Part 3 JavaScript

1.JavaScript是一门什么样的语言，它有哪些特点？

JavaScript一种直译式脚本语言（脚本语言：源代码在发往客户端运行之前不需经过编译，而是将文本格式的字符代码发送给浏览器由浏览器解释运行；直译：主流的引擎都是每次运行时加载代码并解译），是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。
三大部分组成：ECMAScript，描述了该语言的语法和基本对象；文档对象模型（DOM），描述处理网页内容的方法和接口；浏览器对象模型（BOM），描述与浏览器进行交互的方法和接口。
JavaScript脚本语言具有以下特点:
(1)脚本语言。JavaScript是一种解释型的脚本语言，C、C++等语言先编译后执行,而JavaScript是在程序的运行过程中逐行进行解释。
(2)基于对象。JavaScript是一种基于对象的脚本语言,它不仅可以创建对象，也能使用现有的对象。
(3)简单。JavaScript语言中采用的是弱类型的变量类型，对使用的数据类型未做出严格的要求，是基于Java基本语句和控制的脚本语言，其设计简单紧凑。
(4)动态性。JavaScript是一种采用事件驱动的脚本语言,它不需要经过Web服务器就可以对用户的输入做出响应。在访问一个网页时,鼠标在网页中进行鼠标点击或上下移、窗口移动等操作JavaScript都可直接对这些事件给出相应的响应。
(5)跨平台性。JavaScript脚本语言不依赖于操作系统，仅需要浏览器的支持。因此一个JavaScript脚本在编写后可以带到任意机器上使用，前提上机器上的浏览器支持JavaScript脚本语言，目前JavaScript已被大多数的浏览器所支持。

2.介绍JavaScript的数据类型。

基本数据类型:
Undefined、Null、Boolean、Number、String、ECMAScript 2015 新增:Symbol(创建后独一无二且不可变的数据类型);
引用数据类型：
Object (Array, Date, RegExp, Function)；
那么问题来了，如何判断某变量是否为数组数据类型？
方法一：判断其是否具有“数组性质”，如slice()方法。可自己给该变量定义slice方法，故有时会失效；
方法二：obj instanceof Array 在某些IE版本中不正确；
方法三：方法一二皆有漏洞，在ECMA Script5中定义了新方法Array.isArray()，保证其兼容性，最好的方法如下：

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

3.说说写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。
详细见：前端JavaScript规范

4.JavaScript原型，原型链？有什么特点？

每个JS对象一定对应一个原型对象，并从原型对象继承属性和方法，原型对象也是普通的对象，是对象一个自带隐式的 `__proto__` 属性。
在 JavaScript 中，每当定义一个对象（函数也是对象）时候，对象中都会包含一些预定义的属性。其中每个函数对象都有一个`prototype` 属性，这个属性指向函数的原型对象,它包含了对象实例共享的方法和属性。
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，就构成了实例与原型之间的链条，也就是我们平时所说的原型链的概念。
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。

5.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。

6.Javascript如何实现继承？

原型链继承
借用构造函数继承
组合继承(原型+借用构造)
原型式继承
寄生式继承
寄生组合式继承

7.Javascript创建对象的几种方式？

对象字面量
工厂模式
构造函数模式
原型模式
混合构造函数和原型模式
动态原型模式
寄生构造函数模式
稳妥构造函数模式

8.Javascript作用链域?

当代码在一个环境中执行时, 会创建变量对象的一个作用域链(scope chain)来保证对执行环境有权访问的变量和函数的有序访问. 作用域第一个对象始终是当前执行代码所在的环境变量对象(VO).
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。

9.谈谈this对象的理解。

this是Javascript语言的一个关键字。
它代表函数运行时，自动生成的一个内部对象，而且是所有函数运行时的一个隐藏参数，指向函数的运行环境。
this的使用可以分成以下几个场合。
（1）全局环境
在全局环境使用this，它指的就是顶层对象window。
（2）构造函数
构造函数中的this，指的是实例对象。
（3）对象的方法
函数还可以作为某个对象的方法调用，这时this就指这个对象。
（4）Node
在Node中，this的指向又分成两种情况。全局环境中，this指向全局对象global；模块环境中，this指向module.exports。

10.eval是做什么的？

eval()函数会将传入的字符串当做JavaScript代码进行执行。
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 +')');

11.什么是window对象? 什么是document对象?

window 对象表示浏览器中打开的窗口，它也是所有对象的顶层对象.
如果文档包含框架（frame 或 iframe 标签），浏览器会为 HTML 文档创建一个 window 对象，并为每个框架创建一个额外的 window 对象。
在浏览器中，window对象又具有双重角色，它既是通过js访问浏览器窗口的一个接口，又是一个Global（全局）对象。这意味着在网页中定义的任何对象，变量和函数，都以window作为其global对象。

每个载入浏览器的 HTML 文档都会成为 document 对象。
document 对象是HTML文档的根节点与所有其他节点（元素节点，文本节点，属性节点, 注释节点）。
document对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问。
提示：document 对象是 window 对象的一部分，可通过 window.document 属性对其进行访问。

12.null，undefined的区别？

相似性
在JavaScript中，将一个变量赋值为undefined或null，老实说，几乎没区别。undefined和null在if语句中，都会被自动转为false，相等运算符甚至直接报告两者相等。
历史原因
1995年JavaScript诞生时，最初像Java一样，只设置了null作为表示"无"的值。根据C语言的传统，null被设计成可以自动转为0。但是，JavaScript的设计者Brendan Eich，觉得这样做还不够，有两个原因。
首先，null像在Java里一样，被当成一个对象。但是，JavaScript的数据类型分成原始类型（primitive）和合成类型

（complex）两大类，Brendan Eich觉得表示"无"的值最好不是对象。

其次，JavaScript的最初版本没有包括错误处理机制，发生数据类型不匹配时，往往是自动转换类型或者默默地失败。Brendan Eich觉得，如果null自动转为0，很不容易发现错误。因此，Brendan Eich又设计了一个undefined。

最初设计

JavaScript的最初版本是这样区分的：null是一个表示"无"的对象，转为数值时为0；undefined是一个表示"无"的原始值，转为数值时为NaN。

目前的用法

但是，上面这样的区分，在实践中很快就被证明不可行。目前，null和undefined基本是同义的，只有一些细微的差别。

null表示"没有对象"，即该处不应该有值。典型用法是：

（1）作为函数的参数，表示该函数的参数不是对象。

（2）作为对象原型链的终点。

`Object.getPrototypeOf(Object.prototype); //null`

undefined表示"缺少值"，就是此处应该有一个值，但是还没有定义。典型用法是：

（1）变量被声明了，但没有赋值时，就等于undefined。

（2）调用函数时，应该提供的参数没有提供，该参数等于undefined。

（3）对象没有赋值的属性，该属性的值为undefined。

（4）函数没有返回值时，默认返回undefined。

```
var i;
i // undefined
function f(x){console.log(x)}
f() // undefined
var  o = new Object();
o.p // undefined
var x = f();
x // undefined
```

13.写一个通用的事件侦听器函数(机试题)。

```
//摘自《JavaScript高级程序设计》第三版源码,是书上的增强版;
var EventUtil = {

    addHandler: function(element, type, handler) {
        if (element.addEventListener) {
            element.addEventListener(type, handler, false);
        } else if (element.attachEvent) {
            element.attachEvent("on" + type, handler);
        } else {
            element["on" + type] = handler;
        }
    },

    getButton: function(event) {
        if (document.implementation.hasFeature("MouseEvents", "2.0")){
            return event.button;
        } else {
            switch(event.button){
                case 0:
                case 1:
                case 3:
                case 5:
                case 7:
```

```
                return 0;
            case 2:
            case 6:
                return 2;
            case 4: return 1;
        }
    }
},

getCharCode: function(event) {
    if (typeof event.charCode == "number") {
        return event.charCode;
    } else {
        return event.keyCode;
    }
},

getClipboardText: function(event) {
    var clipboardData =  (event.clipboardData || window.clipboardData);
    return clipboardData.getData("text");
},

getEvent: function(event) {
    return event ? event : window.event;
},

getRelatedTarget: function(event) {
    if (event.relatedTarget) {
        return event.relatedTarget;
    } else if (event.toElement) {
        return event.toElement;
    } else if (event.fromElement) {
        return event.fromElement;
    } else {
        return null;
    }
},

getTarget: function(event) {
    return event.target || event.srcElement;
},

getWheelDelta: function(event) {
    if (event.wheelDelta) {
        return (client.engine.opera && client.engine.opera < 9.5 ? -event.wheelDelta : event.w
    } else {
        return -event.detail * 40;
    }
},

preventDefault: function(event) {
    if (event.preventDefault) {
        event.preventDefault();
    } else {
        event.returnValue = false;
    }
},
```

```
    removeHandler: function(element, type, handler) {
        if (element.removeEventListener) {
            element.removeEventListener(type, handler, false);
        } else if (element.detachEvent) {
            element.detachEvent("on" + type, handler);
        } else {
            element["on" + type] = null;
        }
    },

    setClipboardText: function(event, value) {
        if (event.clipboardData) {
            event.clipboardData.setData("text/plain", value);
        } else if (window.clipboardData) {
            window.clipboardData.setData("text", value);
        }
    },

    stopPropagation: function(event) {
        if (event.stopPropagation) {
            event.stopPropagation();
        } else {
            event.cancelBubble = true;
        }
    }

};
```

14.["1", "2", "3"].map(parseInt) 答案是多少？

[1, NaN, NaN]
详细解析

15.关于事件，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。[文档或浏览器窗口中发生的一些特定的交互瞬间。]
(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;）

16.什么是闭包（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。
闭包的最大用处有两个，一个是可以读取函数内部的变量，另一个就是让这些变量始终保持在内存中，即闭包可以使得它诞生环境一直存在。闭包的另一个用处，是封装对象的私有属性和私有方法。

17.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，其主要目的有以下几种：
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。

18.如何判断一个对象是否属于某个类？

JavaScript中判断一个对象是否为一个类的实例主要有两种方法，即instanceof和constructor。
前者的用法是：`object instanceof constructor`；
后者的用法是：`var o = new Object // 或者 o = {} ; o.constructor == Object`；
区别：constructor 更加精确地指向对象所属的类，而对 instanceof 而言，即使是父类也会返回true。

```
var a = [1, 2, 3];
alert(a instanceof Array);  //返回true
alert(a instanceof Object);  //返回true

alert(a.constructor == Array);  //返回true
alert(a.constructor == Object);  //返回false
```

19.new操作符具体干了什么呢?

new命令的作用，就是执行构造函数，返回一个实例对象。
使用new命令时，它后面的函数调用就不是正常的调用，而是依次执行下面的步骤。

- 创建一个空对象，作为将要返回的对象实例；
- 将这个空对象的原型，指向构造函数的prototype属性；
- 将这个空对象赋值给函数内部的this关键字；
- 开始执行构造函数内部的代码。

20.用原生JavaScript的实现过什么功能吗？

21.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()`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)`
如果 object 具有指定名称的属性，那么JavaScript中hasOwnProperty()方法返回 true，反之则返回 false。

22.对JSON的了解？

JSON 指的是 JavaScript 对象表示法（JavaScript Object Notation）。
JSON 是存储和交换文本信息的语法，类似 XML，JSON 比 XML 更小、更快，更易解析。
JSON 是轻量级的文本数据交换格式。
JSON 独立于语言 *。
JSON 具有自我描述性，更易理解。
* JSON 使用 JavaScript 语法来描述数据对象，但是 JSON 仍然独立于语言和平台。JSON 解析器和 JSON 库支持许多不同的编程语言。

优点：
轻量级、易于人的阅读和编写，便于机器（JavaScript）解析，支持复合数据类型（数组、对象、字符串、数字）。
缺点：
和许多好东西都具有两面性一样，JSON的非冗长性也不例外，为此JSON丢失了XML具有的一些特性。命名空间允许不同上下文中的相同的信息段彼此混合，然而，显然在JSON中已经找不到了命名空间。JSON与XML的另一个差别是属性的差异，由于JSON采用冒号赋值，这将导致当XML转化为 JSON时，在标识符（XML CDATA）与实际属性值之间很难区分谁应该被当作文本考虑。
另外，JSON片段的创建和验证过程比一般的XML稍显复杂。从这一点来看，XML在开发工具方面领先于JSON。

23.能解释一下这段代码的意思吗？

```
[].forEach.call($$("*"),function(a){ a.style.outline="1px solid #"+(~~(Math.random()*(1<<24))).toString(16) })
```

24.js延迟加载的方式有哪些？

（1）直接将script节点放置在之前，这样js脚本就会在页面显示出来之后再加载。
（2）使用script标签的defer和async属性，defer属性为延迟加载，是在页面渲染完成之后再进行加载的，而async属性则是和文档并行加载，这两种解决方案都不完美，原因在于不是所有浏览器都支持。 直接插入代码、将脚本放置在底部和使用“defer”或“async”，这几种方法都不能达到先加载页面后加载JS的目的，而且它们肯定不能在各个浏览器上表现一致。
（3）Google帮助页面的推荐方案：
下面是Google推荐的代码。这些代码应被放置在标签前(接近HTML文件底部)。另外，我将外部JS文件名突出显示。

```
function downloadJSAtOnload() {
   var element = document.createElement("script");
   element.src = "defer.js";
   document.body.appendChild(element);
}
if (window.addEventListener)
   window.addEventListener("load",downloadJSAtOnload, false);
else if (window.attachEvent)
   window.attachEvent("onload",downloadJSAtOnload);
else window.onload =downloadJSAtOnload;
```

说明：
1）复制上面代码、粘贴到HTML的标签前(靠近HTML文件底部)，修改“defer.js”为你的外部JS文件。
2）不应该把那些页面正常加载需要依赖的javascript代码放在这里。
3）将JavaScript代码分成两组：一组是因页面需要而立即加载的javascript代码，另外一组是在页面加载后进行操作的javascript代码(例如添加click事件或其他东西)。这些需等到页面加载后再执行的JavaScript代码，应放在一个外部文件，然后再引进来。
（4）通过ajax下载js脚本，动态添加script节点。但是ajax有一个缺点，就是无法引用使用CDN方式提供的js文件，因为这种方式下，你即时通过xhr.open下载了js文件，而向body中注入script节点时还是需要向CDN请求js文件。

25.Ajax 是什么? 如何创建一个Ajax？

AJAX = 异步 JavaScript 和 XML。
AJAX通过原生的XMLHttpRequest对象发出HTTP请求，得到服务器返回的数据后，再进行处理。
AJAX 是一种用于创建快速动态网页的技术。
通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的

情况下，对网页的某部分进行更新。
传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。

```
(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
```

AJAX的优缺点

(1) AJAX的优点

<1> 无刷新更新数据。

AJAX最大优点就是能在不刷新整个页面的前提下与服务器通信维护数据。这使得Web应用程序更为迅捷地响应用户交互，并避免了在网络上发送那些没有改变的信息，减少用户等待时间，带来非常好的用户体验。

<2> 异步与服务器通信。

AJAX使用异步方式与服务器通信，不需要打断用户的操作，具有更加迅速的响应能力。优化了Browser和Server之间的沟通，减少不必要的数据传输、时间及降低网络上数据流量。

<3> 前端和后端负载平衡。

AJAX可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。并且减轻服务器的负担，AJAX的原则是“按需取数据”，可以最大程度的减少冗余请求和响应对服务器造成的负担，提升站点性能。

<4> 基于标准被广泛支持。

AJAX基于标准化的并被广泛支持的技术，不需要下载浏览器插件或者小程序，但需要客户允许JavaScript在浏览器上执行。随着Ajax的成熟，一些简化Ajax使用方法的程序库也相继问世。同样，也出现了另一种辅助程序设计的技术，为那些不支持JavaScript的用户提供替代功能。

<5> 界面与应用分离。

Ajax使WEB中的界面与应用分离（也可以说是数据与呈现分离），有利于分工合作、减少非技术人员对页面的修改造成的WEB应用程序错误、提高效率、也更加适用于现在的发布系统。

(2) AJAX的缺点

<1> AJAX干掉了Back和History功能，即对浏览器机制的破坏。

在动态更新页面的情况下，用户无法回到前一个页面状态，因为浏览器仅能记忆历史记录中的静态页面。一个被完整读入的页面与一个已经被动态修改过的页面之间的差别非常微妙；用户通常会希望单击后退按钮能够取消他们的前一次操作，但是在Ajax应用程序中，这将无法实现。后退按钮是一个标准的web站点的重要功能，但是它没法和js进行很好的合作。这是Ajax所带来的一个比较严重的问题，因为用户往往是希望能够通过后退来取消前一次操作的。那么对于这个问题有没有办法？答案是肯定的，用过Gmail的知道，Gmail下面采用的Ajax技术解决了这个问题，在Gmail下面是可以后退的，但是，它也并不能改变Ajax的机制，它只是采用的一个比较笨但是有效的办法，即用户单击后退按钮访问历史记录时，通过创建或使用一个隐藏的IFRAME来重现页面上的变更。（例如，当用户在Google Maps中单击后退时，它在一个隐藏的IFRAME中进行搜索，然后将搜索结果反映到Ajax元素上，以便将应用程序状态恢复到当时的状态。）但是，虽然说这个问题是可以解决的，但是它所带来的开发成本是非常高的，并与Ajax框架所要求的快速开发是相背离的。这是Ajax所带来的一个非常严重的问题。一个相关的观点认为，使用动态页面更新使得用户难于将某个特定的状态保存到收藏夹中。该问题的解决方案也已出现，大部分都使用URL片断标识符（通常被称为锚点，即URL中#后面的部分）来保持跟踪，允许用户回到指定的某个应用程序状态。（许多浏览器允许JavaScript动态更新锚点，这使得Ajax应用程序能够在更新显示内容的同时更新锚点。）这些解决方案也同时解决了许多关于不支持后退按钮的争论。

<2> AJAX的安全问题。

AJAX技术给用户带来很好的用户体验的同时也对IT企业带来了新的安全威胁，Ajax技术就如同对企业数据建立了一个直接通道。这使得开发者在不经意间会暴露比以前更多的数据和服务器逻辑。Ajax的逻辑可以对客户端的安全扫描技术隐藏起来，允许黑客从远端服务器上建立新的攻击。还有Ajax也难以避免一些已知的安全弱点，诸如跨站点

脚步攻击、SQL注入攻击和基于Credentials的安全漏洞等等。

<3> 对搜索引擎支持较弱。

对搜索引擎的支持比较弱。如果使用不当，AJAX会增大网络数据的流量，从而降低整个系统的性能。

<4> 破坏程序的异常处理机制。

至少从目前看来，像Ajax.dll，Ajaxpro.dll这些Ajax框架是会破坏程序的异常机制的。关于这个问题，曾在开发过程中遇到过，但是查了一下网上几乎没有相关的介绍。后来做了一次试验，分别采用Ajax和传统的form提交的模式来删除一条数据……给我们的调试带来了很大的困难。

<5> 违背URL和资源定位的初衷。

例如，我给你一个URL地址，如果采用了Ajax技术，也许你在该URL地址下面看到的和我在这个URL地址下看到的内容是不同的。这个和资源定位的初衷是相背离的。

<6> AJAX不能很好支持移动设备。

一些手持设备（如手机、PDA等）现在还不能很好的支持Ajax，比如说我们在手机的浏览器上打开采用Ajax技术的网站时，它目前是不支持的。

<7> 客户端过肥，太多客户端代码造成开发上的成本。

编写复杂、容易出错 ；冗余代码比较多（层层包含js文件是AJAX的通病，再加上以往的很多服务端代码现在放到了客户端）；破坏了Web的原有标准。

AJAX注意点及适用和不适用场景

(1) 注意点

Ajax开发时，网络延迟——即用户发出请求到服务器发出响应之间的间隔——需要慎重考虑。不给予用户明确的回应，没有恰当的预读数据，或者对XMLHttpRequest的不恰当处理，都会使用户感到延迟，这是用户不希望看到的，也是他们无法理解的。通常的解决方案是，使用一个可视化的组件来告诉用户系统正在进行后台操作并且正在读取数据和内容。

(2) Ajax适用场景

<1>表单驱动的交互

<2>深层次的树的导航

<3>快速的用户与用户间的交流响应

<4>类似投票、yes/no等无关痛痒的场景

<5>对数据进行过滤和操纵相关数据的场景

<6>普通的文本输入提示和自动完成的场景

(3)Ajax不适用场景

<1>部分简单的表单

<2>搜索

<3>基本的导航

<4>替换大量的文本

<5>对呈现的操纵

26.同步和异步的区别?

同步：提交请求->等待服务器处理->处理完毕返回; 这个期间客户端浏览器不能干任何事;

当JS代码加载到当前AJAX的时候会把页面里所有的代码停止加载，页面出现假死状态，当这个AJAX执行完毕后才会继续运行其他代码页面假死状态解除。

异步：请求通过事件触发->服务器处理（这是浏览器仍然可以作其他事情）->处理完毕;

当ajax发送请求后，在等待server端返回的这个过程中，前台会继续 执行ajax块后面的脚本，直到server端返回正确的结果才会去执行success，也就是说这时候执行的是两个线程，ajax块发出请求后一个线程 和ajax块后面的脚本（另一个线程）。

27.如何解决跨域问题?

什么是跨域？只要协议、域名、端口有任何一个不同，都被当作是不同的域。

（1）主域相同的跨域

1.1 document.domain

document.domain的场景只适用于不同子域的框架间的交互，及主域必须相同的不同源。

（2）完全不同源的跨域（两个页面之间的通信）

2.1 location.hash

原理是利用location.hash来进行传值。

优点：1.可以解决域名完全不同的跨域。2.可以实现双向通讯。

缺点：location.hash会直接暴露在URL里，并且在一些浏览器里会产生历史记录，数据安全性不高也影响用户体验。另外由于URL大小的限制，支持传递的数据量也不大。有些浏览器不支持onhashchange事件，需要轮询来获知URL的变化。

2.2 window.name

window对象有个name属性，该属性有个特征：即在一个窗口(window)的生命周期内,窗口载入的所有的页面都是共享一个window.name的，每个页面对window.name都有读写的权限，window.name是持久存在一个窗口载入过的所有页面中的。

2.3 postMessage（HTML5中的XMLHttpRequest Level 2中的API）

HTML5为了解决这个问题，引入了一个全新的API：跨文档通信 API（Cross-document messaging）。这个API为window对象新增了一个window.postMessage方法，允许跨窗口通信，不论这两个窗口是否同源。

2.4 动态创建script

因为script标签不受同源策略的限制。

（3）AJAX请求不同源的跨域

3.1 CORS CORS（Cross-Origin Resource Sharing）跨域资源共享，定义了必须在访问跨域资源时，浏览器与服务器应该如何沟通。CORS背后的基本思想就是使用自定义的HTTP头部让浏览器与服务器进行沟通，从而决定请求或响应是应该成功还是失败。

所有浏览器都支持该功能，IE浏览器不能低于IE10。通过 XMLHttpRequest 对象发起。但Internet Explorer 8 和 9 可以通过 XDomainRequest 对象来实现CORS。可以把CORS分为：简单请求、预请求和附带凭证信息的请求。

CORS的优点：CORS支持所有类型的HTTP请求，是跨域HTTP请求的根本解决方案。

3.2 JSONP

基本原理：网页通过添加一个`<script>`元素，向服务器请求JSON数据，这种做法不受同源政策限制；服务器收到请求后，将数据放在一个指定名字的回调函数里传回来。

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：简单适用，老式浏览器全部支持，服务器改造小。不需要XMLHttpRequest或ActiveX的支持。

缺点：只支持GET请求。

3.3 web sockets

WebSocket是一种通信协议，使用ws://（非加密）和wss://（加密）作为协议前缀。该协议不实行同源政策，只要服务器支持，就可以通过它进行跨源通信。

（4）服务器上设置代理页面

前端解决跨域问题的8种方案（最新最全）

前端跨域问题及解决方案

28.页面编码和被请求的资源编码如果不一致如何处理？

没碰到过这样的问题。但理论上讲，js代码里面的DOM操作使用的element等等应该是ASCII的。那么就不存在问题。如果要在DOM里面假如UTF-8的字符，可能需要转码了。随便bing了一下"javascript utf8转gbk"，还挺多。

29.模块化开发怎么做？

详解JavaScript模块化开发
Javascript模块化编程（一）：模块的写法
Webpack 中文指南

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

浅析JS模块规范：AMD，CMD，CommonJS
CommonJS规范的核心思想是允许模块通过 require 方法来同步加载所要依赖的其他模块，然后通过 exports 或 module.exports 来导出需要暴露的接口。
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()
    // ...
})
```

31.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？）

如何实现一个 CMD 模块加载器

32.让你自己设计实现一个requireJS，你会怎么做？

33.谈一谈你对ECMAScript6的了解？

30分钟掌握ES6/ES2015核心内容

34.ECMAScript6 怎么写class？为什么会出现class这种东西?

ECMAScript 2015 中引入的 JavaScript 类主要是 JavaScript 现有的基于原型的继承的语法糖。 类语法不是向JavaScript引入一个新的面向对象的继承模型。JavaScript类提供了一个更简单和更清晰的语法来创建对象并处理继承。

类实际上是个“特殊的函数”，就像你可以函数表达式和函数声明一样，类语法有两个组成部分：类表达式和类声明。

定义一个类的一种方法是使用一个类声明。要声明一个类，你可以使用带有class关键字的类名（这里是“Rectangle”）。

类声明

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
}
```

类表达式

一个类表达式是定义一个类的另一种方式。类表达式可以是被命名的或匿名的。赋予一个命名类表达式的名称是类的主体的本地名称。

```
/* 匿名类 */
let Rectangle = class {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
};

/* 命名的类 */
let Rectangle = class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
};
```

JavaScript 类 MDN
JavaScript 原型系统的变迁，以及 ES6 class

35.异步加载的方式有哪些？同步加载、异步加载、延迟加载、预加载

同步加载
我们平时使用的最多的一种方式。同步模式，又称阻塞模式，会阻止浏览器的后续处理，停止后续的解析，只有当当前加载完成，才能进行下一步操作。所以默认同步执行才是安全的。但这样如果js中有输出document内容、修改dom、重定向等行为，就会造成页面堵塞。所以一般建议把`<script>`标签放在`<body>`结尾处，这样尽可能减少页面阻塞。

异步加载
异步加载又叫非阻塞加载，浏览器在下载执行js的同时，还会继续进行后续页面的处理。
主要有三种方式。
方法一：也叫Script DOM Element

```
(function(){
    var scriptEle = document.createElement("script");
    scriptEle.type = "text/javasctipt";
    scriptEle.async = true;
    scriptEle.src = "http://cdn.bootcss.com/jquery/3.0.0-beta1/jquery.min.js";
    var x = document.getElementsByTagName("head")[0];
    x.insertBefore(scriptEle, x.firstChild);
 })();
```

属性是HTML5中新增的异步支持。此方法被称为Script DOM Element 方法。
Google Analytics 和 Google+ Badge 都使用了这种异步加载代码.

```
(function(){;
    var ga = document.createElement('script');
    ga.type = 'text/javascript';
    ga.async = true;
    ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-ar
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(ga, s);
})();
```

但是这种加载方式执行完之前会阻止onload事件的触发，而现在很多页面的代码都在onload时还执行额外的渲染工作，所以还是会阻塞部分页面的初始化处理。

方法二：onload时的异步加载

```
(function(){
        if(window.attachEvent){
        window.attachEvent("load", asyncLoad);
    }else{
        window.addEventListener("load", asyncLoad);
    }
    var asyncLoad = function(){
        var ga = document.createElement('script');
        ga.type = 'text/javascript';
        ga.async = true;
        ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.googl
        var s = document.getElementsByTagName('script')[0];
        s.parentNode.insertBefore(ga, s);
    }
)();
```

这种方法只是把插入script的方法放在一个函数里面，然后放在window的onload方法里面执行，这样就解决了阻塞onload事件触发的问题。
注:DOMContentLoaded与load的区别。前者是在document已经解析完成，页面中的dom元素可用，但是页面中的图片，视频，音频等资源未加载完，作用同jQuery中的ready事件；后者的区别在于页面所有资源全部加载完毕。

方法三：其他方法

由于JavaScript的动态性，还有很多异步加载方法： XHR Injection、 XHR Eval、 Script In Iframe、 Script defer属性、 document.write(script tag)。

XHR Injection(XHR 注入)：通过XMLHttpRequest来获取javascript，然后创建一个script元素插入到DOM结构中。ajax请求成功后设置script.text为请求成功后返回的responseText。

```
//获取XMLHttpRequest对象，考虑兼容性。
    var getXmlHttp = function(){
        var obj;
        if (window.XMLHttpRequest)
            obj = new XMLHttpRequest();
        else
            obj = new ActiveXObject("Microsoft.XMLHTTP");
        return obj;
    };
    //采用Http请求get方式;open()方法的第三个参数表示采用异步(true)还是同步(false)处理
    var xmlHttp = getXmlHttp();
    xmlHttp.open("GET", "http://cdn.bootcss.com/jquery/3.0.0-beta1/jquery.min.js", true);
    xmlHttp.send();
    xmlHttp.onreadystatechange = function(){
        if (xmlHttp.readyState == 4 && xmlHttp.status == 200){
            var script = document.createElement("script");
            script.text = xmlHttp.responseText;
            document.getElementsByTagName("head")[0].appendChild(script);
        }
    }
```

XHR Eval：与XHR Injection对responseText的执行方式不同，直接把responseText放在eval()函数里面执行。

```
//获取XMLHttpRequest对象，考虑兼容性。
    var getXmlHttp = function(){
        var obj;
        if (window.XMLHttpRequest)
            obj = new XMLHttpRequest();
        else
            obj = new ActiveXObject("Microsoft.XMLHTTP");
        return obj;
    };
    //采用Http请求get方式;open()方法的第三个参数表示采用异步(true)还是同步(false)处理
    var xmlHttp = getXmlHttp();
    xmlHttp.open("GET", "http://cdn.bootcss.com/jquery/3.0.0-beta1/jquery.min.js", true);
    xmlHttp.send();
    xmlHttp.onreadystatechange = function(){
        if (xmlHttp.readyState == 4 && xmlHttp.status == 200){
            eval(xmlHttp.responseText);
            //alert($);//可以弹出$,表明JS已经加载进来。click事件放在其它出会出问题，应该是还没加载进来
            $("#btn1").click(function(){
                alert($(this).text());
            });
        }
    }
```

Script In Irame：在父窗口插入一个iframe元素，然后再iframe中执行加载JS的操作。
HTML5新属性：async和defer属性。

延迟加载
有些JS代码在某些情况在需要使用，并不是页面初始化的时候就要用到。延迟加载就是为了解决这个问题。将JS切分成许多模块，页面初始化时只加载需要立即执行的JS，然后其它JS的加载延迟到第一次需要用到的时候再加载。类似图片的延迟加载。
JS的加载分为两个部分：下载和执行。异步加载只是解决了下载的问题，但是代码在下载完成后就会立即执行，在执行过程中浏览器处于阻塞状态，响应不了任何需求。

解决思路：为了解决JS延迟加载的问题，可以利用异步加载缓存起来，但不立即执行，需要的时候在执行。如何进行缓存呢？将JS内容作为Image或者Object对象加载缓存起来，所以不会立即执行，然后在第一次需要的时候在执行。

```
1：模拟较长的下载时间：
利用thread让其sleep一段时间在执行下载操作。
2：模拟较长的JS代码执行时间
var start = Number(new Date());
while(start + 5000 > Number(new Date())){//执行JS}
这段代码将使JS执行5秒才完成！
```

JS延迟加载机制(LazyLoad)：简单来说，就是在浏览器滚动到某个位置在触发相关的函数，实现页面元素的加载或者某些动作的执行。如何实现浏览器滚动位置的检测呢？可以通过一个定时器来实现，通过比较某一时刻页面目标节点位置和浏览器滚动条高度来判断是否需要执行函数。

预加载

预加载是一种浏览器机制，使用浏览器空闲时间来预先下载/加载用户接下来很可能会浏览的页面/资源，当用户访问某个预加载的链接时，如果从缓存命中,页面就得以快速呈现。

36.document.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。
document.write是直接写入到页面的内容流，如果在写之前没有调用document.open, 浏览器会自动调用open。每次写完关闭之后重新调用该函数，会导致页面被重写。
innerHTML则是DOM页面元素的一个属性，代表该元素的html内容。你可以精确到某一个具体的元素来进行更改。如果想修改document的内容，则需要修改document.documentElement.innerElement。
innerHTML很多情况下都优于document.write，其原因在于其允许更精确的控制要刷新页面的那一个部分。

37.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
   getElementById()    //通过元素Id，唯一性
```

38.call() 和 apply() 的含义和区别？

`obj.call(thisObj, arg1, arg2, ...);`

`obj.apply(thisObj, [arg1, arg2, ...]);`

两者作用一致，都是把obj(即this)绑定到thisObj，这时候thisObj具备了obj的属性和方法。或者说thisObj『继承』了obj的属性和方法。绑定后会立即执行函数。
唯一区别是apply接受的是数组参数，call接受的是连续参数。
作用：

调用原生对象的方法

示例：

```
var a = {0:1, 1:"yjc", length: 2};
a.slice(); //TypeError: a.slice is not a function
Array.prototype.slice.call(a);//[1, "yjc"]
```

对象a类似array，但不具备array的slice等方法。使用call绑定，这时候就可以调用slice方法。 实现继承
通过call和apply，我们可以实现对象继承。

示例：

```
var Parent = function(){
    this.name = "yjc";
    this.age = 22;
}
var child = {};
console.log(child);//Object {} ,空对象
Parent.call(child);
console.log(child); //Object {name: "yjc", age: 22}
```

以上实现了对象的继承。

`obj.bind(thisObj, arg1, arg2, ...)`;

把obj绑定到thisObj，这时候thisObj具备了obj的属性和方法。与call和apply不同的是，bind绑定后不会立即执行。

apply的其他巧妙用法：

看到这里，我就会觉得既然apply和call的用法差不多，那么为什么还同时存在这两种方法呢？完全可以丢掉一个呀。后来才发现一篇文章中讲到apply因为它所传参数为数组这一特点还有许多其他的妙用。 a) Math.max可以实现得到数组中最大的一项：

因为Math.max参数里面不支持Math.max([param1,param2])也就是数组，但是它支持Math.max(param1,param2,param3…)，所以可以根据apply的特点来解决var max=Math.max.apply(null,array)，这样轻易的可以得到一个数组中最大的一项。(apply会将一个数组转换为一个参数接一个参数的传递给方法)

这块在调用的时候第一个参数给了一个null，这个是因为没有对象去调用这个方法，只需要用这个方法帮助运算，得到返回的结果就行，所以直接传递了一个null过去。

b) Math.min可以实现得到数组中最小的一项：

同样和max是一个思想var min=Math.min.apply(null,array)。

c) Array.prototype.push 可以实现两个数组合并：

同样push方法没有提供push一个数组，但是它提供了push(param1,param,…paramN) 所以同样也可以通过apply来转换一下这个数组，即:

```
var arr1=new Array("1","2","3");
var arr2=new Array("4","5","6");
Array.prototype.push.apply(arr1,arr2);
```

也可以这样理解，arr1调用了push方法，参数是通过apply将数组装换为参数列表的集合。

d) 小结：通常在什么情况下,可以使用apply类似Math.min等之类的特殊用法:

一般在目标函数只需要n个参数列表,而不接收一个数组的形式（ [param1[,param2[,…[,paramN]]]] ），可以通过apply的方式巧妙地解决这个问题。

39.数组和对象有哪些原生方法，列举一下？

```
Array 对象方法
方法      描述
```

```
concat()        连接两个或更多的数组，并返回结果。
join()  把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。
pop()   删除并返回数组的最后一个元素
push()  向数组的末尾添加一个或更多元素，并返回新的长度。
reverse()       颠倒数组中元素的顺序。
shift() 删除并返回数组的第一个元素
slice() 从某个已有的数组返回选定的元素
sort()  对数组的元素进行排序
splice()        删除元素，并向数组添加新元素。
toSource()      返回该对象的源代码。
toString()      把数组转换为字符串，并返回结果。
toLocaleString()        把数组转换为本地数组，并返回结果。
unshift()       向数组的开头添加一个或更多元素，并返回新的长度。
valueOf()       返回数组对象的原始值
```

40.JS 怎么实现一个类。怎么实例化这个类

41.JavaScript中的作用域与变量声明提升？

42.如何编写高性能的Javascript？

44.JQuery的源码看过吗？能不能简单概况一下它的实现原理？

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

46.jquery中如何将数组转化为json字符串，然后再转化回来？

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

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

49.jQuery 的队列是如何实现的？队列可以用在哪些地方？

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

51.JQuery一个对象可以同时绑定多个事件，这是如何实现的？

52.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？

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

54.针对 jQuery性能的优化方法？

55.Jquery与jQuery UI有啥区别？

57.jquery 中如何将数组转化为json字符串，然后再转化回来？

58.jQuery和Zepto的区别？各自的使用场景？

Zepto是一个轻量级的针对现代高级浏览器的JavaScript库， 它与jquery有着类似的api。
Zepto的设计目的是提供 jQuery 的类似的API，但并不是100%覆盖 jQuery 。Zepto设计的目的是有一个5-10k的通用库、下载并快速执行、有一个熟悉通用的API，所以你能把你主要的精力放到应用开发上。

59.针对 jQuery 的优化方法？

60.Zepto的点透问题如何解决？

没使用过

61.jQueryUI如何自定义组件?

62.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？

63.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。

64.移动端最小触控区域是多大？

iOS HIG 推荐值为 44pt
Google Material Design 推荐值为 48dp
移动应用设计时触摸目标大小多少为合适？

65.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?

66.把 `<script>` 标签 放在页面的最底部的`</body>`之前和`</body>`之后有什么区别？浏览器会如何解析它们？

按照HTML5标准中的HTML语法规则，如果在`</body>`后再出现`<script>`或任何元素的开始标签，都是parse error，浏览器会忽略之前的`</body>`，即视作仍旧在body内。所以实际效果和写在`</body>`之前是没有区别的。
总之，这种写法虽然也能work，但是并没有带来任何额外好处，实际上出现这样的写法很可能是误解了“将script放在页面最末端”的教条。所以还是不要这样写为好。
知乎问题讨论

67.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。 ）

68.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?

69.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？

70.解释JavaScript中的作用域与变量声明提升？

作用域表示变量或函数能够被访问的范围，一般来说，变量和函数可以被定义在全局和局部作用域范围中， 也就是我们所说的全局作用域（Global Scope）和局部作用域（Local Scope ）。
当某个东西是全局的，就意味着它可以在你代码中的任何地方被访问到；
与全局作用域相反，局部作用域表示变量和函数定义在代码的某些区域中，也只能在这些区域中被访问到，例如在

函数内部定义的变量或函数。
在JavaScript中，函数、变量的声明都会被提升（hoisting）到该函数或变量所在的scope的顶部。

71.那些操作会造成内存泄漏？

不再用到的内存，没有及时释放，就叫做内存泄漏（memory leak）。
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)；
关于js闭包是否真的会造成内存泄漏

72.JQuery一个对象可以同时绑定多个事件，这是如何实现的？

73.Node.js的适用场景？

高并发、聊天、实时消息推送

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

75.解释一下 Backbone 的 MVC 实现方式？

76.什么是“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?

segmentfault上关于前端路由的回答

77.知道什么是webkit么? 知道怎么用浏览器的各种工具来调试和debug代码么?

WebKit是一种用来让网页浏览器绘制网页的排版引擎。
wikipedia百科之WebKit

78.如何测试前端代码么? 知道BDD, TDD, Unit Test么? 知道怎么测试你的前端工程么(mocha, sinon, jasmin, qUnit..)?

79.前端templating(Mustache, underscore, handlebars)是干嘛的, 怎么用?

https://www.zhihu.com/question/32524504

80.简述一下 Handlebars 的基本用法？

81.简述一下 Handlerbars 的对模板的基本处理流程， 如何编译的？如何缓存的？

82.用js实现千位分隔符?(来源：前端农民工，提示：正则+replace)

83.检测浏览器版本有哪些方式？

使用JavaScript的内置对象 navigator 的属性userAgent的值来判（navigator.userAgent）。
navigator是javascript的内置对象，通常用于检测浏览器与操作系统的版本。
常用的属性:

```
appCodeName -- 浏览器代码名的字符串表示
appName -- 官方浏览器名的字符串表示
```

```
appVersion -- 浏览器版本信息的字符串表示
cookieEnabled -- 如果启用cookie返回true，否则返回false
javaEnabled -- 如果启用java返回true，否则返回false
platform -- 浏览器所在计算机平台的字符串表示
plugins -- 安装在浏览器中的插件数组
taintEnabled -- 如果启用了数据污点返回true，否则返回false
userAgent -- 用户代理头的字符串表示（就是包含浏览器版本信息等的字符串）
```

JS 获得浏览器类型和版本

84.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，你来说下会执行几次事件，然后会先执行冒泡还是捕获？

分析

其他问题

85.原来公司工作流程是怎么样的，如何与其他人协作的？如何跨部门合作？

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

javascript 设计模式
了解23种设计模式

88.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？

89.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 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
```

90.列举IE与其他浏览器不一样的特性？

```
IE支持currentStyle，FIrefox使用getComputStyle
IE 使用innerText，Firefox使用textContent
滤镜方面：IE:filter:alpha(opacity= num)；Firefox：-moz-opacity:num
事件方面：IE：attachEvent：火狐是addEventListener
鼠标位置：IE是event.clientX；火狐是event.pageX
IE使用event.srcElement；Firefox使用event.target
IE中消除list的原点仅需margin:0即可达到最终效果；FIrefox需要设置margin:0;padding:0以及list-style:none
CSS圆角：ie7以下不支持圆角
```

```
1.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；
2.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；
3.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；
4.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()；
```

91.99%的网站都需要被重构是哪本书上写的？

网站重构：应用 web 标准进行设计（第 2 版）

93.是否了解公钥加密和私钥加密。

RSA的公钥和私钥到底哪个才是用来加密和哪个用来解密？
RSA算法原理

94.WEB应用从服务器主动推送Data到客户端有那些方式？

```
html5 websoket
WebSocket 通过 Flash
XHR 长时间连接
XHR Multipart Streaming
不可见的 Iframe
script标签的长时间连接(可跨域)
```

95.对Node的优点和缺点提出了自己的看法？

Node.js是一个基于Chrome JavaScript运行时建立的平台， 用于方便地搭建响应速度快、易于扩展的网络应用。Node.js使用事件驱动，非阻塞I/O模型而得以轻量和高效，非常适合在分布式设备上运行数据密集型的实时应用。

优点：
(1) 高并发（最重要的优点）
(2) 适合I/O密集型应用

缺点：
(1) 不适合CPU密集型应用；
CPU密集型应用给Node带来的挑战主要是, 由于JavaScript单线程的原因，如果有长时间运行的计算（比如大循环），将会导致CPU时间片不能释放，使得后续I/O无法发起；
解决方案：分解大型运算任务为多个小任务，使得运算能够适时释放，不阻塞I/O调用的发起；
(2) 只支持单核CPU，不能充分利用CPU；
(3) 可靠性低，一旦代码某个环节崩溃，整个系统都崩溃；
原因：单进程，单线程

解决方案：
1）Nnigx反向代理，负载均衡，开多个进程，绑定多个端口；
2）开多个进程监听同一个端口，使用cluster模块；
(4) 开源组件库质量参差不齐，更新快，向下不兼容
(5) Debug不方便，错误没有stack trace

96.你有用过哪些前端性能优化的方法？

前端性能优化最佳实践
前端性能优化相关
前端性能优化指南

97.http状态码有那些？分别代表是什么意思？

```
1字头：消息。这一类型的状态码，代表请求已被接受，需要继续处理。
2字头：成功。这一类型的状态码，代表请求已成功被服务器接收、理解、并接受。
3字头：重定向。这类状态码代表需要客户端采取进一步的操作才能完成请求。
4字头：客户端错误。这类状态码代表了客户端看起来可能发生错误，妨碍了服务器的处理。
5字头：服务器错误。这类状态码代表了服务器在处理请求的过程中有错误或者异常状态发生。

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

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）。
```

98.一个页面从输入 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）

```
总体来说分为以下几个过程:
DNS解析
TCP连接
发送HTTP请求
服务器处理请求并返回HTTP报文
浏览器解析渲染页面
连接结束
```

详细参考segmentfault： 从输入URL到页面加载发生了什么

从输入URL到页面加载完成的过程中都发生了什么事情？

```
输入地址
浏览器查找域名的IP地址：这一步包括DNS具体的查找过程，包括：浏览器缓存->系统缓存->路由器缓存...
浏览器向web服务器发送一个HTTP请求
服务器的永久重定向响应（从http://example.com到http://www.example.com）
浏览器跟踪重定向地址
服务器处理请求
服务器返回一个HTTP响应
浏览器显示HTML
浏览器发送请求获取嵌入在HTML中的资源（如图片、音频、视频、CSS、JS等等）
浏览器发送异步请求
```

99.部分地区用户反应网站很卡，请问有哪些可能性的原因，以及解决方法？

100.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?

101.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？

102.你用的得心应手用的编辑器&开发环境是什么样子？

103.对前端界面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。

104.你怎么看待Web App 、hybrid App、Native App？

（1）Native App
Native App是一种基于智能手机本地操作系统如iOS、Android、WP并使用原生程式编写运行的第三方应用程序,也叫本地app。一般使用的开发语言为JAVA、C++、Objective-C。

想创建Native App，开发者必须编写源代码，使用由操作系统开发商提供的工具，对源代码进行编译。代码编译之后以2进制或者字节码的形式运行在操作系统上，直接调用操作系统的Device API。

虽然不同操作系统上进行的开发过程常常很相似，但是每一种移动操作系统都随带各自的独特工具。平台之间的这些区别导致了Native开发方法的最重大缺点之一：为一种移动平台编写的代码无法在另一种平台上使用。

Native App的优缺点总结如下：
优点

```
1、提供最佳的用户体验，最优质的用户界面，最华丽的交互
2、针对不同平台提供不同体验
3、下载到本地，可节省带宽成本
4、可访问本地资源
5、直接访问系统级API
```

```
6、操作速度更快
7、用户留存率高
```

缺点

```
1、移植到不同平台上比较麻烦
2、需要维护多个版本
3、发布新版本需要通过store或market的确认
4、盈利需要与第三方分成
5、开发的成本比较大，需要针对不同平台开发相应的版本
6、更新体验较差、同时也比较麻烦（每一次发布新的版本，都需要做版本打包，
且需要用户手动更新，或一个让用户反感的提示）。
```

（2）Web App

Web App是运行于网络和标准浏览器上，以HTML+JS+CSS等WEB技术开发实现特定功能的应用。通过浏览器来调用Device API，但是只有数量有限的这些API向浏览器里面运行的Web App公开。基于当下开始普及流行的HTML5，Web App可以实现很多原本Native App才可以实现的功能，，比如LBS的功能、本地数据存储、音视频播放的功能，甚至还有调用照相机和结合GPU的硬件加速功能。

由于它不依赖于操作系统，因此开发了一款Web App后，基本能应用于各种系统平台。并且还有版本升级容易的优势（毕竟服务器是受自己控制的）。但是这种方案的缺点也很明显——无法使用系统级API，只能做为一个临时的入口，用户很难留存。

Web App的优缺点总结如下：

优点

```
（1）开发成本低；
（2）更新快；
（3）更新无需通知用户，不需要手动升级；
（4）能够跨多个平台和终端；
（5）维护比较简单。
```

缺点

```
（1）临时性的入口；
（2）无法获取系统级别的通知，提醒，动效等等；
（3）用户留存率低；
（4）需要依赖网络，体验较差。
```

（3）Hybrid App

Hybrid App是一种用Native技术来搭建App的外壳，壳里的内容由Web技术来提供的移动应用，兼具“Native App良好交互体验的优势”和“Web App跨平台开发的优势”。

根据实现的不同，可以细分为两种实现方案：

1）在Native App中使用WebView加载远端Web资源

```
这种方案的Web资源放置在服务器上，开发者不必经历提交和批准过程—有些App商店要求这个过程，就
可以对App进行小幅更新。遗憾的是，这个方法摈弃了任何离线可用性，因为设备与网络没有连接时，
无法访问设备。
```

2）将一组HTML、JavaScript、CSS和媒体文件，封装到App代码中，存储在设备本地，使用Cordova/PhoneGap等框架通过WebView加载本地资源进行页面渲染

```
把Web代码封装到App里面在一定程度上缓解从远端加载静态资源导致UI展示延迟的问题，可以提高性能
和可访问性，并且还可以通过桥接Native和Web来调用一些Device的API。
但是其劣势也很明显，一是不允许远程更新；二是安装包变大；三是如果想调用相关平台的API，需要针
对平台单独进行开发，如果在应用中用到了大量的Device API，那么开发的效率将大大降低。
```

Hybrid App同时使用网页语言与程序语言开发，但其总体特性更接近Native App。只是因为同时使用了web语言编码，所以开发成本和难度比Native App要小很多。因此，Hybrid App兼具了Native App的优势，也兼具了Web App使用HTML5跨平台开发低成本的优势。

Native App、Web App与Hybrid App的比较——简书
聊聊WEB APP、HYBRID APP与NATIVE APP的设计差异

105.你对移动端前端开发的理解？（和Web前端开发的主要区别是什么？）

移动前端开发和 Web 前端开发的区别是什么？——知乎

106.你对加班的看法？

提高工作效率，尽量少加班，但为了项目顺利上线还是要义无反顾的去加班的。 知乎讨论

107.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
```

程序员如何挽救日渐失控的项目？

108.说说最近最流行的一些东西吧？常去哪些网站？

微信小程序、Vue.js、React、es6、Electron
网站： MDN、W3Cfuns、SegmentFault、hacknews、CSDN、慕课、博客园、oschina等
2017年大前端发展趋势

109.如何设计突发大规模并发架构？

如何设计突发大规模并发架构——segmentfault

111.是否了解开源的工具 bower、npm、yeoman、grunt、gulp，一个 npm 的包里的 package.json 具备的必要的字段都有哪些？（名称、版本号，依赖）

112.每个模块的代码结构都应该比较简单，且每个模块之间的关系也应该非常清晰，随着功能和迭代次数越来越多，你会如何去保持这个状态的？

113.Git是什么？知道branch, diff, merge么?

Git是用C语言开发的分布版本控制系统。版本控制系统可以保留一个文件集合的历史记录，并能回滚文件集合到另一个状态（历史记录状态）。

通过分支，可以创造独立的代码副本。默认的分支叫master。Git消耗很少的资源就能创建分支。
通过git diff命令，用户可以查看更改；
通过Merge我们可以合并两个不同分支的结果。Merge通过所谓的三路合并来完成。分别来自两个分支的最新commit和两个分支的最新公共commit.

115.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？

前端代码的测试系列文章

117.知道什么是SEO并且怎么优化么？知道各种meta data的含义么？

118.移动端（Android IOS）怎么做好用户体验？

119.简单描述一下你做过的移动APP项目研发流程？

120.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？

121.你认为怎样才是全端工程师（Full Stack developer）？

122.介绍一个你最得意的作品吧？

123.你有自己的技术博客吗，用了哪些技术？

124.对前端安全有什么看法？

聊一聊WEB前端安全那些事儿

125.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？

XSS (Cross-Site Scripting)：通过构造特殊数据，在用户浏览器上执行特定脚本，从而造成危害（如以用户身份发帖、转账等）。
由于页面内 JavaScript 几乎可以完成各种事情，因此一旦网站上有 XSS 漏洞，那些没有验证码等确认措施的操作大多都能不知情地完成，其危害甚大。
CSRF (Cross-site request forgery)：跨站请求伪造，也被称为“One Click Attack”或者Session Riding，通常缩写为CSRF或者XSRF，是一种对网站的恶意利用。
其实就是网站中的一些提交行为，被黑客利用，你在访问黑客的网站的时候，进行的操作，会被操作到其他网站上(如：你所使用的网络银行的网站)。

126.项目中遇到过哪些印象深刻的技术难题，具体是什么问题，怎么解决？

分析问题——google/翻书/请教——实现——测试
（1）轮播图相关的，一页四个轮播图，竖向轮播图；

129.如何管理前端团队？

责任，执行力，关怀。
好的前端主管是如何带队的？

130.最近在学什么？能谈谈你未来3，5年给自己的规划吗？

一如既往的学习HTML5, CSS3,Javascript,Angularjs，当然对一些新技术也在学习中，比如新的js框架，vue/react,angular2, 模板引擎Jade, 构建工具webpack, 自动化工具gulp,以及一些代码测试工具。

131.js操作获取和设置cookie

```
//创建cookie
//cookie是以键值对的形式保存的，即key=value的格式。各个cookie之间一般是以“;”分隔。
function setCookie(name,value,expires,path,domain,secure) {
    //encodeURIComponent() 函数可把字符串作为 URI 组件进行编码
    var cookieText = encodeURIComponent(name)+ '='+encodeURIComponent(value);
    if (expires instanceof Date){
        cookieText += ';expires=' + expires;
    }
    if (path){
        cookieText += '; path=' + path;
    }
    if (domain){
        cookieText += '; domain=' + domain;
    }
    if (secure){
        cookieText += '; secure';
    }
    document.cookie = cookieText;
}
//获取cookie
function getCookie(name) {
    var cookieName = encodeURIComponent(name)+'=';
    //indexOf() 方法可返回某个指定的字符串值在字符串中首次出现的位置
    //stringObject.indexOf(searchvalue,fromindex)
    var cookieStart = document.cookie.indexOf(cookieName);
    var cookieValue = null;
    if (cookieStart > -1){
        var cookieEnd = document.cookie.indexOf(';',cookieStart);
        if (cookieEnd == -1){
            cookieEnd = document.cookie.length;
        }
        cookieValue = decodeURIComponent(document.cookie.substring(cookieStart+cookieName.length,c
    }
    return cookieValue;
}
//删除cookie
function unsetCookie(name) {
    document.cookie = name + '= ; expires=' +new Date(0);
}
```

132. JavaScript 的同源策略

URL由协议、域名、端口和路径组成，如果两个URL的协议、域名和端口相同，则表示他们同源。
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。同源政策的目的，是为了保证用户信息的安全，防止恶意的网站窃取数据。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。
为什么要有同源限制？
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。

133. ie各版本和 chrome 可以并行下载多少个资源?

IE6 两个并发， iE7 升级之后的 6 个并发，之后版本也是 6 个
Firefox， chrome 也是 6 个