**正课**

**1. 修改**

**2. 按节点间关系查找**

**3. 添加/删除/替换/克隆**

**4. 事件绑定**

**1. 修改:**

**内容:**

**原始HTML片段: $().html([新HTML片段]) .innerHTML**

**纯文本内容: $().text([新文本]) .textContent**

**表单元素的值: $().val([新值]) .value**

**清空内容: $().empty()**

**属性:**

**HTML标准属性: $().attr("属性名"[,值]) .get/setAttribute()**

**状态属性: $().prop("状态属性"[,值]) .属性名**

**自定义扩展属性: $().attr("属性名"[,值])**

**同时修改多个属性:**

**$().attr/prop({**

**属性名: 值,**

**... : ...**

**})**

**样式:**

**修改时都要修改内联样式:**

**$().css("css属性",值) .style.css属性**

**获取样式时要获得计算后的完整样式:**

**$().css("css属性") getComputeStyle(elem)**

**同时修改多个css属性:**

**$().css({**

**"css属性":值,**

**... : ...**

**})**

**问题: 如果批量修改多个css属性，代码繁琐**

**解决: 用class批量操作样式**

**$().addClass("class1 class2 ...")**

**$().removeClass("class1 class2 ...")**

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

**$().toggleClass("class") 在有与没有"class"两种状态间切换**

**等效于: if($btn.hasClass("down"))**

**$btn.removeClass("down")**

**else**

**$btn.addClass("down")**

**2. 按节点间关系查找:**

**2大类:**

**1. 父子关系:**

**$().parent() .parentNode**

**$().children([selector]) .children**

**$().find(selector) 在所有后代中查找符合selector要求的子元素，必须提供selector参数**

**$().children().first() .firstElementChild**

**$().children().last() .lastElementChild**

**2. 兄弟关系:**

**$().prev() .previousElementSibling**

**.next() .nextElementSibling**

**$().prevAll([selector]) 之前所有兄弟**

**$().nextAll([selector]) 之后所有兄弟**

**$().siblings([selector]) 除自己之外所有**

**3. 添加/删除/替换/克隆**

**添加: 2步:**

**1. 用$("html片段")创建新元素，HTML片段中已经包含了子元素和属性的定义**

**2. 将新元素添加到DOM树**

**$parent.append($新元素) //return $parent**

**.prepend($新元素)**

**$新元素.appendTo(parent) //return $新元素，可继续链式操作**

**.prependTo(parent)**

**$现有元素.before($新元素) //return $现有元素**

**$现有元素.after($新元素)**

**$新元素.insertBefore(现有元素) //return $新元素，可继续链式操作**

**.insertAfter(现有元素)**

**简写: 如果将代码片段添加到父元素后，不需要后续操作：**

**可直接$parent.append("html片段")**

**$("html片段").appendTo(parent)**

**删除: $().remove()**

**替换: $旧.replaceWith($新元素) //return $旧**

**$新元素.replaceAll(旧) //return $新元素**

**克隆: var $elem2=$elem.clone();**

**默认: .clone()是浅克隆，只复制样式和属性，不复制行为**

**修改:.clone(true)是深克隆，即复制样式和属性，又复制行为**

**4. 事件绑定:**

**鄙视: jQuery中共有几种事件绑定的方式**

**1. .bind("事件名",function(){ ... }) 代替.addEventListener()**

**.unbind("事件名",fun) 代替.removeEventListener();**

**.unbind("事件名") 移除该事件上所有处理函数**

**.unbind() 移除该元素上所有事件的的所有处理函数**

**2. .one("事件名",function(){ ... }) 绑定后只能触发一次，触发后自动解绑**

**3. .live/die() ——已废弃**

**4. 事件委托: 其实就是利用冒泡**

**$parent.delegate("selector", "事件名",function(){**

**this->e.target**

**不用if($tar.is("selector")){，应该将选择器写到参数中**

**})**

**$parent.undelegate(...)**

**5. 用on统一了bind和delegate**

**2种:**

**1. 普通绑定:**

**$elem.on("事件名",function(){ ... }) 代替.bind()**

**2. 利用冒泡:**

**$parent.on("事件名","selector",function(){**

**this->e.target**

**不用写if了。。。**

**})**

**.off(...)解绑**

**6. 对21种常用的事件，提供了简化版绑定的API:**

**$elem.事件名(function(){ ... })**

**不支持delegate简化版冒泡**

**页面加载后执行: 2种:**

**1. 仅DOM内容加载完就提前触发：仅等所有HTML和JS**

**DOMContentLoaded**

**何时: 只要不依赖于CSS和图片的代码，都要在DOM内容加载后，就提前执行。用户可早一步用上**

**比如: 所有事件绑定代码，都要放在DOMContentLoaded中**

**问题: DOMContentLoaded有兼容性问题**

**解决: $(document).ready(function(){ ... })**

**简化: $().ready(function(){ ... })**

**更简化: $(function(){ ... })**

**绝大多数jQuery代码都要放在$(function(){ ... })中**

**2. 整个页面内容加载完才触发: 等HTML，CSS，JS，图片**

**何时: 必须依赖于CSS和图片的代码，只能放在window.onload中**

**window.onload=function(){ ... }**

**问题: 用赋值方式绑定事件**

**解决: $(window).load(function(){ ... })**

**鼠标事件:**

**mouseover mouseout问题: 频繁冒泡**

**解决: 都用mouseenter mouseleave 代替over 和out**

**优点: 反复进出子元素，也不会触发父元素的事件**

**简写: 如果同时绑定mouseenter和mouseleave**

**可简写为hover(function(){ ... } , function(){ ... })**

**更简化: 如果两个处理函数可合并为一个处理函数**

**hover(function(){ ... }) function同时给了enter和leave**

**模拟触发:**

**即使没有点击按钮，也可执行按钮的操作**

**如何: $elem.trigger("事件名")**

**简写: $elem.事件名()**