1~4天课程目录

第一天（HTML）

> HTML不是一门编程语言，是一门标签语言。

HTML基本结构

HTML语法

> 自闭标签

> 转义字符

> 标签属性

> HTML标签不区分大小写

> 1. HTML注释：

字体标签：

属性

> color

> size(1~7) 默认为3

标题标签<h1~6>

属性

> align

> left

> rigth

> center

> justify

列表标签`<ul>`/`<li>`

```
<ul type="disc">
    <li>Spring</li>
    <li>Summer</li>
    <li>Autumn</li>
    <li>Winter</li>
</ul>
```

属性

> type 图标样式
>
> > circle 空心圆
> >
> > disc 实心圆
> >
> > square 实心方块

图片标签`<img src="" alt="">`

属性

> src 图片的路径
>
> alt 图片无法加载时代替图片出现的文字
>
> width 宽度 px %
>
> height 高度 px %

锚标签`<a name="" href="">`

> 添加一个书签：`<a name="tag"></a>`
>
> 跳转到指定书签：`<a href="#tag">点击返回上层</a>`

> `<a href="http://www.tmooc.cn" target="_self">点击跳转到tmooc</a>`

表格标签`<table>`

> tr 表格中的行标签
>
> td 表格中的单元格
>
> th 表格中的标题单元格

`<caption>` 定义表格的标题

table的重要属性

> border 边框宽度
>
> cellspacing 单元格之间的空白距离
>
> cellpadding 边框与单元格内容之间的距离
>
> bgcolor 背景颜色
>
> bordercolor 边框颜色
>
> width 宽度
>
> align 对齐方式

tr重要属性:

> align 对齐方式
>
> bgcolor 背景颜色

th/td重要属性

> align 对齐方式

> bgcolor 背景颜色
>
> width 宽度
>
> height 高度
>
> colspan 可横跨的列数
>
> rowspan 可竖跨的行数

第二天（js）

> 脚本语言，没有编译过程，解释运行。

基本数据类型

> 数值类型(Number)、字符串(String)、布尔类型(Boolean)、undefined、null

> Number
>
> > 代表数字的基本类型。JS不区分整形和浮点型，JS中的所有数值类型底层实现都是浮点型
> >
> > > Infinity 无穷大
> > >
> > > -Infinity 负无穷大
> > >
> > > NaN 非数字
>
> String
>
> > Js中的字符串是基本数据类型，字符串常量必须用双引号引起来
>
> Boolean
>
> > Boolean类型只有两个值，true、false。
>
> Undefined
>
> > Undefined类型只有一个值就是undefined，表示变量未定义
>
> Null
>
> > Null只有一个值就是null。null用来表示尚未存在的对象

复杂数据类型

> 对象、数组、函数

定义变量

```
//局部变量
var x = 99;
x = "aa";
x = true;
x = new Object();

//全局变量
function mx(){
    i = 0;
}
```

js语法-语句

> for循环
>
> while
>
> do while
>
> switch case
>
> if

函数的定义

普通函数定义方式

```
function mx(参数列表){
    函数体
}
mx()
```

动态函数定义

```
var mq = new Function("a","b","return a+b");
console.log(mq(2,3));
```

直接量定义函数(匿名函数)

```
var ma = function(a,b){
    return a+b;
}
console.log(ma(4,5));
```

js语法-数组

定义数组

```
var arr = new Array();

var arr = new Array(3);

var arr = new Array(1,"b",true,new Object());

直接量定义数组
var arr = [1,2,3];
```

操作数组

> 向指定下标添加元素
>
> > arr[index]
>
> 移除元素
>
> > pop()最后一个
> >
> > shift()第一个
>
> 添加元素
>
> > push()
>
> 遍历元素
>
> > for循环

js语法-对象

js的内置对象

> String -- 基本数据类型 字符串类型 的包装对象
>
> Boolean -- 基本数据类型 布尔类型 的包装对象
>
> Nubmer -- 基本数据类型 数值类型 的包装对象
>
> Array -- 数组类型 的包装对象
>
> Function -- 函数类型 的包装对象
>
> Math -- 数据对象，封装了很多数学常量和数学方法
>
> Date -- 日期时间对象，封装了很多和日期实现相关的方法
>
> Global -- 全局对象。

自定义对象

1. 无参构造函数定义对象

```
function Person(){
}
var p = new Person();
```

2.有参构造函数定义对象

```
function Person(name,age){
    this.name = name;
    this.age = age;
}
```

3.直接量定义对象

```
var p = {name:"lishuai",age:18}
```

第三天（js&&jquery）

JSON

> JSON本质上就是一段字符串，能够保存较复杂关系的数据，具有良好的数据保存格式，又极为轻量

JSON对象：

```
var data ={
    name:"曹洋",
    age:18,
    addr:"安徽",
    girlfriends:[
        {name:"貂蝉",job:"跳舞",age:18},
        {name:"西施",job:"唱歌",age:19}
    ]
}
console.log(data["girlfriends"][1]["job"]);
```

JSON字符串：

```
var data ={
    "name":"曹洋",
    "age":"18",
    "addr":"安徽",
    "girlfriends":[
        {"name":"貂蝉","job":"跳舞","age":"18"},
        {"name":"西施","job":"唱歌","age":"19"}
    ]
}
```

> 两者的区别：json对象的属性名称没有双引号。json串属性名称和属性值都必须使用双引号。
>
> 两者之间可以转换

jQuery概述

> jQuery可以实现写的更少而做的更多。它是一个由js函数组成的函数库

jQuery的引入

```
<!--引入jQuery函数库-->
 <script type="text/javascript" src="jquery-1.4.2.js"></script>
```

jQuery语法

> jQuery <==> $("")

> 操作css选择器
>
> > $("#test").text("abc");
> >
> > > 表示将页面id为test的元素选中，并且设置其中的text文本内容。
>
> 文档就绪事件
>
> > $(document).ready(function(){ });
> >
> > $().ready(function(){ });
> >
> > $(function(){ });

js对象和jQuery对象相互转换

js对象转换为jQuery对象：

> 将js对象使用$()包起来，这样js对象就会变为一个jQuery对象，这时只能使用jQuery对象身上的api

jQuery对象转换为js对象：

> jQuery选中的元素可以看做是将这些元素放入一个数组中，根据下标获取某一个元素，就素获取数组中的一个js对象。

```
//1.js转换为jQuery
    /*$(function(){
        var div = document.getElementById("test");//js对象
        //转换为jQuery
        alert($(div).text());
    });*/

    //2.jQuery转换为js
    $(function(){
        $("div");//jQuery对象
        var d = $("div")[0];//转换为js
        //$("div").get(0)//转换为js
        alert(d.innerText);
});
```

定义一个变量

`$a = $("div");`

`$b = document.getElementsByTagName("div");`

使用$xxx定义一个变量只是为增强代码可读性，并不是只能在jQuery对象身上使用这样的对象。js同样可以。

第四天（jquery&&mysql）

jQuery选择器

> 元素名选择器
>
> > `$("div") - 匹配所有的div元素`
>
> class选择器
>
> > `$(".c1") - 匹配所有class值为c1 的元素`
> >
> > `$("div.c1") - 匹配所有class值为c1的div元素`
>
> id选择器
>
> > `$("#d1") - 匹配所有id值为d1的元素`
> >
> > `$("div#d1") - 匹配所有id值为d1的div元素`
>
> *号匹配符
>
> > `$("*") - 匹配所有的元素`
>
> 多元素选择器
>
> > `$("div,span,#d1,.c1") - 匹配所有的div/span元素以及id值为d1的元素和class值为c1的元素`

层级选择器（selector案例二）

> `$("div span") - 匹配div下所有的span元素 $("div>span") - 匹配div下所有的span子元素 $("div+span") - 匹配div后面紧邻的span兄弟元素 $("div~span") - 匹配div后面所有的span兄弟元素`

基本过滤选择器（selector案例三）

> `$("div:first") - 匹配所有div中的第一个div元素 $("div:last") - 匹配所有div中的最后一个div元素 $("div:even") - 匹配所有div中索引值为偶数的div元素，0开始 $("div:odd") - 匹配所有div中索引值为奇数的div元素，0开始 $("div:eq(n)") - 匹配所有div中索引值为n的div元素，0开始 $("div:lt(n)") - 匹配所有div中索引值小于n的div元素，0开始 $("div:gt(n)") - 匹配所有div中索引值大于n的div元素，0开始 $("div:not(.one)") - 匹配所有class值不为one的div元素`

内容选择器（selector案例四）

> 见笔记

可见选择器

> `$("div:hidden") – 匹配所有隐藏的div元素 $("div:visible") – 匹配所有可见的div元素`

属性选择器（selector案例五）

> `$("div[id]") – 匹配所有具有id属性的div元素 $("div[id='d1']") – 匹配所有具有id属性并且值为d1的div元素 $("div[id!='d1']") – 匹配所有id属性值不为d1的div元素 $("div[id^='d1']") – 匹配所有id属性值以d1开头的div元素 $("div[id$='d1']") – 匹配所有id属性值以d1结尾的div元素`

子元素选择器

> `$("div:nth-child(n)") – n从1开始，匹配div中第n个子元素。 $("div:first-child") –匹配div中第1个子元素。 $("div:last-child") –匹配div中最后一个子元素。。。`

表单选择器（selector案例六）

> `$(":input") – 匹配所有的input文本框、密码框、单选框、复选框、select 框、textarea、button。 $(":password") – 匹配所有的密码框 $(":radio") – 匹配所有的单选框 $(":checkbox") – 匹配所有的复选框 $(":checked") – 匹配所有的被选中的单选框/复选框/option $("input:checked") – 匹配所有的被选中的单选框/复选框 $(":selected") – 匹配所有被选中的option选项`

文档操作

> parent()
>
> > $("#d1").parent() – 获取id为d1元素的父元素
>
> parents()
>
> > `$("#d1").parents() – 获取id为d1元素的祖先元素 $("#d1").parents("tr") – 获取id为d1元素的tr祖先元素`
>
> next()
>
> > `$("div").next() – 获取所匹配元素后面紧邻的兄弟元素 $("div").next("span") – 获取所匹配元素后面紧邻的span兄弟元素`
>
> nextAll()
>
> > `$("div").nextAll() – 获取所匹配元素后面所有的兄弟元素 $("div").nextAll("span") – 获取所匹配元素后面所有的span兄弟元素`
>
> prev()
>
> > `$("div").prev() – 获取所匹配元素前面紧邻的兄弟元素 $("div").prev("span") – 获取所匹配元素前面紧邻的span兄弟元素`
>
> prevAll()
>
> > `$("div").prevAll() – 获取所匹配元素前面所有的兄弟元素 $("div").prevAll("span") – 获取所匹配元素前面所有的span兄弟元素`
>
> siblings()

> `$("div").siblings() – 获取所匹配元素前后所有的兄弟元素 $("div").siblings("span") – 获取所匹配元素前后所有的span兄弟元素`

append()

> `$("div").append("") –为所匹配元素追加一个span子元素`

remove()

> $("div").remove() – 删除所匹配元素

html()

> `$("div").html() – 获取所匹配元素的html内容 $("div").html("xxx") – 为所匹配元素设置html内容`

text()

> `$("div").text() – 获取所匹配元素的文本内容 $("div").text("xxx") – 为所匹配元素设置文本内容`

attr()

> `$("div").attr("id") – 获取所匹配元素的id属性值 $("div").attr("id", "d2") – 为所匹配元素设置id属性`

css

> `$("div").css("width") – 获取所匹配元素的width样式属性值 $("div").css("width", "200px") – 为所匹配元素设置width样式属性 $("div").css({"width":"200px", "color":"red","font-size":"24px" }) ; – 为所匹配元素设置width样式属性`

事件

click()

> $("div").click(function(){}) – 为所匹配元素绑定点击事件

blur()

> $("input").blur(function(){}) – 为所匹配元素绑定失去输入焦点事件

focus()

> $("input").focus(function(){}) – 为所匹配元素绑定获得输入焦点事件

change()

> $("select").change(function(){}) – 为所匹配元素绑定选项切换事件

ready()

> $(document).ready(function(){}) – 文档就绪事件 其作用相当于: window.onload = function(){}
>
> 简写形式为:
>
> $(function(){}) – 在整个文档加载完成后立即执行

效果

> show()

> $("div").show() – 将隐藏元素设置为显示(底层操作的是display);

hide()

> $("div").hide() – 将显示元素设置为隐藏(底层操作的是display);

toggle()

> $("div").toggle() – 切换元素的可见状态, 如果元素显示则设置为隐藏, 如果元素隐藏则设置为可见.