Document Object Model (DOM) Level 3 Events Specification

Version 1.0

W3C Working Draft *10 April 2001*

This version:
: http://www.w3.org/TR/2001/WD-DOM-Level-3-Events-20010410
: (PostScript file , PDF file , plain text , ZIP file , single HTML file)

Latest version:
: http://www.w3.org/TR/DOM-Level-3-Events

Previous version:
: http://www.w3.org/TR/2000/WD-DOM-Level-3-Events-20000901/

Editors:
: Tom Pixley, *Netscape Communications Corporation*

Abstract

This specification defines the Document Object Model Events Level 3, a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. The Document Object Model Events Level 3 builds on the Document Object Model Events Level 2.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

This is a W3C Working Draft for review by W3C members and other interested parties.

It is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by, or the consensus of, either W3C or members of the DOM working group.

Comments on this document are invited and are to be sent to the public mailing list www-dom@w3.org. An archive is available at http://lists.w3.org/Archives/Public/www-dom/.

This document has been produced as part of the W3C DOM Activity. The authors of this document are the DOM WG members.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

Table of contents

Expanded Table of Contents

Expanded Table of Contents

Copyright Notice

W3C Document Copyright Notice and License

No right to create modifications or derivatives of W3C documents is granted pursuant to this license. However, if additional requirements (documented in the Copyright FAQ) are satisfied, the right to create modifications or derivatives is sometimes granted by the W3C to individuals complying with those requirements.

THIS DOCUMENT IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders.

W3C Software Copyright Notice and License

Note: This section is a copy of the W3C Software Copyright Notice and License and could be found at http://www.w3.org/Consortium/Legal/copyright-software-19980720

Copyright © 1994-2001 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

http://www.w3.org/Consortium/Legal/

This W3C work (including software, documents, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, and modify this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications, that you make:

1. The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.
2. Any pre-existing intellectual property disclaimers. If none exist, then a notice of the following form: "Copyright © [$date-of-software] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/."

1. Document Object Model Events

Editors

Tom Pixley, Netscape Communications Corporation

1.1. Level 3 Events Overview

The goal of the DOM Level 3 Events specification is to expand upon the functionality specified in the DOM Level 2 Event Specification. The specification does this by adding new interfaces which are complimentary to the interfaces defined in the DOM Level 2 Event Specification as well as adding new event modules to those already defined.

This specification requires the previously designed interfaces in order to be functional. It is not designed to be standalone. These interfaces are not designed to supercede the interfaces already provided but instead to add to the functionality contained within them.

1.2. Level 3 Events Interfaces

1.2.1. Key events

A DOM application may use the `hasFeature(feature, version)` method of the `DOMImplementation` interface with parameter values "KeyEvents" and "3.0" (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 3 Core specification .

Note: To create an instance of the `KeyEvent` [p.9] interface, use the feature string "KeyEvents" as the value of the input parameter used with the `createEvent` method of the `DocumentEvent` interface.

Interface *KeyEvent* (introduced in **DOM Level 3**)

The `KeyEvent` interface provides specific contextual information associated with Key Events.

The `detail` attribute inherited from `UIEvent` is used to indicated the number of keypresses which have occurred during key repetition. If this information is not available this value should be 0.

IDL Definition

```
// Introduced in DOM Level 3:
interface KeyEvent : UIEvent {

  // VirtualKeyCode
  const unsigned long       DOM_VK_UNDEFINED              = 0x0;
  const unsigned long       DOM_VK_RIGHT_ALT              = 0x01;
  const unsigned long       DOM_VK_LEFT_ALT               = 0x02;
  const unsigned long       DOM_VK_LEFT_CONTROL           = 0x03;
  const unsigned long       DOM_VK_RIGHT_CONTROL          = 0x04;
```

```
const unsigned long       DOM_VK_LEFT_SHIFT              = 0x05;
const unsigned long       DOM_VK_RIGHT_SHIFT             = 0x06;
const unsigned long       DOM_VK_LEFT_META               = 0x07;
const unsigned long       DOM_VK_RIGHT_META              = 0x08;
const unsigned long       DOM_VK_CAPS_LOCK               = 0x09;
const unsigned long       DOM_VK_DELETE                  = 0x0A;
const unsigned long       DOM_VK_END                     = 0x0B;
const unsigned long       DOM_VK_ENTER                   = 0x0C;
const unsigned long       DOM_VK_ESCAPE                  = 0x0D;
const unsigned long       DOM_VK_HOME                    = 0x0E;
const unsigned long       DOM_VK_INSERT                  = 0x0F;
const unsigned long       DOM_VK_NUM_LOCK                = 0x10;
const unsigned long       DOM_VK_PAUSE                   = 0x11;
const unsigned long       DOM_VK_PRINTSCREEN             = 0x12;
const unsigned long       DOM_VK_SCROLL_LOCK             = 0x13;
const unsigned long       DOM_VK_LEFT                    = 0x14;
const unsigned long       DOM_VK_RIGHT                   = 0x15;
const unsigned long       DOM_VK_UP                      = 0x16;
const unsigned long       DOM_VK_DOWN                    = 0x17;
const unsigned long       DOM_VK_PAGE_DOWN               = 0x18;
const unsigned long       DOM_VK_PAGE_UP                 = 0x19;
const unsigned long       DOM_VK_F1                      = 0x1A;
const unsigned long       DOM_VK_F2                      = 0x1B;
const unsigned long       DOM_VK_F3                      = 0x1C;
const unsigned long       DOM_VK_F4                      = 0x1D;
const unsigned long       DOM_VK_F5                      = 0x1E;
const unsigned long       DOM_VK_F6                      = 0x1F;
const unsigned long       DOM_VK_F7                      = 0x20;
const unsigned long       DOM_VK_F8                      = 0x21;
const unsigned long       DOM_VK_F9                      = 0x22;
const unsigned long       DOM_VK_F10                     = 0x23;
const unsigned long       DOM_VK_F11                     = 0x24;
const unsigned long       DOM_VK_F12                     = 0x25;
const unsigned long       DOM_VK_F13                     = 0x26;
const unsigned long       DOM_VK_F14                     = 0x27;
const unsigned long       DOM_VK_F15                     = 0x28;
const unsigned long       DOM_VK_F16                     = 0x29;
const unsigned long       DOM_VK_F17                     = 0x2A;
const unsigned long       DOM_VK_F18                     = 0x2B;
const unsigned long       DOM_VK_F19                     = 0x2C;
const unsigned long       DOM_VK_F20                     = 0x2D;
const unsigned long       DOM_VK_F21                     = 0x2E;
const unsigned long       DOM_VK_F22                     = 0x2F;
const unsigned long       DOM_VK_F23                     = 0x30;
const unsigned long       DOM_VK_F24                     = 0x31;

         attribute DOMString        outputString;
         attribute unsigned long    keyVal;
         attribute unsigned long    virtKeyVal;
         attribute boolean          inputGenerated;
         attribute boolean          numPad;
boolean            checkModifier(in unsigned long modifer);
void               initKeyEvent(in DOMString typeArg,
                                in boolean canBubbleArg,
                                in boolean cancelableArg,
                                in views::AbstractView viewArg,
                                in unsigned short detailArg,
```

```
                                        in DOMString outputStringArg,
                                        in unsigned long keyValArg,
                                        in unsigned long virtKeyValArg,
                                        in boolean inputGeneratedArg,
                                        in boolean numPadArg);
  void                initModifier(in unsigned long modifier,
                                        in boolean value);
};
```

Definition group *VirtualKeyCode*

An integer indicating which key was pressed.

Defined Constants

`DOM_VK_CAPS_LOCK`
`DOM_VK_DELETE`
`DOM_VK_DOWN`
`DOM_VK_END`
`DOM_VK_ENTER`
`DOM_VK_ESCAPE`
`DOM_VK_F1`
: Constant for the F1 function key.

`DOM_VK_F10`
: Constant for the F10 function key.

`DOM_VK_F11`
: Constant for the F11 function key.

`DOM_VK_F12`
: Constant for the F12 function key.

`DOM_VK_F13`
: Constant for the F13 function key.

`DOM_VK_F14`
: Constant for the F14 function key.

`DOM_VK_F15`
: Constant for the F15 function key.

`DOM_VK_F16`
: Constant for the F16 function key.

`DOM_VK_F17`
: Constant for the F17 function key.

`DOM_VK_F18`
: Constant for the F18 function key.

`DOM_VK_F19`
: Constant for the F19 function key.

`DOM_VK_F2`
: Constant for the F2 function key.

`DOM_VK_F20`
: Constant for the F20 function key.

`DOM_VK_F21`
: Constant for the F21 function key.

`DOM_VK_F22`
: Constant for the F22 function key.

`DOM_VK_F23`
: Constant for the F23 function key.

`DOM_VK_F24`
: Constant for the F24 function key.

`DOM_VK_F3`
: Constant for the F3 function key.

`DOM_VK_F4`
: Constant for the F4 function key.

`DOM_VK_F5`
: Constant for the F5 function key.

`DOM_VK_F6`
: Constant for the F6 function key.

`DOM_VK_F7`
: Constant for the F7 function key.

`DOM_VK_F8`
: Constant for the F8 function key.

`DOM_VK_F9`
: Constant for the F9 function key.

`DOM_VK_HOME`

`DOM_VK_INSERT`

`DOM_VK_LEFT`

`DOM_VK_LEFT_ALT`
: This key is a modifier key

`DOM_VK_LEFT_CONTROL`
: This key is a modifier key

`DOM_VK_LEFT_META`
: This key is a modifier key

`DOM_VK_LEFT_SHIFT`
: This key is a modifier key

`DOM_VK_NUM_LOCK`

`DOM_VK_PAGE_DOWN`

`DOM_VK_PAGE_UP`

`DOM_VK_PAUSE`

`DOM_VK_PRINTSCREEN`

`DOM_VK_RIGHT`

`DOM_VK_RIGHT_ALT`
: This key is a modifier key

`DOM_VK_RIGHT_CONTROL`
: This key is a modifier key

`DOM_VK_RIGHT_META`
: This key is a modifier key

`DOM_VK_RIGHT_SHIFT`
: This key is a modifier key

`DOM_VK_SCROLL_LOCK`
`DOM_VK_UNDEFINED`
Used for key events which do not have a virtual key code available.
`DOM_VK_UP`

Attributes

`inputGenerated` of type `boolean`
The `inputGenerated` attribute indicates whether the key event will normally cause visible output. If the key event does not generate any visible output, such as the use of a function key or the combination of certain modifier keys used in conjunction with another key, then the value will be false. If visible output is normally generated by the key event then the value will be true.
The value of `inputGenerated` does not guarantee the creation of a character. If a key event causing visible output is cancelable it may be prevented from causing output. This attribute is intended primarily to differentiate between keys events which may or may not produce visible output depending on the system state.

`keyVal` of type `unsigned long`
The value of `keyVal` holds the value of the Unicode character associated with the depressed key. If the key has no Unicode representation or no Unicode character is available the value is 0..

`numPad` of type `boolean`
The `numPad` attribute indicates whether or not the key event was generated on the number pad section of the keyboard. If the number pad was used to generate the key event the value is true, otherwise the value is false.

`outputString` of type `DOMString`
`outputString` holds the value of the output generated by the key event. This may be a single Unicode character or it may be a string. It may also be null in the case where no output was generated by the key event.

`virtKeyVal` of type `unsigned long`
When the key associated with a key event is not representable via a Unicode character `virtKeyVale` holds the virtual key code associated with the depressed key. If the key has a Unicode representation or no virtual code is available the value is `DOM_VK_UNDEFINED`.

Methods

`checkModifier`
The `CheckModifier` method is used to check the status of a single modifier key associated with a `KeyEvent`. The identifier of the modifier in question is passed into the `CheckModifier` function. If the modifier is triggered it will return true. If not, it will return false.
The list of keys below represents the allowable modifier paramaters for this method.

- DOM_VK_LEFT_ALT
- DOM_VK_RIGHT_ALT
- DOM_VK_LEFT_CONTROL
- DOM_VK_RIGHT_CONTROL
- DOM_VK_LEFT_SHIFT
- DOM_VK_RIGHT_SHIFT
- DOM_VK_META

Parameters

`modifer` of type `unsigned long`

The modifier which the user wishes to query.

Return Value

`boolean`	The status of the modifier represented as a boolean.

No Exceptions

`initKeyEvent`

The `initKeyEvent` method is used to initialize the value of a `MouseEvent` created through the `DocumentEvent` interface. This method may only be called before the `KeyEvent` 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. This method has no effect if called after the event has been dispatched.

Parameters

`typeArg` of type `DOMString`

Specifies the event type.

`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 prevent.

`viewArg` of type `views::AbstractView`

Specifies the `KeyEvent`'s `AbstractView`.

`detailArg` of type `unsigned short`

Specifies the number of repeated keypresses, if available.

`outputStringArg` of type `DOMString`

Specifies the `KeyEvent`'s `outputString` attribute

`keyValArg` of type `unsigned long`

Specifies the `KeyEvent`'s `keyVal`attribute

`virtKeyValArg` of type `unsigned long`

Specifies the `KeyEvent`'s `virtKeyVal`attribute

`inputGeneratedArg` of type `boolean`

Specifies the `KeyEvent`'s `inputGenerated`attribute

`numPadArg` of type `boolean`

Specifies the `KeyEvent`'s `numPad`attribute

No Return Value

No Exceptions

`initModifier`

The `initModifier` method is used to initialize the values of any modifiers associated with a `KeyEvent` created through the `DocumentEvent` interface. This method may only be called before the `KeyEvent` has been dispatched via the dispatchEvent method, though it may be called multiple times during that phase if necessary. If called multiple times with the same `modifier` property the final invocation takes precedence. Unless explicitly give a value of true, all modifiers have a value of false. This method has no effect if called after the event has been dispatched.

The list of keys below represents the allowable modifier paramaters for this method.

- DOM_VK_LEFT_ALT
- DOM_VK_RIGHT_ALT
- DOM_VK_LEFT_CONTROL
- DOM_VK_RIGHT_CONTROL
- DOM_VK_LEFT_SHIFT
- DOM_VK_RIGHT_SHIFT
- DOM_VK_META

Parameters

`modifier` of type `unsigned long`
: The modifier which the user wishes to initialize

`value` of type `boolean`
: The new value of the modifier.

No Return Value

No Exceptions

There are two major groups of key events. The first contains the `textEvent` event. The `textEvent` event indicates that text information has been entered, either in the form of printable characters or non-printable text information such as modifier keys. `textEvent` events are not necessarily accompanied by the events of the second major groups of key events, `keydown` and `keyup`.

textEvent

The textEvent event indicates that text information has been entered. The text information entered can originate from a variety of sources. It could, for example, be a character resulting from a keypress. It could also be a string resulting from an input method.

- Bubbles: Yes
- Cancelable: Yes

The `keydown` and `keyup` events comprise the second group of key events. These events are fired to indicate the physical motion of the keys on the character generation device. Depending on the input system being used, `textEvent` events may or may not be generated for each pair of `keydown` and `keyup` events.

keydown

The keydown event occurs when a key is pressed down.

- Bubbles: Yes
- Cancelable: Yes

keyup

The keyup event occurs when a key is released.

- Bubbles: Yes
- Cancelable: Yes

1.2.2. EventListener Grouping

EventListener grouping is intended to allow groups of `EventListeners` to be registered which will each have independent event flow within them which is not affected by changes to event flow in any other group. This may be used to control events separately in multiple views on a document. It may also be used to develop an application which uses events without the problem of possible interference by other applications running within the same document.

The new interfaces added for EventListener grouping should not interfere with the interfaces established in the Level 2 DOM. For purposes of interoperability between the Level 2 DOM Event Model and the new interfaces added in Level 3 the implementation can be assumed to define a default `EventGroup` [p.16] . This `EventGroup` is implicitly used in the registration of all `EventListeners` registered via the Level 2 DOM Event Model methods which do not specify an `EventGroup`.

Interface *EventGroup*

The EventGroup interface functions primarily as a placeholder for separating the event flows when there are multiple groups of listeners for a DOM tree.

`EventListeners` can be registered without an `EventGroup` using the existing `EventTarget` interface, or with an associated `EventGroup` using the new `EventTargetGroup` [p.16] interface. When an event is dispatched, it is dispatched independently to each `EventGroup`. In particular, the `stopPropagation` method of the `Event` interface only stops propagation within an `EventListener`'s associated `EventGroup`.

IDL Definition

```
interface EventGroup {
  boolean            isSameEventGroup(in EventGroup eventGroup);
};
```

Methods

`isSameEventGroup`

This method checks if the supplied `EventGroup` is the same as the `EventGroup` upon which the method is called.

Parameters

`eventGroup` of type `EventGroup` [p.16]

The `EventGroup` with which to check equality.

Return Value

`boolean`	Returns true if the `EventGroups` are equal, else returns false.

No Exceptions

Interface *EventTargetGroup*

The EventTargetGroup interface is implemented by the same set of objects that implement the `EventTarget` interface, namely all `EventTargets` in in implementation which supports the Event model and the EventGroup extension.

IDL Definition

```
interface EventTargetGroup {
  void               addEventListener(in DOMString type,
                                      in EventListener listener,
                                      in boolean useCapture,
                                      in EventGroup eventGroup);
  void               removeEventListener(in DOMString type,
                                         in EventListener listener,
                                         in boolean useCapture,
                                         in EventGroup eventGroup);
};
```

Methods

`addEventListener`

This method is equivalent to the `addEventListener` method of the `EventTarget` interface, with the exception of the added `eventGroup` parameter. The listener is registered with this `EventGroup` [p.16] associated.

Parameters

`type` of type `DOMString`

`listener` of type `EventListener`

`useCapture` of type `boolean`

`eventGroup` of type `EventGroup` [p.16]

The `EventGroup` to associate with the listener.

No Return Value

No Exceptions

`removeEventListener`

This method is equivalent to the `removeEventListener` method of the `EventTarget` interface, with the exception of the added `eventGroup` parameter. The listener registered with this `EventGroup` [p.16] associated is removed.

Parameters

`type` of type `DOMString`

`listener` of type `EventListener`

`useCapture` of type `boolean`

`eventGroup` of type `EventGroup` [p.16]

The `EventGroup` to associate with the listener.

No Return Value

No Exceptions

Interface *DocumentEventGroup*

The `DocumentEventGroup` interface provides a mechanism by which the user can create an `EventGroup` [p.16] 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 `EventGroup`extension.

IDL Definition

```
interface DocumentEventGroup {
  EventGroup         createEventGroup();
};
```

Methods

`createEventGroup`

This method creates a new EventGroup for use in the addEventListener and removeEventListener methods of the EventTargetGroup interface.

Return Value

`EventGroup` [p.16]	The newly created `EventGroup`.

No Parameters
No Exceptions

1.3. Issues

Issue getModifier:

Why is modifier state exposed through a method rather than an attribute?
Resolution: The modifier keys are not currently representable as bit flags. Setting them individually would therefore require an attribute for each. Rather than bloat the api, especially given the addition of left and right modifier keys, the modifiers are exposed via a single method.

Issue ISO-IEC-9995:

Have you coordinated this set with that defined by ISO/IEC 9995 which addresses various Keyboard symbol issues.
Resolution: Upon examination of the ISO spec we found it to be insufficient to our needs. It does not represent the left/right differentiation between some keys. It also lacks function keys.

Issue ISO-IEC-14755:

Review ISO/IEC 14755 "Input methods to enter characters from the repertoire of ISO/IEC 10646 with a keyboard or other input device" to insure that the treatment of input state is consistent with that expected by current practice when it comes to platforms which support input methods.

Issue offsets:

(This issue is related with mouse events and Views?)
it would be useful if MouseEvent class had a property that would enable listners to learn about coordinates of the event within the element's own coordinate system.

Issue unicodeidents:

Some of the unicode chars are pretty esoteric (i.e. home, end, scroll lock). Do we want to adopt these or will this be harder on users than defining them in the DOM Event Spec. About a dozen keys fit this pattern.

Issue texteventwithoutchargeneration:

The results of the discussions on switching the keypress event out for the textEvent were inconclusive on the question of whether to fire textEvents for non character generating keys input. This includes modifier keys, function keys, etc.

Appendix A: IDL Definitions

This appendix contains the complete OMG IDL [OMGIDL] for the Level 3 Document Object Model Events definitions.

The IDL files are also available as:
http://www.w3.org/TR/2001/WD-DOM-Level-3-Events-20010410/idl.zip

events.idl:

```
// File: events.idl

#ifndef _EVENTS_IDL_
#define _EVENTS_IDL_

#include "dom.idl"
#include "views.idl"

#pragma prefix "dom.w3c.org"
module events
{

  typedef dom::DOMString DOMString;
  typedef dom::EventListener EventListener;
  typedef dom::UIEvent UIEvent;

  interface EventGroup {
    boolean            isSameEventGroup(in EventGroup eventGroup);
  };

  interface EventTargetGroup {
    void               addEventListener(in DOMString type,
                                        in EventListener listener,
                                        in boolean useCapture,
                                        in EventGroup eventGroup);
    void               removeEventListener(in DOMString type,
                                           in EventListener listener,
                                           in boolean useCapture,
                                           in EventGroup eventGroup);
  };

  interface DocumentEventGroup {
    EventGroup         createEventGroup();
  };

  // Introduced in DOM Level 3:
  interface KeyEvent : UIEvent {

    // VirtualKeyCode
    const unsigned long       DOM_VK_UNDEFINED              = 0x0;
    const unsigned long       DOM_VK_RIGHT_ALT              = 0x01;
    const unsigned long       DOM_VK_LEFT_ALT               = 0x02;
    const unsigned long       DOM_VK_LEFT_CONTROL           = 0x03;
    const unsigned long       DOM_VK_RIGHT_CONTROL          = 0x04;
```

```
const unsigned long       DOM_VK_LEFT_SHIFT             = 0x05;
const unsigned long       DOM_VK_RIGHT_SHIFT            = 0x06;
const unsigned long       DOM_VK_LEFT_META              = 0x07;
const unsigned long       DOM_VK_RIGHT_META             = 0x08;
const unsigned long       DOM_VK_CAPS_LOCK              = 0x09;
const unsigned long       DOM_VK_DELETE                 = 0x0A;
const unsigned long       DOM_VK_END                    = 0x0B;
const unsigned long       DOM_VK_ENTER                  = 0x0C;
const unsigned long       DOM_VK_ESCAPE                 = 0x0D;
const unsigned long       DOM_VK_HOME                   = 0x0E;
const unsigned long       DOM_VK_INSERT                 = 0x0F;
const unsigned long       DOM_VK_NUM_LOCK               = 0x10;
const unsigned long       DOM_VK_PAUSE                  = 0x11;
const unsigned long       DOM_VK_PRINTSCREEN            = 0x12;
const unsigned long       DOM_VK_SCROLL_LOCK            = 0x13;
const unsigned long       DOM_VK_LEFT                   = 0x14;
const unsigned long       DOM_VK_RIGHT                  = 0x15;
const unsigned long       DOM_VK_UP                     = 0x16;
const unsigned long       DOM_VK_DOWN                   = 0x17;
const unsigned long       DOM_VK_PAGE_DOWN              = 0x18;
const unsigned long       DOM_VK_PAGE_UP                = 0x19;
const unsigned long       DOM_VK_F1                     = 0x1A;
const unsigned long       DOM_VK_F2                     = 0x1B;
const unsigned long       DOM_VK_F3                     = 0x1C;
const unsigned long       DOM_VK_F4                     = 0x1D;
const unsigned long       DOM_VK_F5                     = 0x1E;
const unsigned long       DOM_VK_F6                     = 0x1F;
const unsigned long       DOM_VK_F7                     = 0x20;
const unsigned long       DOM_VK_F8                     = 0x21;
const unsigned long       DOM_VK_F9                     = 0x22;
const unsigned long       DOM_VK_F10                    = 0x23;
const unsigned long       DOM_VK_F11                    = 0x24;
const unsigned long       DOM_VK_F12                    = 0x25;
const unsigned long       DOM_VK_F13                    = 0x26;
const unsigned long       DOM_VK_F14                    = 0x27;
const unsigned long       DOM_VK_F15                    = 0x28;
const unsigned long       DOM_VK_F16                    = 0x29;
const unsigned long       DOM_VK_F17                    = 0x2A;
const unsigned long       DOM_VK_F18                    = 0x2B;
const unsigned long       DOM_VK_F19                    = 0x2C;
const unsigned long       DOM_VK_F20                    = 0x2D;
const unsigned long       DOM_VK_F21                    = 0x2E;
const unsigned long       DOM_VK_F22                    = 0x2F;
const unsigned long       DOM_VK_F23                    = 0x30;
const unsigned long       DOM_VK_F24                    = 0x31;

         attribute DOMString        outputString;
         attribute unsigned long    keyVal;
         attribute unsigned long    virtKeyVal;
         attribute boolean          inputGenerated;
         attribute boolean          numPad;
boolean            checkModifier(in unsigned long modifer);
void               initKeyEvent(in DOMString typeArg,
                                in boolean canBubbleArg,
                                in boolean cancelableArg,
                                in views::AbstractView viewArg,
                                in unsigned short detailArg,
```

```
                                    in DOMString outputStringArg,
                                    in unsigned long keyValArg,
                                    in unsigned long virtKeyValArg,
                                    in boolean inputGeneratedArg,
                                    in boolean numPadArg);
    void                 initModifier(in unsigned long modifier,
                                    in boolean value);
  };
};

#endif // _EVENTS_IDL_
```

events.idl:

Appendix B: Java Language Binding

This appendix contains the complete Java [Java] bindings for the Level 3 Document Object Model Events.

The Java files are also available as
http://www.w3.org/TR/2001/WD-DOM-Level-3-Events-20010410/java-binding.zip

org/w3c/dom/events/KeyEvent.java:

```
package org.w3c.dom.events;

import org.w3c.dom.views.AbstractView;
import org.w3c.dom.UIEvent;

public interface KeyEvent extends UIEvent {
    // VirtualKeyCode
    public static final int DOM_VK_UNDEFINED          = 0x0;
    public static final int DOM_VK_RIGHT_ALT          = 0x01;
    public static final int DOM_VK_LEFT_ALT           = 0x02;
    public static final int DOM_VK_LEFT_CONTROL       = 0x03;
    public static final int DOM_VK_RIGHT_CONTROL      = 0x04;
    public static final int DOM_VK_LEFT_SHIFT         = 0x05;
    public static final int DOM_VK_RIGHT_SHIFT        = 0x06;
    public static final int DOM_VK_LEFT_META          = 0x07;
    public static final int DOM_VK_RIGHT_META         = 0x08;
    public static final int DOM_VK_CAPS_LOCK          = 0x09;
    public static final int DOM_VK_DELETE             = 0x0A;
    public static final int DOM_VK_END                = 0x0B;
    public static final int DOM_VK_ENTER              = 0x0C;
    public static final int DOM_VK_ESCAPE             = 0x0D;
    public static final int DOM_VK_HOME               = 0x0E;
    public static final int DOM_VK_INSERT             = 0x0F;
    public static final int DOM_VK_NUM_LOCK           = 0x10;
    public static final int DOM_VK_PAUSE              = 0x11;
    public static final int DOM_VK_PRINTSCREEN        = 0x12;
    public static final int DOM_VK_SCROLL_LOCK        = 0x13;
    public static final int DOM_VK_LEFT               = 0x14;
    public static final int DOM_VK_RIGHT              = 0x15;
    public static final int DOM_VK_UP                 = 0x16;
    public static final int DOM_VK_DOWN               = 0x17;
    public static final int DOM_VK_PAGE_DOWN          = 0x18;
    public static final int DOM_VK_PAGE_UP            = 0x19;
    public static final int DOM_VK_F1                 = 0x1A;
    public static final int DOM_VK_F2                 = 0x1B;
    public static final int DOM_VK_F3                 = 0x1C;
    public static final int DOM_VK_F4                 = 0x1D;
    public static final int DOM_VK_F5                 = 0x1E;
    public static final int DOM_VK_F6                 = 0x1F;
    public static final int DOM_VK_F7                 = 0x20;
    public static final int DOM_VK_F8                 = 0x21;
    public static final int DOM_VK_F9                 = 0x22;
    public static final int DOM_VK_F10                = 0x23;
    public static final int DOM_VK_F11                = 0x24;
    public static final int DOM_VK_F12                = 0x25;
```

```
    public static final int DOM_VK_F13                = 0x26;
    public static final int DOM_VK_F14                = 0x27;
    public static final int DOM_VK_F15                = 0x28;
    public static final int DOM_VK_F16                = 0x29;
    public static final int DOM_VK_F17                = 0x2A;
    public static final int DOM_VK_F18                = 0x2B;
    public static final int DOM_VK_F19                = 0x2C;
    public static final int DOM_VK_F20                = 0x2D;
    public static final int DOM_VK_F21                = 0x2E;
    public static final int DOM_VK_F22                = 0x2F;
    public static final int DOM_VK_F23                = 0x30;
    public static final int DOM_VK_F24                = 0x31;

    public String getOutputString();
    public void setOutputString(String outputString);

    public int getKeyVal();
    public void setKeyVal(int keyVal);

    public int getVirtKeyVal();
    public void setVirtKeyVal(int virtKeyVal);

    public boolean getInputGenerated();
    public void setInputGenerated(boolean inputGenerated);

    public boolean getNumPad();
    public void setNumPad(boolean numPad);

    public boolean checkModifier(int modifer);

    public void initKeyEvent(String typeArg,
                             boolean canBubbleArg,
                             boolean cancelableArg,
                             AbstractView viewArg,
                             short detailArg,
                             String outputStringArg,
                             int keyValArg,
                             int virtKeyValArg,
                             boolean inputGeneratedArg,
                             boolean numPadArg);

    public void initModifier(int modifier,
                             boolean value);

}
```

org/w3c/dom/events/EventGroup.java:

```
package org.w3c.dom.events;

public interface EventGroup {
    public boolean isSameEventGroup(EventGroup eventGroup);

}
```

org/w3c/dom/events/EventTargetGroup.java:

```
package org.w3c.dom.events;

import org.w3c.dom.EventListener;

public interface EventTargetGroup {
    public void addEventListener(String type,
                                 EventListener listener,
                                 boolean useCapture,
                                 EventGroup eventGroup);

    public void removeEventListener(String type,
                                    EventListener listener,
                                    boolean useCapture,
                                    EventGroup eventGroup);

}
```

org/w3c/dom/events/DocumentEventGroup.java:

```
package org.w3c.dom.events;

public interface DocumentEventGroup {
    public EventGroup createEventGroup();

}
```

org/w3c/dom/events/DocumentEventGroup.java:

Appendix C: ECMA Script Language Binding

This appendix contains the complete ECMA Script [ECMAScript] binding for the Level 3 Document Object Model Events definitions.

Prototype Object **KeyEvent**
The **KeyEvent** class has the following constants:
KeyEvent.DOM_VK_UNDEFINED
This constant is of type **Number** and its value is **0x0**.
KeyEvent.DOM_VK_RIGHT_ALT
This constant is of type **Number** and its value is **0x01**.
KeyEvent.DOM_VK_LEFT_ALT
This constant is of type **Number** and its value is **0x02**.
KeyEvent.DOM_VK_LEFT_CONTROL
This constant is of type **Number** and its value is **0x03**.
KeyEvent.DOM_VK_RIGHT_CONTROL
This constant is of type **Number** and its value is **0x04**.
KeyEvent.DOM_VK_LEFT_SHIFT
This constant is of type **Number** and its value is **0x05**.
KeyEvent.DOM_VK_RIGHT_SHIFT
This constant is of type **Number** and its value is **0x06**.
KeyEvent.DOM_VK_LEFT_META
This constant is of type **Number** and its value is **0x07**.
KeyEvent.DOM_VK_RIGHT_META
This constant is of type **Number** and its value is **0x08**.
KeyEvent.DOM_VK_CAPS_LOCK
This constant is of type **Number** and its value is **0x09**.
KeyEvent.DOM_VK_DELETE
This constant is of type **Number** and its value is **0x0A**.
KeyEvent.DOM_VK_END
This constant is of type **Number** and its value is **0x0B**.
KeyEvent.DOM_VK_ENTER
This constant is of type **Number** and its value is **0x0C**.
KeyEvent.DOM_VK_ESCAPE
This constant is of type **Number** and its value is **0x0D**.
KeyEvent.DOM_VK_HOME
This constant is of type **Number** and its value is **0x0E**.
KeyEvent.DOM_VK_INSERT
This constant is of type **Number** and its value is **0x0F**.
KeyEvent.DOM_VK_NUM_LOCK
This constant is of type **Number** and its value is **0x10**.
KeyEvent.DOM_VK_PAUSE
This constant is of type **Number** and its value is **0x11**.
KeyEvent.DOM_VK_PRINTSCREEN
This constant is of type **Number** and its value is **0x12**.

KeyEvent.DOM_VK_SCROLL_LOCK
This constant is of type **Number** and its value is **0x13**.
KeyEvent.DOM_VK_LEFT
This constant is of type **Number** and its value is **0x14**.
KeyEvent.DOM_VK_RIGHT
This constant is of type **Number** and its value is **0x15**.
KeyEvent.DOM_VK_UP
This constant is of type **Number** and its value is **0x16**.
KeyEvent.DOM_VK_DOWN
This constant is of type **Number** and its value is **0x17**.
KeyEvent.DOM_VK_PAGE_DOWN
This constant is of type **Number** and its value is **0x18**.
KeyEvent.DOM_VK_PAGE_UP
This constant is of type **Number** and its value is **0x19**.
KeyEvent.DOM_VK_F1
This constant is of type **Number** and its value is **0x1A**.
KeyEvent.DOM_VK_F2
This constant is of type **Number** and its value is **0x1B**.
KeyEvent.DOM_VK_F3
This constant is of type **Number** and its value is **0x1C**.
KeyEvent.DOM_VK_F4
This constant is of type **Number** and its value is **0x1D**.
KeyEvent.DOM_VK_F5
This constant is of type **Number** and its value is **0x1E**.
KeyEvent.DOM_VK_F6
This constant is of type **Number** and its value is **0x1F**.
KeyEvent.DOM_VK_F7
This constant is of type **Number** and its value is **0x20**.
KeyEvent.DOM_VK_F8
This constant is of type **Number** and its value is **0x21**.
KeyEvent.DOM_VK_F9
This constant is of type **Number** and its value is **0x22**.
KeyEvent.DOM_VK_F10
This constant is of type **Number** and its value is **0x23**.
KeyEvent.DOM_VK_F11
This constant is of type **Number** and its value is **0x24**.
KeyEvent.DOM_VK_F12
This constant is of type **Number** and its value is **0x25**.
KeyEvent.DOM_VK_F13
This constant is of type **Number** and its value is **0x26**.
KeyEvent.DOM_VK_F14
This constant is of type **Number** and its value is **0x27**.
KeyEvent.DOM_VK_F15
This constant is of type **Number** and its value is **0x28**.
KeyEvent.DOM_VK_F16
This constant is of type **Number** and its value is **0x29**.

KeyEvent.DOM_VK_F17
This constant is of type **Number** and its value is **0x2A**.
KeyEvent.DOM_VK_F18
This constant is of type **Number** and its value is **0x2B**.
KeyEvent.DOM_VK_F19
This constant is of type **Number** and its value is **0x2C**.
KeyEvent.DOM_VK_F20
This constant is of type **Number** and its value is **0x2D**.
KeyEvent.DOM_VK_F21
This constant is of type **Number** and its value is **0x2E**.
KeyEvent.DOM_VK_F22
This constant is of type **Number** and its value is **0x2F**.
KeyEvent.DOM_VK_F23
This constant is of type **Number** and its value is **0x30**.
KeyEvent.DOM_VK_F24
This constant is of type **Number** and its value is **0x31**.

Object **KeyEvent**

KeyEvent has the all the properties and methods of the **UIEvent** object as well as the properties and methods defined below.

The **KeyEvent** object has the following properties:

outputString
This property is of type **String**.
keyVal
This property is of type **Number**.
virtKeyVal
This property is of type **Number**.
inputGenerated
This property is of type **Boolean**.
numPad
This property is of type **Boolean**.

The **KeyEvent** object has the following methods:

checkModifier(modifer)
This method returns a **Boolean**.
The **modifer** parameter is of type **Number**.
initKeyEvent(typeArg, canBubbleArg, cancelableArg, viewArg, detailArg, outputStringArg, keyValArg, virtKeyValArg, inputGeneratedArg, numPadArg)
This method has no return value.
The **typeArg** parameter is of type **String**.
The **canBubbleArg** parameter is of type **Boolean**.
The **cancelableArg** parameter is of type **Boolean**.
The **viewArg** parameter is a **AbstractView** object.
The **detailArg** parameter is of type **Number**.
The **outputStringArg** parameter is of type **String**.
The **keyValArg** parameter is of type **Number**.
The **virtKeyValArg** parameter is of type **Number**.
The **inputGeneratedArg** parameter is of type **Boolean**.

The **numPadArg** parameter is of type **Boolean**.

initModifier(modifier, value)

This method has no return value.

The **modifier** parameter is of type **Number**.

The **value** parameter is of type **Boolean**.

Object **EventGroup**

The **EventGroup** object has the following methods:

isSameEventGroup(eventGroup)

This method returns a **Boolean**.

The **eventGroup** parameter is a **EventGroup** object.

Object **EventTargetGroup**

The **EventTargetGroup** object has the following methods:

addEventListener(type, listener, useCapture, eventGroup)

This method has no return value.

The **type** parameter is of type **String**.

The **listener** parameter is a **EventListener** object.

The **useCapture** parameter is of type **Boolean**.

The **eventGroup** parameter is a **EventGroup** object.

removeEventListener(type, listener, useCapture, eventGroup)

This method has no return value.

The **type** parameter is of type **String**.

The **listener** parameter is a **EventListener** object.

The **useCapture** parameter is of type **Boolean**.

The **eventGroup** parameter is a **EventGroup** object.

Object **DocumentEventGroup**

The **DocumentEventGroup** object has the following methods:

createEventGroup()

This method returns a **EventGroup** object.

References

For the latest version of any W3C specification please consult the list of W3C Technical Reports available at http://www.w3.org/TR.

D.1: Normative references

ECMAScript
: ECMA (European Computer Manufacturers Association) ECMAScript Language Specification. Available at http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM

Java
: Sun Microsystems Inc. The Java Language Specification, James Gosling, Bill Joy, and Guy Steele, September 1996. Available at http://java.sun.com/docs/books/jls

OMGIDL
: OMG (Object Management Group) IDL (Interface Definition Language) defined in The Common Object Request Broker: Architecture and Specification, version 2.3.1, October 1999. Available from http://www.omg.org

D.1: Normative references

Index