$('li.box').css('background', '#f00');

如我们前面写的代码，总感觉怪怪的，其实我们要注意的一点是：jq当中的所有操作都是方法函数化。

比如：

// 原生

window.onload = function (){};

// jq

$(function (){});

哪么我们看jq中的$(function (){})，像不像有一个函数名叫$，而我们调用，函数就是我们传的参数。

其实jq中就是相当于有一个function $(){}这样一个函数。我们$()时，就是函数调用。

页面加载完成的另一种写法：

$(document).ready(function() {})

另外：

innerHTML = 123;

html(123);

onclick = function (){}

click(function (){})

发现规律：原生是把后续的操作与元素用等号连接。

而jq是把后续的操作放在函数调用的括号中。做为参数传进去。

**再如：点击div，弹出它的内容**

<div id="div1">div</div>

// 原生

window.onload = function (){

var oDiv = document.getElementById('div1');

oDiv.onclick = function (){

console.log(this.innerHTML);

}

}

// jq方式，对应着上面一步一步写

$(function (){

var oDiv = $('#div1');

oDiv.click(function (){

console.log($(this).html());

})

});

我们发现几个细节：

1、jq中几乎没有等号。在原生中，等号是用来赋值，而在jq 中，我们是用传参的形式完成的。

2、调用的时候，我们一定要带上括号。比如html。单写html是函数名，只有带上括号，才是函数调用。