day01 pm

正课:

1. 什么是:

2. 如何使用:

3. \*\*\*选择器:

1. 什么是:

jQuery是一个快速的，简洁的javascript库。

为什么: 2个原因:

1. 对DOM的终极简化:

核心DOM: 万能，繁琐

HTML DOM: 简单，不是万能

jquery: 4个方面提供了极简化的操作

1. 查找和操作元素(增，删，改，查)

2. 事件处理

3. 动画效果

4. AJAX操作

2. 屏蔽了99%的浏览器兼容性问题

只要jquery允许使用的，都是没有兼容性问题。

何时: 今后，绝大多数公司和前端框架都是利用jquery开发的。

口号: write less, do more

2. 如何使用:

版本:

1. 1.x: 特点: 兼容IE8

2个文件:

1. jquery-1.11.3.js: 未经压缩的可读性良好的js

2. jquery-1.11.3.min.js: 经过代码压缩的最小的js

去掉了所有注释，空白字符，换行。。。

2. 2.x: 特点: 不再兼容IE8

环境准备:

1. zen coding(ZC)->勾选Tab to expend和use emmit

标签名+Tab: 不全单个标签的属性和结束标签

标签名\*n+Tab: 同时生成n个标签，并不全属性和结束标签

标签名#id值+Tab: 自动生成id属性

标签名.class值+Tab: 自动生成class属性

标签名[属性名="值"]+Tab

标签名#id值.class值+Tab: 自动生成id属性和class属性

父标签>子标签+Tab: 自动生成父元素和子元素

兄弟标签+下一个兄弟标签+Tab: 自动生成多个平级的兄弟

()可提高优先级+Tab！

2. 随机生成一段假文: lorem+Tab

3. 常用快捷键:

Tools->preference->tools->key board->Edit

delete line: ctrl+d

duplicate line: ctrl+alt+↓

move up/donw : alt+↑/↓

line comment/uncomment: ctrl+/ ， ctrl+alt+/

如何使用jquery:

1. 引入jquery.js: <script src="jquery-1.11.3.js"></script>

强调: 必须在自定义script之前，先引入jquery.js

2. \*\*\*jQuery对象:

什么是: 封装一个/一组DOM元素，并提供对DOM元素执行操作的简化版API 的类数组对象。

为什么: 普通DOM对象无法使用jquery中简化版API

只有jQuery对象才能使用！

何时: 只要调用任何jqueryAPI之前必须先获得jQuery对象:

如何: 2种:

1. 先用DOMAPI获得DOM对象，再用jQuery工厂函数生成一个jQuery对象

var $jQuery对象=jQuery(DOM对象)

jQuery/$函数也称为工厂函数

2. 直接用jquery简化版API查找，自动获得的就是jQuery对象。

其实, jQuery可简写为$

万一jquery不支持了，希望使用DOM原生API:

从jquery对象中取出DOM 对象:

var dom对象=$jq对象.get(i)

鄙视: DOM对象 vs jq对象

DOM对象: 由W3C的DOM标准规定的，浏览器厂商负责实现的现成对象。

使用DOM API —— 万能，繁琐

jq对象: 由第三方jquery.js库定义的，并提供了简化操作DOM的API的 类数组对象

使用jQuery API——终极简化, 解决浏览器兼容性问题

两套API无法通用！

3. \*\*\*选择器:

今后，都是先找元素，再调用jqueryAPI操作元素

只要用jquery找元素，都用选择器！

用jquery查找元素，返回的直接就是jquery对象，无需再封装

如何: var $jq对象=$("选择器")

基本选择器: 5个: #id .class elem \* 选择器1,选择器2,...

层级选择器: 4个: 选择器1 选择器2 > + ~

练习: jQuery事件绑定: $("selector").事件名(function(){...})

访问元素的css样式:

$("selector").css("css属性名")

读取选中元素的css属性

$("selector").css("css属性名",值)

修改选中元素的css属性为"值"

强调: jquery多数简化版API都自带forEach功能:

$("selector").css("css属性名",值)

相当于对找到的每个元素都执行css操作！

$("selector").事件名(function(){...})

相当于对找到的每个元素都绑定事件！

过滤选择器: 通过特定的过滤规则筛选出所需的DOM元素

包含: 基本过滤(位置过滤), 内容过滤, 可见性过滤,

属性过滤, 子元素过滤和表单过滤

\*\*\*基本过滤: 位置过滤。根据元素在查找结果中的位置来选择

何时: 只要选择特定位置的元素时，都用基本过滤选择器

强调: 和css中的:first-child...伪类完全不一样！

包括:

:first 选择查找结果中排在第一个位置的元素

不考虑元素在父元素下的位置，仅考虑在jquery对象中的下标位置。

vs :first-child 作为父元素下第一个子元素的元素

:last 选择查找结果中排在最后位置的一个元素

vs :last-child 作为父元素下最后一个子元素的元素

:even/odd: 选择查找结果中排在偶数或奇数位置的元素

下标从0开始:

vs :nth-child(even): 下标从1开始

作为父元素下偶数位置的子元素

:eq(i): 选择查找结果中i位置的一个元素，i从0开始

vs :nth-child(i): 下标从1开始

作为父元素下第i个子元素

:gt(i): 选择查找结果中i位置之后的所有元素

:lt(i): 选择查找结果中i位置之前的所有元素

练习: 每个jqureyAPI默认都返回最初的查找结果对象本身

$("selector").css("...",...).css("...",...)....

链式操作

day02 am

快速学习一门新技术:

1. 百度百科/官网查是什么:

2. 官网: start 快速入门实例

3. 官网: 找手册:

正课:

1. \*\*\*选择器:

内容过滤

可见性过滤

\*\*\*属性过滤

1. 内容过滤: 根据内容查找元素

包括: 4个

:contains(text) 查找内容中包含指定文本的元素

比如: span:containes("购物车")

:empty 查找内容为空的元素

比如: span:empty

:parent 选择包含内容或子元素的父元素

比如: span:parent 其实就是:empty的反义

:has(selector) 选择内容中包含符合选择器要求的子元素的父元素

2. 可见性过滤: 2个

:visible 匹配所有可见的元素

:hidden 匹配所有不可见的元素，或type=hidden

display:none; type="hidden"

html+css: 4种隐藏元素的方法:

css: display:none; visibility:hidden; opacity:0

html: type="hidden"

3. \*\*\*属性过滤: 其实就是css中的属性选择器

优: 可见将任意属性作为查找条件！

支持模糊查找！

包含:

[属性名] 选择包含指定属性的元素

[属性名=值] 选择包含指定属性，且属性值精确等于"值"的元素

[属性名!=值] 选择包含指定属性，且属性值不等于"值"的元素——css中没有，jquery新加的

模糊查找:

[属性名^=值] 选择包含指定属性，且属性值以"值"开头的元素

[属性名$=值] 选择包含指定属性，且属性值以"值"结尾的元素

[属性名\*=值] 选择包含指定属性，且属性值中包含"值"的元素

复杂属性过滤: 多条件查询: [属性选择器1][属性选择器2]...

选择同时满足多个选择器条件的元素

练习:

添加/移除class: $("...").addClass("类名")

将指定类名追加到选中的元素上

$("...").removeClass("类名")

将指定类名从选中的元素上移除

$("...").hasClass("类名")

判断选中的元素中是否包含指定类名

$("...").toggleClass("类名")

有类名就移除，否则就添加

jquery事件处理函数中:

this->DOM->$(this)->jq对象

查找父元素: $("...").parent()

查找下一个兄弟: $("...").next()

day02 pm

正课:

1. \*\*\*选择器:

过滤:

子元素过滤:

表单选择器:

表单属性过滤选择器:

2. 基本操作:

修改: 访问属性: 标准属性; css样式

1. \*\*\*选择器:

过滤选择器:

子元素过滤: 其实就是CSS中的位置伪类:

:first-child :last-child :nth-child(n) :only-child

选择作为父元素下指定位置的子元素

强调: :nth-child(n) n从1开始

vs 基本过滤(位置过滤): 下标从0开始

先查找所有符合条件的元素，再选择结果集合中指定位置的元素

表单选择器: 根据表单元素的标签名和type属性值来选择表单中的元素

包括: :input选择所有input, textarea, select, button

:type类型: 选择指定type类型的表单元素

比如: :text 选择单行文本框

:password

:radio :checkbox

:submit :reset :button

:image

:file

:hidden

其实是: [type="xxxx"]的简化写法

表单属性过滤选择器: 其实就是css中的状态伪类:

:disabled :enabled :checked :selected

练习: 访问标准属性:

获取: $("...").attr("标准属性名")

修改: $("...").attr("标准属性名",值)

不能访问三大状态属性:

selected checked disabled

$("...").prop("标准属性名")

$("...").prop("标准属性名",值)

2. 基本操作:

内容:

html内容: $("...").html() -> elem.innerHTML

$("...").html("html代码片段") ->

elem.innerHTML="html代码片段"

清空内容: $("...").empty(); -> elem.innerHTML=""

文本内容: $("...").text() -> elem.textContent

$("...").text("文本") -> elem.textContent=文本

表单元素的内容: $("...").val() -> input.value

$("...").val(值) -> input.value=值

属性: $("...").attr("属性名") -> getAttribute("属性名");

$("...").attr("属性名",值) -> setAttribute("属性名",值)

$("...").removeAttr("属性名") ->removeAttribute(...)

访问状态属性:

$("...").prop("属性名")

$("...").prop("属性名",值)

样式:

1. 直接访问CSS样式属性:

只访问一个单独的css属性:

$("...").css("css属性名") -> 获取的就是计算后的样式！

getComputedStyle(elem).css属性名

$("...").css("css属性名",值)-> 仅设置内联样式

elem.style.css属性名=值

同时修改多个css属性:

$("...").css({

css属性名:值,

css属性名:值,

...

})

问题: 不便于大项目维护

解决: 大型项目中都要用操作class来代替操作单个样式属性

2. 通过操作class来操作样式:

$("...").hasClass("类名") 判断是否包含指定类型

$("...").addClass("类名") 为选中的元素追加指定类名

$("...").removeClass("类名") 移除选中的元素上的类名

$("...").toggleClass("类名") 在有与没有指定类名之间切换

相当于: 如果$("...").hasClass("类名")

$("...").removeClass("类名")

否则

$("...").addClass("类名")

day03 am

正课:

1. 遍历节点

2. 创建，插入，删除

3. 替换，复制

4. \*\*\*事件

1. 遍历节点:

父子关系

$("...").children(["selector"]) 仅在直接子元素中查找

$("...").find("selector") 在所有后代元素中查找符合选择器的

$("...").parent() 选择父元素

兄弟关系

$("...").next/prev(["selector"]) 要求必须紧邻的前/后一个兄弟。如果有选择器，还必须满足选择器的要求。

$("...").siblings(["selector"]) 除自己之外的前后所有兄弟元素。如果有选择器，还要满足选择器的要求

2. 创建，插入和删除:

DOM添加新节点: 3步: 1. 创建空元素，2. 设置关键属性，3. 将新元素添加到父元素下

jq中2步: 1. 用html代码片段创建元素对象(jq对象)

var $elem=$("元素的html代码片段")

仅在内存中

说明: 如果html代码片段中同时包含父元素和子元素，$会同时创建父元素以及子元素对象。

2. 将$elem，添加到指定父元素下:

$(parent).append($elem); 末尾追加

.prepend($elem); 在开头插入

相当于insertBefore(elem,first)

$(child).before($elem); 在child之前插入

.after($elem); 在child之后插入

删除节点: $(child).remove(); //删除child节点

3. 克隆和替换:

替换:

用后边，替换前边

$("要替换的元素").replaceWith("新元素的代码片段")

用前边，替换后边

$("新元素的代码片段").replaceAll("要替换的元素")

复制/克隆:

浅克隆，只复制属性，不复制行为

var $clone=$("...").clone();

深克隆, 不但复制属性，而且复制行为

var $clone=$("...").clone(true);

总结: jquery是对DOM的终极简化，学jquery还是在学DOM

jquery的核心: 增，删，改，查DOM元素

4. \*\*\*事件:

鄙视: jquery有几种事件绑定方式:

DOM: addEventListener("事件名",fn)

removeEventListener("事件名",函数名)

强调: 要想移除事件监听, 必须使用有名的函数绑定事件监听

如果添加事件监听时使用的是匿名函数,则不可能移除

1. $("...").bind("事件名",fn)

.unbind("事件名",函数名)

两个简化:

.unbind("事件名") 移除该事件上绑定的所有监听函数

.unbind() 移除该元素上所有事件绑定

2. $("...").one("事件名",fn)

绑定事件，仅触发一次后，自动解绑

3. 常用的事件: 23个, 更简化: $("...").事件名(fn)

只对部分常用事件提供了简化

bind函数的问题: 只能对当前页面上已有的元素添加事件绑定

解决: 将事件bind到父元素上——利用冒泡！

function(e){

var target=e.target;

if(target必须是...的元素)

}

4. 事件代理: 就是利用冒泡:

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

为parent绑定事件，规定只有符合selector要求的才能触发事件。——节省了if判断

强调: selector一定要相对parent下编写

自动将this替换为了e.target-->$(this)目标元素的jq对象，不用在获取目标元素

day03 pm

正课:

1. \*\*\*事件:

绑定

模拟操作

页面加载后

2. 动画效果: 了解

1. \*\*\*事件:

鄙视题: jquery中绑定事件处理函数有几种方式:

1. $("...").bind("事件名",fn)

.unbind("事件名",fn)

.unbind("事件名")

.unbind()

2. $("...").one("事件名",fn)

3. $("...").事件名(fn)

问题: bind只能对页面现有的元素绑定事件

动态生成的元素无法自动绑定事件

解决1: 利用冒泡: 将事件绑定到父元素上一次

function(e){

var target=e.target

if(target必须是...的元素)

... ...

}

解决2: delegate

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

selector代替了fn中的if，自动筛选符合条件的子元素

只有符合selector要求的子元素才能出发事件

this->代替了e.target->$(this)->目标元素的jq对象

.undelegate("selector", "事件名",fn)

鄙视: delegate vs bind

1. 事件监听的个数:

delegate是仅在父元素添加一次事件监听

bind通常都是在每个子元素都添加事件监听

2. 对动态添加的元素:

delegate让动态添加的元素自动响应指定事件

bind通常仅对现有元素添加绑定，无法自动给动态生成的元素添加绑定

4. 已废弃: $("selector").live("事件名",fn)

$(document).delegate("selector","事件名",fn)

$("selector").die("事件名",fn)

$(document).undelegate("selector","事件",fn)

5. 终极简化: $("...").on("事件名",fn) ->bind

$("parent").on("事件名","selector",fn)

.delegate("selector","事件名",fn)

模拟操作: 用代码模拟触发另一个元素的事件

何时: 如果少量/个别平级元素拥有相同的事件处理函数

不必利用冒泡！就用模拟触发

如何: $("...").trigger("事件名")

触发所有选中元素上的指定事件处理函数

\*\*\*页面加载后执行:

只要将script放在body的结尾，默认就是页面加载后执行

2种:

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

在页面加载后自动触发

问题: onload: 整个加载完成: html , css , js , 图片...

onload中通常执行DOM操作，和css和图片无关

解决: ondomcontentLoaded: 仅DOM树和js加载完成

其实domcontentLoaded时就已经可以执行DOM操作

如何: $(document).ready(function(){...})

ondomcontentLoaded:

简化: $().ready(function(){...})

更简化: $(function(){...})

其实: 将script放在body结尾，就已经是domcontentLoaded

补: hover:

hover=mouseover+mouseout

何时: 今后只要进入元素和移出元素时，都要执行事件操作

如何: $("...").hover(fn1, fn2): over时执行fn1,out时执行fn2

$("...").hover(fn) 无论over还是out都执行fn

2. 动画效果: 了解

jquery中通过动画函数实现的动画效果——高级函数

问题: 1. 高级函数，是用底层函数，再次进行的封装，可能发生变化

2. 可能有兼容性问题

解决: 简单动画: fade+in+transition

简单动画函数:

显示隐藏: $("...").show(); $("...").hide(); $("...").toggle();

->display:none 默认无动画效果:

添加参数: 速度: 默认三挡: slow normal fast

自定义毫秒数:

问题: 直接修改大量内联css属性实现的动画, 不可维护

上下滑动: $("...").slideUp(); $("...").slideDown();

$("...").slideToggle();

淡入淡出: $("...").fadeIn(); $("...").fadeOut();

$("...").fadeToggle();

$("...").动画API(速度,easing,callback)

callback指在动画调用结束后，才调用callback

万能动画函数:

$("...").animate(

{css属性: 目标值, css属性: 目标值},//仅对数值属性有效

speed,

easing,

callback

)

并发与排队:

并发: 多个动画效果同时执行!

$("...").animate({多个属性},speed)

多个属性同时变化

排队: 多个动画先后执行:

$("...").animate({先变化的属性},speed)

.animate({后变化的属性},speed)

练习:

$("...").each(function(i){$(this)...})

对类数组对象中每个元素执行相同的操作

总结: 几乎所有jquery API都返回之前的jquery对象，可用链式操作减少反复调用$(),避免反复查询和反复创建jq对象。

day04 am

正课:

1. 类数组对象操作:

2. \*\*\*jQuery UI

1. 类数组对象操作:

.size() 获得jquery对象查询结果中元素的个数

.get(i) 获得jquery对象中i位置的DOM元素

.index(DOM/jq) 查找元素在jquery对象中的下标位置

经常用于判断是否包含在内

.each(callback) 对查询结果中每个元素执行相同的操作

callback(i){this->当前正在遍历的DOM元素->$(this)->jq}

2. \*\*\*jqueryUI: 对jquery API的更上层封装，更简化版本

只是封装了很多固定的功能和插件。——复用

jqueryUI: PC

jqueryMobile: 移动设备

官网: jqueryui.com

如何: 将jquery-ui-1.11.4文件夹拷贝到项目目录(已完成)

分三部分:

Effects效果:

Interactions:交互组件

Widgets: 小部件

Effects: 效果: 三大改变

addClass/removeClass/toggleClass: 添加了动画效果

比如: toggleClass("类名",speed,callback)

show/hide/toggle: 添加了更多动画效果：

比如: .toggle("效果名",speed)

blind(遮蔽) bounce(弹跳) explode(爆炸)...

color animation: 实现了颜色动画

.animate方法支持颜色动画

Interactions:交互组件:

1. $("...").draggable(): 让选中的元素可拖拽

侵入性: 指第三方程序通过修改宿主程序的部分内容，达到效果

2. $("...").selectable(): 让目标可选择

需要自己实现.ui-selected,选中项的样式

可多选

3. $("...").sortable();

day05 am

正课:

1. \*\*\*自定义插件:

2. cookie

1. \*\*\*自定义插件:

何时: 只要一项功能或一个函数，被反复使用。都要将反复使用的函数或功能封装为一个插件。

优: 复用——DRY——Reflector重构

如何: 2种:

1. 为jQuery添加jQuery全局函数

什么是: 不依赖于任何具体DOM对象，就可直接调用的jQuery函数。

何时: 只要一个函数不需要查找任何DOM对象，就可直接调用时。就要将该函数定义为jquery全局函数

如何:

所有jquery的全局函数都定义在jQuery全局对象中

jQuery全局对象充当jQuery所有API的命名空间

Step1: if(window.$===undefined)

//先判断是否加载了jQuery库

Step2: 在jQuery中添加自定义命名空间对象

$.XXXLib={};

Step3: 在自定义命名空间中添加自定义扩展方法

$.XXXLib.方法名=function(){...}

如何调用: 3步:

1. 先引jquery.js

2. 再引自定义lib.js

3. 调用: $.xxxLib.方法名();

鄙视: jQuery全局函数 vs ES的全局函数

相同点: 执行，不依赖于某一类具体对象

ES的全局函数，存储在全局对象window中

不需要任何对象可直接调用

jQuery全局函数, 存储在jQuery对象中

必须至少使用jQuery.才能调用

鄙视: $.each(list,callback) vs $("...").each(callback)

相同: 都是对类数组对象中每个元素执行相同操作(callback)

不同:

1. 调用方式：

$.each可对任意类数组对象执行操作

$("...").each只能对$查询出的结果集合执行操作

2. 存储位置:

$.each: 直接存在jQuery对象下

$("...").each: jQuery.fn中

jQuery.fn是所有查询结果对象的父对象！

3. 称呼:

$.each: jQuery全局函数

$("...").each: 称为jQuery对象方法

2. 为jQuery对象添加插件方法:

何时: 只要一项完整的效果/功能需要复用，就要将效果/功能封装为jQuery插件函数

如何: jQuery插件函数都要添加到jQuery.fn中

不需要动态生成html, 但是事先规定好html的样子

div>a/button+ul

为组件定义必须的css，保存在专门的css文件中

开始为jQuery.fn添加对象dropdown()方法

先侵入class

再寻找触发事件的元素，绑定处理函数

修改class，达到交互的目的

如何使用:

1. 引入插件的css文件

2. 引入jquery.js

3. 引入插件的js文件

4. 找到父元素，调用插件方法

$("...").dropdown();