正课:

1. 什么是jQuery

2. 如何使用jQuery  
3. 查找

4. 修改

1. 什么是jQuery:

第三方的，极简化的DOM操作函数库

为什么:

1. 是DOM操作的终极简化:

增删改查

事件绑定

动画

ajax

2. 解决了绝大部分浏览器兼容性问题

凡是jQuery允许使用的，都没有兼容性问题！

何时: 几乎所有大型项目，框架都是基于jQuery开发的

2. 如何使用jQuery:

鄙视: 如何快速上手一项新技术

1. 上官网: 看是什么

2. 看快速入门案例:

3. 下载手册:

官网: jquery.com

快速入门: 官网首页

手册: ftp中

下载:

版本: 1.x 支持旧浏览器

未压缩: 不带.min

拥有完整的代码格式，注释和见名知义的变量名

何时: 学习和开发时，可读性好，便于学习和查找问题

压缩: 带.min

去掉空格，注释, 极简化变量名

何时: 生产环境中, 体积小，便于快速下载

2.x 不再支持旧浏览器

3.x 支持更多的功能:

鄙视: 新增功能:

1. 所有API自动运行在严格模式下!

2. 支持包括Promise在内的ES6所有新技术

3. 支持for of代替each

4. ...

引入: 2种:

1. 引入服务器本地的jquery.js文件

2. 引入CDN网络中的jquery.js地址

CDN: 内容分发网络

官网->下载->CDN

注意: 先引jquery.js，再编写自定义脚本

原理: 引入jquery.js:

其实是在网页中添加了一种新的类型——jQuery

1. 构造函数: 负责创建该类型的子对象

2. 原型对象: 负责保存该类型的所有子对象共有的API

如果要对元素调用jQuery简化版API:

1. 必须用jQuery构造函数创建jQuery类型的子对象，包装要操作的DOM元素对象

为什么: 普通DOM元素不是jQuery类型的子对象，默认无法使用jQuery类型的简化版API

何时: 只要向对DOM元素执行简化版API时，都必须先创建jQuery类型的子对象，封装要操作的DOM元素

如何: 2种:

1. 用选择器查找DOM元素，并封装DOM元素

var $xxx=$("选择器");

2. 将已经获得的一个DOM元素直接封装

var $xxx=$(e.target/this);

什么是jQuery对象: 封装要操作的DOM元素的jQuery类型的子对象——类数组对象

jQuery API三大特点:

1. 自带forEach效果:

对jQuery对象调用一次API，等效于对jQuery对象中每个DOM元素分别调用一次API

2. 一个函数两用:

没给新值，就获取；给新值，就修改

3. 几乎每个API都返回当前正在操作的jQuery对象本身:

好处: 避免反复查找或创建对象，降低效率

用途: 链式操作

学习jQuery还是在学DOM:

查找触发事件的元素,绑定事件,查找要修改的元素,修改元素

3. 查找: jQuery主要按选择器查找

jQuery支持CSS3中所有选择器，且解决了兼容性问题

又扩展出了少数jQuery独有的选择器

强调: 优先使用CSS3支持的选择器, 内容排版引擎解析，效率高!

jQuery模拟出的选择器，都是用js实现的，效率低

jQuery中新增的:

1. 基本过滤/位置过滤: vs 子元素过滤相比

2. 内容过滤: 4种:

1. 按内容的文本过滤: :contains(text)

2. 按子元素特征: :has(selector)

3. 仅判断是否父元素: :parent()

4. 仅判断是否空元素: :empty()

3. 可见性过滤:

:hidden 查找或判断隐藏的元素

强调: 只能找到display:none的和type=hidden的

4. 状态过滤: :disabled :checked :selected

5. 表单元素过滤:

:input 选择所有表单元素: input textarea select button

每种type都对应着一种专门的选择器

:text :password :radio :checkbox :submit

:reset :button ...

4. 修改:

内容:

原始HTML代码片段: .innerHTML => .html()

纯文本: .textContent => .text()

值: .value => .val()

属性:

html标准属性: .attr("属性名"[,"值"])

状态: .prop("状态"[,bool])

自定义扩展属性:

样式:

1. 内联样式:

DOM:

修改: elem.style.css属性=值

获取: var style=getComputedStyle(elem)

var value=style.css属性

jQ: .css("css属性名"[,"属性值"])

两用: 提供属性值,就修改,自动执行.style.css属性,自动修改内联样式, 优先级最高，且当前元素独有

不提供属性值,就获取,自动执行getComputedStyle，从计算后的完整样式中获得最终属性值

福利: 数值类型的属性值,不用加px!

强调: 获取值时，结果是包含px的字符串

问题: .css每次只能修改一个css属性的值

解决: .css({

css属性:值,

... : ...

})

2. 样式表:

5. 按节点间关系查找: 6种:

1.父子: 4种+1种

.parentNode => .parent()

.children => .children([selector]) 只能查找直接子元素

.firstElementChild => .children(":first-child")

.lastElementChild => .children(":last-child")

\*\*\*.find(selector) 在所有后代中查找符合selector要求的元素

2. 兄弟: 2种:

.nextElementSibling => .next()

\*\*\*.nextAll([selector]) 之后所有兄弟

.previousElementSibling => .prev()

\*\*\*.prevAll([selector]) 之前所有兄弟

\*\*\*.siblings([selector]) 除我之外所有兄弟