CS 175
Action Script

Input

Introduction

- User interaction, whether by keyboard, mouse, camera, or a combination of these devices, is the foundation of interactivity.

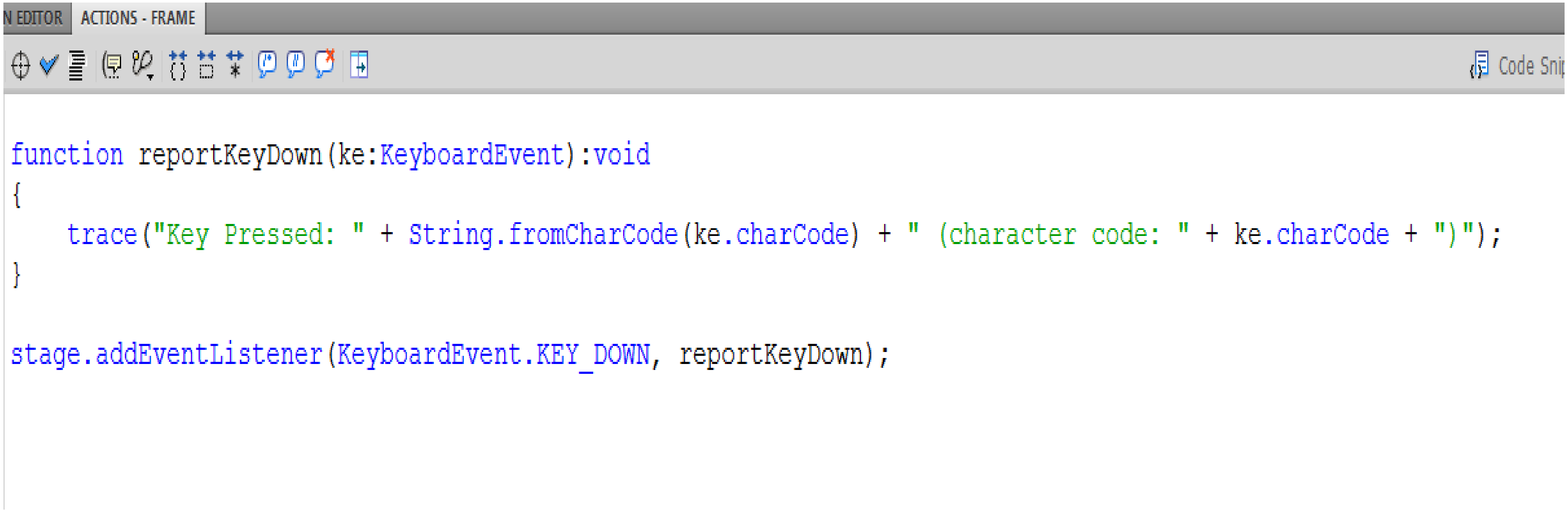
- In ActionScript 3.0, identifying and responding to user interaction primarily involves listening to events.

Keyboard Input

Keyboard Input

- Keyboard input is caught by a ***KeyboardEvent*** listener.

- For example, let's place an event listener on the stage to listen for and respond to keyboard input.

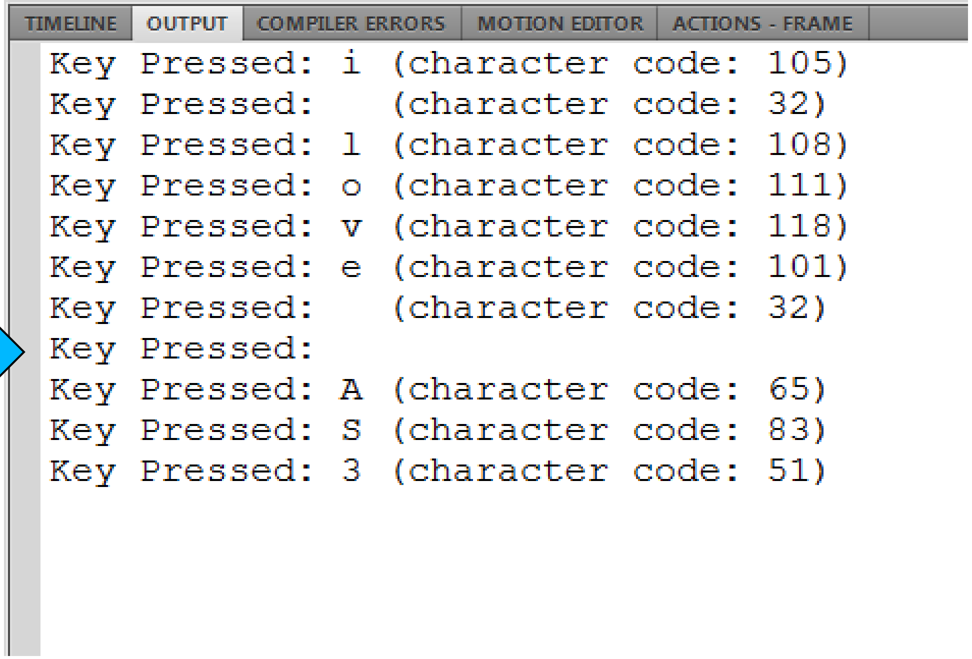


```
function reportKeyDown(ke:KeyboardEvent):void
{
    trace("Key Pressed: " + String.fromCharCode(ke.charCode) + " (character code: " + ke.charCode + ")");
}

stage.addEventListener(KeyboardEvent.KEY_DOWN, reportKeyDown);
```

Keyboard Input

- Testing our code:



```
Key Pressed: i (character code: 105)
Key Pressed:   (character code: 32)
Key Pressed: l (character code: 108)
Key Pressed: o (character code: 111)
Key Pressed: v (character code: 118)
Key Pressed: e (character code: 101)
Key Pressed:   (character code: 32)
Key Pressed:
Key Pressed: A (character code: 65)
Key Pressed: S (character code: 83)
Key Pressed: 3 (character code: 51)
```

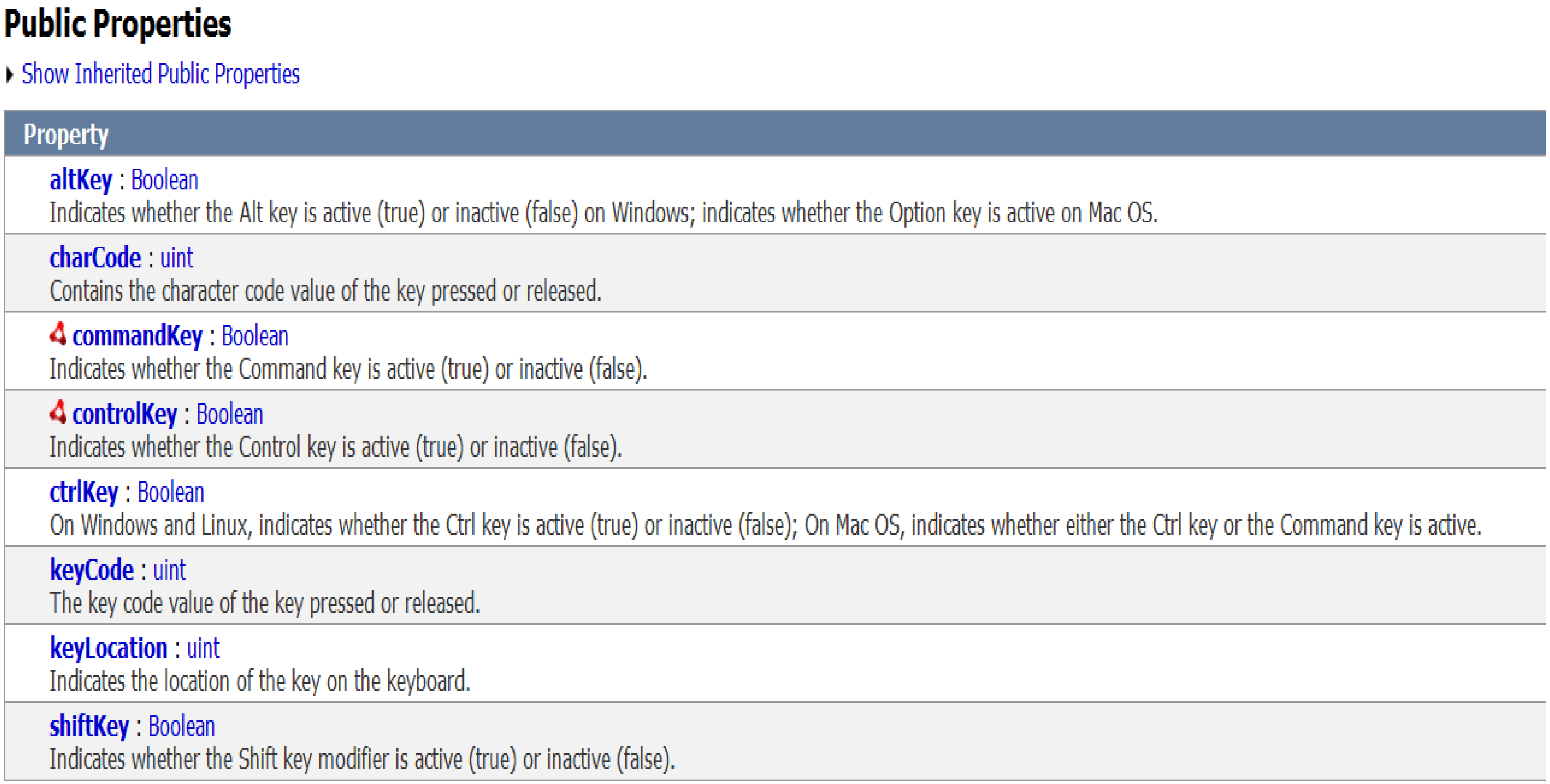
"shift" key pressed →

- The keyboard event listener captured keyboard input for the entire Stage.

Note: You can also write an event listener for a specific display object on the Stage; this event listener is triggered when the object has the focus.

KeyboardEvent Class

- Like any class, the KeyboardEvent class has attributes. In the previous example we used the ***“charCode”*** property.

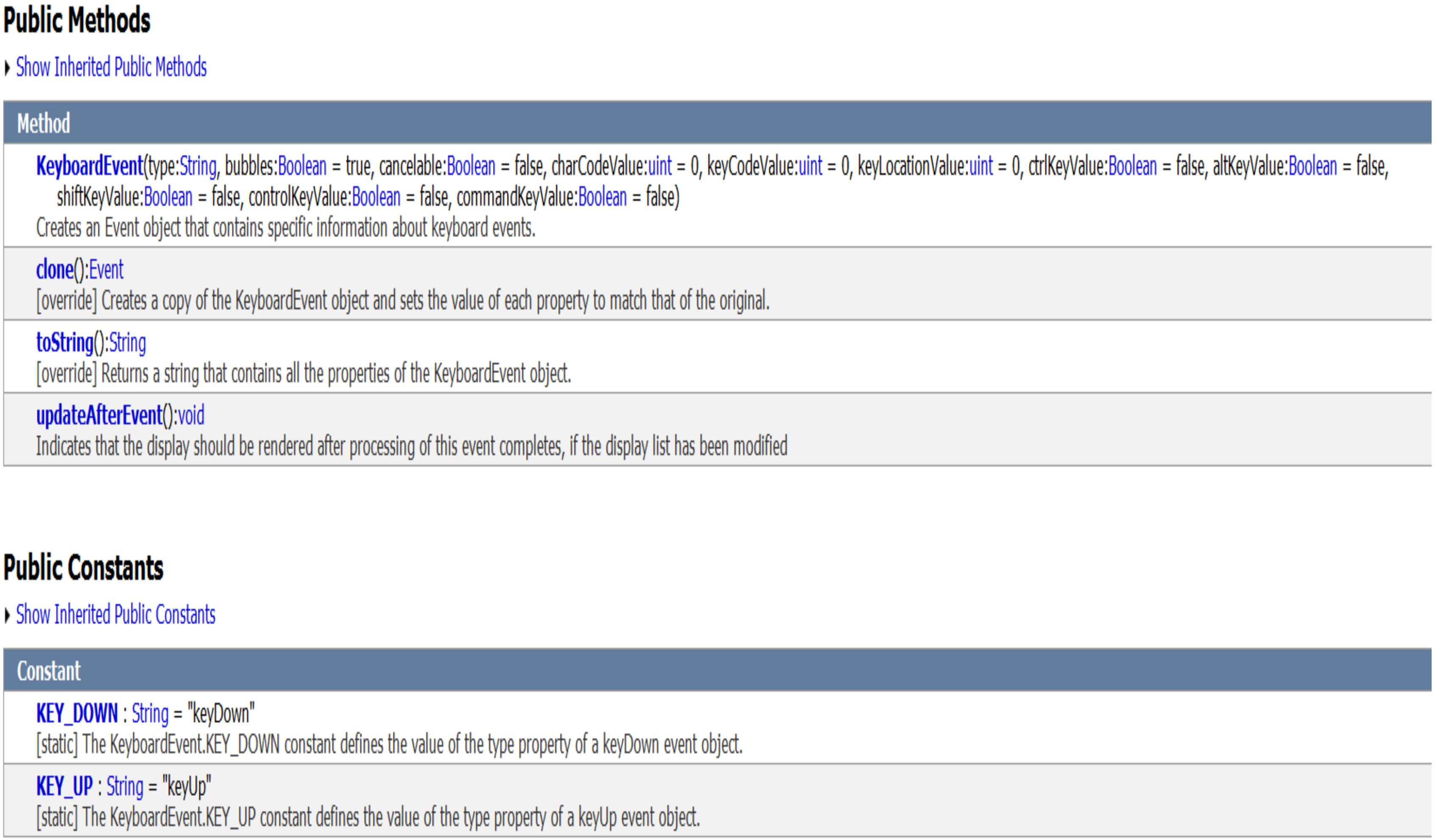


Public Properties

▸ Show Inherited Public Properties

Property
altKey : Boolean Indicates whether the Alt key is active (true) or inactive (false) on Windows; indicates whether the Option key is active on Mac OS.
charCode : uint Contains the character code value of the key pressed or released.
commandKey : Boolean Indicates whether the Command key is active (true) or inactive (false).
controlKey : Boolean Indicates whether the Control key is active (true) or inactive (false).
ctrlKey : Boolean On Windows and Linux, indicates whether the Ctrl key is active (true) or inactive (false); On Mac OS, indicates whether either the Ctrl key or the Command key is active.
keyCode : uint The key code value of the key pressed or released.
keyLocation : uint Indicates the location of the key on the keyboard.
shiftKey : Boolean Indicates whether the Shift key modifier is active (true) or inactive (false).

KeyboardEvent Class



Public Methods

▸ Show Inherited Public Methods

Method
KeyboardEvent(type:String, bubbles:Boolean = true, cancelable:Boolean = false, charCodeValue:uint = 0, keyCodeValue:uint = 0, keyLocationValue:uint = 0, ctrlKeyValue:Boolean = false, altKeyValue:Boolean = false, shiftKeyValue:Boolean = false, controlKeyValue:Boolean = false, commandKeyValue:Boolean = false) Creates an Event object that contains specific information about keyboard events.
clone():Event [override] Creates a copy of the KeyboardEvent object and sets the value of each property to match that of the original.
toString():String [override] Returns a string that contains all the properties of the KeyboardEvent object.
updateAfterEvent():void Indicates that the display should be rendered after processing of this event completes, if the display list has been modified

Public Constants

▸ Show Inherited Public Constants

Constant
KEY_DOWN : String = "keyDown" [static] The KeyboardEvent.KEY_DOWN constant defines the value of the type property of a keyDown event object.
KEY_UP : String = "keyUp" [static] The KeyboardEvent.KEY_UP constant defines the value of the type property of a keyUp event object.

keyCode VS charCode

- You can access the keyCode and charCode properties of a keyboard event to determine what key was pressed and then trigger other actions.
- The keyCode property is a numeric value that corresponds to the value of a key on the keyboard.
- The charCode property is the numeric value of that key in the current character set.

keyCode VS charCode

The primary difference between the key code and character code values:

- key code value represents a particular key on the keyboard

_Eg:_ ***The "1" on a keypad is different than the "1" in the top row***
The "1" in the top row and "!" key are the same

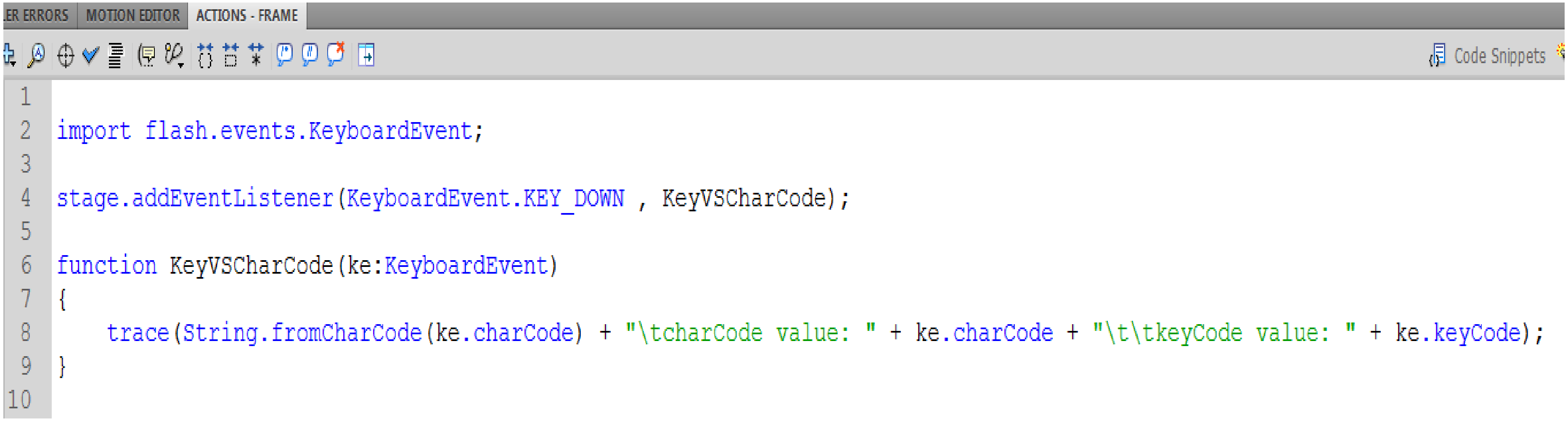
- The character code value represents a particular character

Eg: **The R and r characters are different**

Note: For the mappings between keys and their character code values in ASCII, see the ***flash.ui.keyboard*** class in the ActionScript language reference. ***(Highly Recommended)***

keyCode VS charCode

Eg:



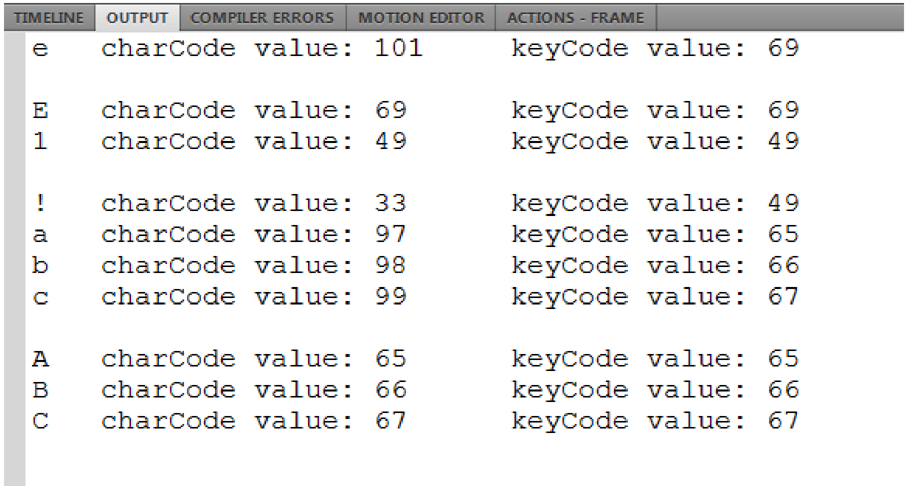
```

import flash.events.KeyboardEvent;

stage.addEventListener(KeyboardEvent.KEY_DOWN , KeyVSCharCode);

function KeyVSCharCode(ke:KeyboardEvent)
{
    trace(String.fromCharCode(ke.charCode) + "\tcharCode value: " + ke.charCode + "\t\tkeyCode value: " + ke.keyCode);
}

```



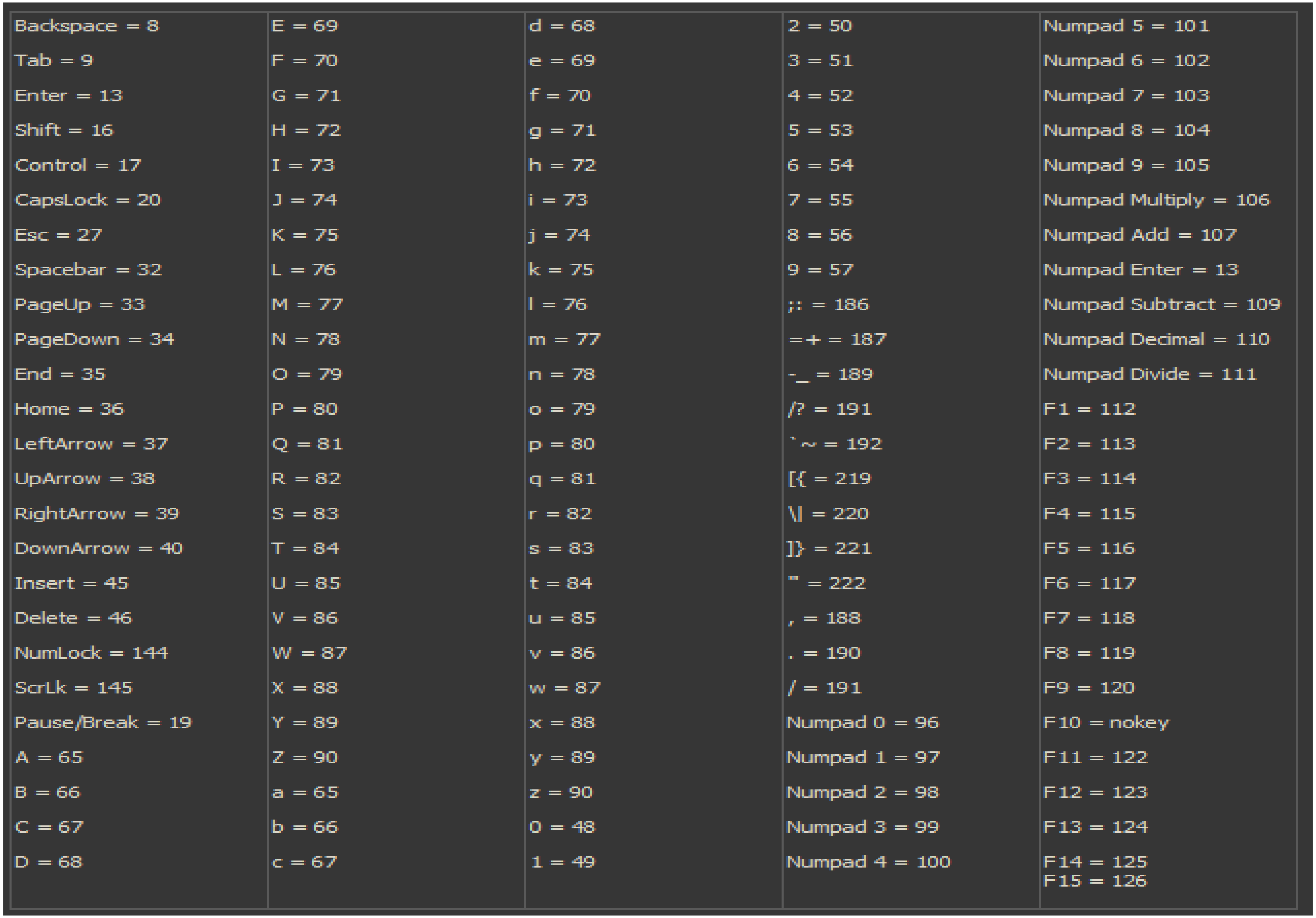
```
e    charCode value: 101      keyCode value: 69

E    charCode value: 69       keyCode value: 69
1    charCode value: 49       keyCode value: 49

!    charCode value: 33       keyCode value: 49
a    charCode value: 97       keyCode value: 65
b    charCode value: 98       keyCode value: 66
c    charCode value: 99       keyCode value: 67

A    charCode value: 65       keyCode value: 65
B    charCode value: 66       keyCode value: 66
C    charCode value: 67       keyCode value: 67
```

Key code table



Backspace = 8	E = 69	d = 68	2 = 50	Numpad 5 = 101
Tab = 9	F = 70	e = 69	3 = 51	Numpad 6 = 102
Enter = 13	G = 71	f = 70	4 = 52	Numpad 7 = 103
Shift = 16	H = 72	g = 71	5 = 53	Numpad 8 = 104
Control = 17	I = 73	h = 72	6 = 54	Numpad 9 = 105
CapsLock = 20	J = 74	i = 73	7 = 55	Numpad Multiply = 106
Esc = 27	K = 75	j = 74	8 = 56	Numpad Add = 107
Spacebar = 32	L = 76	k = 75	9 = 57	Numpad Enter = 13
PageUp = 33	M = 77	l = 76	;: = 186	Numpad Subtract = 109
PageDown = 34	N = 78	m = 77	=+ = 187	Numpad Decimal = 110
End = 35	O = 79	n = 78	-_ = 189	Numpad Divide = 111
Home = 36	P = 80	o = 79	/? = 191	F1 = 112
LeftArrow = 37	Q = 81	p = 80	`~ = 192	F2 = 113
UpArrow = 38	R = 82	q = 81	[{ = 219	F3 = 114
RightArrow = 39	S = 83	r = 82	| = 220	F4 = 115
DownArrow = 40	T = 84	s = 83	]} = 221	F5 = 116
Insert = 45	U = 85	t = 84	'" = 222	F6 = 117
Delete = 46	V = 86	u = 85	, = 188	F7 = 118
NumLock = 144	W = 87	v = 86	. = 190	F8 = 119
ScrLk = 145	X = 88	w = 87	/ = 191	F9 = 120
Pause/Break = 19	Y = 89	x = 88	Numpad 0 = 96	F10 = nokey
A = 65	Z = 90	y = 89	Numpad 1 = 97	F11 = 122
B = 66	a = 65	z = 90	Numpad 2 = 98	F12 = 123
C = 67	b = 66	0 = 48	Numpad 3 = 99	F13 = 124

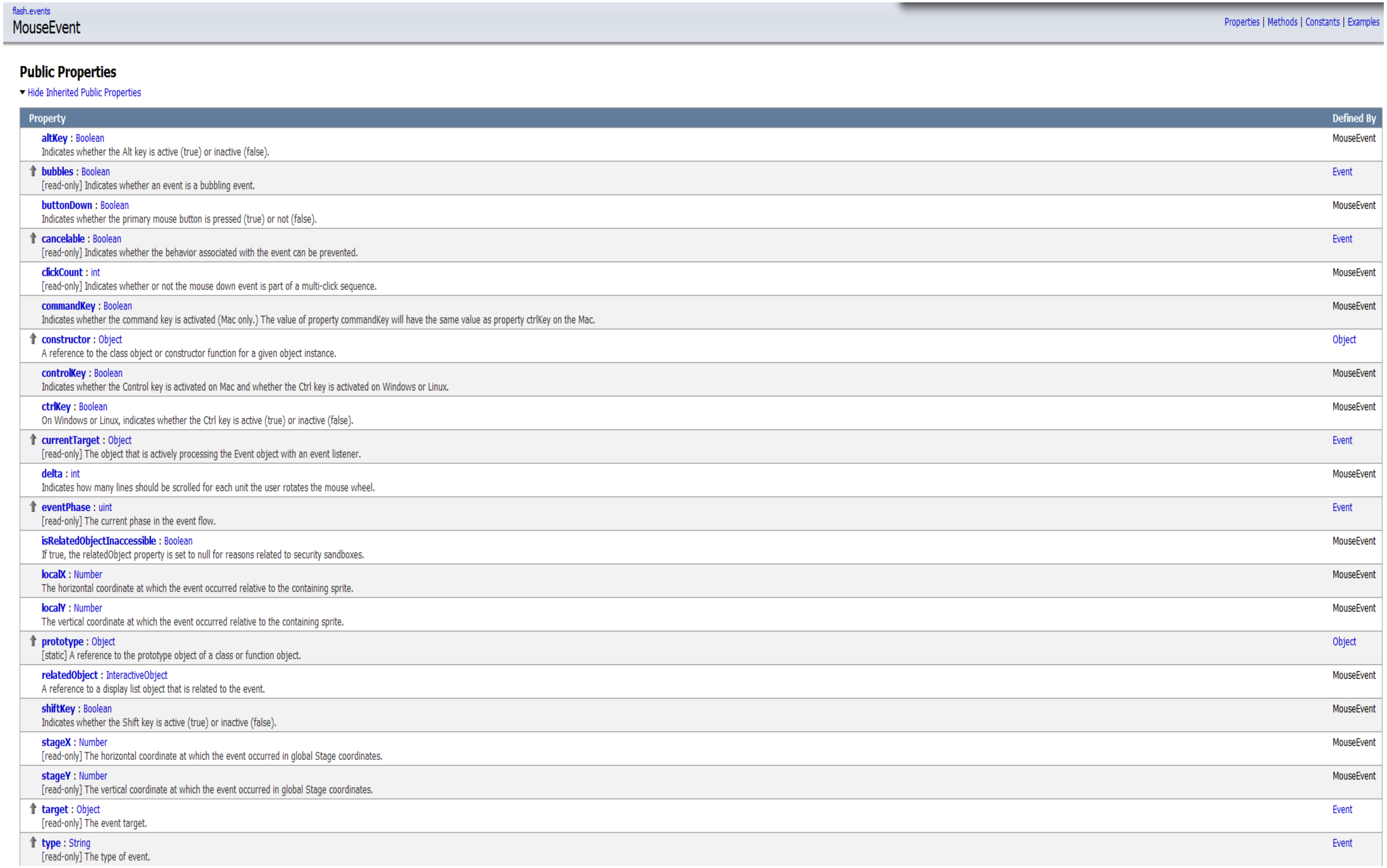
D = 68	c = 67	1 = 49	Numpad 4 = 100	F14 = 125
				F15 = 126

Mouse Input

Mouse events

- Mouse clicks/movement create mouse events that can be used to trigger interactive functionality.
- You can add an event listener to the Stage to listen for mouse events that occur anywhere within the SWF file.
- You can also add event listeners to objects on the Stage (for example, Sprite or MovieClip); these listeners are triggered when the object is clicked, or in other words in focus.

MouseEvent Class Properties



flash.events

MouseEvent

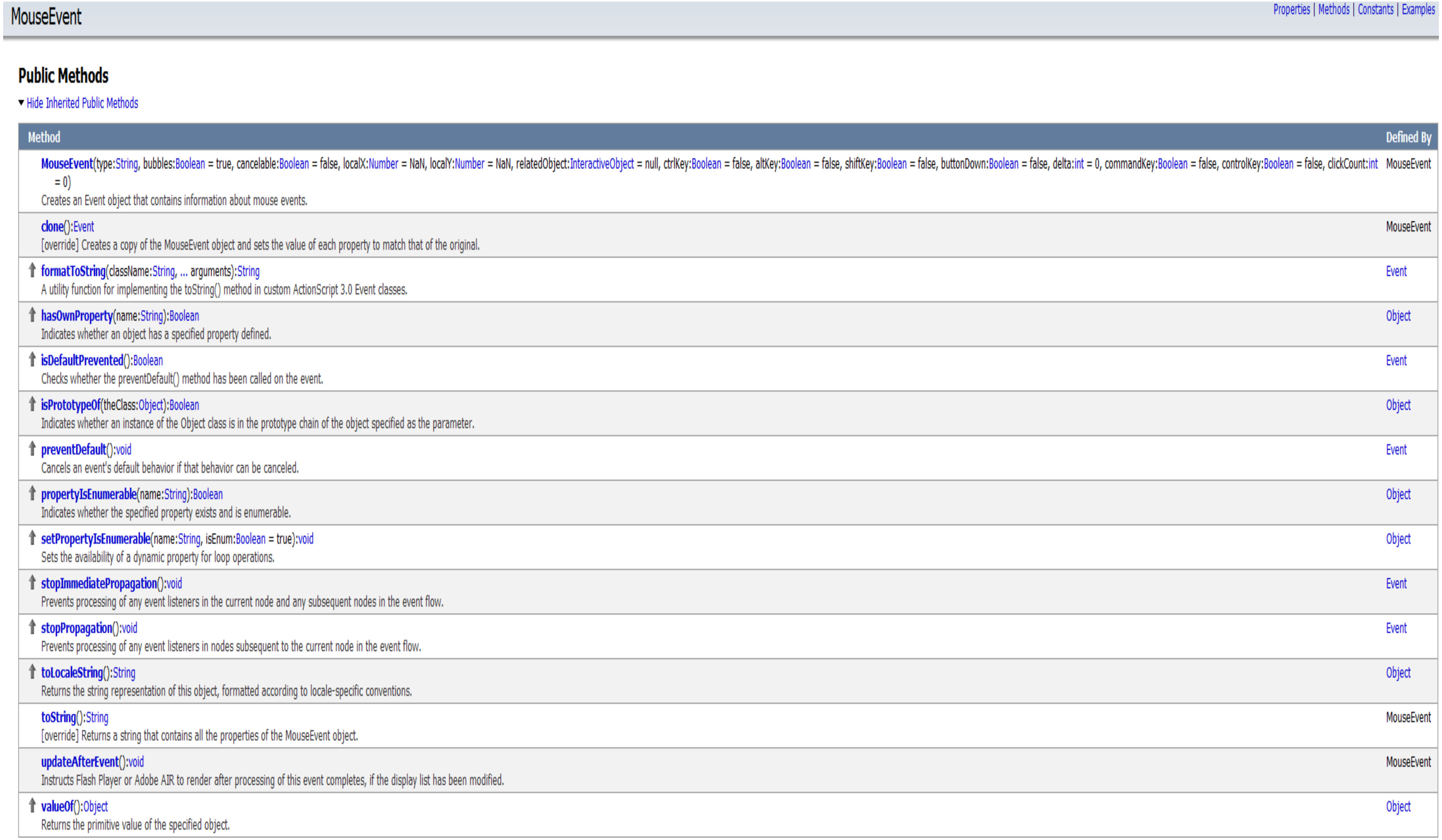
Properties | Methods | Constants | Examples

Public Properties

▼ Hide Inherited Public Properties

Property	Defined By
altKey : Boolean Indicates whether the Alt key is active (true) or inactive (false).	MouseEvent
↑ **bubbles** : Boolean [read-only] Indicates whether an event is a bubbling event.	Event
buttonDown : Boolean Indicates whether the primary mouse button is pressed (true) or not (false).	MouseEvent
↑ **cancelable** : Boolean [read-only] Indicates whether the behavior associated with the event can be prevented.	Event
clickCount : int [read-only] Indicates whether or not the mouse down event is part of a multi-click sequence.	MouseEvent
commandKey : Boolean Indicates whether the command key is activated (Mac only.) The value of property commandKey will have the same value as property ctrlKey on the Mac.	MouseEvent
↑ **constructor** : Object A reference to the class object or constructor function for a given object instance.	Object
controlKey : Boolean Indicates whether the Control key is activated on Mac and whether the Ctrl key is activated on Windows or Linux.	MouseEvent
ctrlKey : Boolean On Windows or Linux, indicates whether the Ctrl key is active (true) or inactive (false).	MouseEvent
↑ **currentTarget** : Object [read-only] The object that is actively processing the Event object with an event listener.	Event
delta : int Indicates how many lines should be scrolled for each unit the user rotates the mouse wheel.	MouseEvent
↑ **eventPhase** : uint [read-only] The current phase in the event flow.	Event
isRelatedObjectInaccessible : Boolean If true, the relatedObject property is set to null for reasons related to security sandboxes.	MouseEvent
localX : Number The horizontal coordinate at which the event occurred relative to the containing sprite.	MouseEvent
localY : Number The vertical coordinate at which the event occurred relative to the containing sprite.	MouseEvent
↑ **prototype** : Object [static] A reference to the prototype object of a class or function object.	Object
relatedObject : InteractiveObject A reference to a display list object that is related to the event.	MouseEvent
shiftKey : Boolean Indicates whether the Shift key is active (true) or inactive (false).	MouseEvent
stageX : Number [read-only] The horizontal coordinate at which the event occurred in global Stage coordinates.	MouseEvent
stageY : Number [read-only] The vertical coordinate at which the event occurred in global Stage coordinates.	MouseEvent
↑ **target** : Object [read-only] The event target.	Event
↑ **type** : String [read-only] The type of event.	Event

MouseEvent Class Methods



MouseEvent

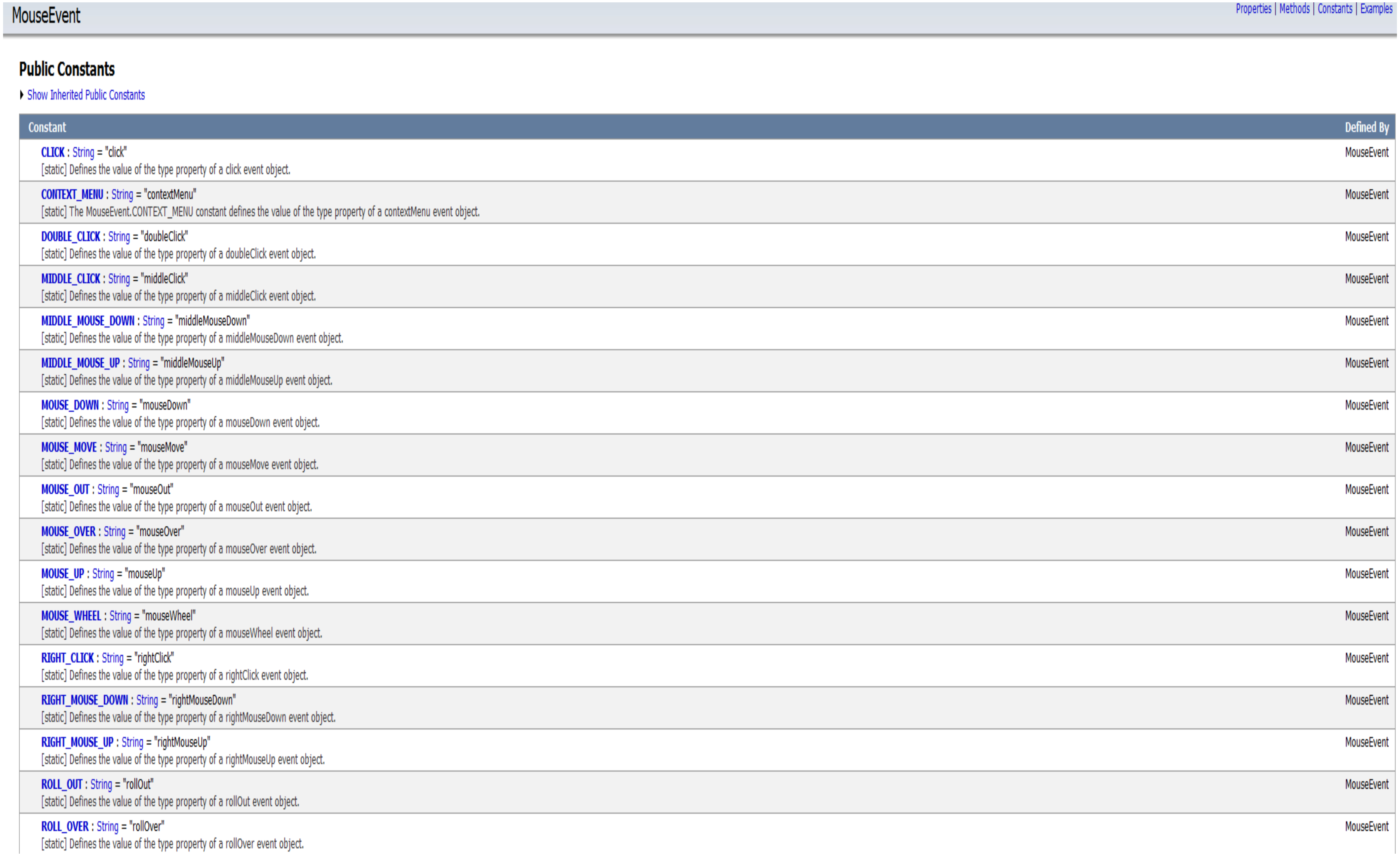
Properties | Methods | Constants | Examples

Public Methods

▼ Hide Inherited Public Methods

	Method	Defined By
	MouseEvent(type:String, bubbles:Boolean = true, cancelable:Boolean = false, localX:Number = NaN, localY:Number = NaN, relatedObject:InteractiveObject = null, ctrlKey:Boolean = false, altKey:Boolean = false, shiftKey:Boolean = false, buttonDown:Boolean = false, delta:int = 0, commandKey:Boolean = false, controlKey:Boolean = false, clickCount:int = 0) Creates an Event object that contains information about mouse events.	MouseEvent
	clone():Event [override] Creates a copy of the MouseEvent object and sets the value of each property to match that of the original.	MouseEvent
↑	**formatToString**(className:String, ... arguments):String A utility function for implementing the toString() method in custom ActionScript 3.0 Event classes.	Event
↑	**hasOwnProperty**(name:String):Boolean Indicates whether an object has a specified property defined.	Object
↑	**isDefaultPrevented**():Boolean Checks whether the preventDefault() method has been called on the event.	Event
↑	**isPrototypeOf**(theClass:Object):Boolean Indicates whether an instance of the Object class is in the prototype chain of the object specified as the parameter.	Object
↑	**preventDefault**():void Cancels an event's default behavior if that behavior can be canceled.	Event
↑	**propertyIsEnumerable**(name:String):Boolean Indicates whether the specified property exists and is enumerable.	Object
↑	**setPropertyIsEnumerable**(name:String, isEnum:Boolean = true):void Sets the availability of a dynamic property for loop operations.	Object
↑	**stopImmediatePropagation**():void Prevents processing of any event listeners in the current node and any subsequent nodes in the event flow.	Event
↑	**stopPropagation**():void Prevents processing of any event listeners in nodes subsequent to the current node in the event flow.	Event
↑	**toLocaleString**():String Returns the string representation of this object, formatted according to locale-specific conventions.	Object
	toString():String [override] Returns a string that contains all the properties of the MouseEvent object.	MouseEvent
	updateAfterEvent():void Instructs Flash Player or Adobe AIR to render after processing of this event completes, if the display list has been modified.	MouseEvent
↑	**valueOf**():Object Returns the primitive value of the specified object.	Object

MouseEvent Class Constants



MouseEvent

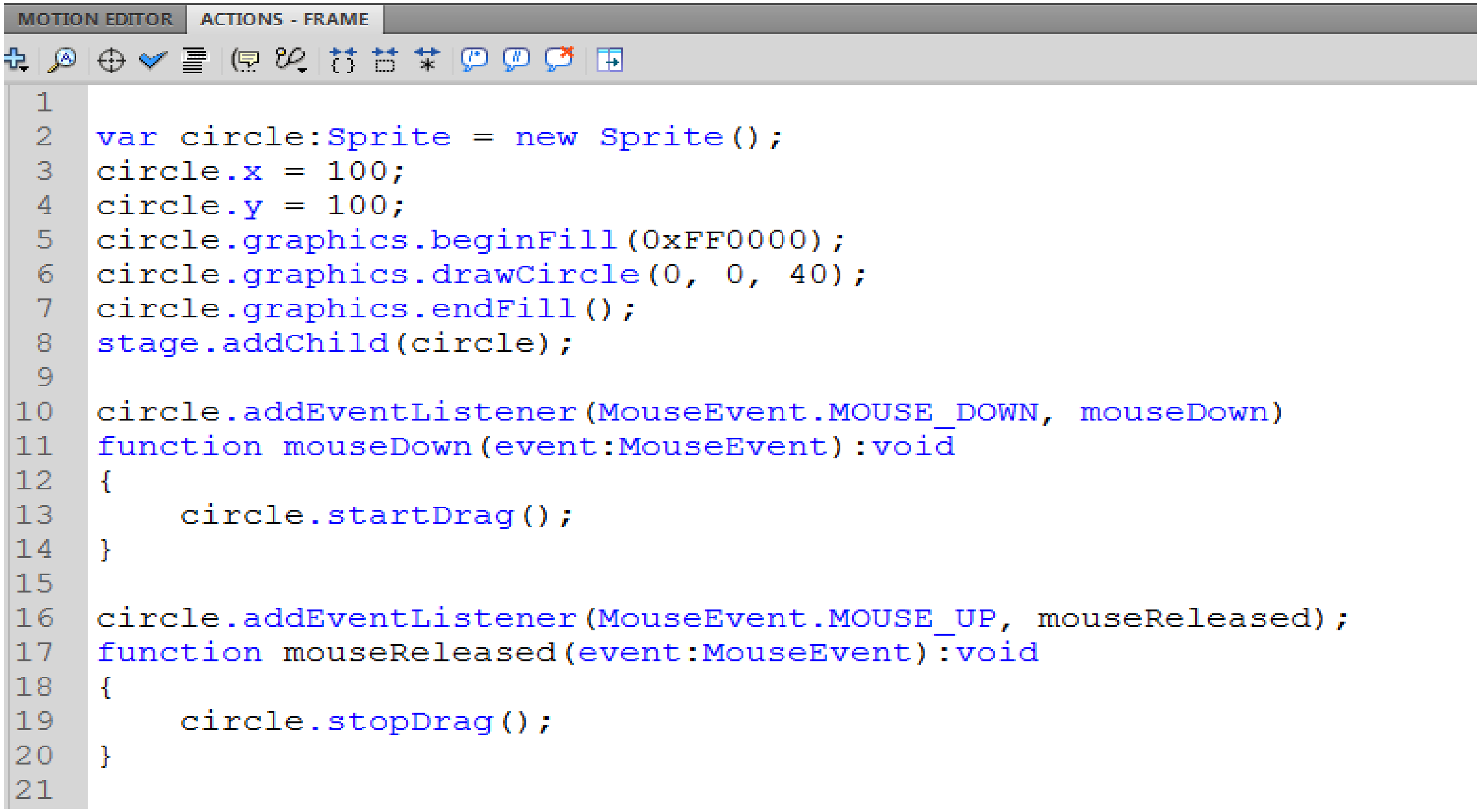
Public Constants

▸ Show Inherited Public Constants

Constant	Defined By
CLICK : String = "click" [static] Defines the value of the type property of a click event object.	MouseEvent
CONTEXT_MENU : String = "contextMenu" [static] The MouseEvent.CONTEXT_MENU constant defines the value of the type property of a contextMenu event object.	MouseEvent
DOUBLE_CLICK : String = "doubleClick" [static] Defines the value of the type property of a doubleClick event object.	MouseEvent
MIDDLE_CLICK : String = "middleClick" [static] Defines the value of the type property of a middleClick event object.	MouseEvent
MIDDLE_MOUSE_DOWN : String = "middleMouseDown" [static] Defines the value of the type property of a middleMouseDown event object.	MouseEvent
MIDDLE_MOUSE_UP : String = "middleMouseUp" [static] Defines the value of the type property of a middleMouseUp event object.	MouseEvent
MOUSE_DOWN : String = "mouseDown" [static] Defines the value of the type property of a mouseDown event object.	MouseEvent
MOUSE_MOVE : String = "mouseMove" [static] Defines the value of the type property of a mouseMove event object.	MouseEvent
MOUSE_OUT : String = "mouseOut" [static] Defines the value of the type property of a mouseOut event object.	MouseEvent
MOUSE_OVER : String = "mouseOver" [static] Defines the value of the type property of a mouseOver event object.	MouseEvent
MOUSE_UP : String = "mouseUp" [static] Defines the value of the type property of a mouseUp event object.	MouseEvent
MOUSE_WHEEL : String = "mouseWheel" [static] Defines the value of the type property of a mouseWheel event object.	MouseEvent
RIGHT_CLICK : String = "rightClick" [static] Defines the value of the type property of a rightClick event object.	MouseEvent
RIGHT_MOUSE_DOWN : String = "rightMouseDown" [static] Defines the value of the type property of a rightMouseDown event object.	MouseEvent
RIGHT_MOUSE_UP : String = "rightMouseUp" [static] Defines the value of the type property of a rightMouseUp event object.	MouseEvent
ROLL_OUT : String = "rollOut" [static] Defines the value of the type property of a rollOut event object.	MouseEvent
ROLL_OVER : String = "rollOver" [static] Defines the value of the type property of a rollOver event object.	MouseEvent

Drag-And-Drop

- Drag-and-drop functionality allows users to select an object while pressing the left mouse button, move the object to a new location on the screen, and then drop it at the new location by releasing the left mouse button.



```

var circle:Sprite = new Sprite();
circle.x = 100;
circle.y = 100;
circle.graphics.beginFill(0xFF0000);
circle.graphics.drawCircle(0, 0, 40);
circle.graphics.endFill();
stage.addChild(circle);

circle.addEventListener(MouseEvent.MOUSE_DOWN, mouseDown)
function mouseDown(event:MouseEvent):void
{
    circle.startDrag();
}

circle.addEventListener(MouseEvent.MOUSE_UP, mouseReleased);
function mouseReleased(event:MouseEvent):void
{
    circle.stopDrag();
}

```

Customizing the mouse cursor

- The mouse cursor (mouse pointer) can be hidden or swapped for any display object on the Stage.

  - ➢ To hide the mouse cursor, call the ***Mouse.hide()*** method.

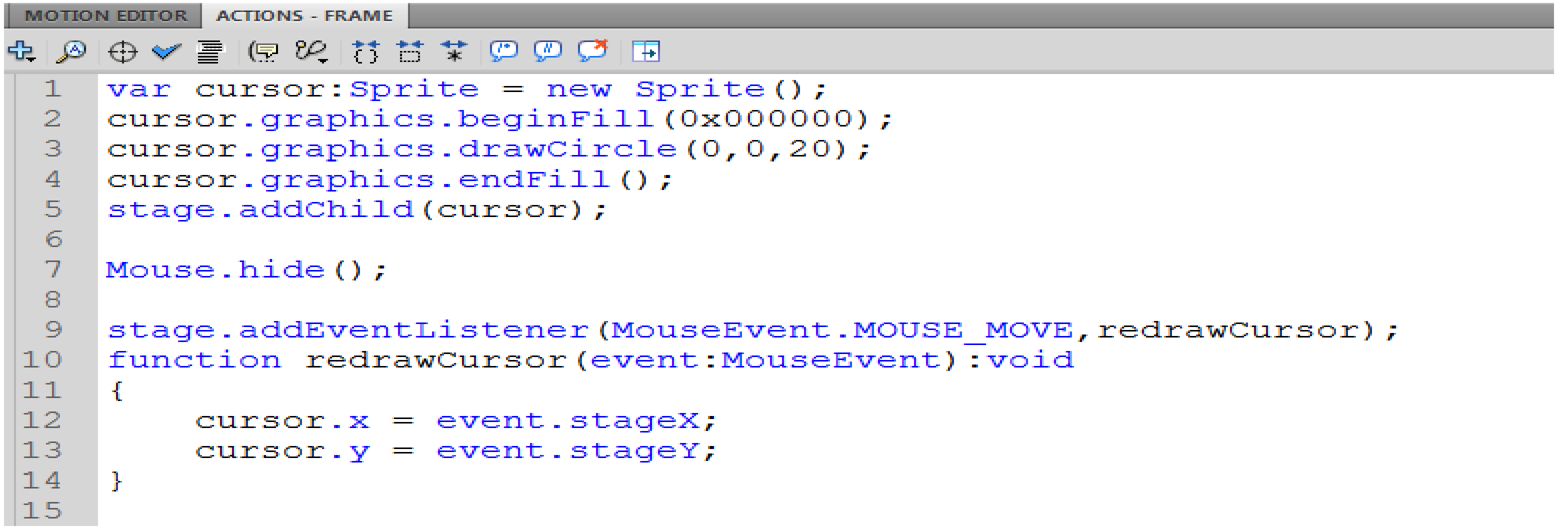
  - ➢ Customize the cursor shape by:

    - ❖ Either listening to the Stage for the **MouseEvent.MOUSE_MOVE** event, and setting the coordinates of a display object (your custom cursor) to the **stageX** and **stageY** properties of the mouse event.

    - ❖ Or, call the display object's **startDrag()** method that will set his drag Boolean to true.

Customizing the mouse cursor

- **Example:**



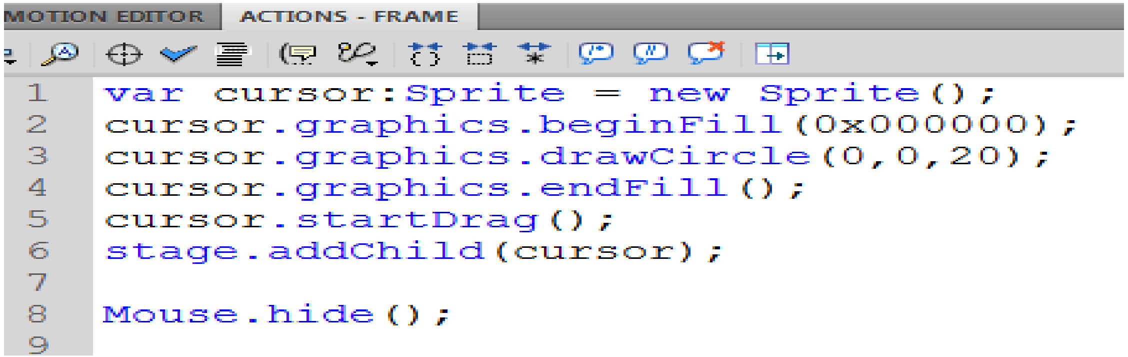
```
var cursor:Sprite = new Sprite();
cursor.graphics.beginFill(0x000000);
cursor.graphics.drawCircle(0,0,20);
cursor.graphics.endFill();
stage.addChild(cursor);

Mouse.hide();

stage.addEventListener(MouseEvent.MOUSE_MOVE,redrawCursor);
function redrawCursor(event:MouseEvent):void
{
	cursor.x = event.stageX;
	cursor.y = event.stageY;
}

```

Or



```
var cursor:Sprite = new Sprite();
cursor.graphics.beginFill(0x000000);
cursor.graphics.drawCircle(0,0,20);
cursor.graphics.endFill();
cursor.startDrag();
stage.addChild(cursor);

Mouse.hide();

```

Simple Input Manager

Why?

- So we have all those event listeners available in the language, why create a manager?
    - ➢ Common place where input is handled instead of having a keyboard event listener in every object that we interact with.
    - ➢ More flexibility and functionality (check if triggered, check if released …)
    - ➢ Test the manager and never worry about input anymore.
    - ➢ Not having to remove event listeners in multiple places.

Working with Booleans

- Every key should have an **IsPressed** Boolean.

- That Boolean will be true when the user is pressing the key and false when he’s not (or in our case, when the key is up).

= 1 or true

= 0 or false

- Since we have more than one key (duh), we will need an array to represent all the Booleans.

- At this point, we just need to check if the key’s Boolean is true or false to know if it is pressed or not.

Is Triggered / Is Released

- To check if something is triggered or is released, we will need data from the previous frame. But Why??!!
- For a key to be triggered, it should be pressed at the current frame but not pressed in the frame before (otherwise it is pressed).
- So in order to do that we will need an array that stores the previous frame's data.
- That array can be the **WasPressed** Boolean array.

Is Triggered / Is Released / Is Pressed

Previous Frame		Current Frame		
NP	+	P	=	Is Triggered
P	+	NP	=	Is Released
		P	=	Is Pressed

Cycle

	Enter Frame					Exit Frame	
Initially	0	0	No	No	No	0	0
Key Pressed	1	0	Yes	Yes	No	1	1
Key Still Pressed	1	1	Yes	No	No	1	1
Key Released	0	1	No	No	Yes	0	0

0	Current Frame	🟥	Check If Pressed
0	Previous Frame	🟨	Check If Triggered
		🟦	Check If Released

The End ☺