阶段知识点考题总结

1、简述mvc和mvvm

早期的mvc是一种架构理念，随着互联网的发展，这种架构理念逐渐被应用到实际的开发当中，成为一种架构模式。最终达到的效果是页面与数据分离
* MVC：
- M层(数据层)
    - 存储数据的实体模型
- C层(控制层)
    - 操作模型数据，更新视图
    - View与Model之间的桥梁
    - 是v层的直接数据源
- V(层)
    - 显示数据
    - 响应用户操作，与用户进行交互
- 在mvc中最核心的是C层，它是M层与V层的纽带
* mvvm
    - 在mvvm中最核心的vm层($scope)，controller层只是起辅助作用，其他的同MVC一样，MVVM是MVC的一个升级版，angular应用的就是MVVM架构模式

2、简述箭头函数的特点，并用箭头函数写出数组map遍历的方法

- 箭头函数没有自己的this，在语法上更为简介。
- 箭头函数的this不是在调用的时候决定的，是在定义的时候决定的，定义时候所处的对象就是箭头函数的this
- 箭头函数的this是看外层有没有函数，如果有，和外层的函数指向的是同一个this，如果没有则指向window
- arr.map((item, index) => {
    console.log(item, index)
  })

3、bind，call，apply的相同点和不同点

- bind，call，apply都能强制指定this
- call，apply指定完this后立即调用当前函数
- bind绑定完this不会立即调用，将当前的函数返回
- call，apply传参形式不一样，apply需要传入数组里
- bind也可以传入参数，同call一样直接传参

4、promise对象的原理及作用

> Promise对象: 代表了 未来 某个将要发生的事件(通常是一个异步操作)
> ES6的Promise是一个构造函数，用来生成promise实例
> 有了promise对象，可以将异步操作以同步的流程表达出来，避免了层层嵌套的 回调函数 (俗称'回调地狱')
> promise有三种状态('初始化状态'， '成功的状态'， '失败的状态')
> 通过执行异步任务返回的结果(通常是发送ajax请求)来修改promise的状态，
> 当promise的状态发生改变的时候会调用promise的实例中的then方法的成功或者失败的回调后函数，去执行相应的操作。

5、package.json中最重要的五个属性，及作用是什么

```
{
  "name": "npm_command", //包名，必不可少
  "version": "1.0.0", //版本， 必不可少
  "scripts": { //配置npm运行命令
    "start": "node bin/www"
  },
  "dependencies": {//运行依赖的包
    "jquery": "^3.2.1"
  },
  "devDependencies": {//开发依赖的包
```

```
    "babel": "^6.23.0"
  }
}
```

6、什么是跨域，解决跨域的方法及原理是什么？

```
1. 不同源就是跨域
2. 同源策略是浏览器的一种安全策略
3. 协议，域名，端口号完全相同就是同源，只要有一处不一样就是跨域
4. 特例： ajax在判断域名的时候只能解析字符串，导致(localhost和127.0.0.1)在它看来也是跨域请求
5. 解决跨域的方式通常用cors和jsonp
6. JSONP
    1. JSONP是一种技巧，不是一门新的技术
    2. 利用scirpt标签的src属性不受跨域的限制的特点
    3. 解决跨域：
        1. 浏览器端：
            动态生成script标签，提前定义好回调函数，在合适的时机添加src属性指定请求的地址。
        2. 服务器端：
            后台接收到回调函数，将数据包括在回调函数调用的句柄中，一起返回。
        3. 只支持get请求
7. cors
    1. 浏览器端什么也不用干
    2. 服务器端设置响应头：Access-Control-Allow-Origin
    3. cors是一门技术，在本质上让ajax引擎允许跨域
    4. get和post请求都支持
```

commonjs和ES6模块化暴露的本质分别是什么，请详细说明

```
1、commonjs暴露的方式
    - module.exports = value;
    - exports.xxx = value;
    - 暴露的本质是exports对象
2、ES6中暴露的方式
    - export xxx （常规暴露，暴露的本质是对象，接收的时候只能以对象的解构赋值的方式来接收值）
    - export default （默认暴露，暴露任意数据类型，暴露什么数据类型，接收什么数据类型）
```

模块化的优点和缺点

```
1、模块缺点：
        - 请求过多
        - 依赖模糊
        - 难以维护
2、模块化优点：
    - 避免命名冲突(减少命名空间污染)
    - 更好的分离，按需加载
    - 更高复用性
    - 高可维护性
```

请写出自动化工具里的至少两种项目构建的方法，特点和需要注意的地方

```
1、grunt
    - 执行任务是同步的
    - 配置文件需要首字母大写 Gruntfile.js
    - 执行任务需要依赖对应的插件
    - 配置任务的时候要注意加载对应的插件任务
2、gulp
    - 执行任务是异步的
    - 任务化
    - 基于流
    - 所有的操作都是在内存中，操作完的数据通过pipe管道流出。
```

- 可全自动加载(connect)也可半自动加载(watch插件)
- gulp-load-plugins可打包加载项目中gulp的所有插件，注意引入后得到的是一个函数，必须调用，返回得一个对象，对象里边封装了所有打包插件的方法。

3、webpack

- 所有的文件都是模块(html除外)
- webpack本身只能加载js模块，加载其他的模块需要依赖对应的loader

```
module.exports = {
    entry: "./src/js/entry.js",//入口文件的路径
    output: {//输出文件的配置
        path: __dirname + '/dist/js/',//输出文件的路径
        publicPath : './dist/js/',//webpack的绝对路径
        filename: "build.js"//输出的文件名
    },
    module: {
        loaders: [//加载对应的loader
            { test: /\.css$/, loader: "style!css" },
            { test: /\.(png|jpg)$/, loader: "url-loader?limit=8192" }
        ]
    },

}
```

- 使用webpack-dev-server的时候需要注意的地方
  - webpack-dev-server为根文件夹下的index.html提供内置的服务
  - 如果其他目录下的文件提供服务需要在此设置目录(我们设置为build文件夹)

```
  devServer:{
      contentBase: './build',//内置服务器生成的打包的js文件的服务路径
      historyApiFallback:true,//不跳转
      inline:true,
      port : 5000
  }
```

- 打包图片使用url-loader时候注意的问题
  - 当图片大于8kb的是无法打包到主文件里
  - 导致图片的路径发生错误
  - 解决办法：
    - publicPath : './dist/js/',//webpack的绝对路径