**javascript**

1.您只能在 HTML 输出流中使用 **document.write**。 如果您在文档已加载后使用它（比如在函数中），会覆盖整个文档。

2.您会经常看到 **document.getElementById("some id")**。这个方法是 HTML DOM 中定义的。

DOM (**D**ocument **O**bject **M**odel)（文档对象模型）是用于访问 HTML 元素的正式 W3C 标准。

3.element.src.match("bulbon")意思是 Src路径里有bulbon匹配则为true.如果一开始的图片是/i/eg\_bulbon.gif,那么就会匹配到bulbon,if语句为ture,执行if

知识框图



1. **关于脚本的添加位置**

（1）HTML 中的脚本必须位于 <script> 与 </script> 标签之间。

脚本可被放置在 HTML 页面的 <body> 和 <head> 部分中。

通常的做法是把函数放入 <head> 部分中，或者放在页面底部。这样就可以把它们安置到同一处位置，不会干扰页面的内容。

提示：我们把 JavaScript 放到了页面代码的底部，这样就可以确保在 <p> 元素创建之后再执行脚本。

## （2）外部的 JavaScript

也可以把脚本保存到外部文件中。外部文件通常包含被多个网页使用的代码。

外部 JavaScript 文件的文件扩展名是 .js。

如需使用外部文件，请在 <script> 标签的 "src" 属性中设置该 .js 文件：

JavaScript 没有任何打印或者输出的函数。

JavaScript 可以通过不同的方式来输出数据：

* 使用 **window.alert()** 弹出警告框。
* 使用 **document.write()** 方法将内容写到 HTML 文档中。
* 使用 **innerHTML** 写入到 HTML 元素。
* 使用 **console.log()** 写入到浏览器的控制台。

使用 document.write() 仅仅向文档输出写内容。

\\\\\\如果在文档已完成加载后执行 document.write，整个 HTML 页面将被覆盖

**5.**

**document.getElementById("demo")** 是使用 id 属性来查找 HTML 元素的 JavaScript 代码 。该方法是 [HTML DOM](https://www.w3cschool.cn/htmldom/) 中定义的。

**innerHTML = "Paragraph changed."** 是用于修改元素的 HTML 内容(innerHTML)的 JavaScript 代码。

6.

变量可以通过变量名访问。在指令式语言中，变量通常是可变的。字面量是一个恒定的值。

JavaScript 标识符必须以字母、下划线（\_）或美元符（$）开始。

JavaScript 中，常见的是驼峰法的命名规则，如 lastName (而不是lastname)。

JavaScript 是脚本语言。浏览器会在读取代码时，逐行地执行脚本代码。而对于传统编程来说，会在执行前对所有代码进行编译。

## 7.空格

JavaScript 会忽略多余的空格。您可以向脚本添加空格，来提高其可读性。下面的两行代码是等效的：

var person="Hege"; var person = "Hege";

您也可能看到不带有分号的案例。   
在 JavaScript 中，用分号来结束语句是可选的。

**8.**

当您向变量分配文本值时，应该用双引号或单引号包围这个值。

当您向变量赋的值是数值时，不要使用引号。如果您用引号包围数值，该值会被作为文本来处理。

声明也可横跨多行：

var lastname="Doe",   
age=30,   
job="carpenter";

## 9.重新声明 JavaScript 变量

如果重新声明 JavaScript 变量，该变量的值不会丢失：

在以下两条语句执行后，变量 carname 的值依然是 "Volvo"：

var carname="Volvo";   
var carname;

var y=123e5; // 12300000

var z=123e-5; // 0.00123

**10.**

极大或极小的数字可以通过科学（指数）计数法来书写：、

var y=123e5; // 12300000

var z=123e-5; // 0.00123

**11.**

<!DOCTYPE html>

<html>

<body>

<script>

var i;

var cars = new Array();

cars[0] = "Saab";

cars[1] = "Volvo";

cars[2] = "BMW";

for (i=0;i<cars.length;i++)

{

document.write(cars[i] + "<br>");

}

</script>

</body>

</html>

**12.JavaScript 对象**

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

对象属性有两种寻址方式：

Undefined 这个值表示变量不含有值。

可以通过将变量的值设置为 null 来清空变量。

**13.声明变量类型**

当您声明新变量时，可以使用关键词 "new" 来声明其类型：

var carname=new String; var x= new Number;

var y= new Boolean;

var cars= new Array;

var person= new Object;

JavaScript 变量均为对象。当您声明一个变量时，就创建了一个新的对象。

提示：JavaScript具有隐含的全局概念，意味着你不声明的任何变量都会成为一个全局对象属性。



若在js里给onclick赋值函数，则不能带括号，正确方法应该是这样的： retage.onclick = function(){ ajax('/php/test.php', callBackFunctiona); } 。但若是内嵌的js，则应该带上()才是正确的方法。

**2.**content.innerHTML=z+":"+y+":"+x; 注意：用加号拼接

**3.setTimeout与setInterval的主要区别是:**

　　setTimeout()方法只运行一次，也就是说当达到设定的时间后就出发运行指定的代码，运行完后就结束了，如果还想再次执行同样的函数，可以在函数体内再次调用setTimeout()，可以达到循环调用的效果。

　　setInterval()是循环执行的，即每达到指定的时间间隔就执行相应的函数或者表达式，是真正的定时器。

**4.实现10秒后刷新当前网页**

（1）

setTimeout(**function**(){

    location.reload();

},10000);

（2）

setTimeout(location(),10000);  
function location(){  
window.location.href='本页面地址'；  
}

**3. 什么是跨域，如何处理？**

（1）跨域指的是浏览器不能执行其他网站的脚本。它是由\*\*浏览器的同源策略\*\*造成的，是\*\*浏览器施加的安全限制\*\*。

（2）常用处理方式：

1) JSONP：动态添加script标签

2）设置请求头允许

3) 反向代理

**1.document.write**

如果页面已经加载完毕，再使用 document.write，那就会覆盖整个文档。鼠标动作是在页面加载完毕后执行的

1. **js对：var 后面的变量、函数参数、函数进行预编译**

匿名函数不参与预编译; 只有在解释执行阶段才会进行变量初始化

<https://blog.csdn.net/q1056843325/article/details/52951114>

1. **为啥隐式声明的全局变量可以删除，显式声明的全局变量就无法删除？**

原因是"delete 不可以删除那些可配置性为false的属性" ，某些内置对象的属性是不可配置的，比如通过变量声明或者函数声明创建的全局对象的属性，以下代码为证

1. **关于es6的和es5的区别**

<https://www.imooc.com/video/16701>

**5.**加号优先级高于 三目运算。低于括号

**6.**import用于引入css，js暂未发现这种用法。

1. **Em rem px的区别**

px就是像素值，em就是根据基准来缩放字体的大小。em是相对于父元素的属性而计算的，Rem是相对于根元素<html>

**5.js '5' + 3 等于**

javascript 中， '5' + 3 的运算结果为 53， 当执行加法运算时，因为第一个操作数'5'为字符串形式，所以优先采用字符串连接。  
而 '5' - 3 的运算结果为 2，因为减法运算时，会尝试将操作数转换为 Number，再进行运算。所以 等同于 5 - 3，结果为 2。

同理，3 + 3 + '5'，计算结果应为 '65' (string)， 从左到右计算，3 + 3，因为两个操作数都是 number，所以直接加法，结果为6，再计算 6 + '5'，因为有一个操作数是 string，所以按字符串连接，结果就是 '65'。（以示区别，将字符串类型的用单引号括起来方便识别）

**6.常见的浏览器端的存储技术有哪些？**

cookie：放在http请求头中，伴随数据传输而传输，数据传输大小有限制，有过期时间

localstorage：存储在本地，不会伴随数据传输，生命周期为永久

sessionstorage：浏览器中，浏览器关闭则消失，即使在同源浏览器中也不能共享.任何一个页面存储的信息在窗口中同一网站的任何页面都可以访问它存储的数据。每个窗口的值都是独立的，它的数据会因窗口的关闭而丢失，不同窗口间的sessionStorage是不可以共享的。

userdata：ie中用于浏览器存储技术

globalstorage：ff中用于浏览器存储,和sessionStorage一样，域中任何一个页面存储的信息都能被所有的页面共享。目前只有FF支持，且只支持当前域下的globalStorage存储。

**7.非严格模式下写出下面表达式结果：**

**parseInt(“123a”)= 123**

parseInt()方法首先查看位置0处的 字符，判断它是否是个有效数字；如果不是，该方法将返回NaN，不再继续执行其他操作。但如果该字符是有效数字，该方法将查看位置1处的字符，进行同样的 测试。这一过程将持续到发现非有效数字的字符为止，此时parseInt()将把该字符之前的字符串转换成数字。

1. **Array 和 string自带的方法区别**

**(1)数组的方法：splice() 方法与 slice() 方法和split**

➀Splice(): arrayObject.splice(index,howmany,item1,.....,itemX)

//splice的用处更广一些，可以用于删除元素，插入元素等操作，由参数来控制

//请注意，splice() 方法与 slice() 方法的作用是不同的，splice() 方法会直接对数组进行修改。

//返回被删除的项目。

➁Slice : arrayObject.slice(start,end)

//返回一个新的数组，包含从 start 到 end （不包括该元素）的 arrayObject 中的元素。

该方法并不会修改数组，而是返回一个子数组。

//您可使用负值从数组的尾部选取元素,即取负值

//test:

console.log(arr.slice(-1,-2));//[]

console.log(arr.slice(-2,-1));//[ 'Adrew' ]

//说明顺序一定要按照数组的顺序来写

➂Split: stringObject.split(separator,howmany)

//用于把一个字符串分割成字符串数组。

<https://blog.csdn.net/cristina_song/article/details/77917404>

<https://blog.csdn.net/zpw91/article/details/53705462>

1. **字符串的方法：slice() 方法和substring() 方法**

➀slice() : stringObject.slice(start,end)

//一个新的字符串。包括字符串 stringObject 从 start 开始（包括 start）到 end 结束（不包括 end）为止的所有字符。

➁stringObject.substring(start,stop)

方法返回的子串包括 start 处的字符，但不包括 stop 处的字符,这一点二者相同

//与 [slice()](http://www.w3school.com.cn/jsref/jsref_slice_string.asp) 和 [substr()](http://www.w3school.com.cn/jsref/jsref_substr.asp) 方法(不推荐)不同的是，substring() 不接受负的参数。

➂Split: 字符串也有split, 与 [Array.join](http://www.w3school.com.cn/jsref/jsref_join.asp) 执行的操作是相反的。

split不改变原数组，事实上只有数组的方法---Splice()会改变原数组

http://www.w3school.com.cn/jsref/jsref\_split.asp

1. **Ajax是什么？Ajax的交互模型？同步和异步的区别？如何解决跨域问题？**
2. Ajax是什么？
3. Ajax全称asynchronous javascript and xml ，可以说是已有技术的组合。
4. 主要用来实现客户端与服务器的异步通信效果，实现页面的局部刷新

3、ajax的实现过程 创建XMLHttpRequest对象，也就是创建一个异步调用对象 创建一个新的http请求，并指定该http请求的方法、URL以及验证信息 设置响应http请求状态变化的函数 发起http请求 获取异步调用返回的数据 使用javascript 和 dom 实现局部刷新

1. 同步、异步的区别 同步:阻塞的 两人吃饭。一人忙，一人等 异步:非阻塞的 两人吃饭。一人忙，另一个不等 三、如何解决跨域问题？
2. 理解跨域的概念:协议、域名、端口都相同是同源，否则都是跨域 .

解决:动态创建script标签，使用标签的src属性访问js文件的形式获取js脚本，并且这个js脚本中的内容是函数调用，该函数调用的参数是服务器返回的数据，为了获取这里的参数数据，需要事先在页面中定义回调函数，在回调函数中处理服务器返回的数据。

<https://blog.csdn.net/u014727260/article/details/72793459>

<https://www.cnblogs.com/itmacy/p/6958181.html>

**10.简述instanceof和type的区别？**

**简述[ ]instanceof Object的值和原因？**

<https://segmentfault.com/a/1190000000730982>

1. 关于数据类型
2. 共有6种类型，其中基本类型5种，undefined null string number boolean, 复杂类型1种：object .
3. 若用typeof来测试，结果和类型并不一样，有6种undefined,boolean,string,number,

Function，object(包括一般object, null, array)

注意：Typeof 是操作符而不是函数，因此圆括号可以不用

**11.split() 方法用于把一个字符串分割成字符串数组。**

var p1 = part1.split(''),//将字符串以空格分割成为数组

**12.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。**

stringObject.indexOf(searchvalue,fromindex)

注释：indexOf() 方法对大小写敏感！

注释：如果要检索的字符串值没有出现，则该方法返回 -1。

1. **forEach，即各种数组的函数，需要看书学习**
2. **编写请给 Array 本地对象增加一个原型方法，它用于删除数字数组中重复的数字（可能有多个），返回值是一个包含被删除的重复条目的新数组。（牛客）**
3. **lastIndexOf() 方法可返回一个指定的字符串值最后出现的位置，在一个字符串中的指定位置从后向前搜索。**

stringObject.lastIndexOf(searchvalue,fromindex)

<http://www.w3school.com.cn/jsref/jsref_lastIndexOf.asp>

1. **下面哪种方式不能改变作用域链？（牛客）**

With, while, try catch, eval

1. **考查的应该是预编译的时候，对函数与变量的处理，以及调用。**

<https://blog.csdn.net/q1056843325/article/details/52951114>

1. **NAN**

表示非数值，NaN和任何数值都不相等，包括自己本身

1. **下面给定的代码中， myFunc() 返回什么？**

**var foo=’foo’;**

**var bar=’bar’;**

**function myFunc() {**

**return foo+bar;**

**}**

<http://www.jb51.net/article/90792.htm>

**2018.5.15**

1. **Object.keys()**

//注意：是对象的方法，而因为array也是对象，所以也可以用

Object.keys() 方法会返回一个由一个给定对象的自身可枚举属性组成的数组，数组中属性名的排列顺序和使用 [for...in](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Statements/for...in) 循环遍历该对象时返回的顺序一致 （两者的主要区别是 一个 for-in 循环还会枚举其原型链上的属性）。

<https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Object/keys>

**2.下列不等于true的是(牛客)**

[undefined === null](https://www.nowcoder.com/question/javascript:void(0);)

[var i = 2; ++i == 2](https://www.nowcoder.com/question/javascript:void(0);)

[1 === '1'](https://www.nowcoder.com/question/javascript:void(0);)

[&quot;&quot; == 0](https://www.nowcoder.com/question/javascript:void(0);)

**3.关于XML和JSON区别的描述正确的是（牛客）**

**4.concat() 方法用于连接两个或多个数组。**

该方法不会改变现有的数组，而仅仅会返回被连接数组的一个副本。

1. **toString的使用**
2. **时间对象 Date的使用**

http://www.cnblogs.com/carekee/articles/1678041.html

2018.5.16

1. **this 测试程序，多层函数里的this始终是window ?**

（1）是的，就算是在函数f里调用它的子函数，子函数的this仍然是window.

即：只要是fun();

这样调用的，不管该语句处于哪里，都是由window调用的。就算该函数都是在父函数里嵌套定义的子函数，再在父函数里fun();这样调用，结果也是由window调用。

只有一种情况例外：

delbtn[i].onclick = function(){

that.del(this);//this为当前按钮元素

}

这种添加按钮事件，是由按钮点击触发调用的，所以这里的this是delbtn[i]，而非window.这种情况严格来说属于对象的方法调用，也不算例外。

1. Apply
2. 单步调试时F10 F11各自的作用
3. **深入理解变量声明提升和函数声明提升//参考test程序**

<https://blog.csdn.net/qq673318522/article/details/50810650>

注意：只有一个function开头的函数，只是个声明而已，所以函数提升时会把整个函数全提到最前面

1. **var 加与不加的区别**
2. 二者可以认为是指的一个。即以下2种情况时，两个输出是一样的输出：

i = 1;// 或改为 var i = 1;

alert(window.i);

alert(i);

alert(window.i);//若没有定义全局变量(或者声明了但没赋值)就用window.i访问，会输出undefined

alert(i);//若是这样直接访问i，不加window，则会报错：i is not defined

function hb(){

console.log(bl);

bl = 'hello world';

}

hb();

//这个的结果是报错，因为变量提升只是提升声明，这里bl并没有声明，所以提升不了。

1. 不加var 的相当于给window加了个属性，可以删除，加var 的是变量，不可删除
2. js里的运行顺序是：先运行函数外的，再运行函数内的，window.onload是函数内的，最后运行
3. **例子**

var num1 = 1;

num2 = 2;

//delete num1; //无法删除，报错

// delete num2; 删除

function model(){

var num1 = 7; // 本地变量

num2 = 6; // window的属性

// 匿名函数

(function(){

var num = 5; // 本地变量

num1 = 9; // 继承作用域（闭包）

console.log(num1);

num3 = 3; // window的属性

}())

}

model();

console.log(num1);

1. **例子**

//test4

console.log(foo);

var foo=10;

console.log(foo);

function foo(){

console.log(20);

}

console.log(foo);

//注意： 函数提升在变量提升上面，第一个console.log(foo);

//第一个为什么会输出函数体呢，原因在于 javascript永远是先解析声明函数，再解析变量，下一步声明变量var foo; 因为a此时并没有被赋值，所以它为 undefined, 还是指向原来的值，即函数 function a;

9.整个代码结构中只有函数可以限定作用域,所以对象并不是块作用域

1. **原型**

//原型相关

var test = function(){

 }

var my = function(){

  this.a = function(){

console.log(this === mytest2);

  }

 }

 var mytest = new my();//这里使用new的是指向函数的一个变量，并不是平时的构造函数

 test.prototype = mytest;

 var mytest2 = new test();

 mytest2.a();//true

//注意原型就是函数有原型，对象也有原型，除此之外没有了

10.重要内容，理解函数 对象等问题

<https://www.cnblogs.com/fengfan/p/3993542.html>

关于变量提升的：

<https://www.cnblogs.com/liu666/p/5747667.html>

1. javascript没有块的概念，而不是没有作用域的概念

typedef instanceof

**2018.5.17//今天是对书上类型一章的总结**

1. **Undefined**
2. var me;

Alert(me);//undefined

Alert(ha);//报错

(2)var me;

alert(typeof me);//undefined

alert(typeof ha);//undefined

1. Null

var car = null;

alert(typeof me);//object

1. 只要是空指针或空对象的话就应该将其置为Null,养成好习惯
2. Alert(null == undefined)//true,和这个==有关
3. **Boolean**
4. 使用boolean将其他类型转换为boolean类型的，见表p27
5. if可以自动完成boolean的转换
6. **NaN**
7. 神奇的NaN,任何数都不和它相等，包括NaN本身
8. isNaN()函数可以尽量将参数转变为数值，不能转换就返回true.见书p29.
9. 对象的valueof toString方法p30
10. **parseInt的用法见书P31**
11. parseInt会忽略字符串前面的空格，直到找到第一个非空格字符，若第一个不是数字字符或负号，会返回NaN.
12. 会认为小数点不是有效的数字字符
13. 能识别各种进制的模式，但注意，输出都是以十进制输出(es5不能识别八进制的0,一律直接忽略0,将后面的数字字符看作十进制的)
14. 由于parseInt对于进制的用法有争议，所以加第二个参数以指定转换的进制数，才是正确的做法p32
15. **parseFloat的用法见书p32**
16. 只认第一个小数点，遇到第二个小数点就截止
17. 与parseInt的第一个区别在于：始终忽略前面的0，由此，它不能解析带进制前缀的数，只解析十进制数，所以也就没有带第二个参数的情况
18. 若小数点后全0，会丢掉小数点和0，以一个整数形式出现。
19. **字符串**
20. ‘ ’和””没有区别，但php里是有区别的
21. 要把一个值转换为字符串的方法有两种，toString和String方法
22. **toString**
23. 数值，布尔值，对象和字符串值都有toString方法(注意是各自的toString)，undefined和null没有。
24. toString方法可以传参，指定进制数
25. **String**

若不知道要转换的数值是否是undefined和null的情况下，用String方法、null会返回null, undefined会返回undefined

1. **对象，每个对象实例都具有下面的方法：**
2. constructor:指向构造函数
3. hasOwnProperty(“propertyName”):检查给定属性是否在当前的实例中（不是在实例的原型中），参数是字符串的形式
4. isPrototypeOf(object):检查传入的对象是否是当前对象的原型
5. propertyIsEnumberable(“propertyName”):检查给定的属性是否能够使用for-in语句枚举。参数是字符串形式
6. toLocalString():返回对象的字符串表示,该字符串与执行环境的地区对应
7. toString():返回对象的字符串表示
8. valueOf():返回对象的字符串，数值，或布尔值表示，通常与toString()方法的返回值相同

//注意：以上方法对于所有的对象都有，但若是宿主对象则不一定支持，比如DOM,BOM中的对象

1. **相等操作符==**

比较过程如下：

1. 类型是否相同，相等则直接比较，不等进入(2)
2. 类型不同的转换规则如下

若有一个是bool值,将bool转换成数值，true是1，false是0；

若有一个是字符串，另一个是数值，将字符串转换为数值//注意：具体的转换规则还需记忆（比如空字符串会转换为0）

若一个是对象，另一个不是，则调用对象的valueOf()方法，用得到的基本类型按照前面的规则进行比较

Null和undefined相等

比较前不能将null和undefined转换为其他的任何值

NaN和任何数都不相等，包括自己

若两个都是对象则比较它们是不是同一个对象，若都指向同一个对象，则相等

//即A.f == B.f 可以判断二者是不是指向同一个对象

1. clientX 事件属性

返回当事件被触发时鼠标指针向对于浏览器页面（或客户区）的水平坐标。

语法：event.clientX

1. **addEventListener()和removeEventListener()**

**14.捕获阶段，冒泡阶段**

<https://blog.csdn.net/liwusen/article/details/50786795>

<https://segmentfault.com/a/1190000005736378> （讲得超级好）

<http://www.jb51.net/article/63740.htm>

（1）在DOM中，既支持事件冒泡，也支持事件捕获，在W3C的标准中，认为任何

事件都是从事件捕获出发，找到最终的结点，此后再进行冒泡，汇到根结点。

DOM中支持事件绑定的函数为：addEventListener("事件名"，函数，userCapture);

对于userCapture参数，默认为false，支持事件冒泡。

若在userCapture参数，为true时，支持事件捕获。

1. 对于很多浏览器，都支持addEventListener方法，但是IE不支持！
2. IE中的事件处理机制，IE中只支持事件冒泡，IE中有一个独有的事件绑定方法

attachEvent方法，此方法有两个参数，attachEvent("事件名"，"函数名")

1. 使用attachEvent()，事件处理程序会在全局作用域中进行，因此this===window
2. **看目录里的分类：**[**http://www.w3school.com.cn/jsref/event\_clientx.asp**](http://www.w3school.com.cn/jsref/event_clientx.asp)
3. 自己的一个总结：

像添加事件addEventListener()为浏览器差异而封装一个函数那样,以后的浏览器差异都可以这么处理，很好！

参考：<https://segmentfault.com/a/1190000005736378>

注意：用的一个对象来封装，很好

1. **回调函数和c中的区别**
2. **若想将github上的小项目以网页的形式展现，用github Pages**
3. **解决devtool显示not detected的问题：**

<http://www.cnplugins.com/tool/fix-vue-js-not-detected.html>

<http://www.cnplugins.com/tool/fix-vue-js-not-detected.html>

2018.5.18

1. **++的优先级高于==**
2. **&quot; 表示什么意思？**

在html里是“＂”（引号），用法：&quot;（&后没有空格,且带分号）

1. **优化效率的方法**

DocumentFragment

1. **DocumentFragment 的 createDocumentFragment()函数**

(1)DocumentFragment节点不属于文档树，继承的parentNode属性总是null。它有一个很实用的特点，当请求把一个DocumentFragment节点插入文档树时，插入的不是DocumentFragment自身，而是它的所有子孙节点。这个特性使得DocumentFragment成了占位符，暂时存放那些一次插入文档的节点。它还有利于实现文档的剪切、复制和粘贴操作。   
另外，当需要添加多个dom元素时，如果先将这些元素添加到DocumentFragment中，再统一将DocumentFragment添加到页面，会减少页面渲染dom的次数，效率会明显提升。

还有一个很重要的特性是，如果使用appendChid方法将原dom树中的节点添加到DocumentFragment中时，会删除原来的节点。

（2）用法：http://blog.163.com/rex\_blog/blog/static/194480101201412652735272/

1. **innerText 与 innerHtml的区别**

<https://blog.csdn.net/k491022087/article/details/52558868>

<https://yjlk123.github.io/yj-repository/Project1/recommendtrend.html>

<https://yjlk123.github.io/yj-repository/Project2/shoppingchart.html>

**6.对我的程序进行优化，上线后反应很慢，cdn是啥？如何用它来加速？**

（1）网页加载的顺序

（2）网易如何实现的瀑布流式布局

（3）图片压缩等方式，cdn等方式

（4）代码压缩

2018.5.22

**1.b.add.bind(a);**

这句话可以将b对象的add方法绑定到a对象上，但注意，这并不是立即调用，只是绑定了，但若换bind为call 或apply ,则是立即调用了

**2.< &lt; &#60; 小于号**

> &gt; &#62; 大于号

1. **优先级：**

条件运算符（？：）的优先级高于赋值运算符（=）

1. **在前端开发过程中使用 use strict 模式的目的是什么？（牛客）**
2. **js、css资源不受同源策略限制**

放开同源策略会导致巨大安全隐患

同源策略只针对浏览器，而服务器不是浏览器

端口、协议、域名相同才叫同源。

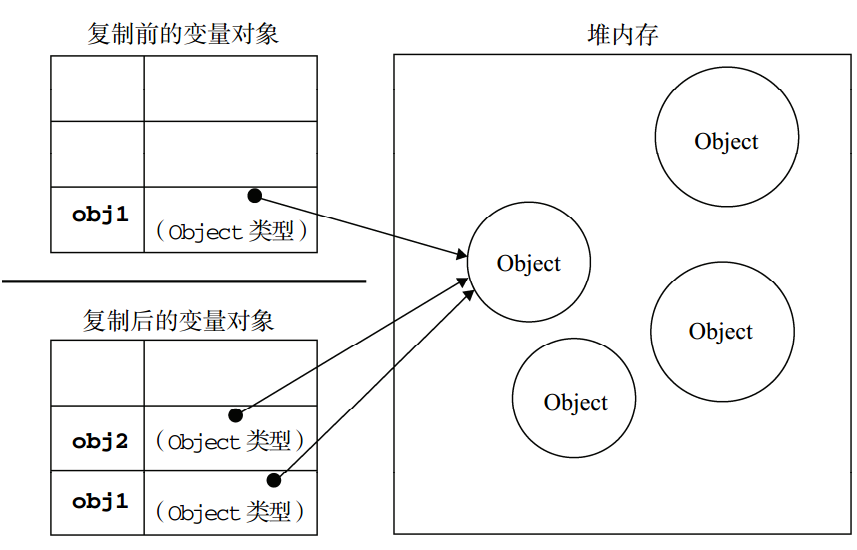
1. **关于栈和堆里存放的东西**

基本类型存放在栈(stack)中

引用类型存放在堆（heap）中

指向引用类型的地址存放在栈(stack)中

IMG_256



1. **以下关于箭头函数的说法正确的是 （牛客）**
2. **Join()见test3.js**
3. **js数组升序降序**

[**https://www.cnblogs.com/olive27/p/6101338.html**](https://www.cnblogs.com/olive27/p/6101338.html)

**2018.5.26**

1. **cookie 和 session 的区别描述正确的是**

cookie数据存放在客户的浏览器上，session数据放在服务器

cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗

session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能

单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie

//注意：Session是另一种记录客户状态的机制，不同的是Cookie保存在客户端浏览器中，而Session保存在服务器上。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上。这就是Session。客户端浏览器再次访问时只需要从该Session中查找该客户的状态就可以了。

如果说Cookie机制是通过检查客户身上的“通行证”来确定客户身份的话，那么Session机制就是通过检查服务器上的“客户明细表”来确认客户身份。Session相当于程序在服务器上建立的一份客户档案，客户来访的时候只需要查询客户档案表就可以了。

**2.下列代码运行的结果（ ） (牛客)**

|  |  |
| --- | --- |
| 1  2  3 | short i=65537;  int j=i+1;  printf("i=%d,j=%d\n",i,j); |

**3.数据库索引优缺点**

**4.整数0x12345678，在采用bigendian中内存的排序序列是（ ）**

<https://blog.csdn.net/qq_33724710/article/details/51056542>

1. **sql语句的匹配语句**
2. **二分查找也称折半查找（Binary Search），它是一种效率较高的查找方法。但是，折半查找要求线性表必须采用顺序存储结构，而且表中元素按关键字有序排列。**

**O(log2n)**

1. **分割字符串的方法名是 (split)，连接数组元素的方法名是 (join) 。**

分割字符串：split http://www.w3school.com.cn/js/jsref\_split.asp

连接数组元素：join http://www.w3school.com.cn/jsref/jsref\_join.asp

连接数组：concat http://www.w3school.com.cn/jsref/jsref\_concat\_array.asp

//字符串也有split()方法，不仅是数组有这个方法的。

1. **[]==[]和[]===[]的返回值都是什么？**
2. **请解释css中的浮动，以及清浮动的方法。（牛客）**

**2018.5.27**

1. **MVC**

Model: 代表着业务数据和业务逻辑处理，（主要包括业务逻辑模块（web项目中的dao类）和数据模块（model类）和JavaBean相同，由Model提供视图的显示数据，同时一个Model可以为多个视图提供业务数据，因此Model也就提高应用程序的可重用性。

View: 代表视图，也就是与用户交互的界面，可以给用户显示业务逻辑数据，同时也可以 接收用户输入的数据。（在web项目中以jsp作为view）

 Controller: 代表控制器，用来接收从视图输入的参数，同时调用对应的Model模型对数据进行业务逻辑处理，同时将处理好的数据传递到对应的视图来显示给用户。（在web项目中以Servlet作为控制器）

//MVC的处理顺序：首先是用户通过视图将参数输入提交到控制器，控制器调用对应的Model去处理，同时将处理好的数据输出到对应的视图显示给用户。

1. **Url**

URL由三部分组成：资源类型、存放资源的主机域名、资源文件名。

1. **试着编写一个 arrayMerge() 函数，实现该函数被调用时，传递任意数量的数组，返回一个合并后的数组（可不局限于一种实现）。(牛客)**

3.

canvas 中绘制的元素不可以通过浏览器提供的接口获取到。

SVG 中绘制的元素可以通过浏览器提供的接口获取到。

html 中 A标签 target 属性的默认取值是 \_self，默认在当前窗口打开。

为了优化网页的SEO效果，常自己手写，而不用 javascript 动态生成网页的title、description、keyword。

**4.float**

一、什么是浮动

通过设置float的值不为none，使得元素按特定的方向脱离文档流，不占据空间。浮动元素碰到包含它的边框或者浮动元素的边框停留。

二、浮动带来的问题

1. 父元素高度塌陷；

2.  浮动元素后的同级非浮动元素紧随其后；

3. 若浮动元素非同级第一个元素，其前面的元素也需要设置浮动，否则会被遮挡。

三、清除浮动的方法

1. 浮动元素后加一个空div，设置clear:both；

2. 父元素overflow: hidden/auto；zoom:1;

3. 父元素加clearfix类，定义如下：

.clearfix:after{

    display:block;

    content:''';

    height:0;

    clear:both;

    visibility:hidden;

}

.clearfix{

    \*zoom:1;    //IE6触发hasLayout

}

4. 固定父元素高度

2018.5.30

1. a;

Alert(typeof a);

Alert(typeof b);

注意：这两个都是返回undefined, 虽然b并未声明，但这就是为了使用typeof时任何情况下都不报错而设计的，若是直接alert(b) 或 if(b)都是会报错的。

1. typeof 和 instanceof的区别
2. typeof返回的类型就是那5种，array 和object 都会返回object,这正是typeof的局限性所在，此时需要instanceof来区别了。还有就是dom对象使用typeof判断，也会返回object,而instanceof对dom对象不支持判断
3. Instanceof 运算符左边是子对象（待测对象），右边是父构造函数（这里是Array），

实例：凡是用new 构造函数()创建出的对象，都称为是构造函数的实例

(3)这里谈到了父对象，就区别一下ifPrototypeOf() :

使用isPrototypeOf()函数

原理：检测一个对象是否是Array的原型（或处于原型链中，不但可检测直接父对象，还可检测整个原型链上的所有父对象）

注意：检查顺序和instanceof相反：

使用方法: parent.isPrototypeOf(child)来检测parent是否为child的原型;

3.多层次声明，会忽略，但多次赋值会取最后一次的值.

4.$('div+.ab')与$('.ab+div')哪个效率更高？

参考答案：前者

2018.6.8

1. unshift() 方法可向数组的开头添加一个或更多元素，并返回新的长度。

注释：该方法会改变数组的长度。

注释：unshift() 方法无法在 Internet Explorer 中正确地工作！

提示：要把一个或多个元素添加到数组的尾部，请使用 push() 方法。

**2018.6.14**

1. 将日期以字符串的形式表现出来，用该方法可以达成

function formatDate(date) {

return date.toLocaleDateString();

}

2.

**移动端的js**

**1.移动浏览器中，触摸一下屏幕会依次触发哪些事件.**

触摸事件，有touchstart touchmove touchend touchcancel 四种之分（1） touchstart：手指触摸到屏幕会触发 （2）touchmove：当手指在屏幕上移动时，会触发（3） touchend：当手指离开屏幕时，会触发 （4）touchcancel：可由系统进行的触发，比如手指触摸屏幕的时候，突然alert了一下，或者系统中其他打断了touch的行为，则可以触发该事件

解2：如果没有滑动，touchstart, touchend, click

touchcancel在别人打电话来的这种情境下触发

2018.6.15

1. isPrototypeOf

判断某对象的原型链里是否继承了某个对象原型

<https://www.cnblogs.com/giggle/p/5208199.html>

注意：不一定是直接父原型，可以是祖先。

1. getPrototypeOf

注意用法，一定要这么用Object.getPrototypeOf(object)

<https://www.cnblogs.com/giggle/p/5208199.html>

3.preventDefault() 方法

event.preventDefault()

该方法将通知 Web 浏览器不要执行与事件关联的默认动作（如果存在这样的动作）。例如，如果 type 属性是 "submit"，在事件传播的任意阶段可以调用任意的事件句柄，通过调用该方法，可以阻止提交表单。注意，如果 Event 对象的 cancelable 属性是 fasle，那么就没有默认动作，或者不能阻止默认动作。无论哪种情况，调用该方法都没有作用。

2018.6.17

1.Math.round(x)

round() 方法可把一个数字舍入为最接近的整数。

2018.6.19

1.string的slice和array 的slice不一样，string的可以参数为空，为空则表示是复制数组，而array的不能为空，第一个参数不能省略

1. isNaN的正确用法、

if (Number.isNaN(input)) {

return '';

}

1. **this**

<https://segmentfault.com/q/1010000006045750>

2018.06.25

1. JavaScript reduce() 方法

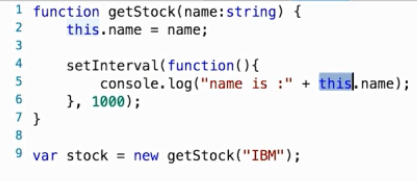
reduce() 方法接收一个函数作为累加器，数组中的每个值（从左到右）开始缩减，最终计算为一个值。

1. concat
2. assign

3.为什么 ["1", "2", "3"].map(parseInt) 返回 [1,NaN,NaN]？

<https://blog.csdn.net/justjavac/article/details/19473199>

4.结果为啥是输出空的



1. bind方法

<http://www.css88.com/react/docs/handling-events.html>

1. 重点

<https://www.liaoxuefeng.com/wiki/001434446689867b27157e896e74d51a89c25cc8b43bdb3000/001435119854495d29b9b3d7028477a96ed74db95032675000>

1. 数组的reduce方法，这个方法是数组的

这个函数的意思是前面的结果作为下一次运算的第一个参数

1. js里函数的高阶用法里，函数可以做参数，也可以做返回值
2. JavaScript Array filter() 方法

filter() 方法创建一个新的数组，新数组中的元素是通过检查指定数组中符合条件的所有元素。

**注意：** filter() 不会对空数组进行检测。

**注意：** filter() 不会改变原始数组。

1. generator函数

http://www.ruanyifeng.com/blog/2015/04/generator.html