# JAVASCRIPT 基础知识理解和运用

## 

## 

## 

## 

## 

## 

## 事件委托/ 事件代理解释

将一个事件侦听器实际绑定到父容器<ul>上，从而免去将事件监听器绑定到每一个的某些元素li上，以便在用户与元素交互时执行某些操作，比每个元素单独绑定事件的处理程序更高效。

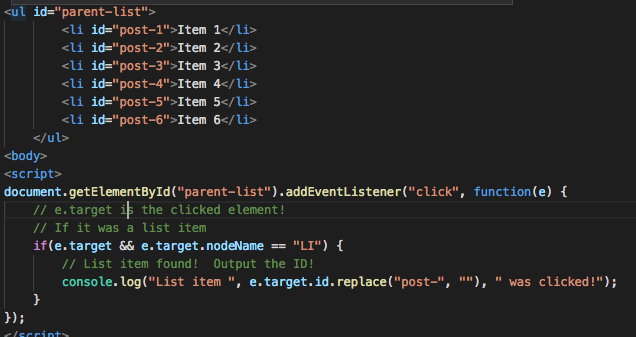
若有元素被点击且被点击目标元素符合要求，执行时间

好处：

1、提高JavaScript性能。事件委托可以显著的提高事件的处理速度，减少内存的占用。

2、动态的添加DOM元素，不需要因为元素的改动而修改事件绑定。

Event delegation is when you bind an event listener to a parent (or ancestor) element rather than the element(s) you are particularly interested in. When the event is triggered you can check the event target to make sure it was actually the triggered on the element of interest.



2. JavaScript的解析与执行过程

key:预处理（函数、变量覆盖）、执行这两项的顺序

1）全局预处理--扫描函数声明，再扫描变量声明

example：



2）全局执行过程



example:

3）. 函数预处理阶段

* 每调用一次，产生一个词法环境（或执行上下文Execution Context）；
* 先传入函数的参数，若参数值为空，初始化undefined；
* 然后是内部函数声明，若发生命名冲突，会覆盖；
* 接着就是内部var变量声明，若发生命名冲突，会忽略；

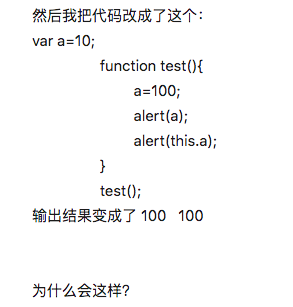
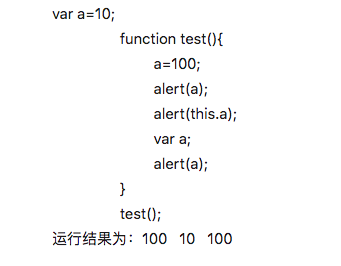




4）. 函数执行阶段

* 给预处理阶段的成员赋值；
* 无var声明的变量，会成为全局成员

## 3. This keyword

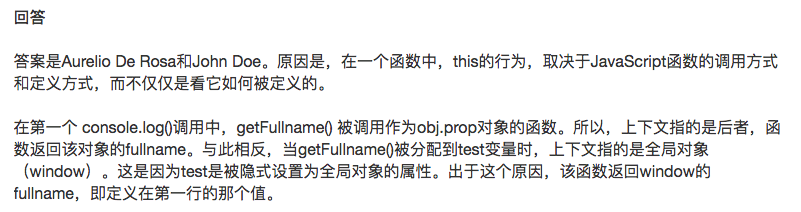


Answer: 第一个程序，经过变量提升（声明前置），var a被提升到第一行，a=100定义的是局部变量a, this指向window； 第二个程序，没有局部变量a, a和this.a都为100

延伸：key(构造函数实例化对象)



错题：



## 4.继承与原型链

解释：当谈到继承时，JavaScript 只有一种结构：对象。每个对象都有一个私有属性（称之为 [[Prototype]]），它指向它的原型对象（**prototype**）。该 prototype 对象又具有一个自己的 prototype ，层层向上直到一个对象的原型为 null。根据定义，null 没有原型，并作为这个**原型链**中的最后一个环节。

Tbs

## 5. CommandJS, AMD TBS AFTER NODE.JS

前端模块规范有三种：CommonJs,AMD和CMD。

CommonJs用在服务器端，AMD和CMD用在浏览器环境

AMD 是 RequireJS 在推广过程中对模块定义的规范化产出。

CMD 是 SeaJS 在推广过程中对模块定义的规范化产出。

AMD:提前执行（异步加载：依赖先执行）+延迟执行

CMD:延迟执行（运行到需加载，根据顺序执行）

<https://www.jianshu.com/p/d67bc79976e6>

## 6. 1) Explain why the following doesn't work as an IIFE: function foo(){ }();. (立即执行函数)

## 2) What needs to be changed to properly make it an IIFE?

## Answer: 1) function foo(){ }()中 function foo(){}是函数声明， 以上代码等于function foo(){ /\* code \*/ } (); // SyntaxError: Unexpected token )

## 2）（function foo(){ })括号里面是函数表达式， need to wrap the anonymous function with parenthesis, so the Javascript parser treats our anonymous function as a *function expression*.

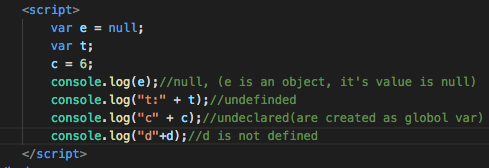
## A function expression is when you assign a function to a variable or property of an object. Anything that is a Javascript expression, including function expression, returns a value. 所以想调用函数，后加（）可运行

## 7. 1）javascript中的null, undefined和undeclared（空值，初始化，声明）

## 2）How would you go about checking for any of these states?

Answer:

1）



细探：**本质区别:**

* null表示一个值**被定义了**，定义为"空值"
* undefined 表示根本**不存在定义**

##### **用法区别:**

* null的用法:
  + 意在保存对象的变量还没有真正保存对象，就应该明确地让该变量保存null值。
  + 对象原型链的终点
  + 作为函数的参数，表示该函数的参数不是对象
  + ***与' '和" "不等价,***这两个指的是字符串为空
* undefined的用法:
  + 变量被声明但没有被赋值. var a;
  + 调用函数时，应该提供的参数没有提供，该参数等于undefined.

function f(x){  
 alert(x);  
}  
f(); //x undefined

* 对象没有赋值的属性，该属性的值为undefined.

var person=new object();  
person.age //age undefined

* 函数没有返回值时，默认返回undefined

var x=f();  
x //undefined

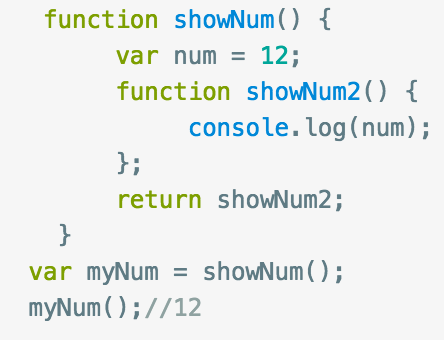
2）检测：null: 使用typeof类型判断结果为Object

Undefined: 使用typeof类型判断会得到字符串类型的'undefined'

## 8. 1）闭包是什么 2）闭包怎么用 3）为何用闭包

1）闭包：闭包（Closure）是词法闭包（Lexical Closure）的简称，是引用了自由变量的函数。这个被引用的自由变量将和这个函数一同存在，即使已经离开了创造它的环境也不例外。

2）**在函数内部再定义一个内部函数，并将这个内部函数当做返回值，这样我们就可以获取函数的局部变量。**

****



详解：第一个函数中，在for循环声明的五个TimeOut Callback函数都有对变量i的**引用**，而不是拷贝。因为5个timeout函数都涉及到**延迟执行**的情况，所以当主线程执行完后（end被打印时），timeout这些回调依次执行（队列：FIFO），此时i的值已经为5了，

第二个函数，对原函数进行了IIFE, 对函数 声明即执行，利用了JavaSrcipt中函数基本类型变量传值，都是值的拷贝，而不是值的引用，然后通过在for循环中执行一个闭包函数，建立一个闭包作用域，来保证引用的i值为注册该回调函数时的值。立即即执行。

说到这里就不得不提下JavaScript的变量作用域问题。变量作用域无非就两种：全局作用域和局部作用域。

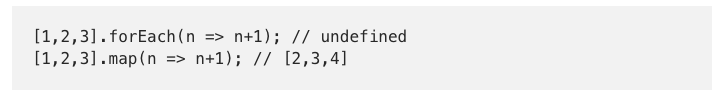
3）

**读取函数内部变量**（函数内部可以直接访问外部变量，但在函数外部无法访问函数内部变量。这也就是Javascript语言特有的“链式作用域”结构（chain scope））

**将创建的变量的值始终保持在内存中，以供本地环境使用**（一般情况下在函数被调用完后不再被引用时，该函数都会被垃圾回收机制（garbage collection））

9. forEach和.map有什么区别？你会选用哪个？

forEach 與 .map 有很大的不同，不過兩個都是屬於 Array 這個物件的方法，forEach 單純像是 for loop 一樣，會把所有遍歷一次，而 .map 會遍歷所有後返回一個新的 Array 物件，forEach 則不會; map 和 filter 都是 immutable methods，也就是说它们只会返回一个新数组，而不会改变原来的那个数组.



何时用forEach: 对原数组进行操作

何时用.map: when you want to transform elements in an array. 因此在确定使用场景的时候，如果我们需要产生一个新的数组，让数组项的值在新数组发生变化，那就用map,

## 10.函数的定义和引用

1）定语函数的方式：

function demo(){

// 它的实质，是将函数主体赋值给了demo变量

}

这个函数主体是：

function(){

// 函数主体部分

}

demo() 函数也可以这样定义：

var demo=function(){

// 函数主体部分

}

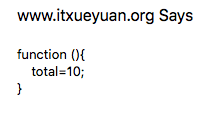
可以这样来调用函数：

demo();

小括号的作用是将demo变量所指向的函数主体（代码块）作为函数执行。

如果不加小括号，函数主体仅仅是一段文本，一段字符串，不会执行。

alert(demo)



## **2）匿名函数的定义**

匿名函数，可以理解为定了一个函数，但是没有将它赋值给变量。定义方式如下：

function([parameters]){

statements;

[return表达式;]

}

各个部分的含义与“ 具名函数 ”（有函数名的函数）相同。

3）**匿名函数的调用**

匿名函数没有函数名，也没有指向它的变量，不能像具名函数一样可以在任何地方调用，匿名函数必需在定义后立即执行。

小括号的作用是将代码块作为函数执行，在小括号内一样可以传递参数，所以，**匿名函数可以使用小括号来调用和执行**

**(function([parameters]){**

**// 函数主体部分**

**})([parameters]);**

**第一个小括号将函数体包围起来，第二个小括号执行函数体，[parameters]是要传递的参数列表。**

**4）匿名函数使用：可和闭包结合，**里面的代码会立即执行。

里面的变量不会和全局变量冲突，一般用来封装一些组件，把需要的接口对外暴露，方便外部调用。

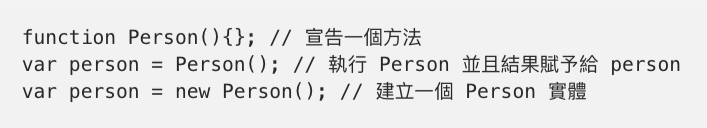
11. Host objects和Native objects的区别：

Native Object: JavaScript语言提供的不依赖于执行宿主的对象，其中一些是内建对象，如：Global、Math；一些是在脚本运行环境中创建来使用的，如：Array、Boolean、Date、Function、Number、Object、RegExp、Error。

Build-in Object: JavaScript语言提供的不依赖于执行宿主的内建对象，如：Global、Math；内建对象都是Native Object。

Host Object: JavaScript语言提供的任何依赖于宿主环境的对象，所有非Native Object的对象都是宿主对象，如：IE中的window，WScript中的wscript实例，任何用户创建的类。

## 12.**Difference between: function Person(){}, var person = Person(), and var person = new Person()?（以上三種的不同）**



這題有個重點就是在於「new」這個字，有「new」的宣告將會建立一個實體來自於 Person，如果 Person 帶有 prototype 則會帶入到 person 的 \_\_proto\_\_ 中，而 Person 方法將變為建構式（constructor）

## 13..call 與 .apply 有何不同

没有本质不同， .call 傳入的是一個一個的參數而 .apply 是丟一個 array最重要的是要知道 .call 與 .apply 作用是什麼，通常是拿來將某個環境丟到一個方法中。对象的方法可以任意指派,而对象本身一直都是没有这方法的,注意是指派,通俗点就是,方法是借给另一个对象的调用去完成任务,原理上是方法执行时上下文对象改变了.

## call, apply作用就是借用别人的方法来调用,就像调用自己的一样.

错题：

## 14.Explain Function.prototype.bind（說明 Function.prototype.bind）

## 每個 Function 都有一個 bind 方法，可以將某個環境丟入此方法中

## fn.bind(Person); //如果是 Person, 則呼叫 fn 時 this 會是 Person 環境

## 15.解释变量提升

https://javascriptweblog.wordpress.com/2010/07/06/function-declarations-vs-function-expressions/

Hoisting 真正发生的是在编译阶段将变量和函数声明放入内存中，但仍然保留在编码中键入的位置。

JavaScript 在执行任何代码段之前，将函数声明放入内存中的优点之一是，这允许你可以在你的代码中使用一个函数，在声明该函数之前。

## 16.document load 事件與 DOMContentLoaded 的不同？

触发的时机不一样，先触发DOMContentLoaded事件，后触发load事件。

详细解释：

DOM文档加载的步骤为：

1.解析HTML结构。

2.加载外部脚本和样式表文件。

3.解析并执行脚本代码。

4.DOM树构建完成。//DOMContentLoaded

5.加载图片等外部文件。

6.页面加载完毕。//load

在第4步，会触发DOMContentLoaded事件；在第6步，触发load事件。

附加代码解释：



Ready事件总是先于load事件触发

## 17. What is the difference between == and ===?（== 與 === 差別）

"==="叫做严格运算符，"=="叫做相等运算符。

## 

相等运算符在比较不同类型的数据时，会先将数据类型进行转化，再用严格运算符进行比较，会出现不严谨的情况，比如：false == '0'

## 18. 解释与javascript相关的同源策略

**概念：**同源策略是客户端脚本(尤其是JavaScript)的重要安全度量标准。其目的是防止某个文档或脚本从多个不同源装载。

**同源指的是**：协议，域名(包括子域名)，端口相同。同源策略是一种安全协议。

**具体表现在JS中**：一段脚本自能读取来自同一来源的窗口的文档和属性。(iframe.contentDocument 等访问错误)

**为什么要有同源限制？**是为了保证用户信息的安全，防止恶意的网站窃取数据。如果没有同源策略，黑客通过iframe伪造一个网银登录界面，在父页面上直接获取iframe里面输入框的值。就能完成窃取账号密码的侵入行为。

同源限制范围：cookie, dom, ajax

规避同源限制： ajax(jsonp, websocket)

\*jsonp:它的基本思想是，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。

\*什么是同源策略通俗解释

同domain（或ip）,同端口，同协议视为同一个域，一个域内的脚本仅仅具有本域内的权限，可以理解为本域脚本只能读写本域内的资源，而无法访问其它域的资源。这种安全限制称为同源策略。

**现代浏览器在安全性和可用性之间选择了一个平衡点。在遵循同源策略的基础上，选择性地为同源策略“开放了后门”。** 例如img script style等标签，都允许垮域引用资源，严格说这都是不符合同源要求的。然而，你也只能是引用这些资源而已，并不能读取这些资源的内容（不信你可以试试：在你自己的域内读取百度logo图片的内容，以读取到二进制数据为准。 你很快发现这是不可能的，你顶多只能判断这张图片是否存在（使用的img标签的onerror属性 ））。

打破同源策略的方法有很多，抛开漏洞不谈，Server端的大数据整合和浏览器的插件往往就充当着这样一个角色。3.如果没有同源策略，普通用户将无密可保。

## 19. Make this work:

## duplicate([1,2,3,4,5]); // [1,2,3,4,5,1,2,3,4,5]

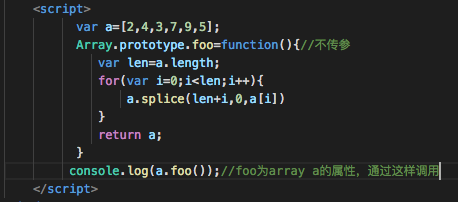
Array.prototype.duplicator = function(){

return this.concat(this);

}

alert([1,2,3,4,5].duplicator());

Or



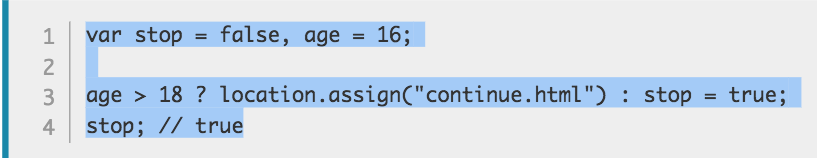
## 20.三元运算符

条件（三元）运算符是 JavaScript 仅有的使用三个操作数的运算符。本运算符经常作为 [if](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/if...else) 语句的简短形式来使用。

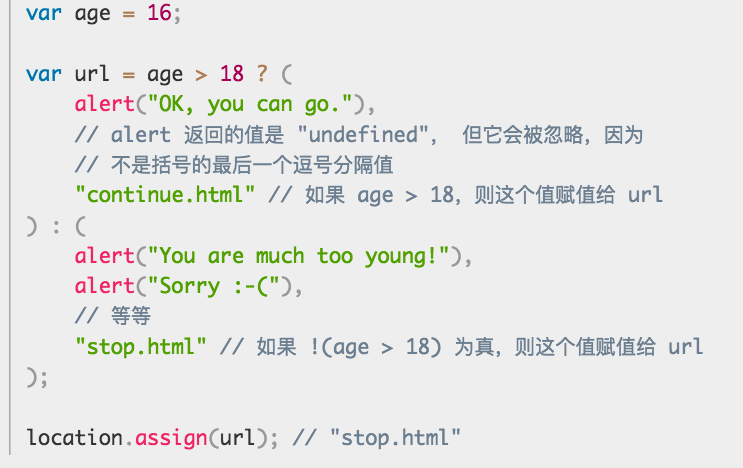
condition ? expr1 : expr2

### 参数： **condition** 计算结果为true或false的表达式；**expr1, expr2** 值可以是任何类型的表达式。

例子：



\*提高：也可以在赋值过程中做多个操作。 如下所示，**括号里的最后一个逗号分隔值**是将被赋值的。



## 21. 什么是js严格模式

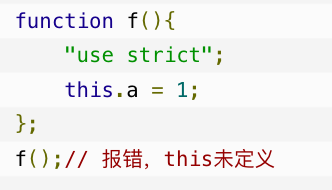
"use strict" 的目的是指定代码在严格条件下执行。\*严格模式下你不能使用未声明的变量。

为什么使用严格模式:

* 消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;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。

严格模式的主要限制：

* 不允许使用未声明的变量。
* 不允许删除变量或对象。
* 不允许删除函数。
* 不允许变量重名。
* 不允许对只读属性赋值。
* val不会在它的外层作用域引入变量
* 禁止this关键字指向全局对象。因此，使用构造函数时，如果忘了加new，this不再指向全局对象，而是报错。例子：



## 22. Create a for loop that iterates up to 100 while outputting "fizz" at multiples of 3, "buzz" at multiples of 5 and "fizzbuzz" at multiples of 3 and 5

## 23.为什么要保留全局作用域，不爱用它

Answer: 因为全局作用域容易被呼叫，很容易污染环境。TB



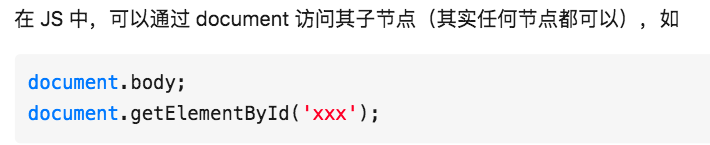
## 24.dom, bom, html, document, window

**总结：DOM 是为了操作文档出现的 API，document 是其的一个对象；**

**BOM 是为了操作浏览器出现的 API，window 是其的一个对象。**

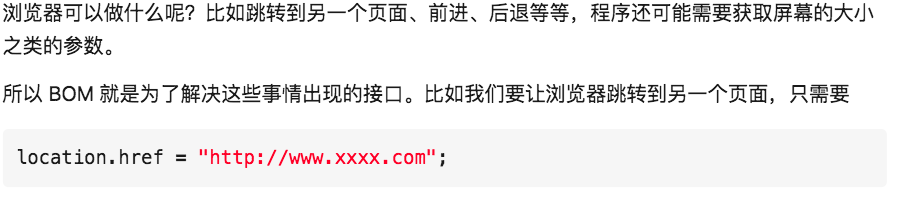
详解：**DOM 全称是 Document Object Model，也就是文档对象模型。**DOM 就是针对 HTML 和 XML 提供的一个API。什么意思？就是说为了能以编程的方法操作这个 HTML 的内容.

当浏览器下载到一个网页，通常是 HTML，这个 HTML 就叫 document（当然，这也是 DOM 树中的一个 node），从上图可以看到，document 通常是整个 DOM 树的根节点。



**BOM是 Browser Object Model，浏览器对象模型。**

BOM 顾名思义其实就是为了控制浏览器的行为而出现的接口。



**window 也是 BOM 的一个对象，**除去编程意义上的“兜底对象”之外，通过这个对象可以获取窗口位置、确定窗口大小、弹出对话框等等。例如我要关闭当前窗口：

window.close();

## 25.Why would you use something like the load event? Does this event have disadvantages? Do you know any alternatives, and why would you use those?

load event tells browser to do something only after **everthing including frames, images, asynchronous JavaScripts** are fully loaded.

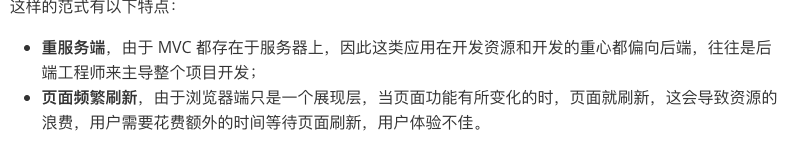
If you want event function to execute before fully loaded frames, images, async scripts, use domcontentloaded instead.

## 26.Explain what a single page app is and how to make one SEO-friendly.

什么是单页应用：

## 

## 传统方式缺点：



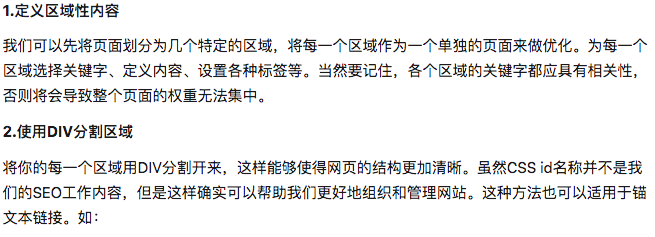
单页应用优缺点：

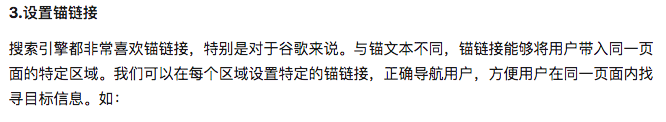
* **无刷新体验**，这个应该是最显著的有点，由于路由分发直接在浏览器端完成，页面是不刷新，对用户的响应非常及时，因此提升了用户体验；
* **完全的前端组件化**，前端开发不再以页面为单位，更多地采用组件化的思想，代码结构和组织方式更加规范化，便于修改和调整；
* **API 共享**，如果你的服务是多端的（浏览器端、Android、iOS、微信等），单页应用的模式便于你在多个端共用 API，可以显著减少服务端的工作量。容易变化的 UI 部分都已经前置到了多端，只受到业务数据模型影响的 API，更容易稳定下来，便于提供鲁棒的服务；
* **组件共享**，在某些对性能体验要求不高的场景，或者产品处于快速试错阶段，借助于一些技术（[Hybrid](https://en.wikipedia.org/wiki/Hybrid)、[React Native](https://facebook.github.io/react-native/)），可以在多端共享组件，便于产品的快速迭代，节约资源。

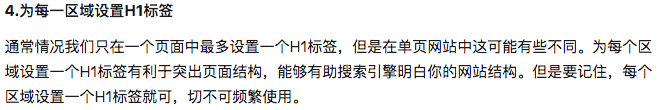
单页应用的优点有时候也是缺点：

* **首次加载大量资源**，要在一个页面上为用户提供产品的所有功能，在这个页面加载的时候，首先要加载大量的静态资源，这个加载时间相对比较长；
* **较高的前端开发门槛**，MVC 前置，对前端工程师的要求提高了，不再是『切切图，画画页面这么简单』；同时工作量也会增加数倍，开发这类应用前端工程师的数量往往多于后端；
* **不利于 SEO**，单页页面，数据在前端渲染，就意味着没有 SEO，或者需要使用变通的方案。

如何seo优化：

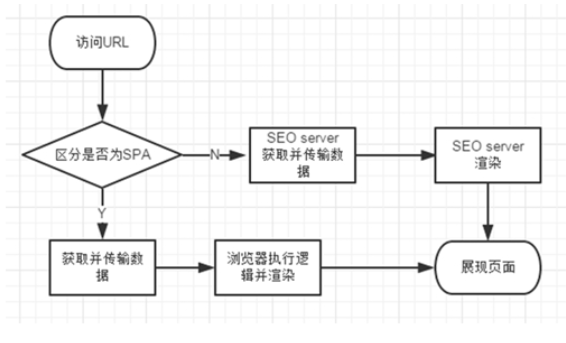






5.　**【方案四：用更高效的方式完成两套页面】实现一套代码两处运行、SEO页面可单独自定义功能、、同一张landing人和蜘蛛没有跳转，没有区别对待、全栈工程师的大量使用、SEO页面永远保持最新版**等等省时省力的需求功能。

我们需要给蜘蛛渲染完成的HTML，那么你的框架需要兼容如下流程的功能。



6.　【方案三：HTML5 history 中的PushState】  
　　还好，开发大大们总是不少奇巧淫技，这是个很”经典”的用法，配合<noscript>这个擦边球标签，既能实现URL的自定义，又能实现还算有效果的内容抓取。蜘蛛、浏览器，两方应对，给蜘蛛不带井号能抓取的URL，给浏览器访问非井号URL时中间做转换，这样的话每张页面都有了可抓取的URL，且依然使用着高逼格的SPA架构。内链可以做了，Sitemap可以做了，适配也轻松了。

16. Promise的使用和优缺点

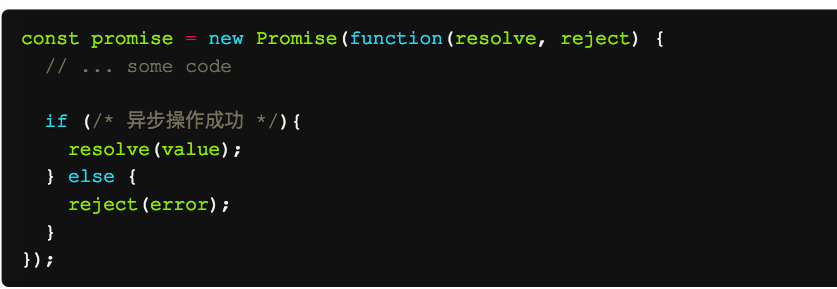
Promise 是异步编程的一种解决方案，比传统的解决方案——回调函数和事件——更合理和更强大。所谓Promise，简单说就是一个容器，里面保存着某个未来才会结束的事件（通常是一个异步操作）的结果。从语法上说，Promise 是一个对象，从它可以获取异步操作的消息。Promise 提供统一的 API，各种异步操作都可以用同样的方法进行处理。

Promise对象有以下两个特点：

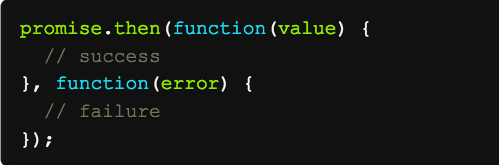
（1）对象的状态不受外界影响。Promise对象代表一个异步操作，有三种状态：pending（进行中）、fulfilled（已成功）和rejected（已失败）。只有异步操作的结果，可以决定当前是哪一种状态，任何其他操作都无法改变这个状态。

（2）一旦状态改变，就不会再变，任何时候都可以得到这个结果。Promise对象的状态改变，只有两种可能：从pending变为fulfilled和从pending变为rejected。只要这两种情况发生，状态就凝固了，不会再变了，会一直保持这个结果，这时就称为 resolved（已定型）。如果改变已经发生了，你再对Promise对象添加回调函数，也会立即得到这个结果。这与事件（Event）完全不同，事件的特点是，如果你错过了它，再去监听，是得不到结果的。

优点：有了Promise对象，就可以将异步操作以同步操作的流程表达出来，避免了层层嵌套的回调函数。



resolve函数的作用是，将Promise对象的状态从“未完成”变为“成功”（即从 pending 变为 resolved），在异步操作成功时调用，并将异步操作的结果，作为参数传递出去；reject函数的作用是，将Promise对象的状态从“未完成”变为“失败”（即从 pending 变为 rejected），在异步操作失败时调用，并将异步操作报出的错误，作为参数传递出去。



Promise实例生成以后，可以用then方法分别指定resolved状态和rejected状态的回调函数。

好处：，当需要在多个同时发出的异步调用之后再执行某个单独的回调函数时，事情就会变得越来越复杂。使用promise模式，只需要简单地传入一个promise数组给call()方法，它就会同时执行数组中每项promises。TBS

<https://www.zhihu.com/question/20761027>

原生

<http://ourjs.com/detail/5%E4%B8%AA%E7%BB%8F%E5%85%B8%E7%9A%84%E5%89%8D%E7%AB%AF%E9%9D%A2%E8%AF%95%E9%97%AE%E9%A2%98>

# 17. 冒泡与捕获

自己的话：冒泡：现象：点击元素时候，会调用它的父级元素绑定的事件，这是因为当点击事件发生在子元素时，这个事件会冒泡到父级元素级其他祖级元素。

想组织冒泡：为该元素加上e.stopPropagation()，则不会点击事件不会冒泡到父级元素，父级元素绑定的事件不会被调用

event.target与this (=event.currentTarget) 的区别：event.target指的是被点击的元素(e:p)，后者指的是被绑定事件的元素（e:div）

在JS中，绑定的事件默认的执行时间是在冒泡阶段执行，而非在捕获阶段（重要），这也是为什么当父类和子类都绑定了某个事件，会先调用子类绑定的事件，后调用父类的事件。

改变默认冒泡：

1. 我们可以通过绑定事件时，指定事件执行时间是在冒泡阶段还是捕获阶段。

obj.addEventListener(event,function(){},bool)

bool:false，代表冒泡阶段执行

Bool:true，代表捕获阶段执行

2. **阻止冒泡**

w3c的方法是e.stopPropagation()，IE则是使用e.cancelBubble = true；

# 18. 何时会用document.write

document.write()方法可以用在两个方面：

1.页面载入过程中，用脚本加入新的页面内容。

2.用延时脚本创建本窗口或新窗口的内容。

这个方法不是很好，原因有二：

1. 对第一个方面，比较麻烦
2. 对第二个方面，没做到结构功能分离

应用实例：百度和第三方的广告

\*附加题：什么情况下document.write会覆盖页面及覆盖原理解释

会覆盖：

不会覆盖：

原理：window.onload事件是在文档内容完全加载完毕再去执行事件处理函数，当然文档流已经关闭了，这个时候执行doucment.writ()函数会自动调用document.open()函数创建一个新的文档流，并写入新的内容，再通过浏览器展现，这样就会覆盖原来的内容。

原理 主要在于是否会重新加载文档。重新加载文档的时机是什么：调用函数

# 19. Ajax的优缺点分别是什么

什么是Ajax和JSON，它们的优缺点。 Ajax是异步JavaScript和XML，用于在Web页面中实现异步数据交互。

\*\*优点：\*\* 可以使得页面不重载全部内容的情况下加载局部内容，降低数据传输量 避免用户不断刷新或者跳转页面，提高用户体验

\*\*缺点：\*\* 对搜索引擎不友好（ 要实现ajax下的前后退功能成本较大 可能造成请求数的增加 tbs

ajax有XMLHttpRequest同源策略：禁止向不同源的地址发起HTTP， 所以存在请求跨域问题限制

# 什么是jsonp：

与ajax共同工作：Ajax与JSONP有着本质上的不同。Ajax的核心是通过XMLHttpRequest获取数据，而JSONP的核心则是动态添加<script>标签来调用服务器提供的js文件。 所以jsonp可被用来解决一般用来解决Ajax跨域的问题。

JSONP 的工作原理: 动态创建一个script标签，利用script标签src属性访问没有限制，实现跨域。web客户端通过与调用脚本一样的方式来调用跨域服务器上动态生成的js格式文件(后缀.json)，服务器之所以要动态生成json文件目的把客户端需要的数据装入进去。允许用户传递一个callback参数给服务端，然后服务端返回数据时会将这个callback参数作为函数名来包裹住json数据 这样客户端就可以随意定制自己的函数来自动处理返回数据

# 20. 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？TBS JS 遍历

for(var key of Object.keys(key) -------键的值

for(var value of Object.values(values)) -------对象的值

for(var key&value of Object.entries(key&value)------对象的属性键和值

# 原生JS **forEach()和map()**遍历

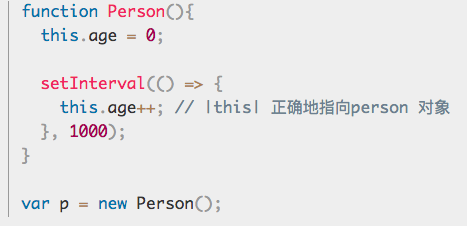
Map（） 理论上这个方法是没有返回值的，仅仅是遍历数组中的每一项，不对原来数组进行修改；但是可以自己通过数组的索引来修改原来的数组；

forEach() map的回调函数中支持return返回值；return的是啥，相当于把数组中的这一项变为啥（并不影响原来的数组，只是相当于把原数组克隆一份，把克隆的这一份的数组中的对应项改变了）

# 21. 箭头函数的用法，与其他函数区别

# **箭头函数表达式**的语法比[函数表达式](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/function)更短，并且不绑定自己的[this](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/this)，[arguments](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Functions/arguments)，[super](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/super)或 [new.target](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/new.target)。这些函数表达式最适合用于非方法函数，并且它们不能用作构造函数。

解释：箭头函数不会创建自己的this；它使用封闭执行上下文的this值



不用箭头函数时：



# 为何不能用箭头函数定义构造函数：

箭头函数的this指向上下文的值，构造函数构造函数 Car 的上下文就是新创建的对象

e：

# 22. 时间循环，调用栈和任务队列

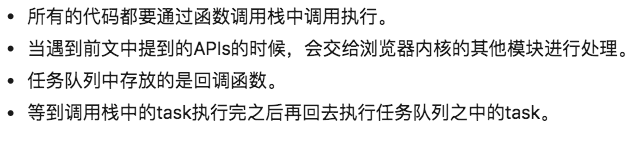
答案：

**事件循环**，我们需要记住以下几点：

* 事件队列严格按照时间先后顺序将任务压入执行栈执行；
* 当执行栈为空时，浏览器会一直不停的检查事件队列，如果不为空，则取出第一个任务；
* 在每一个任务结束之后，浏览器会对页面进行渲染；

调用栈和任务队列：

Javascript有一个main thread 主进程和call-stack（一个调用堆栈），在对一个调用堆栈中的task处理的时候，其他的都要等着。当在执行过程中遇到一些类似于setTimeout等异步操作的时候，会交给浏览器的其他模块(以webkit为例，是webcore模块)进行处理，当到达setTimeout指定的延时执行的时间之后，task(回调函数)会放入到任务队列之中。一般不同的异步任务的回调函数会放入不同的任务队列之中。等到调用栈中所有task执行完毕之后，接着去执行任务队列之中的task(回调函数)。

总结：

详解：

Js的语言特点是单线程，单线程就意味着，所有任务需要排队，前一个任务结束，才会执行后一个任务

如果排队是因为计算量大，CPU忙不过来，倒也算了，但是很多时候CPU是闲着的，因为IO设备（输入输出设备）很慢（比如Ajax操作从网络读取数据），不得不等着结果出来，再往下执行。

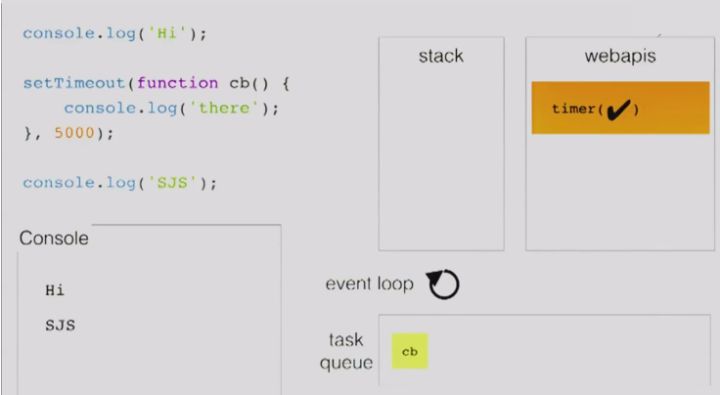
JavaScript语言的设计者意识到，这时主线程完全可以不管IO设备，挂起处于等待中的任务，先运行排在后面的任务。等到IO设备返回了结果，再回过头，把挂起的任务继续执行下去。

于是所有任务分为同步和异步：

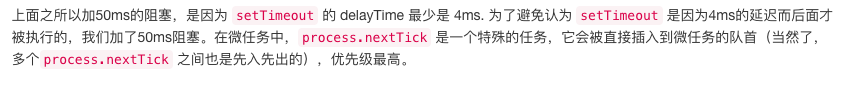
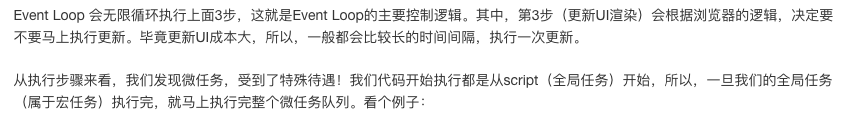
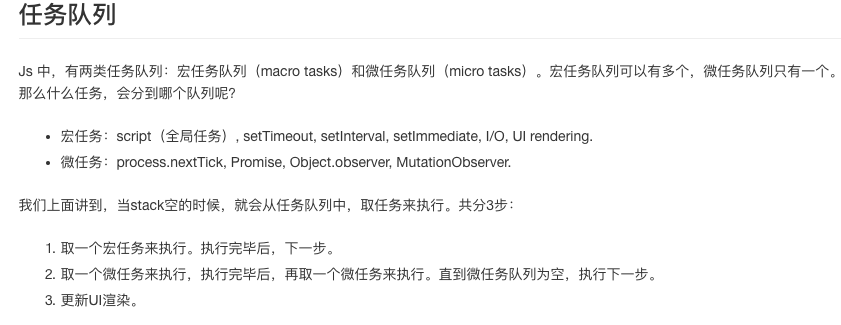
同步任务在主主线程上执行，进入调用栈，先进后出；

对异步任务，将延时执行的函数交给浏览器的timer模块（webcore一共提供DOM Binding、network、timer模块来处理底层实现）进行处理，处理之后，异步任务的回调函数进入任务队列，调用栈中的task执行完毕之后，执行引擎会接着看执行任务队列中是否有需要执行的回调函数。这里的函数被执行引擎添加到调用栈中，接着执行里面的代码，再出栈。

https://zhuanlan.zhihu.com/p/26229293



任务队列扩展：



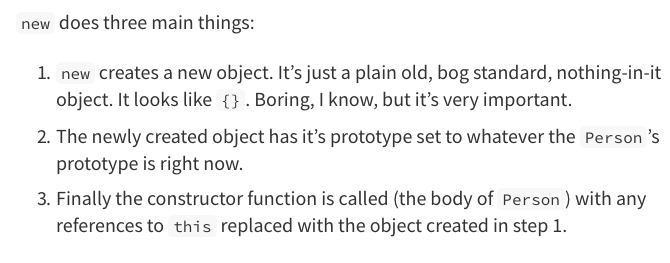
# 23.请指出以下代码的区别：function Person(){}、var person = Person()、var person = new Person()？

1)第一个是函数声明，创建一个新函数person

2)this --global object

3）第二个是用构造函数为对象Person创建新实例person，person的原型是Person,拥有其原型的所有方法

In var person=new person()



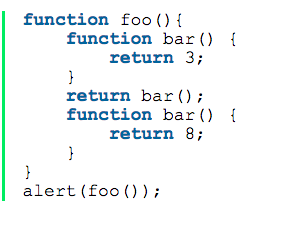
例子：

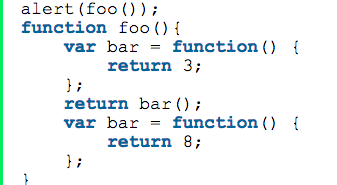
https://amyetheredge.com/interview/js/7.html

# 24.函数声明和函数表达式与变量提升问题

https://javascriptweblog.wordpress.com/2010/07/06/function-declarations-vs-function-expressions/

知识点：变量提升/函数声明提升/函数表达式中的变量部分提升，表达式部分不提升

e：函数提升//8

函数表达式中变量部分提升，函数表达式不提升//3

## 

## 