**JavaScript面试题**

# 介绍js的基本数据类型。

Undefined、Null、Boolean、Number、String

# 介绍js有哪些内置对象？

Object 是 JavaScript 中所有对象的父对象

数据封装类对象：Object、Array、Boolean、Number 和 String

其他对象：Function、Arguments、Math、Date、RegExp、Error

# 说几条写JavaScript的基本规范？

1.不要在同一行声明多个变量。

2.请使用 ===/!==来比较true/false或者数值

3.使用对象字面量替代new Array这种形式

4.不要使用全局函数。

5.Switch语句必须带有default分支

6.函数不应该有时候有返回值，有时候没有返回值。

7.For循环必须使用大括号

8.If语句必须使用大括号

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

# JavaScript原型，原型链 ? 有什么特点？

每个对象都会在其内部初始化一个属性，就是prototype(原型)，当我们访问一个对象的属性时，

如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，

于是就这样一直找下去，也就是我们平时所说的原型链的概念。

关系：instance.constructor.prototype = instance.\_\_proto\_\_

特点：

JavaScript对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。当我们修改原型时，与之相关的对象也会继承这一改变。

当我们需要一个属性的时，Javascript引擎会先看当前对象中是否有这个属性， 如果没有的话，

就会查找他的Prototype对象是否有这个属性，如此递推下去，一直检索到 Object 内建对象。

function Func(){}

Func.prototype.name = "Sean";

Func.prototype.getInfo = function() {

return this.name;

}

var person = new Func();//现在可以参考var person = Object.create(oldObject);

console.log(person.getInfo());//它拥有了Func的属性和方法

//"Sean"

console.log(Func.prototype);

// Func { name="Sean", getInfo=function()}

# JavaScript有几种类型的值？，你能画一下他们的内存图吗？

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

堆：引用数据类型（对象、数组和函数）

两种类型的区别是：存储位置不同；

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

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

在栈中的地址，取得地址后从堆中获得实体

# Javascript如何实现继承？

1、构造继承

2、原型继承

3、实例继承

4、拷贝继承

原型prototype机制或apply和call方法去实现较简单，建议使用构造函数与原型混合方式。

function Parent(){

this.name = 'wang';

}

function Child(){

this.age = 28;

}

Child.prototype = new Parent();//继承了Parent，通过原型

var demo = new Child();

alert(demo.age);

alert(demo.name);//得到被继承的属性

}

# JavaScript继承的几种实现方式？

参考：[构造函数的继承](http://www.ruanyifeng.com/blog/2010/05/object-oriented_javascript_inheritance.html)，[非构造函数的继承](http://www.ruanyifeng.com/blog/2010/05/object-oriented_javascript_inheritance_continued.html)；

javascript创建对象的几种方式？

javascript创建对象简单的说,无非就是使用内置对象或各种自定义对象，当然还可以用JSON；但写法有很多种，也能混合使用。

1、对象字面量的方式

person={firstname:"Mark",lastname:"Yun",age:25,eyecolor:"black"};

2、用function来模拟无参的构造函数

function Person(){}

var person=new Person();//定义一个function，如果使用new"实例化",该function可以看作是一个Class

person.name="Mark";

person.age="25";

person.work=function(){

alert(person.name+" hello...");

}

person.work();

3、用function来模拟参构造函数来实现（用this关键字定义构造的上下文属性）

function Pet(name,age,hobby){

this.name=name;//this作用域：当前对象

this.age=age;

this.hobby=hobby;

this.eat=function(){

alert("我叫"+this.name+",我喜欢"+this.hobby+",是个程序员");

}

}

var maidou =new Pet("麦兜",25,"coding");//实例化、创建对象

maidou.eat();//调用eat方法

4、用工厂方式来创建（内置对象）

var wcDog =new Object();

wcDog.name="旺财";

wcDog.age=3;

wcDog.work=function(){

alert("我是"+wcDog.name+",汪汪汪......");

}

wcDog.work();

5、用原型方式来创建

function Dog(){

}

Dog.prototype.name="旺财";

Dog.prototype.eat=function(){

alert(this.name+"是个吃货");

}

var wangcai =new Dog();

wangcai.eat();

5、用混合方式来创建

function Car(name,price){

this.name=name;

this.price=price;

}

Car.prototype.sell=function(){

alert("我是"+this.name+"，我现在卖"+this.price+"万元");

}

var camry =new Car("凯美瑞",27);

camry.sell();

# Javascript作用链域?

全局函数无法查看局部函数的内部细节，但局部函数可以查看其上层的函数细节，直至全局细节。

当需要从局部函数查找某一属性或方法时，如果当前作用域没有找到，就会上溯到上层作用域查找，

直至全局函数，这种组织形式就是作用域链。

# 谈谈This对象的理解。

this总是指向函数的直接调用者（而非间接调用者）；

如果有new关键字，this指向new出来的那个对象；

在事件中，this指向触发这个事件的对象，特殊的是，IE中的attachEvent中的this总是指向全局对象Window；

# eval是做什么的？

它的功能是把对应的字符串解析成JS代码并运行；

应该避免使用eval，不安全，非常耗性能（2次，一次解析成js语句，一次执行）。

由JSON字符串转换为JSON对象的时候可以用eval，var obj =eval('('+ str +')');

# 什么是window对象? 什么是document对象?

# null，undefined 的区别？

null 表示一个对象被定义了，值为“空值”；

undefined 表示不存在这个值。

typeof undefined

//"undefined"

undefined :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 undefined；

例如变量被声明了，但没有赋值时，就等于undefined

typeof null

//"object"

null : 是一个对象(空对象, 没有任何属性和方法)；

例如作为函数的参数，表示该函数的参数不是对象；

注意：

在验证null时，一定要使用　=== ，因为 == 无法分别 null 和　undefined

再来一个例子：

null

Q：有张三这个人么？

A：有！

Q：张三有房子么？

A：没有！

undefined

Q：有张三这个人么？

A：没有！

参考阅读：[undefined与null的区别](http://www.ruanyifeng.com/blog/2014/03/undefined-vs-null.html)

# 写一个通用的事件侦听器函数。

// event(事件)工具集，来源：github.com/markyun

markyun.Event = {

// 页面加载完成后

readyEvent : function(fn) {

if (fn==null) {

fn=document;

}

var oldonload = window.onload;

if (typeof window.onload != 'function') {

window.onload = fn;

} else {

window.onload = function() {

oldonload();

fn();

};

}

},

// 视能力分别使用dom0||dom2||IE方式 来绑定事件

// 参数： 操作的元素,事件名称 ,事件处理程序

addEvent : function(element, type, handler) {

if (element.addEventListener) {

//事件类型、需要执行的函数、是否捕捉

element.addEventListener(type, handler, false);

} else if (element.attachEvent) {

element.attachEvent('on' + type, function() {

handler.call(element);

});

} else {

element['on' + type] = handler;

}

},

// 移除事件

removeEvent : function(element, type, handler) {

if (element.removeEventListener) {

element.removeEventListener(type, handler, false);

} else if (element.datachEvent) {

element.detachEvent('on' + type, handler);

} else {

element['on' + type] = null;

}

},

// 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)

stopPropagation : function(ev) {

if (ev.stopPropagation) {

ev.stopPropagation();

} else {

ev.cancelBubble = true;

}

},

// 取消事件的默认行为

preventDefault : function(event) {

if (event.preventDefault) {

event.preventDefault();

} else {

event.returnValue = false;

}

},

// 获取事件目标

getTarget : function(event) {

return event.target || event.srcElement;

},

// 获取event对象的引用，取到事件的所有信息，确保随时能使用event；

getEvent : function(e) {

var ev = e || window.event;

if (!ev) {

var c = this.getEvent.caller;

while (c) {

ev = c.arguments[0];

if (ev && Event == ev.constructor) {

break;

}

c = c.caller;

}

}

return ev;

}

};

# ["1", "2", "3"].map(parseInt) 答案是多少？

[1, NaN, NaN] 因为 parseInt 需要两个参数 (val, radix)，

其中 radix 表示解析时用的基数。

map 传了 3 个 (element, index, array)，对应的 radix 不合法导致解析失败。

# 事件是？IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。

2. 事件处理机制：IE是事件冒泡、Firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件；

3. ev.stopPropagation();（旧ie的方法 ev.cancelBubble = true;）

# 什么是闭包（closure），为什么要用它？

闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。

闭包的特性：

1.函数内再嵌套函数

2.内部函数可以引用外层的参数和变量

3.参数和变量不会被垃圾回收机制回收

//li节点的onclick事件都能正确的弹出当前被点击的li索引

<ul id="testUL">

<li> index = 0</li>

<li> index = 1</li>

<li> index = 2</li>

<li> index = 3</li>

</ul>

<script type="text/javascript">

var nodes = document.getElementsByTagName("li");

for(i = 0;i<nodes.length;i+= 1){

nodes[i].onclick = function(){

console.log(i+1);//不用闭包的话，值每次都是4

}(i);

}

</script>

执行say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在

使得Javascript的垃圾回收机制GC不会收回say667()所占用的资源

因为say667()的内部函数的执行需要依赖say667()中的变量

这是对闭包作用的非常直白的描述

function say667() {

// Local variable that ends up within closure

var num = 666;

var sayAlert = function() {

alert(num);

}

num++;

return sayAlert;

}

var sayAlert = say667();

sayAlert()//执行结果应该弹出的667

# javascript 代码中的"use strict";是什么意思 ? 使用它区别是什么？

use strict是一种ECMAscript 5 添加的（严格）运行模式,这种模式使得 Javascript 在更严格的条件下运行,

使JS编码更加规范化的模式,消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为。

默认支持的糟糕特性都会被禁用，比如不能用with，也不能在意外的情况下给全局变量赋值;

全局变量的显示声明,函数必须声明在顶层，不允许在非函数代码块内声明函数,arguments.callee也不允许使用；

消除代码运行的一些不安全之处，保证代码运行的安全,限制函数中的arguments修改，严格模式下的eval函数的行为和非严格模式的也不相同;

提高编译器效率，增加运行速度；

为未来新版本的Javascript标准化做铺垫。

# 如何判断一个对象是否属于某个类？

使用instanceof （待完善）

if(a instanceof Person){

alert('yes');

}

# new操作符具体干了什么呢?

1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。

2、属性和方法被加入到 this 引用的对象中。

3、新创建的对象由 this 所引用，并且最后隐式的返回 this 。

var obj = {};

obj.\_\_proto\_\_ = Base.prototype;

Base.call(obj);

# 用原生JavaScript的实现过什么功能吗？

# Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？

hasOwnProperty

javaScript中hasOwnProperty函数方法是返回一个布尔值，指出一个对象是否具有指定名称的属性。此方法无法检查该对象的原型链中是否具有该属性；该属性必须是对象本身的一个成员。

使用方法：

object.hasOwnProperty(proName)

其中参数object是必选项。一个对象的实例。

proName是必选项。一个属性名称的字符串值。

如果 object 具有指定名称的属性，那么JavaScript中hasOwnProperty函数方法返回 true，反之则返回 false。

# JSON 的了解？

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。

它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小

如：{"age":"12", "name":"back"}

JSON字符串转换为JSON对象:

var obj =eval('('+ str +')');

var obj = str.parseJSON();

var obj = JSON.parse(str);

JSON对象转换为JSON字符串：

var last=obj.toJSONString();

var last=JSON.stringify(obj);

[].forEach.call($$("\*"),function(a){a.style.outline="1px solid #"+(~~(Math.random()\*(1<<24))).toString(16)}) 能解释一下这段代码的意思吗？

# js延迟加载的方式有哪些？

defer和async、动态创建DOM方式（用得最多）、按需异步载入js

# Ajax 是什么? 如何创建一个Ajax？

ajax的全称：Asynchronous Javascript And XML。

异步传输+js+xml。

所谓异步，在这里简单地解释就是：向服务器发送请求的时候，我们不必等待结果，而是可以同时做其他的事情，等到有了结果它自己会根据设定进行后续操作，与此同时，页面是不会发生整页刷新的，提高了用户体验。

(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

# 同步和异步的区别?

同步的概念应该是来自于OS中关于同步的概念:不同进程为协同完成某项工作而在先后次序上调整(通过阻塞,唤醒等方式).同步强调的是顺序性.谁先谁后.异步则不存在这种顺序性.

同步：浏览器访问服务器请求，用户看得到页面刷新，重新发请求,等请求完，页面刷新，新内容出现，用户看到新内容,进行下一步操作。

异步：浏览器访问服务器请求，用户正常操作，浏览器后端进行请求。等请求完，页面不刷新，新内容也会出现，用户看到新内容。

（待完善）

# 如何解决跨域问题?

jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面

# 页面编码和被请求的资源编码如果不一致如何处理？

# 模块化开发怎么做？

[立即执行函数](http://benalman.com/news/2010/11/immediately-invoked-function-expression/),不暴露私有成员

var module1 = (function(){

　　　　var \_count = 0;

　　　　var m1 = function(){

　　　　　　//...

　　　　};

　　　　var m2 = function(){

　　　　　　//...

　　　　};

　　　　return {

　　　　　　m1 : m1,

　　　　　　m2 : m2

　　　　};

　　})();

（待完善）

AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

AMD 规范在这里：<https://github.com/amdjs/amdjs-api/wiki/AMD>

CMD 规范在这里：<https://github.com/seajs/seajs/issues/242>

Asynchronous Module Definition，异步模块定义，所有的模块将被异步加载，模块加载不影响后面语句运行。所有依赖某些模块的语句均放置在回调函数中。

区别：

1. 对于依赖的模块，AMD 是提前执行，CMD 是延迟执行。不过 RequireJS 从 2.0 开始，也改成可以延迟执行（根据写法不同，处理方式不同）。CMD 推崇 as lazy as possible.

2. CMD 推崇依赖就近，AMD 推崇依赖前置。看代码：

// CMD

define(function(require, exports, module) {

var a = require('./a')

a.doSomething()

// 此处略去 100 行

var b = require('./b') // 依赖可以就近书写

b.doSomething()

// ...

})

// AMD 默认推荐

define(['./a', './b'], function(a, b) { // 依赖必须一开始就写好

a.doSomething()

// 此处略去 100 行

b.doSomething()

// ...

})

# requireJS的核心原理是什么？（如何动态加载的？如何避免多次加载的？如何 缓存的？）

# 谈一谈你对ECMAScript6的了解？

# ECMAScript6 怎么写class么，为什么会出现class这种东西?

# 异步加载JS的方式有哪些？

(1) defer，只支持IE

(2) async：

(3) 创建script，插入到DOM中，加载完毕后callBack

# documen.write和 innerHTML的区别

document.write只能重绘整个页面

innerHTML可以重绘页面的一部分

# DOM操作——怎样添加、移除、移动、复制、创建和查找节点?

（1）创建新节点

createDocumentFragment() //创建一个DOM片段

createElement() //创建一个具体的元素

createTextNode() //创建一个文本节点

（2）添加、移除、替换、插入

appendChild()

removeChild()

replaceChild()

insertBefore() //在已有的子节点前插入一个新的子节点

（3）查找

getElementsByTagName() //通过标签名称

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

getElementById() //通过元素Id，唯一性

# .call() 和 .apply() 的区别？

例子中用 add 来替换 sub，add.call(sub,3,1) == add(3,1) ，所以运行结果为：alert(4);

注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

function add(a,b)

{

alert(a+b);

}

function sub(a,b)

{

alert(a-b);

}

add.call(sub,3,1);

# 数组和对象有哪些原生方法，列举一下？

# JS 怎么实现一个类。怎么实例化这个类

# JavaScript中的作用域与变量声明提升？

# 如何编写高性能的Javascript？

# 那些操作会造成内存泄漏？

# JQuery的源码看过吗？能不能简单概况一下它的实现原理？

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

# jquery中如何将数组转化为json字符串，然后再转化回来？

# jQuery 的属性拷贝(extend)的实现原理是什么，如何实现深拷贝？

# jquery.extend 与 jquery.fn.extend的区别？

# jQuery 的队列是如何实现的？队列可以用在哪些地方？

# 谈一下Jquery中的bind(),live(),delegate(),on()的区别？

# JQuery一个对象可以同时绑定多个事件，这是如何实现的？

# 是否知道自定义事件。jQuery里的fire函数是什么意思，什么时候用？

# jQuery 是通过哪个方法和 Sizzle 选择器结合的？（jQuery.fn.find()进入Sizzle）

# 针对 jQuery性能的优化方法？

# Jquery与jQuery UI 有啥区别？

\*jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。

\*jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。

提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等

# JQuery的源码看过吗？能不能简单说一下它的实现原理？

# jquery 中如何将数组转化为json字符串，然后再转化回来？

jQuery中没有提供这个功能，所以你需要先编写两个jQuery的扩展：

$.fn.stringifyArray = function(array) {

return JSON.stringify(array)

}

$.fn.parseArray = function(array) {

return JSON.parse(array)

}

然后调用：

$("").stringifyArray(array)

# jQuery和Zepto的区别？各自的使用场景？

# 针对 jQuery 的优化方法？

\*基于Class的选择性的性能相对于Id选择器开销很大，因为需遍历所有DOM元素。

\*频繁操作的DOM，先缓存起来再操作。用Jquery的链式调用更好。

比如：var str=$("a").attr("href");

\*for (var i = size; i < arr.length; i++) {}

for 循环每一次循环都查找了数组 (arr) 的.length 属性，在开始循环的时候设置一个变量来存储这个数字，可以让循环跑得更快：

for (var i = size, length = arr.length; i < length; i++) {}

# Zepto的点透问题如何解决？

# jQueryUI如何自定义组件?

# 需求：实现一个页面操作不会整页刷新的网站，并且能在浏览器前进、后退时正确响应。给出你的技术实现方案？

# 如何判断当前脚本运行在浏览器还是node环境中？（阿里）

通过判断Global对象是否为window，如果不为window，当前脚本没有运行在浏览器中

# 移动端最小触控区域是多大？

# jQuery 的 slideUp动画 ，如果目标元素是被外部事件驱动, 当鼠标快速地连续触发外部元素事件, 动画会滞后的反复执行，该如何处理呢?

# 把 Script 标签 放在页面的最底部的body封闭之前 和封闭之后有什么区别？浏览器会如何解析它们？

移动端的点击事件的有延迟，时间是多久，为什么会有？ 怎么解决这个延时？（click 有 300ms 延迟,为了实现safari的双击事件的设计，浏览器要知道你是不是要双击操作。）

# 知道各种JS框架(Angular, Backbone, Ember, React, Meteor, Knockout...)么? 能讲出他们各自的优点和缺点么?

Underscore 对哪些 JS 原生对象进行了扩展以及提供了哪些好用的函数方法？

# 解释JavaScript中的作用域与变量声明提升？

# 那些操作会造成内存泄漏？

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

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

setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。

闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）

# JQuery一个对象可以同时绑定多个事件，这是如何实现的？

# Node.js的适用场景？

(如果会用node)知道route, middleware, cluster, nodemon, pm2, server-side rendering么?

# 解释一下 Backbone 的 MVC 实现方式？

# 检测浏览器版本版本有哪些方式？

功能检测、userAgent特征检测

比如：navigator.userAgent

//"Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_2) AppleWebKit/537.36

(KHTML, like Gecko) Chrome/41.0.2272.101 Safari/537.36"

# 我们给一个dom同时绑定两个点击事件，一个用捕获，一个用冒泡。会执行几次事件，会先执行冒泡还是捕获？

# 原来公司工作流程是怎么样的，如何与其他人协作的？如何夸部门合作的？

# 你遇到过比较难的技术问题是？你是如何解决的？

# 设计模式 知道什么是singleton, factory, strategy, decrator么?

# 常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？

# 页面重构怎么操作？

网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。

也就是说是在不改变UI的情况下，对网站进行优化，在扩展的同时保持一致的UI。

对于传统的网站来说重构通常是：

表格(table)布局改为DIV+CSS

使网站前端兼容于现代浏览器(针对于不合规范的CSS、如对IE6有效的)

对于移动平台的优化

针对于SEO进行优化

深层次的网站重构应该考虑的方面

减少代码间的耦合

让代码保持弹性

严格按规范编写代码

设计可扩展的API

代替旧有的框架、语言(如VB)

增强用户体验

通常来说对于速度的优化也包含在重构中

压缩JS、CSS、image等前端资源(通常是由服务器来解决)

程序的性能优化(如数据读写)

采用CDN来加速资源加载

对于JS DOM的优化

HTTP服务器的文件缓存

# 列举IE与其他浏览器不一样的特性？

1、事件不同之处：

触发事件的元素被认为是目标（target）。而在 IE 中，目标包含在 event 对象的 srcElement 属性；

获取字符代码、如果按键代表一个字符（shift、ctrl、alt除外），IE 的 keyCode 会返回字符代码（Unicode），DOM 中按键的代码和字符是分离的，要获取字符代码，需要使用 charCode 属性；

阻止某个事件的默认行为，IE 中阻止某个事件的默认行为，必须将 returnValue 属性设置为 false，Mozilla 中，需要调用 preventDefault() 方法；

停止事件冒泡，IE 中阻止事件进一步冒泡，需要设置 cancelBubble 为 true，Mozzilla 中，需要调用 stopPropagation()；

# 什么叫优雅降级和渐进增强？

优雅降级：Web站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，则代码会针对旧版本的IE进行降级处理了,使之在旧式浏览器上以某种形式降级体验却不至于完全不能用。

如：border-shadow

渐进增强：从被所有浏览器支持的基本功能开始，逐步地添加那些只有新版本浏览器才支持的功能,向页面增加不影响基础浏览器的额外样式和功能的。当浏览器支持时，它们会自动地呈现出来并发挥作用。

如：默认使用flash上传，但如果浏览器支持 HTML5 的文件上传功能，则使用HTML5实现更好的体验；

# 是否了解公钥加密和私钥加密。

一般情况下是指私钥用于对数据进行签名，公钥用于对签名进行验证;

HTTP网站在浏览器端用公钥加密敏感数据，然后在服务器端再用私钥解密。

# WEB应用从服务器主动推送Data到客户端有那些方式？

html5提供的Websocket

不可见的iframe

WebSocket通过Flash

XHR长时间连接

XHR Multipart Streaming

<script>标签的长时间连接(可跨域)

# 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。

# 你有用过哪些前端性能优化的方法？

（1） 减少http请求次数：CSS Sprites, JS、CSS源码压缩、图片大小控制合适；网页Gzip，CDN托管，data缓存 ，图片服务器。

（2） 前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数

（3） 用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。

（4） 当需要设置的样式很多时设置className而不是直接操作style。

（5） 少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。

（6） 避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)。

（7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

（8） 避免在页面的主体布局中使用table，table要等其中的内容完全下载之后才会显示出来，显示比div+css布局慢。

对普通的网站有一个统一的思路，就是尽量向前端优化、减少数据库操作、减少磁盘IO。向前端优化指的是，在不影响功能和体验的情况下，能在浏览器执行的不要在服务端执行，能在缓存服务器上直接返回的不要到应用服务器，程序能直接取得的结果不要到外部取得，本机内能取得的数据不要到远程取，内存能取到的不要到磁盘取，缓存中有的不要去数据库查询。减少数据库操作指减少更新次数、缓存结果减少查询次数、将数据库执行的操作尽可能的让你的程序完成（例如join查询），减少磁盘IO指尽量不使用文件系统作为缓存、减少读写文件次数等。程序优化永远要优化慢的部分，换语言是无法“优化”的。

# http状态码有那些？分别代表是什么意思？

简单版

[

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

200 OK 正常返回信息

201 Created 请求成功并且服务器创建了新的资源

202 Accepted 服务器已接受请求，但尚未处理

301 Moved Permanently 请求的网页已永久移动到新位置。

302 Found 临时性重定向。

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

304 Not Modified 自从上次请求后，请求的网页未修改过。

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

401 Unauthorized 请求未授权。

403 Forbidden 禁止访问。

404 Not Found 找不到如何与 URI 相匹配的资源。

500 Internal Server Error 最常见的服务器端错误。

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

]

完整版

1\*\*(信息类)：表示接收到请求并且继续处理

100——客户必须继续发出请求

101——客户要求服务器根据请求转换HTTP协议版本

2\*\*(响应成功)：表示动作被成功接收、理解和接受

200——表明该请求被成功地完成，所请求的资源发送回客户端

201——提示知道新文件的URL

202——接受和处理、但处理未完成

203——返回信息不确定或不完整

204——请求收到，但返回信息为空

205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件

206——服务器已经完成了部分用户的GET请求

3\*\*(重定向类)：为了完成指定的动作，必须接受进一步处理

300——请求的资源可在多处得到

301——本网页被永久性转移到另一个URL

302——请求的网页被转移到一个新的地址，但客户访问仍继续通过原始URL地址，重定向，新的URL会在response中的Location中返回，浏览器将会使用新的URL发出新的Request。

303——建议客户访问其他URL或访问方式

304——自从上次请求后，请求的网页未修改过，服务器返回此响应时，不会返回网页内容，代表上次的文档已经被缓存了，还可以继续使用

305——请求的资源必须从服务器指定的地址得到

306——前一版本HTTP中使用的代码，现行版本中不再使用

307——申明请求的资源临时性删除

4\*\*(客户端错误类)：请求包含错误语法或不能正确执行

400——客户端请求有语法错误，不能被服务器所理解

401——请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用

HTTP 401.1 - 未授权：登录失败

　　HTTP 401.2 - 未授权：服务器配置问题导致登录失败

　　HTTP 401.3 - ACL 禁止访问资源

　　HTTP 401.4 - 未授权：授权被筛选器拒绝

HTTP 401.5 - 未授权：ISAPI 或 CGI 授权失败

402——保留有效ChargeTo头响应

403——禁止访问，服务器收到请求，但是拒绝提供服务

HTTP 403.1 禁止访问：禁止可执行访问

　　HTTP 403.2 - 禁止访问：禁止读访问

　　HTTP 403.3 - 禁止访问：禁止写访问

　　HTTP 403.4 - 禁止访问：要求 SSL

　　HTTP 403.5 - 禁止访问：要求 SSL 128

　　HTTP 403.6 - 禁止访问：IP 地址被拒绝

　　HTTP 403.7 - 禁止访问：要求客户证书

　　HTTP 403.8 - 禁止访问：禁止站点访问

　　HTTP 403.9 - 禁止访问：连接的用户过多

　　HTTP 403.10 - 禁止访问：配置无效

　　HTTP 403.11 - 禁止访问：密码更改

　　HTTP 403.12 - 禁止访问：映射器拒绝访问

　　HTTP 403.13 - 禁止访问：客户证书已被吊销

　　HTTP 403.15 - 禁止访问：客户访问许可过多

　　HTTP 403.16 - 禁止访问：客户证书不可信或者无效

HTTP 403.17 - 禁止访问：客户证书已经到期或者尚未生效

404——一个404错误表明可连接服务器，但服务器无法取得所请求的网页，请求资源不存在。eg：输入了错误的URL

405——用户在Request-Line字段定义的方法不允许

406——根据用户发送的Accept拖，请求资源不可访问

407——类似401，用户必须首先在代理服务器上得到授权

408——客户端没有在用户指定的饿时间内完成请求

409——对当前资源状态，请求不能完成

410——服务器上不再有此资源且无进一步的参考地址

411——服务器拒绝用户定义的Content-Length属性请求

412——一个或多个请求头字段在当前请求中错误

413——请求的资源大于服务器允许的大小

414——请求的资源URL长于服务器允许的长度

415——请求资源不支持请求项目格式

416——请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段

417——服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求长。

5\*\*(服务端错误类)：服务器不能正确执行一个正确的请求

HTTP 500 - 服务器遇到错误，无法完成请求

　　HTTP 500.100 - 内部服务器错误 - ASP 错误

　　HTTP 500-11 服务器关闭

　　HTTP 500-12 应用程序重新启动

　　HTTP 500-13 - 服务器太忙

　　HTTP 500-14 - 应用程序无效

　　HTTP 500-15 - 不允许请求 global.asa

　　Error 501 - 未实现

HTTP 502 - 网关错误

HTTP 503：由于超载或停机维护，服务器目前无法使用，一段时间后可能恢复正常

# 一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？（流程说的越详细越好）

注：这题胜在区分度高，知识点覆盖广，再不懂的人，也能答出几句，

而高手可以根据自己擅长的领域自由发挥，从URL规范、HTTP协议、DNS、CDN、数据库查询、

到浏览器流式解析、CSS规则构建、layout、paint、onload/domready、JS执行、JS API绑定等等；

详细版：

1、浏览器会开启一个线程来处理这个请求，对 URL 分析判断如果是 http 协议就按照 Web 方式来处理;

2、调用浏览器内核中的对应方法，比如 WebView 中的 loadUrl 方法;

3、通过DNS解析获取网址的IP地址，设置 UA 等信息发出第二个GET请求;

4、进行HTTP协议会话，客户端发送报头(请求报头);

5、进入到web服务器上的 Web Server，如 Apache、Tomcat、Node.JS 等服务器;

6、进入部署好的后端应用，如 PHP、Java、JavaScript、Python 等，找到对应的请求处理;

7、处理结束回馈报头，此处如果浏览器访问过，缓存上有对应资源，会与服务器最后修改时间对比，一致则返回304;

8、浏览器开始下载html文档(响应报头，状态码200)，同时使用缓存;

9、文档树建立，根据标记请求所需指定MIME类型的文件（比如css、js）,同时设置了cookie;

10、页面开始渲染DOM，JS根据DOM API操作DOM,执行事件绑定等，页面显示完成。

简洁版：

浏览器根据请求的URL交给DNS域名解析，找到真实IP，向服务器发起请求；

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

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

载入解析到的资源文件，渲染页面，完成。

# 从打开app到刷新出内容，整个过程中都发生了什么，如果感觉慢，怎么定位问题，怎么解决?

# 除了前端以外还了解什么其它技术么？你最最厉害的技能是什么？

# 你用的得心应手用的熟练地编辑器&开发环境是什么样子？

Sublime Text 3 + 相关插件编写前端代码

Google chrome 、Mozilla Firefox浏览器 +firebug 兼容测试和预览页面UI、动画效果和交互功能

Node.js+Gulp

git 用于版本控制和Code Review

# 对前端工程师这个职位是怎么样理解的？它的前景会怎么样？

前端是最贴近用户的程序员，比后端、数据库、产品经理、运营、安全都近。

1、实现界面交互

2、提升用户体验

3、有了Node.js，前端可以实现服务端的一些事情

前端是最贴近用户的程序员，前端的能力就是能让产品从 90分进化到 100 分，甚至更好，

参与项目，快速高质量完成实现效果图，精确到1px；

与团队成员，UI设计，产品经理的沟通；

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、拥抱最新前端技术。

# 你移动端前端开发的理解？（和 Web 前端开发的主要区别是什么？）

# 你对加班的看法？

加班就像借钱，原则应当是------救急不救穷

# 平时如何管理你的项目？

先期团队必须确定好全局样式（globe.css），编码模式(utf-8) 等；

编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；

标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；

页面进行标注（例如 页面 模块 开始和结束）；

CSS跟HTML 分文件夹并行存放，命名都得统一（例如style.css）；

JS 分文件夹存放 命名以该JS功能为准的英文翻译。

图片采用整合的 images.png png8 格式文件使用 尽量整合在一起使用方便将来的管理

# 如何设计突发大规模并发架构？

# 当团队人手不足，把功能代码写完已经需要加班的情况下，你会做前端代码的测试吗？

# 说说最近最流行的一些东西吧？常去哪些网站？

ES6\WebAssembly\Node\MVVM\Web Components\React\React Native\Webpack 组件化

# 知道什么是SEO并且怎么优化么? 知道各种meta data的含义么?

# 移动端（Android IOS）怎么做好用户体验?

清晰的视觉纵线、

信息的分组、极致的减法、

利用选择代替输入、

标签及文字的排布方式、

依靠明文确认密码、

合理的键盘利用、

# 简单描述一下你做过的移动APP项目研发流程？

# 你在现在的团队处于什么样的角色，起到了什么明显的作用？

# 你认为怎样才是全端工程师（Full Stack developer）？

# 介绍一个你最得意的作品吧？

# 你有自己的技术博客吗，用了哪些技术？

# 对前端安全有什么看法？

# 是否了解Web注入攻击，说下原理，最常见的两种攻击（XSS 和 CSRF）了解到什么程度？

# 项目中遇到国哪些印象深刻的技术难题，具体是什么问题，怎么解决？。

# 最近在学什么东西？

# 你的优点是什么？缺点是什么？

# 如何管理前端团队?

# 最近在学什么？能谈谈你未来3，5年给自己的规划吗？