# 表单事件

目录


- 表单事件的种类
  - input 事件
  - select 事件
  - change 事件
  - invalid 事件
  - reset 事件，submit 事件
- InputEvent 接口


# 表单事件的种类

# input 事件

`input`事件当`<input>`、`<select>`、`<textarea>`的值发生变化时触发。对于复选框（`<input type=checkbox>`）或单选框（`<input type=radio>`），用户改变选项时，也会触发这个事件。另外，对于打开`contenteditable`属性的元素，只要值发生变化，也会触发`input`事件。

`input`事件的一个特点，就是会连续触发，比如用户每按下一次按键，就会触发一次`input`事件。

`input`事件对象继承了`InputEvent`接口。

该事件跟`change`事件很像，不同之处在于`input`事件在元素的值发生变化后立即发生，而`change`在元素失去焦点时发生，而内容此时可能已经变化多次。也就是说，如果有连续变化，`input`事件会触发多次，而`change`事件只在失去焦点时触发一次。

下面是`<select>`元素的例子。

```
/* HTML 代码如下
<select id="mySelect">
  <option value="1">1</option>
  <option value="2">2</option>
```

```
  <option value="3">3</option>
</select>
*/

function inputHandler(e) {
  console.log(e.target.value)
}

var mySelect = document.querySelector('#mySelect');
mySelect.addEventListener('input', inputHandler);
```

上面代码中，改变下拉框选项时，会触发 `input` 事件，从而执行回调函数 `inputHandler`。

# select 事件

`select` 事件当在 `<input>`、`<textarea>` 里面选中文本时触发。

```
// HTML 代码如下
// <input id="test" type="text" value="Select me!" />

var elem = document.getElementById('test');
elem.addEventListener('select', function (e) {
  console.log(e.type); // "select"
}, false);
```

选中的文本可以通过 `event.target` 元素的 `selectionDirection`、`selectionEnd`、`selectionStart` 和 `value` 属性拿到。

# change 事件

`change` 事件当 `<input>`、`<select>`、`<textarea>` 的值发生变化时触发。它与 `input` 事件的最大不同，就是不会连续触发，只有当全部修改完成时才会触发，另一方面 `input` 事件必然伴随 `change` 事件。具体来说，分成以下几种情况。

- 激活单选框（radio）或复选框（checkbox）时触发。
- 用户提交时触发。比如，从下列列表（select）完成选择，在日期或文件输入框完成选择。
- 当文本框或 `<textarea>` 元素的值发生改变，并且丧失焦点时触发。

下面是一个例子。

```
// HTML 代码如下
// <select size="1" onchange="changeEventHandler(event);">
//   <option>chocolate</option>
//   <option>strawberry</option>
//   <option>vanilla</option>
// </select>

function changeEventHandler(event) {
  console.log(event.target.value);
}
```

如果比较一下上面 `input` 事件的例子，你会发现对于 `<select>` 元素来说，`input` 和 `change` 事件基本是等价的。

# invalid 事件

用户提交表单时，如果表单元素的值不满足校验条件，就会触发 `invalid` 事件。

```
<form>
  <input type="text" required oninvalid="console.log('invalid input')" />
  <button type="submit">提交</button>
</form>
```

上面代码中，输入框是必填的。如果不填，用户点击按钮提交时，就会触发输入框的 `invalid` 事件，导致提交被取消。

# reset 事件，submit 事件

这两个事件发生在表单对象 `<form>` 上，而不是发生在表单的成员上。

`reset` 事件当表单重置（所有表单成员变回默认值）时触发。

`submit` 事件当表单数据向服务器提交时触发。注意，`submit` 事件的发生对象是 `<form>` 元素，而不是 `<button>` 元素，因为提交的是表单，而不是按钮。

# InputEvent 接口

`InputEvent` 接口主要用来描述 `input` 事件的实例。该接口继承了 `Event` 接口，还定义了一些自己的实例属性和实例方法。

浏览器原生提供 `InputEvent()` 构造函数，用来生成实例对象。

```
new InputEvent(type, options)
```

`InputEvent` 构造函数可以接受两个参数。第一个参数是字符串，表示事件名称，该参数是必需的。第二个参数是一个配置对象，用来设置事件实例的属性，该参数是可选的。配置对象的字段除了 `Event` 构造函数的配置属性，还可以设置下面的字段，这些字段都是可选的。

- `inputType`：字符串，表示发生变更的类型（详见下文）。
- `data`：字符串，表示插入的字符串。如果没有插入的字符串（比如删除操作），则返回 `null` 或空字符串。
- `dataTransfer`：返回一个 DataTransfer 对象实例，该属性通常只在输入框接受富文本输入时有效。

`InputEvent` 的实例属性主要就是上面三个属性，这三个实例属性都是只读的。

### （1）InputEvent.data

`InputEvent.data` 属性返回一个字符串，表示变动的内容。

```
// HTML 代码如下
// <input type="text" id="myInput">
var input = document.getElementById('myInput');
input.addEventListener('input', myFunction, false);

function myFunction(e) {
  console.log(e.data);
}
```

上面代码中，如果手动在输入框里面输入 `abc`，控制台会先输出 `a`，再在下一行输出 `b`，再在下一行输出 `c`。然后选中 `abc`，一次性将它们删除，控制台会输出 `null` 或一个空字符串。

### （2）InputEvent.inputType

`InputEvent.inputType` 属性返回一个字符串，表示字符串发生变更的类型。

对于常见情况，Chrome 浏览器的返回值如下。完整列表可以参考文档。

- 手动插入文本：`insertText`
- 粘贴插入文本：`insertFromPaste`
- 向后删除：`deleteContentBackward`
- 向前删除：`deleteContentForward`

### （3）InputEvent.dataTransfer

InputEvent.dataTransfer属性返回一个 DataTransfer 实例。该属性只在文本框接受粘贴内容（ insertFromPaste）或拖拽内容（ insertFromDrop）时才有效。

版本号 #2
由 壳壳虫 创建于 2018-05-09 14:15:54 UTC
由 壳壳虫 更新于 2019-01-03 17:58:43 UTC