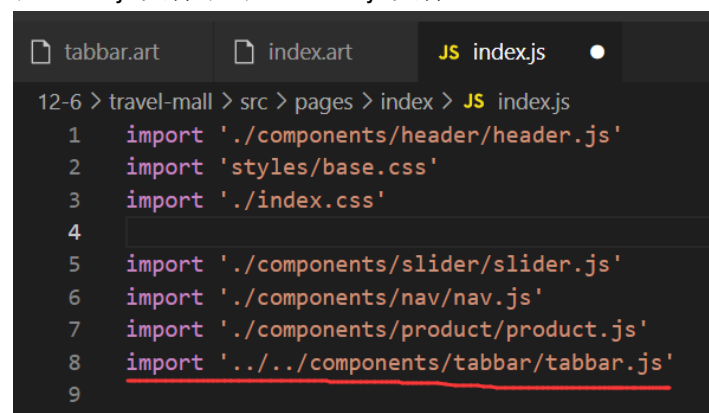
在 `tabbar.js` 中引入 `tabbar.css` 文件。



在 `index.art` 中引入 `tabbar.art` 文件。



在 `index.js` 文件中引入 `tabbar.js` 文件。



(3) tabbar.art 模板编写

```
tabbar.art × JS index.js
12-6 > travel-mall > src > components > tabbar > tabbar.art > div.tabbar > a.tabbar
1 <div class="tabbar">
2 <a href="./index.html" class="tabbar-item tabbar-item-active">
3   <i class="iconfont icon-home"></i>
4   <span>首页</span>
5 </a>
6 <a href="./index.html" class="tabbar-item ">
7   <i class="iconfont icon-dest"></i>
8   <span>目的地</span>
9 </a>
10 <a href="./index.html" class="tabbar-item ">
11   <i class="iconfont icon-personal"></i>
12   <span>我的</span>
13 </a>
14 </div>
```

(4) 编写样式

为 index.art 中的 tabbar-layout 编写样式，放入公共样式文件 src/assets/styles/layout.css 中。

```
index.css tabbar.art layout.css × JS index.js
12-6 > travel-mall > src > assets > styles > layout.css > .backtop-layout
1 .tabbar-layout{
2   position: fixed;
3   z-index: 1000;
4   width: 100%;
5   height: 48px;
6   right: 0px;
7   bottom: 0px;
8   background-color: rgb(254, 254, 254);
9   box-shadow: 0px 0px 3px 4px rgba(0,0,0, 0.15);
10 }
```

在 index.js 中引入上述样式。

```
index.css tabbar.art layout.css JS index.js ×
12-6 > travel-mall > src > pages > index > JS index.js
1 import './components/header/header.js'
2 import 'styles/layout.css'
3 import 'styles/base.css'
4 import './index.css'
```

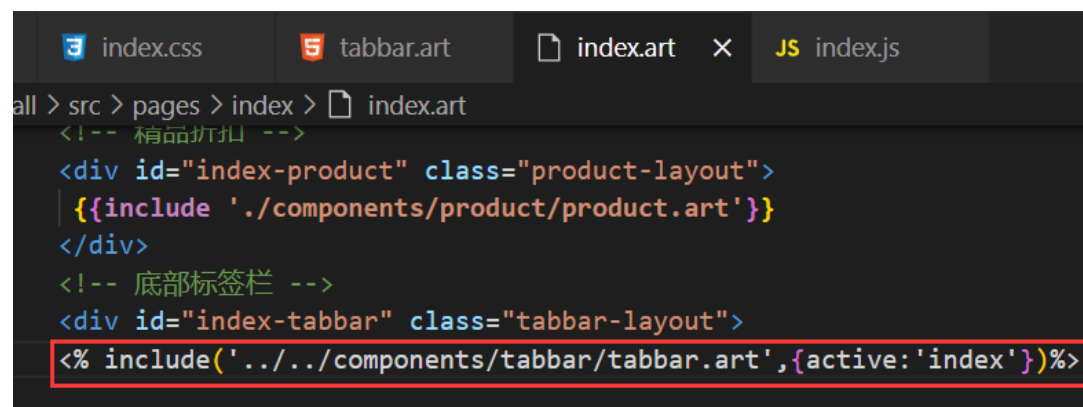
编写 tabbar.css 样式。

```
tabbar.css X index.css tabba
12-6 > travel-mall > src > components > tabbar >
1  .tabbar{
2      display: flex;
3      width: 100%;
4      height: 100%;
5      font-size: 12px;
6  }
7  .tabbar-item{
8      flex: 1;
9      display: flex;
10     justify-content: center;
11     align-items: center;
12     flex-direction: column;
13     color: #666;
14     font-size: 12px;
15 }
16 .tabbar-item-active{
17     color: #bf2a2f;
18 }
19 }
20 .tabbar-item .iconfont{
21     margin-bottom: 1px;
22     font-size: 20px;
23 }
```

(5) 解决底部标签点击变红的效果
修改 tabbar.art 文件。

```
tabbar.css index.css tabbar.art X index.art JS index.js
2-6 > travel-mall > src > components > tabbar > tabbar.art > div.tabbar
1  <div class="tabbar">
2  <a href="./index.html" class="tabbar-item" {{active==='index'? 'tabbar-item-active': ''}}>
3      <i class="iconfont icon-home"></i>
4      <span>首页</span>
5  </a>
6  <a href="./index.html" class="tabbar-item" {{active==='destination'? 'tabbar-item-active': ''}}>
7      <i class="iconfont icon-dest"></i>
8      <span>目的地</span>
9  </a>
10 <a href="./index.html" class="tabbar-item" {{active==='personal'? 'tabbar-item-active': ''}}>
11 <i class="iconfont icon-personal"></i>
12 <span>我的</span>
```

修改 index.art 文件。



```
all > src > pages > index > index.art
<!-- 精品折扣 -->
<div id="index-product" class="product-layout">
  {{include './components/product/product.art'}}
</div>
<!-- 底部标签栏 -->
<div id="index-tabbar" class="tabbar-layout">
  <% include('.././../components/tabbar/tabbar.art',{active:'index'})%>
</div>
```

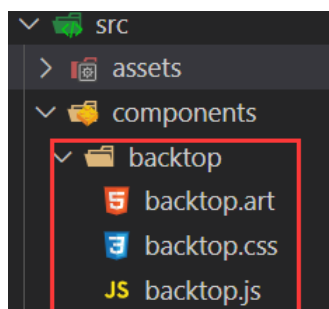
(6) 效果展示



2.8 回到顶部组件创建

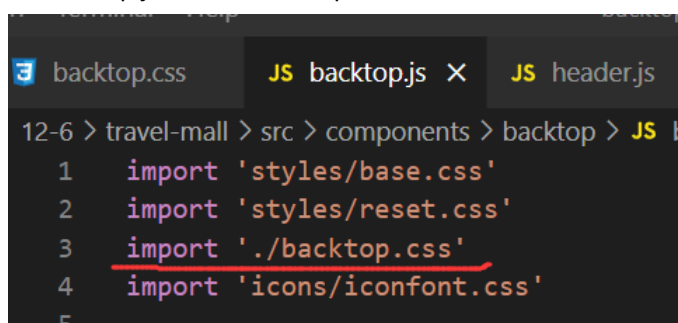
(1) 创建回到顶部组件目录和文件

在 src/components 目录下创建 backtop 目录及 backtop.css、backtop.art、backtop.js 三个文件。



(2) 模板及样式引入

在 backtop.js 中引入 backtop.css 文件。



在 index.js 中引入 backtop.art 文件。

```
backtop.css JS backtop.js JS header.js JS index.js X
12-6 > travel-mall > src > pages > index > JS index.js
1 import './components/header/header.js'
2 import 'styles/layout.css'
3 import 'styles/base.css'
4 import './index.css'
5
6 import './components/slider/slider.js'
7 import './components/nav/nav.js'
8 import './components/product/product.js'
9 import '../components/tabbar/tabbar.js'
10 import ../../components/backtop/backtop.js
11
```

在 index.art 文件中引入 backtop.art 文件。

```
backtop.css index.art X JS backtop.js backtop.art layout
12-6 > travel-mall > src > pages > index > index.art
34
35     </div>
36     <!-- 回到顶部 -->
37     <div id="index-backtop" class="backtop-layout">
38         {{include '../../components/backtop/backtop.art'}}
39     </div>
40 </div>
41 </body>
42
43 </html>
```

(3) backtop.art 模板编写

```
backtop.css index.art JS backtop.js backtop.art X
12-6 > travel-mall > src > components > backtop > backtop.art > a.backtop
1 <a href="javascript:;" class="backtop backtop-hidden">
2     <i class="iconfont icon-up"></i>
3
4 </a>
```

(4) 样式编写

在 layout.css 中编写以下样式，并在 index.js 中引入。

```
backtop.css index.art JS backtop.js backtop.art layout.css X
12-6 > travel-mall > src > assets > styles > layout.css > .backtop-layout
10 }
11 .backtop-layout {
12     position: fixed;
13     right: 10px;
14     bottom: 60px;
15     z-index: 1100;
16     /* background-color: green; */
17 }
```

在 backtop.css 中编写以下样式。

```
backtop.css × index.art JS backtop.js layout.css
12-6 > travel-mall > src > components > backtop > backtop.css > .backtop
1  .backtop {
2      display: flex;
3      align-items: center;
4      justify-content: center;
5      width: 45px;
6      height: 45px;
7      background-color: rgba(39, 195, 147, 0.8);
8      border-radius: 50%;
9  }
10 .backtop .iconfont{
11     color: #fff;
12     font-size: 30px;
13 }
14 .backtop-hidden{
15     display: none!important;
16 }
```

(5) 页面滚动隐藏或显示 backtop 组件

```
backtop.css index.art JS backtop.js × layout.css
12-6 > travel-mall > src > components > backtop > JS backtop.js > CHANGED_STATE
3  import './backtop.css'
4  import 'icons/iconfont.css'
5
6  //backtop组件功能, 页面向下滚动滚动一个页面后(window.innerHeight)出现backtop组件, 向上滚动又隐藏组件
7  const scrollContainer = document.getElementById('index-page')
8  const backtopEl = document.querySelector('.backtop')
9  // console.log(scrollContainer)
10 // console.log(headerEl)
11 const INIT_STATE = 'init'
12 const CHANGED_STATE = 'changed'
13 let state = INIT_STATE
14 // console.log(scrollContainer,headerEl)
15 scrollContainer.addEventListener('scroll',() => {
16     if (state=== CHANGED_STATE && scrollContainer.scrollTop <= window.innerHeight) {
17         state = CHANGED_STATE
18         backtopEl.classList.add('backtop-hidden')
19     } else if (state!== INIT_STATE && scrollContainer.scrollTop > window.innerHeight) {
20         state = INIT_STATE
21         backtopEl.classList.remove('backtop-hidden')
22     }
23 }, false)
```

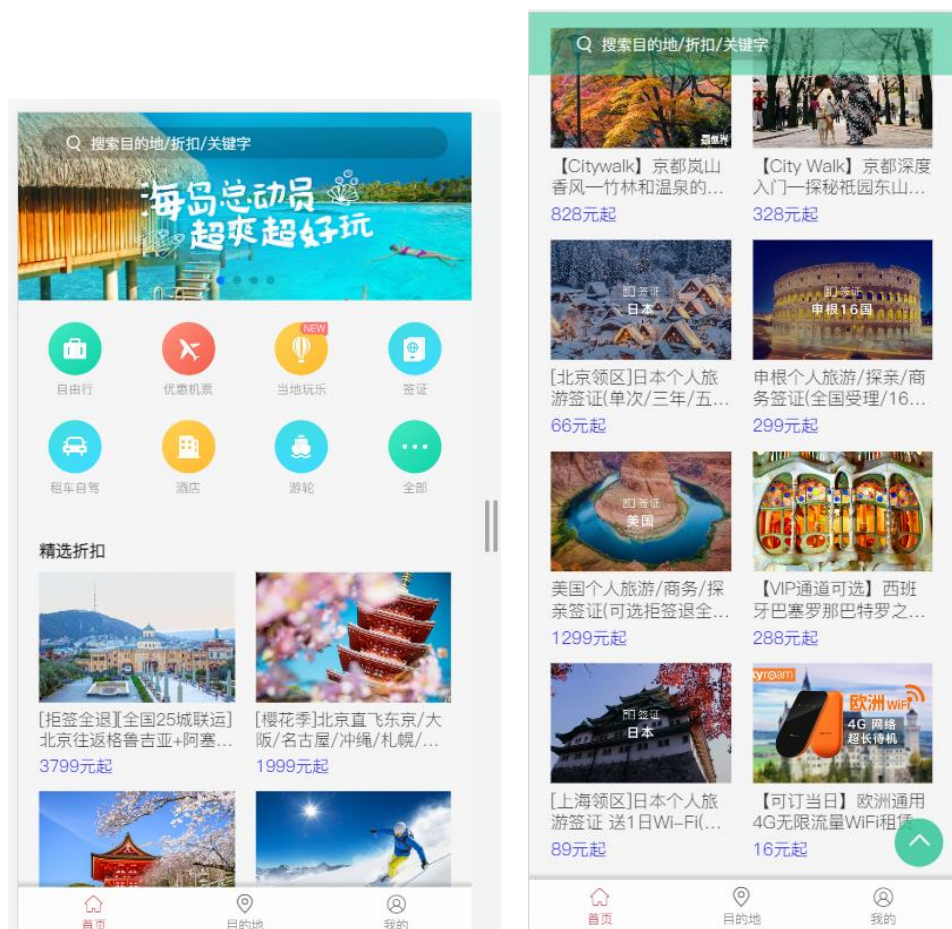
(6) 回到顶部功能


```
backtop.css  index.art  JS backtop.js ×  layout.css
12-6 > travel-mall > src > components > backtop > JS backtop.js > [CHANGED_STAT
22     }
23 }, false)
24 //点击回到顶部
25 backtopEl.addEventListener('click',()=>{
26     scrollContainer.scrollTo({
27         top:0,
28         left:0,
29         behavior:'smooth'
30     })
31
32 },false)
```

(7) 效果展示



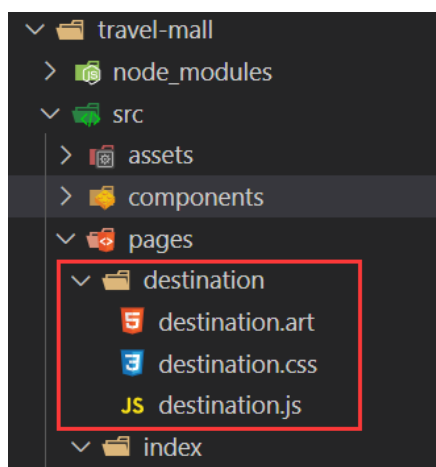
2.9 首页完整效果



4. 目的地页开发

4.1 目的地页目录与文件创建

在 `src/pages` 目录下创建 `destination` 文件夹，并在其中创建 `destination.art`、`destination.css`、`destination.js` 三个文件。



4.2 webpack.config.js 配置

(1) 增加 destination 页打包入口

```
module.exports = {  
  mode: 'development',  
  entry: {  
    index: './src/pages/index/index.js',  
    destination: './src/pages/destination/destination.js'  
  },  
}
```

(2) 增加模板输出

```
plugins: [  
  new HtmlWebpackPlugin(  
    {  
      template: './src/pages/index/index.art',  
      filename: 'index.html'  
    }  
  ),  
  new HtmlWebpackPlugin(  
    {  
      template: './src/pages/destination/destination.art',  
      filename: 'destination.html'  
    }  
  ),  
]
```

4.3 目的地页模板

(1) 目的地页模板编写

目的地页 destination.art 模板编写。目的地页包括头部、主体和标签栏三个部分，放在页面容器 id 为 destination-page 的 div 中。

```
12-6 > travel-mall > src > pages > destination > destination.art > html > body > div#destination  
1 <!DOCTYPE html>  
2 <html lang="en">  
3 <head>  
4   <meta charset="UTF-8">  
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">  
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">  
7   <title>慕云游商城-目的地</title>  
8 </head>  
9 <body>  
10   <div id="destination-page" class="destination-layout">  
11     <!-- 头部 -->  
12     <header class="header-layout"></header>  
13  
14     <!-- 主体 -->  
15     <div class="main-layout"></div>  
16  
17     <!-- 标签栏 -->  
18     <div class="tabbar-layout">  
19  
20     </div>  
21   </div>  
22  
23 </body>  
24 </html>
```

（2）目的地页样式引入

在 `destination.js` 中引入 `destination.css` 及公共样式。

```
destination.css destination.art JS destination.js X
12-6 > travel-mall > src > pages > destination > JS destination.js
1 import 'styles/layout.css'
2 import 'styles/base.css'
3 import './destination.css'
4
```

4.4 tabbar 组件引入

（1）tabbar 组件模板引入

```
destination.css destination.art X JS destination.js JS tabbar.js tabbar.css tabbar
2-6 > travel-mall > src > pages > destination > destination.art > html > body > div#destination-page.destination
17 <!-- 标签栏 -->
18 <div class="tabbar-layout">
19 | <% include('../components/tabbar/tabbar.art',{active:'destination'})%>
20 </div>
21 </div>
22
23 </body>
24 </html>
```

（2）tabbar 目的地按钮链接地址修改

```
tabbar.art X JS destination.js destination.css destination.art
12-6 > travel-mall > src > components > tabbar > tabbar.art
1 <div class="tabbar">
2 <a href="./index.html" class="tabbar-item {{active==='index'? 'tabbar-item-active':''}}">
3 <i class="iconfont icon-home"></i>
4 <span>首页</span>
5 </a>
6 <a href="./destination.html" class="tabbar-item {{active==='destination'? 'tabbar-item-active':''}}">
7 <i class="iconfont icon-dest"></i>
8 <span>目的地</span>
9 </a>
10 <a href="./index.html" class="tabbar-item {{active==='personal'? 'tabbar-item-active':''}}">
11 <i class="iconfont icon-personal"></i>
12 <span>我的</span>
```

（3）tabbar 组件 js 文件引入

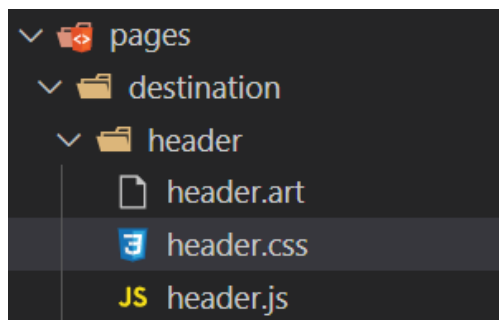
```
JS destination.js X destination.css destination.art
12-6 > travel-mall > src > pages > destination > JS destination.js
1 import './../components/tabbar/tabbar.js'
2 import 'styles/layout.css'
3 import 'styles/base.css'
4 import './destination.css'
5
```

（5）效果展示



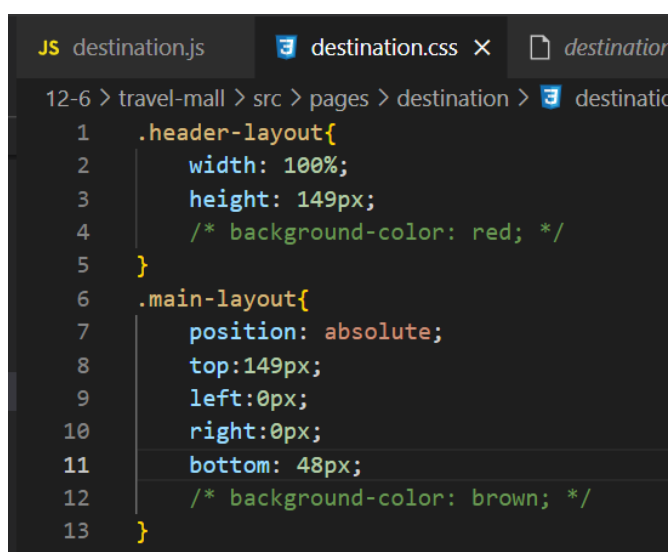
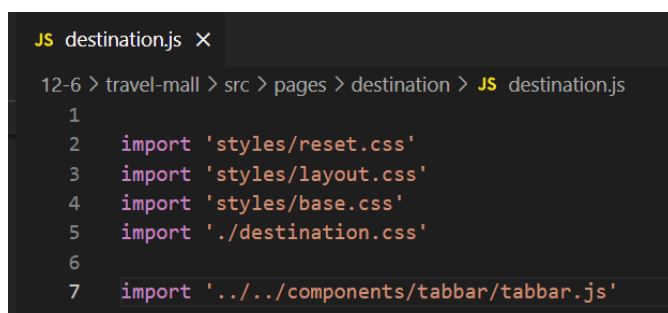
4.5 头部组件创建

(1) 头部组件目录及文件创建



(2) header 组件模板

(3)



```
header.art × JS header.js
12-6 > travel-mall > src > pages > destination > components > header > header.art > header.destination-header
1 <header class="destination-header">
2 <h1 class="header-title">目的地</h1>
3 <%include(' ../../../../components/searchbox/searchbox.art',{placeholder:'搜索你想要去的国家'})%>
4 </header>
```

```
header.art JS header.js ×
12-6 > travel-mall > src > pages > destination > components > header > JS header.js
1 import './header.css'
2 import 'components/searchbox'
```

目的地页数据：

<https://www.imooc.com/api/mall-wepApp/destination/content/1>

<https://www.imooc.com/api/mall-wepApp/destination/content/2>

<https://www.imooc.com/api/mall-wepApp/destination/content/3>

<https://www.imooc.com/api/mall-wepApp/destination/content/4>

<https://www.imooc.com/api/mall-wepApp/destination/content/5>

<https://www.imooc.com/api/mall-wepApp/destination/content/6>

<https://www.imooc.com/api/mall-wepApp/destination/content/7>