jQuery入门

窦连军 @八月虎baidu
北京乐美无限科技有限公司

jQuery简介

- 是啥？
  - 是一种Javascript库。类似其他的库有：YUI…
  - 例：`$('#myid').css('color','red')`
- 有啥用？
  - 当CSS遇见JavaScript：简化DOM操作，“Write less Do more写得更少，做得更多”。
  - 专注于javascript与HTML的交互

jQuery的建设思想

语法：　$(*selector*).*action*()

定位目标元素，CSS选择器语法

例：

```
$("div").addClass("xyz");
```

施加要执行的动作

jQuery 函数对象

几乎所有操作都归结为一个模式：

1. 定位操作目标。
2. 对目标施加具体操作。

jQuery的建设思想

1. 一般情况下，jQuery对象的方法名既可以用于属性值的设置，又可以用于属性值的读取。
2. 给jQuery方法传入属性值，则性质为setter，即：设置新属性值；而不传入属性值，则性质为getter，即：读取当前属性值；

Write less Do more !

例：

```
var value = $(selector).val (); // 读取当前值
$(selector).val (newValue);  // 设置新值


// 读取当前值
var value = $(selector).css (propertyName);
// 设置新值
$(selector).css (propertyName, newValue);
```

jQuery例子

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <link rel="stylesheet" href="my.css">
    <script type="text/javascript" src="lib/jquery-1.8.3.min.js"></script>
</head>
<body>
<div id="bar-container">
    <span id="my1">1</span><span id="my2">2</span><span id="my3">3</span><span id="my4">4</span><span
    id="my5">5</span><span id="my6">6</span><span id="my7">7</span><span id="my8">8</span><span
    id="my9">9</span><span id="my10">10</span>
</div>

<script type="text/javascript">
    var sortNumArr = new Array(11) ;
    for(var i=1;i<=10;i++){
        var num = Math.random()*(60-10)+10 ;
        sortNumArr[i] = num ;

        $('#my'+i).css('height',num+"%") ;
    }

</script>
<script type="text/javascript" src="sort.js"></script>
</body>
</html>
```

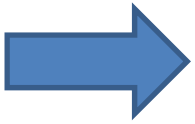
建立开发和调试环境

- IDE： WebStorm
- 运行与调试环境：Chrome浏览器

仅有一个函数（对象）

- 几乎所有的操作都是从jQuery()函数调用开始。
- 由于jQuery()函数调用是如此的频繁，因此将变量$直接作为jQuery函数的别名。

```
jQuery('#nav')
jQuery('div#intro h2')
jQuery('#nav li.current a')
```



```
$('#nav')
$('div#intro h2')
$('#nav li.current a')
```

获得jQuery对象

- 方法1：使用选择器。
  - $(‘XXX’) 返回选择器匹配成功的所有节点元素的集合。该集合本身仍是一个jQuery对象，可以使用jQuery所有的方法和属性。

```
var obj =$('#inputEmail')
obj.css('background-color','#333')
```

- 方法2：使用DOM对象。
  - 可将DOM对象直接传入jQuery函数中，获得相应的jQuery对象。

```
var obj = document.getElementById(‘myId’) ; // 获得DOM对象
$(obj).remove() ;    // 转换为jQuery对象，调用jQuery对象的方法
```

- 方法3：使用HTML片段。
  - 直接将HTML片段作为参数传入jQuery函数中，获得相应的jQuery对象。该对象是游离在DOM之外的对象，需要调用append方法附着在指定的DOM元素下。

```
var j = $('<div><input type="text" value="val" /></div>‘) ;
$(‘body’).append(j) ;
```

可直接使用CSS2，CSS3选择器

- a[rel]
- a[rel="friend"]
- a[href^="http://"]
- ul#nav > li
- li#current ~ li (li元素的兄弟，在#current元素之后)
- li:first-child,li:last-child,li:nth-child(3)

也可使用更强大的选择器

- div:first,h3:last
- :header
- :hidden,:visible
- :animated
- :input,:text,:password,:radio,:submit...
- div:contains(Hello)

见：jQuery选择器大全.pdf　　jQuery强大的jQuery选择器.pdf

访问匹配结果集：遍历

- 获得DOM对象类型的元素
  - 该集合提供了一些类似数组的访问方法，称“伪数组”，伪数组的访问结果是DOM元素。
  - $('div.section').length - 匹配上的元素数量
  - $(‘div.section’)[0] - 第一个DIV **DOM元素对象（非jQuery对象）**
  - $(‘div.section’).**get(0) - 同上**
  - $('div.section')[1]
  - $('div.section')[2]
  - $(‘div.section’).toArray()- 获得**DOM元素对象的集合**。
  - $('div.section').size() = 匹配上的DOM元素数量
  - $('div.section').each(function() {
    console.log(this); // this为DOM元素. 注：each操作速度比较慢。
    });

访问匹配结果集：遍历

- 获得jQuery对象类型的元素
  - jQuery提供了first/last/eq/slice方法，它们与上面提到的get/toArray不同。它们返回的不是HTMLElement，而仍然是jQuery对象，因而也就可以使用jQuery对象的所有方法和属性。
    1. $('div').first() 返回jq对象集合的第一个元素
    2. $('div').last() 返回jq对象集合的最后的元素
    3. $('div').eq(2) 返回jq对象匹配的第三个元素

访问匹配结果集：筛选

- 提供强大的过滤器，对结果集进行筛选，缩小选择结果，其结果仍为jQuery对象。

  - jQuery提供了filter方法。

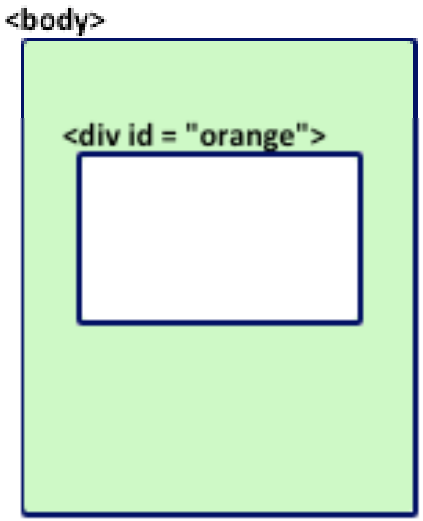
形式	功能
.filter(selector)	筛选出符合选择器要求的元素。
.filter(function(index))	遍历每一个元素，执行指定的函数，如果为true，则放入新集合中。This指DOM形式的元素。
.filter(element)	通过DOM元素来筛选集合中对应的元素。
.filter(jQuery object)	通过一个集合来筛选当前集合中对应的元素。

访问匹配结果集：游历

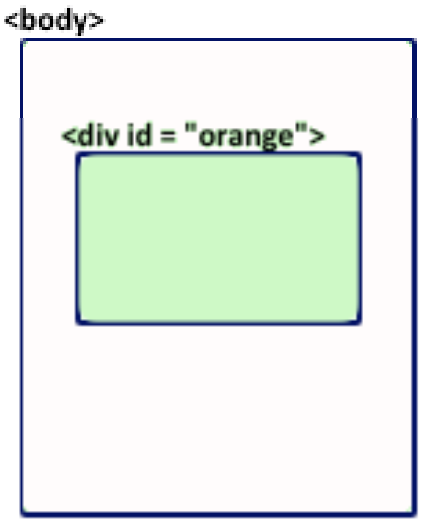
- 有时候，我们需要从结果集出发，移动到附近的相关元素，jQuery也提供了在DOM树上的移动方法（结果仍为jQuery对象）：

用法	说明
$(‘div’).parents()	
$(‘div’).parent()	取div元素的父元素
$(‘div’).next(‘p’)	取div元素后面的第一个p元素
$('div').prev()	
$(‘div’).nextAll(‘div’)	
$('div').first()	选择第1个div元素

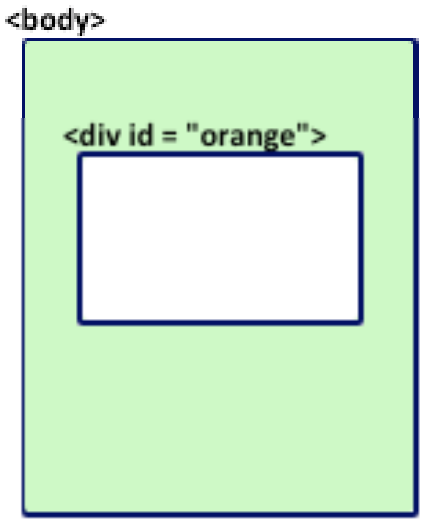
访问匹配结果集：游历



$('body');

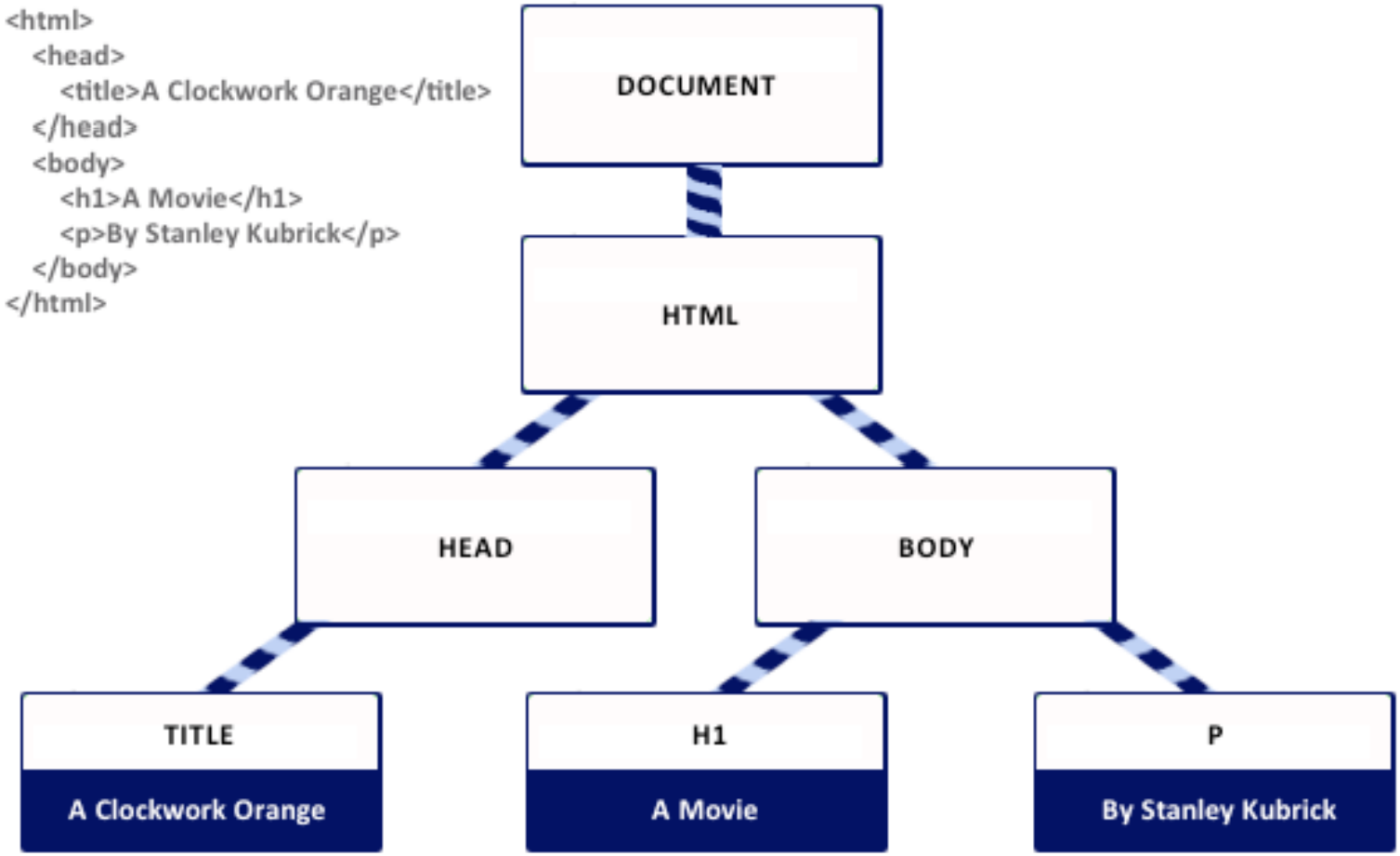


$('#orange'); or
$('div#orange'); or
$('body div#orange');

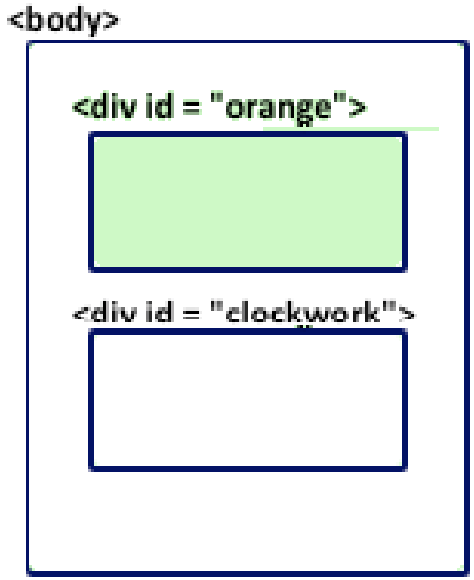


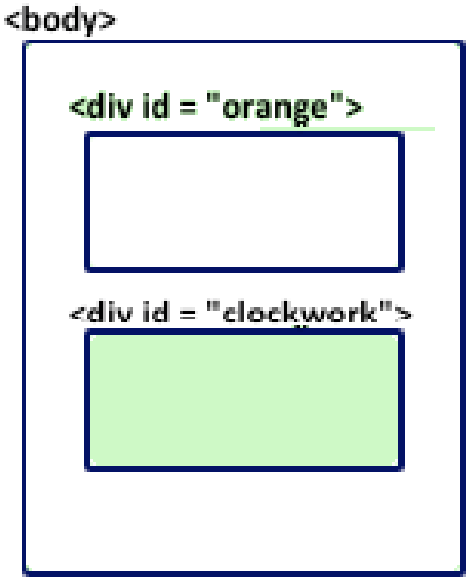
$('#orange').parent();

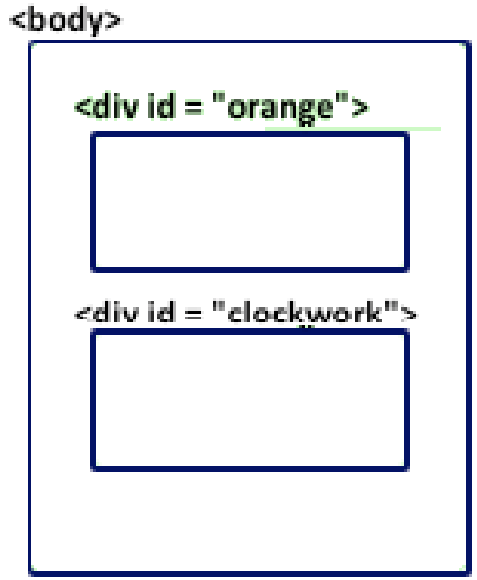
访问匹配结果集：游历


<html>
<head>
<title>A Clockwork Orange</title>
</head>
<body>
<h1>A Movie</h1>
By Stanley Kubrick
</body>
</html>
DOCUMENT
HTML
HEAD
BODY
TITLE
A Clockwork Orange
H1
A Movie
P
By Stanley Kubrick

访问匹配结果集：游历







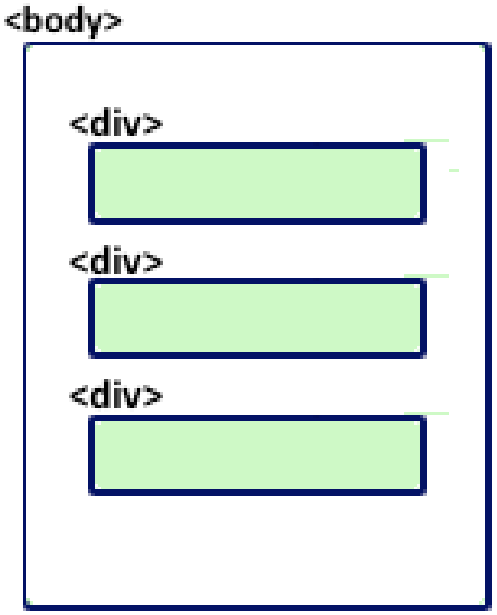
$('#clockwork').prev();

$('#orange').next();

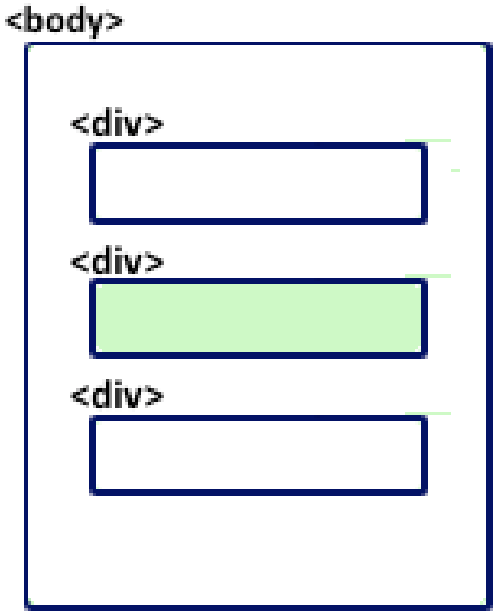
$('#orange').prev();

Nothing is selected because prev() will not select BODY, as it is limited to working within only the parent.

访问匹配结果集：游历

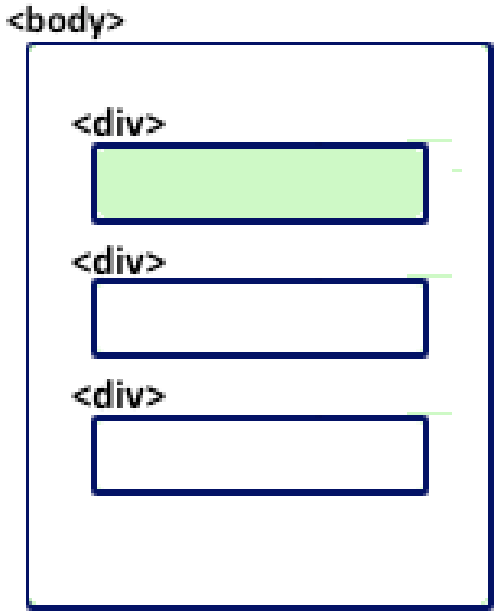


$('div');



$('body :nth-child(2)');

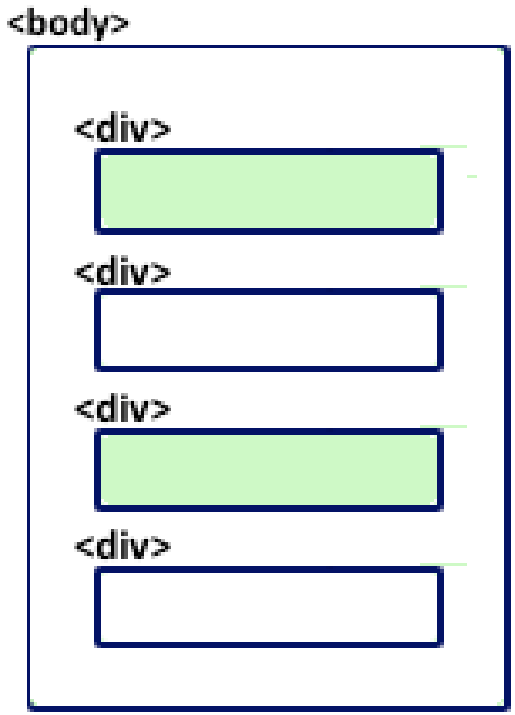
space before :



$('body div:first')

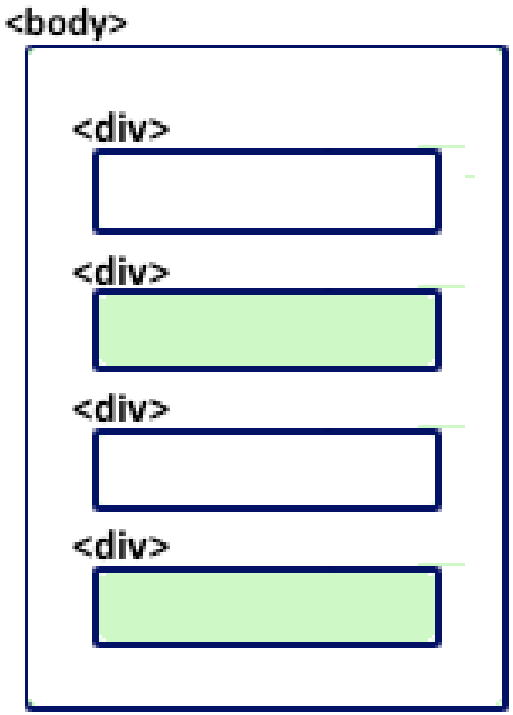
no space before :

访问匹配结果集：游历



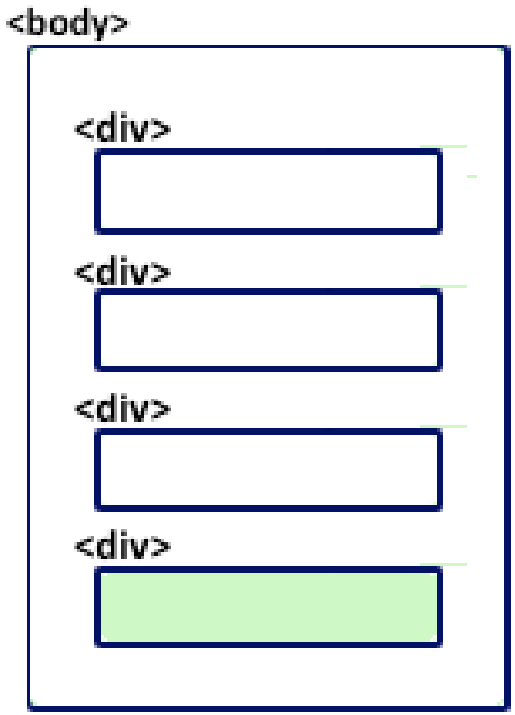
$('body :nth-child(odd)');

space before :



$('body :nth-child(even)');

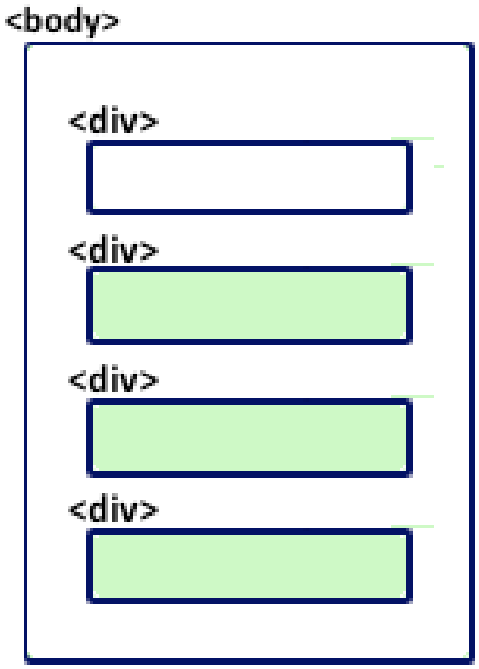
space before :



$('body div:last')

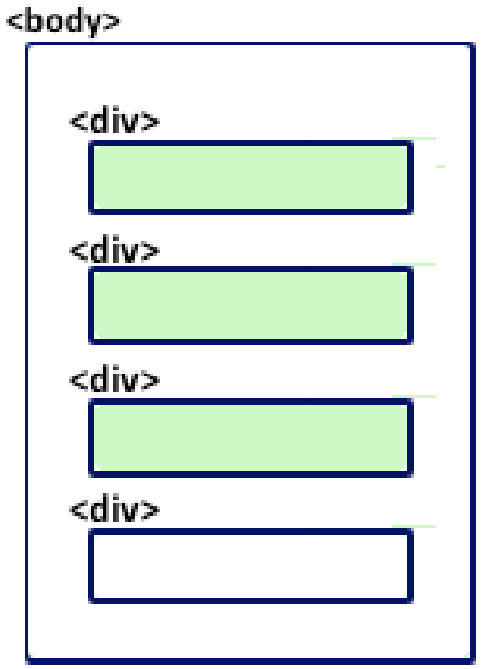
no space before :

访问匹配结果集：游历



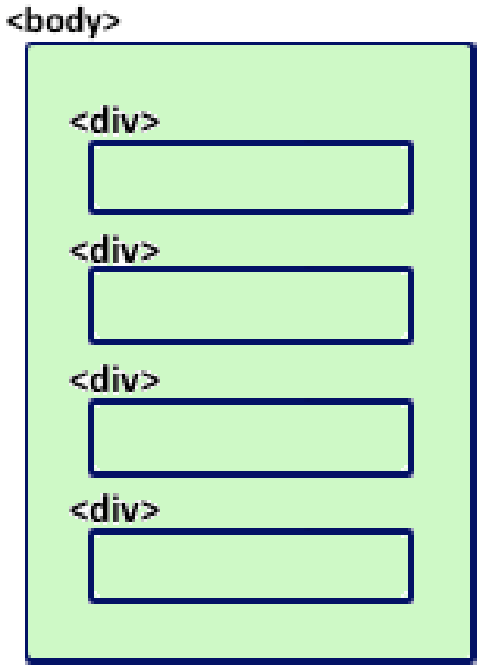
Select all with index > 0

$('body :gt(0)') or
$('div:gt(0)')



Select all with index < 3

$('body :lt(3)') or
$('div:lt(3)')



$('*')

Select Everything

jQuery常用操作方法

- jQuery对象提供了一系列方法来实现DOM的便捷操作。操作网页元素，最常见的需求是取得它们的值，或者对它们进行赋值。
- jQuery对象使用同一个方法名称，来完成取值（getter）和赋值（setter）操作。到底是取值还是赋值，由函数调用时的实际参数决定。
  - `$('h1').html(); //` 参数中没有传值，表示取值操作
  - `$('h1').html('Hello'); //` 参数中有传值，表示赋值操作

jQuery对象常用的方法名	说明
html()	取出或设置html内容
text()	取出或设置text内容
val()	取出某个表单元素的值
attr()	取出或设置某个属性的值
css()	获取或设置css属性
height()	取出或设置某个元素的高度
width()	取出或设置某个元素的宽度

操作方法：text,html,val

- HTML设置：
  - $('span#msg').text('The thing was updated!');//设置文本值
  - $('div#intro').html('Look,HTML');//设置HTML片段
  - $('#myInput').val('hi'); //设置input元素的值

- HTML读取：
  - $('span#msg').text(); //返回消除了HTML标签的文本
  - $('div#intro').html(); //返回包含HTML标签的代码片段
  - $('#myInput').val(); //返回input元素的值

操作方法：attr

- 读取一个属性：
  - `$('a.nav').attr('href');`
- 设置一个属性：
  - `$('a.nav').attr('href','http://flickr.com/');`
- 设置多个属性：
  -
```
$('a.nav').attr({
    'href':'http://flickr.com/',
    'id':'flickr'
    });
```
- 移除一个属性：
  - `$('#intro').removeAttr('id');`

操作方法： css,class

- 读取某一个css属性
  - `$('p').css('font-size');`
- 添加一个class值
  - `$('#intro').addClass('highlighted');`
- 移除一个class值
  - `$('#intro').removeClass('highlighted');`
- 移除或者添加指定的class值
  - `$('#intro').toggleClass('highlighted');`
- 设置某一个css属性
  - `$('p').css('font-size','20px');`
- 设置一批css属性
  - `$('p').css({'font-size':'20px',color:'red'});`

操作注意事项

- 如果结果集包含多个元素，
  - 赋值的时候，将对其中所有的元素赋值；
  - 取值的时候，部分方法直接返回第一个匹配元素的结果：

```
var height = $('div#intro').height();
var src = $('img.photo').attr('src');
var lastP = $('p:last').html()
var hasFoo = $('p').hasClass('foo');
var email = $('input#email').val();
```

例外： **text()**，它取出所有匹配元素的text内容。

jQuery事件表

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit <form>	load 元素或window
dblclick	keydown	change	resize
mouseenter	keyup	focus <input>	scroll
mouseleave		blur	unload

常用事件挂接方法：

1. HTML标签挂接法：
   html元素中，挂接事件的属性名称为：on + 事件名，属性值为js脚本代码片段，如：
   <button id='myInput' onclick="return true;">
2. DOM对象挂接法：
   在DOM中，元素对象以on+事件名为属性名称，同HTML标签挂接法，属性值为函数对象，如：
   Document.getElementById('myInput').onclick = function(){} ;
3. jQuery挂接法：
   在jQuery中，以事件名为jQuery对象的方法名称，以函数对象为参数。挂接例子：
   $('#myInput').click(function(){}) ;

点击事件的监听和触发

```
// 设置事件监听器
$('a:first').click(function(ev) {
  $(this).css({backgroundColor:'orange'});
  return false;
// Or ev.preventDefault();停止事件向上冒泡
});

// 程序主动触发事件监听器
$(‘a:first’).click();
```

挂载网页onload事件

```
$(window).load(function() {
  alert('The DOM is ready!');
});
```

=

```
$(document).ready(function() {
  alert('The DOM is ready!');
});
```

=

```
$(function() {
  alert('The DOM is ready!');
});
```

使用bind方法设置事件监听器

- bind() 方法为被选元素添加一个或多个事件处理程序，并规定事件发生时运行的函数。

语法

```
$(selector).bind(event,data,function)
```

亲自试一试

参数	描述
event	必需。规定添加到元素的一个或多个事件。 由空格分隔多个事件。必须是有效的事件。
data	可选。规定传递到函数的额外数据。
function	必需。规定当事件发生时运行的函数。

- 事件名称：

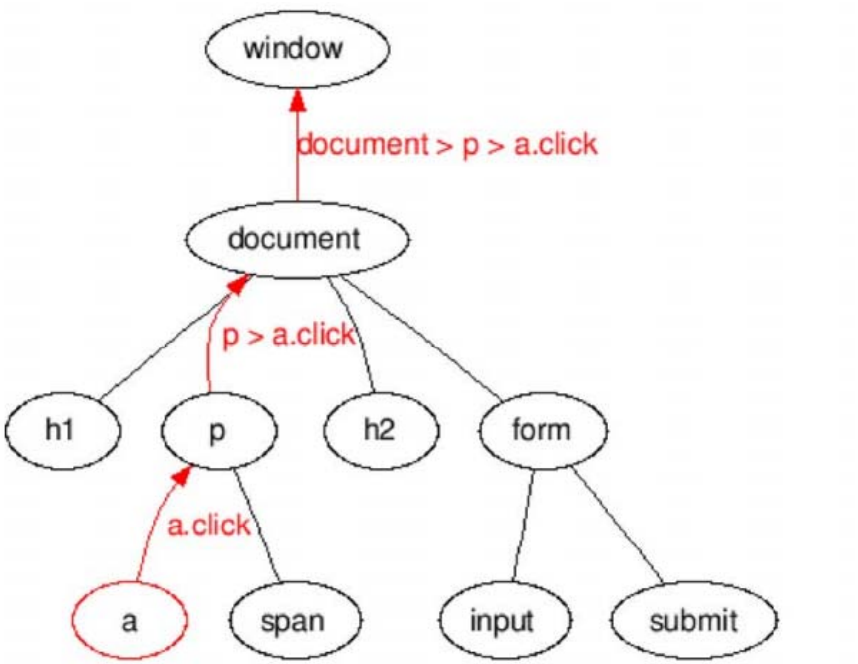
  blur, focus, focusin, focusout, load, resize, scroll, unload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress, keyup, error

- 使用 bind方法:

```
var InputHandler = { startEventType : isTouch ? "touchstart" : "mousedown" }
$(selector).bind(InputHandler.startEventType, function(evnt) {
    evnt.preventDefault(); });
```

使用delegate方法设置事件监听器

- 事件冒泡机制



- 用法：
  - $(document).delegate(“#CashRegister”, “pageinit”, function () {…}) ;

作业

- 用jQuery重新实现计算器功能。

动画

- jQuery内置的动画特效：

```
$('h1').hide('slow');
$('h1').slideDown('fast');
$('h1').fadeOut(2000);
```

- 你可以在其上稍做修改：

```
$('h1').fadeOut(1000).slideDown()
```

- 你可以创建自己的动画特效：

```
$("#block").animate({
  width:"+=60px",
  opacity:0.4,
  fontSize:"3em",
  borderWidth:"10px"
},1500);
```

链式编程

- jQuery的大多数方法都返回另一个jQuery对象。这意味着可以使用链式编程方式：

```
$('div.section').hide().addClass('gone');
```

等同于：

```
var obj = $(‘div.section’) ; // 找到目标元素
    obj = obj.hide() ;   // 将其隐藏
    obj = obj.addClass(‘gone’); // 添加class ‘gone’
```

- 可以调用.end()方法来使得结果集后退一步,返回到先前的集合。

```
$('#intro').css('color','#cccccc').
  find('a').addClass('highlighted').end().
  find('em').css('color','red').end()
```

链式编程的注意事项

- 用数组方法访问集合中的某一元素，返回的是DOM对象，非jQuery对象，不能进行链式编程：
  - `$('div.section')[1].innerHTML //访问DOM对象的属性innerHTML`
  - `$('div.section').get(1) //等同于$('div.section')[1]`
  - `$('div.section')[1].html() // 错误用法!`

  要返回jQuery对象，继续使用链式编程，可使用eq方法：
  - `$('div.section').eq(1).html() // 正确!`

作业

- 为计算器按钮添加iOS软键盘动态效果。

什么是插件

- 插件是jQuery原型对象上扩展出来的一种新的方法（method）。插件一旦生效，所有的jQuery对象都将拥有该方法。
- 插件是对集合中所有元素的一次处理操作，甚至可以将.fadeOut(),.addClass()等都看作一种插件。

如何创建一个插件

- 可以通过插件来扩充jQuery对象的新方法
  - Form插件：更好的Form元素操作能力
  - UI：拖动、移动、视图
  - $('img[@src$=.png]').ifixpng()
  - …

此this, 非 $(this)

```
jQuery.fn.hideLinks = function() {
  this.find('a[href]').hide();
   return this;
}
$('p').hideLinks();
```

return this 确保链式编程原则

用模块化方法创建插件

- 避免$符号冲突，使用私有变量。

```
$.fn.greenify = function() {
    this.css( "color", "green" );
    return this;
}

$( "a" ).greenify().addClass( "greenified" );
```

```
(function ( $ ) {

    var shade = "#556b2f";

    $.fn.greenify = function() {
        this.css( "color", shade );
        return this;
    };

}( jQuery ));
```

节省插件命名空间

```
(function( $ ) {

    $.fn.openPopup = function() {
        // Open popup code.
    };

    $.fn.closePopup = function() {
        // Close popup code.
    };

}( jQuery ));
```

```
(function( $ ) {

    $.fn.popup = function( action ) {

        if ( action === "open") {
            // Open popup code.
        }

        if ( action === "close" ) {
            // Close popup code.
        }

    };

}( jQuery ));
```

使用更复杂的参数选项

```
(function ( $ ) {

    $.fn.greenify = function( options ) {

        // This is the easiest way to have default options.
        var settings = $.extend({
            // These are the defaults.
            color: "#556b2f",
            backgroundColor: "white"
        }, options );

        // Greenify the collection based on the settings variable.
        return this.css({
            color: settings.color,
            backgroundColor: settings.backgroundColor
        });

    };

}( jQuery ));
```

```
$( "div" ).greenify({
    color: "orange"
});
```

使用插件

- 使用插件：　下载→解压→引入到页面中的jQuery库之后。

作业

- 学会使用5个第三方插件。

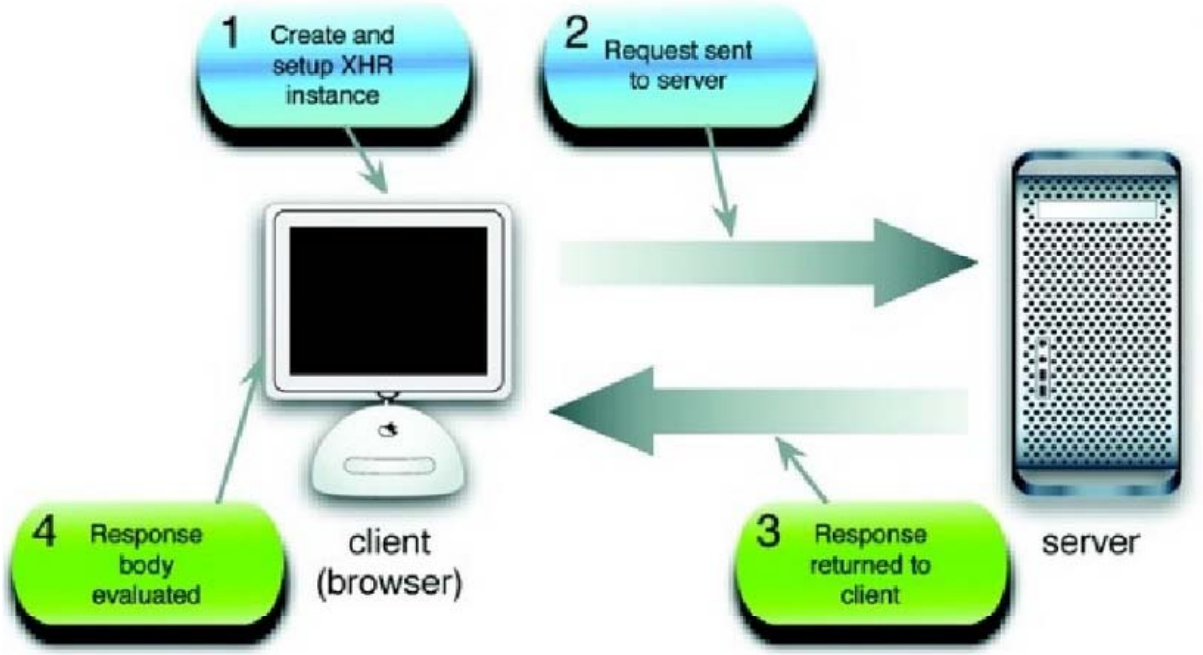
提示：查找jQuery插件，从baidu、谷歌开始。

其他插件

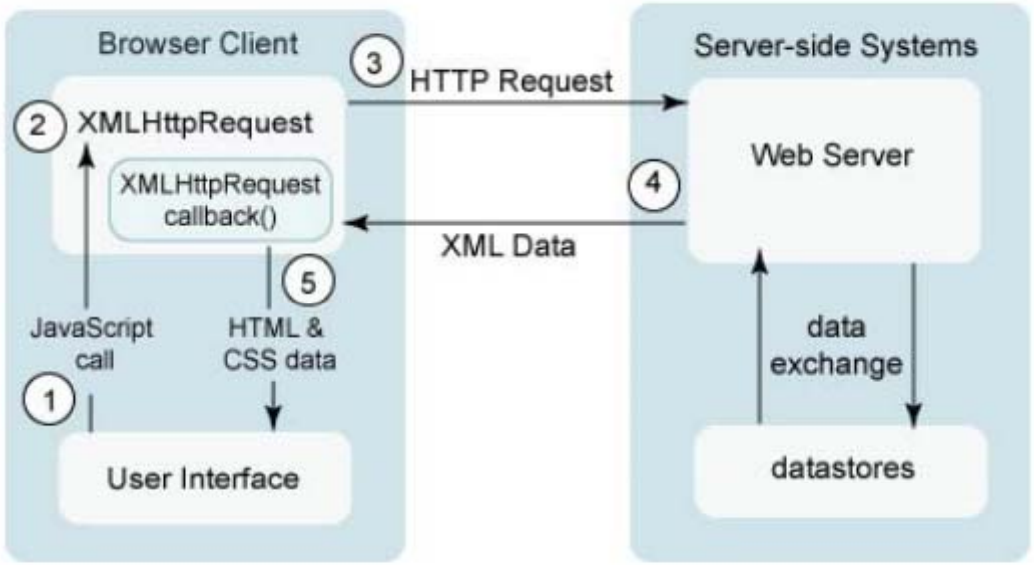
- jQuery UI是构建在jQuery内核之上的一组精心组织的用户界面交互、特效、小窗口和主题风格。由jQuery小组负责维护，质量可靠。

什么是AJAX

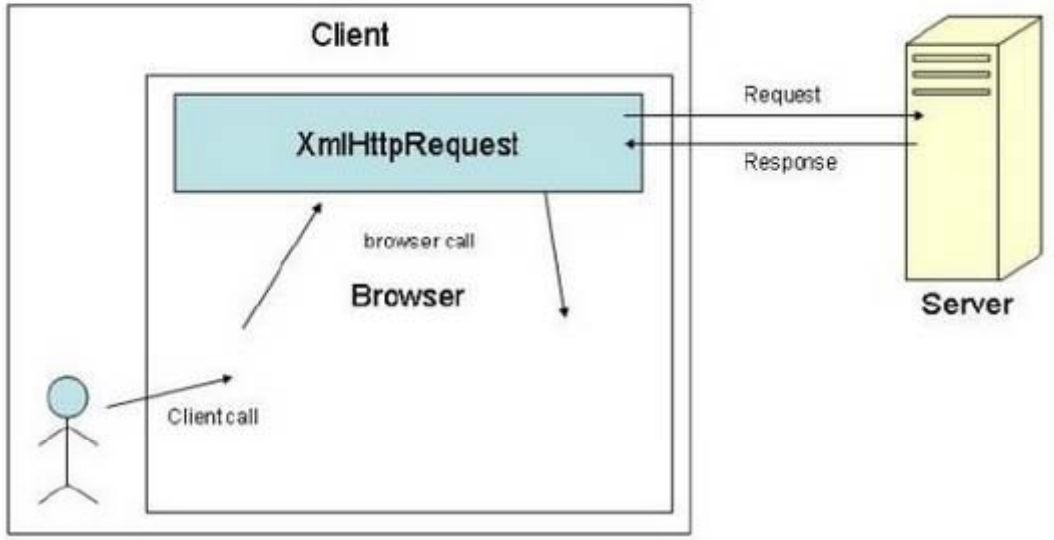
- AJAX： Asynchronous JavaScript and XML（异步JavaScript和XML)



Ajax原理


Browser Client
Server-side Systems
3
HTTP Request
2
XMLHttpRequest
XMLHttpRequest
callback()
Web Server
4
XML Data
5
JavaScript
call
HTML &
CSS data
data
exchange
1
User Interface
datastores

XHR对象


Client
XmlHttpRequest
Request
Response
browser call
Browser
Server
Client call

XHR例子

- 将网络上的菜单数据显示于元素‘response’

```
var request = new XMLHttpRequest();

request.onreadystatechange = function(){
  if(request.readyState == 4)
    document.getElementById(“response”).innerHTML =
                                                request.responseText
}

request.open('POST', '/recipes.xml', true);

request.setRequestHeader('Content-Type',
                         'application/x-www-form-urlencoded');

request.send('recipe[title]=Ice+Water&'+
             'recipe[instructions]=Pour+ice+and+water+in+glass');
```

jQuery例子

- 功能同上

```
jQuery.post( url, [data], [callback], [type] )
```

```
jQuery.post(
    “/recipes.xml”,
  {
    “recipe[title]” : “Ice Water”,
    “recipe[instructions]” : “Pour ice and water in glass”
  },
  function(html){ $(“#response”).text(html)  },
  “html”
)
```

用jQuery实现AJAX

```
var results = $(“#results”).html(“<li>Loading...</li>”);

$.get(“test.html”, function(html){
  results.html( html );
  //“通过闭包来实现回调函数”
});
```

jQuery对Ajax的支持

- 对Ajax有很好的支持：

  $('div#intro').load('/some/file.html'); //将网络上的HTML片段代码嵌入到指定的div元素中。
- 有更多的高级用法：

  $.get(url,params,callback)

  $.post(url,params,callback)

  $.getJSON(url,params,callback)

  $.getScript(url,callback)

例1：同域情况下使用

- .get()方法不能跨域请求：

```
$.get('ajax/test.html', function(data) {
    $( '.result' ).html(data); //加载HTML片段
    alert( '加载完成.');
});
```

例1：同域情况下使用

- 与上例等效！

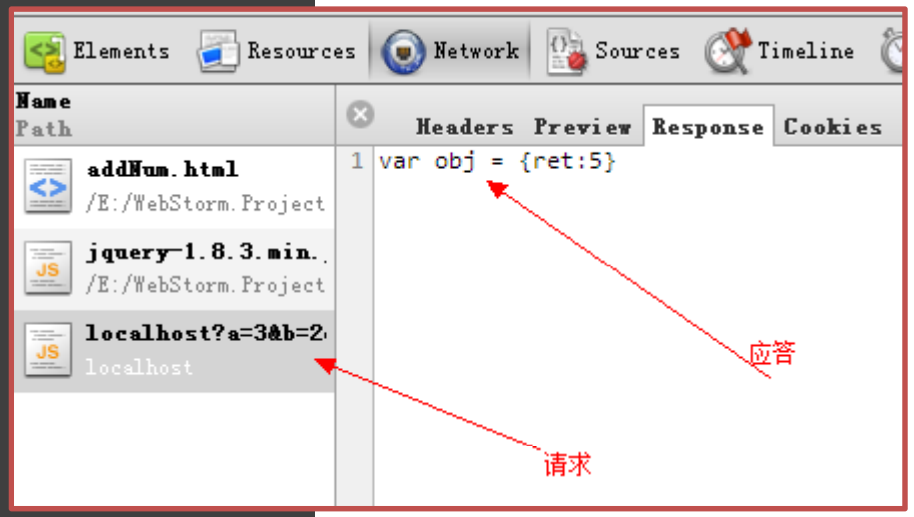
```
$.ajax({
   url: 'ajax/test.html',
   success:  function(data) {
         $(  '.result'  ).html(data);
         alert(  '加载完成.');
    },
});
```

例2： Ajax跨域访问js脚本

```
<body>
<form action="http://localhost:9000/" >
    first num: <input type="text" name="a" />
    second num: <input type="text" name="b" />
    result num: <input type="text" name="ret" id="ret"/>
    <input type="submit" value="加加看!" />

</form>
<script type="text/javascript">
    var frm =  document.forms[0] ;
    function doSubmit(){
        var oResult = $("#ret");
        var successFn = function(){
            oResult.val(obj.ret) ;
        }
        var url = "http://localhost:9000/?" ;
        var params = {
            a : frm.a.value,
            b : frm.b.value
        }
        url += $.param(params) ;
        oResult.val("loading").css("color","red");
        $.getScript(url,successFn ) ;
        return false ;
    }
    frm.onsubmit = doSubmit ;
</script>  </body> </html>
```

$.getScript();



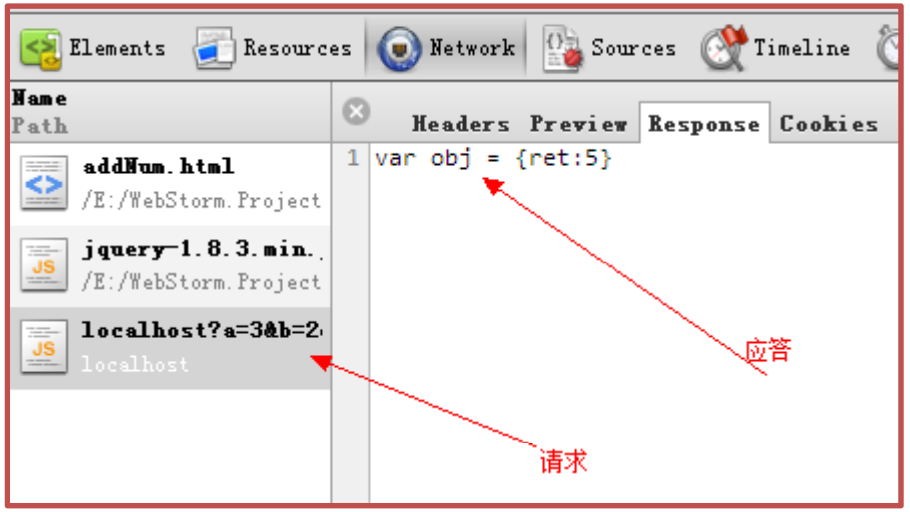
例2：Ajax跨域访问js脚本

- 与上例等效！

$.ajax();

```
<script type="text/javascript">
    var frm =  document.forms[0] ;
    function doSubmit(){
        var oResult = $("#ret");
        var successFn = function(){
            oResult.val(obj.ret) ;
        }
        var url = "http://localhost:9000/" ;
        var params = {
            a : frm.a.value,
            b : frm.b.value
        }
        oResult.val("loading").css("color","red");

        $.ajax({
            url : url,
            data : params,
            success : successFn,
            dataType : "script"
        }) ;
        return false ;
    }
    frm.onsubmit = doSubmit ;
</script>  </body> </html>
```



Ajax各参数含义

- 见附件

补充阅读

- 英文：
  - http://jquery.com/
  - http://docs.jquery.com/
  - http://visualjquery.com/- API reference
  - http://simonwillison.net/tags/jquery/
  - http://www.w3schools.com/jquery/jquery_dom_set.asp

- 中文：
  - http://www.jquery.org.cn