考生姓名 准考证号

**2019年下半年 Web前端开发高级 实操考试**

（考试时间14:00-16:30 共150分钟）

1. 本试卷共4道题，满分100分。

2. 请在指定位置或开发环境下作答。

# 试题一（30分）

阅读下列说明、效果图和代码，填写（1）至（15）代码。

**【说明】**

现有两个项目，项目一为文学信息库系统，项目名为literary，项目二为电子相册系统，项目名为album。

项目1：使用Vue.js框架编程，项目核心文件包括路由文件index.js，模板文件（公共头部模板header.vue、文学库首页模板homeView.vue和书籍详细信息页模板infoView.vue），页面需要的数据采用硬编码方式，存放在文学库首页模板的books对象中。

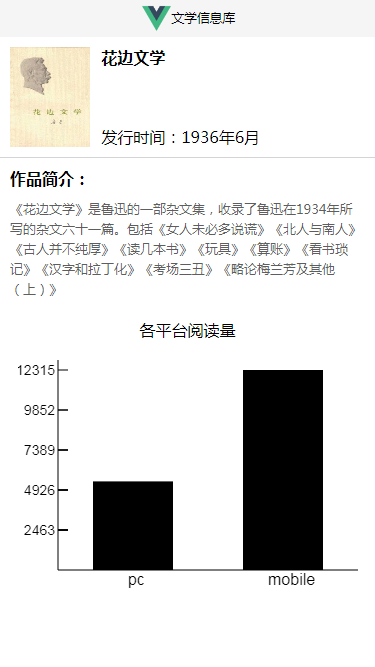
1. 文学库首页分为两部分，最上面是公共头部组件。下面是书籍列表，列表显示书籍封面图、书籍名称、发行时间，点击选中书籍使用vue-router路由方式跳转到infoView.vue模板
2. 书籍详细信息页分为三部分，最上面是公共头部组件。中间是书籍的详细详细信息，详细信息显示书籍封面图、书籍名称、书籍发行时间和作品简介。最下面是书籍PC和Mobile平台阅读量柱形统计图表，图表使用html5的Canvas绘制
3. 文学库首页和书籍详细信息页样式使用less语法编写

项目2：使用jQuery Mobile框架编程，项目核心文件包括电子相册首页index.html、电子相册CSS样式文件style.css、svg图片close.svg和电子相册JavaScript文件index.js。

1. 电子相册首页分为两部分，上面是页面的头部，中间有页面标题。下面是相册图片列表，列表使用jQuery Mobile的Grid网格布局，点击图片后使用弹出层将该图放大显示，弹出层右上角有一个关闭按钮，点击关闭按钮后弹出层关闭。

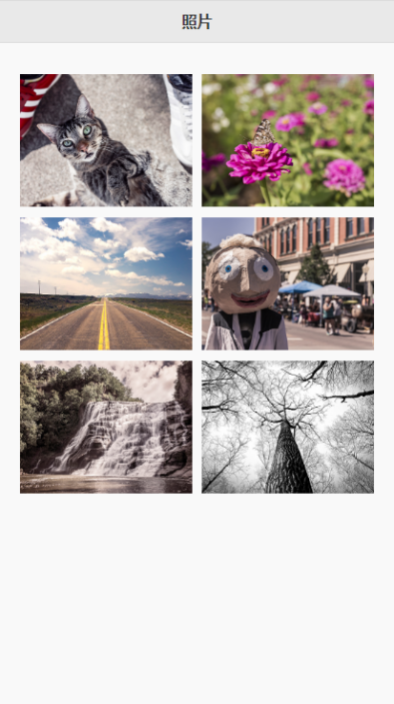
（2）SVG图片绘制了弹出层的关闭按钮图标，使用CSS的背景图引入到页面。

**【项目1效果图】**

文学库首页 书籍详细信息页

**【项目2效果图】**

电子相册首页 图片放大弹出层

**【项目1代码：homeView.vue】**

<template>

<div class="main">

<!--公共头部-->

<subHeader></subHeader>

<div class="wrapper">

<ul class="img\_box">

<!--根据books数据循环输出li标签-->

<li （1）>

<!--路由导航组件-->

<（2） v-bind:to="{ name: 'infoView', params: { bookInfo: item } }">

<!--将books内的图片地址绑定到<img>标签-->

<img （3）/>

<div class="item\_info">

<div class="info\_name">{{ item.name }}</div>

<div class="info\_time">发行时间：{{ item.time }}</div>

</div>

</（2）>

</li>

</ul>

</div>

</div>

</template>

<script>

import subHeader from '@/components/header';

export default {

data() {

return {

books: []

};

},

/\*注册公共头部组件\*/

（4）: { subHeader },

/\*在Vue实例创建完成后被立即调用\*/

（5）() {

/\*name书籍名称 content书籍简介 time发行时间 img封面图片 readCount阅读量\*/

this.books =[{name: '......',content:'......',time: '......',img: '......',readCount: {pc: ......, mobile: ......}, ......}];

}

};

</script>

**【项目2代码：infoView.vue】**

<script>

/\*绘制Vanvas表图\*/

draw() {

let drawElm = this.$refs.chart\_canvas;

/\*获取Vanvas的2d绘制环境\*/

let ctx = drawElm.（6）('2d');

let elmWidth = drawElm.offsetWidth, elmHeight = drawElm.offsetHeight;

//四周留白距离

let topGap = 10,btmGap = 20,leftGap = 40;let maxReadNum = 0;

drawX();

function drawX() {

//绘制X轴

ctx.beginPath();

/\*移动路径到指定坐标\*/

ctx.（7）(leftGap, elmHeight - btmGap);

ctx.lineTo(elmWidth, elmHeight - btmGap);

ctx.stroke();

//绘制X轴标签

ctx.beginPath();

ctx.font = '16px Arial';

/\*\*/

ctx.fillText('pc', 70 + leftGap, elmHeight - 5);

ctx.fillText('mobile', 210 + leftGap, elmHeight - 5);

ctx.stroke();

}

/\*此处省略绘制Y轴和柱形代码\*/

}

</script>

<style lang="less">

/\*定义变量，变量名为contentPadding\*/

（8）: 0.625rem;

/\*定义文本居中方法，方法名为textAlign\*/

（9）(@arguments) {

text-align: @arguments;

}

#book\_content {

padding: @contentPadding;

h4 {

margin: 0;

padding-bottom: @contentPadding;

}

#content\_txt {

font-size: 0.8125rem;

color: #5e5e5e;

line-height: 1.25rem;

}

}

#chart\_box {

padding: @contentPadding;

.textAlign(center)

.name {

margin-bottom: 1.25rem;

}

}

</style>

**【项目2代码：infoView.vue】**

<body>

<div data-role="page" id="page\_photo">

<div data-role="header">

<h2>照片</h2>

</div>

<div data-role="main">

<div class="ui-content">

<!--jQuery Mobile网格布局，每行两列-->

<ul class="ui-grid-a img\_list">

<li class="（10）">

<div class="img\_box"><img src="img/1.jpg" /></div>

</li>

<li class="（11）">

<div class="img\_box"><img src="img/2.jpg" /></div>

</li>

<!--此处省略剩余图片li列-->

</ul>

</div>

</div>

<div id="popup\_photo">

<a id="popup\_close" href="#"></a>

<div id="popup\_img\_box"><img class="popphoto" src="img/1.jpg"></div>

</div>

</div>

</body>

**【项目2代码：style.css】**

.img\_box {

/\*设置被转换元素的的基点位置\*/

（12）: top left;

}

.scale {

animation: imgScale 0.3s linear forwards;

}

@keyframes imgScale {

0% {

transform: scale(1);

opacity: 0;

}

30% {

opacity: 1;

}

100% {

opacity: 1;

/\*设置元素的放大倍率为两倍\*/

transform:（13）

}

}

**【项目2代码：close.svg】**

<?xml version="1.0" standalone="no"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="300" height="300" version="1.1"

xmlns="http://www.w3.org/2000/svg">

<!--14题SVG矩形形状 15题SVG形变-->

<（14） x="50" y="50" rx="10" ry="10" width="20" height="280" （15）="rotate(-45 70 50)" style="fill:white;" xmlns="http://www.w3.org/2000/svg" />

</svg>

# 试题二（20分）

阅读下列说明，根据要求，完成以下题目。

**【说明】**

使用Node.js和Express框架完成一个图书馆项目。

1. 实现一个图书列表接口，URI为http:127.0.0.1:3000/books/list，请求类型为GET，返回数据为JSON格式。
2. 实现JSONP跨域请求，请求图书列表数据并将数据展示在页面上。
3. 数据库选用MySQL，数据库名为library，数据库中有book表，表中字段包括图书id、图书名和作者，图书id为自增主键，如下表所示：

表3-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **图书表book** | | | | | |
| **数据元素名** | **内部名** | **值域** | **值义** | **类型/长度** | **备注** |
| 图书id | book\_id |  | 主键 | int/16 | 自增 |
| 图书名 | book\_name |  |  | varchar/255 |  |
| 作者 | book\_author |  |  | varchar/255 |  |

4、在library项目中核心文件包括服务器启动配置文件app.js，路由文件index.js，数据库配置文件db.js，自定义图书模块book.js。安装mysql和exspress依赖，目录结构如下图：

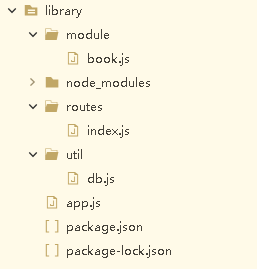


图3-1

5、在library\_client项目中，核心文件包括主页面index.html，js脚本文件index.js，css样式文件index.css，该项目用于发送JSONP请求和展示跨域请求数据，文件目录如下：

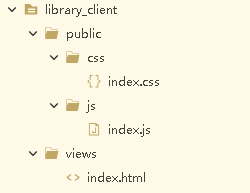


图3-2

**【library项目代码：app.js】**

const express = require('express');

const app = express();

const path = require('path');

//导入路由配置文件，使用相对路径

const indexRouter = （1）

//设置中间件

app.use((req, res, next) => {

console.log('请求路径:' + req.path);

console.log('请求类型:' + req.method);

console.log('当前时间:' + new Date());

//加载下一个中间件

（2）

});

//访问'/'路径，指派到对应的路由配置文件

（3）

//创建服务器

let http = require('http').（4）(app);

//监听3000端口

http.（5）(3000, function(){

console.log('listening on : 3000');

});

**【library项目代码：db.js】**

const mysql = require('mysql');

// 连接配置，创建连接池

const pool = mysql.（6）({

host: 'localhost',

port: 3306,

database: 'library',

user: 'root',

password: ''

});

// 连接方法

exports.query = (sql, callback) => {

//建立连接

pool.（7）((error, conn) => {

if(error) {

throw error;

return;

}

//执行查询方法

conn.（8）(sql, (error, results) => {

//释放连接

conn.（9）();

if(error) {

throw error;

}

//将数据返回

（10）

});

});

};

**【library项目代码：book.js】**

//导入数据库配置文件

const db = （11）

exports.bookList = (req, callback) => {

（12）

}

**【library项目代码：index.js】**

const express = require('express');

const router = express.Router();

const book = require('../module/book.js');

//图书列表接口

router.get('/books/list', (req, res) => {

（13）

})

//跨域请求接口

router.get('/jsonp', (req, res) => {

//获取到callback

let callback = req.url.split('?callback=')[1];

book.bookList(req, (data) => {

//将data数据字符串化

data = JSON.stringify(data);

res.end(（14）);

})

})

module.exports = router;

**【library\_client项目代码：index.html】**

<!DOCTYPE html>

<head>

<title>图书列表</title>

<link rel="stylesheet" type="text/css" href="../public/css/index.css" />

<script type="text/javascript" src="../public/js/index.js"></script>

</head>

<body>

<div>

<div class="area">

<h3>图书列表</h3>

<table border="1px" cellspacing="0px">

<tr>

<td>图书名</td>

<td>作者</td>

</tr>

</table>

</div>

</div>

</body>

</html>

**【library\_client项目代码：index.css】**

body{text-align: center;}

.area{float: left;width: 40%;height: 500px;margin: 0 30% 0 30%;padding-top: 10%;}

table{margin: 0 auto;}

td {width: 100px;height: 30px;}

**【library\_client项目代码：index.js】**

//动态创建script标签，并请求

function creatScript(src) {

var father = document.getElementsByTagName('body');

var script = document.createElement('script');

script.setAttribute('type', 'text/javascript');

script.src = src;

father[0].appendChild(script);

};

//在页面加载时，发送请求

window.onload = function() {

creatScript('http://127.0.0.1:3000/jsonp?（15）');

};

//回调的方法

function books(data) {

let dom = document.getElementsByTagName("table")[0];

creatTab(data, dom);

};

//将响应数据写入表格

function creatTab(book, dom) {

for (let i = 0; i < book.length; i++) {

var row = dom.insertRow(dom.rows.length);

let c1 = row.insertCell(0);

c1.innerHTML = book[i].book\_name;

let c2 = row.insertCell(1);

c2.innerHTML = book[i].book\_author;

}

}

**【问题】（20分）**

1. 基于Express框架，在app.js中填写（1）-（5）空。（每空1分）
2. 基于MySQL模块，在db.js中填写（6）-（10）空。（每空1分）
3. 在book.js中，填写（11）空，导入数据库配置文件。（1分）
4. 分析业务逻辑，在book.js中，完成代码块（12），设计SQL语句，查询book表中的所有数据，调用db.query()方法，传入SQL语句，将数据放在callback中返回。（3分）
5. 在图书列表接口中，完成代码块（13），调用book中的bookList方法，将req作为参数传递，将返回数据data以JSON格式返回，状态码state设置为200。（2分）
6. 在跨域请求接口中，填写（14）空，返回正确内容。（2分）
7. 基于JSONP实现跨域请求，填写（15）空（2分）。

# 试题三（30分）

阅读下列说明、效果图和代码，进行动态网页开发，回答问题1至问题6。

**【说明】**

该程序为一个商品展示系统，使用前后端分离模式，后端使用Node.js+Express框架，项目名为listProduct\_server，前端使用Vue.js框架，使用Vue CLI构建项目,项目名为listProduct\_client。其中后台采用MVC架构，核心文件包括服务器启动配置文件app.js，dao层文件product.dao.js，模型文件product.js，控制器文件product.controller.js，业务逻辑文件product.service.js，工具类文件夹util下的数据库配置文件db.js和自定义响应模块文件response.js。前端核心文件包括配置文件config/index.js，商品列表组件ProductList.vue，使用MySQL数据库存放商品信息。Vue.js框架采用Axios组件实现Ajax请求，项目API遵循RESTful API标准。

商品列表页面包括用户输入部分，商品列表部分。用户输入部分显示input输入框和搜索按钮，商品列表部分显示商品图片，商品名称，商品价格，删除按钮，

用户在input输入框中输入商品名称，点击“搜索”按钮，商品列表部分显示符合搜索条件的商品。点击“删除”按钮，删除对应到商品。

**【效果图】**

**商品列表页面：对应模板文件ProductList.vue**

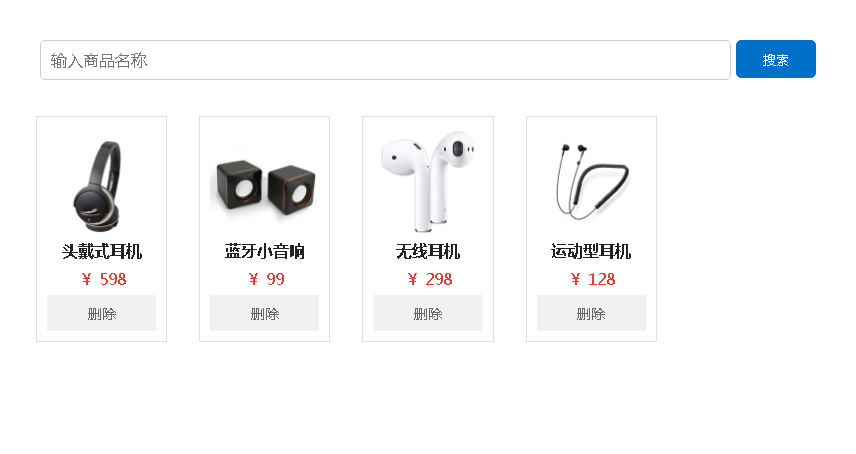


图4-1

**【listProduct\_server项目代码：app.js】**

const express = require('express');

const app = express();

const bodyParser = require('body-parser');

const product = require('./routes/product.controller.js');

const http = require('http');

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: false}));

app.use('/', product);

http.（6）.listen(3000, function() {

console.log('listening on : 3000');

});

app.use(function(err, req, res, next) {

console.error(err.stack)

res.status(500).send('Something broke!')

})

**【listProduct\_server项目代码：product.js】**

class product {

constructor(id, name, price, img){

this.id = id;

this.name = name;

this.price = price;

this.img = img;

}

get id(){

return this.\_id;

}

set id(id){

this.\_id = id;

}

//仅展示部分get/set方法

};

module.exports = product;

**【listProduct\_server项目代码：product.controller.js】**

const express = require('express');

const router = express.Router();

const productService = require('../service/product.service.js');

const back = require('../util/response.js');

router.get('/products/list', (req, res) => {

productService.listProduct(req, (data) => {

back.jsonWrite(res, data);

})

});

router.get('/products/product', (req, res) => {

productService.selectProduct(req, (data) => {

back.jsonWrite(res, data);

})

});

router.delete('/products/product', (req, res) => {

productService.deleteProduct(req, (data) => {

back.jsonWrite(res, data);

})

});module.exports = router;

**【listProduct\_server项目代码：product.service.js】**

const productDao = require('../dao/product.dao.js');

const product = require('../model/product.js');

exports.listProduct = (req, callback\_controller) => {

productDao.listProduct((data) => {

callback\_controller(data);

})

}

exports.selectProduct = (req, callback\_controller) => {

let model = new product(null, req.query.name, null, null);

productDao.selectProduct(model, (data) => {

callback\_controller(data);

})

}

exports.deleteProduct = (req,callback\_controller) => {

let model = new product(req.body["id"], null, null, null);

productDao.deleteProduct(model, (data) => {

callback\_controller(data);

})

}

**【listProduct\_client项目配置文件：config/index.js】**

'use strict'

const path = require('path')

module.exports = {

dev: {

...

proxyTable: {

'/': {

// 配置跨域地址http://localhost:3000

（9）: 'http://localhost:3000',

// 设置是否跨域

（10）: true,

pathRewrite: {

'': ''

}

},

},

...

}

**【listProduct\_client项目项目代码：ProductList.vue】**

<template>

<div class="wrapper">

<!-- 搜索框 -->

<div class="product-search">

<input type="text" placeholder="输入商品名称" v-model="searchText">

<button v-on:click="handleSearch">搜索</button>

</div>

<!-- 商品列表 -->

<!--数据循环输出商品列表-->

<div class="product" （11）>

<a href="javascript:;" class="product-mian">

<img v-bind:src="item.img">

<h4>{{ item.name }}</h4>

<div class="product-prices">￥ {{ item.price }}</div>

<div class="product-delete" v-on:click="handleDel(item.id)">删除</div>

<!-- <div class="product-edit">修改商品</div> -->

</a>

</div>

</div>

</template>

<script>

// 引入请求方法

import axios from 'axios'

export default {

data() {

return {

info: null, // 存放商品数据

searchText: '' // 存放用户搜索数据

}

},

created() {

// 请求列表数据，初始化

this.getList();

},

methods:{

getList() {

// 请求全部商品列表数据

axios.get('/products/list')

.then(res => {

if (res.data.state == 200){

this.info = res.data.data;

}

})

.catch(error => {

console.log(error)

})

},

handleSearch() {

//通过url传参方式请求搜索商品的数据

axios.（12）('/products/product,{

（13）: {

name: this.searchText

}

})

.then(res => {

if (res.data.state == 200){

this.info = res.data.data;

}

})

.catch(error => {

console.log(error)

})

},

handleDel(id){

// 删除请求，将json对象发送到后台

axios.（14）('/products/product',{

（15）: {

id: id

}

})

.then(res => {

if (res.data.state == 200){

this.handleSearch();

}

})

.catch(error => {

console.log(error)

})

}

}

}

</script>

**【问题1】请根据MVC模型，回答下列问题。（每空2分）**

1. 在本项目中，请分析product.js中的代码回答product.js在MVC模型Model、View和Controller中属于（1）层。
2. 在本项目中，请分析product.controller.js中的代码回答product.controller.js在MVC模型Model、View和Controller中属于（2）层。

**【问题2】分析项目路由，回答下列问题。（每空2分）**

在浏览器中输入网址，访问商品列表页面，请分析路由，回答下列问题：

1. 在请求商品列表时，使用GET请求http://127.0.0.1:3000/products/list，以下哪个路由不能被调用（3）

const express = require('express');

const app = express();

1. app.get('/', function (req, res) {})；
2. app.use('/', function (req, res) {})；
3. app.get('/products/list', function (req, res) {})；
4. app.all('/products/list', function (req, res) {})；
5. 在定义URI时，以下哪个URI符合RESTful API规范（4）
6. GET /getProducts
7. DELETE /deleteProduct/1
8. GET /get\_product?id=3
9. GET /products/product

**【问题3】基于Node.js和Express框架，回答下列问题。（每空2分）**

1. 在app.js文件中，下列哪种中间件没有被用到（5）
2. 应用层中间件
3. 路由器级中间件
4. 错误处理中间件
5. 第三方中间件

2、在app.js中，使用http模块创建HTTP服务服务器，在app.js中填写（6）空。

**【问题4】分析模型文件，回答下列问题。（每空2分）**

在product.js文件中，定义了product类，在类中定义了constructor()方法。请分析product类，回答下列问题：

1. 在product.service.js中执行let model = new product(null, req.query.name, null, null);代码时，在product类中依次执行了什么方法（7）
2. set ->constructor
3. constructor->set
4. constructor->set->constructor
5. set
6. 如果在product.service.js中执行let model = new product(null, req.query.name, null, null);代码后，再加一行代码let name = model.name;，那么这行代码在product类中执行了什么方法（8）
7. constructor
8. set
9. get
10. constructor->get

**【问题5】分析配置文件index.js，回答下列问题，填写（9）至（10）（每空2分）。**

在listProduct\_client项目中Ajax请求需要跨域，在config配置文件中的index.js中配置proxy代理来实现Ajax请求跨域。分析config配置文件中的index.js，在proxyTable中配置正确的proxy代理，项目中所有的Ajax请求实现跨域。填写（9）至（10）空。

**【问题6】分析模板文件ProductList.vue，回答下列问题，填写（11）至（15）（每空2分）。**

listProduct项目的后端和前端运行在同一个服务器下，分析ProductList.vue代码，实现用户在搜索商品时，向后台请求数据，渲染页面。点击删除按钮，删除对应的商品。填写（11）至（15）空。

# 试题四（20分）

阅读下列说明，效果图和代码，进行网站性能优化，回答问题1至问题4。

**【说明】**

网站性能对于用户体验有及其重要的影响，网站的功能比以往更丰富，对网站的要求越来越高，因此需要对网站进行性能优化。

项目1：该项目是一个图片相册，项目名称为photoShow，包含主页index.html，图片，其中，主页面index.html显示相册图片和缩略图。主页分为：页头，内容，页脚。

项目2：该项目是一个基于Web的图片资源网站，项目名称为phtot\_cilent，后端使用Node.js的Express框架编写，该项目核心文件pictureServer.js将图片在前端输出。

项目3：把listProduct\_client项目打包，部署到listProduct\_server项目中。

（1）如下图，左图为listProduct\_client项目打包前项目目录结构，右图为listProduct\_client项目打包后的目录结构，dist为打包后的输出目录，打包后的文件在dist目录下面。如图1-1所示。

图1-1

（2）listProduct\_server项目目录结构如下图1-2所示：



图1-2

**【项目1效果图index.html】**

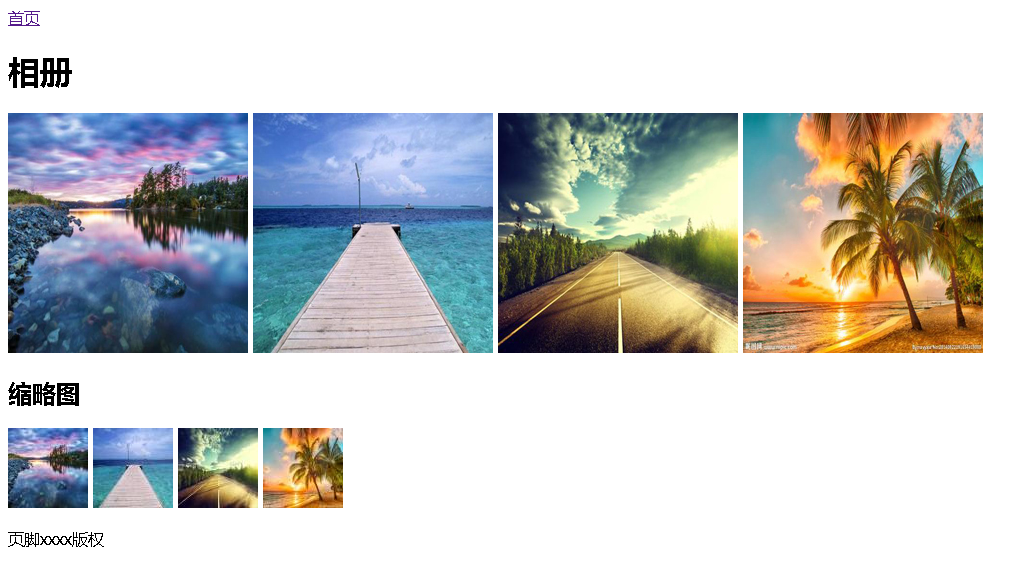


图1-3

**【项目1代码：主页index.html】**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>图片相册</title>

<style>

.normal-img{

width: 240px;

height: 240px;

background-color:#6495ed;

background-image:url('img/1.jpg');

background-repeat:no-repeat;

background-position:right top;

display: inline-block;

}

.img2{

background-image:url('img/2.jpg');

}

.img3{

background-image:url('img/3.jpg');

}

.img4{

background-image:url('img/4.jpg');

}

</style>

</head>

<body>

<!-- 头部 -->

<div>

<!-- 导航 -->

<div>

<a href="">首页</a>

</div>

</div>

<!-- 区块1 -->

<div>

<h1>相册</h1>

<div class="normal-img img1"></div>

<div class="normal-img img2"></div>

<div class="normal-img img3"></div>

<div class="normal-img img4"></div>

</div>

<!-- 区块2 -->

<div>

<h2>缩略图</h2>

<img src="img/1.jpg" class="image">

<img src="img/2.jpg" class="image">

<img src="img/3.jpg" class="image" >

<img src="img/4.jpg" class="image" >

</div>

<!-- 底部 -->

<div>

<p>xxxx版权</p>

</div>

</body>

</html>

**【项目2代码：pictureServer.js】**

let express = require("express");

let router = express.Router();

router.get("/list", (req, res) => {

let results = ['public/img/1.jpg', 'public/img/2.jpg', 'public/img/3.jpg', 'public/img/4.jpg'];

res.json({ // 返回数据

state: "200", // 状态码

msg: "操作成功", // 提示消息

data: results

});

});

module.exports = router;**【问题1】分析HTML代码，回答下列问题，填写（1）至（5）。（每空1分）**

1、对网页进行代码结构优化，保证可读性，选择以下哪种方式对HTML进行结构优化（1）

A. 将div改为语义化标签

B. 将table改为div布局

C. 删除多余空格

D. HTML网页GZIP压缩

2、使用以上选择的方式，在（2）到（5）处填写正确内容;

<!-- 头部 -->

<（2）>

<!-- 导航 -->

<（3）>

<a href="">首页</a>

</（3）>

</（2）>

<!-- 文档中的独立部分，某个区段1 -->

<（4）>

<h1>相册</h1>

...

</（4）>

<!-- 文档中的独立部分，某个区段2 -->

<（4）>

<h2>缩略图</h2>

...

</（4）>

<!-- 底部 -->

<（5）>

<p>xxxx版权</p>

</（5）>

**【问题2】分析CSS样式，回答下列问题，填写（6）至（7）。（每空1分）**

1、在同等条件下缩短浏览器加载CSS代码时间，减少代码量。使用以下哪种方式对index.html中的CSS代码进行优化（6）？

A. 提取相同CSS代码

B. 条例化CSS代码;

C. 缩写CSS代码

D. 移除无匹配的CSS代码

2、使用以上选择的方式，下列四种写法正确的是（7）？

1. .img2,.img3,.img4{width:240px;height:240px;background-color:#6495ed;background-image:url('img/1.jpg');background-repeat:no-repeat;background-position:right top;display: inline-block;}
2. .normal-im,.img2,.img3,.img4{width:240px;height:240px;background-color:#6495ed;background-image:url('img/1.jpg');background-repeat:no-repeat;background-position:right top;display: inline-block;}
3. .normal-img{width:240px;height:240px;background:#6495ed url('img/1.jpg') right top no-repeat ;display: inline-block;}
4. .normal-img{width:240px;height:240px;background:#6495ed url('img/1.jpg') no-repeat right top;display: inline-block;}

**【问题3】分析JavaSscript代码，回答下列问题，填写（8）至（11）。（选择题1分，填空题每空2分）**

1、在做上传图片操作时，由于一些图片非常大，我们在上传的时候会占用大量的网络资源和本地资源，这种情况下，为了减少文件大小，选择以下哪种方式优化图片（8）

1. 图片懒加载

B. 图片压缩

C. 图片缓存

D. 使用CDN

2、使用以上选择的方式，完成图片资源优化，在（9）到（11）处填写正确内容。

<script>

function dealImage(path, obj, callback) {

// 创建图像对象

var img =（9）;

img.src = path;

img.onload = function() {

var that = this;

var w = that.width;

var h = that.height;

w = obj.width || w;

h = obj.height || h;

//生成canvas

var canvas = document.createElement('canvas'),

ctx = canvas.getContext('2d');

canvas.width = w;

canvas.height = h;

// 在画布上绘制图像

ctx.（10）(that, 0, 0, w, h);

// 默认图片质量为0.7

var quality = 0.7;

if (obj.quality && obj.quality <= 1 && obj.quality > 0) {

quality = obj.quality;

}

// 转换图片为base64

var base64 = canvas.（11）('image/jpeg', quality);

// 回调函数返回base64的值

callback(base64);

}

}

</script>

**【问题4】前端资源优化，回答下列问题，填写（12）至（13）。（选择题1分，填空题每空2分）**

1、如果相同图片多次使用，客户端每次都要请求服务器，服务器存在较大压力，客户端每次请求完都要进行页面渲染，用户体验较差，这种情况下，使用什么方式进行优化（12）？

A. 使用不同的图片格式

B. 图片压缩

C. CSS雪碧图

D. HTTP缓存

2、在项目2的pictureSever.js代码中，使用以上方式，在（13）处填写正确内容

**【问题5】打包项目，回答下列问题，填写（14）至（15）。（选择题1分，填空题每空2分）**

1、现在很多网页有着丰富的应用，它们拥有着复杂的JavaScript代码和一大堆依赖包。使用（14），对非JS资源转换成JS，如把一个CSS文件转换成“创建一个style标签并把它插入document”的脚本、把图片转换成一个图片地址的JS变量或base64编码等。

A. Bundle

B. Webpack;

C. Grunt

D. Gulp

2、使用以上的打包方式，打包商品列表项目，输入（15）命令。打包项目。