**正课:**

**1. 修改:**

**用class批量修改样式**

**2. 添加，删除，替换，克隆**

**3. 事件绑定**

**1. 修改:**

**用class批量修改样式:**

**1. 为元素追加一个class: $(...).addClass("class名")**

**2. 为元素移除一个class: $(...).removeClass("class名")**

**3. 判断是否包含一个class: $(...).hasClass("class名")**

**4. 为元素切换一个class: $(...).toggleClass("class名")**

**if($(...).hasClass("class名"))**

**$(...).removeClass("class名")**

**else**

**$(...).addClass("class名")**

**补: .index()**

**2种:**

**1. var i=$("selector").index(jq对象/DOM对象)**

**查找右边的jq对象或DOM对象，在左边的结果集合中的下标位置**

**2. 如果在同一个父元素下找某个子元素的位置**

**var i=$("child").index();**

**2. 添加,删除,替换,克隆:**

**添加: 2步:**

**1. 用$()创建一个新元素: var $新元素=$("html片段")**

**2. 将新元素添加到dom树:**

**$("parent").append($新元素)**

**.prepend($新元素)**

**$("child").before($新元素)**

**.after($新元素)**

**可以更简化: $("parent").append/prepend("html片段")**

**$("child").before/after("HTML片段")**

**删除: $(...).remove();**

**补: .is("selector") 判断当前元素是否符合selector的条件**

**替换: $("selector").replaceWith(jq对象|DOM对象)**

**克隆: var $clone\_elem=$(...).clone()**

**强调: 默认浅克隆: 仅克隆样式和属性, 不可隆行为**

**深克隆: 即克隆样式和属性，又克隆行为**

**$(...).clone(true)**

**3. 事件绑定:**

**鄙视: jQuery中共有几种事件绑定方式，区别:**

**DOM: .addEventListener("事件名",handler)**

**.removeEventListener(...)**

**jq:**

**1. $("target").bind/unbind("事件名",handler)**

**同addEventListener()**

**.unbind三种重载:**

**.unbind("事件名",handler) 移除当前元素上，指定事件上的名为handler的处理函数。**

**.unbind("事件名") 移除当前元素上，指定事件上的所有处理函数**

**.unbind() 移除当前元素上，所有事件的监听**

**2. $("target").one("事件名",handle) 同bind**

**区别: 只触发一次，触发后，自动解绑**

**3. .live/die("事件名",handle)——已废弃**

**原理: 将所有事件集中绑定在顶级document上**

**4. $("parent").delegate("selector","事件名",handler)**

**原理: 简化利用冒泡:**

**1. 获得目标元素: this->e.target**

**2. 筛选目标元素: 第一个参数: "selector"**

**只有满足"selector"要求的元素，才能触发事件**

**鄙视: .bind vs .delegate: 3点**

**1. .bind直接帮在目标元素上**

**.delegate 帮在父元素上**

**2. 监听个数: .bind 监听个数多——每个目标元素都添加**

**.delegate 监听个数少——只给父元素添加一个**

**3. 新增子元素自动获得事件处理函数:**

**.bind 只能对现有元素添加事件监听**

**新增元素无法自动获得监听**

**.delegate 只要父元素下的元素，无论现有，还是新增，都能自动获得父元素上统一的事件监听**

**5. .on/off:**

**1. 代替bind: .on("事件名",handler) 同bind**

**2. 代替delegate: .on("事件名","selector",handler)同delegate**

**6. .事件名:**

**强调: 仅对常用的事件提供了终极简化**

**页面加载后执行: 2种:**

**1. DOMContentLoaded: 仅DOM内容加载完，就可提前执行**

**DOM内容仅包括: html和js**

**提前触发**

**何时: 只要不依赖于css和图片的所有操作都可在DOM内容加载后，提前触发**

**比如: 事件绑定**

**jq: $(document).ready(()=>{**

**//DOM内容加载后，就可提前执行的操作**

**//比如: 事件绑定**

**})**

**简化: $().ready(()=>{**

**更简化: $(()=>{...})**

**其实: 写在body结尾的script中的代码默认就是DOM内容加载后自动执行**

**2. window.onload 在所有页面内容加载完成后自动触发**

**包括: html,css,js,图片**

**何时: 如果js代码必须依赖css或图片才能执行时**

**鼠标事件:**

**mouseover mouseout 进出子元素，会频繁触发父元素的处理函数**

**mouseenter mouseleave 进入子元素，不再频繁触发父元素的处理函数**

**简写: 如果同时绑定鼠标进入和移出事件时，可简写为hover**

**$(...).hover(**

**e=>{...} //给mouseenter**

**,**

**e=>{...} //给mouseleave**

**)**

**更简化: 如果两个处理函数，可用toggle统一为一个处理函数，则只需要传一个参数即可**

**模拟触发:**

**虽然没有触发事件，但是依然可用程序模拟执行元素的事件处理函数**

**如何: $(...).trigger("事件名")**

**其实可以更简单: $(...).事件名()**

**总结: jQuery简化了DOM五大操作:**

**查找, 修改, 添加, 删除, 事件绑定**