# 面试题总结

此版本用于助教老师（助教老师根据实际情况考虑给不给学员答案）

使用于带班期间晚自习或者强制仔细学生演讲时使用

总结有常用面试基础题和学员们从面试经历中带回来的

**特殊标注的为末尾学员必须掌握的，达到讲过以后随时随地问都可以流利回答**

## 一、基础班模块

### 基础部分

#### 1. 什么是HTML？

HTML并不是真正的的程序语言，他是一种 标 记 语 言 ，用来结构化和含义化你想要放

在web 网站上的那些内容。它由一系列的元素（elements）所组成，这些元素可以用来

封装你的内容中担任不同工作的各部分和各个角色。

#### 2. 什么是CSS？

就像 HTML，CSS 也不是真正的编程语言。它是样式表语言，也就是说，它允许你有

选择性的为 HTML 文档的元素添加样式。

#### 3. 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。

#### 4. 简述一下你对HTML语义化的理解？

1. HTML语义化让页面的内容结构化，结构更清晰，便于对浏览器、搜索引擎解析；

2. 即使在没有样式CSS 的情况下也能以一种文档格式显示，并且是容易阅读的；

3. 搜索引擎的爬虫也依赖于HTML标记来确定上下文和各个关键字的权重，有利于SEO；

4. 使阅读源代码的人更容易将网站分块，便于阅读、维护和理解；

#### 5. rgba() 和 opacity 设置透明度的区别是什么？

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！）

#### 6. DOCTYPE的作用？

1. <!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器以

何种模式来渲染文档。

1. 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。

2. 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。

1. DOCTYPE 不存在或格式不正确会导致文档以混杂模式呈现。

#### 7. 介绍一下你对浏览器内核的理解？都有哪些常见的浏览器内核？

要或者说核心的部分是“Rendering Engine”，可大概译为“渲染引擎”，不过我们

一般习惯将之称为“浏览器内核”。负责对网页语法的解释（如标准通用标记语言下的一个

应用 HTML、JavaScript）并渲染（显示）网页。 所以，通常所谓的浏览器内核也就是浏

览器所采用的渲染引擎，渲染引擎决定了浏览器如何显示网页的内容以及页面的格式信息。

不同的浏览器内核对网页编写语法的解释也有不同，因此同一网页在不同的内核的浏览器里

的渲染（显示）效果也可能不同，这也是网页编写者需要在不同内核的浏览器中测试网页显

示效果的原因。

常见浏览器内核：

Trident 内核：IE,MaxThon,TT,The World,360,搜狗浏览器等。[又称 MSHTML]

Gecko内核：Netscape6 及以上版本，FF,MozillaSuite/SeaMonkey 等。

Presto 内核：Opera7及以上。 [Opera内核原为：Presto，现为、Blink;]

Webkit 内核：Safari,Chrome 等。 [ Chrome 的：Blink（WebKit 的分支）]

EdgeHTML内核：Microsoft Edge。 [此内核其实是从 MSHTML fork 而来，删掉了几

乎所有的 IE私有特性

#### 8. CSS选择器权重如何计算？

页面显示样式的优先级取决于其“特殊性”’，特殊性越高，就显示最高的，当特殊性相等

时，显示后者

特殊性表述为 4个部分：0,0,0,0

一个选择器的特殊性如下:

对于选择器是#id的属性值,特殊性值为：0,1,0,0

对于属性选择器，class或伪类，特殊性值为：0,0,1,0

对于标签选择器或伪元素，特殊性值为：0,0,0,1

通配符‘\*’特殊性值为：0,0,0,0

内联样式特殊性值为：1,0,0,0

#### 9. 对WEB标准以及W3C的理解与认识？

1. 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、结构行为表现的分离，

2. 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，

3. 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。

#### 10. CSS中优雅降级和渐进增强有什么区别？

优雅降级和渐进增强印象中是随着css3 流出来的一个概念。由于低级浏览器不支持

css3，但 css3 的效果又太优秀不忍放弃，所以在高级浏览中使用 css3 而低级浏览器只保证

最基本的功能。咋一看两个概念差不多，都是在关注不同浏览器下的不同体验，关键的区别

是他们所侧重的内容，以及这种不同造成的工作流程的差异。

“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。

“渐进增强”观点则认为应关注于内容本身。

#### 11. 对BFC规范的理解有哪些？

1. 定义：

BFC(Block formatting context)直译为"块级格式化上下文"。它是一个独立的渲染区域，

只有 Block-level box 参与，它规定了内部的Block-level Box 如何布局，并且与这个区域

外部毫不相干。

布局规则：

A. 内部的Box 会在垂直方向，一个接一个地放置。

B. Box垂直方向的距离由 margin决定。属于同一个 BFC的两个相邻 Box的 margin 会发

生重叠。

C. 每个元素的 margin box 的左边，与包含块border box 的左边相接触(对于从左往右的

格式化，否则相反)。即使存在浮动也是如此。

D. BFC 的区域不会与 float box 重叠。

E. BFC 就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。反

之也如此。

F. 计算 BFC 的高度时，浮动元素也参与计算。

2. 哪些元素会生成 BFC：

A. 根元素

B. float 属性不为 none

C. position为 absolute 或 fixed

D. display为inline-block， table-cell， table-caption， flex， inline-flex

F. overflow 不为 visible

#### 12. 有多少种清除浮动的方法？

1. 父级div定义 height

原理：父级 div手动定义 height，就解决了父级 div无法自动获取到高度的问题。 简单、

代码少、容易掌握 ，但只适合高度固定的布局.

2. 结尾处加空 div标签 clear：both

原理：在浮动元素的后面添加一个空div兄弟元素，利用 css 提高的clear：both清除浮动，

让父级div能自动获取到高度 ，如果页面浮动布局多，就要增加很多空 div，让人感觉很

不好 .

3. 父级div定义 伪类：after 和 zoom

/清除浮动代码/

.clearfix：after{

content：""；

display：block；

visibility：hidden；

height：0；

line-height：0；

clear：both；

}

.clearfix{zoom：1}

原理：IE8以上和非 IE浏览器才支持：after，原理和方法 2有点类似，zoom(IE专有属性)

可解决ie6，ie7 浮动问题 ，推荐使用，建议定义公共类，以减少CSS 代码。

4. 父级div定义 overflow：hidden

超出盒子部分会被隐藏，不推荐使用.

5. 双伪元素法：

.clearfix：before，.clearfix：after {

content： ""；

display： block；

clear：both；

}

.clearfix {

zoom： 1；}

### 实际工作部分

#### 1. HTML常见兼容性问题？

1. 双边距BUG float引起的 使用display

2. 3像素问题 使用float引起的 使用dislpay:inline -3px

3. 超链接hover 点击后失效 使用正确的书写顺序 link visited hover active

4. Iez-index问题 给父级添加position:relative

5. Png透明 使用js代码改

6. Min-height最小高度 ！Important解决’

7. select在ie6下遮盖使用iframe嵌套

8. 为什么没有办法定义1px左右的宽度容器（IE6默认的行高造成的，使用over:hidden,zoom:0.08line-height:1px）

9. IE5-8不支持opacity，解决办法：

.opacity{

opacity: 0.4

filter: alpha(opacity=60); /\* for IE5-7 \*/

-ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=60)"; /\* for IE 8\*/

}

10. IE6不支持PNG透明背景，解决办法: IE6下使用gif图片

#### 2. 描述一个"reset"的CSS文件并如何使用它。知道`normalize.css`吗？你了解他们的不同之处？

重置样式非常多，凡是一个前端开发人员肯定有一个常用的重置CSS文件并知道如何使用它们。他们是盲目的在做还是知道为什么这么做呢？原因是不同的浏览器对一些元素有不同的默认样式，如果你不处理，在不同的浏览器下会存在必要的风险，或者更有戏剧性的性发生。

　　你可能会用Normalize来代替你的重置样式文件。它没有重置所有的样式风格，但仅提供了一套合理的默认样式值。既能让众多浏览器达到一致和合理，但又不扰乱其他的东西（如粗体的标题）。

　　在这一方面，无法做每一个复位重置。它也确实有些超过一个重置，它处理了你永远都不用考虑的怪癖，像HTML的audio元素不一致或line-height不一致。

#### 3. BFC是什么？

BFC（块级格式化上下文），一个创建了新的BFC的盒子是独立布局的，盒子内元素的布局不会影响盒子外面的元素。在同一个BFC中的两个相邻的盒子在垂直方向发生margin重叠的问题

BFC是指浏览器中创建了一个独立的渲染区域，该区域内所有元素的布局不会影响到区域外元素的布局，这个渲染区域只对块级元素起作用

#### 4. 怎样实现三栏布局，两边宽度固定，中间自适应？

圣杯布局 双飞翼布局

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

\* {

margin: 0;

padding: 0;

}

#left {

width: 200px;

height: 200px;

float: left;

background-color: red;

}

#right {

width: 150px;

height: 200px;

float: right;

background-color: mistyrose;

}

#middle {

height: 200px;

margin: 0 150px 0 200px;

background-color: saddlebrown;

word-break: break-word;

}

</style>

</head>

<body>

<div id="content">

<div id="left">我是左侧内容我是左侧内容我是左侧内容我是左侧内容我是左侧内容</div>

<div id="right">我是右侧内容我是右侧内容我是右侧内容我是右侧内容我是右侧内容我是右侧内容</div>

<div id="middle">我是中间内容我是中间内容我是中间内容我是中间内容我是中间内容我是中间内容我是中间内容我是中间内容我是中间内容我是中间内容</div>

</div>

</body>

</html>

#### 5. 精灵图(CSS Sprites)的优点和缺点

精灵图是一种网页图片应用处理方式。就是把网页中很多小背景图片整合到一张图片文件中，再利用CSS的“background-image”，“background-repeat”，“background-position”的组合进行背景图显示及定位，达到显示某一部分背景图的效果。

**精灵图的优点：**

1. 减少图片的体积，因为每个图片都有一个头部信息，把多个图片放到一个图片里，就会共用同一个头部信息，从而减少了字节数。

2. 减少了网页的http请求次数，从而加快了网页加载速度，提高用户体验。

3. 解决了网页设计师在图片命名上的困扰，只需对一张集合的图片上命名就可以了，不需要对每一个小元素进行命名，从而提高了网页的制作效率。

4. 更换风格方便，只需要在一张或少张图片上修改图片的颜色或样式，整个网页的风格就可以改变。维护起来更加方便。

**精灵图的缺点：**

1. 在图片合并的时候，你要把多张图片有序的合理的合并成一张图片，还要留好足够的空间，防止板块内出现不必要的背景；这些还好，最痛苦的是在宽屏，高分辨率的屏幕下的自适应页面，你的图片如果不够宽，很容易出现背景断裂；

2. 在开发的时候比较麻烦，你要通过photoshop或其他工具测量计算每一个背景单元的精确位置，这是针线活，没什么难度，但是很繁琐；

3. 在维护的时候比较麻烦，如果页面背景有少许改动，一般就要改这张合并的图片，无需改的地方最好不要动，这样避免改动更多的css，如果在原来的地方放不下，又只能（最好）往下加图片，这样图片的字节就增加了，还要改动css。

4. 精灵图不能随意改变大小和颜色。精灵图改变大小会失真模糊，降低用户体验，css3新属性可以改变精灵图颜色，但是比较麻烦，并且新属性有兼容问题。现在一般都是用web字体(图标字体)来代替精灵图。

## 二、JavaScript基础模块

### 基础部分

#### 1. JS中有哪些数据类型？

简单数据类型：Undefined、Null、Boolean、Number 和String。

复杂数据类型：Object

#### 2. "==" 和 "===" 的区别？

前者会自动转换类型,而后者不会。

前者比较的是值，后者比较的是值和类型。

#### 3. JS中的常用内置对象有哪些？并列举该对象的常用方法？

1. Arguments 函数参数集合

arguments[ ] 函数参数的数组

Arguments 一个函数的参数和其他属性

Arguments.callee 当前正在运行的函数

Arguments.length 传递给函数的参数的个数

1. Array 数组

length属性 动态获取数组长度

join() 将一个数组转成字符串。返回一个字符串。

reverse() 将数组中各元素颠倒顺序

delete运算符 只能删除数组元素的值，而所占空间还在，总长度没变(arr.length)。

shift() 删除数组中第一个元素，返回删除的那个值，并将长度减 1。

pop() 删除数组中最后一个元素，返回删除的那个值，并将长度减1。

unshift() 往数组前面添加一个或多个数组元素，长度要改变。

push() 往数组结尾添加一个或多个数组元素，长度要改变。

concat( ) 连接数组

slice( ) 返回数组的一部分

sort( ) 对数组元素进行排序

splice( ) 插入、删除或替换数组的元素

toLocaleString( ) 把数组转换成局部字符串

toString( ) 将数组转换成一个字符串

2. Boolean 布尔对象

Boolean.toString( ) 将布尔值转换成字符串

Boolean.valueOf( ) Boolean对象的布尔值

3. Error 异常对象

Error.message 可以读取的错误消息

Error.name 错误的类型

Error.toString( ) 把Error 对象转换成字符串

EvalError 在不正确使用 eval()时抛出

SyntaxError 抛出该错误用来通知语法错误

RangeError 在数字超出合法范围时抛出

ReferenceError 在读取不存在的变量时抛出

TypeError 当一个值的类型错误时，抛出该异常

URIError 由URl的编码和解码方法抛出

4. Function 函数构造器

Function 函数构造器

Function.apply( ) 将函数作为一个对象的方法调用

Function.arguments[] 传递给函数的参数

Function.call( ) 将函数作为对象的方法调用

Function.caller 调用当前函数的函数

Function.length 已声明的参数的个数

Function.prototype 对象类的原型

Function.toString( ) 把函数转换成字符串

5. Math 数学对象

Math对象是一个静态对象

Math.PI 圆周率。

Math.abs() 绝对值。

Math.ceil() 向上取整(整数加 1，小数去掉)。

Math.floor() 向下取整(直接去掉小数)。

Math.round() 四舍五入。

Math.pow(x，y) 求 x的y次方。

Math.sqrt() 求平方根。

6. Number 数值对象

Number.MAX\_VALUE 最大数值

Number.MIN\_VALUE 最小数值

Number.NaN 特殊的非数字值

Number.NEGATIVE\_INFINITY 负无穷大

Number.POSITIVE\_INFINITY 正无穷大

Number.toExponential( ) 用指数计数法格式化数字

Number.toFixed( ) 采用定点计数法格式化数字

Number.toLocaleString( ) 把数字转换成本地格式的字符串

Number.toPrecision( ) 格式化数字的有效位

Number.toString( ) 将—个数字转换成字符串

Number.valueOf( ) 返回原始数值

7. Object 基础对象

Object 含有所有 JavaScript 对象的特性的超类

Object.constructor 对象的构造函数

Object.hasOwnProperty( ) 检查属性是否被继承

Object.isPrototypeOf( ) 一个对象是否是另一个对象的原型

Object.propertyIsEnumerable( ) 是否可以通过 for/in循环看到属性

Object.toLocaleString( ) 返回对象的本地字符串表示

Object.toString( ) 定义一个对象的字符串表示

Object.valueOf( ) 指定对象的原始值

8. RegExp 正则表达式对象

RegExp.exec( ) 通用的匹配模式

RegExp.global 正则表达式是否全局匹配

RegExp.ignoreCase 正则表达式是否区分大小写

RegExp.lastIndex 下次匹配的起始位置

RegExp.source 正则表达式的文本

RegExp.test( ) 检测一个字符串是否匹配某个模式

RegExp.toString( ) 把正则表达式转换成字符串

9. String 字符串对象

Length 获取字符串的长度。

toLowerCase() 将字符串中的字母转成全小写。

toUpperCase() 将字符串中的字母转成全大写。

charAt(index) 返回指定下标位置的一个字符。如果没有找到，则返回空字符串。

substr() 在原始字符串，返回一个子字符串

substring() 在原始字符串，返回一个子字符串。

split() 将一个字符串转成数组。

charCodeAt( ) 返回字符串中的第 n个字符的代码

concat( ) 连接字符串

fromCharCode( ) 从字符编码创建—个字符串

indexOf( ) 返回一个子字符串在原始字符串中的索引值(查找顺序从左往右查找)。如果没 有找到，则返回-1。

lastIndexOf( ) 从后向前检索一个字符串

localeCompare( ) 用本地特定的顺序来比较两个字符串

match( ) 找到一个或多个正则表达式的匹配

replace( ) 替换一个与正则表达式匹配的子串

search( ) 检索与正则表达式相匹配的子串

slice( ) 抽取一个子串

toLocaleLowerCase( ) 把字符串转换小写

toLocaleUpperCase( ) 将字符串转换成大写

toLowerCase( ) 将字符串转换成小写

toString( ) 返回字符串

toUpperCase( ) 将字符串转换成大写

valueOf( ) 返回字符串

#### 4. 什么是闭包？

简单的说，作用域是针对变量的，比如我们创建一个函数a1，函数里面又包了一 个子函数 a2。此时就存在三个作用域： 全局作用域、a1作用域、a2 作用域；即全局作用域包含了a1的作用域，a2 的作用 域包含了 a1的作用域。 当a1 在查找变量的时候会先从自身的作用域区查找，找不到再到上一级a2 的作用域 查找，如果还没找到就到全局作用域区查找，这样就形成了一个作用域链。 理解闭包首先要理解，js 垃圾回收机制，也就是当一个函数被执行完后，其作用域会被 收回，如果形成了闭包，执行完后其作用域就不会被收回。 如果某个函数被他的父函数之外的一个变量引用，就会形成闭包。 闭包的作用，就是保存自己私有的变量，通过提供的接口（方法）给外部使用，但外部 不能直接访问该变量。

#### 5. 什么是原型链？

Javascript 是面向对象的，每个实例对象都有一个proto\_属性，该属性指向它原 型对象，这个实例对象的构造函数有一个原型属性 prototype，与实例的proto属性指 向同一个对象。当一个对象在查找一个属性的时，自身没有就会根据proto\_ 向它的原型 进行查找，如果都没有，则向它的原型的原型继续查找，直到查到 Object.prototype.proto\_为null，这样也就形成了原型链。

#### 6. 有哪些方式继承？

1. 借用构造函数。也叫伪造对象或经典继承。 思路：在子类构造函数的内部调用超类型构造函数。可以通过使用 apply()和call()方法 在新创建的对象上执行构造函数。 缺点：方法都在构造函数中定义，函数的复用就无从谈起。在超类型的原型中定义的方 法，对子类而言也是不可见的，结果所有的类型都只能使用构造函数模式。

2. 组合继承。也叫伪经典继承。指的是将原型链和借用构造函数的技术组合到一起， 从而发挥二者之长。 思路：使用原型链实现对原型属性属性和方法的继承，通过借用构造函数来实现实例属 性的继承。 优点：既通过在原型上定义方法实现了函数复用，又能保证每一个实例都有它自己的数 组。 组合继承避免了原型链和借用构造函数的缺陷，融合了他们的优点，成为 JavaScript 中常用的继承模式。

3. 原型链继承。 思路：借助原型可以基于已有的对象创建对象，同时还不必因此创建自定义类型。 在object()函数内部，先创建一个临时的构造函数，然后将传入的对象作为这个构造函 数的原型，最后返回了这个临时类型的一个新实例。

4. 寄生式继承。 思路：创建一个仅用于封装继承过程的函数，该函数在内部以某种方式来增强对象，最 后再像真的是它做了所有的工作一样返回对象。缺点：使用寄生式继承来为对象添加函数，会由于不能做到函数复用二降低效率，这一 点和构造函数模式类似。

5. ）寄生组合式继承。是JavaScript 最常用的继承模式。 思路：通过借用构造函数来继承属性，通过原型链的混成形式来继承方法。 本质上，就是使用寄生式继承来继承超类型的原型，然后再将结果指定给子类型的原型。 开发人员普遍认为寄生组合式继承时引用类型最理想的继承范式。 extend（）方法才用了这样的方式。

#### 7. 字符串的常用方法有哪些？

charCodeAt 方法返回一个整数，代表指定位置字符的 Unicode 编码；

charAt方法返回指定索引位置处的字符。如果超出有效范围的索引值返回空字符串；

slice方法返回字符串的片段；

substring方法返回位于String 对象中指定位置的子字符串。

substr方法返回一个从指定位置开始的指定长度的子字符串。

indexOf方法返回 String 对象内第一次出现子字符串位置。如果没有找到子字符串， 则返回-1；

lastIndexOf方法返回 String对象中字符串最后出现的位置。如果没有匹配到子字符 串，则返回-1；

search方法返回与正则表达式查找内容匹配的第一个字符串的位置。

concat 方法返回字符串值，该值包含了两个或多个提供的字符串的连接；

split 将一个字符串分割为子字符串，然后将结果作为字符串数组返回；

#### 8. DOM节点的增删改查？

1. 创建节点、追加节点

createElement（标签名）创建一个元素节点（具体的一个元素）。

createTextNode（节点文本内容）创建一个文本节点。

createDocumentFragment() //创建一个 DOM 片段。

appendChild（节点）追加一个节点。

2. 插入节点

appendChild（节点）也是一种插入节点的方式，还可以添加已经存在的元素，会将其

元素从原来的位置移到新的位置。

insertBefore（a,b）是参照节点，意思是 a节点会插入 b节点的前面。

3. 删除、移除节点

removeChild(节点) 删除一个节点，用于移除删除一个参数（节点）。其返回的被移除

的节点，被移除的节点仍在文档中，只是文档中已没有其位置了。

4. 复制节点

cloneNode() 方法，用于复制节点， 接受一个布尔值参数， true 表示深复制（复制节点

及其所有子节点）， false 表示浅复制（复制节点本身，不复制子节点）。

5. 替换节点

replaceChild(插入的节点，被替换的节点) ，用于替换节点，接受两个参数，第一参数

是要插入的节点，第二个是要被替换的节点。返回的是被替换的节点。

6. 查找节点

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

#### 9. 什么是预解析？

在代码整体执行之前，先解析一部分。

预解析之后，代码才会从上往下依次整体执行，但是预解析执行过的代码不会 重复执行。

js预解析干了什么事：js 中预解析会把声明部分的代码预先执行。

声明相关的代码可以分为两部分：

1、 变量声明 通过 var关键字定义的变量。

2、函数声明 通过 function关键字声明的函数

预解析时如果遇到重复的变量声明，那么忽略。

预解析时如果遇到重复的函数声明，保留后面的函数。

预解析时如果遇到变量与函数重名的情况，保留函数。

#### 10. 什么是变量名提升？

使用 var关键字定义的变量，被称为变量声明；

函数声明提升的特点是，在函数声明的前面，可以调用这个函数。

#### 11. JS中的typeof关键字能返回哪些数据类型？

typeof一般判断基本数据类型。是一个操作符而不是函数，圆括号可有可无。

typeof 返回值有：string，number，boolean，undefined，object ，function，

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

基本数据类型中数字，字符串，布尔类型返回其对类型 undefined返回 undefined

九大内置构造函数及其他所有函数返回function；

其他所有复杂类型对象和null返回 object

#### 12. 简述创建函数的几种方式？

1. 函数声明

function sum1(num1,num2){

return num1+num2;

}

2. 函数表达式

var sum2 = function(num1,num2){

return num1+num2;

}

3. 函数对象方式

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

#### 13. 代码实现数组排序并去重

function fn(arr){

for(var i = 0; i < arr.length-1; i++){

for(var j = 0; j < arr.length-1-i; j++){

if(arr[j]<arr[j+1]){

var temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

for(var k = 0; k < arr.length; k++){

var c = arr[k];

for(var l = k+1; l < arr.length; l++){

if(arr[l] == c){

arr.splice(l, 1);

l--;

}

}

}

return arr

}

var arr = [1, 2, 5, 6, 8, 9, 10, 6, 5, 7, 4, 3, 5]

console.log(fn(arr))

#### 14.写出下面代码输出的结果

A. console.log( undefined || 1 ); --> 1

B. console.log( null || NaN ); --> NaN

C. console.log( 0 && 1 ); --> 0

D. console.log( 0 && 1 || 0 ); --> 0

#### 15. 下列代码将会输出什么？

var foo = 1;

function fn() {

console.log( foo ); --> undefined

var foo = 2;

console.log( foo ); --> 2

}

fn();

### 实际工作部分

#### 1. 什么是短路表达式？

短路表达式只是一种简写形式，也就是用 && 和 || 来赋值或者执行函数的形式

例如：

var foo = foo1 || foo2;

意思是如果foo1是真的，那么就把foo1的值赋给foo，否则把foo2的值赋给foo。

foo && foo()

当foo存在的时候，我们就执行foo函数，如果这个时候foo不是一个函数，就会报错，所以这个只是一种简写形式而已。

#### 2. 控制台中使用哪些部分调试？

主要用console来进行调试

1. console.log 用于输出普通信息

2. console.info 用于输出提示性信息

3. console.error用于输出错误信息

4. console.warn用于输出警示信息

5. console.debug用于输出调试信息

## 三、移动Web模块

### 基础部分

#### 1. 要你出一套适应不同分辨率，不同终端的前端实现方案你有什么思路？

1. 流式布局：

使用非固定像素来定义网页内容，也就是百分比布局，通过盒子的宽度设置成百分比来

根据屏幕的宽度来进行伸缩，不受固定像素的限制，内容向两侧填充。这样的布局方式，就

是移动web 开发使用的常用布局方式。这样的布局可以适配移动端不同的分辨率设备。

2. 响应式开发：

那么 EthanMarcotte 在2010 年 5月份提出的一个概念，简而言之，就是一个网站能

够兼容多个终端。越来越多的设计师也采用了这种设计。

CSS3中的 Media Query（媒介查询），通过查询 screen 的宽度来指定某个宽

度区间的网页布局。

超小屏幕（移动设备） 768px以下

小屏设备 768px-992px

中等屏幕 992px-1200px

宽屏设备 1200px 以上

由于响应式开发显得繁琐些，一般使用第三方响应式框架来完成，比如bootstrap 来

完成一部分工作，当然也可以自己写响应式。

#### 2. px em rem 取用选择依据？

1. px 像素（Pixel）。绝对单位。像素px是相对于显示器屏幕分辨率而言的，是一

个虚拟长度单位，是计算机系统的数字化图像长度单位，如果 px要换算成物理长度，需要

指定精度 DPI。

2. em是相对长度单位，相对于当前对象内文本的字体尺寸。如当前对行内文本的字

体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。它会继承父级元素的字体大小，因

此并不是一个固定的值。

3. rem是 CSS3新增的一个相对单位（root em，根em），使用 rem 为元素设定字

体大小时，仍然是相对大小，但相对的只是HTML根元素。

4. 区别：IE无法调整那些使用 px作为单位的字体大小，而em 和rem可以缩放，rem

相对的只是HTML根元素。这个单位可谓集相对大小和绝对大小的优点于一身，通过它既

可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁

反应。目前，除了 IE8及更早版本外，所有浏览器均已支持rem。

#### 3. Zepto和jQuery的区别？

Zepto相对jQuery更加轻量，主要用在移动端，jQuery也有对应的jQuerymobile移动端框架。

### 实际工作部分

#### 1. 移动端touch事件判断滑屏手势的方向？

当开始一个touchstart事件的时候，获取此刻手指的横坐标startX和纵坐标startY；

当触发touchmove事件时，在获取此时手指的横坐标moveEndX和纵坐标moveEndY;最后，通过这两次获取的坐标差值来判断手指在手机屏幕上的滑动方向。

思路：用touchmove的最后坐标减去touchstart的起始坐标，X的结果如果正数，则说明手指是从左往右划动；X的结果如果负数，则说明手指是从右往左划动；Y的结果如果正数，则说明手指是从上往下划动；Y的结果如果负数，则说明手指是从下往上划动。

具体代码如下：

var mybody = document.getElementsByTagName('body')[0];

//滑动处理

var startX, startY, moveEndX, moveEndY, X, Y;

mybody.addEventListener('touchstart', function(e) {

e.preventDefault();

startX = e.touches[0].pageX;

startY = e.touches[0].pageY;

});

mybody.addEventListener('touchmove', function(e) {

e.preventDefault();

moveEndX = e.changedTouches[0].pageX;

moveEndY = e.changedTouches[0].pageY;

X = moveEndX - startX;

Y = moveEndY - startY;

if ( X > 0 ) {alert(‘向右’);}

else if ( X < 0 ) {alert(‘向左’);}

else if ( Y > 0) {alert(‘向下’);}

else if ( Y < 0 ) { alert(‘向上’);}

else{alert(‘没滑动’); }

});

#### 2. 移动端对图片优化有哪些方式，怎么实现？

懒加载，使用CSS Sprites合并为一张大图，首先从图片格式方面着手，webp(google官方网址)是谷歌推出的一种图片格式，优点在于同等画面质量下，体积比jpg、png少了25%以上，去掉无意义的修饰，使用矢量图替代位图。

按照HTTP协议设置合理的缓存。

详见链接 http://web.jobbole.com/81766/

#### 3. rem布局中的尺寸是怎样计算的，实际举例说明一下？

其实rem布局的本质是等比缩放，一般是基于宽度，试想一下如果UE图能够等比缩放，假设我们将屏幕宽度平均分成100份，每一份的宽度用x表示，x = 屏幕宽度 / 100，如果将x作为单位，x前面的数值就代表屏幕宽度的百分比。

## 四、JavaScript高级模块

### 基础部分

#### 1. 说说你对this关键字的理解？

this 是一个关键字，它代表函数运行时，自动生成的一个内部对象，只能在函数内 部使用。

1.作为纯粹的函数调用 this 指向全局对象

2.作为对象的方法调用 this 指向调用对象

3.作为构造函数被调用 this 指向新的对象（new会改变 this 的指向）

4.apply 调用 this 指向 apply方法的第一个参数

#### 2. 表单验证传输的什么数据？明文还是暗文==加密？如何加密？是每一次传输数据，都是加密之后才传输吗？

1. 概述： GET是从服务器上请求数据，POST 是发送数据到服务器。事实上，GET方法是把数 据参数队列（query string）加到一个URL上，值和表单是一一对应的。比如说， name=John。在队列里，值和表单用一个&符号分开，空格用+号替换，特殊的符号转换 成十六进制的代码。因为这一队列在 URL里边，这样队列的参数就能看得到，可以被记录 下来，或更改。通常GET 方法还限制字符的大小（大概是 256 字节 ）。事实上POST 方法可以没有时间限制的传递数据到服务器，用户在浏览器端是看不到 这一过程的，所以 POST方法比较适合用于发送一个保密的（比如信用 卡号）或者比较大 量的数据到服务器。

2. 区别： Post 是允许传输大量数据的方法，而 Get 方法会将所要传输的数据附在网址后面，然 后一起送达服务器，因此传送的数据量就会受到限制，但是执行效率却比 Post 方法好。

3. 总结：

1、get 方式的安全性较Post 方式要差些，包含机密信息的话，建议用 Post 数据提交 方式；

2、在做数据查询时，建议用 Get 方式；而在做数据添加、修改或删除时，建议用Post 方式；

4. 所以： 表达如果是向服务器传输数据(如帐号密码等)都是加密数据(post)，如果只是单单想要 从服务器获得数据或者传输的数据并不重要， 可以直接使用明文方式传输( get )

#### 3. 如何实现跨域？

JSONP(JSON with Padding 填充式JSON 或参数式 JSON)

在js 中，我们虽然不能直接用XMLHttpRequest 请求不同域上的数据，但是在页面上引入不同域上的js 脚本文件却是可以的，jsonp正是利用这个特性来实现的。

JSONP 由两部分组成：回调函数和数据。回调函数是当响应到来时应该在页面中调用的函数，而数据就是传入回调函数中的JSON 数据。

优点： 它的兼容性更好，在更加古老的浏览器中都可以运行，不需要XMLHttpRequest 或 ActiveX 的支持； 能够直接访问响应文本，支持在浏览器与服务器之间双向通信

缺点： JSONP 是从其他域中加载代码执行。如果其他域不安全，很可能会在响应中夹带一些 恶意代码，而此时除了完全放弃 JSONP 调用之外，没有办法追究。因此在使用不是你自己 运维的Web 服务时，一定得保证它安全可靠。 它只支持 GET请求而不支持 POST等其它类型的 HTTP 请求；它只支持跨域 HTTP 请 求这种情况，不能解决不同域的两个页面之间如何进行 JavaScript调用的问题

#### 4. 说说事件委托机制？这样做有什么好处？

事件委托，就是某个事件本来该自己干的，但是自己不干，交给别人来干。就叫事件委 托。打个比方：一个button 对象，本来自己需要监控自身的点击事件，但是自己不来监控这个点击事件，让自己的父节点来监控自己的点击事件。

优点：

提高性能：列如，当有很多 li同时需要注册事件的时候，如果使用传统方法来注册 事件的话，需要给每一个 li 注册事件。然而如果使用委托事件的话，就只需要将事件委托给 该一个元素即可。这样就能提高性能。

新添加的元素还会有之前的事件；

#### 5. call和apply的区别？

它们的共同之处：都“可以用来代替另一个对象调用一个方法，将一个函数的对象上 下文从初始的上下文改变为由 thisObj 指定的新对象。”

它们的不同之处：

Apply：最多只能有两个参数——新this 对象和一个数组 argArray。如果给该方法传递多 个参数，则把参数都写进这个数组里面，当然，即使只有一个参数，也要写进数组里面。如 果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被 用作 thisObj， 并且无法被传递任何参数

Call：则是直接的参数列表，主要用在js 对象各方法互相调用的时候，使当前 this 实例指 针保持一致，或在特殊情况下需要改变this指针。如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。

#### 6. 在JS的计时器运行原理是怎样的，为什么可以触发计时效果? 计时器是多线程吗?

1. javascript引擎只有一个线程，强迫异步事件排队等待被执行。

2. setTimeout 和setInterval 本质上不同的地方是他们如何执行异步代码的。

3. 如果一个定时器正在执行的时候被阻塞了，那么它将会被推迟到下一个可能的执行点，这既是使得延迟时间有可能会超过声明定时器时设置的值。

4. Interval 如果有足够的时间来执行（大于制定的延迟），那么它将会无延迟的一个紧 接着一个执行。

5. 原理： 计时器通过设定一定的时间段（毫秒）来异步的执行一段代码。因为 Javascript 是一 个单线程语言，计时器提供了一种绕过这种语言限制来执行代码的能力。

6. 总结： 计时器是单线程的， 需要等待上一个执行完， 如果上一个没有执行完， 下一个需要 延迟执行， 直到上一个执行完。

#### 7. 什么是事件的冒泡和捕获？

事件冒泡：子元素事件的触发会影响父元素事件；

开关事件冒泡：

A，开启事件冒泡：element.addEventListener(eventName，handler，false)；

B，关闭事件冒泡：假设传统方式事件的返回值为 e，就可以通过 e.stopPropagation()来关闭事件冒泡；

事件捕获：父元素的事件会影响子元素的事件；

开启事件捕获：element.addEventListener(eventName，handler，true)

#### 8. 面向对象和类的区别？

简单的说类是对象的模版。

在 js 中没有类， 所以在js 中所谓的 类 就是构造函数， 对象就是由构造函数创建

出来的实例对象。面向对象就是使用面向对象的方式处理问题， 面向对象是对面向过程进

行封装。

面向对象有三大特性

抽象性， 需要通过核心数据和特定环境才能描述对象的具体意义

封装性， 封装就是将数据和功能组合到一起， 在js 中对象就是键值对的集合， 对象

将属性和方法封装起来， 方法将过程封装起来

继承性，将别人的属性和方法成为自己的，传统继承基于模板(类)，js 中继承基于 构

造函数

### 实际工作部分

#### 1. JavaScript 中的垃圾回收机制?

在Javascript 中，如果一个对象不再被引用，那么这个对象就会被GC 回收。如 果两个对象互相引用，而不再被第3者所引用，那么这两个互相引用的对象也会被回收。 因为函数 a被b引用，b又被 a外的 c引用，这就是为什么 函数 a 执行后不会被回收的原 因。

#### 2. 列出3条以上 FF 和 IE 的脚本兼容问题

1. window.event：

表示当前的事件对象，IE有这个对象，FF没有，FF通过给事件处理函数传递事件对象

2. 获取事件源

IE用srcElement获取事件源，而FF用target获取事件源

3. 添加，去除事件

IE：element.attachEvent(“onclick”,function) element.detachEvent(“onclick”, function)

FF：element.addEventListener(“click”,function, true) element.removeEventListener(“click”, function, true)

4. 获取标签的自定义属性

IE：div1.value或div1[“value”]

FF：可用div1.getAttribute(“value”)

#### 3. 用正则表达式，写出由字母开头，其余由数字、字母、下划线组成的6~30的字符串？

/^[a-zA-Z]{1}[\w]{5,29}$/

#### 4. 下列JavaScript代码执行后，iNum的值是多少？

- [ ] var iNum = 0;

for(var i = 1; i < 10; i++){

if(i% 5 == 0){

continue;

}

iNum++;

}

答：8

#### 5. 程序中捕获异常的方法？

window.error

try{}catch(error){}finally{}

#### 6. 正则匹配邮箱？

/^([a-zA-Z0-9-])+@([a-zA-Z0-9-])+((.[a-zA-Z0-9\_-]{2,3}){1,2})$/;

#### 7. Javascript中callee和caller的作用？

caller是返回一个对函数的引用，该函数调用了当前函数；

callee是返回正在被执行的function函数，也就是所指定的function对象的正文。

#### 8. 下列代码的输出结果？

function f1(){

var tmp = 1;

this.x = 3;

console.log( tmp );

console.log( this.x );

}

var obj = new f1();

console.log( obj.x );

console.log( f1() );

首先看代码（1），这里实例话化了f1这个类。相当于执行了 f1函数。所以这个时候 A 会输出 1， 而 B 这个时候的 this 代表的是实例化的当前对象 obj ,B 输出 3。

代码（2）毋庸置疑会输出 3，

重点 代码（3）首先这里将不再是一个类，它只是一个函数。那么 A输出 1， B呢？这里的this 代表的其实就是window对象，那么this.x 就是一个全局变量 相当于在外部 的一个全局变量。所以B 输出 3。最后代码由于f没有返回值那么一个函数如果没返回值的话，将会返回 underfined ，所以答案就是 ： 1， 3， 3， 1， 3， underfined 。

#### 9. 下面代码输出结果？

function changeObjectProperty(o){

o.siteUrl = "http://www.csser.com/";

o = new Object();

o.siteUrl = "http://www.popcg.com/";

}

var CSSer = new Object();

changeObjectProperty( CSSer );

console.log( CSSer.siteUrl );

答："http://www.popcg.com/"

## 五、jQuery模块

### 基础部分

#### 1. 谈谈你对jQuery的理解？

JQuery 是继 prototype 之后又一个优秀的 Javascript 库。它是轻量级的js 库 ，它

兼容 CSS3，还兼容各种浏览器（IE 6.0+，FF1.5+，Safari 2.0+，Opera 9.0+），jQuery2.0

及后续版本将不再支持 IE6/7/8 浏览器。jQuery 使用户能更方便地处理 HTML（标准通用

标记语言下的一个应用）、events、实现动画效果，并且方便地为网站提供 AJAX 交互。

jQuery还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得很详细，

同时还有许多成熟的插件可供选择。jQuery 能够使用户的 html页面保持代码和 html 内容

分离，也就是说，不用再在 html里面插入一堆 js 来调用命令了，只需要定义 id即可。

jQuery 是一个兼容多浏览器的 javascript 库，核心理念是write less，do more(写

得更少，做得更多)。 jQuery是免费、开源的，使用 MIT 许可协议。jQuery 的语法设

计可以使开发更加便捷，例如操作文档对象、选择 DOM 元素、制作动画效果、事件处理、

使用 Ajax 以及其他功能。除此以外，jQuery 提供 API让开发者编写插件。其模块化的使

用方式使开发者可以很轻松的开发出功能强大的静态或动态网页。

#### 2.原生JS的window.onload与Jquery的$(document).ready(function() {})，$(function () {})有什么不同？

1.执行时间 window.onload必须等到页面内包括图片的所有元素加载完毕后才能执行。 $(document).ready()是 DOM 结构绘制完毕后就执行，不必等到加载完毕。

2.编写个数不同 window.onload不能同时编写多个，如果有多个 window.onload 方法，只会执 行一个 $(document).ready()可以同时编写多个，并且都可以得到执行

3.简化写法 window.onload没有简化写法

$(document).ready(function (){})

可以简写成$(function(){})；

#### 3. jQuery一个对象可以同时绑定多个事件，是如何实现的？

jQuery可以给一个对象同时绑定多个事件，低层实现方式是使用addEventListner或attachEvent兼容不同的浏览器实现事件的绑定，这样可以给同一个对象注册多个事件。

#### 4. jQuery.fn的init方法返回的this指的是什么对象？为什么要返回this？

this执行init构造函数自身，其实就是jQuery实例对象，返回this是为了实现jQuery的链式操作

#### 5. jQuery.extend和jQuery.fn.extend有什么区别？

Jquery.extend用来扩展jQuery对象本身；jquery.fn.extend用来扩展jQuery实例

### 实际工作部分

1. jQuery框架中$.ajax()的常用参数有哪些？写一个post请求并带有发送数据和返回数据的样例？

async是否异步

url请求地址

contentType发送信息至服务器时内容编码类型

data发送到服务器的数据

dataType预期服务器返回的数据类型

type请求类型

success请求成功回调函数

error请求失败回调函数

$.ajax({

url: "/jquery/test1.txt",

type: 'post',

data: { id: 1 },

success: function ( data ) { alert(data); }

})

#### 2. 举一下jquery中的函数，这些函数实现链式编程的原理？

toggle（fn, fn）

$（“td”）.toggle（

function（）{

$（this）.addClass（“selected”）;

}，

function（）{

$（this）.removeClass（“selected”）;

）

实现函数链式编程的原理：返回自身，其他过程在函数内部实现，其好处是：节约js代码，返回的是同一个对象，提高代码的效率。

## 六、PHP后台开发模块

### 基础部分

#### 1. php inset 和 empty 的区别？

1、empty函数

用途：检测变量是否为空

判断：如果 var 是非空或非零的值，则 empty() 返回 FALSE。换句话说，""、0、"0"、NULL、FALSE、array()、var $var; 以及没有任何属性的对象都将被认为是空的，如果 var 为空，则返回 TRUE。注意：empty() 只检测变量，检测任何非变量的东西都将导致解析错误。换句话说，后边的语句将不会起作用;

2、isset函数

用途：检测变量是否设置

判断：检测变量是否设置，并且不是 NULL。如果已经使用 unset() 释放了一个变量之后，它将不再是 isset()。若使用 isset() 测试一个被设置成 NULL 的变量，将返回 FALSE。同时要注意的是一个NULL 字节（"\0"）并不等同于 PHP 的 NULL 常数。

## 七、Ajax模块

### 基础部分

#### 1. ajax是什么？

Ajax并不算是一种新的技术，全称是asychronous javascript and xml，可以说是已有技术的组合，主要用来实现客户端与服务器端的异步通信效果，实现页面的局部刷新，早期的浏览器并不能原生支持ajax，可以使用隐藏帧（iframe）方式变相实现异步效果，后来的浏览器提供了对ajax的原生支持

使用ajax原生方式发送请求主要通过XMLHttpRequest(标准浏览器)、ActiveXObject(IE浏览器)对象实现异步通信效果

#### 2. 同步和异步执行代码的区别？

同步：阻塞的

=张三叫李四去吃饭，李四一直忙得不停，张三一直等着，直到李四忙完两个人一块去吃饭

=浏览器向服务器请求数据，服务器比较忙，浏览器一直等着（页面白屏），直到服务器返回数据，浏览器才能显示页面

异步：非阻塞的

=张三叫李四去吃饭，李四在忙，张三说了一声然后自己就去吃饭了，李四忙完后自己去吃

=浏览器向服务器请求数据，服务器比较忙，浏览器可以自如的干原来的事情（显示页面），服务器返回数据的时候通知浏览器一声，浏览器把返回的数据再渲染到页面，局部更新

#### 3. 页面编码和被请求的资源编码不一样如何处理？

对于ajax请求传递的参数，如果是get请求方式，参数如果传递中文，在有些浏览器会乱码，不同的浏览器对参数编码的处理方式不同，所以对于get请求的参数需要使用 encodeURIComponent函数对参数进行编码处理，后台开发语言都有相应的解码api。对于post请求不需要进行编码

#### 4. 简述ajax的过程？

1. 创建XMLHttpRequest对象,也就是创建一个异步调用对象

2. 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

3. 设置响应HTTP请求状态变化的函数

4. 发送HTTP请求

5. 获取异步调用返回的数据

6. 使用JavaScript和DOM实现局部刷新

#### 5. 请解释一下JavaScript的同源策略？

同源策略在什么情况下会起作用呢？ 当web 页面使用多个<iframe>元素或者打开其他浏览器窗口的时候，这一策略就 会起作用。

同源策略的含义： 脚本只能读取和所属文档来源相同的窗口和文档的属性。 这里就涉及到了一个浏览器如何判断两者是否同源以及如何判断脚本来源的问题。

注意一点：脚本本身的来源并不作为判断是否同源的依据，而是将脚本所属文档的来源 作为判断依据。

1. 判断脚本来源 例如：文档 A中通过 script 的 src引用了一个外部脚本，这个脚本是 google 提供的，也是从google 的主机上加载到文档 A中的，那么这个脚本的所属文档是谁呢， 答案是文档A。

2. 判断是否同源 理解了脚本来源，接着理解怎么判断是否同源：如果两个文档在协议、主机以 及载入文档的 URL端口这三点中有一点不同，就认为他们不同源。

#### 6. get和post的区别？

GET：一般用于信息获取，使用URL传递参数，对所发送信息的数量也有限制，一般在2000个字符，有的浏览器是8000个字符

POST：一般用于修改服务器上的资源，对所发送的信息没有限制

在以下情况中，请使用 POST 请求：

1. 无法使用缓存文件（更新服务器上的文件或数据库）

2. 向服务器发送大量数据（POST 没有数据量限制）

3. 发送包含未知字符的用户输入时，POST 比GET 更稳定也更可靠

#### 7. 解释jsonp的原理？

Jsonp并不是一种数据格式，而json是一种数据格式，jsonp是用来解决跨域获取数据的一种解决方案，具体是通过动态创建script标签，然后通过标签的src属性获取js文件中的js脚本，该脚本的内容是一个函数调用，参数就是服务器返回的数据，为了处理这些返回的数据，需要事先在页面定义好回调函数，本质上使用的并不是ajax技术

#### 8. 如何实现跨域？

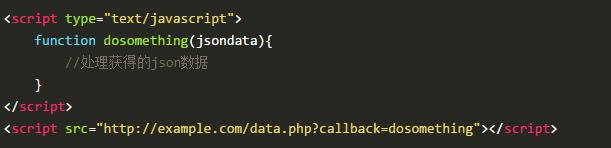
对于端口和协议的不同，只能通过后台来解决。我们要解决的是域名不同的问题。

1.下面是用 php 进行的设置，“\*”号表示允许任何域向我们的服务端提交请求、 header{"Access-Control-Allow-Origin: \*"}

2.JSONP(JSON with Padding 填充式 JSON 或参数式 JSON)

在 js 中，我们虽然不能直接用 XMLHttpRequest 请求不同域上的数据，但是在页面上 引入不同域上的 js 脚本文件却是可以的，jsonp 正是利用这个特性来实现的。

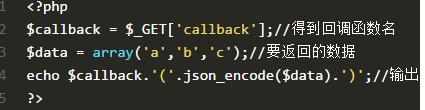
JSONP 由两部分组成:回调函数和数据。回调函数是当响应到来时应该在页面中调用 的函数，而数据就是传入回调函数中的 JSON 数据。



首先第一个 script 标签定义了一个处理数据的函数;

然后第二个 script 标签载入一个 js 文件，http://example.com/data.php 是数据所 在地址，但是因为是当做 js 来引入的，所以 http://example.com/data.php 返回的必须 是一个能执行的 js 文件;

最后 js 文件载入成功后会执行我们在 url 参数中指定的函数，并且会把我们需要的 json 数据作为参数传入。所以 php 应该是这样的:



#### 9. JSONP的优缺点

**优点:**

它的兼容性更好，在更加古老的浏览器中都可以运行，不需要 XMLHttpRequest 或 ActiveX 的支持;

能够直接访问响应文本，支持在浏览器与服务器之间双向通信

**缺点:**

JSONP 是从其他域中加载代码执行。如果其他域不安全，很可能会在响应中夹带一些 恶意代码，而此时除了完全放弃 JSONP 调用之外，没有办法追究。因此在使用不是你自己 运维的 Web 服务时，一定得保证它安全可靠。

它只支持 GET 请求而不支持 POST 等其它类型的 HTTP 请求;它只支持跨域 HTTP 请 求这种情况，不能解决不同域的两个页面之间如何进行 JavaScript 调用的问题

#### 10. ajax请求时如何解释json数据？

使用eval() 或者JSON.parse() 鉴于安全性考虑，推荐使用JSON.parse()更靠谱，对数据的安全性更好。

#### 11. HTTP状态码都有哪些？

100 => 正在初始化（一般是看不到的）

101 => 正在切换协议（websocket浏览器提供的）

200或者以2开头的两位数 => 都是代表响应主体的内容已经成功返回了

202 => 表示接受

301 => 永久重定向/永久转移

302 => 临时重定向/临时转移（一般用来做服务器负载均衡）

304 => 本次获取的内容是读取缓存中的数据，会每次去服务器校验

400 => 参数出现错误（客户端传递给服务器端的参数出现错误）

401 => 未认证，没有登录网站

403 => 禁止访问，没有权限

404 => 客户端访问的地址不存在

500 => 未知的服务器错误

503 => 服务器超负荷（假设一台服务器只能承受10000人，当第10001人访问的时候，如果服务器没有做负载均衡，那么这个人的网络状态码就是503）

### 实际工作部分

#### 1. 浏览器渲染页面的过程

浏览器会把HTML、SVG、XHTML三种格式的文件会产生一个DOM树；

Css，解析css会产生css规则树JavaScript会通过DOM apI 来操作DOM树 和 css规则树

#### 2. 简单描述下同步和异步的区别，并举例实际应用中哪些是同步，哪些是异步？

同步是阻塞的，浏览器向服务器发送请求，服务器比较忙，浏览器一直等着（页面白屏），直到服务器返回数据，页面才可以正常显示；

异步是非阻塞的，浏览器向服务器请求数据，服务器比较忙，浏览器可以干自己原来的事情（显示页面），服务器返回数据的时候通知浏览器一声，浏览器把返回的数据再渲染到页面，局部更新。

#### 3. 简述AJAX的工作原理

Ajax的原理简单来说通过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。这其中最关键的一步就是从服务器获得请求数据。

XMLHttpRequest是ajax的核心机制，它是在IE5中首先引入的，是一种支持异步请求的技术。简单的说，也就是javascript可以及时向服务器提出请求和处理响应，而不阻塞用户。达到无刷新的效果。

#### 4. jsonp是如何实现前后数据交互？

ajax请求受同源策略的影响，不允许进行跨域请求，而script标签的src属性中的链接却可以访问跨域的js脚本，利用这个特性，服务端不在返回json格式的数据，而是返回调用某个函数的js代码，在src中进行了调用，这样就实现了跨域，其原理就是动态创建script标签，通过script标签的src属性进行调用

#### 5. 你知道的HTTP 请求方式有几种？

**HTTPRequestMethod共计17种**

1. GET 请求指定的页面信息，并返回实体主体。

2. HEAD 类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头

3. POST 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。

4. PUT 从客户端向服务器传送的数据取代指定的文档的内容。

5. DELETE 请求服务器删除指定的页面。

6. CONNECT HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。

7. OPTIONS 允许客户端查看服务器的性能。

8. TRACE 回显服务器收到的请求，主要用于测试或诊断。

9. PATCH 实体中包含一个表，表中说明与该URI所表示的原内容的区别。

10. MOVE 请求服务器将指定的页面移至另一个网络地址。

11. COPY 请求服务器将指定的页面拷贝至另一个网络地址。

12. LINK 请求服务器建立链接关系。

13. UNLINK 断开链接关系。

14. WRAPPED 允许客户端发送经过封装的请求。

15. LOCK 允许用户锁定资源，比如可以再编辑某个资源时将其锁定，以防别人同时对其进行编辑。

16. MKCOL 允许用户创建资源

17. Extension-mothed 在不改动协议的前提下，可增加另外的方法。

## 八、H5C3模块

### 基础部分

#### 1. CSS3有哪些新特性？

1. CSS3实现圆角（border-radius），阴影（box-shadow），

2. 对文字加特效（text-shadow、），线性渐变（gradient），旋转（transform）

3. transform:rotate(9deg)scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);// 旋转,缩 放,定位,倾斜

4. 增加了更多的CSS选择器 多背景rgba

5. 在CSS3中唯一引入的伪元素是 ::selection.

6. 媒体查询，多栏布局

7. border-image

#### 2. Html5有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何区分 HTML和HTML5？

新特性：

1. 拖拽释放(Drag and drop) API

2. 语义化更好的内容标签（header,nav,footer,aside,article,section）

3. 音频、视频API(audio,video)

4. 画布(Canvas) API

5. 地理(Geolocation) API

6. 本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失；

7. sessionStorage 的数据在浏览器关闭后自动删除

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

9. 新的技术webworker, websocket,Geolocation

移除的元素：

1. 纯表现的元素：basefont，big，center，font, s，strike，tt，u；

2. 对可用性产生负面影响的元素：frame，frameset，noframes；

支持HTML5新标签：

3. IE8/IE7/IE6支持通过 document.createElement 方法产生的标签，可以利用这一特性让这些浏览器支持 HTML5 新标签，浏览器支持新标签后，还需要添加标签默认的样式（当然最好的方式是直接使用成熟的框架、使用最多的是html5shim框架）：

<!--[if lt IE 9]>

<script>src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>

<![endif]-->

#### 3. 本地存储（Local Storage）和cookies（储存在用户本地终端上的数据）之间的区别是什么？

Cookies:服务器和客户端都可以访问；大小只有4KB左右；有有效期，过期后将会删除；

本地存储：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；每个域5MB；没有过期数据，它将保留知道用户从浏览器清除或者使用Javascript代码移除

#### 4. 如何实现浏览器内多个标签页之间的通信?

调用 localstorge、cookies 等本地存储方式

#### 5. 你如何对网站的文件和资源进行优化？

文件合并

文件最小化/文件压缩

使用CDN托管

缓存的使用

#### 6. 什么是响应式设计？

它是关于网页制作的过程中让不同的设备有不同的尺寸和不同的功能。响应式设计是让所有的人能在这些设备上让网站运行正常

#### 7. 新的 HTML5文档类型和字符集是

HTML5文档类型：<!doctype html>

HTML5使用的编码<meta charset=”UTF-8”>

#### 8. HTML5 Canvas元素有什么用？

Canvas 元素用于在网页上绘制图形，该元素标签强大之处在于可以直接在HTML 上进行图形操作。

#### 9. HTML5 存储类型有什么区别？

Media API、Text Track API、Application Cache API、User Interaction、Data Transfer API、Command API、Constraint Validation API、History API

#### 10. CSS3新增伪类有那些？

p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。

p:last-of-type 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。

p:only-of-type 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。

p:only-child 选择属于其父元素的唯一子元素的每个 <p> 元素。

p:nth-child(2) 选择属于其父元素的第二个子元素的每个 <p> 元素。

:enabled、:disabled 控制表单控件的禁用状态。

:checked，单选框或复选框被选中。

#### 11. 你怎么来实现页面设计图，你认为前端应该如何高质量完成工作?一个满屏品 字布局 如何设计?

首先划分成头部、body、脚部；。。。。。

实现效果图是最基本的工作，精确到2px；

与设计师，产品经理的沟通和项目的参与

做好的页面结构，页面重构和用户体验

处理hack，兼容、写出优美的代码格式

针对服务器的优化、拥抱 HTML5。

#### 12. 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

#### 13. 请谈一下你对网页标准和标准制定机构重要性的理解。

（无标准答案）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。

### 实际工作部分

#### 1. 请描述一下cookies]()，sessionStorage和localStorage的区别？

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。

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。

#### 2. 如何在 HTML5页面中嵌入视频？

HTML5 定义了嵌入视频的标准方法，支持的格式包括：MP4、WebM 和 Ogg;

<video>

<source src="jamshed.mp4" type="video/mp4">

Your browser does'nt support video embedding feature.

</video>

#### 3. HTML5引入什么新的表单属性？

Datalist datetime output keygen date month week time number range emailurl

## 九、Node模块

### 基础部分

#### 1. 对Node的优点和缺点提出了自己的看法

（优点）因为Node是基于事件驱动和无阻塞的，所以非常适合处理并发请求，因此构建在Node上的代理服务器相比其他技术实现（如Ruby）的服务器表现要好得多。此外，与Node代理服务器交互的客户端代码是由javascript语言编写的，因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。

（缺点）Node是一个相对新的开源项目，所以不太稳定，它总是一直在变，

而且缺少足够多的第三方库支持。看起来，就像是Ruby/Rails当年的样子

#### 2. node.js 的应用场景

1. 实时应用：如在线聊天，实时通知推送等等（如socket.io）

2. 分布式应用：通过高效的并行I/O使用已有的数据

3. 工具类应用：海量的工具，小到前端压缩部署（如grunt），大到桌面图形界面应用程序

4. 游戏类应用：游戏领域对实时和并发有很高的要求（如网易的pomelo框架）

5. 利用稳定接口提升Web渲染能力

6. 前后端编程语言环境统一：前端开发人员可以非常快速地切入到服务器端的开发（如著名的纯Javascript全栈式MEAN架构）

#### 3. NodeJS非阻塞I/O模型执行流程

主线程：

1. 执行 node 的代码，把代码放入队列

2. 事件循环程序（主线程）把队列里面的同步代码都先执行了，

3. 同步代码执行完成，执行异步代码

4. 异步代码分 2种状况，

(1)、异步非 io setTimeout() setInterval()

判断是否可执行，如果可以执行就执行，不可以跳过。

(2)、异步io 文件操作

会从线程池当中去取一条线程，帮助主线程去执行。

5. 主线程会一直轮训，队列中没有代码了，主线程就会退出。

子线程：被放在线程池里面的线程，用来执行异步 io操作

1. 在线程池里休息

2. 异步 io 的操作来了，执行异步 io操作。

3. 子线程会把异步 io操作的 callback 函数，扔回给队列

4. 子线程会回到线程池了去休息。

callback，在异步 io 代码执行完成的时候被扔回主线程。

#### 4. nodejs中流(stream)的理解

1. 什么是Stream?

答案:stream是基于事件EventEmitter的数据管理模式．由各种不同的抽象接口组成，主要包括可写，可读，可读写，可转换等几种类型．

2. Stream有什么好处?

答案: 非阻塞式数据处理提升效率，片断处理节省内存，管道处理方便可扩展等.

3. Stream有哪些典型应用?

答案: 文件，网络，数据转换，音频视频等.

4. 怎么捕获 Stream的错误事件?

答案: 监听 error 事件，方法同 EventEmitter.

5. 有哪些常用Stream,分别什么时候使用?

答案:Readable 为可被读流，在作为输入数据源时使用；Writable 为可被写流,在作为

输出源时使用；Duplex 为可读写流,它作为输出源接受被写入，同时又作为输入源被后面的

流读出．Transform 机制和Duplex一样，都是双向流，区别时 Transfrom 只需要实现一

个函数transfrom(chunk, encoding, callback);而 Duplex 需要分别实现read(size)函数

和\_write(chunk, encoding, callback)函数.

6. 实现一个 Writable Stream?

答案: 三步走:1)构造函数call Writable 2) 继承 Writable 3) 实现\_write(chunk,encoding, callback)函数

#### 5. ES6有哪些新特性?

类的支持，模块化，箭头操作符，let/const块作用域，字符串模板，解构，参数默认值/不定参数/拓展参数, for-of遍历, generator, Map/Set, Promise

#### 6. 你对ES6的个人看法

从软件工程角度来看，以前真的很弱，不适合做大型应用，很容易导致烂尾工程。ES6就相当于当年的Java5,是历史性的发展，从此我们可以用js做大型项目了。事实上，各大主流浏览器现在已经支持大部分新特性了，后端的Node.js更是可以直接使用ES6的绝大多数语法。

#### 7. 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．

#### 8. 什么是“前端路由”?什么时候适合使用“前端路由”? “前端路由”有哪些优点和缺点?

1. 什么是前端路由

路由是根据不同的url地址展示不同的内容或页面。前端路由就是把不同路由对应不同的内容或页面的任务交给前端来做，之前通过服务端根据url的不同返回不同的页面实现的

2. 什么时候使用前端路由

在单页面应用，大部分页面结构不变，只改变部分内容的使用

3. 前端路由有什么优缺点

优点：

用户体验好，不需要每次都从服务器全部获取，快速展现给用户

缺点：

使用浏览器的前进，后退键的时候会重新发送请求，没有合理利用缓存

单页面无法记住之前滚动的位置，无法再前进，后退的时候记住滚动的

#### 9. 如何判断当前脚本运行在浏览器还是node环境中？

Exports = typeof window === 'undefined'? global: window ;

获取全局对象的方式

同理可得，typeof window可以用来判断是不是在浏览器环境中

### 实际工作部分

#### 1. node中的异步和同步怎么理解?

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

#### 2. 有哪些方法可以进行异步流程的控制?

1. 多层嵌套回调

2. 为每一个回调写单独的函数，函数里边再回调

3. 用第三方框架比方async, q, promise等

#### 3. npm有哪些常用的命令，请列举并说明作用

$npm init 项目初始化

$npminstall -g <name> 安装并更新package.json中的版本配置

$npm run<name> 执行一段脚本

$npm update -n <name> 更新模块

#### 4. 什么事Webpack？

WebPack 可以看做是模块打包机:它做的事情是，分析你的项目结构，找到 JavaScript 模块以及其它的一些浏览器不能直接运行的拓展语言(Scss，TypeScript 等)，并将其打 包为合适的格式以供浏览器使用。

#### 5. 为什么要使用Webpack

今的很多网页其实可以看做是功能丰富的应用，它们拥有着复杂的 JavaScript 代码和一大 堆依赖包。为了简化开发的复杂度，前端社区涌现出了很多好的实践方法 a:模块化，让我们可以把复杂的程序细化为小的文件; b:类似于 TypeScript 这种在 JavaScript 基础上拓展的开发语言:使我们能够实现目前版 本的 JavaScript 不能直接使用的特性，并且之后还能能装换为 JavaScript 文件使浏览器可 以识别; c:scss，less 等 CSS 预处理器 .........

这些改进确实大大的提高了我们的开发效率，但是利用它们开发的文件往往需要进行额外 的处理才能让浏览器识别,而手动处理又是非常反锁的，这就为 WebPack 类的工具的出现 提供了需求。

## 十、VUE模块

### 基础部分

#### 1. Vue组件间传值

1. 父子之间的传值

父组件向子组件传值通过prop子组件在props中创建一个属性，用以接收父组件传过来的值

子组件向父组件传值在响应该点击事件的函数中使用$emit来触发一个自定义事件在父组件中注册子组件并在子组件标签上绑定对自定义事件的监听

2. 非父子之间的通讯

可以通过eventBus来实现通信.

所谓eventBus就是创建一个事件中心，相当于中转站，可以用它来传递事件和接收事件

eventBus = new Vue();

组件1触发：

<div @click="eve"></div>

methods: {

eve() {

eventBus.$emit('change','hehe'); //Hub触发事件

}

}123456

组件2接收:

<div></div>

created() {

eventBus.$on('change', () => { //Hub接收事件

this.msg = 'hehe';

});

}123456

这样就实现了非父子组件之间的通信了.原理就是把eventBus当作一个中转站！

#### 2. VUE是什么?

vue.js（读音 /vjuː/，类似于 view） 是一套构建用户界面的渐进式框架与其他重量级框架不同的是，Vue 采用自底向上增量开发的设计。Vue 的核心库只关注视图层，它不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与单文件组件和 Vue 生态系统支持的库结合使用时，Vue 也完全能够为复杂的单页应用程序提供驱动。

#### 3. React 和 Vue 的区别React 和 Vue 的相似之处：

1. 使用 Virtual DOM

2. 提供了响应式（Reactive）和组件化（Composable）的视图组件。

3. 将注意力集中保持在核心库，而将其他功能如路由和全局状态管理交给相关的库。

Vue与Angular、React的对比：

1. vue.js 更轻量，gzip后只有20K+，angular:56K ,react:44K

2. vue.js 更易上手，学习曲线平稳

3. 吸收两家之长，有angular 的指令和 react组件化思想

优缺点：

优点：体积小。接口灵活。侵入性好，可用于页面的一部分，而不是整个页面。扩展性好。源码规范简洁。代码较为活跃，作者是中国人，可在官方论坛中文提问。github9000+。基于组件化的开发。它是一个轻量级 mvvm框架、数据驱动+组件化的前端开发、社区完善

缺点：社区不大，如果有问题可以读源码。功能仅限于view 层，Ajax等功能需要额外的库。对开发人员要求较高。开发的话，需要 webpack，不然很难用，最好配合es6。

#### 4. v-show 和 v-if的选择

v-if 也是惰性的：如果在初始渲染时条件为假，则什么也不做——在条件第一次变为真时才开始局部编译（编译会被缓存起来）。

相比之下，v-show 简单得多——元素始终被编译并保留，只是简单地基于 CSS 切换。

#### 5. Vue.js核心思想

Vue.js是一个提供MVVM数据双向绑定的库，专注于UI层面，核心思想是：数据驱动、组件系统。

数据驱动：

Vue.js数据观测原理在技术实现上，利用的是ES5Object.defineProperty和存储器属性: getter和setter（所以只兼容IE9及以上版本），可称为基于依赖收集的观测机制。核心是VM，即ViewModel，保证数据和视图的一致性。

组件式编程

#### 6. template参数选项的有无对生命周期的影响

1. 如果Vue实例对象中有template参数选项，则将其作为模板编译成render函数

2. 如果没有template参数选项，则将外部的HTML作为模板编译（template），也就是说，template参数选项的优先级要比外部的HTML高

3. 如果1,2条件都不具备，则报错

### 实际工作部分

#### 1. 怎么定义vue-router的动态路由？怎么获取传过来的动态参数？

对path属性加上/:id。 使用router对象的params.id

#### 2. vue-router有哪几种导航钩子

1. vue-router 提供的导航钩子主要用来拦截导航，让它完成跳转或取消

三种：

第一种是全局导航钩子：router.beforeEach(to,from,next)，作用：跳转前进行判断拦截。

第二种：组件内的钩子；

第三种：单独路由独享组件

#### 3. 说出至少4种vue当中的指令和它的用法？

v-if：判断是否隐藏；

v-for：数据循环出来；

v-bind:class：绑定一个属性；

v-model：实现双向绑定

#### 4.简述一些vue项目中文件构成？

build文件夹：主要就是webpack的配置；

Config文件夹：主要的就是index.js 这个文件进行配置代理服务器

Src文件夹：

“assets”共用的样式和图片

“components”业务代码存放

“router”路由

“APP.vue”vue 文件入口界面

“main.js”对应App.vue创建的实例，也是入口文件，对应 webpack.base.config.js里的入口配置

Static文件夹：静态资源

Pack.json:scripts 里面设置命令，例如设置了dev用于调试则我们开发时输入的是npm run dev ；例如设置了build 则是输入 npm run build 用于打包;另一部分是这里可以看到我们需要的依赖包,在dependencies和devDependencies中，分别对应全局下载和局部下载的依赖包

## 十一、Angular和React模块

### 基础部分

#### 1. angularJS是什么?

AngularJS 是一个 JavaScript 框架。它可通过 <script> 标签添加到 HTML 页面。

AngularJS 通过 指令 扩展了 HTML，且通过 表达式 绑定数据到 HTML

#### 2. angularJS中的MVVM模式

采用这种方式为合理组织代码提供了方便，降低了代码间的耦合度，功能结构清晰可见。

Model：一般用来处理数据，包括读取和设置数据，一般指的是操作数据库。

View：一般用来展示数据，就是放数据，比如通过 HTML来展示。

Controller：因为一个模块里面可能有多个视图和模型，控制器就起到了连接模型和视图的

作用。

VM--> ViewModel（视图模型） 也就是$scope。

MVC 模式在AngularJS 中的体现：

一、首先定义应用，采用ng-app属性指定一个应用，表示此标签内所包裹的内容都属于

APP的一部分。

<html ng-app="App" lang="en">

#### 3. 模块化与依赖注入

AngularJS 模块：

AngularJS模块是一种容器，把代码隔离并组织成简洁，整齐，可复用的块。

模块本身不提供直接的功能：包含其他提供功能的对象的实例：控制器，过滤器，服务，动

画。可通过定义其提供的对象构建模块。通过依赖注入将模块连接在一起，构建一个完整的

应用程序。

AngularJS建立在模块原理上。大部分 AngularJS 提供的功能都内置到ng-\*模块中。

创建模块：

你可以通过 AngularJS 的 angular.module 函数来创建模块：

<div ng-app="myApp">...</div>

<script>

var app = angular.module("myApp", []);

</script>

依赖注入：（Dependency Injection，简称 DI）是一种软件设计模式，在这种模式下，一个或更多的依赖（或服务）被注入（或者通过引用传递）到一个独立的对象（或客户端）中，然后成为了该客户端状态的一部分。

该模式分离了客户端依赖本身行为的创建，这使得程序设计变得松耦合，并遵循了依赖反转

和单一职责原则。与服务定位器模式形成直接对比的是，它允许客户端了解客户端如何使用

该系统找到依赖

一句话 ，没事你不要来找我，有事我会去找你。

AngularJS 提供很好的依赖注入机制。以下 5个核心组件用来作为依赖注入：value、

factory、service、provider、constant

#### 4. ReactJS的适用场景

一些后台界面，或者是和后台数据比较多，又或者和用户交互比较多，dom 操作频繁的都可以用 react。因为 react 的大特点就是虚拟DOM 技术，这样可以提高渲染的性能。个人的体验就是，用 react可以让整个页面的速度提高很多。

另外这样做的话，和后台的数据通过接口来进行前后端分离，也挺好挺方便的

#### 5. ReactJS是什么

React 是一个用于构建用户界面的 JAVASCRIPT 库。

React主要用于构建 UI，很多人认为 React 是 MVC 中的 V（视图）。

React 起源于 Facebook 的内部项目，用来架设 Instagram 的网站，并于 2013 年 5 月开源。

React 拥有较高的性能，代码逻辑非常简单，越来越多的人已开始关注和使用它。

#### 6. ReactJs 的特点

1. 声明式设计 −React 采用声明范式，可以轻松描述应用。

2. 高效 −React通过对 DOM 的模拟，最大限度地减少与 DOM 的交互。

3. 灵活 −React可以与已知的库或框架很好地配合。

4. JSX − JSX 是 JavaScript 语法的扩展。React 开发不一定使用 JSX ，但我们建议使用它。

5. 组件 − 通过 React 构建组件，使得代码更加容易得到复用，能够很好的应用在大项目的开发中。

6. 单向响应的数据流 − React 实现了单向响应的数据流，从而减少了重复代码，这也是它为什么比传统数据绑定更简单。

#### 7. react Native比起标准Web开发或原生开发能够带来的三大好处

1. 手势识别：基于Web 技术（HTML5/JavaScript）构建的移动应用经常被抱怨缺乏及时响应。而基于原生 UI的React Native能避免这些问题从而实现实时响应。

2. 原生组件：使用 HTML5/JavaScript 实现的组件比起原生组件总是让人感觉差一截，而React Native 由于采用了原生UI组件自然没有此问题。

3. 样式和布局：iOS、Android 和基于Web 的应用各自有不同的样式和布局机制。React Native 通过一个基于 FlexBox的布局引擎在所有移动平台上实现了一致的跨平台样式和布局方案。

### 实际工作部分

#### 1. 看过Angular的源码吗，它是怎么实现双向数据绑定的?

angular对常用的dom事件， xhr事件等做了封装，在里面触发进入angular的digest 流程。 在digest 流程里面， 会从 rootscope 开始遍历， 检查所有的watcher

#### 2. 为什么angular不推荐使用dom操作？

Angular 倡导以测试驱动开发，在的 service 或者 controller 中出现了DOM 操作，那 么也就意味着的测试是无法通过的 使用 Angular 的其中一个好处是啥，那就是双向数据绑定，这样就能专注于处理业务逻辑， 无需关系一堆堆的 DOM 操作。如果在 Angular 的代码中还到处充斥着各种 DOM 操作， 那为什么不直接使用jquery 去开发呢。

#### 3. MVC模式的意思是，软件可以分成三个部分？

视图（View）：用户界面。

控制器（Controller）：业务逻辑

模型（Model）：数据保存

#### 4. 各部分之间的通信方式是什么？

View 传送指令到 Controller

Controller 完成业务逻辑后，要求 Model 改变状态

Model 将新的数据发送到 View，用户得到反馈

## 十二、其他

### 实际工作部分

#### 1. webpack是怎样配置的，简要描述一下过程

第一在项目文件创建一个webpack.config.js文件，配置文件创建好了，第二开始正式配置webpack了，首先下载好node.js，因为webpack基于node.js,装好node.js后，通过命令行窗口找到项目文件，在项目文件webpack.config.js文件所在的目录下输入 npm install webpack -save dev 下载webpack依赖文件到本地项目中，下载好后会在webpack.config.js文件下自动创建node\_modules文件夹，文件夹里就是所有项目中用到的依赖插件，现在只有一个webpack，项目中用到在下载！

#### 2. git的作用是什么，有哪些命令，这些命令的作用是什么？在pull时，如果只想要文件的一部分而不是工程文件的整个，使用什么命令？git的冲突如何解决？

git是一个开源的分布式版本控制系统，用以有效、高速处理很小到非常大的项目版本管理。

常用的命令：

git init 初始化仓库

git config --globaluser.name 配置用户名

git config --globaluser.email 配置邮箱

git add 文件添加到暂存区

git commit 文件添加到仓库

git branch 列出所有的分支

git status 显示有变更的文件

git log 查看当前分支的版本历史

git push 提交本地代码到远程

git pull 从远程仓库下载到本地

git checkout 创建切换分支

解决git分支冲突： 将本地dev删除,在重新checkout一个dev分支(保证了此时我们的本地dev分支是最新的),在进行pull服务器分支,就这样解决了.

#### 3. 怎么防止内存泄露？

1. 减少不必要的全局变量，或者生命周期较长的对象，及时对无用的数据进行垃圾回收

2. 注意程序逻辑，避免“死循环”之类的

3. 避免创建过多的对象 。

#### 4. 网站优化时怎么做数据分析？

网站优化时，我们需要每天在百度统计中查看我们网站的每日流量，还有在各大站长平台中查看网站的收录量，网站关键词流量，还有就是每天要用站长工具查看网站的基本情况。这些数据都是最基础的。