

## [MS-DOM2E]:

# Internet Explorer Document Object Model (DOM) Level 2 Events Standards Support Document

---

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](#) or the [Microsoft Community Promise](#). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](#).
- **Trademarks.** The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit [www.microsoft.com/trademarks](http://www.microsoft.com/trademarks).
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights.** All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

**Tools.** The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

**Support.** For questions and support, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).

## Revision Summary

Date	Revision History	Revision Class	Comments
3/17/2010	0.1	New	Released new document.
3/26/2010	1.0	None	Introduced no new technical or language changes.
5/26/2010	1.2	None	Introduced no new technical or language changes.
9/8/2010	1.3	Major	Significantly changed the technical content.
2/10/2011	2.0	Minor	Clarified the meaning of the technical content.
2/22/2012	3.0	Major	Significantly changed the technical content.
7/25/2012	3.1	Minor	Clarified the meaning of the technical content.
6/26/2013	4.0	Major	Significantly changed the technical content.
3/31/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
1/22/2015	5.0	Major	Updated for new product version.
7/7/2015	5.1	Minor	Clarified the meaning of the technical content.
11/2/2015	5.1	None	No changes to the meaning, language, or formatting of the technical content.
3/22/2016	5.2	Minor	Clarified the meaning of the technical content.
7/19/2016	5.3	Minor	Clarified the meaning of the technical content.
11/2/2016	5.3	None	No changes to the meaning, language, or formatting of the technical content.
3/14/2017	5.3	None	No changes to the meaning, language, or formatting of the technical content.
10/3/2017	5.3	None	No changes to the meaning, language, or formatting of the technical content.
2/22/2018	5.3	None	No changes to the meaning, language, or formatting of the technical content.
3/23/2018	5.3	None	No changes to the meaning, language, or formatting of the technical content.
8/28/2018	5.3	None	No changes to the meaning, language, or formatting of the technical content.

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>4</b>
1.1	Glossary .....	4
1.2	References .....	4
1.2.1	Normative References .....	4
1.2.2	Informative References .....	4
1.3	Microsoft Implementations .....	4
1.4	Standards Support Requirements .....	5
1.5	Notation.....	6
<b>2</b>	<b>Standards Support Statements.....</b>	<b>7</b>
2.1	Normative Variations .....	7
2.1.1	[DOM Level 2 - Events] Section 1.2.1, Basic event flow .....	7
2.1.2	[DOM Level 2 - Events] Section 1.2.2, Event capture .....	7
2.1.3	[DOM Level 2 - Events] Section 1.2.3, Event bubbling .....	8
2.1.4	[DOM Level 2 - Events] Section 1.3.1, Event registration interfaces .....	9
2.1.5	[DOM Level 2 - Events] Section 1.3.2, Interaction with HTML 4.0 event listeners .....	12
2.1.6	[DOM Level 2 - Events] Section 1.4, Event interface .....	13
2.1.7	[DOM Level 2 - Events] Section 1.5, DocumentEvent interface .....	16
2.1.8	[DOM Level 2 - Events] Section 1.6.1, User Interface event types .....	17
2.1.9	[DOM Level 2 - Events] Section 1.6.2, Mouse event types .....	18
2.1.10	[DOM Level 2 - Events] Section 1.6.4, Mutation event types .....	21
2.1.11	[DOM Level 2 - Events] Section 1.6.5, HTML event types.....	23
2.2	Clarifications .....	27
2.3	Error Handling .....	27
2.4	Security .....	27
<b>3</b>	<b>Change Tracking.....</b>	<b>28</b>
<b>4</b>	<b>Index.....</b>	<b>29</b>

# 1 Introduction

This document describes the level of support provided by Microsoft web browsers for the *Document Object Model (DOM) Level 2 Events Specification Version 1.0* [[DOM Level 2 - Events](#)], published 13 November 2000.

The [DOM Level 2 - Events] specification contains guidance for authors of webpages and browser users, in addition to user agents (browser applications). Statements found in this document apply only to normative requirements in the specification targeted to user agents, not those targeted to authors.

## 1.1 Glossary

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as defined in [[RFC2119](#)]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## 1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](#).

### 1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information.

[DOM Level 2 - Events] World Wide Web Consortium, "Document Object Model (DOM) Level 2 Events Specification Version 1.0", W3C Recommendation 13 November 2000, <http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/>

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/rfc/rfc2119.txt>

### 1.2.2 Informative References

None.

## 1.3 Microsoft Implementations

The following Microsoft web browser versions implement some portion of the [[DOM Level 2 - Events](#)] specification:

- Windows Internet Explorer 7
- Windows Internet Explorer 8
- Windows Internet Explorer 9
- Windows Internet Explorer 10
- Internet Explorer 11
- Internet Explorer 11 for Windows 10
- Microsoft Edge

Each browser version may implement multiple document rendering modes. The modes vary from one to another in support of the standard. The following table lists the document modes supported by each browser version.

Browser Version	Document Modes Supported
Internet Explorer 7	Quirks Mode Standards Mode
Internet Explorer 8	Quirks Mode IE7 Mode IE8 Mode
Internet Explorer 9	Quirks Mode IE7 Mode IE8 Mode IE9 Mode
Internet Explorer 10	Quirks Mode IE7 Mode IE8 Mode IE9 Mode IE10 Mode
Internet Explorer 11	Quirks Mode IE7 Mode IE8 Mode IE9 Mode IE10 Mode IE11 Mode
Internet Explorer 11 for Windows 10	Quirks Mode IE7 Mode IE8 Mode IE9 Mode IE10 Mode IE11 Mode
Microsoft Edge	EdgeHTML Mode

For each variation presented in this document there is a list of the document modes and browser versions that exhibit the behavior described by the variation. All combinations of modes and versions that are not listed conform to the specification. For example, the following list for a variation indicates that the variation exists in three document modes in all browser versions that support these modes:

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

**Note:** "Standards Mode" in Internet Explorer 7 and "IE7 Mode" in Internet Explorer 8 refer to the same document mode. "IE7 Mode" is the preferred way of referring to this document mode across all versions of the browser.

## 1.4 Standards Support Requirements

To conform to [\[DOM Level 2 - Events\]](#) a user agent must implement all required portions of the specification. Any optional portions that have been implemented must also be implemented as

described by the specification. Normative language is usually used to define both required and optional portions. (For more information, see [\[RFC2119\]](#).)

The following table lists the sections of [DOM Level 2 - Events] and whether they are considered normative or informative.

Sections	Normative/Informative
1	Normative
Appendix A-D	Informative

## 1.5 Notation

The following notations are used in this document to differentiate between notes of clarification, variation from the specification, and extension points.

Notation	Explanation
C####	This identifies a clarification of ambiguity in the target specification. This includes imprecise statements, omitted information, discrepancies, and errata. This does not include data formatting clarifications.
V####	This identifies an intended point of variability in the target specification such as the use of MAY, SHOULD, or RECOMMENDED. (See <a href="#">[RFC2119]</a> .) This does not include extensibility points.
E####	Because the use of extensibility points (such as optional implementation-specific data) can impair interoperability, this profile identifies such points in the target specification.

For document mode and browser version notation, see also section [1.3](#).

## 2 Standards Support Statements

This section contains all variations and clarifications for the Microsoft implementation of [\[DOM Level 2 - Events\]](#).

- Section [2.1](#) describes normative variations from the MUST requirements of the specification.
- Section [2.2](#) describes clarifications of the MAY and SHOULD requirements.
- Section [2.3](#) considers error handling aspects of the implementation.
- Section [2.4](#) considers security aspects of the implementation.

### 2.1 Normative Variations

The following subsections describe normative variations from the MUST requirements of [\[DOM Level 2 - Events\]](#).

#### 2.1.1 [DOM Level 2 - Events] Section 1.2.1, Basic event flow

V0001:

The specification states:

```
Each event has an EventTarget toward which the event is directed by the DOM implementation.
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

Each event has a **srcElement** property, rather than an **EventTarget**, that represents the object that fired the event.

V0002:

The specification states:

```
It is expected that actions taken by EventListeners may cause additional events to fire. Additional events should be handled in a synchronous manner and may cause reentrancy into the event model.
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

Additional events are handled asynchronously and do not cause reentrancy into the event model.

#### 2.1.2 [DOM Level 2 - Events] Section 1.2.2, Event capture

V0003:

The specification states:

```
An EventListener being registered on an EventTarget may choose to have that EventListener capture events by specifying the useCapture parameter of the addEventListener method to be true.
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **addEventListener** method is not supported. The **attachEvent** method is similar but does not include a **useCapture** parameter.

V0004:

The specification states:

A capturing EventListener will not be triggered by events dispatched directly to the EventTarget upon which it is registered.

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The capture phase of the event flow is not supported.

*IE9 Mode, IE10 Mode, IE11 Mode, and EdgeHTML Mode (All Versions)*

A capturing **EventListener** object is triggered by events that are dispatched directly to the **EventTarget** object upon which it is registered. Capturing event listeners are triggered first, followed by bubbling event listeners.

V0005:

The specification states:

If the capturing EventListener wishes to prevent further processing of the event from occurring it may call the stopPropagation method of the Event interface. This will prevent further dispatch of the event, although additional EventListeners registered at the same hierarchy level will still receive the event. Once an event's stopPropagation method has been called, further calls to that method have no additional effect. If no additional capturers exist and stopPropagation has not been called, the event triggers the appropriate EventListeners on the target itself.

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The following variations apply:

- The **stopPropagation** method is not supported. Similar functionality is provided by the **cancelBubble** property.
- The capture phase of the event flow is not supported.

### **2.1.3 [DOM Level 2 - Events] Section 1.2.3, Event bubbling**

V0006:

The specification states:

EventListeners registered as capturers will not be triggered during this phase.

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The capture phase of the event flow is not supported. A noncapture phase of the event flow model is supported instead.

V0007:

The specification states:

Any event handler may choose to prevent further event propagation by calling the `stopPropagation` method of the Event interface.

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **stopPropagation** method is not supported. Similar functionality is provided by the **cancelBubble** property.

V0008:

The specification states:

Only one call to `stopPropagation` is required to prevent further bubbling.

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The following variations apply:

- The **stopPropagation** method is not supported. Similar functionality is provided by the **cancelBubble** property.
- The **cancelBubble** property can be set multiple times before the event handler terminates. In this case, the final value of **cancelBubble** determines its effective value.

### **2.1.4 [DOM Level 2 - Events] Section 1.3.1, Event registration interfaces**

V0009:

The specification states:

The `EventTarget` interface is implemented by all Nodes in an implementation which supports the DOM Event Model.

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **EventTarget** interface is not supported. Similar functionality is provided by the **attachEvent**, **detachEvent**, and **fireEvent** scripting methods.

V0010:

The specification states:

`addEventListener`

This method allows the registration of event listeners on the event target. If an `EventListener` is added to an `EventTarget` while it is processing an event, it will not be triggered by the current actions but may be triggered during a later stage of event flow, such as the bubbling phase. If multiple identical `EventListeners` are registered on the same `EventTarget` with the same parameters the duplicate instances are discarded. They do not cause the `EventListener` to be called twice and since they are discarded they do not need to be removed with the `removeEventListener` method.

Parameters

type of type `DOMString`  
The event type for which the user is registering

listener of type `EventListener`  
The listener parameter takes an interface implemented by the user which contains the methods to be called when the event occurs. `useCapture` of type `boolean` If `true`, `useCapture` indicates that the user wishes to initiate capture. After initiating capture, all events of the specified type will be dispatched to the registered `EventListener` before being dispatched to any `EventTargets` beneath them in the tree. Events which are bubbling upward through the tree will not trigger an `EventListener` designated to use capture.

No Return Value

No Exceptions

### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **`addEventListener`** method is not supported. Similar functionality is provided by the **`attachEvent(sEvent, fpNotify)`** scripting method.

Parameter:

**`sEvent`** Required. A string that specifies any of the standard DHTML events.

**`fpNotify`** Required. A pointer that specifies the function to call when `sEvent` fires.

Return Value:

Boolean.

`true` = The function is bound successfully to the event.

`false` = The function is not bound to the event.

### *IE9 Mode, IE10 Mode, IE11 Mode, and EdgeHTML Mode (All Versions)*

Duplicate **`EventListener`** objects are discarded only if they are identical instances, without regard to the parameter signatures.

V0011:

The specification states:

`dispatchEvent`  
This method allows the dispatch of events into the implementations event model. Events dispatched in this manner will have the same capturing and bubbling behavior as events dispatched directly by the implementation. The target of the event is the `EventTarget` on which `dispatchEvent` is called.

Parameters `evt` of type `Event` Specifies the event type, behavior, and contextual information to be used in processing the event.

Return Value `boolean`  
The return value of `dispatchEvent` indicates whether any of the listeners which handled the event called `preventDefault`. If `preventDefault` was called the value is `false`, else the value is `true`.

Exceptions  
`EventExceptionUNSPECIFIED_EVENT_TYPE_ERR`: Raised if the Event's type was not specified by initializing the event before `dispatchEvent` was called. Specification of the Event's type as `null` or an empty string will also trigger this exception.

### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **dispatchEvent** method is not supported. Similar functionality is provided by the **fireEvent(sEvent [, oEventObject])** scripting method.

Parameter:

**sEvent**

Required. A string that specifies the name of the event to fire.

**oEventObject**

Optional. An object that specifies the event object from which to obtain event object properties.

Return Value:

true = The event fired successfully.

false = The event was canceled.

V0012:

The specification states:

`removeEventListener`

This method allows the removal of event listeners from the event target. If an `EventListener` is removed from an `EventTarget` while it is processing an event, it will not be triggered by the current actions. `EventListeners` can never be invoked after being removed.

Calling `removeEventListener` with arguments which do not identify any currently registered `EventListener` on the `EventTarget` has no effect.

Parameters

`type` of type `DOMString`

Specifies the event type of the `EventListener` being removed.

`listener` of type `EventListener`

The `EventListener` parameter indicates the `EventListener` to be removed.

`useCapture` of type `boolean`

Specifies whether the `EventListener` being removed was registered as a capturing listener or not. If a listener was registered twice, one with capture and one without, each must be removed separately. Removal of a capturing listener does not affect a non-capturing version of the same listener, and vice versa.

No Return Value

No Exceptions

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **removeEventListener** method is not supported. Similar functionality is provided by the **detachEvent(sEvent, fpNotify)** scripting method.

Parameter:

**sEvent**

Required. A string that specifies the name of the event to raise.

**fpNotify**

Required. Pointer that specifies the function previously set using the **attachEvent** method.

Return Value:

No return value.

V0013:

The specification states:

The `EventListener` interface is the primary method for handling events. Users implement the `EventListener` interface and register their listener on an `EventTarget` using the `AddEventListener` method. The users should also remove their `EventListener` from its `EventTarget` after they have completed using the listener. When a Node is copied using the `cloneNode` method the `EventListeners` attached to the source Node are not attached to the copied Node. If the user wishes the same `EventListeners` to be added to the newly created copy the user must add them manually.

```
IDL Definition
// Introduced in DOM Level 2:
interface EventListener {
    void          handleEvent(in Event evt);
};
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **EventListener** method is not supported. Similar functionality is provided by the **attachEvent(sEvent, fpNotify)** and **detachEvent(sEvent, fpNotify)** scripting methods.

When an event is cloned, the source node is automatically attached to the cloned event.

V0014:

The specification states:

```
handleEvent
This method is called whenever an event occurs of the type for which the
EventListener interface was registered.

Parameters
evt of type Event
The Event contains contextual information about the event. It also contains the
stopPropagation and preventDefault methods which are used in determining the
event's flow and default action.

No Return Value

No Exceptions
```

*Quirks Mode, IE7 Mode, IE8 Mode, IE9 Mode, IE10 Mode, and IE11 Mode (All Versions)*

The **handleEvent** method is not supported.

## 2.1.5 [DOM Level 2 - Events] Section 1.3.2, Interaction with HTML 4.0 event listeners

V0015:

The specification states:

In order to achieve compatibility with HTML 4.0, implementors may view the setting of attributes which represent event handlers as the creation and registration of an `EventListener` on the `EventTarget`. The value of `useCapture` defaults to `false`. This

EventListener behaves in the same manner as any other EventListeners which may be registered on the EventTarget. If the attribute representing the event listener is changed, this may be viewed as the removal of the previously registered EventListener and the registration of a new one. No technique is provided to allow HTML 4.0 event listeners access to the context information defined for each event.

#### *Quirks Mode and IE7 Mode (All Versions)*

Setting the attribute name of an event listener with the **setAttribute** method does not result in the creation and registration of an event listener on the source element.

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

When event listeners are created through the setting of attributes, they do not behave in the same manner as other event listeners. These attribute-associated events take precedence and are processed before other events.

### **2.1.6 [DOM Level 2 - Events] Section 1.4, Event interface**

V0016:

The specification states:

The Event interface is used to provide contextual information about an event to the handler processing the event. An object which implements the Event interface is generally passed as the first parameter to an event handler. More specific context information is passed to event handlers by deriving additional interfaces from Event which contain information directly relating to the type of event they accompany. These derived interfaces are also implemented by the object passed to the event listener.

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **Event** interface is not supported.

V0017:

The specification states:

Definition group PhaseType  
An integer indicating which phase of event flow is being processed.

Defined Constants  
AT\_TARGET  
The event is currently being evaluated at the target EventTarget.

BUBBLING\_PHASE  
The current event phase is the bubbling phase.

CAPTURING\_PHASE  
The current event phase is the capturing phase

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **PhaseType** definition group is not supported.

V0018:

The specification states:

#### Attributes

bubbles of type boolean, readonly

Used to indicate whether or not an event is a bubbling event. If the event can bubble the value is true, else the value is false.

cancelable of type boolean, readonly

Used to indicate whether or not an event can have its default action prevented. If the default action can be prevented the value is true, else the value is false.

currentTarget of type EventTarget, readonly

Used to indicate the EventTarget whose EventListeners are currently being processed. This is particularly useful during capturing and bubbling.

eventPhase of type unsigned short, readonly

Used to indicate which phase of event flow is currently being evaluated.

target of type EventTarget, readonly

Used to indicate the EventTarget to which the event was originally dispatched.

timeStamp of type DOMTimeStamp, readonly

Used to specify the time (in milliseconds relative to the epoch) at which the event was created. Due to the fact that some systems may not provide this information the value of timeStamp may be not available for all events. When not available, a value of 0 will be returned. Examples of epoch time are the time of the system start or 0:0:0 UTC 1st January 1970.

type of type DOMString, readonly

The name of the event (case-insensitive). The name must be an XML name.

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The following attributes are not supported:

- **bubbles**
- **cancelable**
- **currentTarget**
- **eventPhase**
- **target**
- **timeStamp**

The following attribute is supported:

- **type**

V0019:

The specification states:

#### initEvent

The initEvent method is used to initialize the value of an Event created through the DocumentEvent interface. This method may only be called before the Event has been dispatched via the dispatchEvent method, though it may be called multiple times during that phase if necessary. If called multiple times the final invocation takes precedence. If called from a subclass of Event interface only the values specified in the initEvent method are modified, all other attributes are left unchanged.

#### Parameters

eventTypeArg of type DOMString

Specifies the event type. This type may be any event type currently defined in this

specification or a new event type.. The string must be an XML name. Any new event type must not begin with any upper, lower, or mixed case version of the string "DOM". This prefix is reserved for future DOM event sets. It is also strongly recommended that third parties adding their own events use their own prefix to avoid confusion and lessen the probability of conflicts with other new events.

canBubbleArg of type boolean  
Specifies whether or not the event can bubble.

cancelableArg of type boolean  
Specifies whether or not the event's default action can be prevented.

No Return Value

No Exceptions

### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **initEvent** method is not supported. Similar functionality is provided by the **createEventObject([oExistingEvent])** scripting method.

Parameter:

**oExistingEvent** Optional. An object that specifies an existing event object on which to base the new object.

Return Value:

Returns an event object.

Event object properties are read-only after the **fireEvent** method is called.

V0020:

The specification states:

preventDefault  
If an event is cancelable, the preventDefault method is used to signify that the event is to be canceled, meaning any default action normally taken by the implementation as a result of the event will not occur. If, during any stage of event flow, the preventDefault method is called the event is canceled. Any default action associated with the event will not occur. Calling this method for a non-cancelable event has no effect. Once preventDefault has been called it will remain in effect throughout the remainder of the event's propagation. This method may be used during any stage of event flow.

No Parameters

No Return Value

No Exceptions

### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **preventDefault** method is not supported. Similar functionality is provided by the **returnValue** property.

V0021:

The specification states:

stopPropagation

The stopPropagation method is used prevent further propagation of an event during event flow. If this method is called by any EventListener the event will cease propagating through the tree. The event will complete dispatch to all listeners on the current EventTarget before event flow stops. This method may be used during any stage of event flow.

No Parameters

No Return Value

No Exceptions

### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **stopPropagation** method is not supported. Similar functionality is provided by the **cancelBubble** property.

V0022:

The specification states:

```
Event operations may throw an EventException as specified in their method descriptions.
IDL Definition
// Introduced in DOM Level 2:
exception EventException {
    unsigned short    code;
};
// EventExceptionCode
const unsigned short    UNSPECIFIED_EVENT_TYPE_ERR    = 0;
```

### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

Exception handling is not supported. Error objects and JavaScript Error objects are supported.

## **2.1.7 [DOM Level 2 - Events] Section 1.5, DocumentEvent interface**

V0023:

The specification states:

DocumentEvent interface provides a mechanism by which the user can create an Event of a type supported by the implementation. It is expected that the DocumentEvent interface will be implemented on the same object which implements the Document interface in an implementation which supports the Event model.

```
IDL Definition// Introduced in DOM Level 2:

interface DocumentEvent {
    Event createEvent(in DOMString eventType)raises(DOMException);
};
```

### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **DocumentEvent** interface is not implemented. Similar functionality is provided by the **createEventObject([oExistingEvent])** scripting method. Individual events are requested and set according to their name.

Parameters:

**oExistingEvent** Optional. An object that specifies an existing event object on which to base the new object.

Return value:

Returns an event object.

Event object properties are read-only once the **fireEvent** method is called.

### 2.1.8 [DOM Level 2 - Events] Section 1.6.1, User Interface event types

V0025:

The specification states:

```
A DOM application may use the hasFeature(feature, version) method of the
DOMImplementation interface with parameter values "UIEvents" and "2.0"
(respectively) to determine whether or not the User Interface event module is
supported by the implementation. In order to fully support this module, an
implementation must also support the "Events" feature defined in this specification
and the "Views" feature defined in the DOM Level 2 Views specification [DOM Level 2 Views].
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **hasFeature(sFeature [, vVersion])** method does not recognize `UIEvents` as a valid value for **sFeature** or `2.0` as a valid value for **vVersion**.

V0026:

The specification describes the **UIEvent** interface.

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **UIEvent** interface is not supported.

V0027:

The specification describes the following attributes of the **UIEvent** interface:

```
detail of type long, readonly
Specifies some detail information about the Event, depending on the type of event.

view of type views::AbstractView, readonly
The view attribute identifies the AbstractView from which the event was generated.
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **detail** and **view** attributes are not supported.

V0028:

The specification describes the **initUIEvent** method of the **UIEvent** interface.

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The following variations apply:

- The **initUIEvent** method is not supported.
- Event object properties are read-only after the **fireEvent** method is called.

V0029:

The specification describes the **DOMActivate** event of the **UIEvent** interface:

*Quirks Mode, IE7 Mode, IE8 Mode, IE9 Mode, IE10 Mode, and IE11 Mode (All Versions)*

The **DOMFocusIn** event is not supported. The **focusin** event is similar.

V0030:

The specification states:

DOMFocusOut

The DOMFocusOut event occurs when a EventTarget loses focus, for instance via a pointing device being moved out of an element or by tabbing navigation out of the element. Unlike the HTML event blur, DOMFocusOut can be applied to any focusable EventTarget, not just FORM controls.

Bubbles: Yes

Cancelable: No

Context Info: None

*Quirks Mode, IE7 Mode, IE8 Mode, IE9 Mode, IE10 Mode, and IE11 Mode (All Versions)*

The **DOMFocusOut** event is not supported. The **focusout** event is similar.

V0031:

The specification states:

DOMActivate

The activate event occurs when an element is activated, for instance, thru a mouse click or a keypress. A numerical argument is provided to give an indication of the type of activation that occurs: 1 for a simple activation (e.g. a simple click or Enter), 2 for hyperactivation (for instance a double click or Shift Enter).

Bubbles: Yes

Cancelable: Yes

Context Info: detail (the numerical value)

*All Document Modes (All Versions)*

The **DOMActivate** event is not supported. The **click** event is similar.

## **2.1.9 [DOM Level 2 - Events] Section 1.6.2, Mouse event types**

V0032:

The specification states:

A DOM application may use the hasFeature(feature, version) method of the DOMImplementation interface with parameter values "MouseEvents" and "2.0" (respectively) to determine whether or not the Mouse event module is supported by the implementation. In order to fully support this module, an implementation must also support the "UIEvents" feature defined in this specification. Please, refer to additional information about conformance in the DOM Level 2 Core specification [DOM Level 2 Core].

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **hasFeature(sFeature [, vVersion])** method does not recognize `MouseEvents` as a valid value for **sFeature** or `2.0` as a valid value for **vVersion**.

V0033:

The specification describes the **MouseEvent** interface.

*Quirks Mode and IE7 Standards Mode (All Versions)*

The **MouseEvent** interface is not supported.

The **MSEventObj** interface provides similar functionality to **MouseEvent**, with the following differences in behavior:

- To handle initialization, event object properties are read-only after the **fireEvent** method is called.
- The **button** attribute has additional possible values that allow for a mouse with a middle button.
- The **clientX** and **clientY** attributes each apply +2 pixels to the location of the border edge of the content area.
- The **mouseout** event is not cancellable.

*IE8 Mode (All Versions)*

The **MouseEvent** interface is not supported.

The **MSEventObj** interface provides similar functionality to **MouseEvent**, with the following differences in behavior:

- To handle initialization, event object properties are read-only after the **fireEvent** method is called.
- The **button** attribute has additional possible values that allow for a mouse with a middle button.
- The **mouseout** event is not cancellable.

V0036:

The specification states:

`click`

The click event occurs when the pointing device button is clicked over an element. A click is defined as a mousedown and mouseup over the same screen location.

The sequence of these events is: `mousedownmouseupclick`

If multiple clicks occur at the same screen location, the sequence repeats with the detail attribute incrementing with each repetition. This event is valid for most elements.

Bubbles: Yes

Cancelable: Yes

Context Info: `screenX`, `screenY`, `clientX`, `clientY`, `altKey`, `ctrlKey`, `shiftKey`, `metaKey`, `button`, `detail`

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **detail** and **metaKey** properties are not supported.

The **click** event is not supported on **optgroup** or **option** elements.

V0037:

The specification states:

```
mousedown
The mousedown event occurs when the pointing device button is pressed over an
element. This event is valid for most elements.

Bubbles: Yes
Cancelable: Yes
Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey,
metaKey, button, detail
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **detail** and **metaKey** properties are not supported.

The **mousedown** event is not supported on **optgroup** or **option** elements.

V0038:

The specification states:

```
mouseup
The mouseup event occurs when the pointing device button is released over an
element. This event is valid for most elements.

Bubbles: Yes
Cancelable: Yes
Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey,
metaKey, button, detail
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **detail** and **metaKey** properties are not supported.

The **mouseup** event is not supported on **optgroup** or **option** elements.

V0039:

The specification states:

```
mouseover
The mouseover event occurs when the pointing device is moved onto an element. This
event is valid for most elements.

Bubbles: Yes
Cancelable: Yes
Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey,
metaKey, relatedTarget indicates the EventTarget the pointing device is exiting.
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **detail** and **metaKey** properties are not supported.

The **mouseover** event is not supported on **optgroup** or **option** elements.

V0040:

The specification states:

```
mousemove
```

The `mousemove` event occurs when the pointing device is moved while it is over an element. This event is valid for most elements.

Bubbles: Yes  
Cancelable: No  
Context Info: `screenX`, `screenY`, `clientX`, `clientY`, `altKey`, `ctrlKey`, `shiftKey`, `metaKey`

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **detail** and **metaKey** properties are not supported.

The **mousemove** event is not supported on **optgroup** or **option** elements.

V0041:

The specification states:

```
mouseout
The mouseout event occurs when the pointing device is moved away from
an element. This event is valid for most elements.

Bubbles: Yes
Cancelable: Yes
Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey,
metaKey, relatedTarget indicates the EventTarget the pointing device is entering.
```

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **detail** and **metaKey** properties are not supported.

The **mouseout** event is not supported on **optgroup** or **option** elements.

### **2.1.10 [DOM Level 2 - Events] Section 1.6.4, Mutation event types**

V0042:

The specification states:

```
A DOM application may use the hasFeature(feature, version) method of the
DOMImplementation interface with parameter values "MutationEvents" and "2.0"
(respectively) to determine whether or not the Mutation event module is supported
by the implementation. In order to fully support this module, an implementation
must also support the "Events" feature defined in this specification. Please, refer
to additional information about conformance in the DOM Level 2 Core specification
[DOM Level 2 Core].
```

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **hasFeature(sFeature [, vVersion])** method does not recognize `MutationEvents` as a valid value for **sFeature** or `2.0` as a valid value for **vVersion**.

V0043:

The specification describes the **MutationEvent** interface.

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **MutationEvent** interface is not supported.

V0101:

The specification describes the following attributes of the **MutationEvent** interface:

Attributes

`attrChange` of type unsigned short, readonly

`attrChange` indicates the type of change which triggered the `DOMAttrModified` event. The values can be `MODIFICATION`, `ADDITION`, or `REMOVAL`.

`attrName` of type `DOMString`, readonly

`attrName` indicates the name of the changed `Attr` node in a `DOMAttrModified` event.

`newValue` of type `DOMString`, readonly

`newValue` indicates the new value of the `Attr` node in `DOMAttrModified` events, and of the `CharacterData` node in `DOMCharDataModified` events.

`prevValue` of type `DOMString`, readonly

`prevValue` indicates the previous value of the `Attr` node in `DOMAttrModified` events, and of the `CharacterData` node in `DOMCharDataModified` events.

`relatedNode` of type `Node`, readonly

`relatedNode` is used to identify a secondary node related to a mutation event. For example, if a mutation event is dispatched to a node indicating that its parent has changed, the `relatedNode` is the changed parent. If an event is instead dispatched to a subtree indicating a node was changed within it, the `relatedNode` is the changed node. In the case of the `DOMAttrModified` event it indicates the `Attr` node which was modified, added, or removed.

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **`attrChange`**, **`attrName`**, **`newValue`**, **`prevValue`**, and **`relatedNode`** attributes are not supported.

*IE9 Mode, IE10 Mode, IE11 Mode, and EdgeHTML Mode (All Versions)*

The **`prevValue`** attribute is supported. However, it provides only the previous value for **`DOMAttrModified`** events, not the previous value for **`DOMCharacterDataModified`** events. For **`DOMCharacterDataModified`** events, **`prevValue`** provides the empty string.

V0103:

The specification states:

`DOMSubtreeModified`

This is a general event for notification of all changes to the document. It can be used instead of the more specific events listed below. It may be fired after a single modification to the document or, at the implementation's discretion, after multiple changes have occurred. The latter use should generally be used to accommodate multiple changes which occur either simultaneously or in rapid succession. The target of this event is the lowest common parent of the changes which have taken place. This event is dispatched after any other events caused by the mutation have fired.

Bubbles: Yes

Cancelable: No

Context Info: None

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **`DOMSubtreeModified`** event is not supported.

*IE9 Mode, IE10 Mode, and IE11 Mode (All Versions)*

The **`DOMSubtreeModified`** event does not fire when appending text nodes with the **`appendChild`** method. For example, `elm.appendChild(document.createTextNode("abc"))`; does not cause the event to fire.

V0109:

The specification states:

```
DOMCharacterDataModified
Fired after CharacterData within a node has been modified but the node itself has not been
inserted or deleted. This event is also triggered by modifications to PI elements. The target
of this event is the CharacterData node.
```

```
Bubbles: Yes
Cancelable: No
Context Info: prevValue, newValue
```

*All Document Modes (All Versions)*

The **DOMCharacterDataModified** event is not supported.

*IE9 Mode, IE10 Mode, IE11 Mode, and EdgeHTML Mode (All Versions)*

The **DOMCharacterDataModified** event does not fire when appending text nodes with the **appendChild** method. For example, `elm.appendChild(document.createTextNode("abc"))`; does not cause the event to fire.

### 2.1.11 [DOM Level 2 - Events] Section 1.6.5, HTML event types

V0044:

The specification states:

```
A DOM application may use the hasFeature(feature, version) method of the
DOMImplementation interface with parameter values "HTMLEvents" and "2.0"
(respectively) to determine whether or not the HTML event module is supported by
the implementation. In order to fully support this module, an implementation must
also support the "Events" feature defined in this specification. Please, refer to
additional information about conformance in the DOM Level 2 Core specification
[DOM Level 2 Core].
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **hasFeature(sFeature [, vVersion])** method does not recognize `HTMLEvents` as a valid value for **sFeature** or `2.0` as a valid value for **vVersion**.

V0045:

The specification states:

```
load
The load event occurs when the DOM implementation finishes loading all content
within a document, all frames within a FRAMESET, or an OBJECT element.
```

```
Bubbles: No
Cancelable: No
Context Info: None
```

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

Images also cause a load event to occur, regardless of whether images are embedded or included as objects.

V0046:

The specification states:

`unload`  
The `unload` event occurs when the DOM implementation removes a document from a window or frame. This event is valid for `BODY` and `FRAMESET` elements.

Bubbles: No  
Cancelable: No  
Context Info: None

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **onbeforeunload** event allows the script to stop the unload process.

V0047:

The specification states:

`abort`  
The `abort` event occurs when page loading is stopped before an image has been allowed to completely load. This event applies to `OBJECT` elements.

Bubbles: Yes  
Cancelable: No  
Context Info: None

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **abort** event does not bubble and can be canceled.

V0048:

The specification states:

`error`  
The `error` event occurs when an image does not load properly or when an error occurs during script execution. This event is valid for `OBJECT` elements, `BODY` elements, and `FRAMESET` element.

Bubbles: Yes  
Cancelable: No  
Context Info: None

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **error** event does not bubble and can be canceled. Canceling the **error** event on the object prevents Internet Explorer from executing object fallback.

V0049:

The specification states:

`select`  
The `select` event occurs when a user selects some text in a text field. This event is valid for `INPUT` and `TEXTAREA` elements.

Bubbles: Yes

Cancelable: No  
Context Info: None

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **select** event does not bubble and can be canceled. The **select** event is also raised on text selection in the **body** element and child text nodes of the **body** element. When selecting, this event is raised repeatedly as the selection changes. The **onselectstart** and **onselectionchange** events are also supported.

V0050:

The specification states:

```
change
The change event occurs when a control loses the input focus and its value has been
modified since gaining focus. This event is valid for INPUT, SELECT, and TEXTAREA element.

Bubbles: Yes
Cancelable: No
Context Info: None
```

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **onchange** event does not bubble and can be canceled.

For radio input elements, the **onchange** event is fired only when the element is changed from being checked to being unchecked.

#### *IE9 Mode, IE10 Mode, IE11 Mode, and EdgeHTML Mode (All Versions)*

For radio input elements, the **onchange** event is fired only when the element is changed from being unchecked to being checked.

V0051:

The specification states:

```
submit
The submit event occurs when a form is submitted. This event only applies to the
FORM element.

Bubbles: Yes
Cancelable: Yes
Context Info: None
```

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **submit** event does not bubble.

V0052:

The specification states:

```
reset
The reset event occurs when a form is reset. This event only applies to the FORM element.

Bubbles: Yes
```

Cancelable: No  
Context Info: None

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **reset** event does not bubble.

V0053:

The specification states:

focus  
The focus event occurs when an element receives focus either via a pointing device or by tabbing navigation. This event is valid for the following elements: LABEL, INPUT, SELECT, TEXTAREA, and BUTTON.

Bubbles: No  
Cancelable: No  
Context Info: None

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

Any element that has a **tabindex** attribute specified is able to receive the **focus** event.

V0054:

The specification states:

blur  
The blur event occurs when an element loses focus either via the pointing device or by tabbing navigation. This event is valid for the following elements: LABEL, INPUT, SELECT, TEXTAREA, and BUTTON.

Bubbles: No  
Cancelable: No  
Context Info: None

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

Any element that has a **tabindex** attribute specified is able to receive the **blur** event.

V0055:

The specification states:

resize  
The resize event occurs when a document view is resized.

Bubbles: Yes  
Cancelable: No  
Context Info: None

#### *Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)*

The **resize** event does not bubble. This event also fires on elements when their rendered size changes.

#### *IE8 Mode (All Versions)*

CSS property updates that change the size of an element are consolidated into one event to prevent exponential growth of **resize** events.

## **2.2 Clarifications**

There are no clarifications of the MAY and SHOULD requirements of [\[DOM Level 2 - Events\]](#).

## **2.3 Error Handling**

There are no additional error handling considerations.

## **2.4 Security**

There are no additional security considerations.

### 3 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

## 4 Index

### B

[Basic event flow](#) 7

### C

[Change tracking](#) 28

### D

[DocumentEvent interface](#) 16

### E

[Event bubbling](#) 8

[Event capture](#) 7

[Event interface](#) 13

[Event registration interfaces](#) 9

### G

[Glossary](#) 4

### H

[HTML event types](#) 23

### I

[Informative references](#) 4

[Interaction with HTML 4.0 event listeners](#) 12

[Introduction](#) 4

### M

[Mouse event types](#) 18

[Mutation event types](#) 21

### N

[Normative references](#) 4

### R

References

[informative](#) 4

[normative](#) 4

### T

[Tracking changes](#) 28

### U

[User Interface event types](#) 17