Web前端开发高级模拟测试（六）

66分

共54道题 总分：200分 形考总分：0分

形考：0分

单选题 多选题 判断题 客观填空题

一、单选题 共30题，60分

1 2分

在表单中，使用什么指令进行数据双向绑定（）

A v-bind

B v-model

C v-modle

D {{}}

作答结果：错误

打开解析

2 2分

在Less中，以下选项哪一个代表上一层选择器的

A *:

B $:

C @:

D &:

作答结果：正确

打开解析

3 2分

在windows中使用命令行编译style.less并输出css文件，下列命令正确的是。（）

A less style.less style.css

B less style.less

C lessc style.less style.css

D lessc style.css

作答结果：错误

单选题（30题，60分）

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

多选题（15题，30分）

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

判断题（5题，10分）

1 2 3 4 5

客观填空题（4题，100分）

1 2 3 4

正确 错误 批阅 未做

打开解析

4

2分

在Vue.js的MVVM模式中，第二个V指的是什么。（）

A ViewModel

B View

C Views

D ViewView

作答结果：正确

打开解析

5

2分

在Vue.js 为 v-on 提供了事件修饰符，提交事件不再重载页面的事件修饰符是哪个。（）

A .stop

B .prevent

C .capture

D .self

作答结果：正确

打开解析

6

2分

以下哪种属性可以设置Canvas里绘制笔触的颜色。（）

A fillStyle

B strokeStyle

C shadowColor

D shadowBlur

作答结果：正确

打开解析

7

2分

下列是jQuery Mobile中Grid网格布局容器正确的使用方法是。（）

A data-role="ui-grid-a"

B class="ui-block-a"

C class="ui-grid-a"

D data-role="ui-block-a"

作答结果：正确

打开解析

8

2分

以下哪个标签用于定义SVG的路径。（）

A <path>

B <polyline>

C <rect>

D <ellipse>

作答结果：正确

打开解析

9

2分

SVG是用什么来描述二维图形和绘图程序的。（）

A HTML

B CSS

C TXT

D XML

作答结果：正确

打开解析

10

2分

以下哪个URI符合RESTful API规范。（）

A http://127.0.0.1:3000/getbook

B http://127.0.0.1:3000/get_book/book/1

C http://127.0.0.1:3000/books/book

D http://127.0.0.1:3000/BOOK?id=1

作答结果：正确

打开解析

11

2分

下列全局注册组件正确的是。（）

A Vue.methods('component-a', { /* ... */ })

B Vue.props('component-a', { /* ... */ })

C Vue.components('component-a', { /* ... */ })

D Vue.component('component-a', { /* ... */ })

作答结果：正确

打开解析

12

2分

关于JSONP的说法错误的是。（）

A 数据可以使用JSON格式

B 可以实现跨域通信

C 使用GET请求

D 不能解决不同域名的跨域问题

作答结果：正确

打开解析

13

2分

小图标使用图片整合技术，制成雪碧图，主要是为了。（）

A 减少请求次数

B 美化图片

C HTTP缓存

D 图片懒加载

作答结果：正确

打开解析

14

2分

在webpack 配置文件中哪个配置项可以设置Vue组件的解析规则。（）

A mode

B module

C mod

D modular

作答结果：正确

打开解析

15

2分

在webpack 配置文件中，下列哪个是正确的入口配置项。（）

A entry

B enter

C input

D join

作答结果：正确

打开解析

16

2分

下列对于get/set方法，说法错误的是。（）

A 在类实例化的时候调用set方法

B 在类实例化后调用类的属性的时会调用get方法

C getter和setter必须同级出现

D getter和setter必须同时出现

作答结果：错误

打开解析

17

2分

下列生命周期说法错误的是。（）

A 在created阶段，vue实例的数据对象data有了，el还没有

B 在created阶段，vue实例的数据对象data和el都有了

C 在beforeMount阶段，vue实例的$el和data都初始化了

D 在mounted阶段，vue实例挂载完成，data.message成功渲染

作答结果：正确

打开解析

18

2分

关于Express框架的中间件，下列描述错误的是。（）

A 中间件可以执行任何代码

B 错误处理中间件可以自定义参数数量

C 如果当前的中间件功能没有结束请求-响应周期，则必须调用next()将控制权传递给下一个中间件功能，否则，该请求将被挂起

D 使用第三方中间件时，需要先安装对应的模块

作答结果：错误

打开解析

19

2分

为了优化HTML结构，文档的页眉一般使用哪个语义化标签。（）

A <head>

B <nav>

C <section>

D <header>

作答结果：正确

打开解析

20

2分

在使用Express托管静态文件时，下列说法错误的是。（）

A 可以多次调用express.static中间件函数来设置多个静态资源目录

B 可以为静态目录指定安装路径来创建虚拟路径前缀

C 使用express.static时最好使用绝对路径

D 在设置app.use(express.static('public'))后，可以通过访问http://localhost:3000/public/css/style.css来访问public文件夹下的文件

作答结果：错误

打开解析

21

2分

实现一个元素的旋转效果可以使用CSS3里的什么方法。（）

A translateY

B scaleY

C rotate

D skew

作答结果：正确

打开解析

22

2分

关于Node.js的异步编程，下列说法错误的是。（）

A Node.js 是单线程的

B Node.js 异步编程依托于回调来实现

C 利用回调函数可以使代码无阻塞执行，所以连续的回调函数的嵌套是有利的

D Node.js通常会将异常作为回调函数的第一个实参传回

作答结果：正确

打开解析

23

调用Canvas对象的哪个方法来获取绘图环境。（）

2分

A getCanvas

B getContent

C getContext

D getClient

作答结果：正确

打开解析

24

关于events模块，emitter为events.EventEmitter()的实例化对象，下列描述错误的是。（）

2分

A events.EventEmitter() 的核心就是事件触发与事件监听器功能的封装

B 使用emitter.on()对同一事件只能注册一个监听器

C 使用emitter.once()注册的监听器在触发后会立即解除

D 可以使用emitter.removeAllListeners()移除指定事件的所有监听器

作答结果：错误

打开解析

25

以下哪个不是 Node.js 的核心模块。（）

2分

A http

B request

C path

D fs

作答结果：正确

打开解析

26

下列对于Class类中的constructor()方法，说法错误的是。（）

2分

A constructor()方法是类的默认方法，创建类的实例化对象时被调用

B 在一个类中可以有多个名为"constructor"的特殊方法

C 如果没有显式指定构造方法，则会添加默认的constructor方法

D 在一个构造方法中可以使用super关键字来调用一个父类的构造方法

作答结果：正确

打开解析

27

2分

下列对状态码表述错误的是。（）

A 客户端发送请求后，服务器可以只返回数据而不返回状态码

B 200状态码表示操作成功

C 4XX状态码表示客户端错误

D 5XX状态码表示服务器错误

作答结果：正确

打开解析

28

2分

在jQuery Mobile弹出框中，在弹出框内的元素上使用以下哪一个正确的属性，点击该元素弹出框会关闭。（）

A data-rel="back"

B data-rel="close"

C data-rel="home"

D data-rel="off"

作答结果：正确

打开解析

29

2分

在使用npm命令安装依赖包时，如果需要将该依赖添加进package.json中，但是只在开发和测试环境中使用，则应该使用以下哪种命令。（）

A npm install moduleName

B npm install -g moduleName

C npm install -save moduleName

D npm install -save-dev moduleName

作答结果：错误

打开解析

30

2分

设置HTTP缓存时间为20秒，下列正确的是。（）

A Cache-Control:max-age=20

B Cache-Control:max-time=20

C Cache-Control:max-date=20

D Cache-Control:max=20

作答结果：正确

打开解析

二、多选题 共15题，30分

1 2分

下列对于服务器响应的说法，正确的有（）。

A 服务器返回的数据格式，可以是纯文本

B 可以通过设置HTTP头的Content-Type来改变当服务器要返回的数据格式

C 在express中可以通过res.json来将JSON格式数据返回

D 在发生错误时，也可以返回200状态码，把错误信息放在数据体里面

作答结果：错误

打开解析

2 2分

关于jQuery Mobile中page页的说法正确的是（）。

A 在一个html文件中可以有多个page

B 在屏幕中只会显示一个page

C 在html文件中page可有可无

D 可以使用main替代page

作答结果：正确

打开解析

3 2分

下列对于数组创建时Array.of()的用法，正确的有（）。

A let arr = Array.of(1, 2, 3, 4);

B let arr = Array.of(1, '2', true, null);

C let arr = Array.of();

D let arr = Array.of(userName:'作者1');

作答结果：错误

打开解析

4

下列对SVG矢量图形说法正确的是（）。

2分

A SVG图像放大后图像质量不会有损失

B SVG使用XML格式定义图像

C SVG是基于css来绘制的

D SVG是万维网联盟的标准

作答结果：正确

打开解析

5

2分

对于Web模块，下列说法正确的有（）。

A 搭建HTTP服务器需要引入http模块

B response.writeHead()和response.setHeader()作用相同

C response.end(data)方法也可以将data发送给客户端

D 在调用response.write()方法前必须调用response.writeHead()方法

作答结果：正确

打开解析

6

2分

下列对Canvas说法错误的是（）。

A 可以绘制矢量图

B 可以使用JavaScript编程绘图过程

C Canvas背景默认是透明的

D Canvas默认坐标起点在正中心

作答结果：错误

打开解析

7

2分

对于Express框架的路由，下列描述正确的有（）。

A Express支持与所有HTTP请求方法相对应的方法

B 路由路径可以是字符串，字符串模式或正则表达式

C 路径参数的名称可以由（[A-Z，a-z，0-9，_，?，$]）组成

D 一条路由只可以被一个回调函数处理

作答结果：错误

打开解析

8

2分

在webpack 配置文件中配置 output 属性，下列哪些属于output的配置项是（）。

A path

B staticPath

C filename

D file

作答结果：错误

打开解析

9

2分

关于Express框架集成MySQL数据库，下列说法正确的有（）。

A Express框架可以使用 Node.js 支持的所有数据库

B 在调用end()方法后可以继续调用query()方法进行数据库操作

C 在使用连接池时，每次查询后都需要调用release()方法释放连接

D 如果需要多次执行数据库操作，则使用连接池可以减少服务器内存资源的占用

作答结果：错误

打开解析

10

2分

关于Less中变量的说法，下列错误的是（）。

A 变量是没有作用域的

B 变量可以使用运算符号进行计算

C 使用属性变量需要大括号包裹

D 变量声明后不能修改

作答结果：错误

打开解析

11

2分

下列哪些方式可以实现路由跳转（）。

A router-link

B router-view

C this.$route.push({ path:'/user'})

D this.$router.push({ path:'/user'})

作答结果：正确

打开解析

12

2分

下列哪些属于Vue指令（）。

A v-for

B v-show

C v-blind

D v-on

作答结果：正确

打开解析

13

2分

下列哪些属性属于Vuex（）。

A State

B Statu

C Getter

D Mutation

作答结果：正确

打开解析

14

2分

对于Node.js 文件系统，下列说法正确的有（）。

A fs模块中的方法均有同步和异步版本

B 可以使用fs模块的open()方法在异步模式下打开文件

C 使用fs模块的writeFile()方法时，默认写入方式是追加写入

D 异步调用文件系统可以自动处理异常，把一个错误对象作为第一个参数传递

作答结果：错误

打开解析

15

2分

下列对于箭头函数的应用，正确的有（）。

A let f = a => a;

B let f = a,b => a+b;

C let f = () => 1+1;

D let f = (id,name) => {id: id, name: name};

作答结果：错误

打开解析

三、判断题 共5题，10分

1

2分

设置某个元素的上外边距为10px,左右外边距为20px,下外边距为20px，可以缩写为：margin:10px 20px 20px。

A 对

B 错

作答结果：正确

打开解析

2

2分

JSONP使用XMLHttpRequest协议的get请求。

A 对

B 错

作答结果：正确

打开解析

3

2分

在对复杂对象的查询操作时，可以使用POST请求。

A 对

B 错

作答结果：正确

打开解析

4

2分

设置<img>标签的height和width压缩图片，可以改变图片容量大小。

A 对

B 错

作答结果：错误

打开解析

5

2分

当图片资源使用HTTP强缓存后，使用Ctrl+F5刷新页面，页面上被强缓存的图片资源不会再从服务器请求下载。

Ⓐ 对

Ⓑ 错

作答结果：正确

打开解析

四、客观填空题　共4题，100分

1

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。

试题三.zip

【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
            1 : 'http://localhost:3000',
            // 设置是否跨域
            2 : true,
            pathRewrite: {
                '': ''
            }
        },
    },
    ...
}
```

【listProduct_client项目项目代码：ProductList.vue】

```
<template>
    <div>
        <!-- 搜索框 -->
        <div>
            <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 3 >
            <a href="javascript:;">
                <img v-bind:src="item.img">
                <h4>{{ item.name }}</h4>
                <div>￥ {{ item.price }}</div>
                <div v-on:click="handleDel(item.id)">删除</div>
                <!-- <div>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. 4 ('/products/product,{
                    5 : {
                        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. 6 ('/products/product',{
                    7 : {
                        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。

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 8 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 9 层。

【问题2】分析项目路由，回答下列问题。

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 10

```
const express = require('express');
const app = express();
```

A. app.get('/', function (req, res) {});

B. app.use('/', function (req, res) {});

C. app.get('/products/list', function (req, res) {});

D. app.all('/products/list', function (req, res) {});

2、在定义URI时，以下哪个URI符合RESTful API规范 11

A. GET /getProducts

B. DELETE /deleteProduct/1

C. GET /get_product?id=3

D. GET /products/product

【问题3】基于**Node.js**和**Express**框架，回答下列问题。

1、在app.js文件中，下列哪种中间件没有被用到 12

A. 应用层中间件

B. 路由器级中间件

C. 错误处理中间件

D. 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 13 空。

【问题4】分析模型文件，回答下列问题。

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 14

A. set ->constructor

B. constructor->set

C. constructor->set->constructor

D. set

2、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 15

A. constructor

B. set

C. get

D. constructor->get

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

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**）。

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。

1、学生答案:

2、学生答案:

3、学生答案:

4、学生答案:

5、学生答案:

6、学生答案:

7、学生答案:

8、学生答案:

9、学生答案:

10、学生答案:

11、学生答案:

12、学生答案:

13、学生答案:

14、学生答案:

15、学生答案:
1

打开解析

2

阅读下列说明，根据要求，完成以下题目。

20分

试题二.rar

【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：

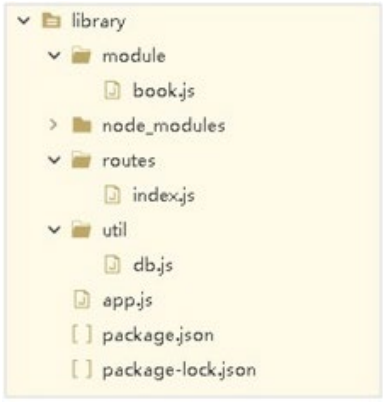


图3-1

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>
            <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】

1、基于Express框架，在app.js中填写（1）-（5）空。

2、基于MySQL模块，在db.js中填写（6）-（10）空。

3、在book.js中，填写（11）空，导入数据库配置文件。

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。

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。

6、在跨域请求接口中，填写（14）空，返回正确内容。基于JSONP实现跨域请求，填写（15）空。

1、学生答案:

2、学生答案:

3、学生答案:

4、学生答案:

5、学生答案:

6、学生答案:

7、学生答案:

8、学生答案:

9、学生答案:

10、学生答案:

11、学生答案:

12、学生答案:

13、学生答案:

14、学生答案:

15、学生答案:
2

打开解析

3

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。

试题四.zip

【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。

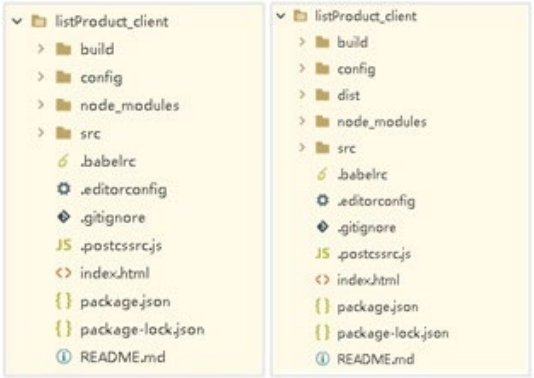


图1-1

（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">
        <img src="img/2.jpg">
        <img src="img/3.jpg">
        <img src="img/4.jpg">
    </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**）。

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）。

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] ？

A. .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;}

B. .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;}

C. .normal-img{width:240px;height:240px;background:#6495ed url('img/1.jpg') right top no-repeat ;display: inline-block;}

D. .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）。

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]

A. 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**）。

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**）。

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。

1、学生答案:

2、学生答案:

3、学生答案:

4、学生答案:

5、学生答案:

6、学生答案:

7、学生答案:

8、学生答案:

9、学生答案:

10、学生答案:

11、学生答案:

12、学生答案:

13、学生答案:

14、学生答案:

15、学生答案:
3

打开解析

4

30分

阅读下列说明、效果图和代码，填写（1）至（15）代码。

试题一.rar

【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

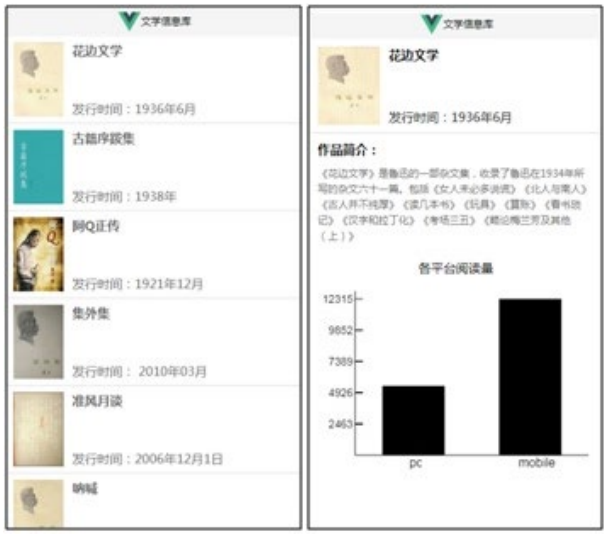
（3）文学库首页和书籍详细信息页样式使用less语法编写

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。

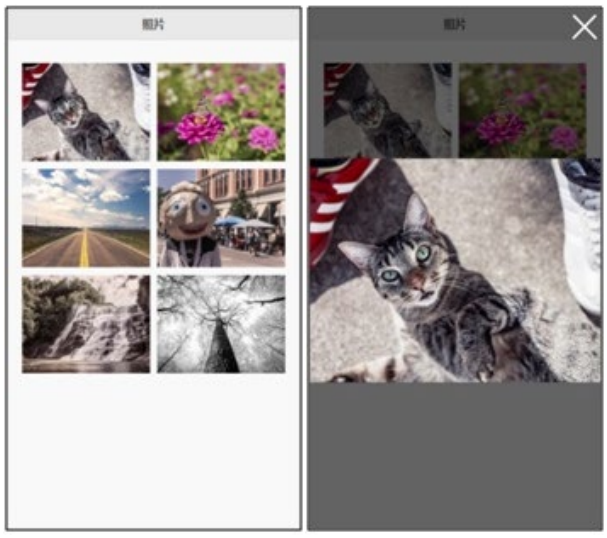
（2）SVG图片绘制了弹出层的关闭按钮图标，使用CSS的背景图引入到页面。

【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>
<!--公共头部-->
<subHeader></subHeader>
<div>
<ul>
            <!--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>
<div>{{ item.name }}</div>
<div>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>
    /*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>
<!--jQuery Mobile网格布局，每行两列-->
<ul class="ui-grid-a img_list">
<li class="    10    ">
<div><img src="img/1.jpg" /></div>
</li>
<li class="    11    ">
<div><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 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>
```

1、学生答案：

2、学生答案：

3、学生答案：

4、学生答案：

5、学生答案：

6、学生答案：

7、学生答案：

8、学生答案：

9、学生答案：

10、学生答案：

11、学生答案：

12、学生答案：

13、学生答案：

14、学生答案：

15、学生答案：

4

打开解析