李姚组

1.js数据类型的分类(李姚)

（1）值类型（基本类型）：字符串（String）、数值（Number）、布尔值（Boolean）、Null、Undefined

1、占用空间固定，保存在栈中

2、保存与复制的是值本身

3、使用typeof检测数据的类型

4、基本类型数据是值类型

```
var a = 1;
var b = a;
a = 2;console.log(b);//结果是1。b不会因为a改变而改变
```

（2）引用类型：对象（Object）、数组（Array）、函数（Function）

1、占用空间不固定，保存在堆中

2、保存与复制的是指向对象的一个指针

3、使用instanceof检测数据类型

4、使用new()方法构造出的对象是引用型

```
var a = {age:20};
var b = a;
b.age = 21;
console.log(a.age) //21
```

实列：

```
var o = new Object();
function foo(obj) {
  obj.name = "xyc";
  obj = new Object();
  obj.name = "lxy";
}
foo(o);
console.log(o.name); //??
```

详解：

1.新建对象`var o = new Object();`

2.在foo的环境下执行`obj.name = "xyc"`由于是参数传递，在局部作用域内相当于执行了`obj = o`

3.在局部作用域内新建对象，并赋值相同的属性值

```
obj = new Object();
obj.name = "lxy";
```

4.foo()执行完毕，局部作用域出栈，obj声明周期结束 此时，新建的对象依然存在，等待下一次内存自动回收机制将堆中的无引用对象销毁。

2.js有那几种情况下为false(李姚)

```
NaN,null,undefined,0,""
```

3.javascript的内置对象(李姚)

即能用typeof得到类型

```
1 typeof undefined // undefined
2 typeof 'abc' // string
3 typeof 123 // number
4 typeof true // boolean
5 typeof {} // object
6 typeof [] // object
7 typeof null //object
8 typeof console.log // function
9 typeof NaN // number
10 typeof class foo {} // function
11 typeof Symbol() // Symbol
```

7. `null` 的值从技术上来说和 `object` 和 `number` 一样，都是最基本的值，按理来说，`null` 的类型也应该是"null"。然而并非如此，因为JavaScript最初设计时出了一点意外。

在JavaScript最初设计时，一个值有两个部分组成：它的类型标签和实际的值。有5个类型标签可以使用，而且对象类型的引用指向 `0`。`null` 的值始终指向 `NULL` 指针，它在大部分平台都是用 `0x00` 来表示。由于这种相似性，`null` 就用过 `0` 类型标签来表示，所以符合对象的引用。

8.因为函数是一个特殊的引用类型，在js中函数的定位很高，在任何情况下，都能typeof识别函数。

9. `NaN` 代表某个值不是一个数字，但出乎意料的是，它是"number"类型。原因是这样的，在计算机内部，`NaN` 是以数字类型储存的。然而，它是一个不能用实际数字来表示的数值类型的值。所以它叫“Not a Number”，这并不意味着它不是数值类型。相反，它意味着这个值不能用数值表示。

10.JavaScript的类只是一个被语法糖包裹的函数方法。实际上创建了一个同样的函数，但是作者的写法不同，只是看起来个简洁。这就是为什么 `typeof` 一个类，得到的仍然是"Function"。

11.es6新类型。**Symbol**是用来定义对象的唯一属性名的不二之选。

```
Symbol("foo") === Symbol("foo"); //输出: false
```

也就是说他自己都不等于自己

4.JS的内存机制与垃圾回收机制

内存管理机制就是分配内存管理，每种编程语言都有它的内存管理机制，JavaScript的内存管理机制是:内存基元在变量（对象，字符串等等）创建时分配，然后在他们不再被使用时“自动”释放。后者被称为垃圾回收。这个“自动”是混淆并给JavaScript（和其他高级语言）开发者一个错觉：他们可以不用考虑内存管理，JS内存空间分为栈(stack)、堆(heap)、池(一般也会归类为栈中)。

其中栈存放变量，堆存放复杂对象，池存放常量，在js中的分配的内存一般有如下的生命周期

> - 内存分配（当我们声明变量，函数，对象时系统自动为他们分配内存）
> - 内存使用（使用变量，函数等）
> - 内存回收（使用完毕，由垃圾回收机制自动回收不再使用的内存）

当内存走到最后一步的时候就开始内存回收，js中使用的是垃圾回收机制

垃圾回收有2种基本方式————

1. 标记清除————

垃圾回收器会在运行时给存储在内存中的所有变量加一个标记，然后去除环境中的变量以及被环境中的变量所引用的变量（闭包）在这些完成后仍存在标记的就是要删除的变量了，因为环境中的变量已经无法访问到这些变量了

2. 引用计数————

引用计数的策略是跟踪记录每个值被使用的次数。当声明了一个变量并将一个引用类型赋值给该变量时，这个值得引用次数就加一，如果该变量的值变成了另一个，则这个值得引用次数就减一，当这个值的引用次数为0的时候，说明没有变量在使用，这个值无法访问。由此可以将其占用的空间回收，这些垃圾回收器就会在运行时清理掉引用次数为0的值占用的空间，但这种方法容易引起内存泄漏，因为这种方式没有解决循环引用的问题，所以不建议使用!

5.JS中的内置函数

1、9个常规函数

2、Array对象4个数组函数

3、Data对象20个日期函数

4、Math对象的属性和函数

5、String对象20个字符串函数

6.https的原理及其局限性

一、原理:

（1）客户使用https的URL访问Web服务器，要求与Web服务器建立SSL连接。

（2）Web服务器收到客户端请求后，会将网站的证书信息（证书中包含公钥）传送一份给客户端。

（3）客户端的浏览器与Web服务器开始协商SSL连接的安全等级，也就是信息加密的等级。

（4）客户端的浏览器根据双方同意的安全等级，建立会话密钥，然后利用网站的公钥将会话密钥加密，并传送给网站。

（5）Web服务器利用自己的私钥解密出会话密钥。

（6）Web服务器利用会话密钥加密与客户端之间的通信。

二HTTPS的优点

尽管HTTPS并非绝对安全，掌握根证书的机构、掌握加密算法的组织同样可以进行中间人形式的攻击，但HTTPS仍是现行架构下最安全的解决方案，主要有以下几个好处：

（1）使用HTTPS协议可认证用户和服务器，确保数据发送到正确的客户机和服务器；

（2）HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全，可防止数据在传输过程中不被窃取、改变，确保数据的完整性。

（3）HTTPS是现行架构下最安全的解决方案，虽然不是绝对安全，但它大幅增加了中间人攻击的成本。

（4）谷歌曾在2014年8月份调整搜索引擎算法，并称“比起同等HTTP网站，采用HTTPS加密的网站在搜索结果中的排名将会更高”。

三、HTTPS的缺点

虽然说HTTPS有很大的优势，但其相对来说，还是存在不足之处的：

（1）HTTPS协议握手阶段比较费时，会使页面的加载时间延长近50%，增加10%到20%的耗电；

（2）HTTPS连接缓存不如HTTP高效，会增加数据开销和功耗，甚至已有的安全措施也会因此而受到影响；

（3）SSL证书需要钱，功能越强大的证书费用越高，个人网站、小网站没有必要一般不会用。

（4）SSL证书通常需要绑定IP，不能在同一IP上绑定多个域名，IPv4资源不可能支撑这个消耗。

（5）HTTPS协议的加密范围也比较有限，在黑客攻击、拒绝服务攻击、服务器劫持等方面几乎起不到什么作用。最关键的，SSL证书的信用链体系并不安全，特别是在某些国家可以控制CA根证书的情况下，中间人攻击一样可行。

.get方式和post方式有什么区别(李许怡安)

.解释一下js原型链(李许)

童建设组

1.$.ajax的传参(戴靓)

2.js中的this怎么理解

徐保山组

1.原生ajax如何实现

2.常用的数组的方法有哪些(尹超)

程超

1.call,apply,bind的作用,之间的区别

2.什么叫域,js是否能跨域,如何跨域

3.promise的原理,如何使用promise

#####