JS面试题

2016-10-08 | JS

请解释事件代理 (event delegation)。

事件委托技术能让你避免对特定的每个节点添加事件监听器；相反，事件监听器是被添加到它们的父元素上。事件监听器会分析从子元素冒泡上来的事件，找到是哪个子元素的事件。

优点：

- 性能得到了优化（需要创建的以及驻留在内存中的事件处理器少了）
- 动态添加的元素也能绑定事件了

请解释 JavaScript 中 this 是如何工作的。

this 永远指向函数运行时所在的对象，而不是函数被创建时所在的对象。

- 函数调用es3和非严格es5为全局对象，严格es5为undefined
- 方法调用this指向调用该方法的对象（调用上下文）
- 构造函数时，this指向新创建的对象
- call() apply() 调用方法时，this指向调用方法的对象，而不是该方法拥有者对象

参考：JavaScript中this的工作原理以及注意事项

请解释原型继承 (prototypal inheritance) 的原理？

原型继承的基础是原型链查找。

原型链查找基本概念：
每一个函数 F 都有一个原型对象（prototype）F.prototype
每一个函数都可以通过 new 关键字化身成为一个类构造函数，new F 会产生一个对象 O
在调用对象的某个属性或者方法，比如 http://O.xxx 的时候，会首先查找对象自身是否有这个方法或者属性，如果没找到就会去对象的构造函数的原型对象中查找（注意有两个定语），也就是查找 O 的构造函数 F 的原型对象 http://F.prototype.xxx
F.prototype 也是一个对象，查找 http://F.prototype.xxx 的时候会重复第 3 步的过程

你怎么看 AMD vs. CommonJS？

请解释为什么接下来这段代码不是 IIFE (立即调用的函数表达式)：function foo(){ }();要做哪些改动使它变成 IIFE?

这里只是声明一个叫foo的function，直接用()执行这样是不成功的，想要变成IIFE就要把声明变成表达式，就可以立即执行了，可以这样(function foo(){})()或者(function foo(){}())，这就是用括号把定义强转成表达式，当然还有其他方法，关键就是声明不可以执行，表达式才可以执行。

参考：揭秘IIFE语法

描述以下变量的区别：null，undefined 或 undeclared？ 该如何检测它们？

undefined

未定义，在变量没有赋值的时候的值即为undefined。“缺少值”，就是此处应该有一个值，但是还没有定义。

underclared

即为被污染的命名，访问没有被声明的变量，会抛出异常，终止执行。尝试访问一个undeclared的变量，浏览器会报错，JS执行会中断。

null

是一个空的对象引用。“没有对象”，即该处不应该有值

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。typeof undefined会返回undefined ，而typeof null 总返回 object(typeof有六种可能：“number”、“string”、“boolean”、“object”、“function”、“undefined”)

false == undefined;//false
false == null;//false
null == undefined;//true

该如何检测它们？

var obj;
obj ===undefined; //检测undfined 方法一
typeof obj === ‘undefined’;//检测undefined方法2
obj = null;
obj === null;//来检测null
typeof null;//‘object’

什么是闭包 (closure)，如何使用它，为什么要使用它？

定义：闭包就是可以读取到其他函数内部变量的函数。
闭包的用途：

- 一个是前面提到的可以读取函数内部的变量。(外界无法访问函数的内部的私有方法和变量，只能通过提供的接口访问)
- 另一个就是让这些变量的值始终保持在内存中。
- 可以避免污染全局变量，实现私有方法或者变量等

注意：

- 由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。
- 闭包会在父函数外部，改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

请举出一个匿名函数的典型用例？

- 匿名函数可以用作**回调函数**执行，可以防止全局变量污染。

- 在 JS **框架中常使用**匿名函数来避免全局变量的污染。
  $.(“input”).each(function(e){this.val(‘OK’)});

你是如何组织自己的代码？是使用模块模式，还是使用经典继承的方法？

请指出 JavaScript 宿主对象 (host objects) 和原生对象 (native objects) 的区别？

原生对象，独立于宿主环境的 ECMAScript 实现提供的对象。为array obj regexp date function等可以new实例化的对象。

内置对象为gload Math 等，开发者不必明确实例化内置对象，它已被实例化了。类似于isNaN()、parseInt()和parseFloat()方法等，看起来都是函数，而实际上，它们都是Global对象的方法。具体可以参考 JavaScript 全局对象

宿主对象。即由 ECMAScript 实现的宿主环境（操作系统和浏览器）提供的对象。所有的BOM和DOM对象都是宿主对象。因为其对于不同的“宿主”环境所展示的内容不同（这就是兼容性和特性检测的缘由）。ECMAScript官方未定义的对象都属于宿主对象。

请指出以下代码的区别：function Person(){}、var person = Person()、var person = new Person()？

第一个为函数声明，第二个将函数person()返回值赋值给person,第三个通过Person()的构造器创建了一个对象让person变量引用该对象；

.call 和 .apply 的区别是什么？

call和apply都是调用一个对象的一个方法，以另一个对象替换当前对象。它们都属于Function.prototype的一个方法，所以每个function实例都有call和apply属性。这两个方法可以用来代替另一个对象调用一个方法，可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。

区别

两者传递的参数不同，虽然函数第一个参数都是要传入给当前对象的对象，但是，apply的第二个参数是一个参数数组，将多个参数组合成为一个数组传入；而call第二个参数则是直接的参数列表。

请解释 Function.prototype.bind？

Function.prototype.bind()其实就是函数绑定。函数的接收者取决于他是如何被调用，可以通过调用.bind()给函数绑定作用域上下文（this的值），即函数的接收者。

```
var foo = { x: 3}
var bar = function(){console.log(    this.x);}

bar(); // undefinedvar
boundFunc = bar.bind(foo);//隐式看作是在foo作用域里调用bar方法
boundFunc(); // 3
```

.bind()创建了一个函数，当这个函数在被调用的时候，它的 this 关键词会被设置成被传入的值（这里指调用bind()时传入的参数）也就是我们传入想要的上下文。 简单的用法： 关于 Function.prototype.bind() 内部，这里有个非常简单的例子：

```
Function.prototype.bind = function (scope) {
    var fn = this;
    return function () {
        return fn.apply(scope);//使用call效果一样
    };
}
```

在什么时候你会使用 document.write()？

document.write()方法可以用在两个方面：

- 页面载入过程中用实时脚本创建页面内容，该方法需要一个字符串参数，它是写到窗口或框架中的HTML内容。
- 以及用延时脚本创建本窗口或新窗口的内容。该方法需要一个字符串参数，它是写到窗口或框架中的HTML内容。

记住，在载入页面后，浏览器输出流自动关闭。在此之后，任何一个对当前页面进行操作的document.write()方法将打开一个新的输出流，它将清除当前页面内容(包括源文档的任何变量或值)。因此，假如希望用脚本生成的HTML替换当前页面，就必须把HTML内容连接起来赋给一个变量，使用一个document.write()方法完成写操作。不必清除文档并打开一个新数据流，一个document.write()调用就可完成所有的操作。

关于document.write()方法还有一点要说明的是它的相关方法document.close()。脚本向窗口(不管是本窗口或其他窗口)写完内容后，必须关闭输出流。在延时脚本的最后一个document.write()方法后面，必须确保含有document.close()方法，不这样做就不能显示图像和表单。并且，任何后面调用的document.write()方法只会把内容追加到页面后，而不会清除现有内容来写入新值。为了演示document.write()方法，我们提供了同一个应用程序的两个版本。

大多数生成的广告代码依旧使用 document.write()，虽然这种用法会让人很不爽。

请指出浏览器特性检测，特性推断和浏览器 UA 字符串嗅探的区别？

检测浏览器的特殊名称和版本（用户代理检测）即浏览器UA字符串嗅探。浏览器嗅探技术可以快捷的将代码进行分支，以便针对不同的浏览器应用不同的指令；针对特定浏览器的特定版本，超出范围之外都是不可靠的

请尽可能详尽的解释 Ajax 的工作原理？

使用 Ajax 都有哪些优劣？

优势：可以刷新局部页面，而不用整体页面都刷新
缺点：用户禁用javascript的情况

请解释 JSONP 的工作原理，以及它为什么不是真正的 Ajax。

工作原理

JSONP动态创建script标签，回调函数。Ajax是页面无刷新请求数据操作，动态添加一个&ltscript>标签，而script标签的src属性是没有**跨域**的限制的。这样说来，这种跨域方式其实与ajax XmlHttpRequest协议无关了。

当GET请求从被调用页面返回时，可以返回一段JavaScript代码，这段代码会自动调用主页面中的一个callback函数。

优点

不受同源策略的影响，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；并且在请求完毕后可以通过调用callback的方式回传结果

缺点

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。

请解释变量声明提升 (hoisting)。、

在JavaScript代码运行之前其实是有一个编译阶段的。编译之后才是从上到下，一行一行解释执行。变量提升就发生在编译阶段，它把变量和函数的声明提升至作用域的顶端。（编译阶段的工作之一就是将变量与其作用域进行关联）。

变量提升需要**注意两点**：

- 提升的部分只是变量声明，赋值语句和可执行的代码逻辑还保持在原地不动
- 提升只是将变量声明提升到变量所在的变量范围的顶端，并不是提升到全局范围、

函数声明：

- 变量声明和函数声明都会得到变量提升，但函数声明会最先得到提升，然后是变量声明（函数时一等公民）
- 对于函数声明来说，如果定义了相同的函数变量声明，后定义的声明会覆盖掉先前的声明

参考： Javascript变量提升

请描述事件冒泡机制 (event bubbling)。

从目标元素开始，往顶层元素传播。途中如果有节点绑定了相应的事件处理函数，这些函数都会被一次触发。如果想阻止事件起泡，可以使用e.stopPropagation()（Firefox）或者e.cancelBubble=true（IE）来组织事件的冒泡传播

参考： 事件委托和冒泡机制有关系吗？

“attribute” 和 “property” 的区别是什么？

DOM元素的attribute和property两者是不同的东西。attribute翻译为“特性”，property翻译为“属性”。

attribute是一个特性节点，每个DOM元素都有一个对应的attributes属性来存放所有的attribute节点，attributes是一个类数组的容器，说得准确点就是NameNodeMap，不继承于Array.prototype，不能直接调用Array的方法。attributes的每个数字索引以名值对(name=”value”)的形式存放了一个attribute节点。

property就是一个属性，如果把DOM元素看成是一个普通的Object对象，那么property就是一个以名值对(name=”value”)的形式存放在Object中的属性。要添加和删除property和普通的对象类似。

很多attribute节点还有一个相对应的property属性，比如上面的div元素的id和class既是attribute，也有对应的property，不管使用哪种方法都可以访问和修改。

总之，attribute节点都是在HTML代码中可见的，而property只是一个普通的名值对属性

为什么扩展 JavaScript 内置对象不是好的做法？

因为你不知道哪一天浏览器或javascript本身就会实现这个方法，而且和你扩展的实现有不一致的表现。到时候你的javascript代码可能已经在无数个页面中执行了数年，而浏览器的实现导致所有使用扩展原型的代码都崩溃了。

需要给Array原型添加一个distinct的方法，最好检查是否存在同名的方法，避免自定义方法覆盖原生方法：

Arrray.prototype.distinct = Arrray.prototype.distinct || function(){/...../}

请指出 document load 和 document DOMContentLoaded 两个事件的区别。

ready 表示文档的 DOM 已经加载完成（不包含图片、视频等资源）；load 表示整个网页加载完成。可以看出，ready 事件发生在 load 事件之前。

== 和 === 有什么不同？

如果两边的操作数具有一致的类型且拥有相同的值时，=== 返回 true，!== 返回 false。

请解释 JavaScript 的同源策略 (same-origin policy)。

同源策略限制了一个源（origin）中加载文本或脚本与来自其它源（origin）中资源的交互方式。

同源策略出于安全，不允许源 A 的脚本读取（read）源 B 的资源的内容，但却允许执行（execute）源 B 的资源。这个概念也有些拗口。简单说，有一个页面调用了 Google CDN 提供的 jQuery，以及其它 CDN 上的 Bootstrap JS、CSS 代码，虽然它们与我的博客不同源，但我可以用它们来操作这个页面，并应用样式，这是执行的概念。

参考：浏览器的同源策略

如何实现下列代码：[1,2,3,4,5].duplicator(); // [1,2,3,4,5,1,2,3,4,5]

将此方法添加至 Array.prototype 实现，代码如下：

```
Array.prototype.duplicator = function(){
    var l = this.length,i;
    for(i=0;i<l;i++){
      this.push(this[i])
    }
}
```

什么是三元表达式 (Ternary expression)？“三元 (Ternary)” 表示什么意思？

一个运算符如果有一个操作数，为一元运算符，两个为二元，三个为三元运算符，三元表达式则为一个三元运算表达式！

什么是 “use strict”; ? 使用它的好处和坏处分别是什么？

ECMAScript5中引入的严格模式,通过让JavaScript运行环境对一些开发过程中最常见和不易发现的错误做出和当前不同的处理,来让开发者拥有一个”更好”的JavaScript语言。

好处

- 消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;
- 消除代码运行的一些不安全之处，保证代码运行的安全；
- 提高编译器效率，增加运行速度；
- 为未来新版本的Javascript做好铺垫。

好处具体体现

- 去除WITH关键词
- 防止意外为全局变量赋值
- 函数中的THIS不再默认指向全局
- 防止重名
- 安全的 EVAL()
- 对只读属性修改时抛出异常

坏处

同样的代码，在“严格模式”中，可能会有不一样的运行结果；一些在“正常模式”下可以运行的语句，在“严格模式”下将不能运行

总结

启用JavaScript严格模式,它能帮你发现代码中未曾注意到的错误。不要在全局环境中启用,但你能尽量多的使用IIFE(立即执行函数表达式)来把严格模式作用到多个函数范围内。一开始,你会遇到之前未曾碰到过的错误提示,这是正常的。当启用严格模式后,请确保在支持的浏览器中做了测试,以发现新的潜在问题。一定不要仅仅在代码中添加一行”use strict”就假定余下的代码能正常工作。

参考

JavaScript严谨模式(Strict Mode)提升开发效率和质量

请实现一个遍历至 100 的 for loop 循环，在能被 3 整除时输出 “fizz”，在能被 5 整除时输出 “buzz”，在能同时被 3 和 5 整除时输出 “fizzbuzz”。

```
for (var i = 1; i <= 30; i++) {
```

```
    if (i % 3 === 0) {
        if (i % 5 === 0) {
            alert('fizzbuzz' + i);
            continue;
        }
        alert('fizz' + i);
        continue;
    } else if (i % 5 === 0) {
        if (i % 3 === 0) {
            alert('fizzbuzz' + i);
            continue;
        }
        alert('buzz' + i);
        continue;
    }
}
```

为何通常会认为保留网站现有的全局作用域 (global scope) 不去改变它，是较好的选择？

它的意思是: 尽量少在全局作用域定义变量。

目的:

减少名称冲突 利于模块化

为何你会使用 load 之类的事件 (event)？此事件有缺点吗？你是否知道其他替代品，以及为何使用它们？

要等到等页面完全加载后(所有图像、javascript文件、CSS等外部文件)。替代：把script标签放到最后面。

请解释什么是单页应用 (single page app), 以及如何使其对搜索引擎友好 (SEO-friendly)。

单页应用是一种特殊的web应用，它将所有的活动局限于一个web页面中，仅在该Web页面初始化时加载相应的HTML、JavaScript 和 CSS。

优点

- 用户体验：对于内容的改动不需要加载整个页面
- 高效：服务器压力很小，消耗更少的带宽，能够与面向服务的架构更好地结合。
- 经典MVC开发模式，前后端各负其责。
- 一套Server API，多端使用（web、移动APP等）
- 重前端，业务逻辑全部在本地操作，数据都需要通过AJAX同步、提交

缺点

- 不利于SEO：解决方案也有一些：H5pushState,通过浏览器历史记录让搜索引擎抓取；url中#！

```
复杂的单页架构页面，对Google来说抓取比较困难，于是给开发者制定一个规范：
    1）、网站提交sitemap给Google；
    2）、Google发现URL里有#!符号，例如example.com/#!/detail/1，于是Google开始抓取example.com/?_escaped_fragment_=/detail
```

- 首屏渲染速度慢

使用 Promises 而非回调 (callbacks) 优缺点是什么？

优点：易读性改善

使用一种可以编译成 JavaScript 的语言来写 JavaScript 代码有哪些优缺点？

以Typescript为例子：

typescript是javascript的强类型版本，在编译期去掉类型和特有语法，生成纯粹的javascript代码。TypeScript 是 JavaScript 的超集，这意味着他支持所有的 JavaScript 语法。并在此之上对 JavaScript 添加了一些扩展，如 class / interface / module 等。这样会大大提升代码的可阅读性。

优点：

- 静态类型检查
- IDE 智能提示（编译阶段即可发现类型不匹配的错误）
- 代码重构
- 可读性

缺点：

- 不指定类型就写不了程序，类型只是辅助信息，并不是程序的本之后
- 灵活性问题

你使用哪些工具和技术来调试 JavaScript 代码？

- alert
- console.log
- 断点调试（这三种调试方式都是打断点）

  js断点调试

  source断点调试

  Debugger断点（具体的说就是通过在代码中添加"debugger;"语句，当代码执行到该语句的时候就会自动断点。）
- DOM断点调试

  当节点内部子节点变化时断点

  当节点属性发生变化时断点

  当节点被移除时断点

参考：前端开发中的JS调试技巧

请解释可变 (mutable) 和不变 (immutable) 对象的区别？

javascript中的原始值（undefined、null、布尔值、数字和字符串）与对象（包括数组和函数）有着根本区别。原始值是不可更改的：任何方法都无法更改（或"突变"）一个原始值。对数字和布尔值来说显然如此——改变数字的值本身就说不通，而对字符串来说就不那么明显了，因为字符串看起来像由字符组成的数组，我们期望可以通过指定索引来假改字符串中的字符。实际上，javascript是禁止这样做的。字符串中所有的方法看上去返回了一个修改后的字符串，实际上返回的是一个新的字符串值。

区别

- 可变性：对象和原始值不同，首先，它们是可变的–它们的值是可修改的
- 值的比较：对象的比较并非值的比较：即使两个对象包含同样的属性及相同的值，它们也是不相等的。各个索引元素相等的两个数组也不相等。

不变性 (immutability) 有哪些优缺点？

优点：

- 因为不能修改一个不变对象的状态，所以可以避免由此引起的不必要的程序错误；一个不变的对象要比一个可变的对象更加容易维护。
- 因为没有任何一个线程能够修改不变对象的内部状态，一个不变对象自动就是线程安全的，这样可以省掉处理同步化的开销。一个不变对象可以自由地被不同的客户端共享。

缺点：

- 一旦需要修改一个不变对象的状态，就只好创建一个新的同类对象。在需要频繁修改不变对象的环境里，会有大量的不变对象作为中间结果被创建出来，这是一种资源上的浪费。

如何用你自己的代码来实现不变性 (immutability)？

可以使用**const 修饰变量不可变**

参考： Immutable 在 JavaScript 中的应用

你会使用怎样的语言结构来遍历对象属性 (object properties) 和数组内容？

请解释同步 (synchronous) 和异步 (asynchronous) 函数的区别。

同步式

当计算机调度线程进行I/O操作命令后，由于文件的读写或者网络通信需要较长的操作时间，操作系统为了充分利用cpu，此时会暂停到当前的I/O线程对CPU的控制（故又称同步式为阻塞式I/O），把cup资源然给其他的线程资源，当I/O线程完成了操作时，此时操作系统会恢复此时的I/O线程，从而当前I/O线程重新获得了cup的的控制权，继续完成其他操作。

异步式

异步式IO又称非阻塞式I/O，异步式与同步式不同的是，当线程进行IO操作时，操作系统并不是暂停当前的线程操作，而是执行完I/O指令后，操作系统继续让当前线程执行下一条指令，当I/O操作完成后，会通过事件（event)通知I/O线程，而线程在接收到通知后，会处理响应事件。

什么是事件循环 (event loop)？

主线程从“任务队列”中读取事件，这个过程是循环不断的，所以整个的这种运行机制又称为Event Loop（事件循环）。

请问调用栈 (call stack) 和任务队列 (task queue) 的区别是什么？

所谓“回调函数”（callback），就是那些会被主线程挂起来的代码。异步任务必须指定回调函数，当异步任务从“任务队列”回到执行栈，回调函数就会执行。

“任务队列”是一个先进先出的数据结构，排在前面的事件，优先返回主线程。主线程的读取过程基本上是自动的，只要执行栈一清空，“任务队列”上第一位的事件就自动返回主线程。但是，由于存在后文提到的“定时器”功能，主线程要检查一下执行时间，某些事件必须要在规定的时间返回主线程。

解释 function foo() {} 与 var foo = function() {} 用法的区别？

第一个未函数声明，第二个为函数定义表达式（函数定义表达式foo，变量声明提前单赋值并未提前）

JS面试题

准备的清粥小菜：

- 面试题一
- 面试题二
- 面试题三
- js答案（本套题）

JS面试题

CSS面试题 HTML面试题