正课:

1. 修改:

2. 按节点间关系查找:

3. 添加，删除，替换，克隆

4. 事件

1. 修改:

所有API都是一个函数两用！

内容:

$(...).html() elem.innerHTML

$(...).text() elem.textContent

$(...).val() elem.value

属性:

$(...).attr("属性名","值") elem.标准属性

如果同时修改多个属性:

$(...).attr({

属性名:值,

... : ...

})

$(...).prop("属性名","值") elem.状态属性

$(...).data("属性名","值") elem.dataset.属性名

样式:

修改单个css属性值:

$(...).css("属性名",值) elem.style.css属性

如果修改多个css属性值:

$(...).css({

css属性名:值,

... : ...

})

通过修改className，批量修改css属性值:

问题: elem.className 只能批量替换或清除className, 无法有选择的添加/移除指定class

解决: 用三个API代替className属性:

$(...).addClass("class")

$(...).removeClass("class")

$(...).hasClass("class")

$(...).toggleClass("class")

等效于: if($(...).hasClass("class"))

$(...).removeClass("class")

else

$(...).addClass("class")

2. 按节点间关系查找:

何时: 如果已经获得一个元素，找周围元素时

如何:

1. 父子:

$(...).parent() elem.parentNode

$(...).children(["selector"]) elem.children

$(...).children(":first-child") elem.firstElementChild

$(...).children(":last-child") elem.lastElementChild

$(...).find("selector") 所有后代元素中符合selector要求的元素

2. 兄弟:

$(...).prev() elem.previousElementSibling

$(...).prevAll() 之前所有兄弟

$(...).next() elem.nextElementSibling

$(...).nextAll() 之后所有兄弟

$(...).siblings() 除自己之外的前后所有兄弟

$(...).index(...)

$(...).index(DOM/$jq) 表示在左边的集合中找右边的元素是第几个

$(...).index("selector") 表示在右边的集合中找左边的元素是第几个

如果在同一个父元素下，找当前子元素的下标:

$(...).index();

3. 添加，删除，替换，克隆

添加: 2步

1. 用HTML代码片段创建新元素

var $any=$("html代码片段");

2. 添加到父元素下

用父元素添加子元素:

$parent.append($child) 末尾追加

$child.appendTo($parent)

$parent.prepend($child) 开头插入

$child.prependTo($parent)

在子元素前后插入兄弟:

$child1.before($child2)

在child1前插入child2

$child1.after($child2)

在child1后插入child2

更简化:

prepend,append,before,after中可直接传入HTML代码片段

删除: $(...).remove();

替换:

$(...).replaceWith("HTML代码片段"/$新元素)

用右边的新元素，代替左边选中的内容

$("HTML代码片段").replaceAll("selector")

用左边的新元素，代替右边选中的内容

克隆:

$(...).clone(); 默认只克隆属性，不克隆事件

$(...).clone(true); 即克隆属性，又克隆事件

$(...).is("selector") 判断$(...)选中的元素，是否符合右边选择器的要求

4. 事件绑定:

鄙视: jQuery中共有几种事件绑定方式:

DOM: elem.addEventListener("事件名",fun)

elem.removeEventListener("事件名",fun)

1. bind/unbind:

代替addEventListener/removeEventListener

用法完全相同

unbind三个重载:

unbind("事件名",函数名) 移除elem上指定事件名上绑定的指定名称的函数

unbind("事件名") 移除elem上指定事件绑定的所有处理函数

unbind() 移除elem上所有事件的处理函数

2. one:

绑定一个只能触发一次的处理函数，触发后，自动解绑！

3. live/die 已废弃

因为太极端，将所有事件处理函数都集中绑在document上

4. 简化利用冒泡: delegate/undelegate

$(parent).delegate("selector","事件名",fun)

将事件绑定在parent父元素上，要求只有满足selector要求的子元素才能触发事件

2个方面进行简化:

1. 用"selector"参数，代替了if判断

2. this->e.target, 不再指向父元素

5. 用on/off统一bind和delegate

$(...).on("事件名",fun) 完全等效于bind

$(parent).on("事件名","selector",fun)

完全等效于delegate

6. 常用事件都有终极简化:

$(...).事件名(fun)