

# JavaScript进阶

---脚本化文档（扩展）



河北师范大学软件学院  
Software College of Hebei Normal University

# 内容提纲

---

- **DOM操作**
- **表单及表单操作**



# DOM综述

---

- DOM ( Document Object Model ) : 文档对象模型
  - 浏览器提供的操作 HTML 文档内容的应用程序接口
  - 用于对文档DOM树进行动态操作，增加、修改、删除、读取
  - HTML文档中所有内容都是节点，称为DOM节点
  - DOM节点类型有12个，之前介绍了3个，元素节点、属性节点和文本节点，最常用的是元素节点
  - DOM节点有父子层次关系

# DOM节点访问



# DOM节点访问

## • DOM节点访问方法梳理

-标红部分之前没有介绍

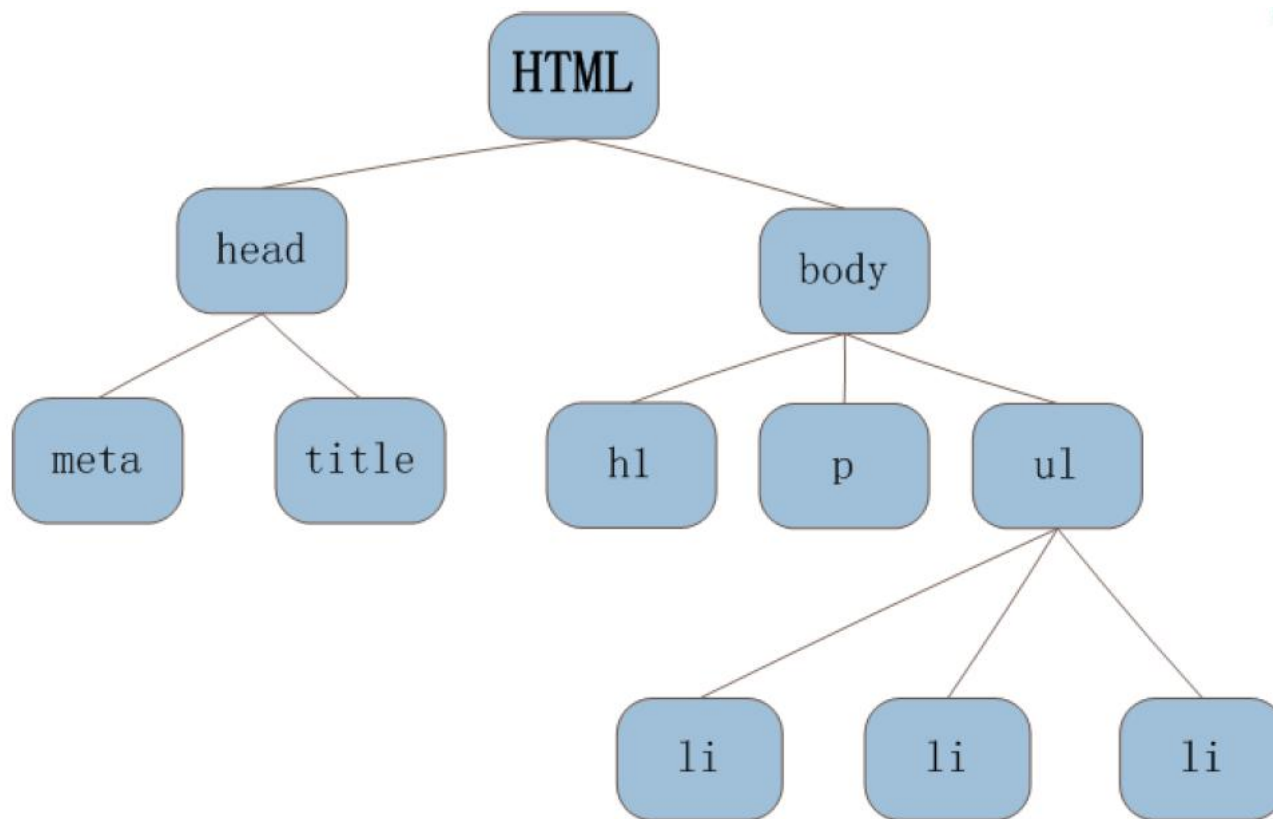
获取方式	获取描述	JavaScript方法
直接获取节点	通过id属性获取	document.getElementById()
	通过标签名获取	document.getElementsByTagName()
	通过类名获取	document.getElementsByClassName()
	通过name属性获取	document.getElementsByName()
通过节点关系获取节点	通过父节点获取子节点	node.childNodes node.firstChild node.lastChild node.children node.firstChild node.lastEelementChild
	通过子节点获取父节点	node.parentNode node.parentElement
	获取前后兄弟节点	node.previousSibling node.nextSibling node.previousElementSibling node.nextElementSibling

# DOM节点访问

- DOM文档遍历有两种方式

- 作为元素树的遍历

- 以HTML标签元素为最小单位

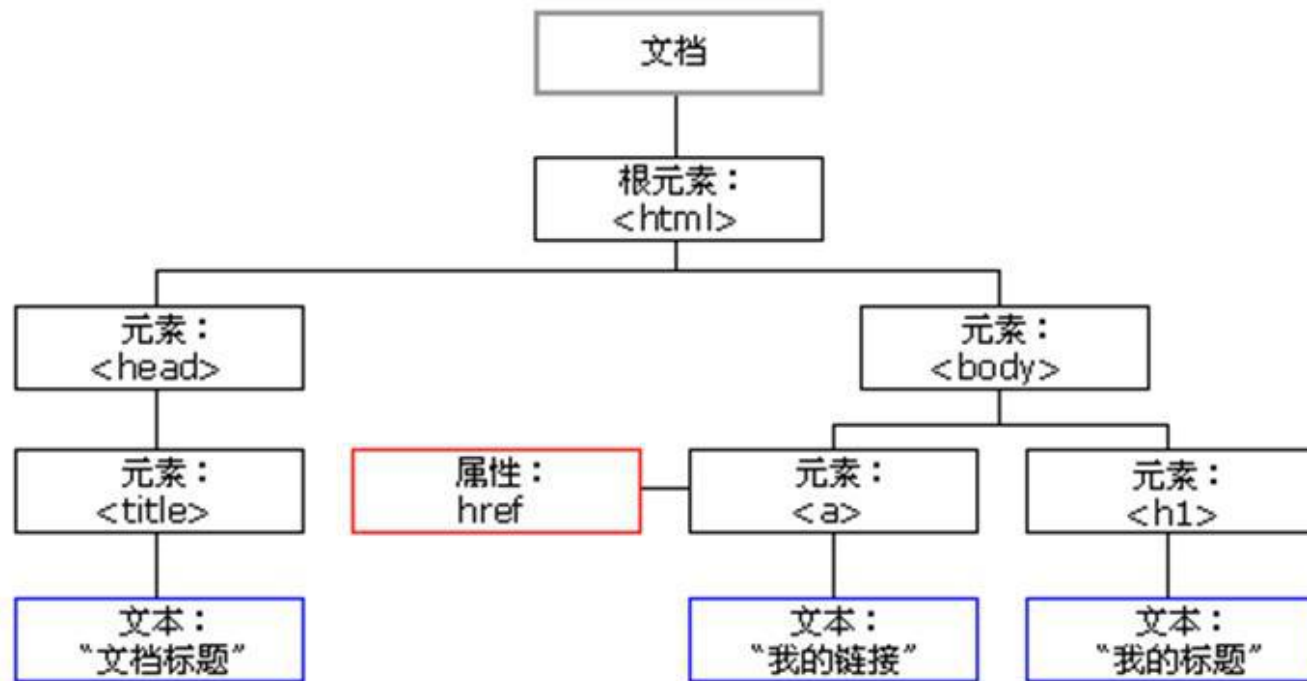


# DOM节点访问

- DOM文档遍历第二种方式

-作为节点树的遍历

- HTML所有内容都是节点
- 包括了元素树中的元素节点
- 元素节点中的属性和内容都是节点





# DOM节点访问

- 通过节点关系获取节点的操作方法分两类

-注意两类方法名称的区别

获取描述	操作分类	JavaScript方法
通过父节点获取子节点	元素树	node.children node.firstElementChild node.lastElementChild
	节点树	node.childNodes node.firstChild node.lastChild
通过子节点获取父节点	元素树	node.parentElement
	节点树	node.parentNode
获取前后兄弟节点	元素树	node.previousElementSibling node.nextElementSibling
	节点树	node.previousSibling node.nextSibling





# 元素树和节点树的区别

`<h1>新闻动态<span id= "time" >2016-10-7</span> <span>更多  
</span> </h1>`

- 元素树访问方法：

```
var h=document.getElementsByTagName( 'h1' )[0].children;  
console.log(h.length);//输出 2  h的类型是HTMLCollection
```

- 节点树的访问方法：

```
var h=document.getElementsByTagName( 'h1' )[0].childNodes;  
console.log(h.length);//输出 4  h的类型是NodeList
```

参见实例index01.html

# 元素树和节点树的区别

`<h1>新闻动态<span id= "time" >2016-10-7</span> <span>更多  
</span> </h1>`

- 两种访问方法获取元素节点是相同的：

```
var domH1 = document.getElementsByTagName( 'h1' )[0];
```

```
var domElemSpan = domH1.firstElementChild;//元素树获取方法
```

```
var domNodeSpan=domH1.childNodes[1];//节点树获取方法
```

```
console.log(domElemSpan === domNodeSpan);//输出 true
```

# 元素树和节点树的区别

---

- 一个不同，一个相同
  - 获取一组节点时返回的类型不同
    - 元素树的是HTMLCollection，元素集合
    - 节点树的是NodeList，节点集合
  - 获取的节点是元素节点时两种方法的返回值是相同的



# DOM节点操作

操作类型	方法描述	操作方法
创建节点	创建元素节点	<code>document.createElement(tagName)</code>
	创建文本节点	<code>document.createTextNode(text)</code>
插入节点	在父节点追加节点	<code>parentNode.appendChild(node)</code>
	在节点前增加节点	<code>parentNode.insertBefore(node,postionNode)</code>
删除节点	从父节点删除节点	<code>parentNode.removeChild(node)</code>
修改节点	修改属性和文本	文本节点 <code>textNode.nodeValue = '文本'</code> 元素节点 <code>elmentNode.attrName = attrValue</code> <code>elmentNode.innerHTML = HTML内容</code>
	替换节点	<code>parentNode.replaceChild(newNode,oldNode)</code>

# DOM节点操作

---

- 节点操作方法列表，创建、



## 本节小结

---

- 回顾DOM操作
- 元素树和节点树的不同



# 内容提纲

---

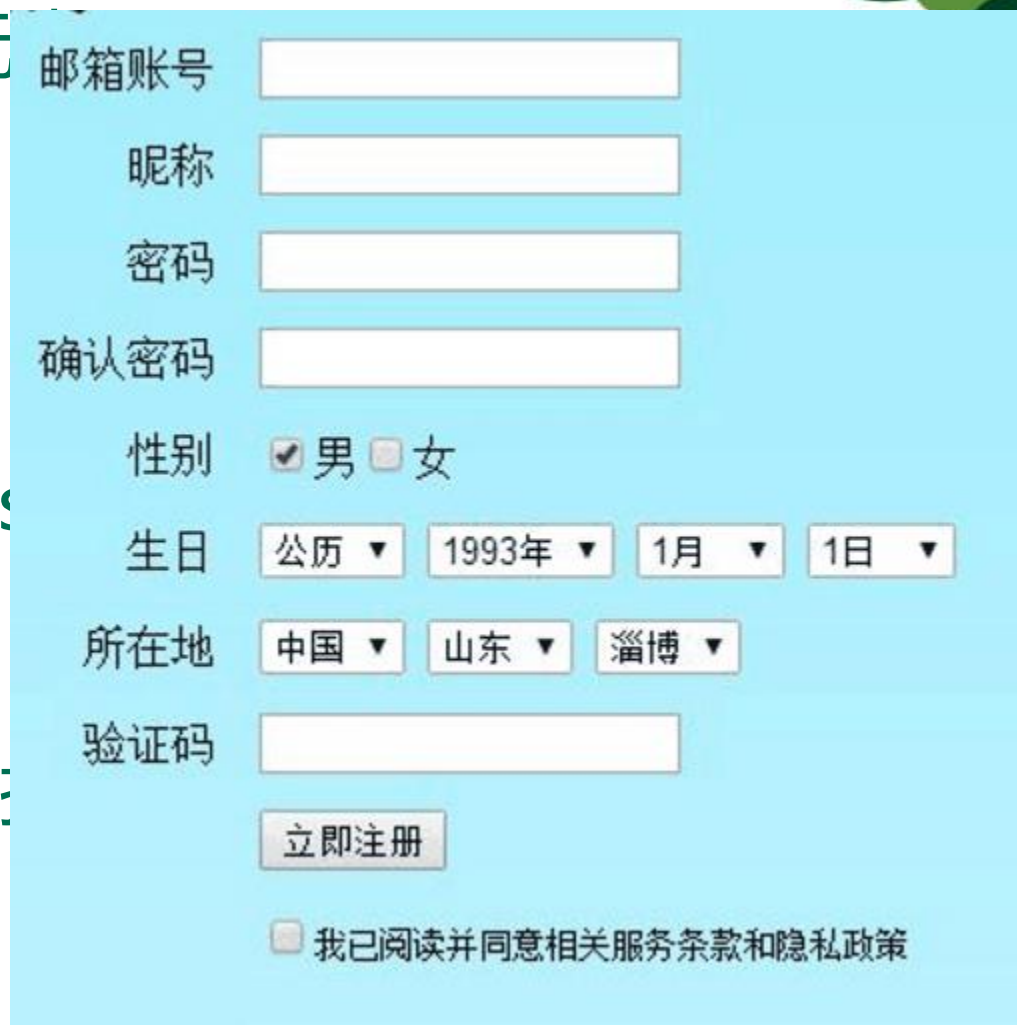
- DOM操作
- 表单及表单操作





# 表单

- 表单元素是重要的用户交互元素
- 表单用于收集用户输入
- 表单定义在<form>元素中
- 表单元素包括有<input>、<select>、<textarea>和<button>
- JavaScript常用来检查表单数据



邮箱账号

昵称

密码

确认密码

性别 ☒ 男 ☐ 女

生日 公历 ▼ 1993年 ▼ 1月 ▼ 1日 ▼

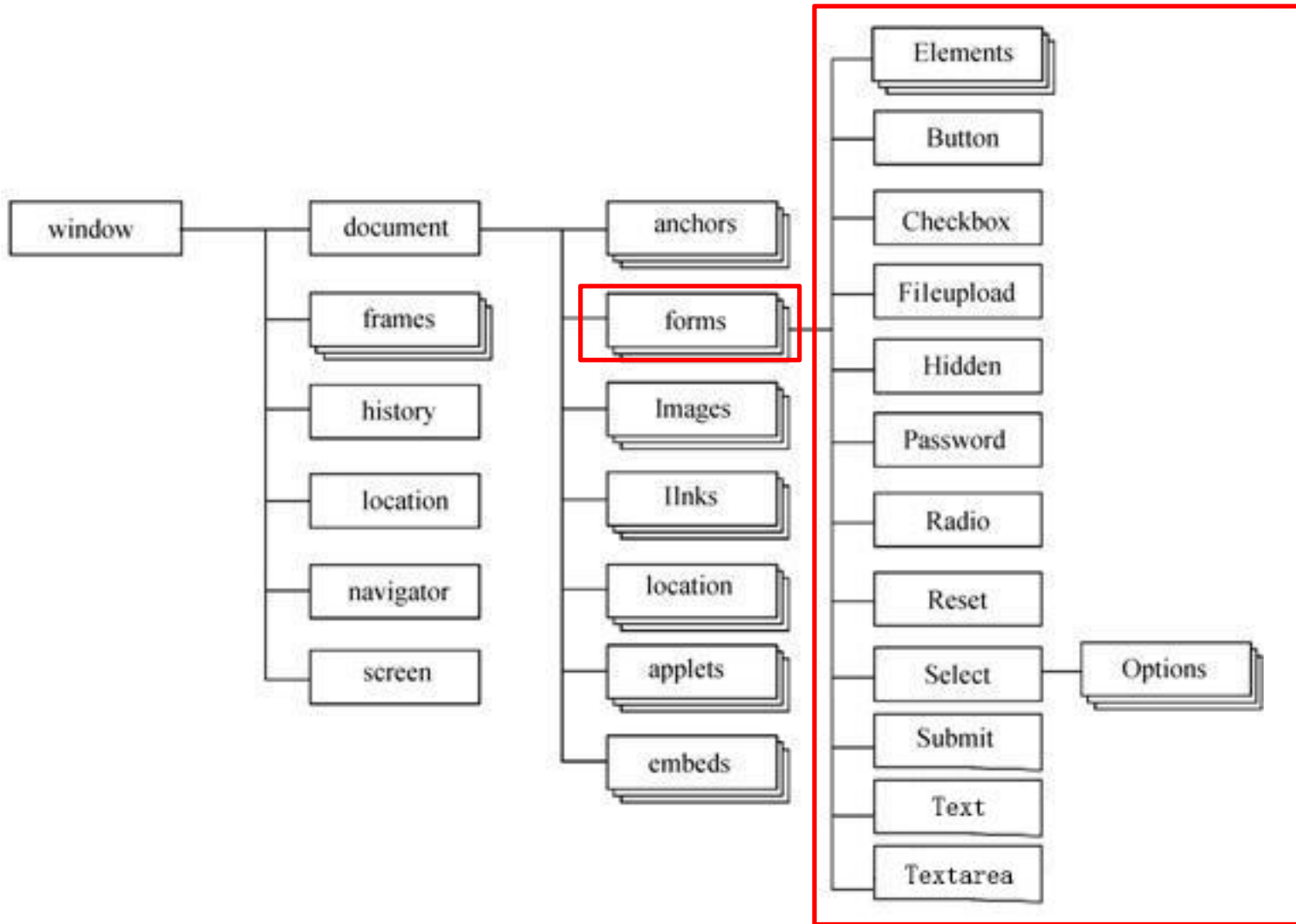
所在地 中国 ▼ 山东 ▼ 淄博 ▼

验证码

☐ 我已阅读并同意相关服务条款和隐私政策



# window对象模型



# 表单操作

---

- 有两类访问方法

- 使用DOM节点访问方法

- 在表单元素中定义id，操作方法同DOM节点

- 使用document.Forms对象

- 不同于节点访问方法
    - 主要通过name属性进行访问
    - form中表单元素的name属性是向后端服务提交数据的关键属性

# form的访问

---

- 通过name属性访问

- var frm1 = document.forms.frm1;
- var frm1 = document.frm1;

- document.forms数字索引访问

- var frm1 = document.forms[0];
- var frm2 = document.forms[1];

# 表单访问

## • 复选框

- 为配合后端复选框name属性值后缀经常增加[]

```
<input type="checkbox" name="fruit[]" value="苹果" checked>苹果  
<input type="checkbox" name="fruit[]" value="香蕉">香蕉  
<input type="checkbox" name="fruit[]" value="桔子">桔子  
<input type="checkbox" name="fruit[]" value="梨">梨
```

- 获取用户选择值需遍历复选框所有元素节点，检查checked属性
- 遍历可以使用getElementsByTagName方法和form.elements属性

# 表单访问

- `getElementsByName`方法

```
var elems = document.getElementsByName("fruit[]");  
for (var elem in elems) {  
    if (elems[elem].checked) {  
        console.log(elems[elem].value);  
    }  
}
```



# 表单访问

- form.elements属性

```
for (var elem in frm.elements) {  
    if (frm.elements[elem].name == "fruit[]") {  
        if (frm.elements[elem].checked) {  
            console.log(frm.elements[elem].value);  
        }  
    }  
}
```

遍历过程中需要判断元素节点name属性值



# 表单访问

- 多选下拉框

- 为配合后端多选下拉框name要加[ ]

```
<select name="city[]" multiple>  
    <option value="北京">北京</option>  
    <option value="上海">上海</option>  
    <option value="广州">广州</option>  
    <option value="深圳">深圳</option>  
</select>
```

- 获取用户选择值需要遍历子元素<option>，检查selected属性
- 可用getElementsByName、getElementsById、form.elements

# 表单访问

- getElementsByName方法

```
var multiSelect = document.getElementsByName("city[]")[0];  
for (var elem in multiSelect.children) {  
    if (multiSelect.children[elem].selected) {  
        console.log(multiSelect.children[elem].value);  
    }  
}
```

使用getElementById需要设置id属性

## •form.elements属性

```
for (var elem in frm.elements) {  
    if(frm.elements[elem].name == "city[]") {  
        for (var op in frm.elements[elem].options) {  
            if(frm.elements[elem].options[op].selected) {  
                console.log(frm.elements[elem].options[op].value);  
            }  
        }  
    }  
}
```



# 表单访问

## •其他表单

```
<input type="text" name="username" value="there is username">
```



`document.frml.username.value`

```
<input type="radio" name="gender" value="男" checked>男
```

```
<input type="radio" name="gender" value="女">女
```



`document.frml.gender.value`

```
<select name="city">
```

```
  <option value="北京">北京</option>
```

```
  <option value="上海" selected>上海</option>
```

```
  <option value="广州">广州</option>
```

```
  <option value="深圳">深圳</option>
```

```
</select>
```



`document.frml.city.value`

```
<textarea name="message"></textarea>
```



`document.frml.message.value`

```
<input type="button" name="btn" value="显示数据" >
```



`document.frml.btn.value`



# 表单操作

- 对表单元素的操作处理

- 对其value值的检测判断与提示，伴随着CSS的样式变化
- 下拉选项的变化，比如下拉选择省，市的下拉列表要跟随变化
  - 可以使用下拉框options属性，有也可以使用appendChild等DOM操作方法

```
document.frm1.city.options.length = 0;
```

```
var option = document.createElement("option");
```

```
document.frm1.city.options[i] = option;
```

```
document.frm1.city.appendChild(option);
```

# 本节小结

---

- 表单元素的访问
  - 复选框和多选下拉框的遍历
  - 普通表单元素的访问
- 表单元素的操作
  - 对下拉框的操作







Thank You !



河北师范大学软件学院  
Software College of Hebei Normal University