nodeTech.节点科技

前端岗前培训

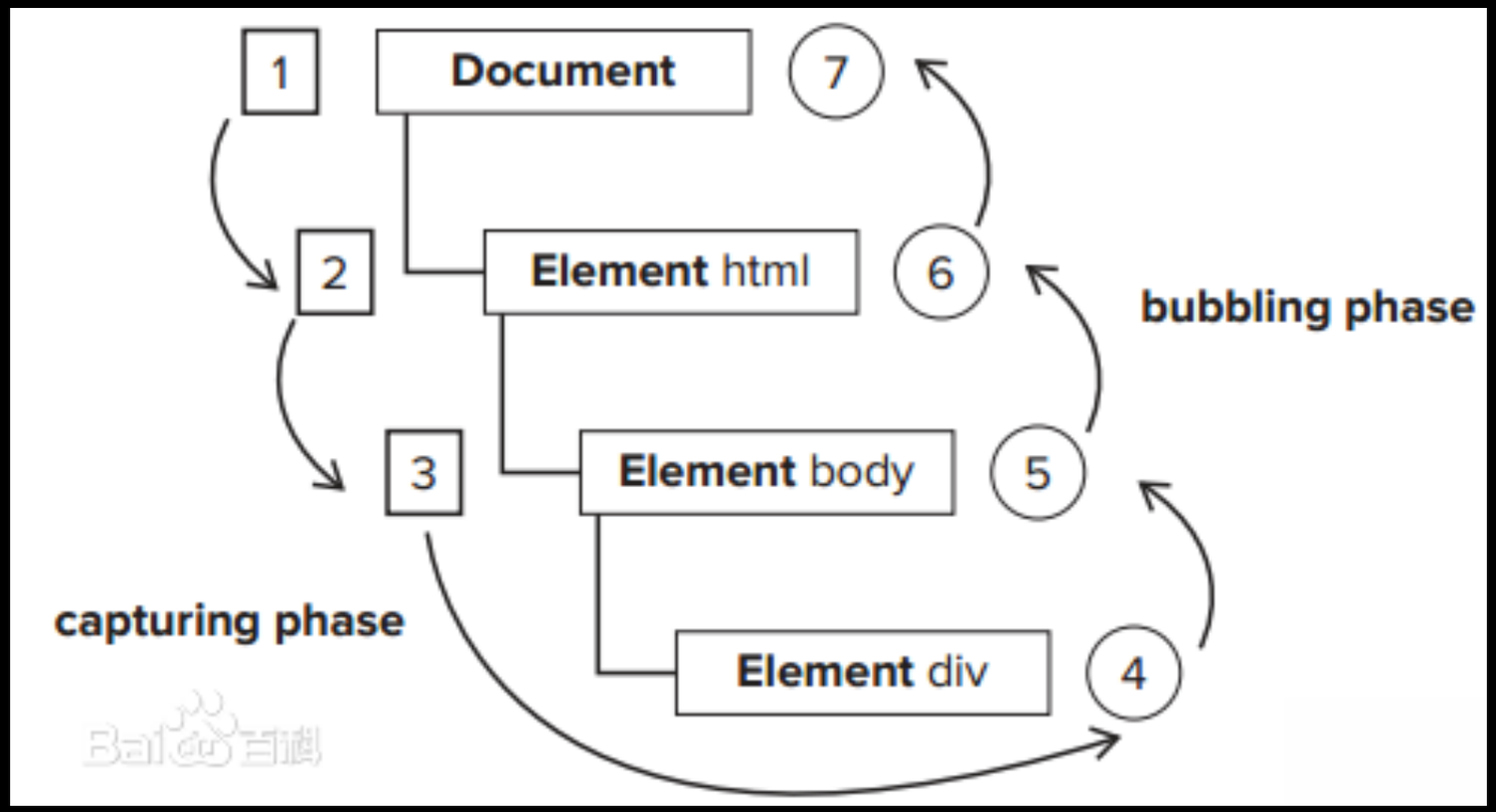
第五课 JS高级 jQuery

节点科技.

作业点评

UNICRONCLASS

事件冒泡 ,阻止事件冒泡以及阻止默认行为




```
        //js阻止事件冒泡
        //oEvent.cancelBubble = true;
        //oEvent.stopPropagation();

        //js阻止链接默认行为，没有停止冒泡
        //oEvent.preventDefault();
        //return false;
```

https://www.cnblogs.com/jsanntq/p/7681942.html

即时函数

```
<body>
    <script>
        //写法1
        (function () {
            alert('watch out!');
            alert(this);  //这里的this是window对象
        }());


        //写法2
        (function () {
            alert('watch out2!');
        })();

        //带参数的即时函数
        (function (who, when) {
            alert("I met " + who + " on " + when);
        }("Lilu", new Date()));


        //全局对象以参数方式传递给即时函数，使代码在浏览器环境之外具有更好的操作性
        (function (global) {
            alert(global);
        }(this));
    </script>
</body>
```

防止变量污染的模块化定义

```
//即时函数也叫 子调用函数 或 自执行函数
//即时函数的优点就是定义的所有变量将会是即时函数的局部变量，不用担心全局空间被临时的变量污染

//文件module1.js中定义的模块module1
(function () {
    //模块1的所有代码
}());
```

即时对象初始化

```
({
    maxwidth: 600,
    maxheight: 400,

    gimmeMax: function () {
        return this.maxwidth + 'X' + this.maxheight;
    },

    //初始化
    init: function () {
        alert(this.gimmeMax());
    }
}).init();
```

apply 和 call

call是apply的语法糖

```
function add(x, y, z) {
    console.log(this);
    console.log(x + y + z);
}


//调用函数
add(5, 4, 3);


//应用函数
//apply有两个参数，第一个参数为将要绑定到该函数内部this的一个对象，第二个参数是一个数组
//如果第一个参数为null，那么this将指向全局对象
add.apply(null, [1, 2, 3]);

//call()方法是apply()的语法糖
add.call(null, 1, 2, 3);


var math = {};
add.apply(math, [3, 2, 3]);
```

模块化编程 原始写法

```
//原始写法
//只要把不同的函数（以及记录状态的变量）简单地放在一起，就算是一个模块。
//这种做法的缺点很明显："污染"了全局变量，无法保证不与其他模块发生变量名冲突，而且模块成员之间看不出直接关系。

var globalvar = "这是全局变量";

function m1() {
  //...
  globalvar = "被m1修改";
}
function m2() {
  //...
  globalvar = "被m2修改";
}

console.log(globalvar);
m1();
m2();
console.log(globalvar);
```

模块化编程 对象写法

```
//对象写法
//可以把模块写成一个对象，所有的模块成员都放到这个对象里面。
var module1 = new Object({
  _count: 0,
  m1: function () {
    //...
    console.log('m1');
  },
  m2: function () {
    //...
    console.log('m2');
  }
});

//函数m1()和m2()，都封装在module1对象里。使用的时候，就是调用这个对象的属性
module1.m1();

//但是，这样的写法会暴露所有模块成员，内部状态可以被外部改写。比如，外部代码可以直接改变内部计数器的值。
module1._count = 5;
console.log(module1);
```

模块化编程 立即执行函数写法

```
//使用"立即执行函数" 可以达到不暴露私有成员的目的
var module1 = (function () {
    var _count = 0;
    var m1 = function () {
        //...
        console.log('m1');
    };
    var m2 = function () {
        //...
        console.log('m2');
    };
    return {
        m1: m1,
        m2: m2,
        m: m1
    };
})();

//外部代码无法读取内部的_count变量。
console.info(module1._count); //undefined

module1.m1();
module1.m2();
module1.m();
```

模块化编程 放大模式

```
//放大模式
//如果一个模块很大，必须分成几个部分，或者一个模块需要继承另一个模块，这时就有必要采用"放大模式"
var module1 = (function (mod) {
    mod.m3 = function () {
        //...
        console.log('m3');
    };
    return mod;
})(module1);
```

模块化编程 输入全局变量

```
//输入全局变量
//module1模块需要使用jQuery库和YUI库，就把这两个库（其实是两个模块）当作参数输入module1
var jQuery = {
    origin: function () {
        console.log('jquery origin function');
    }
};
var YAHOO = {
    origin: function () {
        console.log('YAHOO origin function');
    }
};

var module1 = (function ($, YAHOO) {
    //...
    $.add = function () {
        this.origin();
        console.log('jquery add');
    };
    YAHOO.add = function () {
        this.origin();
        console.log('YAHOO add');
    }

})(jQuery, YAHOO);

jQuery.add();

YAHOO.add();
```

模块规范：CommonJS和AMD

- node.js的模块系统，就是参照CommonJS规范实现的
- 在CommonJS中，有一个全局性方法require()，用于加载模块
- CommonJS规范不适用于浏览器环境
- 浏览器端的模块，不能采用”同步加载"（synchronous），只能采用"异步加载"（asynchronous）

AMD Asynchronous Module Definition

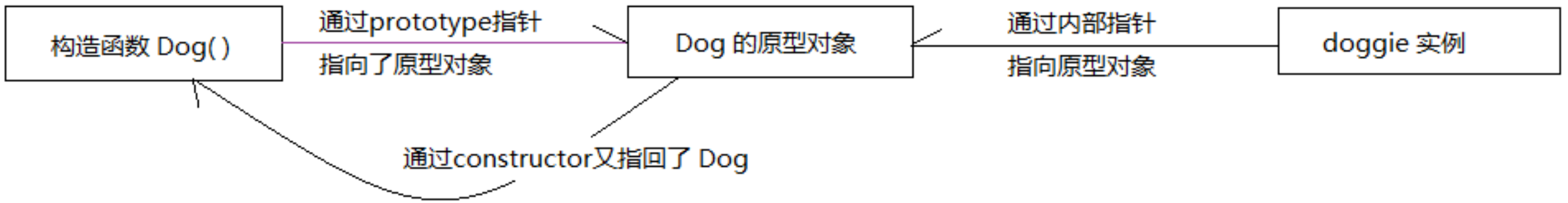
- 主要有两个Javascript库实现了AMD规范：require.js和curl.js

```
//CommonJS规范 同步加载
var math = require('math');
math.add(2, 3);

//AMD也采用require()语句加载模块,它要求两个参数:
//第一个参数[module]，是一个数组，里面的成员就是要加载的模块；第二个参数callback，则是加载成功之后的回调函数
require(['math'], function (math) {
  math.add(2, 3);
});
```

require.js的用法

- 实现js文件的异步加载，避免网页失去响应
- 管理模块之间的依赖性，便于代码的编写和维护
- http://requirejs.org/docs/download.html
- <script src="js/require.js" defer async="true" data-main=“js/main”></script>
- 模块必须采用特定的define()函数来定义

JS原型链与继承

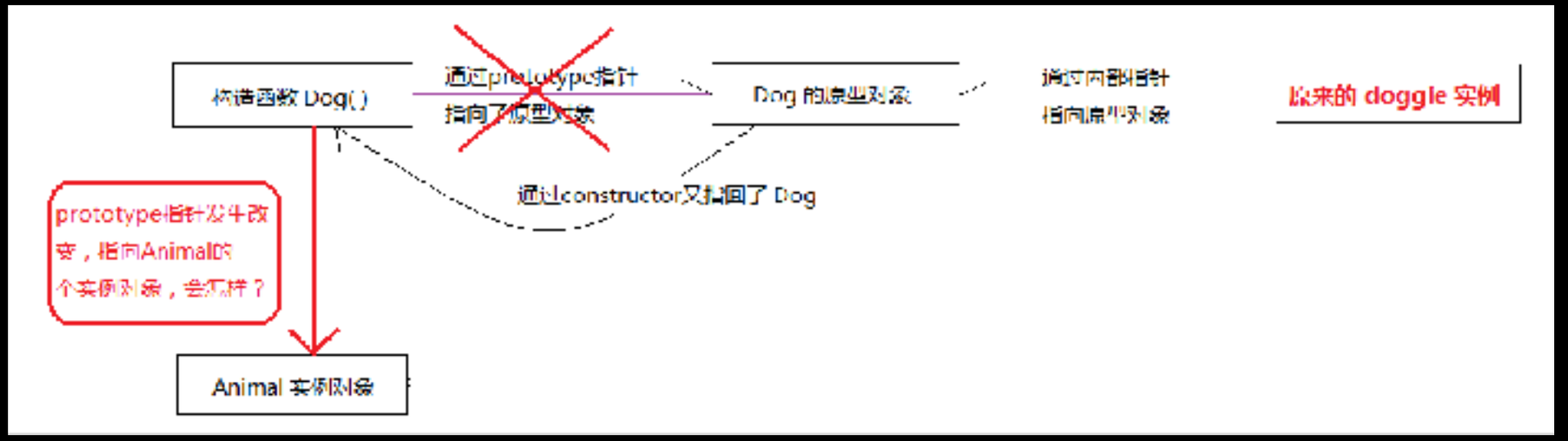



```
function Dog(name) {
    this.name = name;
    this.type = 'Dog';
}
Dog.prototype.speak = function () {
    console.log('dog speak wang wang ');
}
var doggie = new Dog('xiao hei');
doggie.speak();

//每个构造函数都有一个原型对象 .prototype
console.info(Dog.prototype);

//原型对象都包含一个指向构造函数的指针 constructor
//即 Dog.prototype.constructor == Dog


//实例都包含一个指向原型对象的内部指针 __proto__
console.log(doggie);
```

JS原型链与继承

doggie实例指向了Dog的原型对象，可以访问Dog原型对象上的所有属性和方法；如果Dog原型对象变成了某一个类的实例 aaa，这个实例又会指向一个新的原型对象 AAA，那么 doggie 此时就能访问 aaa 的实例属性和 AA A原型对象上的所有属性和方法了。同理，新的原型对象AAA碰巧又是另外一个对象的实例bbb，这个实例bbb又会指向新的原型对象 BBB，那么doggie此时就能访问 bbb 的实例属性和 BBB 原型对象上的所有属性和方法了。

就是JS通过原型链实现继承的方法



JS原型链与继承

```
//定义一个Animal 构造函数 作为 Dog的父类
function Animal() {
    this.superType = 'Animal';
}

Animal.prototype.superSpeak = function () {
    alert(this.superType);
}


function Dog(name) {
    this.name = name;
    this.type = 'Dog';
}

//改变Dog的prototype指针, 指向一个Animal实例
Dog.prototype = new Animal();

Dog.prototype.speak = function () {
    alert(this.type);
}

var doggie = new Dog('xiao hei');
doggie.superSpeak();
console.log(doggie.superType);
```

什么是闭包

- 可以访问外部函数作用域中变量的函数
- 被内部函数访问的外部函数的变量可以保存在外部函数作用域内而不被回收---这是核心，后面我们遇到闭包都要想到，我们要重点关注被闭包引用的这个变量。

什么是闭包

- 可以访问外部函数作用域中变量的函数

- 被内部函数访问的外部函数的变量可以保存在外部函数作用域内而不被回收---这是核心，后面我们遇到闭包都要想到，我们要重点关注被闭包引用的这个变量。

```
//被内部函数访问的外部函数的变量可以保存在外部函数作用域内而不被回收
var sayName = function () {
    var name = 'jozo';
    return function () {
        alert(name);
    }
};
var say = sayName();
say();
```

闭包的实例

```
// 方式1
var a = 0;
var add = function () {
    a++;
    console.log(a)
}
add();
add();
add();
add();
//方式2 : 闭包
//相比之下方式2更加优雅，也减少全局变量，将变量私有化
var add = (function () {
    var b = 0;
    return function () {
        b++;
        console.log(b);
    }
})();
//console.log(b); //undefined
add();
add();
add();
add();
```

闭包的实例

```html
<ul>
    <li>一</li>
    <li>二</li>
    <li>三</li>
    <li>四</li>
    <li>五</li>
</ul>
```

```html
<script>
    var oli = document.getElementsByTagName('li');
    var i;
    for (i = 0; i < 5; i++) {
        oli[i].onclick = function () {
            alert(i);
            alert(this.innerHTML);
        }
    }

    console.log(i);
    //想想为什么每次弹出都是5
</script>
```

闭包的实例

```
//运用闭包
//这里for循环执行时，给点击事件绑定的匿名函数传递i后立即执行返回一个内部的匿名函数，因为参数是按值传递的，所以此时形参num保存的就是当前i的值,然后赋值给局部变量 a，然后这个内部的匿名函数一直保存着a的引用，也就是一直保存着当前i的值。
var oli = document.getElementsByTagName('li');
var i;
for (i = 0; i < 5; i++) {
    oli[i].onclick = (function (num, that) {
        var a = num; // 为了说明问题
        return function () {
            alert(a);
            alert(that.innerHTML);

        }
    })(i, oli[i])
}
console.log(i); //
```

内存泄露及解决方案

- 标记清除:垃圾收集器在运行的时候会给存储在内存中的所有变量都加上标记，然后，它会去掉环境中的变量的标记和被环境中的变量引用的变量的标记，此后，如果变量再被标记则表示此变量准备被删除。 2008年为止，IE，Firefox，opera，chrome，Safari的javascript都用使用了该方式；

- 引用计数:跟踪记录每个值被引用的次数，当声明一个变量并将一个引用类型的值赋给该变量时，这个值的引用次数就是1，如果这个值再被赋值给另一个变量，则引用次数加1。相反，如果一个变量脱离了该值的引用，则该值引用次数减1，当次数为0时，就会等待垃圾收集器的回收. 这个方式存在一个比较大的问题就是循环引用，就是说A对象包含一个指向B的指针，对象B也包含一个指向A的引用。 这就可能造成大量内存得不到回收（内存泄露），因为它们的引用次数永远不可能是 0 。早期的IE版本里（ie4-ie6）采用是计数的垃圾回收机制，闭包导致内存泄露的一个原因就是这个算法的一个缺陷。访问COM对象依然是基于引用计数的，因此只要在IE中设计COM对象就会存在循环引用的问题!

内存泄露及解决方案

- 标记清除:垃圾收集器在运行的时候会给存储在内存中的所有变量都加上标记，然后，它会去掉环境中的变量的标记和被环境中的变量引用的变量的标记，此后，如果变量再被标记则表示此变量准备被删除。 2008年为止，IE，Firefox，opera，chrome，Safari的javascript都用使用了该方式；

- 引用计数:跟踪记录每个值被引用的次数，当声明一个变量并将一个引用类型的值赋给该变量时，这个值的引用次数就是1，如果这个值再被赋值给另一个变量，则引用次数加1。相反，如果一个变量脱离了该值的引用，则该值引用次数减1，当次数为0时，就会等待垃圾收集器的回收. 这个方式存在一个比较大的问题就是循环引用，就是说A对象包含一个指向B的指针，对象B也包含一个指向A的引用。 这就可能造成大量内存得不到回收（内存泄露），因为它们的引用次数永远不可能是 0 。早期的IE版本里（ie4-ie6）采用是计数的垃圾回收机制，闭包导致内存泄露的一个原因就是这个算法的一个缺陷。访问COM对象依然是基于引用计数的，因此只要在IE中设计COM对象就会存在循环引用的问题!

内存泄露及解决方案

```
//执行这段代码的时候，将匿名函数对象赋值给el的onclick属性；然后匿名函数内部又引用了el对象，存在循环引用，所以
不能被回收
window.onload = function () {
    var el = document.getElementById("id");
    el.onclick = function () {
        alert(el.id);
    }
}

//解决方法:
window.onload = function () {
    var el = document.getElementById("id");
    var id = el.id; //解除循环引用
    el.onclick = function () {
        alert(id);
    }
    el = null; // 将闭包引用的外部函数中活动对象清除
}
```

闭包的优缺点

优点：

可以让一个变量常驻内存 (如果用的多了就成了缺点
避免全局变量的污染
私有化变量

缺点:

因为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
引起内存泄露

jQuery包括

核心库 http://jquery.com/

UI http://jqueryui.com/

插件 http://plugins.jquery.com/

jQuery Mobile http://jquerymobile.com/

$是个什么鬼

```
<script>

 function test(tag){
    alert("test"+tag);
 }
 function $(tag){
     alert("我是个$"+tag);
 }

 test('#container');
 $('#container');
</script>
```

就是一个函数名而已

模仿jQuery实现自己的$()函数

```
<script>
function $(tag){
    var string=tag.substr(0,1);
     switch(string){
        case "#":
        return document.getElementById(tag.substring(1));
        break;
        case ".":
        return document.getElementsByClassName(tag.substring(1));
        break;
        default:
        return document.getElementsByTagName(tag);
    }
}

window.onload=function(){
  alert($('.box')[0].innerHTML);
  alert($('#box').innerHTML);
  alert($('p')[1].innerHTML);
  $('button')[0].onclick=function(){
      alert("哎呦，我被点了一下");
  }
}

</script>
```

```
<body>
<button>我是按钮</button>
<div class='box' >
第一个 class=box的 元素内容
</div>
<div class='box' >
第二个 class=box的 元素内容
</div>
<div id='box' >
id=box的 元素内容
</div>
<p>第一个P标签</p>
<p>第二个P标签</p>
</body>
```

实现自己js库

lilu.js

```
function $(tag){
                var string=tag.substr(0,1);
                switch(string){
                  case "#":
                  return document.getElementById(tag.substring(1));
                  break;
                  case ".":
                  return document.getElementsByClassName(tag.substring(1));
                  break;
                  default:
                  return document.getElementsByTagName(tag);
    }
}

```

```
<script src="lilu.js"></script>
<script>
   window.onload=function(){
     $('button')[0].onclick=function(){
         $('#container').innerHTML="<b>哈哈，你已经被我修改了</b>";
         $('#container').setAttribute('class','select');
         $('.box')[0].innerHTML="第一个盒子也被改变了";
     }
   }
</script>
```

jQuery的使用

到官网下载http://jquery.com/download/

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <script src="jquery-2.1.4.js"></script>
        <script>
            $(document).ready(function(){
                //等待DOM元素加载完毕
                alert('Hello World!');
                $('.container .box').html('<a href="#">这里被我改了</a>');
            });

        </script>
    </head>
    <body>
        <div class="container">
            <div class="box">
            <a href="#">这里是内容</a>
          </div>
        </div>
    </body>
</html>
```

Window.onload与$(document).ready()的对比

window.onload和$(document).ready()的对比

	window.onload	$(document).ready()
执行时机	必须等待网页中所有的内容加载完毕（包括图片），才能执行	网页中所有DOM结构绘制完毕之后就执行，可能DOM元素关联的东西并没加载完

编写个数	不能同时编写多个
	以下代码无法正常执行：
	window.onload=function(){
	alert(" test1");
	};
	window.onload=function(){
	alert(" test2");
	};
	结果只会输出test2
		能同时编写多个
		以下代码正确执行：
		$(document).ready(function(){
		alert(" test1");
		});
		$(document).ready(function(){
		alert(" test2");
		});
		结果两次都输出

简化写法	无	$(document).ready (function(){
		//...
		}) ;
		可以简写成：
		$(function(){
		//...
		})

jQuery选择器

结合 CSS选择器 理解

http://www.w3school.com.cn/jquery/jquery_selectors.asp

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
      <script src="jquery-2.1.4.js"></script>
       <style>
            .select{
              background:pink;
            }
      </style>
  </head>
  <body>
        <p>这是一段话</p>
        <p>这是另一段话</p>
        <div id="test">这是id 标签</div>
        <div class="box">这是一个盒子</div>
        <div class="box">这是另一个盒子</div>
        <button type="button">点我</button>

  </body>

  <script>
       $(document).ready(function(){    //等待DOM元素加载完毕
          $('button').click(function(){
              $('p').hide();
              $(".box").css("background-color","red");
              $("#test").addClass("select");
              $("#test").fadeOut(3000);
          });
       });
      </script>
</html>
```

jQuery 效果 - 动画

http://www.w3school.com.cn/jquery/jquery_animate.asp

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
      <script src="jquery-2.1.4.js"></script>

      <script>
      $(document).ready(function(){
          $("button").click(function(){
          $("div").animate({
            left:'250px',
            opacity:'0.5',
            height:'150px',
            width:'150px'
          });
        });
    });
</script>
</head>

<body>

<button>开始动画</button>
<p>默认情况下，所有 HTML 元素的位置都是静态的，并且无法移动。如需对位置进行操作，记得首先把元素的 CSS position 属性设置为 relative、fixed 或 absolute。</p>
<div style="background:#98bf21;height:100px;width:100px;position:absolute;">
</div>
</body>
</html>

```

自学jQuery callback函数

http://www.w3school.com.cn/jquery/jquery_callback.asp

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
      <script src="jquery-2.1.4.js"></script>
      <script type="text/javascript">
      $(document).ready(function(){
        $("button").click(function(){
        $("p").hide(1000,function(){
          alert("The paragraph is now hidden");
          });
        });
      });
      </script>
</head>
<body>
<button type="button">Hide</button>
<p>This is a paragraph with little content.</p>
</body>
</html>
```

常见方法用法

http://www.w3school.com.cn/jquery/jquery_dom_set.asp

css()、addClass()、text()、html() 、 val() 、 append()

```
<script>
    $(document).ready(function(){
      $("#btn1").click(function(){
        $("#test1").text("Hello world!");
        $("#test1").addClass('select');
       // $("#test1").addClass('b');
        //alert($("#test1").text());
       // alert($("#test1").attr('class'));


      });
      $("#btn2").click(function(){
        $("#test2").html("<b>Hello world!</b>");
        $("#test2").css("background-color","red");

        //alert($("#test2").html());
        //alert($("#test2").attr('style'));
      });
      $("#btn3").click(function(){
        $("#test3").val("文本框里的值");
        //alert( $("#test3").val());
      });
    });
    </script>
```

常见方法用法

http://www.w3school.com.cn/jquery/jquery_dom_add.asp

css()、addClass()、text()、html() 、 val() 、 append()

```
<style>
.select{
  background:pink;
}
.b{
  font-weight:bold;
  color:yellow;
}
</style>
```

```
<body>
<p id="test1">这是段落。</p>
<p id="test2">这是另一个段落。</p>
<p>Input field: <input type="text" id="test3" value="Mickey Mouse"></p>
<button id="btn1">设置文本</button>
<button id="btn2">设置 HTML</button>
<button id="btn3">设置值</button>
</body>
```

遍历和AJAX

http://www.w3school.com.cn/jquery/jquery_traversing.asp

http://www.w3school.com.cn/jquery/jquery_ajax_intro.asp

思考 jquery ajax 中的 load()、get()、 post()的异同

AJAX 实例

对上节原生AJAX POST请求实例的改造

http://www.lke.co/backend/teach/js/7-3-1-post.html

http://www.lke.co/backend/teach/js/jquery-4.html

```
<style>
 #msg.show{
  border:1px solid #000;
  background:yellow;
 }
</style>
<script src="jquery-2.1.4.js"></script>
<script>
function CheckPost(){
 //这里写一些基本的验证代码
  return false;
}

function checkuser(){

  $.post("checkuser_post.php",
  {
    username:myform.username.value,
    pwd:"test"
  },
  function(data,status){
    $('#msg').addClass('show');
    $('#msg').html(data);
  });

}
</script>
```

JSON

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。
http://baike.baidu.com/view/136475.htm

JSON符号是Javacript创建对象的三种方法之一
扩展阅读 http://www.jb51.net/article/20428.htm

简单JSON 属性：值

```
var mingxing={name:"刘德华",age:"25",sex:"男"};
alert(mingxing.name);
```

属性值为数组，数组值为对象

```
var somebooks = {
  book:[{name:"三国演义"},{name:"西游记"},{name:"水浒传"},{name:"红楼梦"}],
  author:[{name:"罗贯中"},{name:"吴承恩"},{name:"施耐安"},{name:"曹雪芹"}]
}

alert(somebooks.book[1].name+'的作者是'+somebooks.author[1].name);
```

AJAX 返回值为JSON

改造14页的代码 http://www.1ke.co/backend/teach/js/jquery-6.html
处理返回的JSON数据

```
function CheckPost(){
 //这里写一些基本的验证代码
  return false;
}
function checkuser(){

  $.post("json.php",
  {
    username:myform.username.value,
    pwd:"test"
  },
  function(data,status){
    $('#msg').addClass('show');
    var str="";
    str=data.message;
    str+="<br/>";
    str+="我喜欢的车的品牌有"+data.cars[0]+","+data.cars[1]+","+data.cars[2];
    $('#msg').html(str);
  });

}
</script>
```

AJAX 返回值为JSON

改造代码 http://www.1ke.co/backend/teach/js/jquery-6.html
处理返回的JSON数据

```php
<?php
header("Content-type: application/json;  charset=UTF-8");
$username= $_GET["username"];
$pwd=$_GET["pwd"];

        $a['success']=true;
        $a['message']="申请成功,请等待审核结果!";
        $a['username']=$username;
        $a['password']=$pwd;
        $cars=array("Volvo","BMW","SAAB");
        $a['cars']=$cars;

      echo json_encode($a);

?>
```

作业

作业1 把上次作业改为jquery

作业2 做一个酷炫的todo list ，用户填写好一条，就在页面上显示一条
用到jquery 的选择器
外形可以参考 **http://www.todolist.cn/**

补充学习

http://www.w3school.com.cn/jquery/index.asp