GlobalEventHandlers 接口

指定事件的回调函数，推荐使用的方法是元素的`addEventListener`方法。

```javascript
div.addEventListener('click', clickHandler, false);
```

除了之外，还有一种方法可以直接指定事件的回调函数。

```javascript
div.onclick = clickHandler;
```

这个接口是由`GlobalEventHandlers`接口提供的。它的优点是使用比较方便，缺点是只能为每个事件指定一个回调函数，并且无法指定事件触发的阶段（捕获阶段还是冒泡阶段）。

`HTMLElement`、`Document`和`Window`都继承了这个接口，也就是说，各种 HTML 元素、`document`对象、`window`对象上面都可以使用`GlobalEventHandlers`接口提供的属性。下面就列出这个接口提供的主要的事件属性。

GlobalEventHandlers.onabort

某个对象的`abort`事件（停止加载）发生时，就会调用`onabort`属性指定的回调函数。

各种元素的停止加载事件，到底如何触发，目前并没有统一的规定。因此实际上，这个属性现在一般只用在`<img>`元素上面。

```javascript
// HTML 代码如下
// <img src="example.jpg" id="img">
var img = document.getElementById('img');
img.onabort = function () {
  console.log('image load aborted.');
}
```

GlobalEventHandlers.onerror

`error`事件发生时，就会调用`onerror`属性指定的回调函数。

`error`事件分成两种。

一种是 JavaScript 的运行时错误，这会传到`window`对象，导致`window.onerror()`。

```javascript
window.onerror = function (message, source, lineno, colno, error) {
  // ...
```

}
```

`window.onerror`的处理函数共接受五个参数，含义如下。

- message：错误信息字符串
- source：报错脚本的 URL
- lineno：报错的行号，是一个整数
- colno：报错的列号，是一个整数
- error： 错误对象

另一种是资源加载错误，比如`<img>`或`<script>`加载的资源出现加载错误。这时，Error 对象会传到对应的元素，导致该元素的`onerror`属性开始执行。

```javascript
element.onerror = function (event) {
  // ...
}
```

注意，一般来说，资源的加载错误不会触发`window.onerror`。

GlobalEventHandlers.onload、GlobalEventHandlers.onloadstart

元素完成加载时，会触发`load`事件，执行`onload()`。它的典型使用场景是`window`对象和`<img>`元素。对于`window`对象来说，只有页面的所有资源加载完成（包括图片、脚本、样式表、字体等所有外部资源），才会触发`load`事件。

对于`<img>`和`<video>`等元素，加载开始时还会触发`loadstart`事件，导致执行`onloadstart`。

GlobalEventHandlers.onfocus，GlobalEventHandlers.onblur

当前元素获得焦点时，会触发`element.onfocus`；失去焦点时，会触发`element.onblur`。

```javascript
element.onfocus = function () {
  console.log("onfocus event detected!");
};
element.onblur = function () {
  console.log("onblur event detected!");
};
```

注意，如果不是可以接受用户输入的元素，要触发`onfocus`，该元素必须有`tabindex`属性。

GlobalEventHandlers.onscroll
```

页面或元素滚动时，会触发`scroll`事件，导致执行`onscroll()`。

GlobalEventHandlers.oncontextmenu，GlobalEventHandlers.onshow

用户在页面上按下鼠标的右键，会触发`contextmenu`事件，导致执行`oncontextmenu()`。如果该属性执行后返回`false`，就等于禁止了右键菜单。`document.oncontextmenu`与`window.oncontextmenu`效果一样。

```javascript
document.oncontextmenu = function () {
  return false;
};
```

上面代码中，`oncontextmenu`属性执行后返回`false`，右键菜单就不会出现。

元素的右键菜单显示时，会触发该元素的`onshow`监听函数。

其他的事件属性

鼠标的事件属性。

- onclick
- ondblclick
- onmousedown
- onmouseenter
- onmouseleave
- onmousemove
- onmouseout
- onmouseover
- onmouseup
- onwheel

键盘的事件属性。

- onkeydown
- onkeypress
- onkeyup

焦点的事件属性。

- onblur
- onfocus

表单的事件属性。

- oninput
- onchange
- onsubmit

- onreset
- oninvalid
- onselect

触摸的事件属性。

- ontouchcancel
- ontouchend
- ontouchmove
- ontouchstart

拖动的事件属性分成两类：一类与被拖动元素相关，另一类与接收被拖动元素的容器元素相关。

被拖动元素的事件属性。

- ondragstart：拖动开始
- ondrag：拖动过程中，每隔几百毫秒触发一次
- ondragend：拖动结束

接收被拖动元素的容器元素的事件属性。

- ondragenter：被拖动元素进入容器元素。
- ondragleave：被拖动元素离开容器元素。
- ondragover：被拖动元素在容器元素上方，每隔几百毫秒触发一次。
- ondrop：松开鼠标后，被拖动元素放入容器元素。

`<dialog>`对话框元素的事件属性。

- oncancel
- onclose