5. jQuery最佳实践

使用CDN和本地fallback

```
<script type="text/javascript" src="//
ajax.googleapis.com/ajax/libs/jquery/
2.1.1/jquery.min.js"></script>

//国内CDN http://www.cdnjs.cn/

<script>window.jQuery ||
document.write('<script src="js/
jquery-2.1.1.min.js" type="text/
javascript"><\/script>')</script>
```

版本

- 使用1.x版还是2.x版?
- 使用CDN时，不要在url中使用latest版本
- 指定具体的版本号
- $.noConflict()

jQuery变量

- jQuery变量使用$开头
- jQuery对象都保存在一个变量中，以便重用

- 尽可能使用ID选择器，因为它会直接使用JavaScript原生的document.getElementById().

- 如果使用.class，不要加上标签

```
var $products = $("div.products"); // SLOW

var $products = $(".products"); // FAST
```

```
// BAD, a nested query for Sizzle selector engine

var $productIds = $("#products div.id");



// GOOD, #products is already selected by
document.getElementById() so only div.id needs to
go through Sizzle selector engine

var $productIds = $("#products").find("div.id");
```

15 lines (12 sloc) | 0.314 kB

```
define([
        "./core",
        "sizzle"
], function( jQuery, Sizzle ) {

jQuery.find = Sizzle;
jQuery.expr = Sizzle.selectors;
jQuery.expr[":"] = jQuery.expr.pseudos;
jQuery.uniqueSort = jQuery.unique = Sizzle.uniqueSort;
jQuery.text = Sizzle.getText;
jQuery.isXMLDoc = Sizzle.isXML;
jQuery.contains = Sizzle.contains;

});
```

```
(function( window, undefined ) {
    // 构造jQuery对象
    var jQuery = (function()  {
        var jQuery = function( selector, context ) {
                return new jQuery.fn.init( selector, context, rootjQuery );
            }
        return jQuery;
    })();
    // 工具方法 Utilities
    // 回调函数列表 Callbacks Object
    // 异步队列 Deferred Object
    // 浏览器功能测试 Support
    // 数据缓存 Data
    // 队列 Queue
    // 属性操作 Attributes
    // 事件系统 Events
    // 选择器 Sizzle
    // DOM 遍历 Traversing
    // DOM 操作 Manipulation
    // 样式操作 CSS（计算样式、内联样式）
    // 异步请求 Ajax
    // 动画 Effects
    // 坐标 Offset、尺寸 Dimensions
    window.jQuery = window.$ = jQuery;
})(window);
```

自执行的匿名函数

```
(function( window, undefined ) {
    // jquery code
})(window);

// window从全局变量变为局部变量
// undefined传入，确保undefined是真的未定义
```

右边的选择器尽量明确

```
// Unoptimized

$("div.data .gonzalez");


// Optimized

$(".data td.gonzalez");
```

指定context

```
// SLOWER because it has to traverse the whole DOM for .class

$('.class');



// FASTER because now it only looks under class-container.

$('.class', '#class-container');
```

ID是唯一的!

- `$('#outer #inner'); // BAD`
- `$('div#inner'); // BAD`
- `$('.outer-container #inner'); // BAD`
- `$('#inner'); // GOOD, only calls document.getElementById()`

使用.detach()操作DOM

```
var $myList = $("#list-container > ul").detach();

//...a lot of complicated things on $myList

$myList.appendTo("#list-container");
```

频繁操作DOM，很慢！

```
// BAD

var $myList = $("#list");

for(var i = 0; i < 10000; i++){

    $myList.append("<li>"+i+"</li>");

}
```

只操作一次DOM

```
// GOOD

var $myList = $("#list");

var list = "";

for(var i = 0; i < 10000; i++){

    list += "<li>"+i+"</li>";

}

$myList.html(list);
```

array.join(‘’)比字符串更快

```
// EVEN FASTER

var array = [];

for(var i = 0; i < 10000; i++){

    array[i] = "<li>"+i+"</li>";

}

$myList.html(array.join(''));
```

事件绑定

不要在事件绑定中使用匿名函数，匿名函数不好维护，调试，测试，重用

```
$("#myLink").on("click", function(){...}); // BAD

// GOOD

function myLinkClickHandler(){...}

$("#myLink").on("click", myLinkClickHandler);

//http://learn.jquery.com/code-organization/beware-anonymous-functions/
```

```
$(function(){ ... }); // BAD: You can never reuse or
write a test for this function.



// GOOD

$(initPage); // or $(document).ready(initPage);

function initPage(){

    // Page load event where you can initialize values
and call other initializers.

}
```

不要在HTML中绑定事件

```
<a id="myLink" href="#"
onclick="myEventHandler();">my link</a> <!-- BAD -->

$("#myLink").on("click", myEventHandler); // GOOD
```

```
// BAD, you are attaching an event to
all the links under the list.

$("#list a").on("click",
myClickHandler);



// GOOD, only one event handler is
attached to the parent.

$("#list").on("click", "a",
myClickHandler);
```

```
.live() // 已删除

.delegate() //1.7后不建议

.on( events [, selector ] [, data ], handler )
```

事件冒泡

- <a>
-
- <ul #list>
- <div #container>
- <body>
- <html>
- document root

在“快选择器”上绑定事件

```
// BAD

$( "body" ).on( "click", "#commentForm .addNew", addComment )



// GOOD

$( "#commentForm" ).on( "click", ".addNew", addComment )



// http://api.jquery.com/on/
```

Ajax

```
// Less readable...
$.ajax({
    url: "something.php?param1=test1&param2=test2",
    ....
});

// More readable...
$.ajax({
    url: "something.php",
    data: { param1: test1, param2: test2 }
});
```

使用promise接口

```
//方法1

$.ajax({ ... }).then(successHandler,
failureHandler);



//方法2

var jqxhr = $.ajax({ ... });

jqxhr.done(successHandler);

jqxhr.fail(failureHandler);
```

长链式操作：巧用换行

```
$("#myLink")
    .addClass("bold")
    .on("click", myClickHandler)
    .on("mouseover", myMouseOverHandler)
    .show();
```

糟糕的代码

```
$("#elem").attr("title", $("#elem").text());

$("#elem").css("color", "red");

$("#elem").fadeOut();
```

改进的代码

- var $elem = $(‘#elem’);
- $elem.attr("title", $elem.text())
  .addClass("important")
  .fadeOut();

糟糕的代码

```
var localArr = ["Greg", "Peter", "Kyle", "Danny", "Mark"],
    list = $("ul.people");
  $.each(localArr, function(index, value) {
    list.append("<li id=" + index + ">" + value + "</li>");
  });
```

改进的代码