第一阶段面试题

## 一、MySQL数据库

### 1. MySQL中varchar与char的区别以及varchar(50)中的50代表的涵义？

答案：  
Varchar是变长字符串，最多存储的大小是65535字节，查询速度相对较慢；  
Char 是定常字符串，最多存储的大小是255字节，查询速度相对较快；

varchar(50)表示：Mysql 4.0 表示存放50个字节，5.0存放50个字符。

### 2. 简述浏览器打开www.codeboy.com显示页面，中间都经过哪些过程？

答案：

向DNS服务器获取域名对应的IP地址并返回浏览器，通过IP地址向web服务器发请求，web服务器从文件服务器获取网页中所需要的文件，从数据库服务器获取网页中所需要的数据，最后讲完整的网页相应给浏览器。

### 3. 如何解决MySQL中存储中文乱码问题？

答案：

脚本文件采用UTF8编码

客户端连接数据库使用UTF8编码

服务器端创建数据库使用UTF8编码

### 4.Float和Double的区别是什么？

答案：

FLOAT(单精度浮点数)类型 数据可以存储至多8位十进制数，并在内存中占4字节。

DOUBLE(双精度浮点数)类型 数据可以存储至多18位十进制数，并在内存中占8字节。

### 5. 把数据id等于1的名字oldboy更改为oldgirl？

答案：update test set name='oldgirl' where id=1;

### 6.如何登录mysql数据？

答案：mysql -uroot。

## 二、JS基础

### 1. continue和 break有什么区别？

答案：

break和continue都是用来控制循环结构的：

break终止循环，跳出循环体执行循环后面的语句。

continue跳过本次循环，执行下一次循环。

### 2. i++和++i的区别？

答案：

i++ ：先用i值后加1,

++i ：先加1后用i值，

计算：var n=5; 求 var num=n++ + ++n + n++ + ++n +n; //37

### 3. JavaScript都有哪些数据类型？

答案：

原始类型：数值型/字符串型/布尔型/undefined/null

引用类型：对象

### 4. 自调用函数是什么？用于什么地方？

答案：立即执行的函数。 (function(){ //函数体 })();

用于创建一个局部作用域。

### 5. slice和splice有什么区别？

答案：

slice截取子数组 ，从指定的数组中，截取几个连续的元素组成一个新数组

splice允许从指定数组中，删除一部分元素，同时再添加另一部分元素

### 6. typeof返回的类型有哪些？

答案：

number 、string 、boolean 、undefined 、object 、function

ex:

var arr=[1,2,3,4,5];

console.log(typeof(arr)); //object

function myDemo(){

return function test(){};

}

console.log(typeof(myDemo())); //function

### 7. 取 1~11之间的随机数 (即包括1不包括11)？

答案：

Math.floor(Math.random()\*10+1)

parseInt (Math.random()\*10+1)

### 8. 什么是变量声明提前？

答案：使用var关键字声明的变量，会将声明提升到所在作用域的最前边。。

### 9. push、pop、shift和unshift 区别？

答案：

这两组同为对数组的操作，并且会改变数组的本身的长度及内容。

不同的是 push()、pop() 是从数组的尾部进行增减，unshift()、shift() 是从数组的头部进行增减。。

### 10. 例举4种强制类型转换和3种隐式类型转换？

答案：

强制转换：parseInt/parseFloat/Number/toString

隐式转换：

字符串+数值 数值转字符串

数值+布尔型 布尔型中true转为1，false转为0

字符串+布尔型 布尔型转为字符串。

### 11. 函数声明与函数表达式的区别？

答案：

相同点：两者都可以创建函数。

不同点：函数声明可以存在函数提升（前），函数表达式不存在函数提升（前）。

//函数声明

function myFunction(){

function innerFunction() {}

}

//以下为函数表达式

var myFunc = function(){};

myFunc(function(){

return function(){};

} );

### 12. 请指出JavaScript宿主对象和原生对象的区别？

答案：

宿主对象：指JavaScript解释器提供的对象，由解释器厂家自定义并提供实现，不同的解释器提供的扩展对象存在较大的差异（DOM和BOM对象）。

原生对象：JavaScript语言本身预定义的对象，在ECMAScript标准中定义，由所有的解释器厂家来提供具体实现（Array,Date,Math,Number,String,Boolean等）。

### 13. js中有哪些内置函数？

答案：Object, Array, Boolean, Number, String, Function, Date, Math, RegExp, Error, Global

### 14.列举出获取日期相关函数

答案：

console.log(Date.now()); // 返回当前日期和时间'1970/01/01 00:00:00'之间的毫秒值

var dt = new Date(); // 获取当前时间---年 月 日 时 分 秒

console.log(dt.getTime()); //返回当前日期和时间'1970/01/01 00:00:00'之间的毫秒值

console.log(dt.getFullYear()); // 年份

console.log(dt.getMonth()+1); // 月份（0-11）

console.log(dt.getDate()); // 日期（0/1-31）

console.log(dt.getDay()); //星期（0-6）

console.log(dt.getHours()); // 小时（0-23）

console.log(dt.getMinutes()); // 分钟（0-59）

console.log(dt.getMilliseconds); // 返回毫秒（0-999）

console.log(dt.getSeconds()); // 秒（0-59）

### 15.Math相关函数

答案：

Math.random() - 返回 0 ~ 1 之间的随机数

Math.abs(x) - 返回数的绝对值

Math.ceil(x) - 向上取整

Math.floor(x) - 向下取整

Math.round() 四舍五入

Math.max() 和 Max.min() 获取一组数据中的最大值和最小值

Math.PI  获取圆周率π 的值  
Math.pow() 获取一个值的多少次幂   
Math.sqrt() 对数值开方

Math.pow(10，2) = 100;

Math.sqrt(100) = 10;

### 16.null和undefined的区别？

答案：

undefined是访问一个未初始化的变量时返回的值，

null是访问一个尚不存在的对象时所返回的值。因此，可以把undefined看作是空的变量，而null看作是空的对象。

### 17.==和===有什么不同？

答案：

== 抽象相等，比较时，会先进行类型转换，然后再比较值；

===严格相等，判断两个值相等同时数据类型也得相同。

### 18.setTimeout和setInterval的区别是什么？

答案：

二者都是用来设置定时操作的。

setTimeout: 设置一个定时器，在定时器到期后执行一次函数或代码段

setInterval: 设置一个定时器，以固定的时间间隔重复调用一个函数或者代码段

### 19. 请说出以下代码执行结果

for (var i = 0; i < 3; i++) {

setTimeout(function() {

console.log(i);

}, 0);

console.log(i);

}

答案：0 1 2 3 3 3，执行过程如下

var i = 0;

console.log(i); i++;

console.log(i); i++;

console.log(i);i++;

setTimeout(function() {

console.log(i);

}, 0);

setTimeout(function() {

console.log(i);

}, 0);

setTimeout(function() {

console.log(i);

}, 0); //输出 0 1 2 3 3 3

### 20. 请说出（true+flase）>2+true的执行结果

答案：false

### 21. 当前代码块输出结果是什么？

var z=10;

function foo(){console.log(z);}

(function(funArg){var z=20;funArg();})(foo);

答案： 10，自调用函数。

### 22. setTimeout(function(){},10)表示什么意思？

答案：每隔10毫秒调用一次函数。

### 23. 程序中捕获异常的方法？

答案：try{

}catch(e){

}finally{

}

### 24. 以下代码执行结果？

var uname = 'jack'

function change() {

alert(uname) // ?

var uname = 'lily'

alert(uname) //?

}

change()

答案：undefined lily

### 25. 如何使用npm下载express模块？

答案：npm install express

### 26. split和join的区别？

答案：split()将字符串按照指定的字符分割成一个数组，并返回

join()将数组用指定的字符连接成一个字符串，并返回

### 27. 看下列代码会有什么样的输出？

var foo = "11"+2-"1";

console.log(foo);

console.log(typeof foo);

答案：111 number

### 28. foo = foo||bar ，这行代码是什么意思？为什么要这样写？

答案：这种写法称之为短路表达式

相当于：

var foo;

if(foo){

foo=foo;

}else{

foo=bar;

}

常用于函数参数的空判断

### 29. 用js实现随机选取10–100之间的10个数字，存入一个数组，并排序

答案：function getRandom(istart, iend){

var iChoice = iend - istart +1;

return Math.floor(Math.random() \* iChoice+ istart);

}

var iArray = [];

for(var i=0; i<10; i++){

var result= getRandom(10,100);

iArray.push(result);

}

iArray.sort();

### 30. 如何获取javascript三个数中的最大值和最小值？

答案：Math.max(a,b,c);//最大值

Math.min(a,b,c)//最小值

31. 实现冒泡排序？

var array = [5, 4, 3, 2, 1];

var temp = 0;

for (var i = 0; i <array.length; i++){

for (var j = 0; j <array.length - i; j++){

if (array[j] > array[j + 1]){

temp = array[j + 1];

array[j + 1] = array[j];

array[j] = temp;

}

}

## 三、NODEJS

### 1. 同步和异步有何区别？

答案：

同步：指发送一个请求,需要等待返回,然后才能够发送下一个请求，有等待过程（在一个任务进行中时不能开启其他的任务）。

异步：指发送一个请求,不需要等待返回,随时可以再发送下一个请求，即不需要等待（在一个任务进行中时可以开启其他的任务）。

### 2. NodeJS中有哪些类型模块，文件操作用哪一个模块？

答案：

模块类型： 核心模块、自定义模块、第三方模块。

文件操作：fs模块

### 3. 对NodeJS的优点和缺点提出自己的看法？

答案：

CPU密集型任务的特点是进行大量的计算，消耗CPU资源，比如计算圆周率(上千位)、对视频进行编码等, 全靠CPU的运算能力 (一般用C语言，java)

IO（Input / Output）密集型任务，这类任务的特点是CPU消耗很少，大部分时间都在等待IO操作。常见的大部分任务都是IO密集型任务，比如Web应用（一般用脚本语言：python/Nodejs）。

Nodejs设计思想中以事件驱动、异步、非堵塞I/O密集型为核心，他提供的大多数api都是基于事件的、异步的风格。所以非常适合处理高并发请求。此外，与Node服务器交互的客户端代码是由js语言编写的，因此客户端和服务器端都用同一种语言编写，减少了成本。

### 4. 使用NodeJS完成登录功能(编写HTML页面和路由接口中的代码)？

答案：

HTML页面：

<form method="post" action="/user/login">

用户：<input type="text" name="uname"> <br>

密码：<input type="text" name="upwd"> <br>

<input type="submit" >

</form>

user.js路由

const express=require('express');

const pool=require('../pool.js');

var router=express.Router();

router.post('/login',(req,res)=>{

var obj=req.body;

var $uname=obj.uname;

if(!$uname){

res.send({code:401,msg:'uname required'});

return;

}

var $upwd=obj.upwd;

if(!$upwd){

res.send({code:402,msg:'upwd required'});

return;

}

var sql='SELECT \* FROM xz\_user WHERE uname=? AND upwd=?';

pool.query(sql,[$uname,$upwd],(err,result)=>{

if(err) throw err;

if(result.length>0){

res.send({code:200,msg:'login success'});

}else{

res.send({code:301,msg:'login error'}); }

});

});

//路由器导出

module.exports=router;

### 5. 什么是Nodejs？

Nodejs是一个JavaScript的运行环境，是一个服务器端的“JavaScript解释器”，用于方便高效地搭建一些响应速度快、易于扩展的网络应用。它采用事件驱动、异步编程，为网络服务而设计

### 6. nodejs适用于哪些地方？

高并发、聊天、实时消息推送

### 7. npm是什么？

npm是nodejs包管理和分发的工具，用于管理node包。如安装、卸载、发布、查看等.

### 8. npm的好处是什么

通过npm，可以安装和管理项目的依赖，且可以指明依赖项的具体版本号。

### 9. Node.js中导入模块和导入js文件写法上有什么区别？

nodejs引入模块，直接使用名字导入即可

const express = require("express");

导入js文件，需要使用文件的路径，如：

const student = require("./pool.js");

### 10. console有哪些常用方法？

console.log/info/error/warn/time/timeEnd

11. express response有哪些常用方法？

res.download() 弹出文件下载

res.end() 结束response

res.json() 返回json

res.jsonp() 返回jsonp

res.redirect() 重定向请求

res.render() 渲染模板

res.send() 返回多种形式数据

res.sendFile 返回文件

res.sendStatus() 返回状

12. node中的Buffer如何应用?？

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

### 13. 实现一个简单的http服务器？

var http = require('http'); //加载http模块

var server=http.createServer(function(req, res) {

res.writeHead(200, {'Content-Type': 'text/html'});

res.write('hello world');

res.end(); //结束输出流

});

server.listen(3000);

# 第二阶段面试题

## HTML

### 1. img标记属于什么类型元素？

答案：属于行内(inline)元素

### 2.结构标记有哪些？他们与div有什么相同之处，又有什么不同之处？

答案：header，nav，section，aside，footer，article

都是用来做布局的，但结构标记提高了语义性和可读性

### 3. 行内元素，块元素，空元素有哪些？

答案：

（1）行内元素：

a， span，i，img，b ,s,u，sup,sub,label

（2）块元素：

div ，dl，dt，dd，ul，li，ol，p，h1，h2，h3，h4，h5，结构标记

（3）空元素：

br-换行，hr-水平分割线

### 4. 什么是前端？

答案：

用HTML/CSS/JavaScript以及更多的框架技术，编写网站页面、App应用、小程序、2D&3D游戏、Web VR&AR等应用，通过内容设计、交互动画、数据操作构建项目与用户的交互界面。前端工程师在项目中可以与UI设计师和后台程序员合作，实现设计意图并调用后台接口，也可以独立工作完成强用户交互的各类应用。

### 5. HTML、CSS、JavaScript的作用是什么，为什么会产生这三种语言？

答案：这三种语言都是用来编写网页的，HTML用于搭建网页结构展示内容，CSS用于修饰HTML元素，JavaScript用于添加网页的交互验证。

### 6. 说说HTML5，CSS3的新特性，平时使用过哪些？

答案：

H5新特性：

结构标记（header，footer，section，article，aside，nav），新表单元素（url，email，number，date，month，color等），canvas，audio，video，地理定位，拖放，web存储，Web Workers，WebSocket等。

CSS3新特性：

复杂选择器器（属性选择器，伪类选择器，伪元素选择器，兄弟选择器等），边框倒角，边框阴影，渐变，转换，过渡，动画，弹性盒子，媒体查询技术等。

### 7、Doctype 作用? 严格模式与混杂模式如何区分？它们有何意义?

（1）、 声明位于文档中的最前面，处于 标签之前。告知浏览器的解析器，用什么文档类型 规范来解析这个文档。

（2）、严格模式的排版和 JS 运作模式是 以该浏览器支持的最高标准运行。

（3）、在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。

（4）、DOCTYPE 不存在或格式不正确会导致文档以混杂模式呈现。

### 8、对语义化如何理解？

答案：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 CCS 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。

### iframe 有那些缺点

iframe 会阻塞主页面的 Onload 事件；

iframe 和主页面共享连接池，而浏览器对相同域的连接有限制，所以会影响页面的

并行加载。使用 iframe 之前需要考虑这两个缺点。如果需要使用 iframe，最好是

通过 javascript 动态给 iframe 添加 src 属性值，这样可以可以绕开以上两个问题。

### 10.对 WEB 标准以及 W3C 的理解与认识？

答案：

标签闭合、标签小写、不乱嵌套、提高搜索机器人搜索几率、使用外链 css 和 js 脚本、结构与行为表现的分离、文件下载与页面速度更快、内容能被更多的用户所访问、内容能被更广泛的设备所访问、更少的代码和组件，容易维护、改版方便，不需要变动页面内容、提供打印版本而不需要复制内容、提高网站易用性;

### 11.XHTML 和 HTML 有什么区别？

答案：

HTML 是一种基本的 WEB 网页设计语言，XHTML 是一个基于 XML 的置标语言最主要的不同：

XHTML 元素必须被正确地嵌套。

XHTML 元素必须被关闭。标签名必须用小写字母。

XHTML 文档必须拥有根元素。

### 12.img 标签上 title 与 alt 属性的区别是什么?

答案：alt ,当图片不显示时，在图片位置所显示的文字。

title,鼠标移入到元素上时，为该元素添加提示信息。

### 13.你如何对网站的文件和资源进行优化?

答案：解决方案包括：

文件合并

文件最小化/文件压缩

使用 CDN 托管

缓存的使用

## CSS3

### 1.隐藏网页中的元素有几种方式？这些方法有什么区别？

答案：三种方法，

display:none; 脱离文档流，不占页面空间,会改变页面布局。

visivility:hidden ；不会脱离文档流，不会改变页面布局，仍占页面空间

opacity:0，该元素隐藏起来了，但不会改变页面布局，并且，如果该元素已经绑定一些事件，如click事件，那么点击该区域，也能触发点击事件

### 2.如何解决子元素的上外边距溢出？

答案：在父元素的第一个子元素位置处添加一个空的table标记.

### 3.常用来设置透明度的方式有几种？他们之间有何区别？

答案：opacity或rgba

opacity能设置与元素相关各种颜色的透明度

rgba()只能设置当前某一种颜色的透明度

### 4.浮动会对父元素的高度带来什么影响？如何解决这个问题？

答案：子元素一旦浮动，父元素的高度为0，在父元素最后一个子元素位置处添加一个空的块级元素，并设置其clear属性值为both.

.clear:after{

display:block;

content:””;

clear:both;

}

### 5.哪些元素默认有外边距？

答案：body，p，h1-h6，ul，ol，dl，pre

### 6.网页中如何定义锚点？如何连接到锚点位置？

答案：a标记用name定义锚点，其他标记用id属性定义锚点，

连接到锚点的位置：href="#锚点名称"

7.框模型默认的计算方式是什么？要改变默认的计算方式用哪个属性及值？

答案：

实际占地宽度=左右外边距+左右边框+左右内边距+width

实际占地高度=上下外边距+上下边框+上下内边距+height

改变计算方式 box-sizing:border-box

8.实现时针转动效果需要用到CSS中哪些关键技术？

答案：旋转，动画，绝对定位

### 9.设置行内元素居中对齐用什么属性？

答案：text-align：center

### 10.如果要将表单中控件的值提交给服务器必须为控件设置什么属性？

答案：name和value属性

### 11.transition和animation的区别？

答案：

Animation和transition大部分属性是相同的，他们都是随时间改变元素的属性值，他们的主要区别是transition需要触发一个事件才能改变属性，

而animation不需要触发任何事件的情况下也会随时间改变属性值，并且transition为2帧，从from .... to，而animation可以一帧一帧的，也可以理解为一个animation是由多个transition组成。

### 12. 如何实现水平垂直居中？

答案：

1、定位+外边距 盒子宽高已知， position: absolute; left: 50%; top: 50%; margin-left:-自身一半宽度; margin-top: -自身一半高度;

2、定位+transform 盒子高宽已知，

#container{ position:relative; border:1px solid red; width:800px; height:600px; }

#center{ width:100px; height:100px; background:blue; position:absolute; top:50%; left:50%; transform: translate(-50%,-50%); }

3、flex 布局

父级：

display: flex;

align-items: center;

justify-content: center;

### 13. 盒子模型你是怎么理解的？

答案：

元素框处理元素内容，内边距外边距以及边框的计算方式。

盒子模型有两种，W3C和IE盒子模型

（1）w3c的盒模型width=content，不包括margin，padding,border

（2）IE盒模型width=content+padding+border

（3）box-sizng border-box在已设定的宽度和高度之内去设定padding和border

content-box在已设定的高度和宽度之外设置padding和border

14. src和href的区别？

答案：

（1）href：

href是Hypertext Reference的缩写，表示超文本引用。用来建立当前元素和文档之间的链接。

并行下载该文档，不会停止对当前文档的处理

（2）src：

src是source的缩写，src指向的内容会嵌入到文档当前标签所在位置

如img、script、iframe当浏览器解析到该元素时，会暂停浏览器的渲染，直到该资源加载完毕。

15. 说出link和import的区别？

答案：

（1）使用方法不同：

link一般在html头部定义，作为html标签，链接资源，主要用于链接外部的样式表

import一般定义在css内部，并且一定要在其他所有规则之前，也就是一般写在文件头部，并且专门拿来链css的。

（2）加载顺序差别

import比link延迟一些，它会等到页面下载完后才加载，因而可能会产生闪烁

16. 改变页面元素位置的定位方式及他们的区别？

答案：

relative 相对定位 相对于自身原来位置进行偏移，仍处于标准文档流中；

absolute 绝对定位 相对于最近的已定位的祖先元素, 有已定位(指position不是static的元素)祖先元素, 以最近的祖先元素为参考标准。如果无已定位祖先元素, 以body元素为偏移参照， 完全脱离文档流；

fixed 固定定位的元素会相对于视窗来定位,这意味着即便页面滚动，它还是会停留在固定的位置。固定定位也会脱离文档流。

### 17. display有哪些值，说明他们的作用？

答案：

display的取值可以是none/inline/block/inline-block/table

none:元素隐藏不可见，并且元素也不占页面空间。

Inline:元素将呈现行内元素的特点，多个元素在一行中显示，不允许修改尺寸，也不能设置垂直外边距。

block:元素将呈现块级元素的特点，每个元素独占一行，允许修改尺寸。

inline-block:元素将呈现行内块元素的特点，多个元素在一行中显示，但是可以修改尺寸。

table:元素将呈现表格的特点，由内容决定表格的显示方式。元素独占一行，允许修改尺寸。

### 18. 有两个div，外框和内框高度宽度都不定，内框垂直水平居中，利用css如何实现？

答案：.center{

display:flex;

justify-content:center;

align-items:center;

}

### 19. 说说em和rem的区别

答案：rem是基于html元素的字体大小来决定，而em则根据使用它的元素的字体大小决定（很多人错误以为是根据父类元素，实际上是元素继承了父类的属性才会产生的错觉）

20、浏览器的内核分别是什么?

答案：IE 浏览器的内核 Trident、Mozilla 的 Gecko、Chrome 的 Blink（WebKit 的分支）、 Opera 内核原为 Presto，现为 Blink；

### 21、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，但力求影响最小的情况下初始化。最简单的初始化方法就是： \* {padding: 0; margin: 0;} （不建议）淘宝的样式初始化： body, h1, h2, h3, h4, h5, h6, hr, p,, dl, dt, dd, ul, ol, li, pre, form, fieldset, legend, button, input, textarea, th, td { margin:0; padding:0; }

body, button, input, select, textarea { font:12px/1.5tahoma, arial, /5b8b/4f53; }

h1, h2, h3, h4, h5, h6{ font-size:100%; }

ul, ol{ list-style:none;}

a{text-decoration:none; }

a:hover {text-decoration:underline; }

img{ border:0;}

button,input, select,textarea{ font-size:100%; }

table { border-collapse:collapse; border-spacing:0; }

### 22、CSS哪些属性可以继承？优先级算法如何计算？ CSS3 新增伪类有那些？

答案：

可继承的样式： font-size ,font-family ,color, background等;

不可继承的样式：border, padding ,margin ,width ,height ;

优先级就近原则，同权重情况下以最近者为准;

优先级为: !important > id > class > tag

important 比 内联优先级高

CSS3 新增伪类：

:first-child 选择属于其父元素的首个元素。

:last-child 选择属于其父元素的最后一个元素。

:only-child 选择属于其父元素唯一的元素。

:nth-child(n) 选择属于其父元素的任意一个子元素。

:empty 选择没有子元素的元素。

:not(selector) 将满足指定选择器的元素给排除在外。

### 23.描述 css reset 的作用。

答案：reset 重置浏览器为元素设置的默认样式，浏览器的种类不同，样式可能不同，重置是为了让他们统一样式。

### 24.清除浮动带来影响的几种方式，各自的优缺点

答案：

1、直接设置父元素高度，但不是每次都知道父元素的高度。

2、设置父元素也浮动，但不是任何时候父元素都需要浮动，而且浮动会影响后续元素。

3、为父元素设置 overflow属性，但如果有内容需要溢出显示的话，也会一同被隐藏。

4、在父元素中，追加空子元素，并设置其clear:both,但页面中会多出一个空元素。

5、用内容生成的方式:after{content:””;display:block;clear:both;}

### 25.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

## AJAX

### 1.用post方式异步向服务器端提交数据时，需要在发送请求前设置什么？提交的数据放在什么位置？

答案：xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

数据放在send（）括号中

### 2.什么是异步数据加载？

答案：浏览器在向服务器发送请求的时候，不耽误用户在网页上做其它操作，可以同时开启多个任务，并且可以以无刷新的效果来更改页面中的局部内容。

### 3.异步请求数据的步骤分为哪几步？

答案：创建异步对象，绑定监听事件，创建异步请求，发送异步请求

var xhr = new XMLHttpRequest();

xhr.onreadystatechange=function(){

if(xhr.readyState==4&&xhr.status==200){

var resText = xhr.responseText;

}

}

xhr.open(method,url,true);

xhr.send(body);

### 4.异步请求中满足什么条件时才能取出响应的结果数据？

答案：请求状态码为4，响应状态码为200时

### 5.如何理解JSON？

答案：

JSON是JS对象的一种表现方式，即以js对象的数据格式表现出来的字符串，JSON中的两个api如下：

将JSON字符串转换成JSON对象 JSON.parse()

将JSON对象转换成JSON字符串 JSON.stringify()

### 6. http和https的区别？

答案：

http传输的数据都是未加密的，也就是明文的

https协议是由http和ssl协议构建的可进行加密传输和身份认证的网络协议，比http协议的安全性更高。

两者使用不同的链接方式，端口也不同，一般而言，http协议的端口为80，https的端口为443

### 7.列举几种常见的3和4开头的状态码以及他们的意义？

答案：

301 ：永久性重定向

302 ：临时重定向

304 ：请求资源未被修改，命中缓存

400 (错误请求) 服务器不理解请求的语法。

403 (禁止) 服务器拒绝请求。

404 (未找到) 服务器找不到请求的网页。

### 8. 什么是ajax？有哪些优缺点？

答案：AJAX是“Asynchronous JavaScript and XML”的缩写。他是指一种创建交互式网页应用的网页开发技术。

优点：

1.局部刷新页面,减少用户心理和实际的等待时间,带来更好的用户体验。

2.减轻服务器的压力,按需取数据,最大程度的减少冗余数据请求。

3.基于xml标准化,并被广泛支持,不需安装插件。

4.促进页面和数据的分离。

缺点：

1.AJAX破坏了浏览器的Back和History功能

2.AJAX安全问题

3.对搜索引擎支持较弱

4.破坏程序的异常处理机制

5.AJAX不是很好支持移动设备

## Bootstrap

### 1.用bootstrap中的栅格做网页布局的基本结构是什么？默认将一行分为多少列？

答案：容器中设置行，行内设置列

默认将一行分为12个单元格（列）

### 2.如何编写响应式网页？

答案：

(1)声明viewport元标签==也就是设置视口

(2)所有容器/文字/图片 使用相对尺寸

(3)流式布局+弹性布局

(4)使用CSS3 Media Query技术

### 3.同时监听多个scss文件的命令是什么？

答案：node-sass -w  scss文件夹  –o  css文件夹

### 4.在scss中如何定义混合器以及引用？

答案：

@mixin 混合器名称{ 样式声明 }

通过@include来使用混合器。

### 5.对于各类尺寸的设备，Bootstrap设置的class前缀分别是什么？

答案

超小设备手机（<768px）：.col-xs-

小型设备平板电脑（>=768px）：.col-sm-

中型设备台式电脑（>=992px）：.col-md-

大型设备台式电脑（>=1200px）：.col-lg-

### 6.Bootstrap如何设置响应式表格？

增加class="table-responsive"

### 7、使用Bootstrap激活或禁用按钮要如何操作？

激活按钮：给按钮增加.active的class

禁用按钮：给按钮增加disabled="disabled"的属性

### 8、Bootstrap有哪些关于<img>的class？

（1）.img-rounded 为图片添加圆角

（2）.img-circle 将图片变为圆形

（3）.img-thumbnail 缩略图功能

（4）.img-responsive 图片响应式 (将很好地扩展到父元素)

### 9、Bootstrap中有关元素浮动及清除浮动的class？

class="clearfix" 清除浮动

### 10、Bootstrap如何制作下拉菜单？

（1）将下拉菜单包裹在class="dropdown"的<div>中；

（2）在触发下拉菜单的按钮中添加：class="btn dropdown-toggle" id="dropdownMenu1" data-toggle="dropdown"

（3）在包裹下拉菜单的ul中添加：class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu1"

（4）在下拉菜单的列表项中添加：role="presentation"。其中，下拉菜单的标题要添加class="dropdown-header"，选项部分要添加tabindex="-1"。

### 11、Bootstrap如何制作按钮组？

用class="btn-group"的<div>去包裹按钮组；class="btn-group-vertical"可设置垂直按钮组。

### 12、Bootstrap中的导航都有哪些？

（1）导航元素：有class="nav nav-tabs"的标签页导航，还有class="nav nav-pills"的胶囊式标签页导航；

（2）导航栏：class="navbar navbar-default" role="navigation"；

（3）面包屑导航：class="breadcrumb"

### 13、Bootstrap中设置分页的class？

默认的分页：class="pagination"

### 14、Bootstrap中如何制作徽章？

<span class="badge">26</span>

### 15、Bootstrap中超大屏幕的作用是什么？

设置class="jumbotron"可以制作超大屏幕，该组件可以增加标题的大小并增加更多的外边距。

16、Bootstrap 网格系统列与列之间的间隙宽度是多少？

间隙宽度为30px（一个列的每边分别是15px）。

### 17、用Bootstrap，如何设置文字的对齐方式？

class="text-center" 设置居中文本

class="text-right" 设置向右对齐文本

class="text-left" 设置向左对齐文本

# 第三阶段面试题

## 一、JavaScript高级

### 1. 判断以下程序的输出结果:

var age=100;

function test(){

this.age=50;

return function(){

return this.age;

}

}

var m=new test();

alert(m());

var n=test();

alert(n());

答案：

10050

构造函数一旦返回一个对象，就不再创建新对象

m获得的是function(){ return this.age; }

n=test()，this指向window。先将全局变量age变为50,又返回一个函数function(){ return this.age; }保存在变量n中

调用n时，this指向window。

### 2. 判断以下程序的输出结果:

var name="The Window";

var obj={

name:"My obj",

getName:function(){

return function(){

return this.name;

}

}

};

console.log(obj.getName()());

答案：

the window

obj.getName() 返回一个函数对象function(){ return this.name; }

(function(){ return this.name; }()) 相当于匿名函数自调，this指向window

### 3. 判断以下程序的输出结果:

var length=10;

function fn(){

console.log(this.length);

}

var obj={

length:5,

method:function(fn){

fn();

arguments[0]();

}

};

obj.method(fn,1)

答案：

10 2

fn() this指向window，所以输出10

arguments[0]() 属于特殊情况，this->arguments,相当于arguments.0(), 所以，this指向arguments。所以length输出的是obj.method()的参数个数，为2.

### 4. 统计一个字符串中出现次数最多的字符是? 共出现多少次

答案：

var dict={};

var c="", max=1;

for(var i=0;i<str.length;i++){

var char=str[i];

if(dict[char]===undefined)

dict[char]=1;

else{

dict[char]+=1;

if(dict[char]>max){

max=dict[char];

c=char;

}

}

}

console.log(c,max);

提前创建一个空对象，用于保存每个字母出现的次数。

提前创建变量，准备保存出现次数最多的字符和出现的次数。

然后，遍历字符串中每个字母，每遍历一个字母就判断结果对象中是否包含以当前字母为属性名的属性。如果不包含以当前字母为属性名的属性，说明是首次遇见该字母，就向结果对象中强行添加以该字母为属性名的属性，值暂时为1。如果结果对象中已经包含以当前字母为属性名的属性，说明不是第一次碰见该字母。则取出该字母名属性对应的次数+1。只要当前字母出现的次数>之前变量中记录的最大次数，就用当前字母和出现次数，取而代之。

### 5. 判断以下程序的输出结果:

for(var i=0;i<5;i++){

setTimeout(function(){

console.log(i);

},0)

}

console.log(i);

答案：

5 5 5 5 5

函数定义时，函数内容是不执行的，所以i还是i，不会变成0，1，2，3，4

定时器中的回调函数只能再主程序执行完才能开始执行

当主程序执行完，循环变量i，已经被改为5了。

### 6. 判断以下程序的输出结果:

window.color="red";

let color="green";

let obj={

color:"blue"

};

let sayColor=()=>{

return this.color;

}

console.log(sayColor.apply(obj));

let a=10;

console.log(window.a);

答案：

red undefined

let相当于匿名函数自调，所以，let声明的变量，不会自动加入到window

剪头函数内外this通用，所以apply也无法替换sayColor函数内的this，所以this指向window，所以输出red

### 7. 判断以下程序的输出结果:

var c=1;

function c(c){

console.log(c);

var c=3;

}

c(2);

答案：

报错: TypeError: c不是一个函数

function c(c){} 整体被声明提前，后又被c=1代替。所以，c最后不是一个函数，而是数字1

### 8. 判断以下程序的输出结果:

function change(){

alert(typeof fn)

function fn(){ alert('hello') }

var fn;

}

change();

答案：

function

function fn(){…}被整体声明提前了

var fn发现已经有fn变量了，就不再重复创建，所以，var fn没作用。

### 9. 判断以下程序的输出结果:

a=3

a.prop=4;

alert(a+a.prop)

答案：

NaN

a.prop=4，等效于new Number(a).prop=4, 但是new Number(a)，使用后自动释放，4也不存在了

再次使用a.prop，又等效于新的new Number(a)，所以没有prop属性，值为undefined。

数字+undefined, undefined隐式转为数字NaN，导致计算结果为NaN

### 10. 判断以下程序的输出结果:

var o={

a:10,

b:{

a:12,

fn:function(){

var a=13;

console.log(this.a);

}

}

}

o.b.fn();

答案：

12

this指.前的o.b对象，所以a为12

### 11. 判断以下程序的输出结果:

var obj1 = {

name: 'obj1',

fn: function() {

document.write(this.name);

}

};

var obj2 = {name: 'obj2'};

var obj3 = {name: 'obj3'};

obj1.fn();

var newFn = obj1.fn;

newFn();

newFn.call(obj2);

obj3.fn = newFn;

obj3.fn();

答案：

obj1 空字符串 obj2 obj3

this指.前的obj1

因为newFn调用时，前边没有.，所以this->window，

call是强行替换newFn中的this为obj2

this指.前的obj3

### 12. 一个数组 par 中存放有多个人员的信息，每个人员的信息由年龄 age 和姓名 name 组成，如{age: 2, name: 'xx'}。请写一段 JS 程序，对这个数组按年龄从小到大进行排序。

答案:

function parSort(arr, propName) {

arr.sort(function(a, b) {

return a[propName]-b[propName];

});

}

parSort(arr, “age”)

1)数组的sort函数的参数，是一个比较器函数。比较器函数的形参a和b，指当前要排序的数组中的任意两个作比较的元素。如果作比较的a和b两个元素是简单的数字类型，则a直接和b相减，就可比较两数大小。但是，如果a和b都是对象类型的元素。要比较两个对象中某个属性的值得大小，就必须用a.属性-b.属性。如果属性名是灵活的，来自于变量，则必须用a[属性名变量]-b[属性名变量]，就可比出两个对象的某个属性值得大小。

2)最后，数组是引用类型的对象，在函数内修改数组，等效于修改原数组。所以不用返回值。

### 13. 有字符串 var = 'abc345efgabcab'，请写出 3 条 JS 语句分别实现如下 3 个功能：

1）去掉字符串中的a、b、c 字符，形成结果：'345efg'

2）将字符串中的数字用中括号括起来，形成结果：'abc[345]efgabcab'

3）将字符串中的每个数字的值分别乘以 2，形成结果：'abc6810efgabcab'

答案:

1)str.replace(/([a-c])/g, '');

2)str.replace(/(\d+)/g, '[$1]');

3)str.replace(/(\d)/g, function(num) {return num\*2;});

1)将字符串中的a，b，c三个字符都替换为空字符串

2)找到字符串中多个连续的数字，分为一组，然后将这一组的内容，替换为用[]包裹。$1，可获得关键词中第一个()包裹的内容。

3)找到字符串中每个数字，\*2后，再放回原位置。

### 14. 判断以下程序的输出结果:

var a=10;

var obj={

a:20,

intr:function(){

var a=30;

console.log(this.a);

}

}

obj.intr();

var intr=obj.intr;

intr();

答案:

20 10

obj.intr()，this指.前的obj，所以输出20

intr(), this指window，所以输出10

### 15. 判断以下程序的输出结果:

function fun(){

for(var i=0,arr=[];i<3;i++){

arr[i]=function(){

console.log(i);

}

}

return arr;

}

var funs=fun();

funs[0]();

funs[1]();

funs[2]();

答案:

3 3 3

### 16. 定义函数实现深克隆一个对象:

答案:

var lilei={

sname:"Li Lei",

score:null,

friends:["jack","rose"],

address:{

prov:"北京",

city:"北京",

area:"海淀",

street:"万寿路"

},

sage:11

}

function clone(obj){

if(obj===null){

return null;

}else if({}.toString.call(obj)==="[object Array]"){

var newArr=[];

newArr=obj.slice();

return newArr;

}

var newObj={};

//遍历原obj的每个属性

for(var key in obj){

//如果原对象中当前属性值是原始类型

if(typeof obj[key]!=="object"){

//在新对象中添加和原对象中同名的属性

newObj[key]=obj[key];

//原始类型复制，就是复制副本

}else{//否则,当前属性不是原始类型的值，再次调用clone函数，继续复制当前属性值

newObj[key]=clone(obj[key])

}

}

return newObj;

}

console.log(lilei);

var lilei2=clone(lilei);

console.log(lilei2);

console.log(lilei==lilei2);//true

lilei2.address.area="朝阳";

console.log(lilei.address);

### 17. 介绍 JavaScript 的原型，原型链？有什么特点？

答案:

原型：

- JavaScript 的所有对象中都包含了一个 [proto] 内部属性，这个属性所对应的就是该对象的原型

- JavaScript 的函数对象，除了原型 [proto] 之外，还预置了 prototype 属性

- 当函数对象作为构造函数创建实例时，该 prototype 属性值将被作为实例对象的原型 [proto]。

原型链：

- 当一个对象调用的属性/方法自身不存在时，就会去自己 [proto] 关联的前辈 prototype 对象上去找

- 如果没找到，就会去该 prototype 原型 [proto] 关联的前辈 prototype 去找。依次类推，直到找到属性/方法或 undefined 为止。从而形成了所谓的“原型链”

原型特点：

- JavaScript 对象是通过引用来传递的，当修改原型时，与之相关的对象也会继承这一改变

### 18. 谈谈Javascript 垃圾回收方法

答案:

标记清除（mark and sweep）

- 这是 JavaScript 最常见的垃圾回收方式，当变量进入执行环境的时候，比如函数中声明一个变量，垃圾回收器将其标记为“进入环境”，当变量离开环境的时候（函数执行结束）将其标记为“离开环境”

- 垃圾回收器会在运行的时候给存储在内存中的所有变量加上标记，然后去掉环境中的变量以及被环境中变量所引用的变量（闭包），在这些完成之后仍存在标记的就是要删除的变量了

引用计数(reference counting)

- 在低版本 IE 中经常会出现内存泄露，很多时候就是因为其采用引用计数方式进行垃圾回收。引用计数的策略是跟踪记录每个值被使用的次数，当声明了一个 变量并将一个引用类型赋值给该变量的时候这个值的引用次数就加 1，如果该变量的值变成了另外一个，则这个值得引用次数减 1，当这个值的引用次数变为 0 的时 候，说明没有变量在使用，这个值没法被访问了，因此可以将其占用的空间回收，这样垃圾回收器会在运行的时候清理掉引用次数为 0 的值占用的空间

### 19. 说说严格模式的限制

答案:

严格模式主要有以下限制：

- 变量必须声明后再使用

- 函数的参数不能有同名属性，否则报错

- 不能使用 with 语句

- 不能对只读属性赋值，否则报错

- 不能使用前缀 0 表示八进制数，否则报错

- 不能删除不可删除的属性，否则报错

- 不能删除变量 delete prop，会报错，只能删除属性 delete global[prop]

- eval 不会在它的外层作用域引入变量

- eval 和 arguments 不能被重新赋值

- arguments 不会自动反映函数参数的变化

- 不能使用 arguments.callee

- 不能使用 arguments.caller

- 禁止 this 指向全局对象

- 不能使用 fn.caller 和 fn.arguments 获取函数调用的堆栈

- 增加了保留字（比如 protected、static 和 interface）

### 20. 使用正则表达式验证邮箱格式

答案:

var reg = /^(\w)+(\.\w+)\*@(\w)+((\.\w{2,3}){1,3})$/;

var email = "example@qq.com";

console.log(reg.test(email)); // true

### 21. 使用typeof bar ===“object”来确定bar是否是一个对象时有什么潜在的缺陷？这个陷阱如何避免？

答案:

尽管typeof bar ===“object”是检查bar是否是对象的可靠方法，但JavaScript中令人惊讶的问题null也被认为是一个对象！

因此，对于大多数开发人员来说，下面的代码会将true（而不是false）打印到控制台：

var bar = null;

console.log(typeof bar === "object"); // logs true!

只要知道这一点，就可以通过检查bar是否为空来轻松避免该问题：

console.log((bar !== null) && (typeof bar === "object")); // logs false

为了让我们的答案更加的完整，还有两件事值得注意： 首先，如果bar是一个函数，上面的解决方案将返回false。在大多数情况下，这是所期望的行为，但是在您希望函数返回true的情况下，您可以将上述解决方案修改为：

console.log((bar !== null) && ((typeof bar === "object") || (typeof bar === "function")));

其次，如果bar是数组，则上述解决方案将返回true（例如，如果var bar = [];）。在大多数情况下，这是所希望的行为，因为数组确实是对象，但是在您想要对数组也是false的情况下，可以将上述解决方案修改为：

console.log((bar !== null) && (typeof bar === "object") && (toString.call(bar) !== "[object Array]"));

但是，还有一个替代方法对空值，数组和函数返回false，但对于对象则为true：

console.log((bar !== null) && (bar.constructor === Object));

或者，如果您使用jQuery：

console.log((bar !== null) && (typeof bar === "object") && (! $.isArray(bar)));

ES5使得数组的情况非常简单，包括它自己的空检查：

console.log(Array.isArray(bar));

### 22. 以下代码的输出是什么？解释你的答案

var a={},

b={key:'b'},

c={key:'c'};

a[b]=123;

a[c]=456;

console.log(a[b]);

答案:

456

原因如下：设置对象属性时，JavaScript会隐式地将参数值串联起来。在这种情况下，由于b和c都是对象，它们都将被转换为“[object Object]”。因此，a [b]和a [c]都等价于[“[object Object]”]，并且可以互换使用。因此，设置或引用[c]与设置或引用[b]完全相同。

## 二、DOM

### 1. 利用冒泡和不利用冒泡的差别

答案：

1.绑定位置不同: 不利用冒泡绑定在目标元素上，利用冒泡绑定在父元素上

2.监听对象的个数不同: 不利用冒泡会反复创建多个监听，利用冒泡始终只有一个监听

3.动态生成的元素: 不利用冒泡无法自动获得事件处理函数，必须反复绑定

利用冒泡可让动态添加的子元素自动获得父元素的处理函数，无需反复绑定

### 2. 按HTML查找和按选择器查找的差别

答案：

1.返回值不同: 按HTML查找返回动态集合，按选择器查找返回非动态集合

2.效率不同: 按HTML查找效率高，按选择器查找效率低

3.易用性不同: 当条件复杂时，按HTML查找繁琐，而按选择器查找简单

### 3. 列举DOM中常用优化

答案：

1.查找时，如果之用一个条件就可查询出结果时，优先选择按HTML查找。如果查找条件复杂，则优先选择易用的按选择器查找

2.添加时，尽量减少操作DOM树的次数，减少重排重绘。如果同时添加父元素和子元素，应先将子元素添加到到父元素，最后再将父元素添加到DOM树。如果添加多个平级子元素，则应先将子元素添加到文档片段，最后，再将文档片段添加到DOM树

3.修改时，尽量减少重排重绘。如果同时修改多个元素的内容或样式，应尽量使用innerHTML和cssText方式修改元素的内容和样式。应使用class批量修改样式

4.事件绑定时，应尽量利用冒泡减少事件监听的个数。

### 4. 如何鉴别浏览器的名称和版本号

答案：

var browser,version,ua=navigator.userAgent;

if(ua.indexOf("IE")!=-1) browser="IE"

else if(ua.indexOf("Edge")!=-1) browser="Edge"

else if(ua.indexOf("Firefox")!=-1) browser="Firefox"

else if(ua.indexOf("OPR")!=-1) browser="OPR"

else if(ua.indexOf("Chrome")!=-1) browser="Chrome"

else if(ua.indexOf("Safari")!=-1) browser="Safari"

else if(ua.indexOf("Trident")!=-1) browser="IE",version=11;

document.write(`<h1>${browser}</h1>`);

if(version===undefined){

//截取: 从浏览器名称所在位置，再跳过浏览器名称长度+1 之后的3位

var i=ua.indexOf(browser);

i+=browser.length+1;

version=parseFloat(ua.slice(i,i+3))

}

document.write(`<h1>${version}</h1>`);

### 5. 为按钮绑定事件，实现事件节流:

答案:

//节流

var canClick=true;

btn.onclick=function(){

if(canClick){

canClick=false;

console.log("发送ajax请求，加载更多...");

setTimeout(function(){

console.log("请求完成!");

canClick=true;

},3000);

}

}

### 6. 为页面绑定滚动事件，实现事件防抖:

### 答案:

//防抖

var timer1;

window.onscroll=function(){

if(timer1!==undefined){

clearTimeout(timer1);

}

timer1=setTimeout(function(){

console.log("发送ajax请求，加载更多");

timer1=undefined;

},200)

}

### 7. DOM 元素 e 的 e.getAttribute(propName)和 e.propName 有什么区别和联系

答案:

- e.getAttribute()，是标准 DOM 操作文档元素属性的方法，具有通用性可在任意文档上使用，返回元素在源文件中设置的属性

- e.propName 通常是在 HTML 文档中访问特定元素的特性，浏览器解析元素后生成对应对象（如 a 标签生成 HTMLAnchorElement），这些对象的特性会根据特定规则结合属性设置得到，对于没有对应特性的属性，只能使用 getAttribute 进行访问

- e.getAttribute()返回值是源文件中设置的值，类型是字符串或者 null（有的实现返回""）

- e.propName 返回值可能是字符串、布尔值、对象、undefined 等

- 大部分 attribute 与 property 是一一对应关系，修改其中一个会影响另一个，如 id，title 等属性

- 一些布尔属性`<input hidden/>`的检测设置需要 hasAttribute 和 removeAttribute 来完成，或者设置对应 property

- 像`<a href="../index.html">link</a>`中 href 属性，转换成 property 的时候需要通过转换得到完整 URL

- 一些 attribute 和 property 不是一一对应如：form 控件中`<input value="hello"/>`对应的是 defaultValue，修改或设置 value property 修改的是控件当前值，setAttribute 修改 value 属性不会改变 value property

### 8. 如何最小化重绘(repaint)和回流(reflow)？

答案:

- 需要要对元素进行复杂的操作时，可以先隐藏(display:"none")，操作完成后再显示

- 需要创建多个 DOM 节点时，使用 DocumentFragment 创建完后一次性的加入 document

- 缓存 Layout 属性值，如：var left = elem.offsetLeft; 这样，多次使用 left 只产生一次回流

- 尽量避免用 table 布局（table 元素一旦触发回流就会导致 table 里所有的其它元素回流）

- 避免使用 css 表达式(expression)，因为每次调用都会重新计算值（包括加载页面）

- 尽量使用 css 属性简写，如：用 border 代替 border-width, border-style, border-color

批量修改元素样式：elem.className 和 elem.style.cssText 代替 elem.style.xxx

9. 描述一下DOM事件模型：

答案:

捕获、目标触发、冒泡

捕获阶段: 由外向内，依次记录各级父元素上的事件处理函数。只记录，暂不触发。

目标触发：优先触发目标元素上的事件处理函数。

冒泡: 由内向外，依次触发各级父元素上的事件处理函数。

### 10. 谈谈事件委托/代理:

答案:

事件委托是指将事件绑定目标元素的到父元素上，利用冒泡机制触发该事件

优点：

- 可以减少事件注册，节省大量内存占用

- 可以将事件应用于动态添加的子元素上

缺点： 使用不当会造成事件在不应该触发时触发

示例：

js

ulEl.addEventListener('click', function(e){

var target = event.target || event.srcElement;

if(!!target && target.nodeName.toUpperCase() === "LI"){

console.log(target.innerHTML);

}

}, false);

### 11. IE 的事件处理和 W3C 的事件处理有哪些区别？

答案:

绑定事件

- W3C: targetEl.addEventListener('click', handler, false);

- IE: targetEl.attachEvent('onclick', handler);

删除事件

- W3C: targetEl.removeEventListener('click', handler, false);

- IE: targetEl.detachEvent(event, handler);

事件对象

- W3C: var e = arguments.callee.caller.arguments[0]

- IE: window.event

事件目标

- W3C: e.target

- IE: window.event.srcElement

阻止事件默认行为

- W3C: e.preventDefault()

- IE: window.event.returnValue = false'

阻止事件传播

- W3C: e.stopPropagation()

- IE: window.event.cancelBubble = true

### 12. 解释一下这段代码的意思吗？

```js

[].forEach.call($$("\*"), function(el){

el.style.outline = "1px solid #" + (~~(Math.random()\*(1<<24))).toString(16);

})

```

答案:

获取页面所有的元素，遍历这些元素，为它们添加 1 像素随机颜色的轮廓(outline)

- $$(sel) // $$函数被许多现代浏览器命令行支持，等价于 document.querySelectorAll(sel)

- [].forEach.call(NodeLists) // 使用 call 函数将数组遍历函数 forEach 应到节点元素列表

- el.style.outline = "1px solid #333" // 样式 outline 位于盒模型之外，不影响元素布局位置

- (1<<24) // parseInt("ffffff", 16) == 16777215 == 2^24 - 1 // 1<<24 == 2^24 == 16777216

- Math.random()\\*(1<<24) // 表示一个位于 0 到 16777216 之间的随机浮点数

- ~~Math.random()\\*(1<<24) // ~~ 作用相当于 parseInt 取整

- (~~(Math.random()\\*(1<<24))).toString(16) // 转换为一个十六进制-

### 13. 前端优化的方法：

答案:

1.减少dom操作

    2.部署前，图片压缩，代码压缩

    3.优化js代码结构，减少冗余代码

    4.减少http请求，合理设置 HTTP缓存

    5.使用内容分发cdn加速

    6.静态资源缓存

    7.图片延迟加载

## 三、jQuery

### 1. $的原理

答案：

1)$(“选择器”) 是先查找DOM元素，再将DOM元素放入jQuery对象中

其中自带优化:

如果选择器是#id，则自动调用getElementById

如果选择器是.class，则自动调用getElementsByClassName

如果选择器是标签名，则自动调用getElementsByTagName

否则，其它选择器，都自动调用querySelectorAll()

2)$(DOM元素) 是直接将DOM元素放入jQuery对象中

3)$(“HTML片段”) 是创建一个新元素

4)$(function(){}) 是绑定事件，在DOM内容加载后就提前触发。

### 2. 实现动画有几种方式，哪种好？

答案：

1)CSS: transition, animateion

优点: 由专门的排版引擎解析，效率高

缺点: 无法随意控制交互行为

2)JS: 定时器， $().animate()

优点: 可随意控制交互行为

缺点: 效率不如css动画

3)requestAnimationFrame()

优点: 可根据浏览器的刷新频率自动优化动画效果

缺点: 新API，有兼容性问题

### 3. 实现跨域访问有几种方式

答案：

主要有两种

1)JSONP:

客户端: 客户端动态添加script元素，用script发送请求，代替ajax请求，并携带客户端一个函数名到服务端

服务端: 接收客户端发来的函数名，将函数名和要返回的数据拼接为一条可执行的js语句，返回

2)CORS: Cross-Origin Resources Sharing

客户端正常发送ajax请求，服务端定义响应头，允许指定来源的请求跨域访问:

res.writeHead(200,{

…，

“Access-Control-Allow-Origi

n”:”允许的请求来源域名”

})

### 4. 请解释一下 JavaScript 的同源策略, 为什么要有同源限制

答案：

概念:同源策略是客户端脚本（尤其是 Javascript）的重要的安全度量标准。它最早出自 Netscape Navigator2.0，其目的是防止某个文档或脚本从多个不同源装载。这里的同源策略指的是：协议，域名，端口相同，同源策略是一种安全协议

- 指一段脚本只能读取来自同一来源的窗口和文档的属性

我们举例说明：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，密码就轻松到手了。]

缺点: 现在网站的 JS 都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被 merge 后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节

### 5. $(document).ready() 是个什么函数？为什么要用它？

答案：

这个问题很重要，并且常常被问到。 ready() 函数用于在文档进入ready状态时执行代码。当DOM 完全加载（例如HTML被完全解析DOM树构建完成时），jQuery允许你执行代码。使用$(document).ready()的最大好处在于它适用于所有浏览器，jQuery帮你解决了跨浏览器的难题。

### 6. JavaScript window.onload 事件和 jQuery ready 函数有何不同？

答案：

这个问答是紧接着上一个的。JavaScript window.onload 事件和 jQuery ready 函数之间的主要区别是，前者除了要等待 DOM 被创建还要等到包括大型图片、音频、视频在内的所有外部资源都完全加载。如果加载图片和媒体内容花费了大量时间，用户就会感受到定义在 window.onload 事件上的代码在执行时有明显的延迟。

另一方面，jQuery ready() 函数只需对 DOM 树的等待，而无需对图像或外部资源加载的等待，从而执行起来更快。使用 jQuery $(document).ready() 的另一个优势是你可以在网页里多次使用它，浏览器会按它们在 HTML 页面里出现的顺序执行它们，相反对于 onload 技术而言，只能在单一函数里使用。鉴于这个好处，用 jQuery ready() 函数比用 JavaScript window.onload 事件要更好些。

## 四、Vue

### 1. Vue的双向数据绑定原理是什么？

答案：

vue.js 是采用数据劫持结合发布者-订阅者模式的方式，通过Object.defineProperty()来劫持各个属性的setter，getter，在数据变动时发布消息给订阅者，触发相应的监听回调。

具体步骤：

第一步：需要observe的数据对象进行递归遍历，包括子属性对象的属性，都加上 setter和getter

这样的话，给这个对象的某个值赋值，就会触发setter，那么就能监听到了数据变化

第二步：compile解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图

第三步：Watcher订阅者是Observer和Compile之间通信的桥梁，主要做的事情是:

1、在自身实例化时往属性订阅器(dep)里面添加自己

2、自身必须有一个update()方法

3、待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调，则功成身退。

第四步：MVVM作为数据绑定的入口，整合Observer、Compile和Watcher三者，通过Observer来监听自己的model数据变化，通过Compile来解析编译模板指令，最终利用Watcher搭起Observer和Compile之间的通信桥梁，达到数据变化 -> 视图更新；视图交互变化(input) -> 数据model变更的双向绑定效果。

### 2. 请详细说下你对vue生命周期的理解

答案：

总共分为8个阶段创建前/后，载入前/后，更新前/后，销毁前/后。

创建前/后： 在beforeCreated阶段，vue实例的挂载元素$el和数据对象data都为undefined，还未初始化。在created阶段，vue实例的数据对象data有了，$el还没有。

载入前/后：在beforeMount阶段，vue实例的$el和data都初始化了，但还是挂载之前为虚拟的dom节点，data.message还未替换。在mounted阶段，vue实例挂载完成，data.message成功渲染。

更新前/后：当data变化时，会触发beforeUpdate和updated方法。

销毁前/后：在执行destroy方法后，对data的改变不会再触发周期函数，说明此时vue实例已经解除了事件监听以及和dom的绑定，但是dom结构依然存在

### 3. 封装 vue 组件的过程

答案：

首先，组件可以提升整个项目的开发效率。能够把页面抽象成多个相对独立的模块，解决了我们传统项目开发：效率低、难维护、复用性等问题。

然后，使用Vue.extend方法创建一个组件，然后使用Vue.component方法注册组件。子组件需要数据，可以在props中接受定义。而子组件修改好数据后，想把数据传递给父组件。可以采用emit方法。

### 4. mvc与mvvm

答案：

MVC

模型－视图－控制器（Model-View-Controller）

Model和View永远不能相互通信，只能通过Controller传递。

Controller可以直接与Model对话（读写调用Model），Model通过Notification和KVO机制与Controller间接通信。

Controller可以直接与View对话，通过outlet,直接操作View,outlet直接对应到View中的控件,View通过action向Controller报告事件的发生(如用户Touch我了)。Controller是View的直接数据源（数据很可能是Controller从Model中取得并经过加工了）。Controller是View的代理（delegate),以同步View与Controller。

MVVM

Model －ViewModel － View

什么是 MVVM：一个 MVC 的增强版，我们正式连接了视图和控制器，并将表示逻辑从 Controller 移出放到一个新的对象里，即 View Model。MVVM 听起来很复杂，但它本质上就是一个精心优化的 MVC 架构

Model层是少不了的了，我们得有东西充当DTO(数据传输对象)，当然，用字典也是可以的，编程么，要灵活一些。Model层是比较薄的一层，如果学过Java的小伙伴的话，对JavaBean应该不陌生吧。

ViewModel层，就是View和Model层的粘合剂，他是一个放置用户输入验证逻辑，视图显示逻辑，发起网络请求和其他各种各样的代码的极好的地方。说白了，就是把原来ViewController层的业务逻辑和页面逻辑等剥离出来放到ViewModel层。

View层，就是ViewController层，他的任务就是从ViewModel层获取数据，然后显示。

### 5. Vue首屏加载非常慢，如何解决?

答案：

Vue首屏加载非常慢

原因: 当打包应用时，将所有JavaScript代码打包在一个文件中，导致js代码非常庞大，严重影响页面加载速度。

解决:

1. 配置打包工具，将组件分别打包到不同的js代码块中

build/webpack.base.conf.js

output:{

path: config.build.assetsRoot,

filename:’[name].js’,

//新增

chunkFilename:”[name].js”,

publicPath: process.env.NODE\_ENV===”production”

?config.build.assetsPublicPath

:config.dev.assetsPublicPath

}

2. 当路由请求到该组件时，才动态加载组件的内容

路由字典中，路由配置和以前完全一样

但是在引入组件对象时:

import Index from ‘@/views/Index.vue’

改为

const Index=()=>import(‘@/views/Index.vue’)//仅定义函数

//暂未执行

//暂时不引入Index.vue

当用户在vue中请求当前组件对应的路由地址时，由vue-router自动调用加载函数，动态请求Index.vue组件对象

### 6. 实现订阅/发布者模式？

答案：

var ublisher = {};

// 定义发布者

publisher.list =[];

// 缓存列表 存放订阅者回调函数

// 增加订阅者

publisher.listen =function(fn){

publisher.list.push(fn);

// 订阅消息添加到缓存列表

}

// 发布消息

publisher.trigger =function(){

for(vari = 0,fn; fn = this.list[i++];){

var that =this

fn.apply(null,arguments);

}

}

### 7. 什么是虚拟DOM树:

答案：

什么是: 仅包含可能变化的节点和可能变化的属性的树结构

<body>

<div id=”app”>

<img src=”logo.png”> alt title id ….

<h1>{{uname}}</h1>id class title name

<hr>

<h2>{{score}}</h2>

</div>

{

el:”#app”,

children:[

{el:”h1”, innerText:uname},

{el:”h2”,innerText:score}

]

}

为什么: 内容少，便于快速遍历比较不同

如何发挥作用:

当data中模型变量改变时，会通知虚拟DOM树

虚拟DOM树先缓存本次的修改在元素对象上

将一批修改生成新的DOM子树和原虚拟DOM树做对比。

一旦发现不同的元素或内容，就只更新有修改的元素

虚拟DOM树中，封装了传统DOM API: createElement() appendChild() .innerHTML，避免了程序员编写大量重复的代码。

### 8. Vue 如何去除url中的 #

答案：

vue-router 默认使用 hash 模式，所以在路由加载的时候，项目中的 url 会自带 #。如果不想使用 #， 可以使用 vue-router 的另一种模式 history

new Router({

mode: 'history',

routes: [ ]

})

需要注意的是，当我们启用 history 模式的时候，由于我们的项目是一个单页面应用，所以在路由跳转的时候，就会出现访问不到静态资源而出现 404 的情况，这时候就需要服务端增加一个覆盖所有情况的候选资源：如果 URL 匹配不到任何静态资源，则应该返回同一个 index.html 页面

### 9. Vue组件间如何通信:

答案：

组件通信

父组件向子组件通信

子组件通过 props 属性，绑定父组件数据，实现双方通信

子组件向父组件通信

将父组件的事件在子组件中通过 $emit 触发

非父子组件、兄弟组件之间的数据传递

/\*新建一个Vue实例作为中央事件总嫌\*/

let event = new Vue();

/\*监听事件\*/

event.$on('eventName', (val) => {

//......do something

});

/\*触发事件\*/

event.$emit('eventName', 'this is a message.')

# 第四阶段面试题

## 一、html5部分

### 1. sessionStorage和localStorage的区别

答案：

sessionStorage用于本地存储一个会话（session）中的数据，这些数据只有在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。而localStorage用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

### 2. Canvas和SVG的区别是什么

答案：

canvas绘制2d位图svg绘制2d 矢量图

canvas通过js程序指令创建图形，svg通过标签创建图形

canvas可以只能将事件绑定在画布上，svg可以将事件绑定在任意元素上

canvas应用方向，网页特效与小游戏，svg创建统计图与地图应用

### 3.如果通过canvas路径绘制一个圆心 250 200 半径为100 描边红色的圆

提示：画布id 为box 画布宽高相同500px

var can = docuemnt.getElementById("box");

var ctx = can.getContext("2d");

ctx.beginPath();

ctx.arc(250,200,100,0,2\*Math.PI);

ctx.stroke();

//创建画布，获取画笔，开始一条路径，画出宽高，描边

### 4.html5有哪些新特性举五种

canvas 绘图技术

svg 绘图技术

webworker 新线程 (服务器nodejs cluster)

websocket 网络协议(服务器 socketio)

webstorage 客户端/本地存储

drag 拖拽

### 5:html5表单中新input type 属性值有哪些

Color 颜色

Date 日期

Datetime

Email 邮箱

Time

Url 验证是否是一个网址

Range

Tel 电话

Number 数字

Search 搜索

### 6：html5中如何创建一个worker线程,以及接收和发送数据

var worker = new Worker("1.js");

worker.postMessage();

worker.onmessage = function (e)

{

e.data;

};

### 7:Local storage和cookies之间的区别是什么？

区别一:客户端和服务端都能访问数据。

Cookie的数据通过每一个请求发送到服务端

Local storage只有本地浏览器端可访问数据，

服务器不能访问本地存储直到故意通过POST或者GET的通道发送到服务器

区别二:大小

每个cookie有4095byte==4k左右

Local storage每个域5MB

区别三:过期

Cookies有有效期，所以在过期之后cookie和cookie数据会被删除

Local storage有过期数据，如需删除只能手工删除

### 8、HTML5 中如何嵌入音频？

HTML5 支持 MP3、Wav 和 Ogg 格式的音频，下面是在网页中嵌入音频的简单示例：

<audio controls>

<source src=”jamshed.mp3″ type=”audio/mpeg”>

您的浏览器不支持音频播放

</audio>

### 9、HTML5 中如何嵌入视频？

HTML5 支持 MP4、WebM 和 Ogg 格式的视频，下面是简单示例：

<video width=”450″ height=”340″ controls>

<source src=”jamshed.mp4″ type=”video/mp4″>

您的浏览器不支持视频播放

</video>

### 10、html5视频专用的css属性有哪些

object-fit

fill: "填充" 默认值 将视频拉伸操作填满空间

contain: "包含"保持原有尺寸比例，容器空白

cover: "覆盖"保持原有尺寸比例,宽度或者高度

至少一个可以与父元素一致[裁切]

### 11、Html5 视频常用的方法和事件有哪些

-controls -是否显示播放控件

-autoplay -是否自动播放

-loop -是否循环播放

-muted -是否是静音播放

-poster -在播放视频之前广告

-preload -预加载策略

取值 -auto:预加载一定时长视频与元数据

-metadata:预加载元数据

-none:不预加载数据

-volume:1 -当前音量 (0~1)

-playbackRate -回放速率 大于1快放 小于1慢播

-paused:true -当前视频是否处理暂停状态

-play() -播放视频

-pause() -暂停播放视频

-onplay -当前视频开始播放时触发事件

-onpause -当前视频暂停播放时触发事件

### 12、如何解决html5视频移动播放时同层播放问题

<video src="http://xxx.mp4" x5-video-player-type="h5"/>

### 13、Canvas 绘图技术常用路径方法与属性有哪些

ctx.beginPath(): 开始路径

ctx.closePath(); 结束路径

ctx.moveTo(x,y); 从哪个点开始

ctx.lineTo(x,y); 到哪个点结束

ctx.stroke(); 描边

ctx.fill(); 填充

ctx.arc(cx,cy,r,start,end); 圆形

### 14、Canvas 绘图技术常用图像方法与属性有哪些

var p3 = new Image(); 获取画布

p3.src = "x.png" 设置路径

p3.onload = function(){ //加载完毕

ctx.drawImage(p3,x,y); //绘制

}

### 15、Canvas 绘图技术常用变形方法与属性有哪些

--rotate(deg)

-translate(x,y)

-save()

-restore();

### 16、Svg 绘图技术例举三个常用标签

<ellipse rx="坐标" ry="坐标" cx="水平方向" cy="垂直方向"></ellipse> //椭圆

<line x1="坐标1" y1="坐标1" x2="坐标2" y2="坐标2" stroke-width="" stroke-linecap="" stroke=""></line> //线

<rect x="" y="" width="宽" height="高" fill="" fill-opacity=""></rect> //矩形

<circle cx="" cy="" r=""></circle> //圆形

### 17、Canvas 绘图技术如何绘制贝赛尔曲线

moveTo(x,y); 起点坐标

quadraticCurveTo(cp1x,cp1y,x,y); 控制点和终点坐

### 18、Canvas 绘图技术如何使用平滑动画处理

window.requestAnimationFrame()

### 19、Html5拖放api提供哪些事件

-拖动源对象

dragstart 拖动开始

drag 拖动中

dragend 拖动结束

-拖动目标对象(不会动)

dragenter 拖动进入

dragover 拖动悬停

dragleave 拖动离开

drop 释放

### 20、Html5 地于定位对象与方法有哪些

navigator.geolocation.getCurrentPosition

navigator.geolocation.watchPosition()

navigator.geolocation.clearWatch();

### 21说说HTML5，CSS3的新特性，使用过哪些？

答案：

H5新特性：

结构标记（header，footer，section，article，aside，nav），

新表单元素（url，email，number，date，month，color等），

canvas，audio，video，地理定位，拖放，web存储，Web Workers，WebSocket等。

CSS3新特性：

复杂选择器器（属性选择器，伪类选择器，伪元素选择器，兄弟选择器等），

边框倒角，边框阴影，渐变，转换，过渡，动画，弹性盒子，媒体查询技术等。

## 二、vue 与vue组件部分

### 1. vuex作用？哪种功能场景使用它

答案：

vue框架中状态管理。

场景有：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

### 2.解释vuex最常用的二种属性

答案：

分别是 State、 Getter

vuex的State特性

A、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，对应于一般Vue对象里面的data

B、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

C、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

vuex的Getter特性

A、getters 可以对State进行计算操作，它就是Store的计算属性

B、 虽然在组件内也可以做计算属性，但是getters 可以在多组件之间复用

C、 如果一个状态只在一个组件内使用，是可以不用getters

### 3.请列举出3个Vue中常用的生命周期钩子函数

答案：

created: 实例已经创建完成之后调用,在这一步,实例已经完成数据观测, 属性和方法的运算, watch/event事件回调. 然而, 挂载阶段还没有开始, $el属性目前还不可见

mounted: el被新创建的 vm.$el 替换，并挂载到实例上去之后调用该钩子。如果 root 实例挂载了一个文档内元素，当 mounted 被调用时 vm.$el 也在文档内。

activated: keep-alive组件激活时调用

### 4.mint-ui是什么？怎么使用？至少说出三个组件

答案：

基于 vue 的前端组件库。npm 安装，然后 import 样式和js，

Vue.use(mintUI)全局引入。

-在单个组件内部引入：import { Toast } from ‘mint-ui’

-

组件1：Toast(‘登录成功’)；

组件2：mint-header

组件3：mint-swiper

### 5.vue-loader是什么？使用它的用途有哪些

答案：

解析.vue文件的一个加载器，跟template/js/style转换成js模块。

用途：js可以写es6、style样式可以scss或less、template可以加jade等

### 6.axios的特点有哪些

答案：

Axios 是一个基于 promise 的 HTTP 库，支持promise所有的API

它可以拦截请求和响应

它可以转换请求数据和响应数据，并对响应回来的内容自动转换成 JSON类型的数据

安全性更高，客户端支持防御 XSRF

### 7.axios有哪些常用方法

答案：

axios.get(url[, config]) //get请求用于列表和信息查询

axios.delete(url[, config]) //删除

axios.post(url[, data[, config]]) //post请求用于信息的添加

axios.put(url[, data[, config]]) //修改/更新操作

### 8. Vuex的理解

答案：Vuex是一个专为Vue开发的应用程序的状态管理模式，它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化

核心概念：

Store：要使用Vuex，我们要创建一个实例 store ，我们称之为仓库，利用这个仓库 store 来对我们的状态进行管理

State

Vuex 使用单一状态树，用一个对象State包含了整个应用层级的所有状态，你可以理解为这些状态就是一堆全局变量和数据

Getter

还记得Vue中的计算属性computed 吗？当我们需要在data的基础上再派生一些状态的时候，就会使用computed 来实现。同样，当我们需要从 state 中派生出一些状态的时候，就会使用到getters，你可以将 getters 理解state的计算属性。

Mutation

Vuex相比传统的全局变量有两个优势，其中一个是Vuex中的状态不能随意修改，要修改必要按照Vuex提供的方式才能修改

Action

Action 类似于 mutation，不同在于：

Action 提交的是 mutation，而不是直接变更状态。

Action 可以包含任意异步操作，通过 store.dispatch 方法触发

### 9.Vue和angular的区别

答案：

Vue：上手容易 指令以 v-xxx html+json,然后再new Vue()实例化对象 个人维护项目

适合移动端

Angular：上手难 指令以 ng-xxx 列表内容 所有属性和方法都挂载到$scope上

由google维护 适合web端

### 10.vue 两种模式

答案：

vue-router里面除了要配置路由词典外 还要选择 模式 有 hash模式和history模式

### 11：vue的生命周期

答案：

总共8个阶段：

创建前/后: 创建模型数据和响应系统

beforeCreated 挂在元素$el：undefined 数据data：undefined

Created 挂在元素$el：undefined 数据data：{.....}

载入前/后: 挂载 创建虚拟DOM 并生成真实DOM

认为Vue实例/组件首次加载完成

可以发送ajax请求

也可以操作DOM元素

beforeMount:挂载元素$el undefined 数据data {.....}

Mounted: 挂载元素$el DOM 数据data{....}

更新前/后: 模型数据发生变化时触发

BeforeUpdate：

Updated：

销毁前/后: destroy()方法被调用时，要销毁组件/实例时

beforeDestroy

Destroyed

### 12.v-show和v-if指令的共同点和不同点?

v-show指令是通过修改元素的displayCSS属性让其显示或者隐藏

v-if指令是直接销毁和重建DOM达到让元素显示和隐藏的效果

## 三、微信小程序

### 1. 微信小程序组件的生命周期

答案：

生命周期函数:

-onLoad: 页面加载

一个页面只会调用一次，可以在 onLoad 中获取打开当前页面所调用的 query 参数。

-onShow: 页面显示

每次打开页面都会调用一次。

-onReady: 页面初次渲染完成

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

当navigateTo或底部tab切换时调用。

-onUnload: 页面卸载

### 2 简单描述下微信小程序的相关文件类型

答案：

微信小程序项目结构主要有四个文件类型,如下

.json 后缀的 JSON 配置文件

.wxml 后缀的 WXML 模板文件

.wxss 后缀的 WXSS 样式文件

.js 后缀的 JS 脚本逻辑文件

### 3、app.json 是对当前小程序的全局配置，讲述三个配置各个项的含义

答案：

pages字段 —— 用于描述当前小程序所有页面路径，这是为了让微信客户端知道当前你的小程序页面定义在哪个目录。

window字段 —— 小程序所有页面的顶部背景颜色，文字颜色定义在这里的

tab字段—小程序全局顶部或底部tab

### 4.小程序的wxss和css有哪些不一样的地方

答案：

新像素单位 rpx

样式可直接使用import导入

### 5:小程序有哪些参数传值的方法?

答案：

1.给HTML元素添加data-\*属性来传递我们需要的值，然后通过e.currentTarget.dataset或onload的param参数获取。

但data-名称不能有大写字母和不可以存放对象；

2.设置id 的方法标识来传值通过e.currentTarget.id获取设置的id的值,然后通过设置全局对象的方式来传递数值；

3.在navigator中添加参数传值。

### 6：小程序有哪些参数传值的方法？

答案：

一、给HTML元素添加data-\*属性来传递我们需要的值，然后通过e.currentTarget.dataset或onload的param参数获取。

但data-名称不能有大写字母和不可以存放对象

二、设置id 的方法标识来传值通过e.currentTarget.id获取设置的id的值,然后通过设置全局对象的方式来传递数值

三、在navigator中添加参数传值

### 7：如何提高微信小程序的应用速度？

答案：

一、提高页面加载速度

二、用户行为预测

三、减少默认data的大小

四、组件化方案

### 8.微信小程序与H5的区别？

答案：

1.运行环境的不同

传统的HTML5的运行环境是浏览器，包括webview，而微信小程序的运行环境并非完整的浏览器，是微信开发团队基于浏览器内核完全重构的一个内置解析器，针对小程序专门做了优化，配合自己定义的开发语言标准，提升了小程序的性能。

2.开发成本的不同

只在微信中运行，所以不用再去顾虑浏览器兼容性，不用担心生产环境中出现不可预料的奇妙BUG

3.获取系统级权限的不同

所有系统级权限都可以和微信小程序无缝衔，微信小程序获取的系统权限更高。

### 9小程序获取场景值的方法

let options = wx.getLanchOptionsSync()

### 10 小程序onPageScroll方法的使用注意什么

由于此方法调用频繁，不需要时，可以去掉，不要保留空方法，并且使用onPageScroll时，尽量避免使用setData()，尽量减少setData()的使用频次。

### 11 小程序视图渲染结束回调

使用setData(data, callback)，在callback回调方法中添加后续操作代码

### 12小程序同步API和异步API使用时注意事项

同步API使用try-catch来查看异常，

异步API通过回调函数来获取结果。

==

像wx.setStorageSync这种以Sync结尾的API为同步API，使用时使用try-catch来查看异常，

如果判定API为异步，可以在其回调方法success、fail、complete中进行下一步操作。

### 13小程序对wx:if 和 hidden使用的理解

wx:if 有更高的切换消耗。

hidden 有更高的初始渲染消耗。

因此，如果需要频繁切换的情景下，用 hidden 更好，如果在运行时条件不大可能改变则 wx:if 较好。

### 14.小程序简单介绍下三种事件对象的属性列表

基础事件（BaseEvent）

type: 事件类型

timeStamp：事件生成时的时间戳

target：触发事件的组件的属性值集合

currentTarget：当前组件的一些属性集合

自定义事件（CustomEvent）

detail

触摸事件（TouchEvent）

touches

changedTouches

### 15.小程序的双向绑定和vue的异同？

答案：大体相同,

但小程序直接this.data的属性是不可以同步到视图的，必须调用this.setData()方法！

vue:v-model

### 16小程序页面间有哪些传递数据的方法？

答案：

使用全局变量实现数据传递

页面跳转或重定向时，使用url带参数传递数据

使用组件模板 template传递参数

使用数据库传递数据

### 17.小程序wxml与标准的html的异同？

答案：

都是用来描述页面的结构；

都由标签、属性等构成；

标签名字不一样，且小程序标签更少，单一标签更多；

多了一些 wx:if 这样的属性以及 {{ }} 这样的表达式

WXML仅能在微信小程序开发者工具中预览，而HTML可以在浏览器内预览

组件封装不同， WXML对组件进行了重新封装，

小程序运行在JS Core内，没有DOM树和window对象，小程序中无法使用window对象和document对象。

### 18.小程序WXSS和CSS的异同？

答案：都是用来描述页面的样子；

WXSS 具有 CSS 大部分的特性，也做了一些扩充和修改；

WXSS新增了尺寸单位，WXSS 在底层支持新的尺寸单位 rpx；

WXSS 仅支持部分 CSS 选择器；

WXSS 提供全局样式与局部样式

# 第五阶段面试题

## 一、Angular

### 1. Angular中组件之间通信的方式

答案：

Props down

1、调用子组件时 通过自定义属性传值

2、子组件内部通过Input来接受属性的值

Events up

1.在父组件中定义一个有参数的方法

2.调用子组件时，绑定自定义事件和上一步的方法

3.子组件内部通过Output和EventEmitter来触发事件并传值

### 2. Angular的八大组成部分并简单描述

答案：

Module 是Angular开发中的基本单位，是一个容器，可以包含组件、指令、管道等

Components 是可被反复使用的 带有特定功能的视图

Templates 是经过指令和管道、组件等增强过的html

Bindings 结合着事件绑定 属性绑定 双向数据绑定等扩展html的功能

Directives 分为结构性和属性型指令还有其他模块中比如路由模块中的指令等，主要是增强html

Pipes 可以对数据做一个筛选、过滤、格式化从而得到目的数据

Service 将组件、应用中的可共用的部分 ，比如数据或者方法等 封装成服务以方便服用

DependencyInjection 依赖注入

### 3、Angular中常见的生命周期的钩子函数?

ngOnChanges：当Angular设置其接收当前和上一个对象值的数据绑定属性时响应。

ngOnInit：在第一个ngOnChange触发器之后，初始化组件/指令。这是最常用的方法，用于从后端服务检索模板的数据。

ngDoCheck：检测并在Angular上下文发生变化时执行。每次更改检测运行时，会被调用。

ngOnDestroy：在Angular销毁指令/组件之前清除。取消订阅可观察的对象并脱离事件处理程序，以避免内存泄漏。

组件特定hooks：

ngAfterContentInit：组件内容已初始化完成

ngAfterContentChecked：在Angular检查投影到其视图中的绑定的外部内容之后。

ngAfterViewInit：Angular创建组件的视图后。

ngAfterViewChecked：在Angular检查组件视图的绑定之后。

### 4、Angular中路由的工作原理

Angular应用程序具有路由器服务的单个实例，并且每当URL改变时，相应的路由就与路由配置数组进行匹配。在成功匹配时，它会应用重定向，此时路由器会构建ActivatedRoute对象的树，同时包含路由器的当前状态。在重定向之前，路由器将通过运行保护（CanActivate）来检查是否允许新的状态。Route Guard只是路由器运行来检查路由授权的接口方法。保护运行后，它将解析路由数据并通过将所需的组件实例化到<router-outlet> </ router-outlet>中来激活路由器状态。

### 5、解释rxjs在Angular中的使用场景

Rxjs是微软所提供的一种的异步处理数据的方式，在Angular中处理网络通信时用到了。

创建一个Observable并subsribe

比如: this.http.get(‘’).subscribe((data)=>{})

## 二、React

## 1. ReactJS中渲染根组件的方式以及注意事项

答案：

 ReactDOM.render(A,B);将A渲染到B指定的容器中

注意事项:

不允许一次渲染多个标签，但是可以放在同一个顶层标签

每一个标记都要有对应的结束

### 2.ReactJS中父子组件通信的方式

(1)父与子通信

借助于属性 往下传值

传值:

<son myName=”123”></son>

接受值:

this.props.myName

(2)子与父通信

通过属性传递有参数的方法 让子组件调用是传值

①定义有参数的方法

rcvMsg(msg){}

②传递给子组件

<son funcRcv={this.rcvMsg}></son>

③子组件来调用

This.props.funcRcv(123)

### 3.如何在组件渲染时调用组件内部嵌套的子组件

This.props.children

### 4.组件的生命周期

   mount:

    componentWillMount

    componentDidMount

   update:

    componentWillUpdate

    componentDidUpdate

    componentWillReceiveProps

   unmount:

    componentWillUnmount

### 5.在组件渲染时 实现判断和循环功能

(1)判断

短路逻辑

 {      expression      &&     <Any></Any>  }

(2)循环

遍历集合

 this.state.list

      .map((value,index)=>{

        return <li key={index}>

          {value}

        </li>

    })

### 6.描述ReactNative的特点和开发理念

ReactNative是实现原生开发的框架

可以使用react的语法和js来编写

开发理念:

Learn once,write anywhere

### 7、react中是如何处理网络通信的

fetch(url).then((response)=>response.json()).then((result)=>{})

### 8、react中循环创建多个组件时指定key的作用？

在dom变化时 快速定位元素 提升更新效率

### 9、react的生态圈(技术栈)中有哪些常见技术?

reactjs、 reactNative、 react360 、flux、 redux、 ssr、 reactNavigation....

### 10、基于reactNative的reactNavigation中的基础用法？

跳转:

this.props.navigation.navigate()

传参:

this.props.navigation.navigate(‘detail’,{id:10})

this.props.navigation.getParam(‘id’)

### 11、reactNative中如何实现一个高性能列表

import {FlatList} from ‘react-native’

showItem=(info)=>{

return \*\*\*\*

}

<FlatList data={[1,2,3]} renderItem=”showItem”></FlatList>

### 12、reactNative中如何完成自定义布局

可以使用c3中的flexbox

## 三、Python

### 1. 列出python的常见基础数据类型

答案：

数值（整数 浮点数 布尔值）字符串、列表、元组、字典、集合

### 2 文件常用操作方法

Open/read/readline/readlines/write/close

### 3、异常的处理

Try :

pass

except 异常类型

pass

finally:

pass

### 4、有哪些常见的标准库和第三方库

标准库

re time datetime urllib...

第三方库

numpy Pandas Matplotlib tensorflow beautifulSoup

## 四、Ng

### 1、ng中\*ngFor和\*ngIf的使用注意事项

不能在一个元素上同时使用结构行指令，同时使用 可以通过ng-container来避免报错

### 2、ng的生态圈(技术栈)中常见技术

Angular官方集成的各种模块：FormsModule,HttpClientModule、rxjs。。

第三方技术:ionic\NgZorro\AngularMaterial

### 3、ng中如何实现一个服务

①通过cli工具 创建服务：ng g service my-service

②在my-service.service.ts中封装数据和方法即可

### 4、ng中管道的本质

管道的本质是有参数有返回值的方法

### 5、ng中如何使用路由守卫

ng中是通过封装服务实现CanActivate来完成一个路由功能的

### 6、描述下angular中路由的背后原理(spa的工作方式)

监控地址栏的变化

解析地址栏中的路由地址

将路由地址和路由数组中的每个路有对象的path属性进行匹配，如果匹配上就会加载

的视图内容到routerOutlet中进行显示

### 7、Python中数组常见的api

myList = [1,2,3]

myList.insert/append/pop/clear/

### 8、Python中如何使用mysql-connector完成一个数据库的连接池的创建

pip i mysql-connector

import mysql.connector.pooling as pooling

myPool = pooling.MySQLConnectionPool({})

### 9、Python中使用mysql-connector完成数据的增删改查

pip i mysql-connector

import mysql.connector

myConnection = mysql.conenctor.connect()

myCursor = myConnection.cursor()

myCursor.executr(‘sql语句’)

### 10、Django中路由系统搭建的基本步骤

①创建视图

Views.py

②配置二级路由

新建urls.py

③配置一级路由

设置project/urls.py

### 11、Django中如何处理get/post请求中的参数

get ： req.GET[‘id’]

post: req.body.decode(‘utf-8’)

### 12、Django中如何处理跨域问题?

可借助于一个第三方的包:django-cors-headers

设置 CORS\_ORIGIN\_WHITELIST

### 13、Django中模板系统的基础用法

①创建模板文件

app/tempaltes/app/test.html

②引入载入器，准备载入

from ‘django.tempalte ’ import loader

tpl = loader.get\_template(‘./app/test.html’)

③渲染

tpl.render()

### 14、解释Django中遵循的MVT设计模式

M:Models

V:Views

T:Templates

Views是桥梁，可以从Models中取到数据，然后渲染在Templates中

### 15、简述下Django中ORM的基本用法

ORM是一种对象关系映射技术，可以使用对象的方法来完成数据库中数据的CRUD操作

①创建模型类

②激活模型类

③调用模型类

### 16、使用框架有什么优势

提高代码的复用率

提高代码的开发质量

提高代码的开发速度

### 17、简述vue、angular、react各自使用场景

vue因为上手快、语法简洁 可以被快速的使用在中小型项目中

anguar模块多，学习曲线陡峭，更多的用在中大型超大型的项目中

react被用在实现一些对于DOM操作非常频繁的场景里

### 18、描述常见的异步处理方案

Promise 、Async\Await、RXJS、Ajax

### 19、如果封装一个自己的前端MVVM框架，你会怎么设计,可以简单说下思路

核心思路是绑定的实现，可以借助于现有的观察者模式配合数据流的监听，当数据变化时，去通过对比算法，快速定位哪些视图调用了数据，然后更新DOM