



传智播客 · 前端与移动开发学院
<http://web.itcast.cn>

前端基本功—javascript 第四天

目录

目录	2
1. 复习	4
2. 两个小循环	4
3. 多分支语句 switch	4
4. 下拉菜单的事件 onchange	6
5. 数组常用方法	7
5.1.添加数组的内容	7
5.2.删除数组的内容	8
5.3.连接两个数组	8
5.4.join() 把数组连接成字符串	8
5.5.把字符串转换为数组 split()	8
6. DOM (重点)	9
6.1.DOM 定义	9
6.2.节点	10
6.3.访问节点	10
6.4.封装自己class类	10
7. 判断真假	11
8. 访问关系	11
8.1.父节点	12
8.2.兄弟节点	12
8.3.子节点	13
8.4.孩子节点	13
9. dom 的节点操作	15
9.1.创建节点	15
9.2.插入节点	15
9.3.移除节点	16
9.4.克隆节点	16
10. 作业	17

1. 复习

1. 数组 看电影 电影院 座位

大的变量 里面可以放很多的值

```
var arr = [1,3,57];
```

```
var ar = new Array();   new object();   new Date()
```

```
var txt = ["宋江","张飞"];
```

使用数组: 数组名[索引号] txt[1] == 张飞

txt.length; 属性

遍历数组:

```
for(var i=0;i<txt.length;i++){ console.log( txt[i] )}
```

txt[i] txt 数组

2. 两个小循环

循环 for(初始化; 条件; 增量){}

while() 当 do {} while()

while(条件) { 语句 }

```
var j = 1;
while(j<=100)
{
    sum1+=j;
    j++;
}
console.log(sum1);
```

do while 至少执行一次 while 不一定

3. 多分支语句 switch

switch 跟 if else if else if else 几乎一样的 但是switch效率更好。

作用其实就是: 多选1 从多个里面选1个。

语法格式:

```
switch(参数)
{
    case 参数1:
        语句;
        break; 退出的意思
    case 参数2:
        语句;
        break; 退出的意思
    .....
    default: 默认的
        语句;
}
```

```
<script>
window.onload = function(){

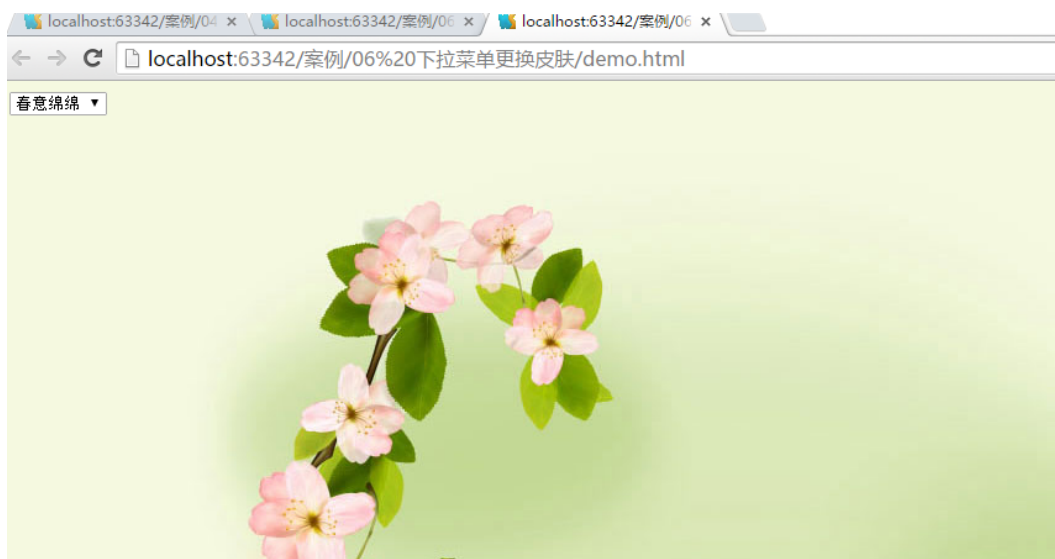
    //获取元素
    var txt = document.getElementById("txt");
    var btn = document.getElementById("btn");
    btn.onclick = function(){
        var val = txt.value;
        switch(val)
        {
            case "苹果":
                alert("苹果的价格是：5元");
                break;
            case "香蕉":
                alert("香蕉的价格是：2元");
                break;
            case "梨":
                alert("梨的价格是：1.5元");
                break;
            case "大白菜":
                alert("大白菜的价格是：9毛");
                break;
            default:
                alert("今天没进货");
        }
    }
}
</script>
```

4. 下拉菜单的事件 onchange

sele.onchange = function(){}
当改变的时候 事件

```
<script>
window.onload = function(){
    var sele = document.getElementById("sele");
    sele.onchange = function(){
        // alert(this.value);
        switch(this.value)
        {
            case "1":
                document.body.style.backgroundColor = "url(images/
chun1.jpg)";
                break;
            case "2":
                document.body.style.backgroundColor = "url(images/
xia1.jpg)";
                break;
            case "3":
                document.body.style.backgroundColor = "url(images/
qiu1.jpg)";
                break;
            case "4":
                document.body.style.backgroundColor = "url(images/
dong1.jpg)";
                break;
        }
    }
}
</script>
```

效果：



5. 数组常用方法

我们经常要对数组进行操作，可能追加，也可能删除 等等，何如？

5.1. 添加数组的内容

```
var arr = [1,3,5];
```

我们想要吧 7 这个数字 放到 这个数组的后面

我想要 [1,3,5,7];

1. push() ★★★★★ 后面推进去

push() 方法可向数组的末尾添加一个或多个元素，并返回新的长度。

push 推进去 放

var arr =[1,3,5] → arr.push(7) → 结果变成：[1,3,5,7];

2. unshift() 从数组的前面放入

unshift() 方法可向数组的开头添加一个或更多元素，并返回新的长度

var arr = [1,3,5] → arr.unshift(0) → 结果变成 [0,1,3,5]

注意：

```
var dom = [1,3,5];  
console.log(dom.push(7)); // 返回的是 数组的长度 4
```

5.2.删除数组的内容

1. pop() 删除最后一个元素

pop() 移除最后一个元素

返回值 是 **返回最后一个值**

var arr = [1,3,5] → arr.pop() → 结果 [1,3]

2. shift() 删除第一个元素

shift() 方法用于把数组的第一个元素从其中删除，**并返回第一个元素的值**

var arr = [1,3,5] → arr.shift() → 结果 [3,5]

5.3.连接两个数组

concat()

该方法用于连接两个或多个数组。它不会改变现有的数组，而仅仅会返回被连接数组的一个副本

var aa = [1,3,5]; var bb = ["a","b","c"];

aa.concat(bb); 结果: [1,3,5,"a","b","c"];

5.4.join() 把数组连接成字符串

join()

将数组各个元素 通过指定的分隔符进行连接成为一个字符串。

语法:

数组名.join(符号)

符号可选。指定要使用的分隔符。如果省略该参数，则使用逗号作为分隔符。

var arr = [1,2,3];

console.log(arr.join("-")) 结果就是: 1-2-3 字符串

5.5.把字符串转换为数组 split()

join <=> split

split() 方法用于把一个字符串分割成字符串数组

语法

stringObject.split(separator,howmany)

参数 separator 可选。指定要使用的分隔符。如果省略该参数，则使用逗号作为分隔符。

howmany 可选。该参数可指定返回的数组的最大长度

```
7 console.log(txt); // 数组中为什么没有换行符
8
9 // 字符 转换为 数组
10 var txt = "aa-bb-cc";
11 console.log(txt.split("-"));
12
13 </script>
14 </body>
15 </html>
```

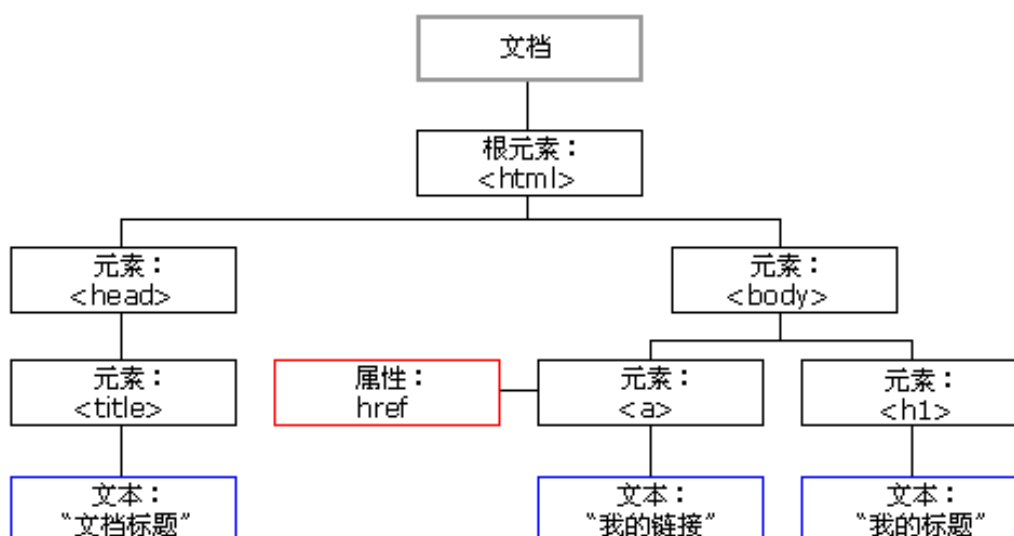
这两个符号保持一致

6. DOM (重点)

window.alert() 很大的兼容问题

6.1.DOM 定义

DOM 树



6.2.节点

标签 标记 元素 节点

由结构图中我们可以看到，整个文档就是一个文档节点。

而每一个HTML标签都是一个元素节点。

标签中的文字则是文字节点。

标签的属性是属性节点。

一切都是节点.....

6.3.访问节点

我们学过几个访问节点：

getElementById() id 访问节点

getElementsByTagName() 标签访问节点

getElementsByClassName() 类名 有兼容性问题

主流浏览器支持 ie 6 7 8 不认识

怎么办？我们自己封装自己的类。

6.4.封装自己class类

原理：（核心）

我们要取出所有的盒子，利用遍历的方法，通过每一个盒子的className 来判断。如果相等就留下。

```
<script>
    window.onload = function(){
        //封装自己class类名
        function getClass(classname){
            //如果浏览器支持，则直接返回
            if(document.getElementsByClassName)
            {
                return document.getElementsByClassName(classname);
            }
            // 不支持的 浏览器
            var arr = []; // 用于存放满足的数组
            var dom = document.getElementsByTagName("*");
            //alert(dom.length);
            for(var i=0;i<dom.length;i++)
            {
                if(dom[i].className == classname)
                {

```

```

        arr.push(dom[i]);
    }
}
return arr;
}
console.log(getClass("demo").length);
}
</script>

```

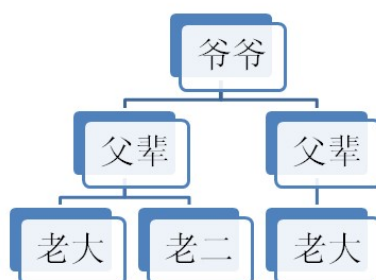
结束，分割版本

7. 判断真假

我们用条件语句来判断5大数据类型中的真假；	
数据	结论
数字类型	所有数字都是真，0是假
字符串	所有字符串都是真，''串是假
对象	所有对象都是真，null是假
未定义	undefined是假，没有真

8. 访问关系

各个节点的相互关系



父节点	兄弟节点	子节点	所有子节点
parentNode	nextSibling	firstChild	childNodes
	nextElementSibling	firstElementChild	children
	previousSibling	lastChild	
	previousElementSibling	lastElementChild	

8.1. 父节点

父： parentNode 亲的 一层

```
<script>
  window.onload = function(){
    var x = document.getElementById("x");
    x.onclick = function(){
      this.parentNode.style.display = "none";
      // 关掉的是他的 父亲
    }
  }
</script>
```

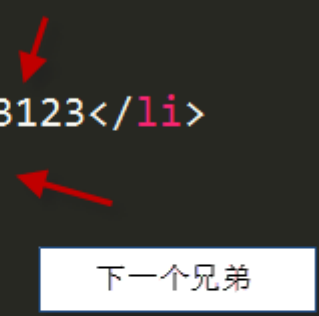
8.2. 兄弟节点

nextSibling 下一个兄弟 亲的 ie 678 认识
nextElementSibling 其他浏览器认识的

previousSibling 同理 上一个兄弟
previousElementSibling

我们想要兼容 我们可以合写 || 或者

```
<li>123123</li>
<li>123123</li>
<li id="one">123123</li>
<li>123123</li>
<li>123123</li>
<li>123123</li>
<li>123123</li>
<li>123123</li>
</ul>
</body>
</html>
```



```
var div = one.nextElementSibling ||
one.nextSibling;
div.style.backgroundColor = "red";
```

必须先写 正常浏览器 后写 ie678

8.3. 子节点

firstChild 第一个孩子 ie678

firstElementChild 第一个孩子 正常浏览器

var one.firstElementChild || one.firstChild;

lastChild

lastElementChild

8.4. 孩子节点

childNodes 选出全部的孩子

childNodes: 它是**标准属性**，它返回指定元素的子元素集合，包括HTML节点，所有属性节点，文本节点（嫡出）

火狐 谷歌等高本版会把换行也看做是子节点

```
7 <body>
8 <ul id="ul">
9     <li>123</li>
10    <li>123</li>
11    <li>123</li>
12 </ul>
13 <script>
```

节点 7个

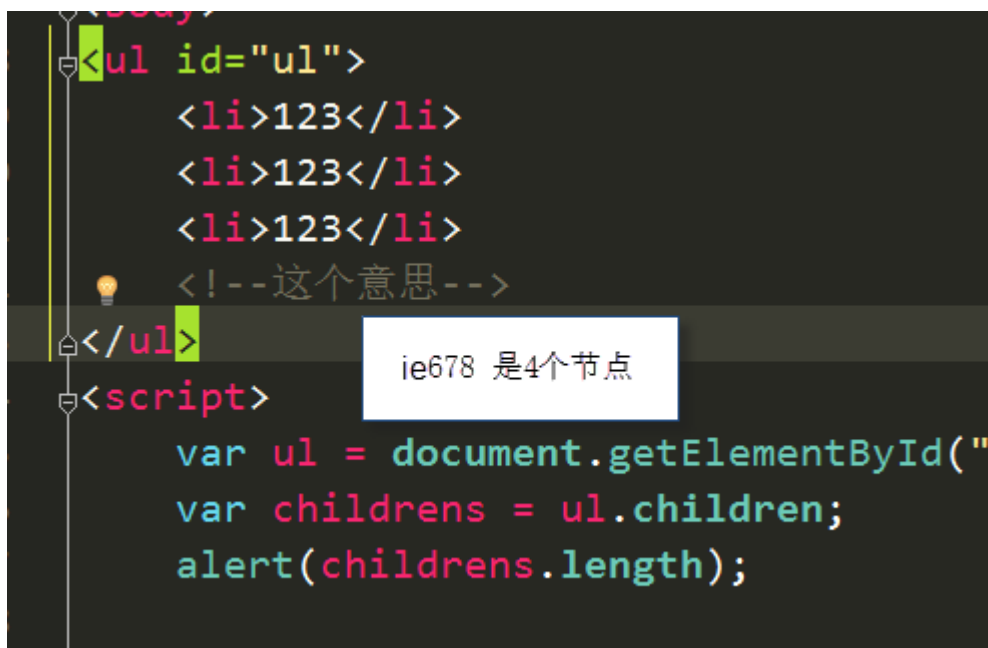
利用 属性 `nodeType == 1` 时才是元素节点 来获取元素节点

```
<script>
    var ul = document.getElementById("ul");
    var childrens = ul.childNodes; // 选择全部的孩子 亲的
    //alert(childrens.length);
    for(var i=0;i<childrens.length;i++)
    {
        if(childrens[i].nodeType == 1)
        {
            childrens[i].style.backgroundColor = "pink";
        }
    }
</script>
```

children 重要 选取所有的孩子（只有元素节点）

这个更好 跟喜欢它。（庶出）

ie 678 包含 注释节点 这个要避免开。



9. dom 的节点操作

新建节点 插入节点 删除节点 克隆节点 等等

9.1. 创建节点

```
var div = document.createElement("li");
```

上面的意思就是 生成一个新的 li 标签

9.2. 插入节点

1. appendChild(); 添加孩子 append 添加的意思
意思： 添加孩子 放到盒子的 **最后面**。
2. insertBefore(插入的节点, 参照节点) 子节点 添加孩子
写满两个参数

```
demo.insertBefore(test, childrens[0]);
```

放到了第一个孩子的前面

如果第二个参数 为 null 则 默认这新生成的盒子放到最后面。

```
demo.insertBefore(test, null);
```

9.3. 移除节点

`removeChild()` 孩子节点

```
var da = document.getElementById("xiongda");  
demo.removeChild(da);
```

9.4. 克隆节点

`cloneNode()`;

复制节点

括号里面可以跟参数，如果里面是 `true` 深层复制，除了复制本盒子，还复制子节点

如果为 `false` 浅层复制 只复制 本节点 不复制 子节点。

10. 作业

作业

- 利用 图片的张数 动态生成 li 个数

