- jQuery介绍
- jQuery的优势
- jQuery下载和运行
- 基础核心
  - 1.代码风格
  - 2.加载模式 入口函数
  - 3.对象互换及处理多个库之间的冲突
- jQuery选择器
  - 基础选择器
  - 层次选择器
  - 过滤选择器
  - 内容过滤选择器
  - 属性过滤选择器
  - 子元素过滤选择器
  - 表单过滤选择器
  - 表单选择器
  - jQuery选择器完善的处理机制
- jQuery中的DOM操作
  - 1.HTML DOM 操作
    - 插入节点
    - 删除节点
      - remove()
      - detach()
      - empty()
    - 复制节点
    - 其它方法
  - 2. CSSDOM操作

jQuery介绍

官方介绍

```
jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

翻译:
jQuery是一个快速,小,功能丰富的JavaScript库。它使诸如HTML文档遍历和操作,事件处理、动画和Ajax更简单和易于使用的API,跨越多种浏览器。结合的通用性和可扩展性,jQuery已经改变了数百万人的方式编写JavaScript。
```

参考文档

write less,do more

jQuery在线中文文档

jQuery官方文档

jQuery的优势

jQuery 作为 JavaScript 封装的库，他的目的就是为了简化开发者使用 JavaScript。主要 功能有以下几点：

- 强大的选择器
- 像 CSS 那样访问和操作 DOM
- 修改 CSS 控制页面外观
- 简化 JavaScript 代码操作
- 事件处理更加容易
- 各种动画效果使用方便
- 让 Ajax 技术更加完美
- 基于 jQuery 大量插件
- 自行扩展功能插件
- 出色的浏览器兼容
- 链式操作方式
- 隐式迭代
- 完善的文档
- 开源
- .....

jQuery 最大的优势，就是特别的方便。比如模仿 CSS 获取 DOM，比原生的 JavaScript 要方便太多。并且在多个 CSS 设置上的集中处理非常舒服，而最常用的 CSS 功能又封装到 单独的方法，感觉非常有心。

最重要的是 jQuery 的代码兼容性非常好，你不需要总是头疼着考虑不同浏览器的兼容问题。

jQuery下载和运行

下载

CDN下载

第一个jQuery程序

```
<!--引入jquery文件，在body内下入如下代码-->
<button>按钮</button>
<script src='./jquery.js'></script>
<script type="text/javascript">
    $(function(){
        $("button").click(function() {
            alert("hello jQuery!");
        });
    });
</script>
```

基础核心

1.代码风格

在jQuery程序中，不管是页面元素的选择，内置的功能函数，都是美元符号`$`来开始的。而这个`$`就是jQuery当中最红要且独有的对象:jQuery对象，所以我们在页面元素选择或执行功能函数的时候可以这样写：

```
$(function () {}); //执行一个匿名函数
$('#box'); //进行执行的ID元素选择
$('#box').css('color', 'red'); //执行功能函数
```

由于$本身就是jQuery对象的缩写形式，那么也就是说上面的三段代码也可以写成如下形式：

```
jQuery('#box').css('color','red');
console.log(jQuery===$);//true
```

并且，每次执行函数后，都会返回一个jQuery对象。如下：

```
$('#box').css('color', 'red').css('font-size', '50px'); //链式编程
```

2.加载模式 入口函数

> 我们在之前的代码一直在使用`$(function () {});`这段代码进行首尾包裹，那么为什么必须 要包裹这段代码呢?

- 我们jQuery库文件是在body元素之前加载的，我们必须等待所有的DOM元素加载后，延迟支持DOM操作，否则就无法获取到。使用`document.ready()`,只需要等待DOM加载完成就执行。
- 我们的原生Javascript也有一个延迟加载的方法`onload`，当网页内容全部加载完成后执行（例如图片等大文件未加载完成之前，JS功能处于假死状态）。

onload和ready区别

区别	`window.onload`	`$(document).ready()`
执行时机	必须等待网页全部加载完毕（包括图片等），然后再执行包裹代码	只需要等待网页中的DOM结构加载完毕，就能执行包裹的代码
执行次数	只能执行一次，如果第二次，那么第一次的执行会被覆盖	可以执行多次，第N次都不会覆盖上一次
简写方案	无	`$(function () {});`

3.对象互换及处理多个库之间的冲突

对象互换

首先我们先看一段代码

```
alert($);//返回jQuery对象方法内部函数
alert($());//[object object]，返回jQuery对象
alert($("#box"));//[object object]，返回jQuery对象
alert(document.getElementById("box"));//[object HTMLDivElement]，返回原生DOM对象
```

如何进行转换呢?

jQuery想要获取原生的DOM对象，可以这样处理

```
$('#box').get(0);
//或者
$('#box')[0]
```

通过这个索引0可以看出jQuery 是可以进行批量处理 DOM 的，这样可以在很多需要循环遍历的处理上更加得心应手。

当然要重新转化成jQuery对象的话，只需要使用$()包裹原生对象就可以了。

```
var oBox = document.getElementById('box');
$(oBox);
```

多个库之间的冲突

当一个项目中引入多个第三方库的时候，由于没有命名空间的约束（命名空间就好比同 一个目录下的文件夹一样，名字相同就会产生冲突），库与库之间发生冲突在所难免。

jQuery 只不过是 DOM 操作为主的库，方便我们日常 Web 开发。但有时，我们的项目有更多特殊的功能需要引入其他的库，比如用户界面 UI 方面的库，游戏引擎方面的库等等一系列。

所以jQuery提供了一个方法：`jQuery.noConflict();`：将$符所有权剔除。

```
<script src="my.js"></script>
<script src="https://cdn.bootcss.com/jquery/3.4.1/jquery.js"></script>
<script>
    jQuery.noConflict(); //将变量$控制权交给其他的js库
    console.log($); //3
    (function($) { // 定义匿名函数并设置形参 $
        //里面照样使用$
        console.log($);
    })(jQuery); // 执行匿名函数并设置形参 jQuery
</script>
```

如果jQuery库在其它库之前导入 照常工作，不需要以上设置

jQuery选择器

基础选择器

模仿的是CSS选择器，只不过在使用jQuery选择器时，我们首先必须使用“$()”函数来包装我们的 CSS 规则。 而CSS规则作为参数传递到jQuery对象内部后，再返回包含页面中对应元素的 jQuery 对象。随后可以进行节点操作，例如：`$('#box').css('color', 'red');`。

那么除了 ID 选择器之外，还有两种基本的选择器，分别为：元素标签名和类(class)：

选择器	描述	返回
#id	根据给定的 ID 匹配一个元素	单个元素
.class	根据给定的类名匹配一个元素	集合元素
element	根据给定的元素名匹配一个元素（相当于 tagName ）	集合元素
*	匹配所有元素	集合元素
select1,select2,select3	将每一个选择器匹配到的元素合并后一起返回	集合元素

我们可以采用jQuery核心自带的一个属性`length`来查看返回的元素个数。

层次选择器

选择器	描述	返回
ancestor descendant(空格)	选取 ancestor 元素里所有的 descendant（后代）元素	集合元素
parent > child	选取子元素	集合元素
prev + next	选取紧接在 prev 元素后面的 next 元素	集合元素
prev ~ siblings	选取 prev 元素之后的所有 siblings 元素	集合元素

过滤选择器

选择器	描述	返回
:first	选取第一个元素	单个元素
:last	选取最后一个元素	单个元素
:not(selector)	去除所有与给定选择器匹配的元素	集合元素
:even	索引为偶数（索引从 0 开始）	集合元素
:odd	索引为奇数（索引从 0 开始）	集合元素
:eq(index)	索引等于 index 的元素（index 从 0 开始）	单个元素
:gt(index)	索引大于 index	集合元素
:lt(index)	索引小于 index	集合元素

内容过滤选择器

选择器	描述	返回
:contains(text)	查找文本中含有 “text” 的元素	集合元素
:empty	匹配所有不包含子元素或者文本的空元素	集合元素
:has(selector)	含有选择器所匹配的元素	集合元素
:hidden	选取所有不可见的元素	集合元素
:visible	选取所有可见的元素	集合元素

属性过滤选择器

选择器	描述	返回	示例
[attribute]	拥有此属性的元素	集合元素	`$("div[id]")` 选择所有拥有 id 属性的 div
[attribute=value]	属性的值为 value 的元素	集合元素	`$("div[tittle = test]")` 属性 `title` 为 `test` 的 div
[attribute!=value]	属性的值不为 value 的元素	集合元素	`$("div[tittle != test]")` 属性 `title` 不为 `test` 的 div
[attribute^=value]	属性的值以 value 开始的元素	集合元素	`$("div[tittle^ = test]")` 属性 `title` 以 `test` 开始的 div

子元素过滤选择器

选择器	描述	返回
:nth-child(index)	选取每个父元素下的第 index 个子元素或者奇偶元素（index 从 1 开始）	集合元素
:first-child	选取每个父元素第一个子元素	集合元素
:last-child	选取每个父元素最后一个子元素	集合元素
:only-child	如果某个元素是它父元素中唯一的子元素，则会被匹配	集合元素

表单过滤选择器

选择器	描述	返回
:enabled	选取所有可用元素	集合元素
:disable	选取所有不可用元素	集合元素
:checked	选取所有被选中元素（复选框、单选框）	集合元素
:selected	选取所有被选中元素（下拉列表）	集合元素

表单选择器

选择器	描述	返回
:input	选取所有的 `<input>` `<textarea>` `<select>` `<button>`	集合元素
:text	选择所有单行文本框	集合元素
		集

:password	选择所有的密码框	合元素
:radio	选择所有的单选框	集合元素
:checkout	选择所有的多选框	集合元素
:submit	选择所有的提交按钮	集合元素
:image	选择所有的图像按钮	集合元素
:reset	选择所有的重置按钮	集合元素
		集

input:button	选择所有的按钮	合元素
input:file	选择所有的上传域	集合元素
:hidden	选择所有的不可见元素	集合元素

jQuery选择器完善的处理机制

- 如果元素不存在时，JS 不会保存阻塞其他代码的运行。
- `$(#ID)` 或者其他选择器获取的永远是对象，即使网页上没有此元素。使用 jQuery 检查某个元素是否存在要不能使用

```
if($('#box')){
    //dosomething
}
```

而是根据元素是否有长度判断：

```
if($('#box').length>0 ){
    //dosomething
}
```

或者转化为 DOM 元素来判断

```
if($('#box')[0]){
    //dosomething
}
```

jQuery中的DOM操作

1.HTML DOM 操作

插入节点

方法	描述	示例
append()	向每个匹配的元素内部追加内容	$(A).append(B) 将 B 追加到 A 中
appendTo()	将所有匹配的元素追加到指定元素中	$(B).appendTo(A) 将 B 追加到 A 中
prepend() prependTo()	向每个匹配的元素内部前置内容	$(A).prepend(B) 将 B 插入到 A 前面
after()	在每个匹配的元素之后插入内容	$(A).after(B) 将 B 插入到 A 后面
insertAfter()	将所有匹配的元素插入到指定元素的后面	$(B).insert After(A) 将 B 插入到 A 后
before()	在每个匹配的元素之前插入内容	$(A).before(B) 将 B 插入在 A 的前面
insertBefore()	将所有匹配的元素插入到指定元素的前面	$(B).insertBefore(A) 将 B 插入在 A 的前面

删除节点

remove()

从 DOM 中删除所有匹配的元素，传入的参数用于根据 jQuery 表达式来删选元素

```
$("ul li:eq(1)").remove();  // 获取第二个 <li> 元素节点后，将它从网页中删除
$li.appendTo("ul");         // 把刚才删除的元素添加到 <ul> 元素中复制代码
```

这个方法的返回值是一个指向已被删除的节点的引用，因此可以将其保存在一个变量中，以后还可以使用。

detach()

detach() 和 delete() 一样，也是从 DOM 中去掉所有匹配的元素，但是两者的区别是，这个方法不会把匹配的元素从 jQuery 对象中删除，去掉的元素的所有绑定的事件、附加的数据等都会保留下来。

empty()

清空元素中所有的后代节点。**注意是清空元素内的所有节点，并不清除选中的元素**

复制节点

复制节点可以使用 `clone()` 方法

```
$("ul li").click(function(){
    $(this).clone().appendTo("ul");
})
```

但是这样复制的节点，被复制的新元素并不具有任何行为，如果需要新元素也具有相同的行为，那么就需要在 `clone()` 方法中传入参数 `true`

```
$("ul li").click(function(){
    $(this).clone(true).appendTo("ul");
})
```

其它方法

方法名	描述
replaceWith('**</b')**	将所有匹配的元素替换成指定的HTML或DOM元素。
replaceAll(selector)	用匹配的元素替换掉所有selector匹配到的元素。
wrap()	把所有匹配的元素用其他元素的结构化标记包裹起来。
unwrap()	这个方法将移出元素的父元素
wrapInner()	将每一个匹配的元素的子内容(包括文本节点)用一个HTML结

	构包裹起来
attr()	获取和设置元素属性，传递一个参数为获取元素属性，传递两个参数为设置元素属性
removeAttr()	删除文档中某个元素的特定属性
addClass() removeClass()	追加样式
toggleClass()	切换样式 如果类名存在则删除，如果类名不存在则添加
hasClass()	是否含有某个类，返回布尔值
html()	读取或者设置某个元素中的 HTML 内容 **没有参数为获取 HTML 中的内容，传递一个参数为设置 HTML 的内容**
text()	读取或者设置某个元素中的文本内容 **没有参数为获取文本内容，传递一个参数为设置文本内容**
val()	读取或设置元素的值 **在用于表单元素时，可以设置相应的元素被选中**
	获得匹配元素的子元素的集

children()	合 （**子元素非后代元素**）
next()	获得匹配元素后面紧邻的同辈元素
siblings()	获得匹配元素前后面紧邻的同辈元元素
parent()	获得集合中每个元素的父级元素
parents()	获得集合中每个元素的祖先元素
prev()	获得匹配元素前面紧邻的同辈元素
eq()	

2. CSSDOM操作

方法	描述
css()	读取和设置 style 对象的各种属性（如果值是数字，将会自动转化为像素值，样式名不带 “”样式使用驼峰写法）
offset()	获取元素在当前视窗的相对偏移，返回的对象包含两个属性`top`、`left`
position()	获取元素相对于最近一个position样式属性设置为relative或者absolute的父节点相对偏移
scrollTop()	获取元素滚动条距离顶端的距离
scrollLeft()	获取元素滚动条距离左侧的距离
width()/height()	获取和设置宽度和高度
innerWidth()/innerHeight()	获取匹配元素内部区域宽度(包括补白,不包括边框)
outerWidth()/outerHeight()	获取匹配元素外部宽度(默认包括补白和边框)

JQ中的事件

事件绑定

```
bind(type [, data ], fn )
```

- 第一个参数是事件类型，类型包括：`blur` `focus` `load` `resize` `scroll` `unload` `click` `dblclick` `mousedown` `mouseup` `mouseover` `mouseout` `mouseenter` `mouseleave` `change` `select` `submit` `keyup` `keydown` `keypress` `keyup` `error`
- 第二个参数为可选参数，作为 `event.data` 属性值传递给事件对象的额外数据对象
- 第三个参数是用来绑定的处理函数

合成事件

jQuery 中有两个合成事件，`hover()` `toggle()`

hover()

`hover(fn1,fn2,...fnN)` 方法用于模拟光标悬停事件，当光标移动到元素上时，会触发第一个函数（mouseenter），当光标移出这个元素时会触发第二个函数（mouseleave）

toggle()

toggle() 方法用于模拟鼠标的连续点击事件，第一次单击元素，触发第一个函数，第二次单击同一个元素，会触发第二个函数，如果有更多的函数，则依次触发，直到最后一个。

事件冒泡

假设网页上有两个元素，其中一个嵌套在另一个元素里面，并且都被绑定了 click 事件。同时 `<body>` 元素上也绑定了 click 事件，这样的话，点击最内层的元素，会触发三次 click 事件。这是因为 JavaScript 的事件冒泡机制。

在 jQuery 中，提供了 `stopPropagation()` 方法来停止冒泡。

阻止默认事件

网页中有自己的默认行为，例如单击超链接会跳转，单击“提交”按钮后表单会提交，有时需要阻止默认行为。

jQuery 提供了 `preventDefault()` 方法来阻止元素的默认行为。

事件对象的属性

方法名称	描述
event.type	获取到事件的类型
event.preventDefault()	阻止默认的事件行为
stopPropagation()	阻止事件冒泡
event.tagent()	获取到触发事件的元素
event.relatedTarget()	mousover 和 mouseout 所发生的元素
event.pageX event.pageY	获取到光标相对于页面的 x 坐标和 y 坐标
event.which()	鼠标单击事件中获取到的左、中、右键，在键盘事件中获取键盘的按键
event.metaKey()	为键盘事件获取 `ctrl` 键

移除事件

```
unbind([type],[data])
```

第一个参数是事件类型，第二个参数是要移除的函数。如果没有参数，则删除所有的绑定事件

one() 方法

对于只要触发一次，随后要立即解除绑定的情况，jQuery 提供了 `one()` 方法。 当处理函数触发一次后，立即被删除。

jQuery中的动画

方法名	说明
hide() show()	同时修改多个样式属性，即高度、宽度和不透明度
fadeIn() fadeOut()	只改变不透明度
slideUp() slideDown()	只改变高度
toggle()	用来代替 hide() 和 show() 方法
slideToggle()	用来代替 slideUp() 和 slideDown()
fadeToggle()	用来代替 fadeIn() 和 fadeOut()
animate()	属于自定义动画的方法

jQuery 中的任何动画效果，都可以指定三种速度参数，slow、normal、fast，对应的时间长度分别是 0.6 秒，0.4 秒和 0.2 秒，也可以传入参数，传入数字作为参数不需要加引号，使用关键字需要加引号。

动画队列

当一个 `animate()` 方法中应用多个属性时，动画是同时发生的。 当以链式方法调用时，动画是按顺序发生（除非 `queue` 选项为 `false`）。 默认情况下，动画都是同时发生的。 当以回调的形式应用动画方式时，按照回调顺序发生。

停止动画

```
stop([clearQueue,gotoEnd])
```

`clearQueue` 是否要清空未执行的动画队列 `gotoEnd` 是否直接跳转到末状态

判断元素是否处于动画状态

要始终避免动画累计而导致的动画与用户行为不一样的情况。当用户快速在某个元素上执行 `animate()` 时，就会出现动画累加。

解决方法是判断元素是否处于动画状态，如果用户不处于动画状态，才为元素添加新的动画，否则不添加。

延迟动画

在动画执行的过程中，如果想对动画进行延迟操作，那么可以使用 `delay()` 方法。