



## 常见js笔试面试题（持续更新）

原 javascript

澹台宇鹏 2017年11月06日发布

# js基础知识

## 1. javascript typeof返会的数据类型有哪些

object,string,undefined,number,function,boolean

基本数据类型：

string,number,boolean,undefined,null

## 2. 列举三种强制类型转换和两种隐式类型转换

parseInt(),parseFloat(),Number()

==,!!

## 3. 数组相关集合

### 3.1. 创建数组方法

```
var array = new Array()  
var array = []
```

Array.of(1,2) //[1,2]

这是es6新增的一个Array方法，创建一个具有可变数量参数的新数组实例，而不考虑参数的数量或类型。

（感谢 haru 的宝贵建议）

## 4. 判断是否为数组的方法

- console.log(arr instanceof Array)
- console.log(arr.constructor === Array)
- console.log(Array.isArray(arr))

## 5. pop(),push(),unshift(),shift()

- pop()尾部删除
- push()尾部插入
- unshift()头部插入
- shift()头部删除

## 6. DOM0 DOM2

dom0级



首页



问答



专栏



讲堂



更多

- 不支持添加多个事件，后面的会覆盖前面的
- 无法取消

```
var btn = document.getElementById("button");
btn.onclick = function(){
    alert(1);
}
btn.onclick = function(){
    alert(2);
} //只弹出2
```

dom2

- 可以添加多个事件
- 不兼容低版本IE
- 支持事件冒泡，事件捕获

```
var btn = document.getElementById("button");
btn.addEventListener("click",function(){
    alert("1");
})
btn.addEventListener("click",function(){
    alert("2");
}) //先弹出1，再弹出2
```

## 7. IE和DOM事件流的区别

- 执行顺序不一样
- 参数不一样 低版本ie没有回调函数，只能进行冒泡
- 第一个参数是否加"on",低版本IE不支持addEventListener(),支持attachEvent,第一个参数需要加"on"
- this指向问题，IE指向windows,不指向触发的函数

## 8. IE标准下有哪些兼容性写法

```
var ev = ev || window.event
document.documentElement.clientWidth || document.body.clientWidth
var target = ev.srcElement || ev.target
```

## 9. call apply bind

改变this的指向，  
其中call的写法

```
function add(a,b)
{
    alert(a+b);
}
function sub(a,b)
{
    alert(a-b);
}
add.call(sub,3,1);
```

这个例子中的意思就是用 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。

apply写法

```
function add(a,b)
{
    alert(a+b);
}
function sub(a,b)
```



首页



问答



专栏



讲堂



更多

```
}  
add.apply(sub,[4,2]);
```

不同就在于第二个参数，apply写成数组

bind写法

```
function add(a,b)  
{  
  
alert(a+b);  
  
}  
function sub(a,b)  
{  
  
alert(a-b);  
  
}  
add.bind(sub,[4,2])();
```

bind是返回了一个改变上下文的一个函数，可以稍后调用，而apply，call是立即执行函数

## 10. b继承a的方法（js面向对象复习）

- 原型链继承
- 构造函数继承
- 实例继承
- 组合继承
- 拷贝继承
- 寄生组合继承

## 11. 如何阻止事件冒泡和默认事件

- cancelBubble(IE),
- return false,
- event.preventDefault,
- event.stopPropagation()

## 12. 添加 删除 替换 插入到某个接点的方法

- obj.appendChild()
- obj.insertBefore()
- obj.replace()
- obj.remove()

## 13. window.onload和\$(document).ready的区别

- window.onload只能出现一次，\$(document).ready能出现多次
- window.onload需要等所有文件都加载完才开始加载，\$(document).ready只需等文档结构加载完了就开始加载

## 14. == 和 === 区别



首页



问答



专栏



讲堂



更多

前者会自动转换类型  
后者不会

## 15. javascript的同源策略（跨域问题）

跨域是什么：实际上就是一个网站不能执行其他网站上的网址，是由浏览器同源策略造成的，是浏览器对js施加的安全限制

所谓同源，实际上是指域名，协议，端口都相同

也就是说当，域名或者协议，或者端口不同的时候，就是跨域，

### 15.1. 解决方法：

jsonp

json with padding,是一种json的一种使用模式

产生的原因，ajax不支持跨域，由于浏览器的同源策略，但是script的src支持跨域

主要的原理是动态创建一个script标签的，通过src调用服务器提供的js脚本，该脚本的内容是一个函数调用，该函数在本地js文件中进行定义，其中的参数就是，本地函数请求的数据，也就是服务器所将返回的数据

与ajax的不同，ajax是通过xhr获取非本页面的数据内容，而jsonp获取的是服务器提供js脚本

代理

- 例如www.123.com/index.html需要调用
- www.456.com/server.php，可以写一个接口
- www.123.com/server.php，由这个接口在后端去调用
- www.456.com/server.php并拿到返回值，然后再返回给 index.html，这就是一个代理的模式。相当于绕过了浏览器端，自然就不存在跨域问题。

PHP端修改header（XHR2方式）

在php接口脚本中加入以下两句即可：

```
header('Access-Control-Allow-Origin: *');//允许所有来源访问
```

```
header('Access-Control-Allow-Method: POST,GET');//允许访问的方式
```

## 16. javascript是一种什么样的语言

- 解释性脚本语言，代码不进行预编译
- 主要用来向HTML页面添加交互行为
- 可以直接嵌入HTML页面，但单独写成js文件有利于结构和行为的分离
- 跨平台性，在绝大多数浏览器支持下，可以在多种平台下运行，linux,windows

## 17. javascript基本数据类型和引用数据类型

基本类型 undefind null number string boolean

- 基本类型的值是不能改变的
- 基本类型不能添加属性和方法
- 基本类型的比较是值的比较
- 基本类型变量存放在栈区（栈内存）
- 也就是说基本类型在赋值操作后，两个变量是相互不受影响的。

引用类型 object Function Array



首页



问答



专栏



讲堂



更多

- 引用类型可以添加属性和方法，属性方法内又可以添加基本类型
- 引用类型的值是可变的
- 引用类型的值同时保存在栈内存和堆内存里的对象，准确地说，引用类型的存储需要内存的栈区和堆区（堆区是指内存里的堆内存）共同完成，栈区内存保存变量标识符和指向堆内存中该对象的指针，
- 引用类型的比较是引用的比较 引用类型时按引用访问的，换句话说就是比较两个对象的堆内存中的地址是否相同，那很明显，person1和person2在堆内存中地址是不同的
- 引用类型的赋值其实是对象保存在栈区地址指针的赋值，因此两个变量指向同一个对象，任何的操作都会相互影响

## 18. js原生不要与jq搞混

- document.getElementById("ID").value

获取值的时候原生不是方法，不带括号

- 获取所有checkbox

```
var boxs =document.getElementsByTagName("input");
var boxArray = [];
var len = boxs.length;
while(len--){
    if(boxs[len].type == 'checkbox'){
        boxArray.push(boxs[len]);
    }
}
```

- 设置div html内容以及设置样式

```
var dom = document.getElementById("ID");
dom.innerHTML = "xxx"
dom.style.color="#000"
```

## 19. DOM,BOM

javascript由ECMAScript,DOM,BOM三部分组成，

- ECMAScript也是一种语言，也就是对规定的语法，操作，关键字，语句等的一个描述，javascript实现了ECMAScript
- DOM是文档对象模型，包括了获取元素，修改样式，操作元素三方面内容，也是我们进行最多的操作，有很多兼容性写法
- BOM是浏览器对象模型，包括浏览器的一些操作，window.onload,window.open等还有浏览器事件，监听窗口的改变onresize,监听滚动事件onscroll等

## 20. null和undefind的区别

- null是表示一个空的对象，转为数值为0，undefind表示一个空的原始值，转为数值为NAN
- undefind指本该有一个值，但却并无定义，null表示没有对象，不应该有值

## 21. XML和JSON的区别

- JSON相对于XML来讲传递速度更快，因为光看代码量就能看出
- JSON与js的交互更容易，解析更方便

## 22. 实现多个标签之间的通信

调用localStorage,cookies等本地存储进行存储相关信息

三者的共同点：都保存在浏览器。

三者的区别：



首页



问答



专栏



讲堂



更多

### 与服务器的交互

- cookie数据始终在同源的http请求中携带（即使不需要），即cookie在浏览器和服务器间来回传递。
- 而sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。cookie数据还有路径（path）的概念，可以限制cookie只属于某个路径下。

### 存储大小限制也不同，

- cookie数据不能超过4k，同时因为每次http请求都会携带cookie，所以cookie只适合保存很小的数据，如会话标识。
- sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。

### 数据有效期不同，

- sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；
- localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；
- cookie只在设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭。

### 作用域不同，

- sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；
- localStorage 在所有同源窗口中都是共享的；
- cookie也是在所有同源窗口中都是共享的。

## 23. 哪些操作会造成内存泄露

内存泄露指任何对象在不再拥有或不再需要它之后依然存在

- setTimeout第一个参数是字符串而不是函数的时候就会造成内存泄露
- 闭包
- 控制台日志
- 循环（两个对象彼此引用且彼此保留）

## 24. js垃圾回收方式

- 标记清除：这是js最常用的垃圾回收方法，当一个变量进入执行环境时，例如函数中声明一个变量，将其标记为进入环境，当变量离开环境时，（函数执行结束），标记为离开环境
- 引用计数：跟踪记录每个值被引用的次数，声明一个变量，并将引用 类型赋值给这个变量，则这个值的引用次数+1，当变量的值变成了另一个，则这个值的引用次数-1，当值的引用次数为0的时候，就回收

## 25. 闭包

- 函数嵌套函数
- 子级函数调用父级函数的参数或变量

### 经典闭包

```
function outer(){  
  var a = 1;  
  function inner(){  
    alert(a);  
  }  
  return inner  
}
```



首页



问答



专栏



讲堂



更多

点击li返回li下标

```
<ul id="test">
  <li>1</li>
  <li>2</li>
  <li>3</li>
</ul>

<script>
  var oUl = document.getElementById("test");
  var oLi = oUl.getElementsByTagName("li");
  for(var i=0;i<oLi.length;i++){
    oLi[i].index = i;
    oLi[i].onclick = function(){
      alert(this.index);
    }
  }
</script>

<!-- 闭包 -->
<script>
  var oUl = document.getElementById("test");
  var oLi = oUl.getElementsByTagName("li");
  for(var i=0;i<oLi.length;i++){
    oLi[i].index = i;
    oLi[i].onclick = (function(a){
      return function(){
        alert a;
      }
    })(i)
  }
</script>
```

## 26. this指向问题

普通函数调用，指向windows

```
window.value=1;
function getValue(){
  console.log(this.value);
}
getValue();//输出1，此时的this指向window
```

对象的方法调用，指向对象

```
var Obj={
  value:2,
  getValue:function(){
    console.log(this.value);//输出2,this指向Obj
  }
}
```

构造器方法调用，指向构造函数实例出来的对象

```
function main(val){
  this.value=val;
}
main.prototype.getValue=function(){
  console.log(this.value);
}

var fun=new main(3);
fun.getValue();
fun.value;//输出3， this指向main的实例对象fun
```

call,apply,bind可以自定义this指向第一个参数

```
function showValue(){
  console.log(this.value);
}
var obj={
  value:4
}
showValue.call(obj)//输出4， this指向了obj对象
```

```
function showValue(){
```

```
var obj={
  value:4
}
var showValue2=showValue.bind(obj);
showValue2();//输出4，this指向了obj对象
```

## 27. 高阶函数

- 函数作为参数传递，
- 函数作为返回值输出

## 28. new操作符到底干了什么

- 创建一个新对象
- 将构造函数的作用域赋值给新对象（所以this指向了这个新对象）
- 执行构造函数的代码（为这个新对象添加属性）
- 返回新对象

## 29. js严格模式

"use strict"

消除js一些不合理的用法

消除代码运行的一些不安全之处

增加运行速度

为未来新版本js做铺垫

- 变量必须声明
- 对象不能出现重复属性名
- arguments改变，不会影响函数参数
- eval，arguments变为关键字，不能作为变量名
- 不允许使用with
- 不用call，apply，bind改变this指向，一般函数调用指向null

## 30. 事件代理事件委托

- 原理是使用dom的冒泡，将事件绑定到父元素上，让父元素进行监听，提高性能

## 31.什么是版本控制，

版本控制是一种记录一个或若干文件内容变化，以便将来查阅修改以及更新。

## 32.ajax请求

ajax请求四步

- 创建一个xhr对象 var xhr = new XMLHttpRequest()
- 判断就绪状态为4时执行代码

```
xhr.onreadystatechange = function(){
  if(xhr.readyState == 4){
    console.log(responseText);
  }
}
```



首页



问答



专栏



讲堂



更多



- 创建请求 xhr.open('get','url',true)
- 发送请求 xhr.send(null)

## 33.在浏览器中输入URL到整个页面显示在用户面前时这个过程中到底发生了什么

- 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：

首先根据url中的域名，在远程服务器中查询对应

## 34.ajax和json

ajax用于web页面中实现异步数据交互，实现页面局部内容刷新

- 优点：能够进行内容局部加载刷新，减少带宽，避免用户不断刷新以及页面跳转，提高用户体验
- 缺点：对搜索引擎不友好；浏览器不支持ajax的后退；

json是一种请求轻量级的数据交互格式

- 优点：轻量级，编译人的阅读理解，便于机器解析

## 35.http考点

### 常用的HTTP方法有哪些

GET:

POST:

PUT:

DELETE:

### GET与POST方法的区别

- get主要是从服务器获取资源，post主要是像服务器发送数据
- get传输数据通过url请求，利用k=v的形式放在url后面，用?连接，多个用&连接而post是存放在，ajax中的data中的，get传输的过程使用户可见的，而post是对用户不可见的。
- get传输的数据量小，以为受url的长度限制，但是效率高，post能上传的数据量大
- post较get更安全一些
- get方式传递的中文字符可能会乱码，post支持标准字符集，可以正确传递中文字符

### http请求报文与响应报文格式

请求报文包含三部分：

- 请求行：包含请求方法、URI、http版本信息

请求报文包含三部分：



首页



问答



专栏



讲堂



更多

响应报文包含三部分：

- 状态行：包含HTTP版本、状态码、状态码的原因短语
- 响应首部字段
- 响应内容实体

## http状态码

- 100-199：成功接收请求，但需要进行下一步请求
- 200-299：成功接收请求，并完成整个处理过程
- 300-399：为完成全部请求，客户需进一步细化需求
- 400-499：客户端请求有错误，包括语法错误或不能正常执行
- 500-599：服务器端出现错误

## http缺点与https

- 通信使用明文不加密，内容可能被窃听
- 不验证通信方身份，可能遭到伪装
- 无法验证报文完整性，可能被篡改

https就是加上加密处理（一般是SSL安全通信线路）+认证+完整性保护

常用：

- 200 正常，表示一切正常，返会的是正常请求结果
- 302/307 临时重定向，表示请求的文档，已被临时移动到别处
- 304 未修改，调用缓存的数据
- 403 服务器拒绝客户请求
- 404 服务器不存在客户想要找的资源
- 500 服务器内部错误

## 36.数组去重的一种相对好理解的方法

利用indexOf方法的去重

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。

```
var arr = [1,1,2,3,4,2,6,4,5,7];
var nArr = [];
function removeItem(arr){
  for(var i=0;i<arr.length;i++){
    if(nArr.indexOf(arr[i])!=-1){
      nArr.push(arr[i]);
    }
  }
  return nArr;
}
console.log(removeItem(arr));
```

## es6

let const

- let相当于给js新增了块级作用域，声明的变量只在let命令所在的代码块内有效
- const也是声明变量，它声明的变量，不能改变，可以用来声明第三方库变量的应用



首页



问答



专栏



讲堂



更多

- class定义一个类，其中有一个construct方法，construct方法中的this代表实例对象，construct以外还有其他的方法，construct内定义的方法属性是实例对象自己的，construct外的方法属性是所有实例对象共享的
- class之间可以通过extends实现继承
- super指代父类的实例，子类construct中必须先调用super()方法，因为子类没有自己的this对象，是继承父类的this对象

#### arrow function(箭头函数)

除了书写简洁了很多，最大的优点是this指向，使用箭头函数，函数内部的this就是定义时所在的对象。箭头函数根本没有自己的this，this是继承外面的，它内部的this就是外层代码块的this

#### template string(模板字符串)

ajax调用数据库，需要向文档中插入大段html的时候，传统的字符串拼接太麻烦，引入模板工具库会稍微好点，不过还是没有es6的template string简单，可以直接用反单引号包括代码块``,用\${}来引用变量，所有的空格缩进都会保留到输出中

#### destructuring(解构赋值)

es6按照一定模式，从数组和对象中提取值，对变量进行赋值，这就成为解构，也就是说，运用es5的方法，数组和对象中的变量需要，一个个进行赋值，而es6可以一步到位

#### default,rest(默认值，扩展语法)

当函数忘记传参的时候，给它一个默认值，传统方法是在函数中运用||，es6可以直接在参数中写上

```
function animal(type){
  type = type || 'cat'
  console.log(type)
}
animal()
```

```
function animal(type = 'cat'){
  console.log(type)
}
animal()
```

```
function animals(...types){
  console.log(types)
}
animals('cat', 'dog', 'fish') //["cat", "dog", "fish"]
```

## gulp

gulp是一种自动化构建工具，前端工程化开发的一种工具，增强开发流程

使用方便，npm安装，新建gulpfile.js,导入gulp模块，let gulp = require('gulp')

通过default任务去定义 workflow

最后在终端执行gulp来进行自动化操作

#### api很简单只有四种

- gulp.task 创建任务：参数任务名称，前置任务数组，回调函数
- gulp.src 寻找文件：通过路径找到一个或多个文件
- gulp.dest 输出到指定目录：如果没有就新建一个
- gulp.watch 监听文件变化，执行任务
- pipe具体不清楚，总之，除了gulp.src之外，其他执行条件都要放在.pipe()中



首页



问答



专栏



讲堂



更多

# Bootstrap

## Bootstrap和Foundation的比较

- UI元素的不同
- Bootstrap给出了能想到的一切元素，也就是试图提供所有定义好的UI，比如一个导航，给予一个默认导航的样式
  - Foundation只给定了限定的几种元素，可以自己自定义，更适合创造
- 尺寸单位不一样，
- Bootstrap是px
  - Foundation是rem
- 网格布局有所不同
- Foundation 的网格可以自动适配当前浏览器的宽度，Foundation 则会灵活适配当前的浏览器宽度, 这是一种新的技术手段, 自动适配的同时, 可以表现的与 Transformer 一样的效果.
  - Bootstrap 则是预定义了几种网格尺寸来适配主流的设备 and 屏幕.Bootstrap 会在你改变浏览器宽度的时候突然改变它的网格.
- 移动设备
- Foundation移动设备优先
  - Bootstrap也支持移动设备
- 社区
- Bootstrap有一个完备的社区,有什么问题几乎都可以迅速解决
  - Foundation则没有，需要自力更生

2017年11月06日发布

...

...

赞 | 22

收藏 | 205

### 你可能感兴趣的文章

- [ 前端笔试 ] 2016阿里巴巴校招前端笔试部分试题 ( 持续更新...) 103 收藏，17.7k 浏览
- 百度web前端—2015笔试 16 收藏，2.7k 浏览
- 腾讯前端求职直播课——笔试篇 132 收藏，1.6k 浏览

### 8 条评论

默认排序

时间排序

- 


haru · 2017年11月07日
- 创建数组还有Array.of()
- 

赞 +3

回复
- 1

嗯是的，这是es6中新增的Array方法，感谢你的建议，已更新

— 澹台宇鹏 作者 · 2017年11月07日

王海强 · 2017年11月07日

22 localhost?

👍 赞 +1


回复

1

打错了打错了，多谢提醒啊，已改

— 澹台宇鹏 作者 · 2017年11月07日

添加回复

Y\_xx · 2017年11月08日

总结的不错，涵盖的也比较全面。

但是有些答案太简单了或描述的不够准确。比如 闭包可不是两句话能说明白的哦，比如const 声明的“不能改变的变量”，我们一般说是『常量』。

总之很不错的了，适合对照着知识点再深入了解了解~

👍 赞 +1

回复

嗯好的，我会慢慢丰富内容的

— 澹台宇鹏 作者 · 2017年11月09日

添加回复

随缘 · 2017年11月09日

GET与POST方法的区别

中的 git 是什么鬼😏😏😏

👍 赞 +1

回复

好的，已改

— 澹台宇鹏 作者 · 2017年11月09日

添加回复

文明社会，理性评论


发布评论

NordVPN

2018年最佳的NordVPN优惠

最可靠的  
在线媒体流的VPN

立即获取VPN!

澹台宇鹏

495 声望

关注作者