

JavaScript入门

今日内容介绍

- ◆ 使用 JS 完成表格的隔行换色
- ◆ 使用 JS 完成复选框的全选效果
- ◆ 使用 JS 完成省市联动效果

今日内容学习目标

- ◆ 使用 JS 可以获得指定元素
- ◆ 使用 JS 可以创建元素
- ◆ 使用 JS 可以对元素的属性进行操作
- ◆ 使用 JS 可以对元素的标签体进行操作
- ◆ 使用 JS 可以对指定元素的样式进行操作（获得或修改）

第1章 案例：表格隔行换色

1.1 案例描述

开发中，需要使用表格陈列数据，数据过多不易查看，通常使用隔行换色进行表示。

<input type="checkbox"/>	1	手机数码	手机数码类商品	修改 删除
<input checked="" type="checkbox"/>	2	电脑办公	电脑办公类商品	修改 删除
<input type="checkbox"/>	3	鞋靴箱包	鞋靴箱包类商品	修改 删除
<input checked="" type="checkbox"/>	4	家居饰品	家居饰品类商品	修改 删除

1.2 案例相关 JS 函数介绍

1.2.1 相关 JS 事件

- `onmouseover()` 鼠标移入事件。鼠标从外部移入到当前元素时触发。
- `onmouseout()` 鼠标移出事件。鼠标从当前元素移出时触发。
- `onload()` 页面加载成功触发
- 方式 1: `<body>` 使用 `onload` 属性确定需要执行的函数

```
<head>
    <meta charset="UTF-8">
    <title></title>
    <script type="text/javascript">
        //js 代码在<body>之前，不能获得<body>标签体中的内容，还没有加载到
        var e01 = document.getElementById("e01");
        alert(e01); //打印: null

        function init () {
            //页面加载成功之后执行
            var e02 = document.getElementById("e01");
            alert(e02.value); //打印: 传智播客
        }
    </script>
</head>
<body onload="init()">
    <input type="text" name="" id="e01" value="传智播客" />
</body>
```

- 方式 2: 通过 `window.onload` 设置匿名函数

```
<head>
    <meta charset="UTF-8">
    <title></title>
    <script type="text/javascript">
        window.onload = function () {
            //页面加载成功之后执行
            var e02 = document.getElementById("e01");
            alert(e02.value); //打印: 传智播客
        }
    </script>
</head>
<body>
    <input type="text" name="" id="e01" value="传智播客" />
</body>
```

1.2.2 this 关键字

- 在函数内部 this 表示：当前操作的元素。

1.2.3 setAttribute(name,value)设置属性

- this.setAttribute(name,value) 给当前元素设置属性

1.3 案例实现

- 在提供 html 页面的基础上，编写 js 代码



```
<script type="text/javascript">
    window.onload = function () {
        var allTr = document.getElementsByTagName("tr");
        // 跳过前 2 行
        for (var i = 2 ; i < allTr.length ; i++) {
            //给行 tr 设置背景颜色，奇数行白色，偶数行指定颜色
            if(i % 2 == 0){
                allTr[i].style.backgroundColor="#FFF";
            } else {
                allTr[i].style.backgroundColor="#4E7FD1";
            }

            /* 添加事件，鼠标移上和移出，背景色改变。
             * * 使用自定义属性 “bgc” 缓存当前行自己原来的颜色
             */
            allTr[i].onmouseover = function(){
                this.setAttribute("bgc",this.style.backgroundColor);
                this.style.backgroundColor = "#A0B9E1";
            }
            allTr[i].onmouseout = function(){
                this.style.backgroundColor = this.getAttribute("bgc");
            }
        }
    }

```

```
        }
    </script>
```

第2章 案例：复选框全选/全不选

2.1 案例描述

开发中，经常需要对表格数据进行“批量处理”，就需要快速的对列表项进行操作，本案例我们来完成“全选和全不选”

2.2 案例相关的 JS 属性介绍

2.2.1 单选/复选选中

ele.checked 表示元素是否选中，true 表示选中，false 表示没有选中

例如：ele.checked = true; //设置元素被选中。

2.3 案例实现

- 步骤 1：给复选框添加 onclick 事件

```
<input type="checkbox" onclick="selectAll(this)" >
```

- 步骤 2：编写 selectAll(this) 处理列表项的复选框是否勾选

```
<script type="text/javascript">
    function selectAll (obj) {
        //当前复选框是否选择,如果选中其他都选中,如果没有选中其他都不选中。
        // * getElementsByTagName 通过标签 class 属性的名称获得对应的所有标签。<xx
class="">
        var allCheckbox = document.getElementsByTagName("itemSelect");
        for (var i = 0 ; i < allCheckbox.length ; i ++) {
            allCheckbox[i].checked = obj.checked;
        }
    }
</script>
```

2.4 扩展：反选

- 步骤 1：提供超链接，用于触发事件

```
<!-- href: 用于指定<a>跳转的位置。
```

开发中为了阻止 href 所设置的位置，通常编写“javascript:void(0)”，void() 函数参数任意数字都可以，习惯写成 0

```
-->
```

```
<a href="javascript:void(0)" onclick="inverseSelect()">反选</a>
```

- 步骤 2：编写 inverseSelect() 函数

```
function inverseSelect() {  
    // 反选：如果选择就不选中，如果不选中就选中  
    var allCheckbox = document.getElementsByClassName("itemSelect");  
    for (var i = 0 ; i < allCheckbox.length ; i++) {  
        allCheckbox[i].checked = !allCheckbox[i].checked;  
    }  
}
```

第3章 案例：省市二级联动

3.1 案例介绍

在日常应用中，我们需要完善个人信息，“所在地”要求选择省市，当选择省时，该省对应的市将自动的更新。

The screenshot shows a user interface for selecting a location. At the top, there are two dropdown menus labeled "所在地". The first dropdown has options like "---请选择-省---" and "---请选择-市---". The second dropdown is currently empty. Below this, there is another set of dropdown menus for "所在地" with options "北京" and "市辖区".

3.2 案例相关的 JS 函数

3.2.1 数组：Array

创建语法

```
new Array();  
new Array(size); // size 数组元素个数，数组成员默认值 undefined
```

```
new Array(element0, element0, ..., elementn); //参数列表，为数组初始化数据
```

- 数组中的每一个成员没有类型限制，及可以存放任意类型
- 数组的长度可以自动修改，类似 Java 中的 List 集合等。

3.2.2 元素操作：createElement、appendChild

```
document.createElement() 创建元素节点。
```

```
ele.appendChild() 向标签体末尾添加新的子节点。
```

3.3 案例实现

1. 步骤 1：给注册页面添加 select 标签

```
01.省市二级级联.html
104 105    <select id="provinceId" onchange="selectCity(this.value)"> 106        <option value="">----请-选-择-省----</option> 107        <option value="0">北京</option> 108        <option value="1">吉林省</option> 109        <option value="2">山东省</option> 110        <option value="3">河北省</option> 111        <option value="4">江苏省</option> 112    </select> 113    <select id="cityId" style="width:150px"> 114        <option value="">----请-选-择-市----</option> 115    </select> 116 <select id="provinceId" onchange="selectCity(this.value)" style="width:150px">     <option value="">----请-选-择-省----</option>     <option value="0">北京</option>     <option value="1">吉林省</option>     <option value="2">山东省</option>     <option value="3">河北省</option>     <option value="4">江苏省</option> </select> <select id="cityId" style="width:150px">     <option value="">----请-选-择-市----</option> </select> | | |
```

2. 步骤 2：js 实现

```
<script type="text/javascript">
```

```

// 定义二维数组，初始化数据
var cities = new Array(4);
cities[0] = new Array("市辖区", "县");
cities[1] = new Array("长春市", "吉林市", "松原市", "延边市");
cities[2] = new Array("济南市", "青岛市", "烟台市", "潍坊市", "淄博市");
cities[3] = new Array("石家庄市", "唐山市", "邯郸市", "廊坊市");
cities[4] = new Array("南京市", "苏州市", "扬州市", "无锡市");

//通过选择的省，显示对应的所有的市
function selectCity(index){
    //通过索引获得对应的所有的市
    var allCity = cities[index];
    // 获得 city select 对象
    var cityObj = document.getElementById("cityId");
    cityObj.innerHTML = "<option value='>----请-选-择-市----</option>";
    //遍历所有的市
    for(var i = 0 ; i < allCity.length ; i ++){
        // 获得具体的市
        var cityName = allCity[i];
        // 创建 option
        var option = document.createElement("option");
        // 创建文本节点
        var textNode = document.createTextNode(cityName);
        //将文本添加到 option 中
        option.appendChild(textNode);
        // 将 option 追加到 select 中
        cityObj.appendChild(option);
    }
}
</script>

```

3.4 总结：DOM

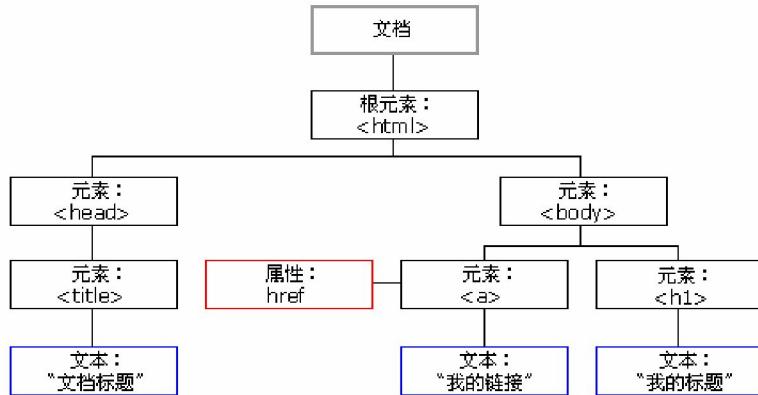
从昨天到现在，所有的案例中，我们获得元素，创建元素等操作，统称为 DOM 操作。接下来我们一起总结一下 DOM。

3.4.1 什么是 DOM

- DOM: Document Object Model 文档对象模型，定义访问和操作结构化文档（HTML）的方式。
 - 创建的结构化文档: html、xml 等
 - DOM 包括: 核心 DOM、HTML DOM、XML DOM。通常情况下 HTML DOM 和 XML DOM 是

可以相互使用的。

- HTML DOM 将 整个 HTML 文档呈现成一颗 **DOM** 树，树中有元素、属性、文本等成员。



3.4.2 document 文档对象

- 浏览器加载整个 HTML 文档形成 Document 对象，Document 对象可以访问和操作 HTML 文档中的所有元素。
- 获得元素

`getElementsById()` 通过 id 属性值获得元素（整个 HTML 文档 id 位置）

`<xxx id="">`

`getElementsByName()` 通过 name 属性值获得所有元素（整个 HTML 文档中 name 可能相同）

`<xxx name="">`

`getElementsByClassName()` 通过 class 属性值获得所有元素

`<xxx class="">`

`getElementsByTagName()` 通过标签名获得所有的元素

`<xxx>`

- 创建元素

`createElement()` 创建指定名称的元素

- 常见属性

`childNodes`, 获得所有的子节点

`nodeName`, 返回节点名称。(标签名)

`nodeType`, 返回节点类型。(元素、属性、文本 等)

`nodeValue`, 节点的值。(只有文本节点才有该属性)

3.4.3 element 元素对象

- Element 对象表示 HTML 文档中的元素 (HTML 称为标签)。元素可包含属性、其他元素或文本。也就是说 HTML 标签可以包含属性，其他子标签或文本。

`appendChild()` 给元素追加子元素

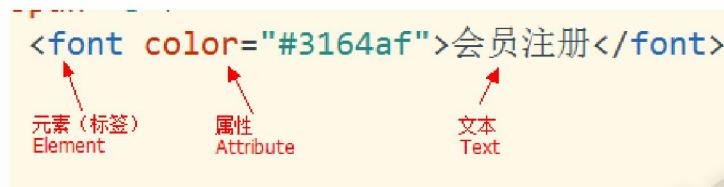
`<a>`

`...`

```
追加位置  
</a>  
insertBefore() 给当前元素前追加兄弟元素  
插入位置  
<a>  
setAttribute(k,v) 给元素设置属性  
<xxx k=v >
```

3.4.4 名称解释

- 元素（标签）Element、属性 Attribute、文本 Text 统称为：节点 Node



3.5 总结

3.5.1 全局函数

分类	函数名	描述
转换	parseFloat()	解析一个字符串并返回一个浮点数。
	parseInt()	解析一个字符串并返回一个整数。
执行	eval()	计算 JavaScript 字符串，并把它作为脚本代码来执行。
编码	encodeURI()	把字符串编码为 URI。
	decodeURI()	解码某个编码的 URI。