### JavaScript

#### 在JavaScript源文件的开头包含'use strict'的意义和有什么好处？

消除JS的某些怪异行为，使JS更安全。

严格模式下函数的参数不能重名

防止变量意外被提升为全局变量

不能使用未声明的变量

this的默认绑定，在严格模式下this绑定到undefined，否则绑定到全局对象

在eval()中的定义的变量不能被调用

#### Call 和apply的区别？bind的作用是什么？

A全call单：apply的参数是数组，而call是单个单个的。他们都是在特定的作用域中调用函数。

Bind是返回绑定特定作用域的一个新的函数。

#### TDZ（temporal Dead Zone,暂时性死区），是由于代码中变量未被初始化而不能被引用的情况。

1. {
2. console.log(a); // ReferenceError: a is not defined
3. let a = 2;
4. }

这个代码会抛错, 但是 将let 改为 var就没有问题了，打印为是undefined

1. {
2. typeof a; // undefined
3. typeof b; // ReferenceError (TDZ)
4. let b;
5. }

#### This理解

This 取决于调用时的位置。1. new foo() 2. foo.call(obj2) 3. obj1.foo(), 4. var bar = foo();

ES6中的 箭头函数中的this是根据外层的词法作用域来决定的。

#### 下面的代码将输出到控制台的是什么？，为什么？

1. **var** myObject = {
2. foo: "bar",
3. func: **function**() {
4. **var** self = **this**;
5. console.log("outer func: this.foo = " + **this**.foo);
6. console.log("outer func: self.foo = " + self.foo);
7. (**function**() {
8. console.log("inner func: this.foo = " + **this**.foo);
9. console.log("inner func: self.foo = " + self.foo);
10. }());
11. }
12. };
13. myObject.func();

以上代码将输出到控制台：

1. outer **func**: **this**.**foo** = **bar**
2. outer **func**: **self**.**foo** = **bar**
3. inner **func**: **this**.**foo** = **undefined**
4. inner **func**: **self**.**foo** = **bar**

原因是this的值取决于它的调用位置, func的调用是隐式调用，所以他的this是指向myObject的, 即this.foo = myObject.foo， 但是内部的立即执行函数的调用位置是foo内部，this指向func，而func内部是没有foo的，即为undefined。

#### 什么是NaN？它的类型是什么？如何可靠地测试一个值是否等于NaN？

NaN是指不是个数字，理解为无效的数字，他的类型是number, 可用Object.is(a, NaN)

#### 什么是JavaScript中的“闭包”？举一个例子。

当函数能够记住并访问所在词法作用域的时候，就产生闭包了。

Function foo() {

Var a = 2;

Function innerFun() {

Console.log(a);

}

Return innerFun;

}

Var test = foo();

Test(); // 2 闭包， innerFun在自己定义的词法作用域之外执行

#### 以下代码的输出是什么：

1. **for** (**var** i = 0; i < 5; i++) {
2. setTimeout(**function**() { console.log(i); }, i \* 1000 );
3. }

5个5，如何改进？

1. for (**var** i = 0*; i < 5; i++) {*
2. (**function**(**x**) {
3. setTimeout(**function**() { console.log(**x**)*; }, x \* 1000 );*
4. })(**i**)*;*
5. }

#### 考虑下面的两个函数。他们都会返回同样的值吗？为什么或者为什么不？

1. **function** **foo1**()
2. {
3. **return** {
4. bar: "hello"
5. };
6. }
8. **function** **foo2**()
9. {
10. **return**
11. {
12. bar: "hello"
13. };
14. }

因为分号在JS里不是必须的。是可选的。所以提倡开头的大括号放在行尾。

#### 下面的代码输出什么？解释你的答案。

1. console.log(0.1 + 0.2);
2. console.log(0.1 + 0.2 == 0.3);

一个典型的解决方案是比较两个数字与特殊常数Number.EPSILON之间的绝对差值：

1. Number.EPSILON=(function(){ //解决兼容性问题
2. return Number.EPSILON ? Number.EPSILON : Math.pow(2,-52);
3. })();
4. **function** roughEqual(num1, num2) {
5. **return** Math.abs( num1 - num2 ) < **Number**.EPSILON;
6. }
7. console.log(roughEqual(0.1 + 0.2, 0.3));

#### 以下几行代码输出到控制台？

0 || 1 = 1

1 || 2 = 1

0 && 1 = 0

3 && 2 = 2 // &&运算符的有趣之处在于，当表达式评估为“真”时，则返回表达式本身

#### == 和=== 区别

== 在判定时候会进行强制类型转换，而 === 不会进行。

== 是判定值相等否，而=== 是判定值和类型是否都相等

JS有六种基本类型（2nbus2）和一种复杂类型（object）；

#### 以下代码的输出是什么？解释你的答案。

1. var a={},
2. **b={key:'b'},**
3. c={key:'c'}*;*
5. a[**b]=123;**
6. a[c]=456*;*
8. console.log(a[**b]);**

此代码的输出将是456（不是123）。//主要因为b 和c都是对象，他们都被转化为”[object object]”

原因如下：设置对象属性时，JavaScript会隐式地将参数值串联起来。在这种情况下，由于b和c都是对象，它们都将被转换为“[object Object]”。因此，a [b]和a [c]都等价于[“[object Object]”]，并且可以互换使用。因此，设置或引用[c]与设置或引用[b]完全相同。

#### 以下代码将输出到控制台以及为什么

1. **var** hero = {
2. \_name: 'John Doe',
3. getSecretIdentity: **function** (){
4. **return** **this**.\_name;
5. }
6. };
8. **var** stoleSecretIdentity = hero.getSecretIdentity;
10. console.log(stoleSecretIdentity());
11. console.log(hero.getSecretIdentity());

这段代码有什么问题，以及如何解决这个问题。

该代码将输出：

1. undefined
2. John Doe

#### 以下代码将输出到控制台以及为什么

1. (**function** () {
2. **var** x, y; *// outer and hoisted*
3. **try** {
4. **throw** **new** Error();
5. } **catch** (x */\* inner \*/*) {
6. x = 1; *// inner x, not the outer one*
7. y = 2; *// there is only one y, which is in the outer scope*
8. console.log(x */\* inner \*/*);
9. }
10. console.log(x); // outer x = undefined
11. console.log(y); // outer y = 2
12. })();

#### 你如何克隆一个对象？

1. **var** obj = *{a: 1 ,b: 2}*
2. **var** objDeepClone = **JSON.parse(JSON.stringify(obj))**;

#### 如何在数组的开头添加元素？最后如何添加一个？

1. var myArray = ['a', 'b', 'c', 'd'];
2. myArray.push('end');
3. myArray.unshift('start');
4. console.log(myArray); // ["start", "a", "b", "c", "d", "end"]

使用ES6，可以使用扩展运算符：

1. **myArray** = ['start', ...myArray];
2. **myArray** = [...myArray, 'end'];

或者，简而言之：

1. **myArray** = ['start', ...myArray, 'end'];

#### typeof undefined == typeof NULL的值是什么？

该表达式将被评估为true，因为NULL将被视为任何其他未定义的变量。

*注意：JavaScript区分大小写，我们在这里使用NULL而不是null。*

#### 代码返回后会怎么样？

1. console.log(**typeof** **typeof** 1); // string

string

因为typeof 出来的都是加引号的字符串

typeof 1将返回“number”，typeof“number”将返回字符串。

下面a值为多少？

function foo(x) {

x.push(4);

console.log(x); // [1,2,3,4]

x = [4, 5, 6];

x.push(7);

console.log(x); // [4,5,6,7]

}

var a = [1,2,3];

foo(a);

console.log(**a**); // [1,2,3,4]

相当于 b = a; 然后调用foo(b)

然后第一个x 和b 指向共同的堆值【1,2,3】， 然后进行push，这是b和第一个x的值都变了。

紧接着，x 的指向发生变化了，指向另外一个堆的值【4,5,6】，紧接着改变当前的值。于是结果如上了。

### HTML5

### CSS

#### CSS 选择器有哪些

有属性选择器，开始，相等，不相等，结束，任意，语言a[attr|=”LANG”]，多条件选择器a[attr=’value1’][attr=’value2’]{}

a[attr] {}

### 数据库

### 算法

### Linux

### 网络协议

#### 11、你怎么理解http协议？

基于请求和响应，无状态的应用层协议，常基于TCP/IP协议传输数据的。

基于请求与响应，无状态的，应用层的协议，常基于TCP/IP协议传输数据

#### 12、说说http协议的工作流程。

在HTTP1.0中，通讯流程可以概括为以下几步：（短链接流程）

             1. 客户端与服务器建立TCP连接，一般使用80端口。

             2. 客户端向服务器发送请求。

             3. 服务器向客户端发送响应。

             4. TCP连接断开。

             5. 重复上述步骤

客户端输入URL回车，DNS解析域名得到服务器的IP地址，服务器在80端口监听客户端请求，端口通过TCP/IP协议（可以通过Socket实现）建立连

#### 13、http有哪些请求提交方式？

增删改查，post，delete，post，get

替换：put

#### 14、http中的200,302,403,404,500,503都代表什么状态？

* 200 - 请求成功
* 302 - 临时移动。但资源只是临时被移动。客户端应继续使用原有URI
* 403 - 服务器理解请求客户端的请求，但是拒绝执行此请求
* 404 - 请求的资源（网页等）不存在
* 500 - 内部服务器错误
* 503 - 由于超载或系统维护，服务器暂时的无法处理客户端的请求。延时的长度可包含在服务器的Retry-After头信息中

|  |  |
| --- | --- |
| **分类** | **分类描述** |
| 1\*\* | 信息，服务器收到请求，需要请求者继续执行操作 |
| 2\*\* | 成功，操作被成功接收并处理 |
| 3\*\* | 重定向，需要进一步的操作以完成请求 |
| 4\*\* | 客户端错误，请求包含语法错误或无法完成请求 |
| 5\*\* | 服务器错误，服务器在处理请求的过程中发生了错误 |

#### 15、http get和post有什么区别？

Get 是获取信息，而post是增加或者更改信息。

#### 19、什么是http代理服务器，有什么用？

就是一台本地机器和服务器中间的中间机器。

作用：匿名上网；通过代理缓存，加快上网速度；可虑器；

#### 16、你怎么理解cookie和session，有哪些不同点？

cookie以文本格式存储在浏览器上，存储量有限（4KB）；

而会话存储在服务端，可以无限量存储多个变量并且比cookie更安全

#### 17、什么是web缓存？有什么优点？

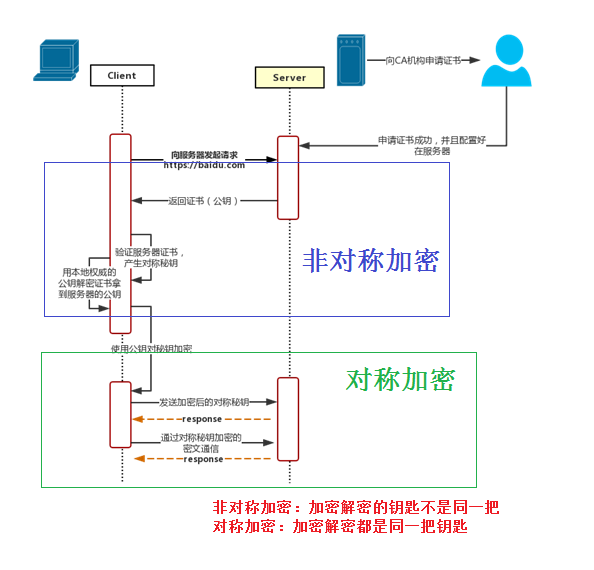
Web缓存是为了加快数据访问速度，和节约流量的一种技术。

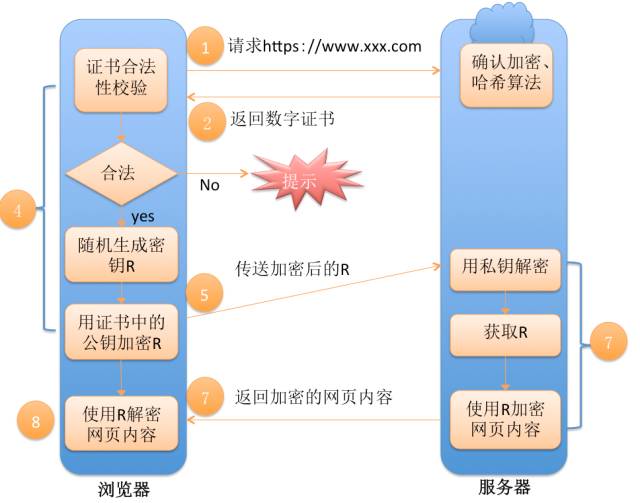
优点： 加速访问速度，让用户可以后退而不用从新加载数据。

#### 18、什么是https，说说https的工作原理？

穿着TLS外衣的http。内容加密，验证身份，防止数据篡改。

Transport Layer Security，TLS，网络通信提供安全及数据完整性的一种安全协议





①client向server发送请求https://baidu.com，然后连接到server的443端口。

服务端必须要有一套数字证书，可以自己制作，也可以向组织申请。区别就是自己颁发的证书需要客户端验证通过，才可以继续访问，而使用受信任的公司申请的证书则不会弹出提示页面，这套证书其实就是一对公钥和私钥。

②传送证书

这个证书其实就是公钥，只是包含了很多信息，如证书的颁发机构，过期时间、服务端的公钥，第三方证书认证机构(CA)的签名，服务端的域名信息等内容。

③客户端解析证书

这部分工作是由客户端的TLS来完成的，首先会验证公钥是否有效，比如颁发机构，过期时间等等，如果发现异常，则会弹出一个警告框，提示证书存在问题。如果证书没有问题，那么就生成一个随即值（秘钥）。然后用证书对该随机值进行加密。

④传送加密客户端

这部分传送的是用证书加密后的秘钥，目的就是让服务端得到这个秘钥，以后客户端和服务端的通信就可以通过这个随机值来进行对称加密解密了。

⑤服务段加密要回复的信息

服务端用私钥解密秘密秘钥，得到了客户端传过来的私钥，然后把内容通过该值进行对称加密。

⑥传输加密后的请求回复信息

这部分信息是服务端用私钥加密后的信息，可以在客户端被还原。

⑦客户端解密信息

客户端用之前生成的私钥解密服务端传过来的信息，于是获取了解密后的内容。

#### 21、讲一讲TCP协议的三次握手和四次挥手流程。

主要重点在于TCP是**全双工**的通信。

握手协议：打电话掉线之后两人的谈话一样

* 1. C->S:你能听到我说话吗？ SYN=1，Seq=10001
  2. S->C：可以！（ACK=1，Seq=10002） S->C：你可以听到说的话吗？(SYN=1, Seq=30001)
  3. C->S：我也可以，然后通道建立了。(ACK=1, Seq=30002)

挥手协议：

* 1. C->S:我说完了(FIN=1, Seq=2000)
  2. S->C：好的！(ACK=1, Seq=2001)
  3. S->C：那我也不说了(FIN=1, Seq=4000)
  4. C->S：那好吧！(ACK=1, Seq=4001)

#### 27、为什么TCP建立连接协议是三次握手，而关闭连接却是四次握手呢？为什么不能用两次握手进行连接？

为了确保双方的都能收到消息，所以最少需要三次握手；

而关闭时，有可能另外一方还在传送数据，所以用四次挥手更好

如果是两次握手的话：一方接收到消息后不能确保应答消息到另一方，那么有可能在接收正确消息后，对方可能会一直重发消息。

#### 28、为什么TCP TIME\_WAIT状态还需要等2MSL后才能返回到CLOSED状态？

可以让TCP再次发送最后的ACK以防止这个ACK丢失（另一端超时并重发最后的FIN）

**确保最后两次挥手失败的时候，可以再发起最后两次挥手。**

#### 61、为什么说TCP/IP协议是不可靠的？

#### 70、OSI有哪七层模型？TCP/IP是哪四层模型。

OSI：物数网传会表应

TCP/IP： 数据，网络，传输，应用

36、怎么理解栈、堆？堆中存什么？栈中存什么？

37、为什么要把堆和栈区分出来呢？栈中不是也可以存储数据吗？