## 1.JavaScript 写一个从0到59依次循环的计时器

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Document</title>  
 <style>  
 input{  
 border: 2px solid black;  
 text-align: right;  
 font-size: 20px;  
 font-weight: bold;  
 }  
 </style>  
</head>  
<body>  
   
 seconds:<input type="text" value="0">  
 <script>  
 // 定义参数，获取输入框中的value值  
   
 var secondsValue = document.getElementsByTagName('input')[0];  
 // 赋值为0  
 secondsValue.value =0;  
 // 定义计时器，每秒调用一次  
 var timer = setInterval(function () {  
 // value值自增  
 secondsValue.value ++;  
 // 判定条件，value值大于0  
 if(secondsValue.value > 59){   
 // 清除计时器  
 clearInterval(timer);  
 // value值置0  
 secondsValue.value = 0;  
 // 页面自动刷新，解决数字不动，手动刷新问题  
 location.reload([true])   
 }  
   
 },1000)  
  
  
 </script>  
</body>  
</html>

## 2.用纯CSS创建一个三角形的原理是什么?

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>css实现三角形原理</title>  
 <style>  
 /\* 原理：定义一个块状元素，将宽高都置0 ，通过设置边框的宽度，显示三角形 \*/  
 div {   
 border:12px solid;   
 border-color:transparent red transparent transparent;   
 height:0px;   
 width:0px;   
 }  
   
 </style>  
</head>  
  
<body>  
 <div></div>  
</body>  
</html>

## 3.title 与 h1的区别、b 与 strong 的区别、i 与 em 的区别？

转载：https://www.cnblogs.com/1463069300limingzhi/p/10853675.html

title与h1的区别

定义：

title是网站标题，

h1是文章主题

作用：

title概括网站信息，可以直接告诉搜索引擎和用户这 个网站是关于什么主题和内容的，是显示在网页Tab栏里的；

h1突出文章主题，面对用户，更突出其视觉效果，指向 页面主体信息，是显示在网页中的。

b与strong的区别

定义：

b(bold)是实体标签，用来给文字加粗，

strong是逻辑标签，作用是加强字符语气。

区别：

b标签只是加粗的样式，没有实际含义，常用来表达无强调或着重意味的粗体文字，比如文章摘要中的关键词、 评测文章中的产品名称、文章的导言；

strong表示标签内字符重要，用以强调，其默认格式是加粗，但是可以通 过CSS添加样式，使用别的样式强调。

建议：为了符合CSS3的规范，b应尽量少用而改用strong

i与em的区别

定义：

i(italic)是实体标签，用来使字符倾斜，

em(emphasis)是逻辑标签，作用是强调文本内容

区别：

i标签只是斜体的样式，没有实际含义，常用来表达无强调或着重意味的斜体，比如生物学名、术语、外来语；

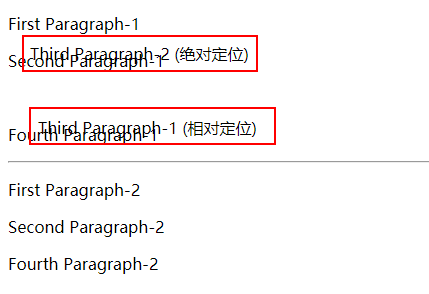
em表示标签内字符重要，用以强调，其默认格式是斜体，但是可以通过CSS添加样式。

建议：为了符合CSS3的规 范，i应尽量少用而改用em

## 4.relative和absolute的区别

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>绝对定位和相对定位的区别</title>  
 <style>  
 .p\_relative{  
 position: relative;  
 top:30px;  
 left:30px;  
 }  
 .p\_absolute{  
 position: absolute;  
 top:30px;  
 left:30px;  
 }  
 </style>  
</head>  
<body>  
 <!-- 1 在没有给div设置position属性前，默认为static,每个元素都处在常规文档流中 -->  
 <!-- 一下各元素都是块级元素，所以在页面中从上到下的堆叠在一起 -->  
 <!-- <p>First Paragraph</p>   
 <p>Second Paragraph</p>   
 <p >Third Paragraph (with special)</p>   
 <p>Fourth Paragraph</p>  
 <hr> -->  
 <!--2 相对定位 -->  
 <!-- 相对定位下，这个元素只是相对于该元素自己的原始位置挪动了了位置，页面 不会发生其他的变化。这个元素原来占据的空间没动，其他元素也没动-->  
 <p>First Paragraph-1</p>   
 <p>Second Paragraph-1</p>   
 <p class="p\_relative">Third Paragraph-1 (相对定位)</p>   
 <p>Fourth Paragraph-1</p>  
 <hr>  
 <!--2 绝对定位 -->  
 <!-- 绝对定位下，元素之前占据的空间会被"回收"，绝对定位的元素完全脱离常规文档流，他会相对于顶级元素body定位-->  
 <p>First Paragraph-2</p>   
 <p>Second Paragraph-2</p>   
 <p class="p\_absolute">Third Paragraph-2 (绝对定位)</p>   
 <p>Fourth Paragraph-2</p>  
</body>  
</html>

效果图：



相对定位相对于自己曾经在文档流中的位置发生变化

绝对定位，完全脱离文档流，相对于body根元素发生变化

## 5.在 CSS 样式中 px、em 、rem的区别

转载：https://www.cnblogs.com/leejersey/p/3662612.html

PX特点

1. IE无法调整那些使用px作为单位的字体大小；
2. 国外的大部分网站能够调整的原因在于其使用了em或rem作为字体单位；
3. Firefox能够调整px和em，rem，但是96%以上的中国网民使用IE浏览器(或内核)。

* px像素（Pixel）。相对长度单位。像素px是相对于显示器屏幕分辨率而言的。(引自CSS2.0手册)
* em是相对长度单位。相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。(引自CSS2.0手册)

任意浏览器的默认字体高都是16px。所有未经调整的浏览器都符合: 1em=16px。那么12px=0.75em,10px=0.625em。为了简化font-size的换算，需要在css中的body选择器中声明Font-size=62.5%，这就使em值变为 16px\*62.5%=10px, 这样12px=1.2em, 10px=1em, 也就是说只需要将你的原来的px数值除以10，然后换上em作为单位就行了。

EM特点

1. em的值并不是固定的；
2. em会继承父级元素的字体大小。

* 所以我们在写CSS的时候，需要注意两点：

1. body选择器中声明Font-size=62.5%；
2. 将你的原来的px数值除以10，然后换上em作为单位；
3. 重新计算那些被放大的字体的em数值。避免字体大小的重复声明

也就是避免1.2 \* 1.2= 1.44的现象。比如说你在#content中声明了字体大小为1.2em，那么在声明p的字体大小时就只能是1em，而不是1.2em, 因为此em非彼em，它因继承#content的字体高而变为了1em=12px。

rem特点

rem是[CSS3](http://www.html5cn.org/portal.php?mod=list&catid=16)新增的一个相对单位（root em，根em），这个单位引起了广泛关注。这个单位与em有什么区别呢？区别在于使用rem为元素设定字体大小时，仍然是相对大小，但相对的只是HTML根元素。这个单位可谓集相对大小和绝对大小的优点于一身，通过它既可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁反应。目前，除了IE8及更早版本外，所有浏 览器均已支持rem。对于不支持它的浏览器，应对方法也很简单，就是多写一个绝对单位的声明。这些浏览器会忽略用rem设定的字体大小。下面就是

一个例子：

p {font-size:14px; font-size:.875rem;} 注意：

选择使用什么字体单位主要由你的项目来决定，如果你的用户群都使用最新版的浏览器，那推荐使用rem，如果要考虑兼容性，那就使用px,或者两者同时使用。

## 6.Ajax是什么？简述 JavaScript AJAX 的原理

Ajax:A new approach to Web Application ,能够向服务器请求额外的数据而无需加载页面，带来更好的用户体验

Ajax的核心技术：XMLHttpRequest对象(简称XHR)。

XHR为向服务器发出请求和解析服务器响应提供了流畅的接口。

以异步方式从服务器取得更多信息，意味着用户单击后，不用刷新页面也能取得新数据。

可以使用XHR对象取得数据，然后再通过DOM将新数据插入到页面中。

虽然，名字中包含XML的成分，但Ajax通信与数据格式无关；

这种技术就是无需刷新即可从服务器取得数据，但不一定是XML数据

## 7.JavaScript 原型，原型链 ？有什么特点？

**原型：**

我们创建的每一个函数都有一个prototype(原型)属性，这个属性是一个指针，指向一个对象，这个对象的用途是包含可以由特定类型的所有实例共享的属性和方法。

prototype通过调用构造函数创建那个对象实例的原型对象。

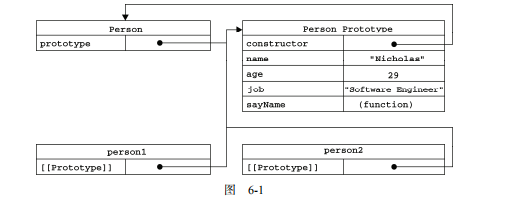
使用原型对象的好处：可以让所有的对象实例共享它所包含的属性和方法，不必在构造函数中定义对象实例的信息，而是可以将这些信息直接添加到原型对象中

function Person(){   
}   
 Person.prototype.name = "Nicholas";   
 Person.prototype.age = 29;   
 Person.prototype.job = "Software Engineer";   
 Person.prototype.sayName = function(){   
 alert(this.name);   
 };   
var person1 = new Person();   
person1.sayName(); //"Nicholas"   
var person2 = new Person();  
person2.sayName(); //"Nicholas"   
alert(person1.sayName == person2.sayName); //true

将sayName()方法和所有属性都直接添加到了Person的prototype中，构造函数为空。

通过调用构造函数创建新对象，新对象还会具有相同的属性和方法，这些属性和方法是由所有的实例共享的。

Person1和Person2访问的是同一组属性和同一个sayName()函数



两个Person实例都不包含属性和方法，但是可以调用personal.sayName(),这是通过查找对象属性的过程来实现的

**原型链：**

基本思想：

利用原型让一个引用类型继承另一个引用类型的属性和方法。

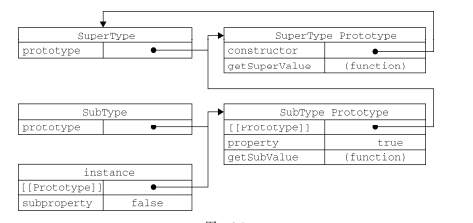
每一个构造函数都有一个原型对象，原型对象都包含一个指向构造函数大呢指针，实例都包含一个指向原型对象的内部指针。假如，我们让原型对象等于另一个类型的实例，此时的远行对象将包含一个指向另一个原型的指针，相应地，另一个原型中也包含一个指向另一个构造函数的指针。假如另一个原型又是另一个对象的实例，那么上述关系仍然成立，层层递进，构成了实例与原型的链条，这就是原型链的基本概念。

function SuperType(){   
 this.property = true;   
}  
SuperType.prototype.getSuperValue = function(){   
 return this.property;   
};   
function SubType(){   
 this.subproperty = false;   
}   
//继承了 SuperType   
SubType.prototype = new SuperType();   
SubType.prototype.getSubValue = function (){   
 return this.subproperty;   
};   
var instance = new SubType();   
alert(instance.getSuperValue()); //true

两个类型：SuperType和SubType。每个类型分别有一个属性和一个方法。

主要区别：SubType继承了SuperType，继承是通过创建SuperType实例，并将该实例赋给SubType.prototype实现的。

实现的本质是重写原型对象，代之以一个新类型的实例



在上述代码中，我们没有使用SubType默认提供的原型，而是给他换了一个新的原型:SuperType的实例。

于是这个原型不仅具有作为一个SuperType的实例所拥有的全部属性和方法，而且其内部还存在一个指针，指向了SuperType的原型。最终结果：instance指向SubType的原型，SubType的原型又指向SuperType的原型。

propetry是一个实例属性，getSuperValue()则是一个原型方法。因为SubType.prototype现在是SuperType的实例，所以property位于该实例当中。注意：instance.constructor指向的是SuperType，这是因为原来的SubType.prototype中的constructor被重写的缘故

## 8.css 伪类与伪元素区别

转载：https://www.cnblogs.com/zhaofeis/p/7725615.html

伪类和伪元素的根本区别在于：**它们是否创造了新的元素(抽象)**。从我们模仿其意义的角度来看，如果需要添加新元素加以标识的，就是伪元素，反之，如果只需要在既有元素上添加类别的，就是伪类。

最为混淆的，可能是大部分人都将 :before 和 :after 这样的伪元素随口叫做伪类，而且即使在概念混淆的情况下，实际使用上也毫无问题——因为即使概念混淆，对真正使用也不会造成多少麻烦:) CSS Selector Level 3 为了区分这两者的混淆，而特意用冒号加以区分： 伪类用一个冒号表示 :first-child 伪元素则使用两个冒号表示 ::first-line

目前为止，伪元素在一个选择器里只能出现一次，并且只能出现在末尾。实则，伪元素是选中了某个元素的符合逻辑的某个实际却不存在的部分，所以应用中也不会有人将其误写成多个。伪类则是像真正的类一样发挥着类的作用，没有数量上的限制，只要不是相互排斥的伪类，也可以同时使用在相同的元素上。

伪类的例子有：

:hover

:active

:first-child

:visited

等。

伪元素的例子有：

:first-line

:first-letter

:after

:before

## 10.Javascript 中的闭包是什么

闭包：有权访问另一个函数作用域中的变量的函数

创建闭包的常见方式：在一个函数内部创建另一个函数

createComparisonFunction()  
function createComparisonFunction(propertyName) {   
   
 return function(object1, object2){   
 var value1 = object1[propertyName];   
 var value2 = object2[propertyName];   
   
 if (value1 < value2){   
 return -1;   
 } else if (value1 > value2){   
 return 1;   
 } else {   
 return 0;   
 }   
 };   
}

var value1 = object1[propertyName]; var value2 = object2[propertyName];

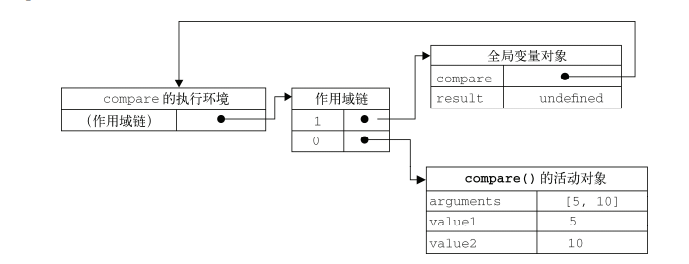
这两行代码是内部函数(一个匿名函数)中的代码，他们访问了外部函数中的变量propertyName.即使这个内部函数被返回了，而且是被其他地方调用了，但他仍然可以访问变量propertyName。原因在于：内部函数的作用域链中包含createComparisonFunction()的作用域。

作用域链

当某个函数被调用时，会创建一个执行环境及相应的作用域链，然后，使用arguments和其他命名的参数的值来初始化函数的活动对象。但在作用域链中，外部函数的活动对象始终处于第二位，外部函数的外部函数的活动对象处于第三位，...直至作为作用域链终点的全局执行环境

在函数执行过程中，为读取和写入变量的值，就需要在作用域链中查找变量。

function compare(value1, value2){   
 if (value1 < value2){   
 return -1;   
 } else if (value1 > value2){   
 return 1;   
 } else {   
 return 0;   
 }   
}   
var result = compare(5, 10);

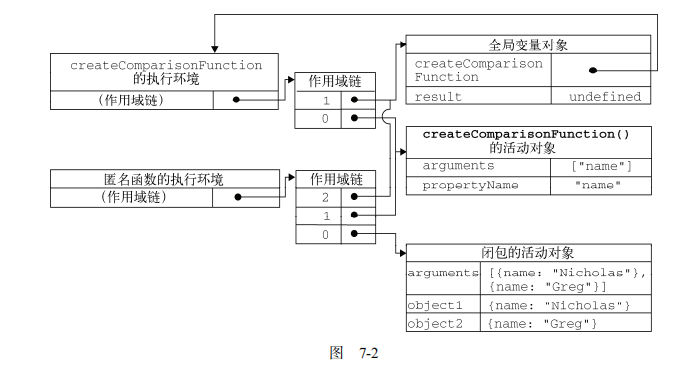


在另一个函数内部定义的函数会将包含函数（即外部函数）的活动对象添加到它的作用域链中。因 此，在 createComparisonFunction()函数内部定义的匿名函数的作用域链中，实际上将会包含外部 函数 createComparisonFunction()的活动对象。

var compare = createComparisonFunction("name");   
var result = compare({ name: "Nicholas" }, { name: "Greg" });

在匿名函数从 createComparisonFunction()中被返回后，它的作用域链被初始化为包含 createComparisonFunction()函数的活动对象和全局变量对象。这样，匿名函数就可以访问在 createComparisonFunction()中定义的所有变量。更为重要的是，createComparisonFunction() 函数在执行完毕后，其活动对象也不会被销毁，因为匿名函数的作用域链仍然在引用这个活动对象。换 句话说，当 createComparisonFunction()函数返回后，其执行环境的作用域链会被销毁，但它的活 动对象仍然会留在内存中；直到匿名函数被销毁后，createComparisonFunction()的活动对象才会 被销毁

//创建函数  
var compareNames = createComparisonFunction("name");   
//调用函数  
var result = compareNames({ name: "Nicholas" }, { name: "Greg" });   
//解除对匿名函数的引用（以便释放内存）  
compareNames = null;



**注意**

由于闭包会携带包含它的函数的作用域，因此会比其他函数占用更多的内存。谨慎使用闭包。

## 11.js中的同步和异步的区别

转载：https://www.cnblogs.com/lchsirblog/p/12048695.html

**同步和异步的差别就在于这条流水线上各个流程的执行顺序不同**。 **同步任务指的是，在主线程上排队执行的任务，只有前一个任务执行完毕，才能执行后一个任务；异步任务指的是，不进入主线程、而进入"任务队列"（task queue）的任务，只有等主线程任务执行完毕，"任务队列"开始通知主线程，请求执行任务，该任务才会进入主线程执行。**

这里说到了一个“队列”（即任务队列），该队列放的是什么呢，放的就是setTimeout中的function，这些function依次加入该队列，即该队列中所有function中的程序将会在该队列以外的所有代码执行完毕之后再以此执行，这是为什么呢？因为在执行程序的时候，浏览器会默认setTimeout以及ajax请求这一类的方法都是耗时程序（尽管可能不耗时），将其加入一个队列中，该队列是一个存储耗时程序的队列，在所有不耗时程序执行过后，再来依次执行该队列中的程序。

又回到了最初的起点——javascript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。如果前一个任务耗时很长，后一个任务就不得不一直等着。于是就有一个概念——任务队列。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。于是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。

具体来说，异步运行机制如下：

（1）所有同步任务都在主线程上执行，形成一个执行栈（execution context stack）。 （2）主线程之外，还存在一个"任务队列"（task queue）。只要异步任务有了运行结果，就在"任务队列"之中放置一个事件。 （3）一旦"执行栈"中的所有同步任务执行完毕，系统就会读取"任务队列"，看看里面有哪些事件。那些对应的异步任务，于是结束等待状态，进入执行栈，开始执行。 （4）主线程不断重复上面的第三步。

## 12.let ，const ，var 有什么区别

转载：https://blog.csdn.net/unionz/article/details/80032048?utm*medium=distribute.pc*relevant*t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.channel*param&depth*1-utm*source=distribute.pc*relevant*t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.channel\_param

**1. 块级作用域 {}**

ES5 中作用域有：全局作用域、函数作用域。没有块作用域的概念。

ES6 中新增了块级作用域。块作用域由 { } 包括，if语句和 for语句里面的{ }也属于块作用域。

<script type="text/javascript">  
   
 {  
 var a = 1;  
 console.log(a); // 1  
 }  
 console.log(a); // 1  
 // 通过var定义的变量可以跨块作用域访问到。  
   
 (function A() {  
 var b = 2;  
 console.log(b); // 2  
 })();  
 // console.log(b); // 报错，  
 // 可见，通过var定义的变量不能跨函数作用域访问到  
   
 if(true) {  
 var c = 3;  
 }  
 console.log(c); // 3  
 for(var i = 0; i < 4; i ++) {  
 var d = 5;  
 };  
 console.log(i); // 4 (循环结束i已经是4，所以此处i为4)  
 console.log(d); // 5  
 // if语句和for语句中用var定义的变量可以在外面访问到，  
 // 可见，if语句和for语句属于块作用域，不属于函数作用域。  
   
</script>

**2. var、let、const的区别**

1. var定义的变量，没有块的概念，可以跨块访问, 不能跨函数访问。
2. let定义的变量，只能在块作用域里访问，不能跨块访问，也不能跨函数访问。
3. const用来定义常量，使用时必须初始化(即必须赋值)，只能在块作用域里访问，而且不能修改。

<script type="text/javascript">  
 // 块作用域  
 {  
 var a = 1;  
 let b = 2;  
 const c = 3;  
 // c = 4; // 报错  
 var aa;  
 let bb;  
 // const cc; // 报错  
 console.log(a); // 1  
 console.log(b); // 2  
 console.log(c); // 3  
 console.log(aa); // undefined  
 console.log(bb); // undefined  
 }  
 console.log(a); // 1  
 // console.log(b); // 报错  
 // console.log(c); // 报错  
   
 // 函数作用域  
 (function A() {  
 var d = 5;  
 let e = 6;  
 const f = 7;  
 console.log(d); // 5  
 console.log(e); // 6   
 console.log(f); // 7   
   
 })();  
 // console.log(d); // 报错  
 // console.log(e); // 报错  
 // console.log(f); // 报错  
</script>

## 13.JS执行上下文和执行栈的理解

参考：https://www.cnblogs.com/echolun/p/11438363.html

## 14.js里==和===有什么区别

转载：https://blog.csdn.net/fjh19950514/article/details/84198190?utm*medium=distribute.pc*relevant.none-task-blog-title-3&spm=1001.2101.3001.4242

==：运算符称作相等，用来检测两个操作数是否相等，这里的相等定义的非常宽松，可以允许进行类型转换 ===：用来检测两个操作数是否严格相等 1、对于string,number等基础类型，==和===是有区别的 不同类型间比较，==之比较“转化成同一类型后的值”看“值”是否相等，===如果类型不同，其结果就是不等 同类型比较，直接进行“值”比较，两者结果一样 2、对于Array,Object等高级类型，==和===是没有区别的 3、基础类型与高级类型，==和===是有区别的 对于==，将高级转化为基础类型，进行“值”比较，因为类型不同，===结果为false