Web 前端面笔试 题库

日期：2022.04.22

《Web 前端面笔试题库》

提示：

1、 题目类型包括：单选题、多选题、简答题、填空题、编程题等

2、 题目前缀【类型】【知识点】

知识点目录			
HTTP	HTML	CSS	JavaScript
jQuery	Bootstrap	Layui	Node.js
npm/yarn	Express	模块化	多线程
RESTful	微信小程序	网络基础	UML 建模
Webpack	Vue	Axios	Uniapp
Sass			

-业务场景——微信小程序

1. 【简答题】【微信小程序】先介绍一下你做的小程序项目吧。

喵喵社区，一款基于微信小程序平台的在线社区应用。应用的主要目的是给爱好猫猫的铲屎官提供在线交流、沟通；喵喵线下活动；商城等功能。主要包含：喵喵社区、活动、商城、云吸猫等功能。后台主要通过 Express 实现，数据存储使用 MySQL 来完成。

2. 【简答题】【微信小程序】为什么要用微信小程序而不使用像 iOS、Android 甚至是混合应用等其他移动应用方案？

首先从流量上来看，目前微信作为目前普及范围最广的范综合社交软件，其有这无比的用户量，小程序借助自身平台更加容易引入更多的流量，借助微信的成熟社交网络达到爆发式传播；其次从用户体验上来看，微信小程序是要用的时候打开，不用的时候关掉，“即用即走”，不需要用户安装下载，也不需要用户更新，无序管理磁盘空间，对用户体验来说比较友好。其次从开发技术角度来看，微信小程序可以在不同平台中使用，只要能够安装微信的设备基本都可以使用，无序考虑跨平台兼容，节省开支。小程序开发技术接近与前端，开发周期短，开发人员可

快速上手，只要会 web 前端开发，基本都能完成小程序的开发等等。

3. 【简答题】【微信小程序】说说小程序的开发流程。

1、注册微信小程序账号

2、获取微信小程序的 AppID

3、下载微信小程序开发者工具

4、创建小程序项目

5、去微信公众平台配置域名

6、前后端代码的编写

7、手机预览体验

8、代码上传

9、提交审核

10、审核通过后发布

4. 【简答题】【微信小程序】请简单说说小程序的原理吧。

微信小程序采用 JavaScript、WXML、WXSS 三种技术进行开发,本质就是一个单页面应用，所有的页面渲染和事件处理，都在一个页面内进行，但又可以通过微信客户端调用原生的各种接口

微信的架构，是数据驱动的架构模式，它的 UI 和数据是分离的，所有的页面更新，都需要通过对数据的更改来实现

小程序分为两个部分 webview 和 appService 。其中 webview 主要用来展现 UI ，appService 用来处理业务逻辑、数据及接口调用。它们在两个进程中运行，通过系统层 JSBridge 实现通信，实现 UI 的渲染、事件的处理

5. 【简答题】【微信小程序】请说说项目中商城购物实现的流程。

喵喵社区中的商城主要是提供铲屎官购买猫零食、猫玩具、猫砂、猫窝等一系列猫宠物产品的，商城和日常生活中的商城基本类似，具体实现流程如下：

在用户加入商品至购物车时，首先需要进行登录拦截，判断用户是否处于登录状态？如果未登录，则会弹出微信授权登录的界面提示用户进行授权；当用户已登录了，则根据添加商品的信息发送请求，将商品添加至购物车列表数据。

添加商品成功后，转至购物车页面时，在 onShow 生命周期函数中判断是否登录，如果未登录则提醒微信授权登录，如果已登录，则将购物车列表数据请求下来，以便渲染页面。如果没有数据，则显示购物车空空，如果有数据则渲染页面。

页面渲染完毕后，就涉及到商品总数量以及总价的计算。总价和总数量可以通过计算属性获得，通过遍历购物车列表数据，就可以实现总价和总数量的获取。

单个商品数量加、减和删除，以及总价总数量的事实更新。加、减：先实现视图化的加、减，再将加减后的数

值连带着 token、cartid 去请求数据以实现后台数据的实时更新。

删除：将点击删除的商品的 cartid 以及 token 获取到，并发送后台请求数据，以实现后台数据的实时更新，并将视图化的数据更新。

6. 【简答题】【微信小程序】小程序的登录流程是怎样的，你的项目中是怎么保持登录状态的？

实现小程序用户登录流程，大致分为以下几个步骤：

1、调用 wx.login() 获取 临时登录凭证 code ，并回传到开发者服务器。

2、调用 auth.code2Session 接口，换取用户唯一标识 OpenID 和会话密钥 session_key。自定义登录状态关联 openid 和 session_key，返回自定义登录状态

3、前端缓存自定义状态到 storage，wx.request()携带自定义登录状态请求数据

4、开发者服务器通过自定义登录状态查询 openid 和 session_key，验证通过返回业务数据

持久登录状态：由于 wx.request()发起的每次请求对于服务器来说都是不同的会话（wx.request()请求是先经过微信服务器再到达我们的服务器），这样会导致后续请求都相当于未登录的状态。因此需要将登录时后端返回的 session 保存在本地， 然后将 session 存放在 cookie 中以请求头的方式带回给服务端。

7. 【简答题】【微信小程序】你是如何封装微信小程序的数据请求的？

1、将所有的接口放在统一的 js 文件中并导出。

2、在 app.js 中创建封装请求数据的方法。

3、在子页面中调用封装的方法请求数据。

8. 【简答题】【微信小程序】你项目中的支付功能要怎么实现，有哪些流程？

1、小程序注册，要以公司的以身份去注册一个小程序，才有微信支付权限；

2、绑定商户号。

3、在小程序填写合法域

4、调用后台接口获取必要参数

5、调用 wx.requestPayment({})发起微信支付

9. 【简答题】【微信小程序】结合你的项目，请说说微信小程序存在哪些弊端？

1、限制较多。页面大小不能超过 1M。不能打开超过 5 个层级的页面。

2、样式单一。小程序的部分组件已经是成型的了，样式不可以修改。例如：幻灯片、导航。

3、推广面窄，不能分享朋友圈，只能通过分享给朋友，附近小程序推广。其中附近小程序也受微信的限制。

4、依托于微信，无法开发后台管理功能。

5、缺乏自由度，小程序要面对很多来自微信的限制，从功能接口，甚至到类别内容，都要接受微信的管控。

6、小程序功能不够全面，解决不了复杂的行业问题。

- 业务场景——Uniapp

1. 【简答题】【Uniapp】简单介绍一下你的项目吧。

绿色优选是一款基于 Uniapp 实现的一个在线蔬菜购物商城类 APP，主要实现的功能有：时令蔬菜水果推荐，蔬菜水果营养表，购菜大厅，购物车，我的历史订单等。前端采用了前端 Hybrid 开发框架 Uniapp，后端服务是 Express，数据库采用了 MySQL。

2. 【简答题】【Uniapp】为什么要用 Uniapp 而直接用 Android 或者微信小程序开发呢？

设计项目的时候考虑到我们的产品希望发布到多个平台使用，而如果针对每个平台都单独开发一套代码的话会严重拖沓项目完成的进度。Uniapp 框架可以将我们写好的代码通过翻译直接转换成多个平台的代码，在同类产品里面也调研过 react-native 和 flutter，但是由于方向课学习的是 Vue，而 Uniapp 与 Vue 语法又很相近，学习成本较低，因此团队最终选择了 Uniapp。

3. 【简答题】【Uniapp】简述一下你购物实现的流程。

在用户加入商品至购物车时，首先得判断，当前是否处于登录状态？如果未登录，则不能添加，并跳转至登录界面；如果已登录，则根据添加商品的信息发送请求，将商品添加至购物车列表数据。

添加商品成功后，转至购物车页面时，在 onshow 生命周期函数中判断是否登录，如果未登录则跳转至登录界面，如果已登录，则将购物车列表数据请求下来，以便渲染页面。如果没有数据，则显示购物车空空，如果有数据则渲染页面。

页面渲染完毕后，就涉及到商品总数量以及总价的计算。总价和总数量可以通过计算属性获得，通过遍历购物车列表数据，就可以实现总价和总数量的获取。

单个商品数量加、减和删除，以及总价总数量的事实更新。加、减：先实现视图化的加、减，再将加减后的数值连带着 token、cartid 去请求数据以实现后台数据的实时更新。

删除：将点击删除的商品的 cartid 以及 token 获取到，并发送后台请求数据，以实现后台数据的实时更新，并将视图化的数据更新。

4. 【简答题】【Uniapp】项目中是怎么保存用户登录状态的呢？

前端在用户输完账号密码后会将数据发送到服务端，在 Express 中使用了 express-jwt 和 jsonwebtoken 来进行 jwt

鉴权，然后将 token 返回给前端，前端拿到 token 后调用 setStorage 方法将 token 保存到本地缓存。想要判断用户是否登录就看缓存中有没有 token 信息。

5. 【简答题】【Uniapp】在项目中还用到其他插件吗，为什么要使用？

有的，项目中由于要录入用户地址，为了方便用户录入，提升用户体验，所以融入了定位功能，但是 Uniapp 原生的定位在有些平台无法获取定位城市，仅能获取经纬度，为了更好的兼容不同平台，最好的方式是使用第三方平台的经纬度解析功能。因此我们在项目中使用了腾讯定位服务。

6. 【简答题】【Uniapp】既然提到了多平台，那么在不同平台上样式如何保证统一呢？

第一个是为了保证项目中的元素可以针对不同机型进行自适应，因此大量采用了 Uniapp 官方推荐的尺寸单位 upx。uni-app 规定屏幕基准宽度 750upx，upx 是相对于基准宽度的单位，可以根据屏幕宽度进行自适应。

第二是在样式设置的时候，针对一些特有平台的特性使用 #ifdef 来进行特殊化的处理，这样就可以保证我们的样式在不同平台和机型上都可以达到一致的效果

7. 【简答题】【Uniapp】项目开发中遇到过什么问题吗？如何解决的？

第一个是开发中遇到一个问题是打包到微信小程序中背景图片使用本地路径后不显示，会报错，最终的解决方案是使用 base64 或者 https 连接的方式进行了图片的使用。

第二个是项目运行到 ios 上，swiper 组件不显示图片，点击原本的图片区域正常显示预览图，最后将图片 image 组件换为 view，使用背景图将问题解决了。

- 业务场景——Vue

1. 【简答题】【Vue】请描述你做过的项目。

“前辈说”是一个大学生学习、交友、分享交流经验的平台，致力于为大学生提供丰富宝贵的学习经验，平台内设有四六级及各种等级考试、考研、考公、实习就业等经验。用户可在平台上搜寻相关的经验，与前辈在线聊天对话，发布自己的学习经验，在学习路上找到良师益友和志同道合的朋友！

该项目的核心技术采用 Vue.js，针对移动端页面布局，使用 VantUI 框架，后端采用 PHP，数据库使用 MySQL。

2. 【简答题】【Vue】为什么选用 Vue.js 作为核心框架？

Vue 是一套构建用户界面的渐进式自底向上增量开发的 MVVM 框架，vue 的核心只关注视图层。

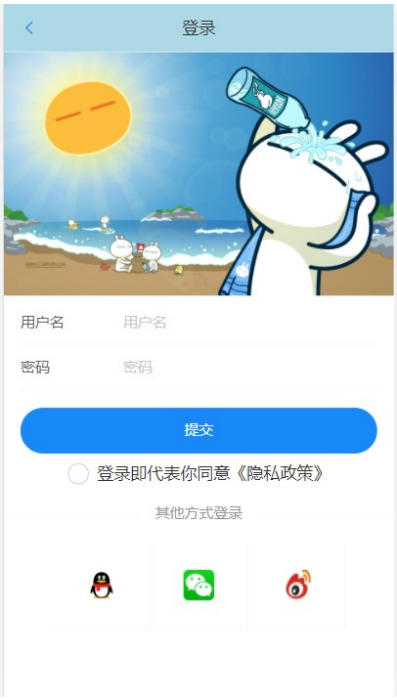
核心思想：

数据驱动（视图的内容随着数据的改变而改变）；

组件化（可以增加代码的复用性，可维护性，可测试性，提高开发效率，方便重复使用，体现了高内聚低耦合） 。

另外，Vue.js 入门相对比较容易，有 HTML、CSS、JavaScript 的基础就可以，方便上手。

3. 【简答题】【Vue】请说明项目中登录实现的基本流程。



“用户名”、“密码”输入框通过“v-model”实现数据的双向绑定，当用户输入用户名和密码后提交，使用 axios 向服务端发送数据。服务端对数据进行校验，并以 JSON 格式返回结果，最后页面显示对应的登录结果。

4. 【简答题】【Vue】进行登录时遇到了跨域的问题怎么办?

后台更改 header；

使用 jQuery 提供 jsonp；

用 http-proxy-middleware（配置代理服务器的中间件）；

5. 【简答题】【Vue】刚刚提到了双向绑定，请说明 Vue 双向绑定实现的原理。

vue 双向数据绑定是通过 数据劫持 和 发布订阅模式（观察者模式）的方式来实现的， 也就是说数据和视图同步，数据发生变化，视图跟着变化，视图变化，数据也随之发生改变；

观察者模式：当属性发生改变的时候，使用该数据的地方也发生改变

核心：关于 VUE 双向数据绑定，其核心是 Object.defineProperty()方法；

介绍一下 Object.defineProperty()方法

1、Object.defineProperty(obj, prop, descriptor) ，这个语法内有三个参数，分别为 obj （要定义其上属性的对象） prop （要定义或修改的属性） descriptor （具体的改变方法）

2、简单地说，就是用这个方法来定义一个值。当调用时我们使用了它里面的 get 方法，当我们给这个属性赋值时，又用到了它里面的 set 方法；

```
var obj = {};
Object.defineProperty(obj, 'hello', {
    get: function(){
        console.log('调用了get方法')
    },
    set: function(newVal){
        console.log('调用了set方法，方法的值是' + newVal);
    }
});
obj.hello; // => '调用了get方法'
obj.hello = 'hi'; // =>'调用了set方法，方法的值是'hi
```


6. 【简答题】【Vue】在项目首页中，页面是如何渲染的，如何保证性能？

首先，通过接口获取首页的列表数据，通过 v-for 指令对表示单个用户图文的组件进行列表渲染，就出现首页的效果。

为了保证性能，在 v-for 指令中使用唯一的 id 作为 key，因为 Key 值的存在保证了唯一性。Vue 在执行时，会对节点进行检查，如果没有 key 值，那么 vue 检查到这里有 dom 节点，就会对内容清空并赋新值，如果有 key 值存在，那么会对新老节点进行对比，比较两者 key 是否相同，进行调换位置或删除操作。

另外，还使用了 keep-alive 组件提升性能。

7. 【简答题】【Vue】keep-alive 组件为什么能提升性能？

概念：keep-alive 是 Vue 的内置组件，当它动态包裹组件时，会缓存不活动的组件实例，它自身不会渲染成一个 DOM 元素也不会出现在父组件链中。

作用：在组件切换过程中将状态保留在内存中，防止重复渲染 DOM，减少加载时间以及性能消耗，提高用户体验。

生命周期函数：Activated 在 keep-alive 组件激活时调用，deactivated 在 keep-alive 组件停用时调用。

- 面试/ 笔试题

1. 【多选题】【HTML】下面关于表格标签说法正确的是（）

```
<table cellspacing="30">
    <tr><td colspan="2" align="center">姓名</td></tr>
    <tr><td rowspan="2" align="center">成绩</td>
        <td align="center">语文</td>
    </tr>
    <tr><td colspan="2" align="center">数学</td></tr>
</table>
```

A、该表格共有两行三列

B、该表格边框宽度为 30 像素

C、该表格中的文字均居中显示

D、姓名”单元格跨 2 列

答案：C、D

解析：

该表格有三行三列，A 选项错误；cellspacing 是单元格之间的空间，而不是表格边框宽度，B 选项错误。

2. 【单选题】【HTML】以下关于 HTML 标签说法正确的是（）

A、标题标签、段落标签、标签都是块级元素

B、 … … 标签是内联元素

C、 标签可以包含于标签中

D、display 属性可以控制块级元素和内联元素的显示方式

答案：D

解析：

标签是内联元素，A 选项错误；

标签是块级元素，B 选项错误；

标签可以包含于 标签中，与 C 选项表述正好相反。

3. 【简答题】【HTML】语义化的理解。

用正确的标签做正确的事情！

HTML 语义化就是让页面的内容结构化，便于对浏览器、搜索引擎解析；

在没有样式 CSS 情况下也以一种文档格式显示，并且是容易阅读的。

搜索引擎的爬虫依赖于标记来确定上下文和各个关键字的权重，利于 SEO 。

使阅读源代码的人对网站更容易将网站分块，便于阅读维护理解

4. 【简答题】【HTML】行内元素和块级元素？img 算什么？行内元素怎么转化为块级元素？

行内元素：和有他元素都在一行上，高度、行高及外边距和内边距都不可改变，文字图片的宽度不可改变，只能容纳文本或者其他行内元素；其中 img 是行元素

块级元素：总是在新行上开始，高度、行高及外边距和内边距都可控制，可以容纳内敛元素和其他元素；

行元素转换为块级元素方式：display：block；

5. 【简答题】【HTML】HTML5 有哪些新特性？

HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加：

绘画 canvas

用于媒介回放的 video 和 audio 元素

本地离线存储 localStorage 存储数据，浏览器关闭后数据不丢失 sessionStorage 的数据在浏览器关闭后自动删除

语意化更好的内容元素，如 article 、 footer 、 header 、 nav 、 section

表单控件， calendar 、 date 、 time 、 email 、 url 、 search

新的技术 webworker 、 websocket 、 Geolocation

6. 【简答题】【HTML】href 和 src 区别? title 和 alt 呢？

href (Hypertext Reference)指定网络资源的位置（超文本引用），从而在当前元素或者当前文档和由当前属性定义的需要的锚点或资源之间定义一个链接或者关系，在 link 和 a 等元素上使用。

src (Source)属性仅仅嵌入当前资源到当前文档元素定义的位置，是页面必不可少的一部分，是引入。在 img、script、iframe 等元素上使用。

title：既是 html 标签，又是 html 属性，title 作为属性时，用来为元素提供额外说明信息.

alt：alt 是 html 标签的属性，alt 属性则是用来指定替换文字，只能用在 img、area 和 input 元素中（包括 applet 元素），用于网页中图片无法正常显示时给用户提供文字说明使其了解图像信息.

7. 【简答题】【HTML】HTML5 移除了那些元素？

纯表现的元素： basefont 、 big 、 center 、 font 、 s 、 strike 、 tt 、 u

对可用性产生负面影响的元素： frame 、 frameset 、 noframes

8. 【简答题】【HTML】HTML5 的离线储存怎么使用？

页面头部像下面这样加入一个 manifest 的属性；

在 cache.manifest 文件的编写离线存储的资源；

在离线状态时，操作 window.applicationCache 进行需求实现。

9. 【简答题】【HTML】HTML5 离线储存的工作原理是什么？

HTML5 的离线存储是基于一个新建的 .appcache 文件的缓存机制(不是存储技术)，通过这个文件上的解析清单离线存储资源，这些资源就会像 cookie 一样被存储了下来。之后当网络在处于离线状态下时，浏览器会通过被离线存储的数据进行页面展示。

10. 【简答题】【HTML】描述一下 cookies，sessionStorage 和 localStorage 的区别？

cookie 是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）

cookie 数据始终在同源的 http 请求中携带（即使不需要），记会在浏览器和服务器间来回传递

sessionStorage 和 localStorage 不会自动把数据发给服务器，仅在本地保存

存储大小：

cookie 数据大小不能超过 4k

sessionStorage 和 localStorage 虽然也有存储大小的限制，但比 cookie 大得多，可以达到 5M 或更大

有期时间：

localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动删除数据

sessionStorage 数据在当前浏览器窗口关闭后自动删除

cookie 设置的 cookie 过期时间之前一直有效，即使窗口或浏览器关闭

11.【简答题】【HTML】HTML 全局属性(global attribute)有哪些？

class :为元素设置类标识

data-* : 为元素增加自定义属性

draggable : 设置元素是否可拖拽

id : 元素 id ，文档内唯一

lang : 元素内容的的语言

style : 行内 css 样式

title : 元素相关的建议信息

12.【简答题】【HTML】viewport 有哪些属性？

<meta name="viewport" content="width=device-width,initial-scale=1.0,minimu

width 设置 viewport 宽度，为一个正整数，或字符串‘device-width’

device-width 设备宽度

height 设置 viewport 高度，一般设置了宽度，会自动解析出高度，可以不用设置

initial-scale 默认缩放比例（初始缩放比例），为一个数字，可以带小数

minimum-scale 允许用户最小缩放比例，为一个数字，可以带小数

maximum-scale 允许用户最大缩放比例，为一个数字，可以带小数

user-scalable 是否允许手动缩放

13.【简答题】【HTML】渐进增强和优雅降级之间有什么不同？

渐进增强：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

优雅降级：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

区别：优雅降级是从复杂的现状开始，并试图减少用户体验的供给，渐渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要。降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带。

14.【简答题】【HTML】简述一下 src 与 href 的区别。

src 用于替换当前元素，href 用于在当前文档和引用资源之间确立联系。

src 是 source 的缩写，指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求 src

资源时会将其指向的资源下载并应用到文档内，例如 js 脚本，img 图片和 frame 等元素

说明：<script src ="js.js"></script> 当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行完毕，图片和框架等元素也如此，类似于将所指向资源嵌入当前标签内。这也是为什么将 js 脚本放在底部而不是头部。

15.【简答题】【HTML】一个页面上有大量的图片（大型电商网站），加载很慢，你有哪些方法优化这些图片的加载，给用户更好的体验。

图片懒加载，在页面上的未可视区域可以添加一个滚动事件，判断图片位置与浏览器顶端的距离与页面的距离，如果前者小于后者，优先加载。

如果为幻灯片、相册等，可以使用图片预加载技术，将当前展示图片的前一张和后一张优先下载。

如果图片为 css 图片，可以使用 CSSsprite ， SVGsprite ， Iconfont 、 Base64 等技术。

如果图片过大，可以使用特殊编码的图片，加载时会先加载一张压缩的特别厉害的缩略图，以提搞用户体验。

如果图片展示区域小于图片的真实大小，则因在服务器端根据业务需要先进行图片压缩，图片压缩后大小与展示一致。

16.【简答题】【HTML】HTTP request 报文结构是怎样的？

首行是 Request-Line 包括：请求方法，请求 URI，协议版本，CRLF

首行之后是若干行请求头，包括 general-header，request-header 或者 entity-header，每一行以 CRLF 结束

请求头和消息实体之间有一个 CRLF 分隔

根据实际请求需要可能包含一个消息实体

17.【简答题】【HTML】HTTP response 报文结构是怎样的？

首行是状态行包括：HTTP 版本，状态码，状态描述，后面跟一个 CRLF

首行之后是若干行响应头，包括：通用头部，响应头部，实体头部

响应头部和响应实体之间用一个 CRLF 空行分隔

最后是一个可能的消息实体

18.【简答题】【HTML】让行内元素水平居中的两种方法？

1、找到对应其标签的父级，给其父级设置 text-align : center；

2、将元素转化成块元素，设置 margin : 0 auto，（必须是块元素，而且有宽度）

19.【简答题】【HTML】为什么利用多个域名来存储网站资源会更有效？

CDN 缓存更方便

突破浏览器并发限制

节约 cookie 带宽

节约主域名的连接数， 优化页面响应速度

防止不必要的安全问题

20.【简答题】【HTML】请谈一下你对网页标准和标准制定机构重要性的理解。

网页标准和标准制定机构都是为了能让 web 发展的更‘健康’， 开发者遵循统一的标准， 降低开发难度， 开发成本， SEO 也会更好做， 也不会因为滥用代码导致各种 BUG、 安全问题，最终提高网站易用性。

21.【简答题】【HTML】web storage 和 cookie 的区别？

Web Storage 的概念和 cookie 相似， 区别是它是为了更大容量存储设计的。 Cookie 的大小是受限的， 并且每次你请求一个新的页面的时候 Cookie 都会被发送过去， 这样无形中浪费了带宽， 另外 cookie 还需要指定作用域， 不可以跨域调用。

除此之外， Web Storage 拥有 setItem,getItem,removeItem,clear 等方法， 不像 cookie 需要前端开发者自己封装 setCookie， getCookie。 但是 Cookie 也是不可以或缺的： Cookie 的作用是与服务器进行交互， 作为 HTTP 规范的一部分而存在 ， 而 Web Storage 仅仅是为了在本地“存储” 数据而生。

22.【简答题】【HTML】知道什么是微格式吗？谈谈理解。在前端构建中应该考虑微格式吗？

微格式（Microformats） 是一种让机器可读的语义化 XHTML 词汇的集合， 是结构化数据的开放标准。 是为特殊应用而制定的特殊格式。

优点：将智能数据添加到网页上，让网站内容在搜索引擎结果界面可以显示额外的提示。（应用范例： 豆瓣）

23.【简答题】【HTML】新的 HTML5 文档类型和字符集是？

HTML5 文档类型很简单： HTML5 使用 UTF-8 编码。

24.【简答题】【HTML】 Canvas 元素有什么用？

Canvas 元素用于在网页上绘制图形，HTML5 的 canvas 元素使用 JavaScript 在网页上绘制图像

25.【简答题】【HTML】HTML5 存储类型有什么区别？

HTML5 能够本地存储数据，在之前都是使用 cookies 使用的。HTML5 提供了下面两种本地存储方案：localStorage 用于持久化的本地存储，数据永远不会过期，关闭浏览器也不会丢失。 sessionStorage 同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此 sessionStorage 不是一种持久化的本地存储，仅仅是会话级别的存储

26.【简答题】【HTML】你做的页面在哪些流览器测试过？这些浏览器的内核分别是什么？

IE: trident 内核

Firefox： gecko 内核

Safari:webkit 内核

Opera:以前是 presto 内核， Opera 现已改用 Google Chrome 的 Blink 内核

Chrome:Blink(基于 webkit， Google 与 Opera Software 共同开发)

27.【简答题】【HTML】每个 HTML 文件里开头都有个很重要的东西，Doctype，知道这是干什么的吗？

<!DOCTYPE> 声明位于文档中的最前面的位置， 处于 <html> 标签之前。 此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范。 （重点： 告诉浏览器按照何种规范解析页面）

28.【简答题】【HTML】div+css 的布局较 table 布局有什么优点？

改版的时候更方便 只要改 css 文件。

页面加载速度更快、 结构化清晰、 页面显示简洁。

表现与结构相分离。

易于优化（seo） 搜索引擎更友好， 排名更容易靠前。

29.【简答题】【HTML】img 的 alt 与 title 有何异同？ strong 与 em 的异同？

a:alt(alt text):为不能显示图像、 窗体或 applets 的用户代理（UA）， alt 属性用来指定替换文字。 替换文字的语言由 lang 属性指定。 (在 IE 浏览器下会在没有 title 时把 alt 当成 tool tip 显示)

title(tool tip):该属性为设置该属性的元素提供建议性的信息。

strong:粗体强调标签， 强调， 表示内容的重要性

em:斜体强调标签， 更强烈强调， 表示内容的强调点

30.【单选题】【CSS】在 html 中，实现如下图所示效果,则横线处应填写的代码是（）

代码：

```
<style>
li{
width:150px;
font:28px 隶书;
list-style:_____;
float:_____;
}
</style>
<div>
<ul>
<li>登录</li>
<li>注册</li>
<li>帮助</li>
</ul>
</div>
```

A、第一个横线:inherit，第二个横线:right

B、第一个横线:none，第二个横线:right

C、第一个横线:inherit，第二个横线:left

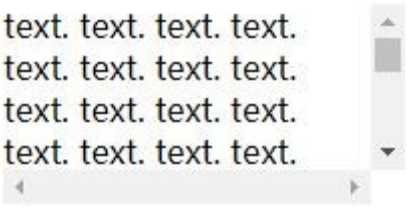
D、第一个横线:none，第二个横线:left

答案：D

解析：

list-style:“none”去除文字左边的小黑点，float:“left”向左浮动。

31.【单选题】【CSS】在 HTML 中， 标签默认样式下是不带滚动条的，若要使 标签出现滚动条，如以下效果图，需要为该标签定义什么样式（）



A、overflow:hidden

B、display:block

C、overflow:scroll

D、display:scroll

答案：C

解析：

overflow 属性规定当内容溢出元素框时发生的事情。

scroll：内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容；

hidden：内容会被修剪，并且其余内容是不可见的。

32.【多选题】【CSS】能够正确的在一个 HTML 页面中导入在同一目录下的“StyleSheet1.css”样式表的是以下哪个？

A、 <style>@import StyleSheet1.css;</style>

B、<link rel="stylesheet" type="text/css" href="StyleSheet1.css">

C、<link rel="StyleSheet1.css" type="text/css">

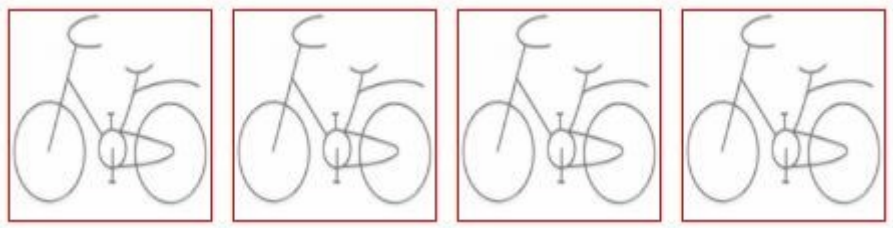
D、<style rel="stylesheet" type="text/css" href="StyleSheet1.css"></style>

答案：A、B

解析：

注意题目给的“同一目录下”，在这样的情况下，可以采用 A 选项和 B 选项的写法，但“不在同一目录下”的话，A 选项的写法不可行。C 选项和 D 选项没有这种写法，错误。

33.【编程题】【CSS】在页面的排版中，我们经常见到多张图片显示在同一行的情况，以下效果图是简化版，每张图之间间隔 10px，请补全缺失的代码。



代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        img {
            width: 100px;
            height: 100px;
        }
        div {
            /* code */


        }
    </style>
</head>
<body>
<div><img src="img1.jpg"></div>
<div><img src="img1.jpg"></div>
<div><img src="img1.jpg"></div>
<div><img src="img1.jpg"></div>
</body>
</html>
```

答案：

float: left;

border: 1px solid red;

margin-right: 10px;

解析：

图片显示在同一行；图片之间有间隔；DIV 的边框不做要求。

34.【编程题】【CSS】我们在设计页面的时候，通常会接触到页面顶部的设计，以下是简化版的设计图，请根据提示完成相应的代码。

左边的 logo 区宽度为 100px，高度为：35px，边框宽度为 2px，且绿色。

右边的功能选项区宽度为 200px，高度为：35px，边框宽度为 2px，且绿色。

```
<!DOCTYPE html>
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 <title>Float</title>
</head>
<body>
 <div style="border:2px red solid;width:500px;height:40px;padding:5px;">
 <!-- code -->
 </div>
</body>
</html>
```

答案：

这里是 logo

这里是功能选项

解析：

使用 float 可实现 div 在页面左右两边排列。

35.【单选题】【CSS】阅读下面 HTML 代码，如果期望 tabs 位于 box 容器的右下角，例如以下的效果图，则需要添加的 CSS 样式是哪项（）

```
<div id="box"><div id="tabs"></div></div>
```

A、 #tabs { position:absolute; right:0; bottom:0; }

B、 #tabs { position:relative; right:0; bottom:0; }

C、 #box { position:relative; } #tabs { position:absolute; right:0; bottom:0; }

D、 #box { position:relative; } #tabs { position:right bottom; }

答案：C

解析：

D 选项中的 position:right bottom 没有这样的写法。

题目可以这样理解，tabs 相对于 box 位于 box 的右下角，需要将大的 DIV 设为 relative，小的 DIV 设为 absolute 才能达到预期效果。

36.【编程题】【CSS】请运用绝对定位的方式，将太阳向下移动 20px，向右移动 20px。达到图片中的效果。编辑区已给出部分代码，请在"???"处完善。

```
<!DOCTYPE html>
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Position-相对定位</title>
</head>
<body>
 <div>
      <img src="img1.jpg" style="width: 1000px;height: 500px;">
      <img src="img2.jpg" style="position:???;???:???;???:???; width: 150px;height: 150px">
</div>
</body>
</html>
```

答案：

position:absolute; top:20px; left:20px ;

解析：

position 中的 absolute 实现绝对定位。

37.【单选题】【CSS】实现背景平铺效果，对应的 CSS 为哪项（）

A、div{backgroud-image:url(images/bg.gif);}

B、div{backgroud-image:url(images/bg.gif) repeat-x;}

C、div{backgroud-image:url(images/bg.gif) repeat-y;}

D、div{backgroud-image:url(images/bg.gif) no-repeat;}

答案：A

解析：

repeat-x 表示背景图像将在水平方向重复；

repeat-y 表示背景图像将在垂直方向重复；

no-repeat 表示背景图像将仅显示一次；

repeat(不写即默认为 repeat)表示背景图像将在垂直方向和水平方向重复。

38.【单选题】【CSS】阅读下面 HTML 代码，段落标签<p>内文本最终显示的颜色是哪项（）

```
<style type="text/css">
body { color:#333; }
#text{ color:#666; }
.content { color:#00f; }
.gray { color:#f00; }
</style>
......
<p id="text" class="content gray">我本将心向明月，奈何明月照沟渠。</p>
......
```

A、#f00

B、#00f

C、#666

D、#333

答案：C

解析：

层叠优先级：浏览器缺省 < 外部样式表 < 内部样式表 < 内联样式。

选择器优先级：通用选择器 < 标签选择器 < 类选择器 < 类派生选择器 < ID 选择器 < ID 派生选择器。

repeat-y 表示背景图像将在垂直方向重复；

no-repeat 表示背景图像将仅显示一次；

repeat(不写即默认为 repeat)表示背景图像将在垂直方向和水平方向重复。

39.【编程题】【CSS】使用 html 和 css 知识完成横向的二级导航，并且为第一个和最后一个导航项设置圆角属性（border-radius），鼠标放在二级标签上背景颜色会变红色。

效果如下图：

代码框架：

```
<!DOCTYPE html>
<html>
 <head>
        <meta charset="UTF-8">
        <title>HTML 与 CSS</title>
 </head>
 <body>
            <!--code-->
 </body>
</html>
```

答案：

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
```

```
<title>导航</title>
  <style type="text/css">
      ul{
          list-style: none;
      }
      .nav>li{
       float: left;
      }
      ul a{
          display: block;
          text-decoration: none;
          width: 100px;
          height: 50px;
          text-align: center;
          line-height: 50px;
          color: white;
          background-color: #2f3e45;
      }
      .nav>li:first-child a{
          border-radius: 10px 0 0 10px;
      }
      .nav>li:last-child a{
          border-radius: 0 10px 10px 0;
      }
      .drop-down{
          /*position: relative;*/
      }
      .drop-down-content{
          padding: 0;
          display: none;
          /*position: absolute;*/
      }
      .drop-down-content li:hover a{
          background-color:red;
      }
      .nav .drop-down:hover .drop-down-content{
```

```
                display: block;
            }
    </style>
    </head>
    <body>
        <ul class="nav">
            <li><a href="#">下拉菜单</a></li>
            <li class="drop-down"><a href="#">下拉菜单</a>
                <ul class="drop-down-content">
                    <li><a href="#">我是 1</a></li>
                   <li><a href="#">我是 2</a></li>
                    <li><a href="#">我是 3</a></li>
                </ul>
                </li>
            <li><a href="#">下拉菜单</a></li>
            <li><a href="#">下拉菜单</a></li>
            <li><a href="#">下拉菜单</a></li>
        </ul>
    </body>
    </html>
```

解析：

二级导航正常显示；第一个和最后一个导航项添加圆角属性；鼠标放在二级标签上背景颜色会变色。

40.【编程题】【CSS】下图是一个购物网站的商品选购页面，现在我们要运用盒子模型的知识完成代码开发，注意内外边距及边框。

代码框架：

```
<!DOCTYPE html>
<html>
 <head>
        <meta charset="UTF-8">
        <title>讯飞教育</title>
 </head>
 <body>
      <!--code-->
 </body>
</html>
```

答案：

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>css 盒子模型</title>
    <meta http-equiv="content-type" content="text/html;charset=UTF-8">
  <style type="text/css">
    /*去掉边距和填充*/
    body {
      margin: 0px;
      padding: 0px;
    }

    /*最外围的 div, 控制显示的位置*/
    .outerDiv {
      width: 480px;
      height: 410px;
      /*border: 1px solid gray;*/
      margin-left: 50px;
      margin-top: 50px;
    }

    /*中间的边框 ul, 控制显示图片区域的宽度和高度*/
    .faceul {
      width: 405px;
      height: 385px;
      /*border: 1px solid red;*/
      list-style-type: none; /*去掉列表项的标识*/
      padding: 0;
      margin: 10px auto;
    }

    /*控制单个图片区域的大小*/
    .faceul li {
      width: 120px;
      height: 170px;
      /*border: 1px solid blue;*/
      float: left;   /*左浮动, 让列表水平放置*/
      margin-left: 10px; /*控制图片间距*/
```

```
    margin-top: 10px;
    text-align: center; /*让超链接和图片居中*/

}

/*对图片进行控制，原始大小  120X120 */
.faceul img {
    width: 110px;
    /*margin-left: 5px;*/
    margin-top: 5px;
    margin-bottom: 5px; /*调整与超链接的间距*/
    /*border: 1px solid red;*/
}

.faceul a {
    font-size: 8px;
    /*margin-left: 40px;*/
    /*margin-bottom: 0px;*/
    margin-top: 115px;
    /*border: 1px solid black;*/
  text-align: left;

}
/*金额显示*/
span{
        display: block;/*使金额独占一行*/
        color: red;
        font-size: 15px;
        text-align: left;
}

a:link {
    text-decoration: none;
    color: black;
}
a:hover {
```

```
            font-size: 8px;
            text-decoration: underline;
            color: blue;
        }
        b{
            color: red;
        }
    </style>
</head>
<body>
    <div class="outerDiv">
        <ul class="faceul">
            <li><img src="img/1.jpg"/><span>￥4299</span><a href="#">布艺<b>沙发</b>实木家具组合布艺胡桃木中式家具</a></li>
            <li><img src="img/2.jpg"/><span>￥3299</span><a href="#">中式客厅实木布艺<b>沙发</b>转角木<b>沙发</b>组合多功能</a></li>
            <li><img src="img/3.jpg"/><span>￥3499</span><a href="#">透气棉麻客厅家具套装，灰白色，三人位<b>沙发</b></a></li>
            <li><img src="img/4.jpg"/><span>￥6500</span><a href="#">客厅<b>沙发</b>小户型北欧<b>沙发</b></a></li>
            <li><img src="img/5.jpg"/><span>￥6200</span><a href="#">中式客厅实木布艺<b>沙发</b>转角木<b>沙发</b>组合多功能</a></li>
            <li><img src="img/6.jpg"/><span>￥7800</span><a href="#">布艺<b>沙发</b>实木家具组合布艺胡桃木中式家具</a></a></li>
        </ul>
    </div>
</body>
</html>
```

解析：

页面显示正常；商品的排版与图片中的类似；

41.【简答题】【CSS】将多个元素设置为同一行？清除浮动有几种方式？

将多个元素设置为同一行：float，inline-block

清除浮动的方式：方法一：添加新的元素 、应用 clear：both；

方法二：父级 div 定义 overflow: hidden；

方法三：利用:after 和:before 来在元素内部插入两个元素块，从面达到清除浮动的效果。

.clear{zoom:1;}

.clear:after{content:””;clear:both;display:block;height:0;overflow:hidden;visibility:hidden;}

42.【简答题】【CSS】怪异盒模型 box-sizing？弹性盒模型|盒布局？

在标准模式下的盒模型：盒子总宽度/高度=width/height+padding+border+margin

在怪异模式下的盒模型下，盒子的总宽度和高度是包含内边距 padding 和边框 border 宽度在内的，盒子总宽度/高度=width/height + margin = 内容区宽度/高度 + padding + border + margin;

box-sizing 有两个值一个是 content-box，另一个是 border-box。

当设置为 box-sizing:content-box 时，将采用标准模式解析计算；

当设置为 box-sizing:border-box 时，将采用怪异模式解析计算。

43.【简答题】【CSS】transform、animation、animation-duration 区别？

Transform:它和 width、left 一样，定义了元素很多静态样式实现变形、旋转、缩放、移位及透视等功能，通过一系列功能的组合我们可以实现很炫酷的静态效果（非动画)。

Animation:作用于元素本身而不是样式属性,属于关键帧动画的范畴，它本身被用来替代一些纯粹表现的 javascript 代码而实现动画,可以通过 keyframe 显式控制当前帧的属性值.

animation-duration：规定完成动画所花费的时间，以秒或毫秒计。

44.【简答题】【CSS】nth-of-type | nth-child 作用？

举例说明：

 111 222 1 2 3

li:nth-of-type(2):表示 ul 下的第二个 li 元素

li:nth-child(2)：表示既是 li 元素又是在 ul 下的第二个元素（找不到）。

建议一般使用 nth-of-type，不容易出问题。

45.【简答题】【CSS】:before 和 ::before 区别？

单冒号(:)用于 CSS3 伪类，双冒号(::)用于 CSS3 伪元素。 对于 CSS2 之前已有的伪元素，比如:before，单冒号和双冒号的写法::before 作用是一样的。

46.【简答题】【CSS】清除浮动的几种方式，各自的优缺点。

结尾处加空 div 标签 clear:both

父级 div 定义 height

父级 div 定义伪类 :after 和 zoom

父级 div 定义 overflow:hidden

父级 div 也浮动，需要定义宽度

结尾处加 br 标签 clear:both

47.【简答题】【CSS】为什么要初始化 CSS 样式？

因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对 CSS 初始化往往会出现浏览器之间的页面显示差异。

当然，初始化样式会对 SEO 有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化。

48.【简答题】【CSS】介绍一下标准的 CSS 的盒子模型？低版本 IE 的盒子模型有什么不同的？

有两种， IE 盒子模型、 W3C 盒子模型；

盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border)；

区别： IE 的 content 部分把 border 和 padding 计算了进去;

49.【简答题】【CSS】CSS 优先级算法如何计算？

优先级就近原则，同权重情况下样式定义最近者为准载入样式以最后载入的定位为准

优先级为: !important > id > class > tag ; !important 比内联优先级高

50.【简答题】【CSS】position 的值，relative 和 absolute 定位原点是？

absolute ：生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位

fixed ：生成绝对定位的元素，相对于浏览器窗口进行定位

relative ：生成相对定位的元素，相对于其正常位置进行定位

static 默认值。没有定位，元素出现在正常的流中

inherit 规定从父元素继承 position 属性的值

51.【简答题】【CSS】行内元素 float:left 后是否变为块级元素？

行内元素设置成浮动之后变得更加像是 inline-block （行内块级元素，设置成这个属性的元素会同时拥有行内和块级的特性，最明显的不同是它的默认宽度不是 100% ），这时候给行内元素设置 padding-top 和 padding-bottom 或者 width 、 height 都是有效果的。

52.【简答题】【CSS】CSS 不同选择器的权重(CSS 层叠的规则)。

！important 规则最重要，大于其它规则

行内样式规则，加 1000

对于选择器中给定的各个 ID 属性值，加 100

对于选择器中给定的各个类属性、属性选择器或者伪类选择器，加 10

对于选择其中给定的各个元素标签选择器，加 1

如果权值一样，则按照样式规则的先后顺序来应用，顺序靠后的覆盖靠前的规则

53.【简答题】【CSS】stylus/sass/less 区别？

均具有“变量”、“混合”、“嵌套”、“继承”、“颜色混合”五大基本特性

Scss 和 LESS 语法较为严谨， LESS 要求一定要使用大括号“{}”， Scss 和 Stylus 可以通过缩进表示层次与嵌套关系

Scss 无全局变量的概念， LESS 和 Stylus 有类似于其它语言的作用域概念

Sass 是基于 Ruby 语言的，而 LESS 和 Stylus 可以基于 NodeJS NPM 下载相应库后进行编译；

54.【简答题】【CSS】postcss 的作用？

可以直观的理解为：它就是一个平台。为什么说它是一个平台呢？因为我们直接用它，感觉不能干什么事情，但是如果让一些插件在它上面跑，那么将会很强大

PostCSS 提供了一个解析器，它能够将 CSS 解析成抽象语法树通过在 PostCSS 这个平台上，我们能够开发一些插件，来处理我们的 CSS ，比如热门的： autoprefixer

postcss 可以对 sass 处理过后的 css 再处理 最常见的就是 autoprefixer

55.【简答题】【CSS】行内元素和块级元素的区别是是什么？

行内元素：

（1）行内元素不换行

（2）行内元素不可以设置大小

（3）行内元素大小由内容决定

块元素：

（1）块元素独立成行

（2）块元素可以设置大小

（3）块元素如果不设置宽度，宽度会自适应其父级的宽度

56.【简答题】【CSS】使块元素在一行显示？

方法 1:float : left

方法 2:在父级元素设置 display:flex

57.【简答题】【CSS】设置 css 方式有哪些？

行内样式、内部样式（写一个 style）、外部样式

58.【简答题】【CSS】 实现一个函数 clone，可以对 JavaScript 中的 5 种主要的数据类型（包括 Number、String、Object、Array、Boolean）进行值复制。

考察点 1：对于基本数据类型和引用数据类型在内存中存放的是值还是指针这一区别是否清楚

考察点 2：是否知道如何判断一个变量是什么类型的

考察点 3：递归算法的设计

// 方法一：

```
Object.prototype.clone = function(){
var o = this.constructor === Array ? [] : {};
for(var e in this){
o[e] = typeof this[e] === "object" ? this[e].clone() : this[e];
}
return o;
}
```

//方法二：

```
/**
* 克隆一个对象
* @param Obj
* @returns
*/
```

```
function clone(Obj) {
var buf;
if (Obj instanceof Array) {
buf = [];//创建一个空的数组
var i = Obj.length;
while (i--) {
buf[i] = clone(Obj[i]);
}
return buf;
}else if (Obj instanceof Object){
buf = {};//创建一个空对象
for (var k in Obj) { //为这个对象添加新的属性
buf[k] = clone(Obj[k]);
}
return buf;
}else{ //普通变量直接赋值
return Obj;
}
}
```

59.【简答题】【CSS】px 和 em 的区别？

px 和 em 都是长度单位，区别是，px 的值是固定的，指定是多少就是多少，计算比较容易。 em 得值不是固定的，并且 em 会继承父级元素的字体大小。 浏览器的默认字体高都是 16px。所以未经调整的浏览器都符合: 1em=16px。那么 12px=0.75em, 10px=0.625em。

60.【简答题】【CSS】 CSS 的盒子模型分为哪两种？

IE 盒子模型、标准 W3C 盒子模型；IE 的 content 部分包含了 border 和 pading。

61.【简答题】【CSS】超链接访问过后 hover 样式就不出现的问题是什么？如何解决？

被点击访问过的超链接样式不在具有 hover 和 active 了,解决方法是改变 CSS 属性的排列顺序: L-V-H-A（link,visited,hover,active）

62.【简答题】【CSS】有哪项方式可以对一个 DOM 设置它的 CSS 样式？

外部样式表， 引入一个外部 css 文件；

内部样式表， 将 css 代码放在 <head> 标签内部；

内联样式， 将 css 样式直接定义在 HTML 元素内部。

63.【简答题】【CSS】CSS 都有哪些选择器？

派生选择器（ 用 HTML 标签申明）

id 选择器（ 用 DOM 的 ID 申明）

类选择器（ 用一个样式类名申明）

属性选择器（ 用 DOM 的属性申明， 属于 CSS2， IE6 不支持， 不常用， 不知道就算了）

除了前 3 种基本选择器， 还有一些扩展选择器， 包括

后代选择器（ 利用空格间隔， 比如 div .a{ }）

群组选择器（ 利用逗号间隔， 比如 p,div,#a{ }）

那么问题来了， CSS 选择器的优先级是怎么样定义的？

基本原则：

一般而言， 选择器越特殊， 它的优先级越高。 也就是选择器指向的越准确， 它的优先级就越高。

64.【简答题】【CSS】行内元素和块级元素的具体区别是什么？行内元素的 padding 和 margin 可设置吗？

块级元素(block)特性：

总是独占一行， 表现为另起一行开始， 而且其后的元素也必须另起一行显示;

宽度(width)、 高度(height)、 内边距(padding)和外边距(margin)都可控制;

内联元素(inline)特性：

和相邻的内联元素在同一行;

宽度(width)、 高度(height)、 内边距的 top/bottom(padding-top/padding-bottom)和外边距的 top/bottom(margin-top/margin-bottom)都不可改变（ 也就是 padding 和 margin 的 left 和 right 是可以设置的）， 就是里面文字或图片的大小。

65.【简答题】【CSS】什么是外边距重叠？重叠的结果是什么？

外边距重叠就是 margin-collapse。

在 CSS 当中， 相邻的两个盒子（ 可能是兄弟关系也可能是祖先关系） 的外边距可以结合成一个单独的外边

距。 这种合并外边距的方式被称为折叠，并且因而所结合成的外边距称为折叠外边距。

折叠结果遵循下列计算规则：

两个相邻的外边距都是正数时，折叠结果是它们两者之间较大的值。

两个相邻的外边距都是负数时，折叠结果是两者绝对值的较大值。

两个外边距一正一负时，折叠结果是两者的相加的和。

66.【简答题】【CSS】rgba()和 opacity 的透明效果有什么不同？

rgba()和 opacity 都能实现透明效果， 但最大的不同是 opacity 作用于元素， 以及元素内的所有内容的透明度，而 rgba()只作用于元素的颜色或其背景色。 （设置 rgba 透明的元素的子元素不会继承透明效果！ ）

67.【简答题】【CSS】CSS 中可以让文字在垂直和水平方向上重叠的两个属性是什么？

垂直方向： line-height

水平方向： letter-spacing

关于 letter-spacing 的妙用，可以用于消除 inline-block 元素间的换行符空格间隙问题。

68.【简答题】【CSS】你能描述一下渐进增强和优雅降级之间的不同吗？

渐进增强 progressive enhancement： 针对低版本浏览器进行构建页面， 保证最基本的功能， 然后再针对高级浏览器进行效果、 交互等改进和追加功能达到更好的用户体验。

优雅降级 graceful degradation： 一开始就构建完整的功能， 然后再针对低版本浏览器进行兼容。

区别： 优雅降级是从复杂的现状开始， 并试图减少用户体验的供给， 而渐进增强则是从一个非常基础的，能够起作用的版本开始， 并不断扩充， 以适应未来环境的需要。 降级（功能衰减） 意味着往回看； 而渐进增强则意味着朝前看， 同时保证其根基处于安全地带。“优雅降级”观点

“优雅降级” 观点认为应该针对那些最高级、 最完善的浏览器来设计网站。 而将那些被认为“过时” 或有功能缺失的浏览器下的测试工作安排在开发周期的最后阶段， 并把测试对象限定为主流浏览器（ 如 IE、 Mozilla 等） 的前一个版本。

在这种设计范例下， 旧版的浏览器被认为仅能提供“简陋却无妨 (poor, but passable)”的浏览体验。 你可以做一些小的调整来适应某个特定的浏览器。 但由于它们并非我们所关注的焦点， 因此除了修复较大的错误之外，其它的差异将被直接忽略。

“渐进增强” 观点

“渐进增强” 观点则认为应关注于内容本身。

内容是我们建立网站的诱因。 有的网站展示它， 有的则收集它， 有的寻求， 有的操作， 还有的网站甚至会包含以上的种种， 但相同点是它们全都涉及到内容。 这使得“渐进增强”成为一种更为合理的设计范例。 这也

是它立即被 Yahoo! 所采纳并用以构建其“ 分级式浏览器

支持 (Graded Browser Support)” 策略的原因所在。

69.【简答题】【JavaScript】明确等于操作符（==）与严格等于操作符（===）的区别和使用场景，下面结果为 false 的是（）

A.null == underfined

B.+0 === -0

C.NaN === NaN

D.true === true

解析：

答案 C。undefined 和 null 与其他类型的值比较时，结果都为 false，它们互相比较时结果为 true。

在严格相等运算符中（===）

两个值都为数值且值相等 true

两个值其中之一为 NaN false

两个值都是 null/undefined/true/false true

70.【编程题】【JavaScript】使用运算符、流程控制语句完成逻辑运算。

有两个班级，分别为 classOne、classTwo，已知班级中每个同学的年龄（用数组表示），比较两个班级的平均年龄，并输出平均年龄较大的班级名称与平均年龄。

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>平均年龄对比</title>
</head>
<body>
<script type="text/javascript">
    var classOne = [18,19,20,21,19,17,18,19,20,19,22,19];
    var classTwo = [19,17,21,19,19,19,20,17,19,20];
//     input your code
</script>
</body>
```

```
</html>
```

答案：

```
var countOne = 0, countTwo = 0;
for (var i = 0; i < classOne.length; i++) {
    countOne += classOne[i];
}
for (var i = 0; i < classTwo.length; i++) {
    countTwo += classTwo[i];
}
var averageOne = countOne / classOne.length;
var averageTwo = countTwo / classTwo.length;
if (averageOne > averageTwo) {
    console.log('平均年龄较大班级为 classOne，平均值为：' + averageOne );
} else {
    console.log('平均年龄较大班级为 classTwo，平均值为：' + averageTwo );
}
```

解析：

从 0 开始遍历依次相加到 sum，当 i=11 时 for 循环结束

71.【编程题】【JavaScript】要求实现输入某年某月某日，判断这一天是这一年的第几天。

请在 code1、2、3 处填入合适的代码，在填写时请删除/*code*/。

代码框架：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>计算某一天</title>
</head>
<body>
<script type="text/javascript">
    var y=parseInt(prompt("输入年份"));
```

```
var m=parseInt(prompt("输入月份"));
var d=parseInt(prompt("输入日期"));
/*输入的月份小于等于 12，日期小于等于 31*/
if (m<=12 && d<=31)
{
    var day=0;
    switch(m)
    {
        case 1:{day=0;break;}
        case 2:{day=31;break;}
        case 3:{day=59;break;}
        case 4:{day=90;break;}
        case 5:{day=120;break;}
        case 6:{day=151;break;}
        case 7:{day=181;break;}
        case 8:{day=212;break;}
        case 9:{day=243;break;}
        case 10:{day=273;break;}
        case 11:{day=304;break;}
        case 12:{day=334;break;}
    }
    day=day+d;
    /*平年的二月不能输入 29、30、31 号*/
    if (/*code 1*/)
    {
        document.write("您输入有误！");
    }
    /*判断输入的年份是闰年还是平年*/
    if(/*code 2*/)
    {
        /*闰年*/
        /*code 3*/
        document.write(y+"年"+m+"月"+d+"日是一年中的第"+day+"天。");
    }else {
        /*平年*/
        document.write(y+"年"+m+"月"+d+"日是一年中的第"+day+"天。");
```

```
        }
    }else {
        /*输入的月份大于 12 或日期大于 31*/
        document.write("您的输入有误！");
    }
```

答案：

```
/*平年的二月不能输入 29、30、31 号*/
if (y%4!=0 && m==2 && d==29 || d==30 || d==31)
{
    document.write("您输入有误！");
}
/*判断输入的年份是闰年还是平年*/
if(((y%4==0&&y%100!=0)||(y%400==0))&&(m>2))
{
    /*闰年*/
    day=day+1;
    document.write(y+"年"+m+"月"+d+"日是一年中的第"+day+"天。");
}else {
    /*平年*/
    document.write(y+"年"+m+"月"+d+"日是一年中的第"+day+"天。");
}
```

解析：

无

72.【编程题】【JavaScript】准确判断对象的类型（使用 Object. prototype. toString）

准确的判断 arr 是否为一个数组（Array），并返回 bool 值。

```
<!DOCTYPE html>
<html>
 <head>
    <meta charset=utf-8" />
    <title>数组判断</title>
 </head>
```

```
 <body>
        <script>
        var a=[];
        //判断 arr 是否为一个数组，返回一个 bool 值
        function isArray(arr) {
        }
        alert(isArray(a))
        </script>
 </body>
</html>
```

答案：

return Object.prototype.toString.call(arr) === '[object Array]';

解析：

数组（Array）是复杂数据类型 Object 中的一类，需要使用 Object 自带的方法来判断（Object.prototype.toString）

73.【编程题】【JavaScript】对数组进行去重操作，只考虑数组中元素为数字或字符串，返回一个去重后的数组。

代码框架：

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>数组去重</title>
</head>
<body>
<script type="text/javascript">
    function uniqArray(arr) {
        <!-- code -->
}
</script>
</body>
</html>
```

答案：

```
if (arr.length < 2) return arr;
for (var i = 0; i < arr.length - 1; i++) {
    for (var j = i + 1; j < arr.length; j++) {
        if (arr[i] == arr[j]) {
            arr.splice(j, 1);    // 数组的 splice 方法：删除数组中的某一个元素
        }
    }
}
return arr;
```

74.【编程题】【JavaScript】使用正则表达式完成特定数据的验证在下划线出写出邮箱的正则表达式，验证邮箱弹出 true。

```
<!DOCTYPE html>
<html>
 <head>
    <meta charset=utf-8" />
    <title>邮箱验证</title>
 </head>
 <body>
    <script>
    // 判断是否为邮箱地址
    function isEmail(emailStr) {
        // 开头必须以字母和数字跟着 1 一个“@”，后边是小写字母或数字，最后就是域名（2-4 位）。
        var re=_____;
        return re.test(emailStr);
    }
    alert(isEmail('dangjingtao@ds.cc'));
    </script>
 </body>
</html>
```

答案：

/^\w+@[a-z0-9]+\.[a-z]{2,4}$/

75.【编程题】【JavaScript】以下代码中 weatherStr 是天气预报对应的 json 字符串，请将 json 字符串转换为 JavaScript 对象，并在页面上打印出 forecast 中的日期与天气数据。

代码框架：

```
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <title>天气预报</title>
</head>
<body>
<script>
    var weatherStr = '{"data":{"city":"北京","forecast":[{"date":"1 月 1 日","type":"晴"},{"date":"1 月 2 日","type":"多云"}],"temperature":"12"},"status":1000,"desc":"OK"}';
//      input your code
</script>
</body>
</html>
```

答案：

```
var weather = JSON.parse(weatherStr);
for (var i = 0; i < weather.data.forecast.length; i ++){
    var oneDay = weather.data.forecast[i];
    document.write(oneDay.date + ': ' + oneDay.type + '<br>');
}
```

76.【多选题】【JavaScript】文档对象模型中，以下说法正确的是（）

A、节点类型包含节点属性、文本节点、元素节点

B、文档对象模型中的所有节点组成了一个节点树

C、DOM 定义了 JavaScript 操作 HTML 标签的方法

D、元素节点的 nodeValue 返回值为 Undefined

答案：B、C

解析：

节点类型包含属性节点、文本节点、元素节点；元素节点的 nodeValue 返回值为 Null

77.【编程题】【JavaScript】在大部分网站主页，都会有一个导航栏菜单，帮助用户完成功能的查找，从而选择自己需要的功能。请在本题中实现效果，当鼠标移入不同的选项，会弹出对应的下拉菜单；当鼠标移出，下拉菜单会隐藏（只需完成 DOM 操作，CSS 相关样式已给出）。

效果如下图：

代码框架：

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="UTF-8">
 <title>导航栏</title>
 <style>
        * {
            margin: 0;
            padding: 0;
        }

        a {
            text-decoration: none;
        }

        .nva {
            width: 100%;
            height: 40px;
```

```
        margin-top: 70px;
        background-color: #222;
}

.list {
        width: 80%;
        height: 40px;
        margin: 0 auto;
        list-style-type: none;
}

.list li {
        float: left;
        text-align: center;
        width: 110px;
}

.list li a {
        padding: 0 30px;
        color: #aaa;
        line-height: 40px;
        display: block;
}

.list li a:hover {
        background: #333;
        color: #fff;
}

.list li a.on {
        background: #333;
        color: #fff;
}

.menu {
        background-color: #222;
```

```
            display: none;
        }
    </style>
    <script>
        window.onload = function () {
            //在此处编码
        }
    </script>
</head>

<body>
    <header class="header">
        <div class="nva">
            <ul class="list">
                <li>
                    <a href="#">
                        <b>首页</b>
                    </a>
                </li>
                <li id="a">
                    <a href="#">
                        <b>Android</b>
                    </a>
                    <div id="androidMeun" class="menu">
                        <a href="#">选项一</a>
                        <a href="#">选项二</a>
                        <a href="#">选项三</a>
                    </div>
                </li>
                <li>
                    <a href="#">
                        <b>C++</b>
                    </a>
                    <div id="cMeun" class="menu">
                        <a href="#">选项一</a>
                        <a href="#">选项二</a>
```

```
                    <a href="#">选项三</a>
                </div>
            </li>
            <li>
                <a href="#">
                    <b>iOS</b>
                </a>
                <div id="iosMeun" class="menu">
                    <a href="#">选项一</a>
                    <a href="#">选项二</a>
                    <a href="#">选项三</a>
                </div>
            </li>
            <li>
                <a href="#">
                    <b>Java</b>
                </a>
                <div id="javaMeun" class="menu">
                    <a href="#">选项一</a>
                    <a href="#">选项二</a>
                    <a href="#">选项三</a>
                </div>
            </li>
            <li>
                <a href="#">
                    <b>Ruby</b>
                </a>
                <div id="rubyMeun" class="menu">
                    <a href="#">选项一</a>
                    <a href="#">选项二</a>
                    <a href="#">选项三</a>
                </div>
            </li>
        </ul>
    </div>
</header>
```

```
</body>
</html>
```

答案：

```
var selectionArrays = document.getElementsByTagName('li');
var menuArrays = document.getElementsByClassName('menu');

for (var i = 0, j = menuArrays.length; i < j; i++) {

    (function (i) {
        selectionArrays[i + 1].onmouseover = function () {
            menuArrays[i].style.display = 'block';
        }

        selectionArrays[i + 1].onmouseout = function () {
            menuArrays[i].style.display = 'none';
        }
    })(i);
}
```

78.【编程题】【JavaScript】在网站中展示大量数据时，我们通常需要使用表格。为了更快的选出目标数据，当鼠标移动到某一行数据上时，改变它的背景颜色；当鼠标移出该行，其背景色复原。

。效果如图：

代码框架：

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
 <meta charset="UTF-8">
 <title>表格事件</title>
 <style type="text/css">
        .style0 {
            background-color: rgb(57, 139, 251);
        }

        .style1 {
            background-color: rgb(225, 225, 225);
        }
 </style>
 <script type="text/javascript">
        window.onload = function () {
            //在此处编写代码
        }
 </script>
</head>
<body>
 <table width="576" height="79" border="1">
        <tr>
            <td class="style1" align="center">数据 1</td>
        </tr>
        <tr>
            <td class="style1" align="center">数据 2</td>
        </tr>
        <tr>
            <td class="style1" align="center">数据 3</td>
        </tr>
 </table>
</body>
</html>
```

答案：

```
    for (var i = 0; i < 3; i++) {                    document.getElementsByTagName("td")[i].onmousemove =
function () {
```

```
                this.className = 'style0';
            }
 document.getElementsByTagName("td")[i].onmouseout = function () {
                this.className = 'style1';
            }
        }
```

79.【编程题】【JavaScript】选中任意一个 select 的选项，并用弹窗显示选中的值。

代码框架：

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
 <title>绑定事件监听函数</title>
 <script type="text/javascript">
window.onload = function () {
//在此处编写代码
    }
 </script>
</head>
<body>
 <select id="select-some">
            <option>第一个选项</option>
            <option>第二个选项</option>
            <option>第三个选项</option>
    </select>
</body>
</html>
```

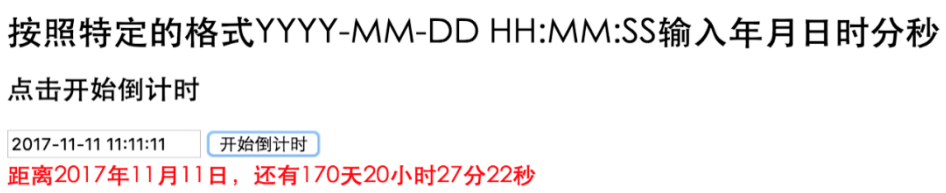
答案：

```
window.onload = function () {
        var selectSome = document.getElementById("select-some");
        selectSome.addEventListener("change", function () {
            alert(this.value);
        }, false);
```

```
    }
```

80.【编程题】【JavaScript】在购物网站中，每当到了节假日就会有抢购活动，并且在网站做一个抢购倒计时，提醒用户距离抢购还有多久。请用 JavaScript 中 window 相关方法实现倒计时功能。

效果如下图：



代码框架：

```
<!DOCTYPE html>
<html>

<head>
 <meta charset="UTF-8">
 <title>task0002</title>
 <style>
        body {
            padding: 40px;
        }

        div {
            font-size: 1em;
            line-height: 1.5;
            color: red;
            font-family: "微软雅黑";
            font-weight: 700;
        }
 </style>
 <script type="text/javascript">
        window.onload = function () {
            function clickStart() {
```

```
                    //在此处编码
                    var show = document.getElementById("showdiv");
                    var time = document.getElementById("time").value.match(/(^\d{4})-(\d{2})-
(\d{2})\s(\d{2}):(\d{2}):(\d{2}$)/); //获取年月日的正则方法

                    if (time != null) {
                        var setTime = new Date(time[1], time[2] - 1, time[3], time[4], time[5], time[6]); //设置目标
时间 new Date(yyyy,mth,dd,hh,mm,ss);

    //在此处编写代码

                    } else {
                        show.innerHTML = "输入有误,请按指定格式输入";
                    }

                }

                function start() {
                    //显示剩余时间的 div
                    document.getElementById("time").value = "2017-11-11 11:11:11";

                    document.getElementById("btn").onclick = clickStart;
                }
                start();
            }
        </script>
    </head>

    <body>
        <h2>按照特定的格式 YYYY-MM-DD HH:MM:SS 输入年月日时分秒</h2>
        <h3>点击开始倒计时</h3>
        <input type="text" id="time">
        <button id="btn">开始倒计时</button>
        <div id="showdiv"></div>
    </body>
    </html>
```

答案：

```
var thisTime = new Date(); //获取到当前的时间。


var date = setTime.getTime() - thisTime.getTime(); //得出相差的时间为毫秒数
var lefttime = parseInt((setTime.getTime() - thisTime.getTime()) / 1000); //得到相差秒数
var d = parseInt(lefttime / (60 * 60 * 24)); //天
var h = parseInt(lefttime / (60 * 60) % 24); //时
var m = parseInt(lefttime / 60 % 60); //分
var s = parseInt(lefttime % 60); //秒
show.innerHTML = "距离" + setTime.getFullYear() + "年" + (setTime.getMonth() + 1) + "月" +
setTime.getDate() +
 "日，还有" + d + "天" + h + "小时" + m + "分" + s + "秒";
setTimeout(clickStart, 1000); //使用回调函数的方式，进行定时执行。
if (date <= 0) { //到达目标
 clearTimeout(clickStart);
 show.innerHTML = "时间已经到了";
}
```

81.【编程题】【JavaScript】实现一个动态列表，点击“添加数据”按钮会在列表末尾增加一条记录。每条记录（包括已有的和新添加的）中都有“删除”按钮，现需要给“删除”按钮添加点击事件，点击“删除”后从列表中删除该记录。

效果图如下：

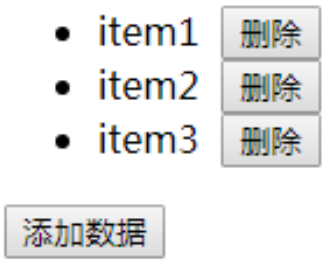


图 1-初始列表

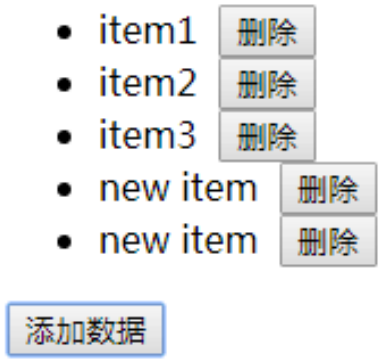


图 2-添加数据

图 3-删除记录

代码框架：

```
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <title>事件流模型</title>
</head>
<body>
<ul id="list">
    <li id="item1">item1  <button>删除</button></li>
    <li id="item2">item2  <button>删除</button></li>
    <li id="item3">item3  <button>删除</button></li>
</ul>
<button onclick="addItem()">添加数据</button>

<script>
    var list = document.getElementById('list');
    function addItem() {
        list.innerHTML +=    '<li>new item  <button>删除</button></li>';
```

```
    }
//input your code
</script>
</body>
</html>
```

答案：

```
list.addEventListener('click', function (event) {
    var ev = event || window.event;
    var target = ev.target || ev.srcElement;
    if (target.nodeName.toLowerCase() == 'button') {
        list.removeChild(target.parentNode);
    }
})
```

解析：

利用事件流模型中事件冒泡的原理，监听<ul>的点击事件。列表中的子级元素被点击，事件能在<ul>上被捕获，从而给所有“删除”按钮添加事件。

82.【简答题】【JavaScript】如何判断某变量是否为数组数据类型？

```
if (typeof Array.isArray === "undefined"){
 Array.isArray = function(arg){
      return Object.prototype.toString.call(arg)==="[object Array]";
 }
}
```

83.【简答题】【JavaScript】JavaScript 的数据类型都有什么？

基本数据类型：Number、String、Boolean、Null、Undefined

复杂数据类型：Object（Function、Array、Date、RegExp）

84.【简答题】【JavaScript】”==”和“===”的不同 ？

前者会自动转换类型

后者不会

85.【简答题】【JavaScript】JavaScript 变量和函数都会声明提升吗？

变量声明和函数声明都会提升，但函数会提升到变量前。

86.【简答题】【JavaScript】js 存储方式。

cookie

sessionStorage

localStorage

indexedDB

87.【简答题】【JavaScript】什么情况下会遇到跨域，怎么解决？

同源策略是浏览器的一个安全功能，不同源的客户端脚本在没有明确授权的情况下，不能读写对方资源。若地址里面的协议、域名和端口号均相同则属于同源。

jsonp 跨域、nginx 反向代理、node.js 中间件代理跨域、后端设置 http header、后端在服务器上设置 cors。

88.【简答题】【JavaScript】Promise 中的执行顺序。

参考阮一峰老师书中的例子

```
let promise = new Promise(function(resolve, reject) {
  console.log('Promise');
  resolve();
});
promise.then(function() {
  console.log('resolved.');
});
console.log('Hi!');
// Promise
// Hi!
// resolved
```

上面代码中，Promise 新建后立即执行，所以首先输出的是 Promise。然后，then 方法指定的回调函数，将在当前脚本所有同步任务执行完才会执行，所以 resolved 最后输出。

89.【简答题】【JavaScript】说说 JavaScript 事件循环机制。

事件循环机制的概念

关键字：单线程非阻塞、执行栈、事件队列、宏任务(setTimeout()、setInterval())、微任务(new Promise())

可参考： zhuanlan.zhihu.com/p/33058983

宏任务、微任务、同步任务的执行顺序

```
setTimeout(function () {
    console.log(1);
});

new Promise(function(resolve,reject){
    console.log(2)
    resolve(3)
}).then(function(val){
    console.log(val);
})
console.log(4);
// 2
// 4
// 3
// 1
```

先按顺序执行同步任务，Promise 新建后立即执行输出 2，接着输出 4，异步任务等同步任务执行完后执行，且同一次事件循环中，微任务永远在宏任务之前执行。这时候执行栈空了，执行事件队列，先取出微任务，输出 3，最后取出一个宏任务，输出 1。

90.【编程题】【JavaScript】for 循环中的作用域问题。

写出以下代码输出值，尝试用 es5 和 es6 的方式进行改进输出循环中的 i 值。

```
for (var i=1; i<=5; i++) {
    setTimeout(function timer() {
        console.log(i);
    }, i*1000);
}
```

解析：输出 5 个 6，因为回调函数在 for 循环之后执行，所有函数共享一个 i 的引用。

es5:

```
for (var i=1; i<=5; i++) {
    (function(j) {
        setTimeout(function timer() {
            console.log(j);
        }, j*1000);
    })(i);
}
```

es6:

```
for (let i=1; i<=5; i++) {
    setTimeout(function timer() {
        console.log(i);
    }, i*1000);
}
```

91.【简答题】【JavaScript】闭包的作用。

闭包的目的是外部函数可以访问内部函数的作用域（局部作用域）。比如访问到内部作用域的变量。

92.【简答题】【JavaScript】原型及原型链。

原型的理解

所有的引用类型（数组、对象、函数），都具有对象特性，即可自由扩展属性（null 除外）

所有的引用类型（数组、对象、函数），都有一个__proto__属性，属性值是一个普通的对象

所有的函数，都有一个 prototype 属性，属性值也是一个普通的对象

所有的引用类型（数组、对象、函数），__proto__属性值指向它的构造函数的 prototype 属性值

原型链的理解

一段代码如下：

```
// 构造函数
function Foo(name, age) {
    this.name = name
}
Foo.prototype.alertName = function () {
    alert(this.name)
}
```

```
// 创建示例
var f = new Foo('zhangsan')
f.printName = function () {
    console.log(this.name)
}
// 测试
f.printName()
f.alertName()
f.toString()
```

因为 f 本身没有 toString()，并且 f.proto（即 Foo.prototype）中也没有 toString。当试图得到一个对象的某个属性时，如果这个对象本身没有这个属性，那么会去它的__proto__（即它的构造函数的 prototype）中寻找。

如果在 f.__proto__中没有找到 toString，那么就继续去 f.proto.__proto__中寻找，因为 f.__proto__就是一个普通的对象而已嘛！

f.__proto__即 Foo.prototype，没有找到 toString，继续往上找 f.proto.__proto__即 Foo.prototype.proto。Foo.prototype 就是一个普通的对象，因此 Foo.prototype.__proto__就是 Object.prototype，在这里可以找到 toString。因此 f.toString 最终对应到了 Object.prototype.toString 这样一直往上找，你会发现是一个链式的结构，所以叫做“原型链”。如果一直找到最上层都没有找到，那么就宣告失败，返回 undefined。最上层是什么 —— Object.prototype.proto === null

93.【简答题】【JavaScript】重绘和回流。

重绘：当页面中元素样式的改变并不影响它在文档流中的位置时（例如：color、background-color、visibility 等），浏览器会将新样式赋予给元素并重新绘制它，这个过程称为重绘。

回流：当 Render Tree（DOM）中部分或全部元素的尺寸、结构、或某些属性发生改变时，浏览器重新渲染部分或全部文档的过程称为回流。

回流要比重绘消耗性能开支更大。

回流必将引起重绘，重绘不一定会引起回流。

94.【简答题】【JavaScript】实现一个深拷贝（思路）。

对象中可能又存在对象，所以需要深拷贝。首先需要知道这是一个递归调用，然后要判断一些特殊类型（数组，正则对象，函数）进行具体的操作，可以通过 Object.prototype.toString.call(obj)进行判断。

95.【简答题】【JavaScript】js 浮点数运算精度问题（0.1+0.2!==0.3）。

比如在 JavaScript 中计算 0.1 + 0.2 时，到底发生了什么呢？

首先，十进制的 0.1 和 0.2 都会被转换成二进制，但由于浮点数用二进制表达时是无穷的，例如。

JavaScript 代码:

0.1 -> 0.0001100110011001...(无限)

0.2 -> 0.0011001100110011...(无限)

IEEE 754 标准的 64 位双精度浮点数的小数部分最多支持 53 位二进制位，所以两者相加之后得到二进制为：

JavaScript 代码: 0.0100110011001100110011001100110011001100110011001100

因浮点数小数位的限制而截断的二进制数字，再转换为十进制，就成了 0.30000000000000004。所以在进行算术计算时会产生误差。

96.【简答题】【JavaScript】new 操作符具体干了什么呢？

创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型

属性和方法被加入到 this 引用的对象中

新创建的对象由 this 所引用，并且最后隐式的返回 this

97.【简答题】【JavaScript】Ajax 原理。

Ajax 的原理简单来说是在用户和服务器之间加了一个中间层(AJAX 引擎)，通过 XmlHttpRequest 对象来向服务器发异步请求，从服务器获得数据，然后用 javascript 来操作 DOM 而更新页面。使用户操作与服务器响应异步化。这其中最关键的一步就是从服务器获得请求数据

Ajax 的过程只涉及 JavaScript 、 XMLHttpRequest 和 DOM 。 XMLHttpRequest 是 ajax 的核心机制

98.【简答题】【JavaScript】那些操作会造成内存泄漏？

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在

setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏闭包使用不当

99.【简答题】【JavaScript】XML 和 JSON 的区别？

数据体积方面

JSON 相对 于 XML 来讲，数据的体积小，传递的速度更快些。

数据交互方面

JSON 与 JavaScript 的交互更加方便，更容易解析处理，更好的数据交互

数据描述方面

JSON 对数据的描述性比 XML 较差

传输速度方面

JSON 的速度要远远快于 XML

100.【简答题】【JavaScript】为什么要有同源限制？

同源策略指的是：协议，域名，端口相同，同源策略是一种安全协议

举例说明：比如一个黑客程序，他利用 Iframe 把真正的银行登录页面嵌到他的页面上，当你使用真实的用户名，密码登录时，他的页面就可以通过 Javascript 读取到你的表单中 input 中的内容，这样用户名，密码就轻松到收了

101.【简答题】【JavaScript】Javascript 有哪些方法定义对象？

对象字面量： var obj = {};

构造函数： var obj = new Object();

Object.create(): var obj = Object.create(Object.prototype);

102.【简答题】【JavaScript】说几条写 JavaScript 的基本规范。

不要在同一行声明多个变量

请使用 ===/!== 来比较 true/false 或者数值

使用对象字面量替代 new Array 这种形式

不要使用全局函数

Switch 语句必须带有 default 分支

If 语句必须使用大括号

for-in 循环中的变量应该使用 var 关键字明确限定作用域，从而避免作用域污染

103.【简答题】【JavaScript】call 和 apply 的区别？

call 和 apply 相同点：

都是为了用一个本不属于一个对象的方法，让这个对象去执行

```
toString.call([],1,2,3)
toString.apply([],[1,2,3])
Object.call(this,obj1,obj2,obj3)
Object.apply(this,arguments)
```

104.【简答题】【JavaScript】b 继承 a 的方法。

function b(){} b.prototype=new a;

105.【简答题】【JavaScript】JavaScript this 指针、闭包、作用域。

this：指向调用上下文

闭包：内层作用域可以访问外层作用域的变量

作用域：定义一个函数就开辟了一个局部作用域，整个 js 执行环境有一个全局作用域

106.【简答题】【JavaScript】如何阻止事件冒泡和默认事件？

e. stopPropagation();//标准浏览器

event.canceBubble=true;//ie9 之前 阻止默认事件：

为了不让 a 点击之后跳转，我们就要给他的点击事件进行阻止

```
return false
e.preventDefault();
```

107.【简答题】【JavaScript】javascript 的本地对象，内置对象和宿主对象？

本地对象为 array obj regexp 等可以 new 实例化

内置对象为 gload Math 等不可以实例化的

宿主为浏览器自带的 document,window 等

108.【简答题】【JavaScript】希望获取到页面中所有的 checkbox 怎么做？(不使用第三方框架)？

```
var domList = document.getElementsByTagName(‘input’)
 var checkBoxList = [];//返回的所有的 checkbox
var len = domList.length; //缓存到局部变量
while (len--) { //使用 while 的效率会比 for 循环更高
if (domList[len].type == ‘checkbox’) {
checkBoxList.push(domList[len]);
}
}
```

109.【简答题】【JavaScript】简述创建函数的几种方式。

第一种（函数声明）：

function sum1(num1,num2){ return num1+num2; }

第二种（函数表达式）：

var sum2 = function(num1,num2){ return num1+num2; }

匿名函数： function(){}:只能自己执行自己

第三种（函数对象方式）：

var sum3 = new Function("num1","num2","return num1+num2");

110.【简答题】【JavaScript】JavaScript 有几种类型的值?

栈：原始数据类型（ Undefined ， Null ， Boolean ， Number 、 String ）

堆：引用数据类型（对象、数组和函数）

两种类型的区别是：存储位置不同；

原始数据类型直接存储在栈(stack)中的简单数据段，占据空间小、大小固定，属于被频繁使用数据，所以放入栈中存储；

引用数据类型存储在堆(heap)中的对象,占据空间大、大小不固定,如果存储在栈中，将会影响程序运行的性能；引用数据类型在栈中存储了指针，该指针指向堆中该实体的起始地址。当解释器寻找引用值时，会首先检索其在栈中的地址，取得地址后从堆中获得实体。

111.【简答题】【JavaScript】例举 3 种强制类型转换和 2 种隐式类型转换。

强制（ parseInt,parseFloat,Number()）

隐式（ ==）

1==” 1” //true

null==undefined//true

112.【简答题】【JavaScript】数组方法 pop() push() unshift() shift()。

push()尾部添加 pop()尾部删除

unshift()头部添加 shift()头部删除

113.【简答题】【JavaScript】事件绑定和普通事件有什么区别？

div1.onclick=function(){};

<button onmouseover=””></button>

1、 如果说给同一个元素绑定了两次或者多次相同类型的事件， 那么后面的绑定会覆盖前面的绑定

2、 不支持 DOM 事件流 事件捕获阶段 目标元素阶段=>事件冒泡阶段

addEventListener

1、 如果说给同一个元素绑定了两次或者多次相同类型的事件， 所有的绑定将会依次触发

2、 支持 DOM 事件流的

3、 进行事件绑定传参不需要 on 前缀

addEventListener(“click”,function(){},true);//此时的事件就是在事件冒泡阶段执行

ie9 开始， ie11 edge： addEventListener

ie9 以前： attachEvent/detachEvent

1、 进行事件类型传参需要带上 on 前缀

2、 这种方式只支持事件冒泡， 不支持事件捕获事件绑定是指把事件注册到具体的元素之上， 普通事件指的是可以用来注册的事件

114.【简答题】【JavaScript】闭包是什么，有什么特性，对页面有什么影响？

闭包就是能够读取其他函数内部变量的函数。

闭包的缺点： 滥用闭包函数会造成内存泄露， 因为闭包中引用到的包裹函数中定义的变量都永远不会被释放， 所以我们应该在必要的时候， 及时释放这个闭包函数

115.【简答题】【JavaScript】foo = foo||bar，这行代码是什么意思？为什么要这样写？

if(!foo) foo = bar; //如果 foo 存在， 值不变， 否则把 bar 的值赋给 foo。

短路表达式： 作为” &&” 和” ||” 操作符的操作数表达式， 这些表达式在进行求值时， 只要最终的结果已经可以确定是真或假， 求值过程便告终止， 这称之为短路求值。

注意 if 条件的真假判定， 记住以下是 false 的情况：空字符串、 false、 undefined、 null、 0

116.【简答题】【JavaScript】看下列代码， 将会输出什么？

代码如下：

```
var foo = 1;
function(){
console.log(foo);
var foo = 2;
console.log(foo);
}
```

输出 undefined 和 2。

上面代码相当于：

```
var foo = 1;
function(){
var foo;
```

```
console.log(foo); //undefined
foo = 2;
console.log(foo); // 2;
}
```

函数声明与变量声明会被 JavaScript 引擎隐式地提升到当前作用域的顶部， 但是只提升名称不会提升赋值部分。

117.【简答题】【JavaScript】谈谈 Cookie 的弊端？

1. Cookie 数量和长度的限制。 每个 domain 最多只能有 20 条 cookie， 每个 cookie 长度不能超过 4KB，否则会被截掉。

2.安全性问题。 如果 cookie 被人拦截了， 那人就可以取得所有的 session 信息。 即使加密也与事无补， 因为拦截者并不需要知道 cookie 的意义， 他只要原样转发 cookie 就可以达到目的了。

3.有些状态不可能保存在客户端。 例如， 为了防止重复提交表单， 我们需要在服务器端保存一个计数器。如果我们把这个计数器保存在客户端， 那么它起不到任何作用。

118.【简答题】【JavaScript】DOM 操作——怎样添加、 移除、 移动、 复制、 创建和查找节点？

1. 创建新节点

createDocumentFragment() // 创建一个 DOM 片段

createElement() // 创建一个具体的元素

createTextNode() // 创建一个文本节点

2. 添加、 移除、 替换、 插入

appendChild()

removeChild()

replaceChild()

insertBefore() // 在已有的子节点前插入一个新的子节点

3. 查找

getElementsByTagName() // 通过标签名称

getElementsByName() // 通过元素的 Name 属性的值(IE 容错能力较强， 会得到一个数组，其中包括 id 等于 name 值的)

getElementById() // 通过元素 Id， 唯一性

119.【简答题】【JavaScript】哪些操作会造成内存泄漏？

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

垃圾回收器定期扫描对象， 并计算引用了每个对象的其他对象的数量。 如果一个对象的引用数量为 0（ 没有其他对象引用过该对象） ， 或对该对象的惟一引用是循环的， 那么该对象的内存即可回收。

1、setTimeout 的第一个参数使用字符串而非函数的话， 会引发内存泄漏。

2.、闭包

3、控制台日志

4、循环（ 在两个对象彼此引用且彼此保留时， 就会产生一个循环）

操作错误不是 bug，是系统的问题，例如超时或者硬件故障。而程序错误（programmer errors）是实际的错误。

120.【简答题】【JavaScript】如何获取查找 DOM 元素？

document.getElementById(idName)

通过 id 查找元素。返回的是元素 DOM，如果页面上有多个相同 ID 的元素，则只会返回第一个元素，不会返回多个（原则上 ID 只有一个，但伟大的程序员们。。。）

document.getElementsByClassName(className)

通过 class 查找。返回的是类数组结构，要想进行 forEach 遍历，需要先转化为数组结构

const doms = document.getElementsByClassName('xxx')

const domsArr = Array.from(doms)

domsArr.forEach(dom=>{})

document.getElementsByTagName(tagName)

更具标签名获取元素，使用方式和 getElementsByClassName 一样。

document.querySelector(selectors) | document.querySelectorAll(selectors)

这两个是唯一支持使用选择器来查找元素的 api，有个这个 api 我们在进行深层次查找的时候方便很多

```
<div class="warp">
    <p>name<p>
    <p>age<p>
<div>
<p>...</p>
<script>
```

// 目标 获取到 warp 下面的 p 元素

1）不使用 querySelectorAll

cons warp = document.getElementsByClassName("warp")[0];

const allp = warp.getElementsByTagName(p)

2）使用 querySelectorAll

const allp = document.querySelectorAll(".warp p")

</script>

querySelector 获取单个元素，querySelectorAll 获取多个元素返回类数组结构

121.【简答题】【JavaScript】如何给 DOM 增加样式？

给元素增加样式

Ele.style.width = xxx

给元素增加 class

```
Ele.className='aaa'   // 设置元素的 class 为 aaa ，如果元素上原本有 class 则会覆盖
Ele.classList.add("aaa") // 给 Ele 新增 aaa
Ele.className += " aaa"   // 给 Ele 新增 aaa
```

行内样式、内部样式（写一个 style）、外部样式=

122.【简答题】【JavaScript】如何复制 DOM 元素？

Ele.cloneNode(true | false)

示例：

```
const box = document.getElementsByClassName("box")[0];
const p = document.createElement("p");
p.innerText = "欢迎关注码不停息微信公众号";
const p2 = p.cloneNode(true); // 复制一个 p  参数 true 标识深度复制，如果 p 里面有子节点也复制过来
box.appendChild(p);
box.appendChild(p2);
```

123.【单选题】【jQuery】如果想在一个指定的元素后添加内容，下面哪个是实现该功能的（）

A、pend(content)

B、pendTo(content)

C、sertAfter(content)

D、ter(content)

答案：D

解析：

after()方法在被选元素后插入指定的内容。

append()方法在被选元素的结尾（仍然在内部）插入指定内容。

appendTo()方法在被选元素的结尾（仍然在内部）插入指定内容。

提示：append()和 appendTo()方法执行的任务相同。不同之处在于：内容和选择器的位置，以及 append()能够使用函数来附加内容。insertAfter()方法在被选元素之后插入 HTML 标记或已有的元素。

124.【简答题】【jQuery】你觉得 jQuery 源码有哪些写的好的地方？

jquery 源码封装在一个匿名函数的执行自执行环境中，有助于防止变量的全局污染，然后通过传入 window 对象参数，可以使 window 对象作为局部变量使用，好处是当 jquery 中访问 window 对象的时候，就不用将作用域链退回到顶层作用域了，从而可以更快的访问 window 对象。同样，传入 undefined 参数，可以缩短查找 undefined 时的作用域链

jquery 将一些原型属性和方法封装在了 jquery.prototype 中，为了缩短名称，又赋值给了 jquery.fn ，这是很形象的写法

有一些数组或对象的方法经常能使用到， jQuery 将其保存为局部变量以提高访问速度 jquery 实现的链式调用可以节约代码，所返回的都是同一个对象，可以提高代码效率

125.【单选题】【HTTP】下列关于 HTTP 协议说法不正确的是（）

A、HTTP 是 Hyper Text Transfer Protocol（超文本传输协议）的缩写

B、HTTP 协议是用于从 WWW 服务器传输超文本到本地浏览器的传送协议

C、HTTP 是一个传输层协议，由请求和响应构成，是一个标准的客户端服务器模型

D、 HTTP 协议不仅保证计算机正确快速地传输超文本文档，还确定传输文档中的哪一部分，以及哪部分内容首先显示(如文本先于图形)

答案：C。

解析：HTTP 协议是应用层协议，不是传输层协议。

126.【单选题】【HTTP】关于 HTTP 协议的理解，下面哪个是正确的（）

A、HTTP 协议是有状态的

B、HTTP 请求返回的 HTTP 状态码中，404 表示服务器无法根据客户端的请求找到资源

C、HTTP 请求返回的 HTTP 状态码中，403 表示临时重定向

D、HTTP 请求返回的 HTTP 状态码中，500 表示服务器不支持请求的功能，无法完成请求

答案：B。

解析：HTTP 协议是无状态的。

HTTP 状态码及其含义如下：

403：服务器理解请求客户端的请求，但是拒绝执行此请求

404：服务器无法根据客户端的请求找到资源（网页）

500：服务器内部错误，无法完成请求

临时重定向是 307；服务器不支持请求的功能，无法完成请求是 501。

127.【单选题】【HTTP】在网址 http://www.baidu.com 中，http 表示（）

A、超文本

B、文件传输协议

C、服务器名称

D、超文本传输协议

答案：D

解析：

http://www.baidu.com 中，http 为采用的协议，此处是采用超文本传输协议；www 表示服务；baidu.com 一起构成了域名，并且 baidu 是标识串，.com 代表了其是一个私营企业。

128.【单选题】【HTTP】以下哪个不是合法的 HTTP 请求方法（）

A、GET

B、POST

C、SET

D、PUT

答案：C

解析：

到目前为止，HTTP 协议中定义了 8 种请求方法：

HTTP1.0 定义了三种请求方法：GET, POST 和 HEAD 方法。

HTTP1.1 新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法

129.【简答题】【HTTP】HTTP 的几种请求方法与用途？

答案：

GET 方法

发送一个请求来取得服务器上的某一资源

POST 方法

向 URL 指定的资源提交数据或附加新的数据

PUT 方法

跟 POST 方法很像，也是想服务器提交数据。但是，它们之间有不同。 PUT 指定了资源在服务器上的位置，而 POST 没有

HEAD 方法

只请求页面的首部

DELETE 方法

删除服务器上的某资源

OPTIONS 方法

它用于获取当前 URL 所支持的方法。如果请求成功，会有一个 Allow 的头包含类似 “GET,POST” 这样的信息

TRACE 方法

TRACE 方法被用于激发一个远程的，应用层的请求消息回路

CONNECT 方法

把请求连接转换到透明的 TCP/IP 通道

130.【简答题】【HTTP】从浏览器地址栏输入 url 到显示页面的步骤？

答案：

浏览器根据请求的 URL 交给 DNS 域名解析，找到真实 IP ，向服务器发起请求；

服务器交给后台处理完成后返回数据，浏览器接收文件（ HTML、JS、CSS 、图象等）；

浏览器对加载到的资源（ HTML、JS、CSS 等）进行语法解析，建立相应的内部数据结构（如 HTML 的 DOM ）；

载入解析到的资源文件，渲染页面，完成。

131.【简答题】【HTTP】HTTP 状态码及其含义？

答案：

1XX ：信息状态码

100 Continue 继续，一般在发送 post 请求时，已发送了 http header 之后服务端将返回此信息，表示确认，之后发送具体参数信息

2XX ：成功状态码

200 OK 正常返回信息

201 Created 请求成功并且服务器创建了新的资源

202 Accepted 服务器已接受请求，但尚未处理

3XX ：重定向

301 Moved Permanently 请求的网页已永久移动到新位置。

302 Found 临时性重定向。

303 See Other 临时性重定向，且总是使用 GET 请求新的 URI 。

304 Not Modified 自从上次请求后，请求的网页未修改过。

4XX ：客户端错误

400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。

401 Unauthorized 请求未授权。

403 Forbidden 禁止访问。

404 Not Found 找不到如何与 URI 相匹配的资源。

5XX: 服务器错误

500 Internal Server Error 最常见的服务器端错误。

503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。

132.【简答题】【HTTP】浏览器从加载到渲染的过程，比如输入一个网址到显示页面的过程。

加载过程：

浏览器根据 DNS 服务器解析得到域名的 IP 地址

向这个 IP 的机器发送 HTTP 请求

服务器收到、处理并返回 HTTP 请求

浏览器得到返回内容

渲染过程：

根据 HTML 结构生成 DOM 树

根据 CSS 生成 CSSOM

将 DOM 和 CSSOM 整合形成 RenderTree

根据 RenderTree 开始渲染和展示

133.【简答题】【HTTP】说说 POST 请求和 GET 请求的区别。

GET 请求能缓存，POST 不能；

POST 相对 GET 更安全，因为 GET 请求都包含在 URL 里，且会被浏览器保存历史纪录，POST 不会，但是在抓包的情况下都是一样的；

POST 可以通过 request body 来传输比 GET 更多的数据，GET 没有这个技术，且 URL 有长度限制，会影响 GET 请求，但是这个长度限制是浏览器规定的，不是 RFC 规定的；

POST 支持更多的编码类型且不对数据类型限制。

134.【简答题】【HTTP】https 用哪些端口进行通信，这些端口分别有什么作用？

443：端口用来验证服务器端和客户端的身份，比如验证证书的合法性；

80：端口用来传输数据（在验证身份合法的情况下，用来数据传输）。

135.【简答题】【HTTP】简述 ajax 的过程？

创建 XMLHttpRequest 对象,也就是创建一个异步调用对象

创建一个新的 HTTP 请求,并指定该 HTTP 请求的方法、URL 及验证信息

设置响应 HTTP 请求状态变化的函数

发送 HTTP 请求

获取异步调用返回的数据

使用 JavaScript 和 DOM 实现局部刷新

136.【简答题】【HTTP】HTTP 和 HTTPS 之间有什么区别？

1.HTTPS 是加密的传输协议，HTTP 是文本传输协议；

2.HTTPS 需要 SSL 证书，但 HTTP 不需要；

3.HTTPS 比 HTTP 更安全，对搜索引擎更友好，并且对 SEO 有益。

4.HTTPS 标准端口 443，HTTP 标准端口 80；

5.HTTPS 基于传输层，HTTP 基于应用程序层；

6.HTTPS 在浏览器中显示绿色的安全锁，但不显示 HTTP；

通常，HTTPS 比 HTTP 更安全，并且可以有效地保护网站用户的隐私。

137.【简答题】【HTTP】在 css/js 代码上线之后开发人员经常会优化性能，从用户刷新网页开始，一次 js 请求一般情况下有哪些地方会有缓存处理？

dns 缓存， cdn 缓存， 浏览器缓存， 服务器缓存

138.【多选题】【Node.js】以下哪些是 Node.js 中的全局对象（）

A、process

B、console

C、module

D、exports

答案：ABCD。

解析：Node.js 中的全局对象有：global、 process、 console、 module 和 exports 等。

139.【简答题】【Node.js】Node.js 与 JavaScript 有什么不同？

JavaScript	Node.js
JavaScript 是一种编程语言，可以在任何具有合适浏览器引擎的网络浏览器中运行。	Node.js 是一个为 JavaScript 设计的解释器和运行时环境。Node.js 内置了一些增强 JavaScript 编程功能模块。
除了 Node.js，JavaScript 用于网络应用程序的客户端，特别是用于开发动态特性。	Node.js 可以在任何操作系统上用于开发与系统硬件交互的应用程序，特别是对于 web 后端。
JavaScript 可以在不同的浏览器引擎上运行，比如 V8（Google Chrome）、Spider Monkey（Firefox）和JavaScript Core（Safari）。	Node.js 仅在 Chrome 使用的 V8 引擎上运行。

140.【简答题】【Node.js】Node.js 有什么优点，我们为什么要使用 Node.js？

Node.js 有以下优点：

（1）简单， Node. js 用 JavaScript、JSON 进行编码，简单好学。

（2）功能强大，非阻塞式 I/O，在较慢的网络环境中，可以分块传输数据，事件驱动，擅长高并发访问。

（3）轻量级， Node. js 本身既是代码又是服务器，前后端使用同一语言。

（4）可扩展，可以轻松应对多实例、多服务器架构，同时有海量的第三方应用组件。

141.【简答题】【Node.js】请说说 Node.js 的使用场景。

Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。这种模型使得 Node.js 可以避免了由于需要等待输入或者输出（数据库、文件系统、Web 服务器...）响应而造成的 CPU 时间损失。所以，Node.js 适合运用在高并发、I/O 密集、少量业务逻辑的场景，如实时聊天、实时消息推送、客户端逻辑强大的 SPA（单页面应用程序）等场景。

142.【简答题】【Node.js】Node.js 中的异步和同步如何理解？

Node.js 是单线程的，异步是通过一次次的循环事件队列来实现的。同步则是阻塞式的 IO，这在高并发环境中会是一个很大的性能问题，所以同步一般只在基础框架启动时使用，用来加载配置文件、初始化程序等。

143.【简答题】【Node.js】操作错误（operational errors）和程序错误（programmer errors）的区别是什么？

操作错误不是 bug，是系统的问题，例如超时或者硬件故障。 而程序错误（programmer errors）是实际的错误。

144.【简答题】【Node.js】node 中的 Buffer 是干什么的？

Buffer 是用来处理二进制数据的，比如图片，mp3,数据库文件等，Buffer 支持各种编码解码，二进制字符串互转。

145.【简答题】【npm/yarn】npm 如何安装配置全局模块、如何安装配置局部模块，它们有什么区别？

安装全局模块: npm install 模块名称 -g

安装局部模块：npm install 模块名称 -S

全局安装的模块在当前计算机中所有的 node 项目中都可以使用，而局部安装的模块只能在当前项目中使用

146.【简答题】【npm/yarn】npm 是如何管理全局和局部依赖的？有什么其他的替代方案吗？

npm 是一个 nodejs 包管理器。

全局依赖，npm 在安装全局依赖时，将依赖的模块文件下载到计算机 node 应用指定的全局文件夹中，如默认 c:/Users/用户名/.npm/node_modules/目录下，提供给当前计算机中所有项目使用；

局部依赖，npm 在安装局部依赖时，将依赖的模块下载到当前项目中的 node_modules/文件中，提供给当前项目使用

npm 管理依赖的方式已经非常完善了，在新的项目模块管理中，除了使用 npm 以外，还可以使用 安装 cnpm、yarn、yum、pm2 等方式。

147.【简答题】【npm/yarn】什么是 yarn 和 npm？为什么要用 yarn 代替 npm 呢？

npm 是与 Node.js 自带的默认包管理器，它有一个大型的公共库和私有库，存储在 npm registry 的数据库中（官方默认中心库 http://registry.npmjs.org/，国内淘宝镜像 http://registry.npm.taobao.org/），用户可以通过 npm 命令行访问该数据库。在 npm 的帮助下，用户可以轻松管理项目中的依赖项。

yarn 也是一个包管理器，为了解决 npm 的一些缺点。yarn 依赖 npm 注册中心为用户提供对包访问。yarn 底

层结构基于 npm，因此从 npm 迁移到 yarn，项目结构和工作流不需要大改。　　在某些情况下，yarn 提供了比 npm 更好的功能，它会缓存下载的每个包，不必重新下载。通过校验和验证包的完整性来提供更好的安全性，保证在某个系统上运行的包在任何其他系统中的工作方式完全相同，这就是为什么选择 yarn 而不是 npm 来进行包管理。

148.【多选题】【npm/yarn】以下关于 npm 的理解，正确的是（）

A、可以安装和管理项目的依赖

B、可以通过 package.json 文件来描述项目信息和依赖包信息

C、能够指明依赖项的具体版本号

D、package.json 文件使 npm 可以启动我们的项目、运行脚本、安装依赖项等

答案：ABCD

解析：

npm 是一个连接 js 和操作系统的桥梁，也叫做 Node 包管理器（Node Package Manager）。通常下载 Nodejs 时，npm 便会包含在内。下载成功后的各种依赖包会被放在文件夹 node_modules 下，也同时生成并更新一个 package.json 文件（记录了各种依赖包的名称 name、包的版本号 version 等其它信息）。有了 package.json 文件，npm 可以启动你的项目、运行脚本、安装依赖项、发布到 NPM 注册表以及许多其他有用的任务。

149.【单选题】【npm/yarn】如果想要通过 npm 卸载 lodash，应该使用以下哪个命令（）

A、npm i lodash

B、npm uninstall lodash

C、npm delete lodash

D、uninstall lodash

答案：B

解析：卸载命令：npm uninstall <package-name>

150.【单选题】【npm/yarn】npm 中，能够用来执行自定义的 serve 脚本的命令是（）

A、npm run serve

B、run serve

C、serve run

D、serve

答案：A

解析：运行自定义脚本的命令：npm run <task-name>

151.【简答题】【Express】Express 作为其它流行 Node 框架的底层库，主要提供了哪些机制？

Express 框架的机制如下：

为不同 URL 路径中使用不同 HTTP 动词的请求（路由）编写处理程序。

集成了“视图”渲染引擎，以便通过将数据插入模板来生成响应。

设置常见 web 应用设置，比如用于连接的端口，以及渲染响应模板的位置。

在请求处理管道的任何位置添加额外的请求处理“中间件”。

152.【简答题】【Express】请说说 Express 有什么优点？

Express 的优点是线性逻辑：路由和中间件完美融合，通过中间件形式把业务逻辑细分，简化，一个请求进来经过一系列中间件 处理后再响应给用户，再复杂的业务也是线性了，清晰明了。

153.【简答题】【Express】说说 Express 常用函数及其作用？

express.Router 路由组件；

app.get 路由定向；

app.configure 配置；

app.set 设定参数；

app.use 使用中间件

154.【简答题】【Express】在 Express 中如何获取路由的参数？

/users/:nam--使用 req.params.name 来获取；req.body.username 则是获得表单传入参数 username; express 路由支持常用通配符 ?, +, *, and ()

155.【简答题】【Express】请说说 Express 框架有哪些特性？

Express 框架核心特性：

可以设置中间件来响应 HTTP 请求。

定义了路由表用于执行不同的 HTTP 请求动作。

可以通过向模板传递参数来动态渲染 HTML 页面。

156.【简答题】【多线程】什么是线程？什么是进程？线程和进程有什么区别和联系？

线程：线程是进程的子任务，是 CPU 调度和分派的基本单位，用于保证程序的实时性，实现进程内部的并发；线程是操作系统可识别的最小执行和调度单位。

进程：进程是对运行时程序的封装，是系统进行资源调度和分配的的基本单位，实现了操作系统的并发。

区别与联系：一个进程是一个独立(self contained)的运行环境，它可以被看作一个正在运行的程序或应用。而线程是在进程中执行的一个任务。线程是进程的子集，一个进程可以有很多线程，每条线程并行执行不同的任务。不同的进程使用不同的内存空间，而所有的线程共享一片相同的内存空间。 每个线程都拥有单独的栈内存用来存储本地数据。

157.【简答题】【多线程】JavaScript 是单线程还是多线程，如何实现多线程？

JavaScript 语言的一大特点就是单线程，作为浏览器脚本语言，JavaScript 的主要用途是与用户互动，以及操作 DOM，这决定了它只能是单线程，否则会带来很复杂的同步问题。

实现多线程：Web Worker

Web Worker 是 HTML5 提出的新标准，为 JavaScript 创造多线程环境，允许主线程创建 Worker 线程，将一些任务分配给后者运行。在主线程运行的同时，Worker 线程在后台运行，两者互不干扰。等到 Worker 线程完成计算任务，再把结果返回给主线程。这样的好处是，一些计算密集型或高延迟的任务，被 Worker 线程负担了，主线程（通常负责 UI 交互）就会很流畅，不会被阻塞或拖慢。

当我们使用这个类的时候，它就会向浏览器申请一个新的线程。这个线程就用来单独执行一个 js 文件。

var worker = new Worker(js 文件路径);

158.【简答题】【RESTful】请用一句话概括 RESTful。

用 URL 定位资源，用 HTTP 描述操作

159.【单选题】【RESTful】在 RESTful 接口中，某些动作是 HTTP 动词表示不了的，如汇款时从账户 1 向账户 2 汇款 500 元，正确的写法是（）

A、POST /accounts/1/transfer/500/to/2

B、GET /accounts/1/transfer/500/to/2

C、POST /transaction?from=1&to=2&amount=500.00

D、GET /transaction?from=1&to=2&amount=500.00

答案：C。

解析：GET 用来查询操作读取资源的，此处需要更新资源，因此使用 POST。

资源不能是动词，但是可以是一种服务，因此需要使用名词 transaction。

160.【简答题】【RESTful】请谈谈对 RESTful 架构的理解。

RESTful 架构中：

每一个 URI 代表一种资源；

客户端和服务器之间的交互过程就是传递这种资源的某种表现层的过程；

客户端通过 GET（获取数据）、POST（新增数据）、PATCH（更新数据）、PUT（更新数据）、DELETE（删除数据）五个 HTTP 动词请求统一接口，对服务器端资源进行操作，实现"表现层状态转化"。

RESTful 架构的 API 应该像 WEB 页面一样，通过超媒体（类似 WEB 页面的超链接）建立起一个 API 与另一个 API 之间的关系（类似 WEB 页面里通过超链接进行不同页面之间的切换），即每个 API 返回的数据中应该包含当前请求接口地址和相关联的接口地址。

161.【单选题】【layui】以下关于 layui 的理解，错误的是（）

A、layui 是由外国人在 2016 年推出来的一套前端 UI 框架。

B、layui 提供了丰富的内置模块，他们皆可通过模块化的方式按需加载，比如：layer、upload、form 等。

C、layui 是一款采用自身模块规范编写的前端 UI 框架，遵循原生 HTML/CSS/JS 的书写与组织形式。

D、layui 一般可作为 PC 网页端后台系统与前台界面的速成开发方案。

答案：A。

解析：layui 是由我们国人自己在 2016 年推出来的一套前端 UI 框架。

162.【单选题】【layui】关于 layui 中，按钮的使用错误的是（）

A、使用 layui-btn 类名来设置具有默认样式的按钮

B、使用 layui-btn-lg 类名来制作一个大按钮

C、使用 layui-btn-disabled 类名来设置按钮不可点击

D、使用 layui-btn-fluid 类名来设置圆角按钮

答案：D。

解析：layui-btn-fluid 是设置流式按钮（最大化的适应），设置圆角按钮应该使用 layui-btn-radius 类名

163.【单选题】【layui】以下选项中关于 layui 内置模块及其对应功能的描述错误的是（）

A、通过 laydate 模块，我们可以使用 layui 的日期与时间功能
B、通过 laypage 模块，我们可以使用分页功能
C、如果我们要实现文件上传功能，则可以使用 file 模块实现
D、如果我们要展示数据表格，则可以使用 table 模块实现

答案：C。
解析：要实现文件上传，应该使用 upload 模块实现，模块加载名称不是 file

164.【简答题】【Layui】请谈谈 Layui 是什么？

layui（谐音：类 UI) 是一套开源的 Web UI 解决方案，采用自身经典的模块化规范，并遵循原生 HTML/CSS/JS 的开发方式，极易上手，拿来即用，非常适合网页界面的快速开发。layui 区别于那些基于 MVVM 底层的前端框架，更多是面向后端开发者，可作为 Web 页面速成的方案。

165.【单选题】【Layui】以下哪个可以定义 Layui 的导航（）

A、layui-row
B、layui-badge
C、layui-nav
D、layui-progress

答案：C
解析：
layui-row：定义栅格系统中的行
layui-badge: 定义徽章修饰元素
layui-progress：定义进度条

166.【单选题】【Layui】Layui 通过以下哪个方法来加载模块（）

A、use( )
B、on( )
C、define( )
D、config( )

答案：A

解析：layui 通过 use 方法加载模块。

167.【简答题】【微信小程序】说说微信小程序中有哪些类型的文件，以及每种文件的用处是什么？

WXML：是微信自己定义的一套标签语言，结合基础组件、事件系统，可以构建出页面的结构，类似于 HTML 文件

WXSS : 用于描述 WXML 的组件样式 ，类似于 CSS 文件

js : 逻辑处理

json : 小程序页面配置

168.【单选题】【微信小程序】以下关于微信小程序生命周期函数的理解错误的是（）

A、onLoad 页面加载时触发

B、onShow 页面显示/切入前台时触发

C、onReady 页面初次渲染完成时触发。

D、onUnload 页面隐藏或切入后台时触发

答案：D。

解析：onLoad 页面加载时触发。一个页面只会调用一次，可以在 onLoad 的参数中获取打开当前页面路径中的参数

onShow页面显示/切入前台时触发

onReady 页面初次渲染完成时触发。一个页面只会调用一次，代表页面已经准备妥当，可以和视图层进行交互

onHide页面隐藏/切入后台时触发。 如 navigateTo 或底部 tab 切换到其他页面，小程序切入后台等

onUnload 页面卸载时触发。如 redirectTo 或 navigateBack 到其他页面时

169.【单选题】【微信小程序】以下哪个是微信小程序中的尺寸单位（）

A、pt

B、px

C、rpx

D、upx

答案：C。

解析：rpx：小程序的尺寸单位，规定屏幕为 750rpx，可适配不同分辨率的屏幕。

170.【简答题】【微信小程序】请简述微信小程序的原理。

小程序本质就是一个单页面应用，所有的页面渲染和事件处理都在一个页面内进行，但又可以通过微信客户端调用原生的各种接口；

它的架构是数据驱动的架构模式，它的 UI 和数据是分离的，所有的页面更新，都需要通过对数据的更改来实现；

它从技术讲和现有的前端开发差不多，采用 JavaScript、WXML、WXSS 三种技术进行开发；

功能可分为 webview 和 appService 两个部分；webview 用来展现 UI，appService 有来处理业务逻辑、数据及接口调用； 两个部分在两个进程中运行，通过系统层 JSBridge 实现通信，实现 UI 的渲染、事件的处理等。

171.【简答题】【微信小程序】小程序与原生 App 哪个好？

小程序的优点：

基于微信平台开发，享受微信本身自带的流量，这个是最大的优势。无需用户单独安装，只要打开微信就能用，不占用用户手机内存，体验好 开发周期短，一般最多一个月可以上线完成开发所需的资金少，所需资金是开发原生 APP 一半不到。小程序名称是唯一性的，在微信的搜索里权重很高容易上手，只要之前有 HTML+CSS+JS 基础知识，写小程序基本上没有大问题；当然如果了解 ES6+CSS3 则完全可以编写出即精简又动感的小程序； 基本上不需要考虑兼容性问题，只要微信可以正常运行的机器，就可以运行小程序； 发布、审核高效，基本上上午发布审核，下午就审核通过，升级简单，而且支持灰度发布； 开发文档比较完善，开发社区比较活跃。

小程序的缺点：

1、局限性很强，（比如页面大小不能超过 1M。不能打开超过 5 个层级的页面。样式单一。小程序的部分组件已经是成型的了，样式不可以修改。例如：幻灯片、导航。）只能依赖于微信，依托于微信，无法开发后台管理功能。

2、不利于推广，推广面窄，不能分享朋友圈，只能通过分享给朋友，附近小程序推广。其中附近小程序也受到微信的限制

3、后台调试麻烦，因为 API 接口必须 https 请求，且公网地址，也就是说后台代码必须发布到远程服务器上；当然我们可以修改 host 进行 dns 映射把远程服务器转到本地，或者开启 tomcat 远程调试；不管怎么说终归调试比较麻烦。

4、前台测试有诸多坑，最头疼莫过于模拟器与真机显示不一致 。

5、 js 引用只能使用绝对路径，而基于安全性及 MINA 框架实现原理，小程序中对 js 使用做了很多限制，不能使用：new Function，eval，Generator，不能操作 cookie，不能操作 DOM。

原生 App 优点：

1、原生的响应速度快

2、对于有无网络操作时，譬如离线操作基本选用原生开发

3、需要调用系统硬件的功能（摄像头、方向传感器、重力传感器、拨号、GPS、语音、短信、蓝牙等功能）

4、在无网络或者若网的情况下体验好。

原生 App 缺点：

开发周期长，开发成本高 需要下载

172.【单选题】【微信小程序】关于微信小程序中各文件的使用描述错误的是（）

A、JSON 文件用来定义数据结构的

B、WXML 文件用来定义网页结构和模板的，相当于 HTML

C、WXSS 文件用来定义样式

D、JS 文件用来定义逻辑交互的

答案：A

解析：在小程序中，JSON 扮演的静态配置的角色

173.【单选题】【微信小程序】以下哪个选项可以用来配置小程序导航栏标题颜色（）

A、navigationBarTitleText

B、navigationStyle

C、navigationBarTextStyle

D、navigationBarBackgroundColor

答案：C

解析：

navigationBarTitleText：设置导航栏标题文字内容

navigationStyle：配置导航栏样式

navigationBarTextStyle：设置导航栏标题颜色

navigationBarBackgroundColor：设置导航栏背景颜色

174.【多选题】【微信小程序】以下关于小程序宿主环境的理解，正确的是（）

A、微信客户端给小程序所提供的环境即为宿主环境

B、小程序的运行环境分成渲染层和逻辑层

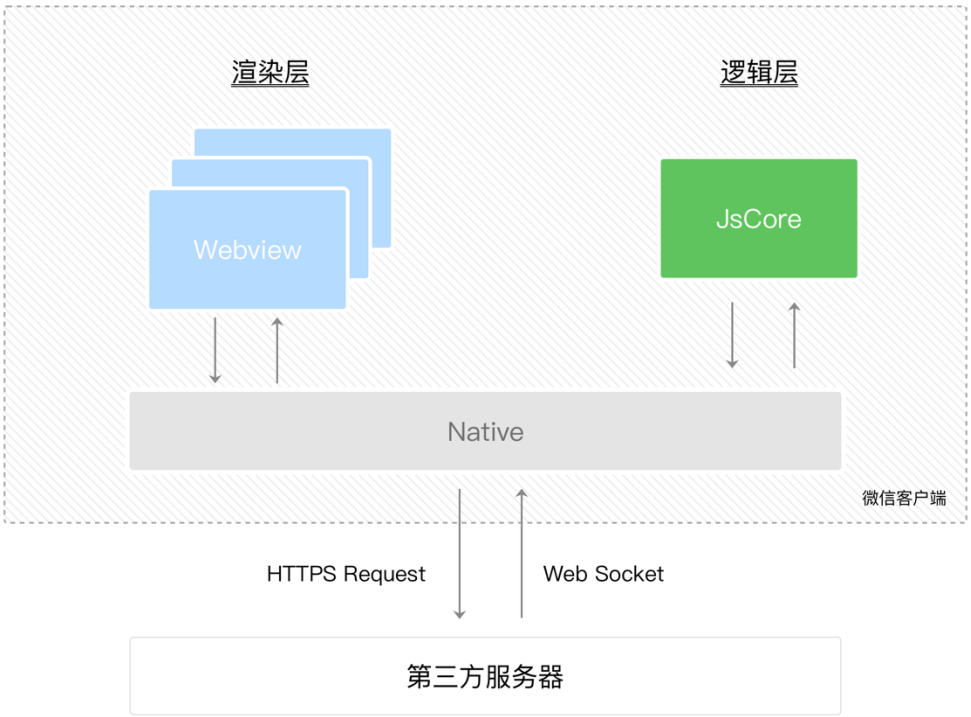
C、WXML 模板和 WXSS 样式工作在渲染层

D、一个小程序存在多个界面，所以渲染层存在多个 WebView 线程

答案：ABCD

解析：我们称微信客户端给小程序所提供的环境为宿主环境。小程序借助宿主环境提供的能力，可以完成许多普通网页无法完成的功能。小程序的运行环境分成渲染层和逻辑层，其中 WXML 模板和 WXSS 样式工作在渲染

层，JS 脚本工作在逻辑层。 小程序的渲染层和逻辑层分别由 2 个线程管理：渲染层的界面使用了 WebView 进行渲染；逻辑层采用 JsCore 线程运行 JS 脚本。一个小程序存在多个界面，所以渲染层存在多个 WebView 线程，这两个线程的通信会经由微信客户端（下文中也会采用 Native 来代指微信客户端）做中转，逻辑层发送网络请求也经由 Native 转发，小程序的通信模型下图所示。



175.【单选题】【微信小程序】以下关于小程序的目录结构说法错误的是（）

A、小程序包含一个描述整体程序的 app 和多个描述各自页面的 page

B、 一个小程序主体部分由 app.js、app.json、app.wxss 三个文件组成，app.wxss 且必须放在项目的根目录

C、一个小程序页面由 js、wxml、json、wxss 四个文件组成

D、描述页面的四个文件，所在路径和文件名可以不相同

答案：ABCD

解析：为了方便开发者减少配置项，描述页面的四个文件必须具有相同的路径与文件名

176.【简答题】【Webpack】前端为什么要进行打包和构建？

代码层：

体积更小（Tree-shaking、压缩、合并），加载更快

编译高级语言和语法（TS、ES6、模块化、scss）

兼容性和错误检查（polyfill,postcss,eslint）

研发层：

统一、高效的开发环境

统一的构建流程和产出标准

集成公司构建规范（提测、上线）

177.【简答题】【Webpack】Loader 机制的作用是什么？

webpack 默认只能打包 js 文件，配置里的 module.rules 数组配置了一组规则，告诉 Webpack 在遇到哪些文件时使用哪些 Loader 去加载和转换打包成 js。

178.【简答题】【Webpack】Plugin（插件）的作用是什么？

Plugin 是用来扩展 Webpack 功能的，通过在构建流程里注入钩子实现，它给 Webpack 带来了很大的灵活性。

Webpack 是通过 plugins 属性来配置需要使用的插件列表的。plugins 属性是一个数组，里面的每一项都是插件的一个实例，在实例化一个组件时可以通过构造函数传入这个组件支持的配置属性。

179.【单选题】【webpack】下列关于常见的 Loader 的描述中有误的是（）

A、source-map-loader：加载额外的 Source Map 文件，以方便断点调试

B、image-loader：加载并且压缩图片文件

C、style-loader：加载 CSS，支持模块化、压缩、文件导入等特性

D、eslint-loader：通过 ESLint 检查 JavaScript 代码

答案：C

解析：css-loader：加载 CSS，支持模块化、压缩、文件导入等特性，style-loader：把 CSS 代码注入到 JavaScript 中，通过 DOM 操作去加载 CSS。

180.【单选题】【webpack】关于下列常见的 Plugin 描述错误的是（）

A、define-plugin：定义环境变量

B、commons-chunk-plugin：提取公共代码

C、uglifyjs-webpack-plugin：通过 UglifyES 压缩 ES6 代码

D、copy-webpack-plugin：自动移除目录插件

答案：D

解析：clean-webpack-plugin 是自动移除目录插件。

181.【多选题】【webpack】利用 webpack 从哪些方面可以优化前端的性能（）

A、压缩代码

B、CDN 加速

C、删除死代码

D、提取公共代码

答案：ABCD

解析：A.uglifyJsPlugin 压缩 js 代码，mini-css-extract-plugin 压缩 css 代码。B.利用 CDN 加速，将引用的静态资源修改为 CDN 上对应的路径，可以利用 webpack 对于 output 参数和 loader 的 publicpath 参数来修改资源路径 C.tree shaking，css 需要使用 Purify-CSS D.webpack4 移除了 CommonsChunkPlugin (提取公共代码)，用 optimization.splitChunks 和 optimization.runtimeChunk 来代替。

182.【单选题】【webpack】下列说法中错误的是（）

A、JavaScript 驱动整个前端的业务，js 文件作为打包的入口

B、webpack 可以用来编译转换代码

C、Loader 可以加载任意类型的资源

D、use 配置多个 loader 的执行顺序是从后向前

答案：B

解析：webpack 只是打包工具，加载器可以用来编译转换代码。

183.【简答题】【webpack】webpack 中热部署用来解决什么问题？

热部署的功能是当一个模块发生变化时，只会重新打包这一个模块，而不是把所有模块重新打包一次，它可以极大的提升项目构建速度

184.【简答题】【Webpack】说几个常见的 loader？

file-loader：把文件输出到一个文件夹中，在代码中通过相对 URL 去引用输出的文件

url-loader：和 file-loader 类似，但是能在文件很小的情况下以 base64 的方式把文件内容注入到代码中去

source-map-loader：加载额外的 Source Map 文件，以方便断点调试

image-loader：加载并且压缩图片文件

babel-loader：把 ES6 转换成 ES5

css-loader：加载 CSS，支持模块化、压缩、文件导入等特性

style-loader：把 CSS 代码注入到 JavaScript 中，通过 DOM 操作去加载 CSS。

eslint-loader：通过 ESLint 检查 JavaScript 代码

185.【简答题】【Webpack】说几个常见的 plugin？

define-plugin：定义环境变量

terser-webpack-plugin：通过 TerserPlugin 压缩 ES6 代码

html-webpack-plugin 为 html 文件中引入的外部资源，可以生成创建 html 入口文件

mini-css-extract-plugin：分离 css 文件

clean-webpack-plugin：删除打包文件

happypack：实现多线程加速编译。

186.【简答题】【Webpack】分别介绍 bundle，chunk，module 是什么？

bundle：是由 webpack 打包出来的文件，

chunk：代码块，一个 chunk 由多个模块组合而成，用于代码的合并和分割。

module：是开发中的单个模块，在 webpack 的世界，一切皆模块，一个模块对应一个文件，webpack 会从配置的 entry 中递归开始找出所有依赖的模块。

187.【简答题】【Webpack】什么是模热更新？有什么优点？

模块热更新是 webpack 的一个功能，它可以使得代码修改之后，不用刷新浏览器就可以更新。

在应用过程中替换添加删出模块，无需重新加载整个页面，是高级版的自动刷新浏览器。

优点：只更新变更内容，以节省宝贵的开发时间。调整样式更加快速，几乎相当于在浏览器中更改样式。

188.【简答题】【Webpack】说说 webpack 的几个核心概念？

Entry：入口，Webpack 执行构建的第一步将从 Entry 开始，可抽象成输入。告诉 webpack 要使用哪个模块作为构建项目的起点，默认为./src/index.js

output ：出口，告诉 webpack 在哪里输出它打包好的代码以及如何命名，默认为./dist

Module：模块，在 Webpack 里一切皆模块，一个模块对应着一个文件。Webpack 会从配置的 Entry 开始递归找出所有依赖的模块。

Chunk：代码块，一个 Chunk 由多个模块组合而成，用于代码合并与分割。

Loader：模块转换器，用于把模块原内容按照需求转换成新内容。

Plugin：扩展插件，在 Webpack 构建流程中的特定时机会广播出对应的事件，插件可以监听这些事件的发生，

在特定时机做对应的事情。

189.【简答题】【Webpack】Webpack 的基本功能有哪些？

代码转换：TypeScript 编译成 JavaScript、SCSS 编译成 CSS 等等

文件优化：压缩 JavaScript、CSS、html 代码，压缩合并图片等

代码分割：提取多个页面的公共代码、提取首屏不需要执行部分的代码让其异步加载

模块合并：在采用模块化的项目有很多模块和文件，需要构建功能把模块分类合并成一个文件

自动刷新：监听本地源代码的变化，自动构建，刷新浏览器

代码校验：在代码被提交到仓库前需要检测代码是否符合规范，以及单元测试是否通过

自动发布：更新完代码后，自动构建出线上发布代码并传输给发布系统。

190.【简答题】【Webpack】前端为什么要进行打包和构建？

代码层面：

体积更小（Tree-shaking、压缩、合并），加载更快

编译高级语言和语法（TS、ES6、模块化、scss）

兼容性和错误检查（polyfill,postcss,eslint）

研发流程层面：

统一、高效的开发环境

统一的构建流程和产出标准

集成公司构建规范（提测、上线）

191.【多选题】【Vue】Vue 实例的 data 属性，可以在哪些生命周期中获取到（）

A、beforeCreate

B、created

C、beforeMount

D、mounted

答案：BCD

解析：beforeCreate 主要是完成 vue 中的一些初始化工作，此时还拿不到 data 中的数据，从 created 开始后面的生命周期函数才可以拿到

192.【单选题】【Vue】下列关于 Vuex 的描述，不正确的是哪项（）

A、Vuex 通过 Vue 实现响应式状态，因此只能用于 Vue

B、Vuex 是一个状态管理模式

C、Vuex 主要用于多视图间状态全局共享与管

D、在 Vuex 中改变状态，可以通过 mutations 和 actions

答案：D

解析：Vuex 改变状态是通过 actions 来改变，mutations 是由 actions 来调用的

193.【单选题】【Vue】下列关于 vue-router 的描述，不正确的是哪项（）

A、vue-router 的常用模式有 hash 和 history 两种

B、可通过 addRoutes 方法动态添加路由

C、可通过 beforeEnter 进行组件内守卫

D、vue-router 借助 Vue 实现响应式的路由，因此只能用于 Vue

答案：C

解析：组件内守卫使用的是 beforeRouteEnter、beforeRouteLeave 和 beforeRouteUpdate

194.【单选题】【Vue】关于 Vue 的生命周期，下列哪项是不正确的（）

A、DOM 渲染在 mounted 中就已经完成了

B、Vue 实例从创建到销毁的过程，就是生命周期

C、created 表示完成数据观测、属性和方法的运算和初始化事件，此时 $el 属性还未显示出

D、页面首次加载过程中，会依次触发 beforeCreate，created，beforeMount，mounted，beforeUpdate，updated

答案：D

解析：首次加载是不会加载 beforeUpdate，updated 的

195.【简答题】【Vue】为什么 Vue 中给 img 标签设置 src 属性会不生效？

因为动态添加 src 被当做静态资源处理了，没有进行编译，要加上 require 进行资源引入

196.【简答题】【Vue】 vue 中怎么重置 data？

Object.assign（）方法用于将所有可枚举属性的值从一个或多个源对象复制到目标对象

this.$data 获取当前状态下的 data

this.$options.data()获取该组件初始状态下的 data。

bject.assign(this.$data, this.$options.data())

197.【简答题】【Vue】你知道 style 加 scoped 属性的用途和原理吗？

scoped 属性会让当前组件设定的样式生效的范围限定在当前组件内，原理就是 vue 会给每个 dom 节点打上一个 data-v-hash 值，然后为你的 css 代码额外添加一个属性选择器

198.【简答题】【Vue】在组件中怎么访问到根实例？

可以通过 this.$root 来获取

199.【简答题】【Vue】SPA 首屏加载速度慢的怎么解决？

1.通过 Gzip 压缩

2.使用路由懒加载

3.利用 webpack 中的 externals 这个属性把打包后不需要打包的库文件都分离出去，减小项目打包后的大小

200.【简答题】【Vue】vuex 是什么？哪种功能场景使用它？

Vuex 是 vue 框架中状态管理，场景：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

201.【简答题】【Vue】怎样理解 Vue 的单向数据流？

数据从父级组件传递给子组件，只能单向绑定。子组件内部不能直接修改从父级传递过来的数据。

所有的 prop 都使得其父子 prop 之间形成了一个单向下行绑定：父级 prop 的更新会向下流动到子组件中，但是反过来则不行。这样会防止从子组件意外改变父级组件的状态，从而导致你的应用的数据流向难以理解。

额外的，每次父级组件发生更新时，子组件中所有的 prop 都将会刷新为最新的值。这意味着你不应该在一个子组件内部改变 prop。如果你这样做了，Vue 会在浏览器的控制台中发出警告。

子组件想修改时，只能通过 $emit 派发一个自定义事件，父组件接收到后，由父组件修改。

202.【简答题】【Vue】$route 和 $router 的区别是什么？

$router 为 VueRouter 的实例，是一个全局路由对象，包含了路由跳转的方法、钩子函数等。

$route 是路由信息对象||跳转的路由对象，每一个路由都会有一个 route 对象，是一个局部对象，包含 path,params,hash,query,fullPath,matched,name 等路由信息参数。

203.【简答题】【Vue】vuex 的 State 特性是？

1、Vuex 就是一个仓库，仓库里面放了很多对象。其中 state 就是数据源存放地，对应于与一般 Vue 对象里面的 data。

2、state 里面存放的数据是响应式的，Vue 组件从 store 中读取数据，若是 store 中的数据发生改变，依赖这个数据的组件也会发生更新。

3、它通过 mapState 把全局的 state 和 getters 映射到当前组件的 computed 计算属性中。

204.【简答题】【Vue】为什么选择 VUE，解决了什么问题？

vue.js 正如官网所说的，是一套构建用户界面的渐进式框架。与其它重量级框架不同的是，vue 被设计为可以自底向上逐层应用。vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另外一方面，当与现代化工具链以及各种支持类库结合使用时，vue 也完全能够为复杂的单页应用提供驱动。

vue.js 有声明式，响应式的数据绑定，组件化开发，并且还使用虚拟 DOM 等技术，统一编程规范和模块等，将项目功能模块化更方便组织和构建复杂应用，便于项目的扩展和维护。 vue 框架维护及时，且 Vue 3 将在 2022 年 2 月 7 日 成为新的默认版本。

205.【简答题】【Vue】如果加入 keep-alive，第一次进入组件会执行哪些生命周期函数？

会执行的钩子函数以及它们的顺序分别为：

beforeCreat、created、beforeMount、mounted、activated

206.【简答题】【Vue】key 的作用和工作原理？

key 的作用主要是为了高效地更新虚拟 DOM，其原理是 vue 中在 patch 过程中，通过 key 可以精准判断两个节点是否是同一个，从而避免频繁更新不同元素，使得整个 patch 过程更加高效，减少 DOM 操作量，提高性能。

另外，若不设置 key 还可能在列表更新时，引发一些隐蔽的 bug 。vue 在使用相同标签名元素的过滤或切换时，也会使用到 key 属性，其目的也是为了让 vue 可以区分它们，否则 vue 只会替换其内部属性而不会触发过滤效果。

207.【简答题】【Vue】v-if 和 v-for 的优先级哪个高？

v-for 的优先级更高。

如果 v-if 和 v-for 同时出现，每次渲染都会先执行循环，再判断条件，无论如何循环都不可避免，浪费了性

能。

情景一：每次遍历时，都需要执行 v-if 解析一次，浪费性能。

```
<ul>
  <li
    v-for="user in users"
    v-if="shouldShowUsers"
    :key="user.id"
  >
    {{ user.name }}
  </li>
</ul>
```

要避免出现这种情况，则在外层嵌套 template ，在这一层进行 v-if 判断，然后在内部进行 v-for 循环。可以改为：

```
<ul>
  <template v-if="shouldShowUsers">
   <li
     v-for="user in users"
    :key="user.id"
  >
    {{ user.name }}
  </li>
 </template>
</ul>
```

情景二：v-if 和 v-for 同时出现在一个标签，过滤一个列表中的项目，比如：

```
<ul>
  <li
    v-for="user in users"
    v-if="user.isActive"
    :key="user.id"
  >
    {{ user.name }}
  </li>
</ul>
```

在这种情况下，请将 users 替换为一个计算属性，让其返回过滤后的列表。

```
<ul>
  <li
    v-for="user in activeUsers"
    :key="user.id"
  >
    {{ user.name }}
  </li>
</ul>
computed: {
  activeUsers: function () {
    return this.users.filter(function (user) {
      return user.isActive
    })
  }
}
```

208.【简答题】【Vue】谈谈对 vue 组件化的理解？

5.1、组件化的定义

组件是独立和可复用的代码组织单元，组件系统是 vue 核心特性之一，它使开发者使用小型、独立和通常可复用的组件构建大型应用。

也可以通俗介绍，把一些用户在程序中一些独立的功能和模块单独提取出来，然后切分为更小的块，这些块有独立的逻辑，有更好的复用性。

组件按照分类有：页面组件(导航)、业务组件(登录)、通用组件(输入框)。

5.2、组件化特点

vue 的组件是基于配置的，通常编写的组件是组件配置而非组件，框架后续会生成其构造函数，它们基于 VueComponent 这个类扩展于 vue 。

常见的组件化技术有：prop 属性、自定义事件、插槽等，这些主要用于组件之间的通信等。

组件之间遵循单向数据流原则。

5.3、组件化的优点

组件化的开发能大幅提高开发效率、测试性和复用性等。

合理的划分组件能够大幅提升应用性能，组件应该是高内聚，低耦合的。

209.【简答题】【Vue】为什么 data 在组件内必须是函数，而 vue 的根实例则没有此限制？

vue 组件可能存在多个实例，如果使用对象形式定义 data ，则会导致它们公用一个 data 对象，那么状态变更将会影响所有组件实例，这是不合理的。

如果采用函数的形式，在实例化组件时，data 会被当做工厂函数返回一个全新的 data 对象，有效规避多实例之间状态污染问题。

所以在组件中的 data 必须是函数，不能使用对象形式。那为什么 vue 根实例没有限制呢？

在 vue 中根实例只能有一个，所以不需要担心多实例的问题，所以根实例中的 data 可以是函数也可以是对象。

210.【简答题】【Vue】你了解哪些 vue 性能优化的方法？

我所了解的 vue 性能优化方法分别有：

1>、路由懒加载

Vue.use(VueRouter)

```
// 传统写法
import Home from '@/views/login/index.vue'
```

//路由懒加载

```
const    Login = ()=> import('@/views/login/index.vue')
const router = new VueRouter({
 routes: [
   { path: '/login', component: Login },
   { path: '/home', component: Home },
   ]
```

export default router

使用路由懒加载，项目打包的时候体积会大幅减小，访问项目时，这些组件也会按需进行加载，大大提升了项目性能。

2>、keep-alive 缓存页面

```
<template>
 <keep-alive>
   <router-view />
 </keep-alive>
```

```
</template>
```

使用 keep-alive 之后会缓存页面，第一次加载之后，关闭再次打开，页面不会重新渲染。keep-alive 的属性：

include：字符串或正则表达式。如果只缓存个别页面，可以使用 include 属性，只缓存匹配组件。

exclude：字符串或正则表达式。如果个别页面不需要缓存时，可以使用 exclude 属性，任何匹配的组件都不会缓存。

3>、v-for 遍历避免同时使用 v-if

```
<ul>
  <li
    v-for="user in activeUsers"
    :key="user.id"
  >
    {{ user.name }}
  </li>
</ul>
computed: {
  activeUsers: function () {
    return this.users.filter(function (user) {
      return user.isActive
    })
  }
}
```

4>、长列表性能优化

如果列表是纯粹的数据展示，不会有任何的改变，就不需要做响应式。

```
export default{
 data(){
  return {
   users:[]
  }
 },
 created(){
  const user = await axios("/api/user")
  this.users = Object.freeze(user)
 }
}
```

Object.freeze() 方法可以冻结一个对象，对象被冻结之后不能被修改，可以让性能大幅度提升。

如果是大数据长列表，可采用虚拟滚动，只渲染少部分区域的内容。可采用三方 vue-virtual-scroll。

5>、事件的销毁

vue 组件销毁时，会自动解绑它的全部指令及事件监听器，但是仅限于组件本身的事件。

```
created(){
  this.timer = setInterval( this.refresh, 2000 )
},
beforeDestory(){
  clearInterval( this.timer )
}
```

6>、图片懒加载

对于图片过多的页面，为了加快页面的加载速度，所以很多时候，需要把未出现在可视区域的图片暂不进行加载，滚动到可视区域之后再开始加载。

可以使用三方的 vue-lazyload 库。

```
<img v-lazy="/src/img/01.jpg" />
```

7>、第三方插件按需引用

使用三方库时，可以按需引入避免体积太大。比如 element-ui ：

```
import { Button  } from "element-ui"
```

8>、无状态的组件标记为函数式组件

```
<template functional>
  <div>组件内容</div>
</template>
```

通过 functional 将组件标记为函数式组件，因为函数式组件没有实例，所以运行时耗费资源较少。

另外还有 v-show 复用 DOM、子组件分割、SSR 等。

211.【简答题】【Vue】computed 与 methods 、watch 的区别？

computed VS methods

```
computed:{
  yyds(){
    log("computed show")
```

```
    return "计算属性"
  }
},
methods:{
  show(){
    log("method show")
    return "计算属性"
  }
}
```

computed 是计算属性，methods 内都是方法，所以调用不同分别为：

yyds

show()

computed 是有缓存的，而 methods 没有缓存，所以 computed 性能比 methods 的好。

computed VS watch

computed 是计算某一个属性的改变，如果某一个值改变了，计算属性会监测到，然后进行返回值。

watch 是监听某一个数据或路由，改变了才会响应，只有改变了才会执行操作。

212.【简答题】【Vue】你怎么理解 vue 中的 diff 算法？

1.diff 算法是虚拟 DOM 技术的必然产物：通过新旧虚拟 DOM 作对比（即 diff），将变化的地方更新在真实 DOM 上；

另外，也需要 diff 高效的执行对比过程，从而降低时间复杂度为 O(n)。(what)

2.vue2.x 中为了降低 Watcher 粒度，每个组件只有一个 Watcher 与之对应，只有引入 diff 才能精确找到发生变化的地方。(why)

3.vue 中 diff 执行的时刻是组件实例执行其更新函数时，它会比对上一次渲染结果 oldVnode 和新的渲染结果 newVnode,此过程称为 patch。(where)

4.diff 过程整体遵循深度优先、同层比较的策略；两个节点之间比较会根据它们是否拥有子节点或者文本节点做不同操作；(How)

比较两组子节点是算法的重点，首先假设头尾节点可能相同做 4 次比对尝试，如果没有找到相同节点才按照通用方式遍历查找，查找结束再按情况处理剩下的节点；

借助 key 通常可以非常精确找到相同节点，因此整个 patch 过程非常高效。

213.【简答题】【Vue】props 和 data 的优先级谁高？

vue 组件内数据相关的属性它们的样式优先级从高到底分别为：

props > methods > data > computed > watch

214.【简答题】【Vue】v-show 和 v-if 指令的共同点和不同点？

共同点：都能控制元素的显示和隐藏；

不同点：实现本质方法不同，v-show 本质就是通过控制 css 中的 display 设置为 none，控制隐藏，只会编译一次；v-if 是动态的向 DOM 树内添加或者删除 DOM 元素，若初始值为 false，就不会编译了。而且 v-if 不停的销毁和创建比较消耗性能。

总结：如果要频繁切换某节点，使用 v-show(切换开销比较小，初始开销较大)。如果不需要频繁切换某节点使用 v-if（初始渲染开销较小，切换开销比较大）。

215.【简答题】【Vue】如何让 CSS 只在当前组件中起作用？

在组件中的 style 前面加上 scoped

216.【简答题】【Vue】如何获取 dom?

ref="domName" 用法：this.$refs.domName

217.【简答题】【Vue】说出几种 vue 当中的指令和它的用法？

v-model 双向数据绑定；

v-for 循环；

v-if v-show 显示与隐藏；

v-on 事件；v-once: 只绑定一次。

218.【简答题】【Vue】为什么使用 key?

需要使用 key 来给每个节点做一个唯一标识，Diff 算法就可以正确的识别此节点。

作用主要是为了高效的更新虚拟 DOM。

219.【简答题】【Vue】请说出 vue.cli 项目中 src 目录每个文件夹和文件的用法？

assets 文件夹是放静态资源；

components 是放组件；

router 是定义路由相关的配置;

app.vue 是一个应用主组件；main.js 是入口文件。

220.【简答题】【Vue】$nextTick 的使用？

当你修改了 data 的值然后马上获取这个 dom 元素的值，是不能获取到更新后的值，

你需要使用$nextTick 这个回调，让修改后的 data 值渲染更新到 dom 元素之后在获取，才能成功。

221.【简答题】【Vue】vue 组件中 data 为什么必须是一个函数？

因为 JavaScript 的特性所导致，在 component 中，data 必须以函数的形式存在，不可以是对象。组建中的 data 写成一个函数，数据以函数返回值的形式定义，这样每次复用组件的时候，都会返回一份新的 data，相当于每个组件实例都有自己私有的数据空间，它们只负责各自维护的数据，不会造成混乱。而单纯的写成对象形式，就是所有的组件实例共用了一个 data，这样改一个全都改了。

222.【判断题】【Vue】在 vue 中，destroyed 表示的是 vue 实例销毁后调用？

答案：正确

解析：vue 的生命周期函数中 destroyed 是实习销毁后调用

223.【单选题】【Vue】以下选项中不属于 vuex 中的属性（）

A、state

B、getters

C、actions

D、init

答案：D

解析：vuex 中有 state、getters、mutations、actions、modules 五个属性

224.【单选题】【Vue】以下代码打印结果为（）

```
<div id="app"> {{ message.split('').reverse().join('') }} </div>
    <script>
        new Vue({
            el: '#app',
            data: {
                message: 'hello'
            }
        })
```

```
    </script>
```

A、hello

B、hel

C、olleh

D、llo

答案：C

解析：split('')分割数组，reverse()反转数组，join()把数组按指定字符转成字符串

225.【简答题】【Vue】Vue 核心思想是什么？

vue 两大核心思想为数据驱动和组件化。

226.【简答题】【Vue】有做过 SPA 类型的项目吗？怎么实现的？

SPA 就是单页面应用程序，主要依靠路由来实现，路由根据不同的值来展示不同的组件。

227.【简答题】【Vue】你对 vue 中 mvvm 的模式设计有什么感想？

mvvm 设计模式采用的双向绑定，当 view 发生变化，model 会跟着变化，model 发生变化，view 也会同步，这样的话，我们就可以更多的关注逻辑，从而减少 dom 的操作，代码的耦合性也更好。

228.【简答题】【Vue】Vue 页面加载的 DOM 渲染在哪个生命周期中就已经完成？

DOM 渲染在 mounted 中就已经完成。

229.【简答题】【Vue】v-on 可以监听多个方法吗？

可以，一个元素绑定多个事件的写法有两种：

1、修饰符的使用

doSomething

2、在 method 方法里分别写两个事件

<button @click="a(), b()">点我 ab<button>

230.【简答题】【Vue】Vue 是自己渐进式 JavaScript 框架，如何理解渐进式？

vue“渐进式”：是指先使用 vue 核心库，在 vue 核心库的基础上，根据自己需要再去逐渐增加功能。

231.【单选题】【Vue】vue 路由元是（）

A、mate

B、meta

C、meat

D、mtea

答案：B

解析：有时，你可能希望将任意信息附加到路由上，如过渡名称、谁可以访问路由等。这些事情可以通过接收属性对象的 meta 属性来实现，并且它可以在路由地址和导航守卫上都被访问到。

232.【单选题】【Vue】vue 的配置文件是（）

A、webpack.js

B、babel.config.js

C、vue.config.js

D、package.json

答案：C

解析：A 是配置 webpack 使用，B 是 babel 的配置文件，D 是包模块的配置文件

233.【单选题】【Vue】vue 中的 v-model 是（）

A、可以实现双向绑定

B、可以实现单向绑定

C、只做渲染，不做数据的获取

D、都对

答案：A

解析：v-model 指令的作用是在表单控件或者组件上创建双向绑定。

234.【多选题】【Vue】Vue 和 jQuery 的区别是（）

A、jQuery 需直接对 DOM 进行节点操作

B、Vue 是数据驱动

C、JQuery 通过数据来驱动视图层显示

D、没有区别

答案：A、B

解析：Jquery 知识 JavaScript 的语法糖，本质上还是操作 DOM 节点，而 Vue 是通过数据来驱动视图层的显示，并不会直接操作 DOM 节点

235.【单选题】【Vue】在 Vue 中，插槽（Slot）主要在 Vue 的（）中使用

A、兄弟组件中

B、父子组件中

C、任意选项中

D、以上说法都错误

答案：B

解析：插槽就是子组件中的提供给父组件使用的一个占位符，用<slot></slot> 表示，父组件可以在这个占位符中填充任何模板代码，如 HTML、组件等，填充的内容会替换子组件的<slot></slot>标签。

236.【多选题】【Vue】vue 中，关于 vue 的表单修饰符说法正确的是（）

A、lazy 可以让 v-model 在 change 事件中同步

B、lazy 可以让 v-model 在 keyup 事件中同步

C、number 自动将用户的输入值转为 Number 类型

D、number 自动将用户的输入值转为 string 类型

答案：A、C

解析：在输入框中，v-model 默认是同步数据，使用 .lazy 会转变为在 change 事件中同步 ，也就是在失去焦点 或者 按下回车键时才更新.

.number 修饰符可以将 输入的值转化为 Number 类型 ，否则虽然你输入的是数字但它的类型其实是 String。

237.【单选题】【Vue】在 vue 中，使用 vue 实例的（）来定义一个局部指令。

A、directive

B、Vue.directive（）

C、directives

D、Vue.directive

答案：C

解析：在 Vue 实例中的 directives 属性中设置私有指令。new Vue({ directives: {} })

238.【单选题】【Vue】以下选项中不可以进行路由跳转的是（）

A、push()

B、replace()

C、route-link

D、jump()

答案：D

解析：Vue 中可以使用 push、replace、route-link 三种方式进行路由跳转

239.【单选题】【Vue】以下获取动态路由{ path: '/user/:id' }中 id 的值正确的是（）

A、this.$route.params.id

B、this.route.params.id

C、this.$router.params.id

D、this.router.params.id

答案：A

解析：$route:路由信息对象，只读对象，通过其可以获取路由信息，进而获取参数

$router:路由操作对象 ，只写对象。

240.【简答题】【vue】Vue.js 3.0 放弃 defineProperty，使用 Proxy 的原因？

Object.defineProperty 缺陷

1.监控到数组下标的变化时，开销很大。所以 Vue.js 放弃了下标变化的检测；

2.Object.defineProperty 只能劫持对象的属性，而 Proxy 是直接代理对象。Object.defineProperty 需要遍历对象的每个属性，如果属性值也是对象，则需要深度遍历。而 Proxy 直接代理对象，不需要遍历操作。

3.Object.defineProperty 对新增属性需要手动进行 Observe。vue2 时需要使用 vm.$set 才能保证新增的属性也是响应式

4.Proxy 支持 13 种拦截操作，这是 defineProperty 所不具有的

5.Proxy 作为新标准，长远来看，JS 引擎会继续优化 Proxy，但 getter 和 setter 基本不会再有针对性优化

241.【简答题】【vue】Vue 2 中给 data 中的对象属性添加一个新的属性时会发生什么？如何解决？

视图并未刷新。这是因为在 Vue 实例创建时，新属性并未声明，因此就没有被 Vue 转换为响应式的属性，自然就不会触发视图的更新，这时就需要使用 Vue 的全局 api $set()：this.$set(this.obj, 'new_property', 'new_value')

242.【简答题】【vue】在 Vue. js 开发环境下调用 API 接口，如何避免跨域？

在 vue.config.js 中配置 proxy 代理

243.【简答题】【vue】vue-router 路由的两种模式，有什么区别？

vue-router 中默认使用的是 hash 模式

hash 模式, 带#。如：http://localhost:8080/#/pageA。改变 hash，浏览器本身不会有任何请求服务器动作的，但是页面状态和 url 已经关联起来了。

history 模式，不带#， 如：http://localhost:8080/ 正常的而路径，并没有#。基于 HTML5 的 pushState、replaceState 实现

244.【简答题】【vue】怎么解决 vue 打包后静态资源图片失效的问题？

设置 assetsPublicPath 将 assetsPublicPath: '/' 改为 assetsPublicPath: './'

最新的 vue-cli 需要在根目录下建一个 vue.config.js 在里面配置 publicPath 即可

245.【判断题】【Vue】vue-cli 是快速创建 Vue 项目的脚手架工具？

答案：正确

解析：

Vue CLI 是一个基于 Vue.js 进行快速开发的完整系统。Vue CLI 致力于将 Vue 生态中的工具基础标准化。它确保了各种构建工具能够基于智能的默认配置即可平稳衔接，这样你可以专注在撰写应用上，而不必花好几天去纠结配置的问题。

246.【简答题】【Sass】为什么要使用 Sass？

使用 Sass 可以更加高效的进行网页样式开发，节省时间。并且与所有版本的 CSS 兼容。可以使用嵌套语法和有用的功能，例如颜色处理，数学和其他值等。

247.【简答题】【sass】说说 SASS 上有哪些类型的运算符？

数字运算符

颜色运算符

字符串运算符

布尔运算符

列表运算符

248.【简答题】【Sass】如何定义 sass 的变量？

sass 的变量以$符号开头，并且变量的分配以分号(;)完成

249.【简答题】【网络基础】谈谈 DNS 的作用？

DNS，即域名系统（英文：Domain Name System，缩写：DNS），作用就是通过域名查询到具体的 IP。因为 IP 存在数字和英文的组合（IP6），很不利于人类记忆，所以就出现了域名。你可以把域名看成是某个 IP 的别名，DNS 就是去查询这个别名的真正名称是什么。

250.【单选题】【网络基础】下面各选项中，非法的 IP 地址是（）

A、196.96.2.8

B、204.112.6.212

C、10.9.14.258

D、203.112.1.36

答案：C

解析：258 不是 ip 地址中的有效值，每个值必须是 0~255 之间的十进制整数。

251.【单选题】【网络基础】在 Windows 系统中的命令行中，以下哪个命令可以用来查看 ip 地址的（）

A、ip

B、ipconfig

C、ifconfig

D、config

答案：B

解析：在 windows 系统中的命令行中，通过 ipconfig 可以查看 ip 地址。

252.【简答题】【网络基础】请谈谈什么是 TCP 三次握手？

在 TCP/IP 协议中，TCP 协议通过三次握手建立一个可靠的连接。

第一次握手：客户端尝试连接服务器，向服务器发送 syn 包（同步序列编号 Synchronize Sequence Numbers），syn=j，客户端进入 SYN_SEND 状态等待服务器确认；

第二次握手：服务器接收客户端 syn 包并确认（ack=j+1），同时向客户端发送一个 SYN 包（syn=k），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态；

第三次握手：第三次握手：客户端收到服务器的 SYN+ACK 包，向服务器发送确认包 ACK(ack=k+1)，此包发送完毕，客户端和服务器进入 ESTABLISHED 状态，完成三次握手

253.【简答题】【模块化】在 JavaScript 中，使用模块化开发有什么好处？

JavaScript 模块化开发的优势如下：

（1）将功能分离出来

（2）具有更好的代码组织方式

（3）可以按需加载

（4）避免了命名冲突

（5）解决了依赖管理问题

254.【单选题】【模块化】以下哪个是模块化规范（）

A、ECMA-262

B、CommonJS

C、TC39

D、RESTful

答案：B。

解析：ECMA-262 是 JavaScript 的官方语言规范，不是模块化规范。

TC39 是一个推动 JavaScript 发展的委员会，也不是模块化规范。

RESTful 是 API 设计规范，它是用于 Web 数据接口的设计。

255.【简答题】【模块化】请谈谈 CommonJS 和 ES6 模块的区别。

两者间最大的两个差异是：

CommonJS 模块输出的是一个值的拷贝，ES6 模块输出的是值的引用；

CommonJS 模块是运行时加载，ES6 模块是编译时输出接口。

CommonJS 加载的是一个对象（即 module.exports 属性），该对象只有在脚本运行完才会生成。而 ES6 模块不是对象，它的对外接口只是一种静态定义，在代码静态解析阶段就会生成。

256.【简答题】【模块化】请简单说说 ES6 模块的使用。

通过 exprot 关键字导出模块，使用 import 关键字引入模块：

```
// utils.js (导出)
const log = s=>{ console.log(s) };
export { log };
```

----------文件分割线----------

```
// index.js
import { log } from './utils.js'
log("you see see you");
```

257.【简答题】【模块化】请说说为什么要进行模块化开发？

原因如下：

高内聚低耦合，有利于团队开发。当项目很复杂时，将项目划分为子模块并分给不同的人开发，最后再组合在一起，这样可以降低模块与模块之间的依赖关系，实现 低耦合，模块中又有特定功能体现高内聚特点。

可重用，方便维护。模块的特点就是有特定功能，当两个项目都需要某种功能时，定义一个特定的模块来实现该功能，这样只需要在两个项目中都引入这个模块就能够实现该功能，不需要书写重复性的代码。 另外，当需要变更该功能时，直接修改该模块，这样就能够修改所有项目的功能，维护起来很方便。

258.【简答题】【模块化】前端模块化是否等同于 JavaScript 模块化？

前端开发相对其他语言来说比较特殊，因为我们实现一个页面功能总是需要 JavaScript、CSS 和 HTML 三者相互交织。如果一个功能只有 JavaScript 实现了模块化， CSS 和 Template 还处于原始状态，那么调用这个功能的时候并不能完全通过模块化的方式，这样的模块化方案并不是完整的。所以我们真正需要的是一种可以将 JavaScript 、CSS 和 HTML 同时都考虑进去的模块化方案，而非只使用 JavaScript 模块化方案。综上所述，前端模块化并不等同于 JavaScript 模块化。

259.【单选题】【UML 建模】E-R 图是什么（）

A、实体映射图

B、实体定义图

C、实体关系图

D、实体转化图

答案：C。

解析：E-R 图，即 Entity Relationship Diagram，实体关系图，用来建立数据模型，帮我们清晰的展现出数据库表与表之间的关系

260.【单选题】【UML 建模】以下哪个不是用例图的组成部分（）

A、用例

B、生命线

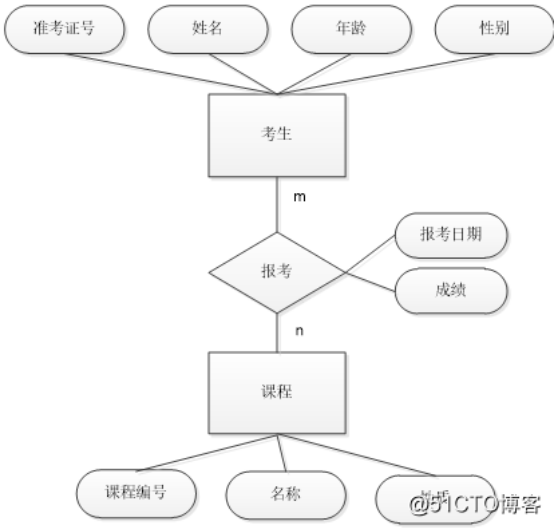
C、参与者

D、系统边界

答案：B。

解析：生命线不属于用例图，生命线是时序图的组成部分

261.【简答题】【UML 建模】现有学生报考系统，实体“考生”有属性：准考证号、姓名、年龄、性别，实体“课程”有属性：课程编号、名称、性质。一名考生可以报考多门课程，考生报考还有报考日期、成绩等信息。 请画出 ER 图，并注明属性和联系类型。



262.【单选题】【UML 建模】下列关于用例图的描述，错误的是（）

A、用例图主要用来描述“用户、需求、系统功能单元”之间的关系

B、参与者是指在系统之外，但与系统直接交互的对象，即 actor，也叫执行者、活动者

C、用例是用户期望系统具备的功能，每一个用例说明一个系统提供给它的使用者的一种服务或功能

D、任何用例都可以在缺少参与者的情况下独立存在

答案：D

解析：任何用例都不能在缺少参与者的情况下独立存在，参与者也必须要有与之关联的用例

263.【单选题】【UML 建模】以下关于类图的理解，错误的是（）

A、类图显示了模型的动态结构

B、类图描述了模型中存在的类、类的内部结构以及它们与其他类的关系

C、类图主要由“类名”、“属性”、“操作”三部分组成

D、类图中的类，与面向对象语言中的类的概念是对应的

答案：A

解析：类图显示了模型的静态结构

264.【简答题】【Axios】为什么使用 axios？

在 vue 中用 axios 进行网络请求。这个是 vue 的作者推荐使用的，而且它本身也有很多特点：例如：支持 promise API、支持拦截请求和响应、支持转换请求和响应数据等

265.【简答题】【Axios】axios 相比原生 ajax 的优点？

ajax 的缺点

本身是针对 MVC 的编程,不符合现在前端 MVVM 的浪潮

基于原生的 XHR 开发，XHR 本身的架构不清晰。

JQuery 整个项目太大，单纯使用 ajax 却要引入整个 JQuery 非常的不合理（采取个性化打包的方案又不能享受 CDN 服务）

不符合关注分离（Separation of Concerns）的原则

配置和调用方式非常混乱，而且基于事件的异步模型不友好