表单操作

表单操作

1. 获取表单
2. 表单操作
3. 表单验证
4. 表单提交

获取表单元素

获取表单元素的方式，大体可以分为以下几种方式：

1. Document对象提供的定位页面元素的一系列方法

```
<form id="myform" name="myform" class="login" action="#"></form>
<script>
        var formId = document.getElementById('myform’);
        var formName = document.getElementsByName('myform’);
        var formElement = document.getElementsByTagName('form')[0];
        var formClass = document.getElementsByClassName('login')[0];

        var formId2 = document.querySelector('#myform’);
        var formElement2 = document.querySelectorAll('form')[0];
</script>
```

获取表单元素

2. Document对象提供了forms属性

该属性用于获取当前 HTML 页面中所有表单的集合，返回HTMLCollection对象，该对象封装了当前HTML页面中的所有表单对象。

```
<form id="myform" name="myform" class="login" action="#"></form>
<script>
        var forms = document.forms;
        console.log(forms);
</script>
```

获取表单元素

3. Document对象通过表单的name属性值获取指定表单元素

较早的浏览器还会将定义了name属性的表单保存在document对象中。

```
<form id="myform" name="myform" class="login" action="#"></form>
<script>
        var form = document.myform;
        console.log(form);
</script>
```

注意：这种方式并不推荐，因为在新版本的浏览器中可能不再支持。

获取表单组件元素

获取表单组件元素的方式，大体可以分为以下几种方式：

1. Document对象提供的定位页面元素的一系列方法

```
<form id="myform" name="myform" class="login" action="#">
        用户名: <input type="text" id="username" name="username"><br>
        密码: <input type="password" id="password" name="password"><br>
        <input type="submit">
</form>
<script>
        var username = document.getElementById('username’);
        var password = document.getElementsByName('password');
</script>
```

获取表单组件元素

2. HTMLFormElement 对象的 elements 属性

该属性用于获取指定表单的所有组件的集合。

```
<form id="myform" name="myform" class="login" action="#">
        用户名: <input type="text" id="username" name="username"><br>
        密码: <input type="password" id="password" name="password"><br>
        <input type="submit">
</form>
<script>
        var myform = document.forms[0];
        var formElements = myform.elements;
        console.log(formElements);
</script>
```

注意：HTMLFormElement对象提供了elements属性获取的表单组件中，不包含type为image的input元素。

表单操作

1. 获取表单
2. **表单操作**
3. 表单验证
4. 表单提交

文本框的操作

<input>元素对应DOM中的对象是HTMLInputElement对象，而<textarea>元素对应DOM中的对象是HTMLTextAreaElement对象。

HTMLInputElement对象和HTMLTextAreaElement对象的共同父级对象是HTMLElement对象。所以，这两个对象在很多操作上是比较相似的。

文本内容的选择

HTMLInputElement对象和HTMLTextAreaElement对象都提供了select()方法，该方法用于选择当前文本框的所有文本内容。

```
<form id="myform" action="#">
	<input type="text" id="username" name="username" value="请输入你的用户名">
</form>
<script>
	var username = document.getElementById('username');
	username.select();
</script>
```

select事件

select()方法对应着select事件。也就是说，当调用select()方法时，会触发select事件。

```
<form id="myform" action="#">
	<input type="text" id="username" name="username" value="请输入你的用户名">
</form>
<script>
	var username = document.getElementById('username’);
	username.addEventListener('select',function(){
		console.log("this input element's select()");
	});
</script>
```

获取选择的文本内容

select 事件只是让我们知道用户在什么时候选择了指定文本框的文本内容，但并不能让我们知道用户选择了什么文本内容。

HTML5新版本中通过新增两个属性来解决用户选择了什么文本内容的问题。

属性名	描述
selectionStart	选择文本内容的开始索引值
selectionEnd	选择文本内容的结束索引值

注意：IE 8及之前版本的浏览器并不支持上述两个属性，而是提供了document.selection对象，用于保存整个HTML页面文档范围内选择的文本内容。

设置部分的文本内容

HTML5新版本中提供了setSelectionRange()方法，该方法用于设置一个获取焦点的文本框中选择指定的文本内容。

```
inputElement.setSelectionRange(selectionStart, selectionEnd, [optional]
selectionDirection);
```

在上述语法格式中的参数，具体说明如下：

属性名	描述
selectionStart	被选中的第一个字符的位置
selectionEnd	被选中的最后一个字符的下一个位置
selectionDirection	一个指明选择方向的字符串。有"forward"、"backward"和"none"3个可选值，分别表示"从前往后","从后往前"和"选择方向未知或不重要"

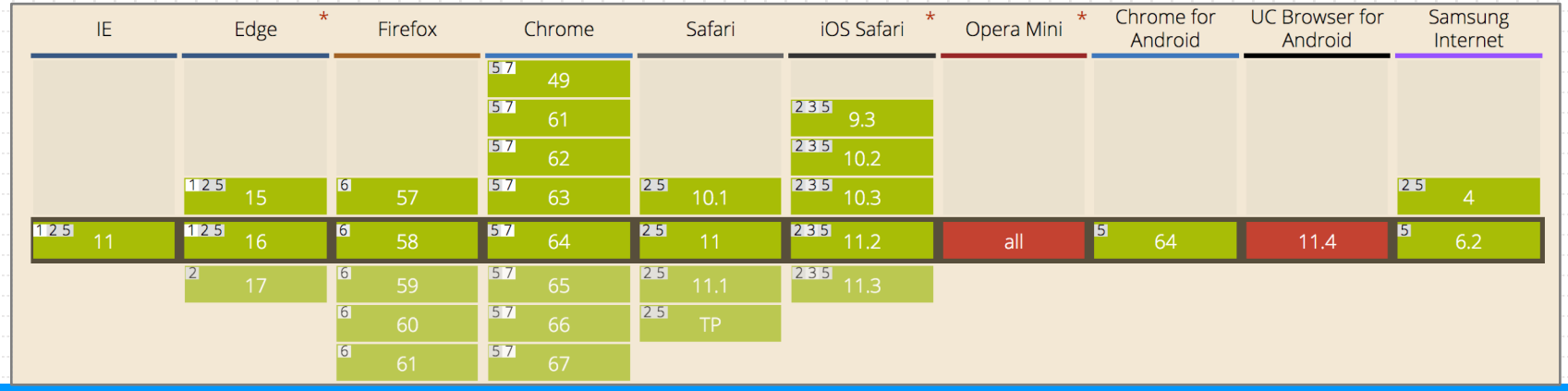
注意：IE 8及之前版本的浏览器并不支持这个方法。

操作剪切板

剪切板功能是经常被忽略，却很重要的功能，可以增强用户体验，方便用户交互。以下三个事件是剪切板的主要操作：

事件名称	描述
copy	在发生复制操作时触发，对应的快捷键为 Ctrl/Cmd + C
cut	在发生剪切操作时触发，对应的快捷键为 Ctrl/Cmd + X
paste	在发生粘贴操作时触发，对应的快捷键为 Ctrl/Cmd + V

于没有针对剪贴板操作的标准，这些事件及相关对象会因浏览器而异。



操作剪切板

事件对象的clipboardData属性存储了由用户触发剪切板事件时所影响的带有MIME类型的数据。

注意：IE 8及之前版本的浏览器，clipboardData属性需要通过window对象获取。

```
var clipboardData = event.clipboardData || window.clipboardData;
```

该属性得到的是一个DataTransfer对象，该对象提供了操作数据的常用方法，如下表所示：

方法名	描述
setData(in String type, in String data)	为一个给定的类型设置数据
getData(in String type)	根据指定的类型检索数据
clearData([in String type])	删除与给定类型关联的数据

下拉列表的操作

下拉列表是由<select>和<option>元素创建的。<select>元素在DOM中对应的是HTMLSelectElement对象，<option>元素在DOM中对应的是HTMLOptionElement对象，这两个对象都提供了一些属性和方法，方便操作下拉列表。

HTMLSelectElement对象

HTMLSelectElement对象是<select>元素在DOM中对应的对象。

- 该对象提供的属性如下表所示：

属性名	描述
length	表示当前<select>元素中<option>元素的个数
multiple	表示<select>元素是否允许多项选择，等价于HTML中的multiple属性
options	当前<select>元素中<option>元素对象的集合
selectedIndex	代表第一个被选中的<option>元素。-1 代表没有元素被选中
size	当前<select>元素中可见的行数，等价于HTML中的size属性

- 该对象提供的方法如下表所示：

方法名	描述
add(item[, before])	将<option>元素添加到当前<select>元素的选项元素集合中
item(idx)	放回索引值为idx的<option>元素。如果没有，则返回null
remove(index)	从当前<select>元素的选项元素集合中删除指定索引值的<option>元素

HTMLOptionElement对象

HTMLOptionElement对象是<option>元素在DOM中对应的对象。

- 该对象提供的属性如下表所示：

属性名	描述
index	当前<option>元素在其所属的选项列表中的索引值
selected	表示当前<option>元素是否被选中
text	当前<option>元素的文本内容
value	当前<option>元素的 value 属性值

表单操作

1. 获取表单
2. 表单操作
3. **表单验证**
4. 表单提交

HTML5验证API

HTML5提供了一组用于表单验证的API，目前主流浏览器对HTML5提供的验证API支持越来越好。

- 验证API的属性

属性名	描述
validity	一个ValidityState对象，描述元素的验证状态
validity.customError	如果元素设置了自定义错误，返回true；否则返回false
validity.patternMismatch	如果元素的值不匹配所设置的正则表达式，返回true，否则返回false
validity.rangeOverflow	如果元素的值高于所设置的最大值，返回true，否则返回false
validity.rangeUnderflow	如果元素的值低于所设置的最小值，返回true，否则返回false
validity.stepMismatch	如果元素的值不符合 step 属性的规则，返回true，否则返回false
validity.tooLong	如果元素的值超过所设置的最大长度，返回true，否则返回false
validity.typeMismatch	如果元素的值出现语法错误，返回true，否则返回false
validity.valueMissing	如果元素设置了 required 属性且值为空，返回true，否则返回false
validity.valid	如果元素的值不存在验证问题，返回true，否则返回false

HTML5验证API

- 验证API的方法

方法名	描述
checkValidity()	如果元素的值不存在验证问题，返回true，否则返回false
setCustomValidity(message)	为元素添加一个自定义的错误消息；如果设置了自定义错误消息，则该元素被认为是无效的，并显示指定的错误

表单操作

1. 获取表单
2. 表单操作
3. 表单验证
4. **表单提交**

submit事件

提交表单时，会触发submit事件。

```
<form action="" method="post">
	<input type="text" name="data">
	<input type="submit" value="提交">
</form>
<script>
	var myform = document.forms[0];
	myform.addEventListener('submit',function(){
		alert('表单被提交.....’);
	});
</script>
```

submit()方法

表单还提供了submit()方法用于提交表单，使用该方法时允许表单内使用任一普通按钮即可（并非提交按钮）。

```
<form action="" method="post">
	<input type="text" name="data">
	<input id="btn" type="button" value="提交">
</form>
<script>
	var btn = document.getElementById('btn’);
	btn.addEventListener('click',function(){
		var myform = document.forms[1];
		myform.submit();
	});
</script>
```

注意：使用submit()方法提交表单不会触发submit事件。