Autodesk®
Scaleform®

액션스크립트 확장 레퍼런스

이 문서는 Scaleform 4.2 에서 사용 가용한 액션스크립트 확장을 설명한다.

집필: Artem Bolgar

버전: 4.03

최종 편집: 2012 년 9 월 17 일

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.2

Disclaimer

연락처:

문서	액션스크립트 확장 레퍼런스
주소	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
홈페이지	www.scaleform.com
이메일	info@scaleform.com
직통전화	(301) 446-3200
팩스	(301) 446-3199

목차

1 서문

Autodesk Scaleform SDK 는 가볍고 고성능이며 미디어가 잔뜩 들어간 Flash® 벡터 그래픽 엔진으로 특히 콘솔 및 PC 게임 개발자를 위한 클린룸 구현을 기반으로 제작되었다. Scaleform 는 플래시 스튜디오처럼 확장성 및 편리한 개발이 증명된 비주얼 제작 툴을 통해서 최신 게임 개발 욕구를 만족시켜주는데, 이때, 첨단 하드웨어 그래픽 가속도 지원한다.

플래시가 주로 게임이나 어플리케이션이 아닌 웹을 위해 고안되었기 때문에 포커스 처리, 마우스 지원, 텍스트 필드 지원 및 IME 입력 처리와 같은 몇몇 분야에선 제한된 기능을 갖고 있다. Scaleform 는 액션스크립트 클래스에 "확장"이라는 개념을 추가함으로써 해당 영역에 있는 기본 기능을 향상시킨다. Scaleform Player 에서 확장 지원을 사용하려면 `_global.gfxExtensions` 변수를 true 로 설정할 필요가 있다.

```
_global.gfxExtensions = true;
```

대부분의 경우 개발자는 이러한 설명을 FLA 내 최초 프레임에 추가하여 확장을 사용하길 원한다. 이러한 변수를 설정하지 않으면 Scaleform Player 는 확장으로의 모든 참조를 무시하고 최대 수준의 플래시 호환성을 얻기 위해 노력한다.

개발자는 이 문서에서 기술한 확장 기능이 표준 플래시 플레이어에선 작동하지 않는다는 점을 이해해야 하는데 이는 Scaleform 에서만 구현되기 때문이다. 각 확장과 관련된 사항은 Scaleform 버전 번호와 어떤 확장을 추가했는지를 플레이어 SDK 의 릴리즈 번호로 확인하는 것이다. 확장은 이전 버전의 Scaleform Player 에선 작동하지 않는다.

Scaleform 이 확장 API 를 지원하고 서로 다른 릴리즈를 통해서도 일관적으로 유지될 수 있도록 최선을 다 하지만 차후 Scaleform 릴리즈에서 확장 API 의 변경, 명칭 변경 또는 제거를 할 권리를 갖는다. 중요한 변화가 발생하는 경우 주요 릴리즈에서 반영할 것이며 최대한 빨리 공지할 것이다.

2 마우스 클래스 확장

표준 마우스 클래스 메소드 전체를 지원하는 것뿐만 아니라 Scaleform 는 또한 다중 마우스 커서의 추적과 RIGHT, MIDDLE 및 기타 마우스 버튼 (현재 최대 16 개)의 탐지를 가능하게 하는 확장을 도입한다. 이 장에선 우선 기본 마우스 기능을 다루고 그 후 Scaleform 마우스 확장에 관한 보다 자세한 사항을 제공할 것이다. 마우스 객체에 대한 보다 자세한 사항은 Flash Mouse Class 문서를 참고하라.

2.1 다중 커서 지원

Wii 게임 콘솔과 같은 몇몇 플랫폼은 하나 이상의 마우스 커서 지원을 필요로 한다. Scaleform 3.2 는 최대 4 개의 마우스 커서를 지원한다. GFx::Movie::SetMouseCursorCount(unsigned n)는 커서의 수를 설정한다. 매개변수로서 GFx::MouseEvent 이벤트가 있는 GFx::Movie::HandleEvent 는 Scaleform 코어에 마우스 이벤트를 공급하기 위해 사용해야 한다. GFx::MouseEvent 와 GFx::MouseCursorEvent 는 0 기반의 마우스 인덱스를 지정하는 추가 매개변수가 있다.

2.2 버튼 이벤트 확장

onRollOver, onRollOut, onDragOver, onDragOut, onPress, onRelease, onReleaseOutside 버튼 이벤트는 Scaleform 확장을 켰을 때 하나 혹은 두개의 매개변수를 수신한다.
첫 번째 매개변수는 이벤트를 유발하는 0 을 기준으로 하는 마우스 인덱스다. 그러므로 onPress 이벤트가 인덱스가 1 인 마우스 커서를 통해 발생되는 경우 매개변수는 1 값을 갖고있다.

```
mc.onPress = function(mouseIdx:Number)
{
       if (mouseIndex == 0)
              . . .
       else if (mouseIndex == 1)
              . . .
      . . .
}
```

onPress 와 onRelease 의 두 번째 파라미터는 이벤트의 발생 원인이 마우스/커서인지 키보드인지를 표시하는 숫자 프로퍼티입니다. 값은 키보드일 경우 1, 마우스/커서일 경우 0 입니다. 이 프로퍼티는 이벤트의 원인을 판별하는 데 매우 유용합니다.

```
mc.onPress = function(mouseIdx:Number, keyboardOrMouse:Number)
{
	if (keyboardOrMouse == 0)
		. . .
	else
		. . .
      . . .
}
```

두 번째 매개변수는 onRollOver/Out 및 onDragOver/Out 이벤트에 대한 선택사항이다. 이 매개변수는 내포한 rollover/dragover 이벤트를 동일한 문자에 대해 정의한다. 만약 이 매개변수가 함수 핸들러에 대해 선언되지 않은 경우라면 onRollOver/onRollOut 와 onDragOver/onDragOut 이벤트 쌍은 단 한 번만 작동한다. onRollOver/onDragOver 는 첫 번째 커서가 문자에 대해 roll over 되면 작동하며 onRollOut/onDragOut 은 마지막 커서가 나갈 때 작동한다.

만약 두번째 매게변수가 이 핸들러 용으로 선언되었다면, 내포된 onRollOver/onRollOut 과 onDragOver/onDragOut 이벤트가 (각각의 마우스 커서마다 분리 되어)생성되며 인접한 0 을 기준으로 하는 인덱스를 나타낸다. 초기 onRollOver/onDragOver 이벤트는 두 번째 매개변수로 0 을 갖는다. 만약 두 번째 커서가 동일한 문자에 대해 작동하는 경우 두 번째 onRollOver/onRollOut 이벤트는 1 로 설정된 두 번째 매개변수와 작동한다. 만약 커서가 문자에서 나가는 경우 onRollOut/onDragOut 은 1 로 설정된 두 번째 매개변수와 작동한다. 마지막 onRollOut/onDragOut 이벤트는 0 으로 설정된 두 번째 매개변수와 작동한다.

따라서, 두 번째 매개변수를 선언하면 onRollOver/onRollOut 과 onDragOver/onDragOut 이벤트가 어떻게 발생될 지를 바꿀 수 있다. 두 번째 매개변수를 선언하는 경우 내포된 이벤트를 관리하는 것은 사용자 책임이다.

```
mc.onRollOver = function(mouseIdx:Number, nestingIdx:Number)
{
	// do roll over animation only if nestingIdx == 0
	if (nestingIdx == 0)
		doOverAnimation();
      . . .
}
mc.onRollOut = function(mouseIdx:Number, nestingIdx:Number)
{
```

```
        // do roll out animation only if nestingIdx == 0
        if (nestingIdx == 0)
                doOutAnimation();
        . . .
}
```

On(rollOver), on(rollOut) 등 구형 스타일의 이벤트가 추가 매개변수와 함께 제공되므로 이벤트를 유발하는 마우스 커서를 구별할 방법이 없다는 점에 유의한다. 그러므로 다중 마우스 커서와 함께 사용하는 것은 권장하지 않는다.

onPress, onRelease 등의 이벤트에 대해 마우스 왼쪽 버튼 이외의 버튼을 할당할 수 있도록 다음과 같은 보조 버튼 이벤트가 새롭게 추가되었습니다.

- onPressAux
- onReleaseAux
- onReleaseOutsideAux
- onDragOver
- onDragOut

이러한 보조 이벤트 핸들러는 이벤트 대상에 정의되었을 때 실행되며, index != 0 인 버튼에 대해서만 실행됩니다(index 값이 0 이면 마우스 왼쪽 버튼). 표준 핸들러는 보조 이벤트 핸들러와 상관없이 항상 마우스 왼쪽 버튼에 대해서만 실행됩니다. 또한, 이전 섹션에서 정의한 것처럼, 이러한 핸들러는 동일한 함수 원형을 핸들러의 표준 사본으로 가집니다. 그러나, 보조 이벤트 핸들러는 버튼 인덱스에 대해 추가 파라미터를 제공합니다.

```
mc.onPressAux = function(mouseIdx:Number, keyboardOrMouse:Number, buttonIdx:Number)
{
        if (buttonIdx == 1)  // 오른쪽 마우스 버튼
                . . .
        . . .
}
```

2.3 Mouse 클래스 이벤트

액션스크립트에서 마우스 리스너는 마우스 움직임, 왼쪽 마우스 버튼 클릭 및 마우스 휠 이벤트에 대한 통지를 수신하기 위해 설치할 수 있다. 플래시에서 이러한 이벤트는 일반적으로

Mouse.addListener 메소드를 사용해서 처리되며 다음과 같은 메소드를 구현하기 위해 리스너 객체를 설치한다.

- onMouseMove = function() { }
- onMouseDown = function() { }
- onMouseUp = function() { }
- onMouseWheel = function(delta : Number, targetPath : String) { }

리스너 객체를 설치한 후, 적절한 메소드가 해당 이벤트가 발생할 때마다 호출된다. 플래시에서 onMouseDown 과 onMouseUp 메소드는 LEFT 마우스 버튼에서만 호출되며 기타 마우스 버튼 이벤트를 수신하기 위해 어떠한 공개 인터페이스를 제공하지 않는다

이러한 제한사항 (및 다중 마우스 커서를 지원하기 위한)을 넘어서기 위해 Scaleform 는 이들 메소드 호출을 확장하는데 그 방법은 _global.gfxExtensions 변수를 true 로 설정하고 추가 전달인자를넘기는 것이다. 새로운 함수 사용은 다음과 같다.

- onMouseDown = function(button : Number, [targetPath : String],
  [mouseIdx : Number], [x : Number], [y : Number],
  [dblClick : Boolean]) { }
- onMouseUp = function(button : Number, [targetPath : String],
  [mouseIdx : Number], [x : Number], [y : Number]) { }

할당된 리스너가 최소 하나의 추가 전달인자를 취할 때 Scaleform 는 RIGHT, MIDDLE 및 기타 마우스 버튼에 대한 리스너 메소드를 호출하여 이러한 이벤트를 탐지할 수 있도록 한다. LEFT 값은 1, RIGHT 는 2, MIDDLE 은 3 의 값을 갖는다. 호환성을 위해 만약 함수가 전달인자를 갖지 않도록 선언되면 LEFT 마우스 버튼에 대해서만 호출되며 이는 플래시와 동일하다. 아래의 내용은 이러한 핸들러를 어떻게 설치할 수 있는 지를 보여준다.

```
var mouseListener:Object = new Object;

mouseListener.onMouseDown = function(button, target)
{
   trace("mouseDown - button '" + button +
        "'  target = '" + target + "'");
}
mouseListener.onMouseUp = function(button, target)
{
   trace("mouseUp - button '" + button +
        "'  target = '" + target + "'");
}
```

```
Mouse.addListener(mouseListener);
```

새로운 마우스 다운/업 핸들러는 최소 두 개의 전달인자(버튼과 타겟)을 취한다. 첫 번째 인자인 버튼은 누른 마우스 버튼을 나타낸다. 버튼을 `Mouse["LEFT"], Mouse["RIGHT"]` 및 `Mouse["MIDDLE"]` 확장 상수와 비교함으로써 해석이 가능하다 (`Mouse["LEFT"]`는 액션스크립트 2.0 컴파일러를 중지시키기 위해 `Mouse.LEFT` 대신 사용된다). 두 번째 사항인 타겟은 경로를 마우스를 눌렀을 때 마우스 커서 아래에 있는 최상위 객체에 경로를 제공한다. 이러한 경로를 사용할 수 없을 때 'undefined'라고 정의할 수 있다.

플래시에서 모든 추가 인자는 'undefined'가 되는데 이는 왼쪽 마우스 버튼에 대해서만 핸들러를 호출하기 때문이다. 'undefined' 값을 LEFT 마우스 버튼 값으로 해석하여 플래시 플레이어와의 호환성을 보장할 수 있다.

다중 마우스 커서를 지원하기 위해 각 핸들러에 대한 추가 매개변수가 있다.

- `onMouseMove = function([mouseIdx : Number], [x : Number], [y : Number]) { }`
- `onMouseDown = function(button : Number, [targetPath : String], [mouseIdx : Number], [x : Number], [y : Number], [dblClick : Boolean]) { }`
- `onMouseUp = function(button : Number, [targetPath : String], [mouseIdx : Number], [x : Number], [y : Number]) { }`
- `onMouseWheel = function(delta : Number, targetPath : String, [mouseIdx : Number], [x : Number], [y : Number]) { }`

매개변수 `mouseIdx`는 이벤트를 발생시킨 마우스 커서의 0을 기준으로 한 인덱스를 담고있다. x와 y 매개변수는 마우스 커서의 좌표를 갖고 있다. (_root 좌표계 공간).
`onMouseDown` 핸들러의 `dblClick` 매개변수는 더블클릭을 나타낸다. 이 경우에는 `true` 값을 가질 것이다.

2.4 마우스 커서 변경

플래시에는 세 가지 마우스 커서 형태가 있다.

- *Arrow* - 커서가 대부분의 일반적인 객체 위에 있을 때 사용
- *Hand* - useHandCursor 속성이 true 일 때 버튼 및 링크 위에 사용
- *I-Beam* - 커서가 텍스트 필드 위에 있을 때 사용

커서 변경이 발생할 때 플래시가 이를 탐지하는 쉬운 방법을 제공하지 않는데 반해서, Scaleform 는 C++ API 와 액션스크립트 마우스 클래스 확장을 지원하며 이를 통해 커스텀 커서를 관리할 수 있다. C++ 측면에서 `GFx::StateBag::SetUserEventHandler` 함수를 사용해서 마우스 커서 변화가 필요할 때마다 이를 통보하는 이벤트 핸들러 함수를 설치할 수 있다. 이 핸들러를 사용해서 FxPlayer.cpp 에서 기술한 바와 같이 게임 엔진이 그린 커서 이미지 또는 시스템 커서를 변경할 수 있다. 바꿔 말하면 액션스크립트에서 완벽히 구현한 커스텀 커서를 구현하도록 선택할 수 있다. 이를 어떻게 그만 하느냐에 대한 예제가 SDK 에 탑재된 *Mouse.fla/swf* 에 포함되어있다.

마우스 샘플은 커스터마이즈가 가능한 마우스 커서를 'Cursor_M'이라고 하는 스테이지 상의 단순한 무비 클립 객체로 구현한다. 커서는 "hand", "arrow" 및 'ibeam" 프레임 라벨로 정의되는 여러 프레임을 갖는다. 커서 이미지는 단순히 타임라인을 이 프레임 중 하나로 이동시킴으로써 변경이 가능하다. 마우스 움직임이 탐지될 때마다 무비 클립의 _x 및 _y 좌표계가 마우스를 따라가기 위해 수정된다.

아래에 있는 객체를 기준으로 커서 타입을 변경하기 위해 Scaleform 확장 `Mouse.setCursorType` 함수를 오버라이드한다. 이 함수는 cursorType 와 mouseIndex 인자와 함께 호출되며 이는 커서 변경이 필요한 경우마다 발생함으로써, 이러한 변화를 탐지할 수 있게 해준다. 마우스 샘플은 이 샘플을 해석하는 데 있어 cursorType 을 마우스 상수 중 하나 (Mouse["ARROW"], Mouse["HAND"], Mouse["IBEAM"])와 비교하며 그에 맞춰 커서 프레임을 변경한다.

다음 코드는 마우스 샘플에서 다중 커스텀 커서를 처리한다. 보다 자세한 내용은 실제 샘플을 참고하도록 한다.

```
_global.gfxExtensions = 1;
// Hide system cursor.
Mouse.hide();

var mouseListener:Object = new Object;
mouseListener.onMouseMove = function(mouseIdx, x,y)
{
    if (mouseIdx == undefined)
   mouseIdx = 0;
    if (x == undefined)
    {
   x = _xmouse;
      y = _ymouse;
```

```
    }
    var cmc = eval("_root.Cursor_M"+(mouseIdx+1));

    cmc._x = x;
    cmc._y = y;
}
Mouse.addListener(mouseListener);

Mouse["setCursorType"] = function(cursorType, mouseIdx)
{
   if (mouseIdx == undefined)
        mouseIdx = 0;
   var cmc = eval("_root.Cursor_M"+(mouseIdx+1));
   switch(cursorType)
   {
      case Mouse["HAND"]:
         cmc.Cursor.gotoAndPlay("hand");
         break;
      case Mouse["ARROW"]:
         cmc.Cursor.gotoAndPlay("arrow");
         break;
      case Mouse["IBEAM"]:
         cmc.Cursor.gotoAndPlay("ibeam");
         break;
      default: return;
   }
}

// Override Mouse.show and Mouse.hide functions so
// that they affect our cursor.
ASSetPropFlags(Mouse, "show,hide", 0, 7);

Mouse.show = function(mouseIdx)
{
    if (mouseIdx == undefined)
       mouseIdx = 0;
    var cmc = eval("_root.Cursor_M"+(mouseIdx+1));
     cmc._visible = true;
}
Mouse.hide = function(mouseIdx)
{
    if (mouseIdx == undefined)
       mouseIdx = 0;
    var cmc = eval("_root.Cursor_M"+(mouseIdx+1));
     cmc._visible = false;
}
```

2.5 마우스 클래스 액션스크립트 확장

Mouse 액션스크립트 클래스는 몇가 지 정적 메소드와 속성을 추가하는 형태로 GFx 에서 확장되었다. 액션스크립트 1.0 에선 이러한 확장을 `Mouse.setCursorType(…)`형태로 직접 참조하는

것이 가능했다. 액션스크립트 2.0 은 허용범위가 좁아서 이러한 호출에 대해서 투덜거리게 될 것이다. 액션스크립트 2.0 컴파일러의 입을 닥치게 하려면 다음과 같은 대체 접근 문장을 사용해야 한다: `Mouse["setCursorType"](..)`.

getButtonsState() 정적 메소드

```
public function getButtonsState(mouseIndex : Number) : Number
```

Scaleform version: 2.2

마우스 버튼의 상태를 반환한다. 반환값은 비트 마스크로서 0 을 기준으로 하는 마우스 버튼 인덱스의 비트 인덱스와 동일하다. 해당 비트가 설정되면 버튼이 눌린 것이다.

Parameters

`mouseIndex : Number` – 0 을 기준으로 하는 마우스 인덱스

getTopMostEntity() 정적 메소드

```
public function getTopMostEntity([mouseIndex : Number]) : Object
public function getTopMostEntity(testAll : Boolean,
                                 [mouseIndex : Number]) : Object
public function getTopMostEntity(x : Number, y : Number) : Object
public function getTopMostEntity(x : Number, y : Number, testAll: Boolean) : Object
```

Scaleform version: 1.2.34, 2.2 이후 다중커서 지원

이 정적 메소드는 마우스 커서 밑의 타겟문자 또는 특정 좌표계에서 타겟문자를 반환한다. 이 메소드는 (onPress, onMouseDown, onRollOver 같은)버튼 핸들러가 설정 되었을때와 안되어있을 때의 값이 다를 수 있다. 이러한 구분은 마우스 이벤트를 처리하는 핸들러가 없는 문자를 채우는 데 유용할 수 있다.

이 메소드는 `MovieClip.hitTest` 와 반대다. 왜냐하면 `hitTest` 가 `x`, `y` 좌표가 특정 객체 내에 있는지를 체크하고, `getTopMostEntity` 가 지정된 `x`, `y` 좌표계에서 정의된 실제 객체를 반환하기 때문이다. 따라서 `Mouse.getTopMostEntity(x, y).hitTest(x, y, true) == true`.

Parameters

`mouseIndex : Number` – 0 을 기준으로 하는 마우스 인덱스

`testAll : Boolean` – 버튼 핸들러가 있는(false) 경우와 없는(true) 경우를 가리킨다. 이 값이 없으면 `getTopMostEntity` 는 이를 'true'로 가정한다.

`x : Number, y : Number` – 문자를 찾기 위한 대체 좌표계

사용예:

```
var listenerObj = new Object;
listenerObj.onMouseMove = function()
{
   var target = Mouse["getTopMostEntity"]();
   trace("Mouse moved, target = "+target);
}
Mouse.addListener(listenerObj);
var target = Mouse["getTopMostEntity"](480, 10);
trace("The character at the (480, 10) is " + target);
```

getPosition 정적 메소드

```
public function getPosition(mouseIndex : Number) : flash.geom.Point
```

Scaleform version: 2.2

이 메소드는 _root 좌표 공간에서 마우스 커서에 대응하는 좌표계를 반환한다. 반환값은 flash.geom.Point 의 인스턴스다.

Parameters

`mouseIndex : Number` – 0 을 기준으로 하는의 마우스 인덱스

setCursorType() 정적 메소드

```
public function setCursorType(cursorType : Number, [mouseIndex : Number]) : void
```

Scaleform version: 1.2.32

이 정적 메소드는 매개변수 `cursorType` 에 따라 마우스 커서를 변환한다. 세부사항은 "마우스 커서 변경"을 참고한다.

Parameters

`cursorType : Number` – Mouse.ARROW, Mouse.HAND, Mouse.IBEAM 중 하나로 커서형태 지정

`mouseIndex : Number` – 0 을 기준으로 하는의 마우스 인덱스

3 Button 클래스 확장

hitTestDisable 속성

`hitTestDisable:Boolean [read-write]`

Scaleform version: 2.1.51

true 로 설정하면 `MovieClip.hitTest` 함수는 이 버튼을 히트 테스트 시에 무시한다. 또한, 기타 모든 마우스 이벤트가 버튼에 전달되지 않는다.
기본값은 false 다.

관련자료:

```
TextField.hitTestDisable
MovieClip.hitTestDisable
```

topmostLevel 속성

`topmostLevel:Boolean [read-write]`

Scaleform version: 2.1.50

속성이 true 로 설정되면 이 문자는 깊이 값에 상관없이 모든 문자들의 상단에 출력된다. 이는 모든 레벨에서 커서를 객체 위에 그리는 커스텀 마우스 커서를 구현하는 데 유용하다
기본값은 false 다.

몇몇의 문자들이 "topmostLevel " 로 마킹된 경우, 그리기에 대한 순서는 다음과 같다:

• Scaleform 3.0.71 까지의 버전들에서는 "topmostLevel " 로 마크되어 있는 문자들의 그리기 순서는 이 속성을 true 로 설정하는가 그렇지 않은가에 따라 달려있다. (그러므로 "topmostLevel" 로 설정된 문자는 첫 번째로 그려질 것임.)
• Scaleform 3.0.72 버전 부터의 그리기 순서는 문자의 가장 윗부분을 마킹하지 않아도 동일하게 순서대로 그릴 수 있다. 예: 만약 ObjectA가 ObjectB의 아래에 그려졌다면 "topmostLevel"속성을

true로 설정하는 순서에 관계없이 ObjectA가 가장 윗부분으로 마킹이 되어있어도 ObjectB 아래에 위치한다.

주석: 문자를 "topmostLevel"로 설정되면 swapDepth 액션스크립트는 이 문자에 아무런 영향을 미치지 않는다.

관련자료:

```
TextField.topmostLevel
MovieClip.topmostLevel
```

focusGroupMask 프로퍼티

focusGroupMask : Number

Scaleform version: 3.3.84

이 프로퍼티는 스테이지 캐릭터와 **모든** 자식에게 비트마스크를 설정합니다. 이 비트마스크는 캐릭터에 focus group 소유권을 할당하며, 이는 비트마스크에 표시된 controller 만이 focus 를 캐릭터 내부로 이동시킬 수 있음을 의미합니다. Focus group 은 `setController` 과 FocusGroup 확장 메소드를 통해서 controller 와 연결할 수 있습니다.

예를 들어, "button1"이 controller 0 과 1 을 통해서만 focus 를 맞출 수 있다고 가정해 보겠습니다. 이러한 동작을 얻기 위해서는 focus group 과 controller 를 연결해야 합니다.

```
Selection.setControllerFocusGroup(0, 0);
Selection.setControllerFocusGroup(1, 1);

button1.focusGroupMask = 0x1; // bit 0 – focus group 0
movieclip2.focusGroupMask = 0x1 | 0x2 // bits 0 and 1 – focus groups 0 and 1
```

"focusGroupMask" 비트마스크는 부모 movieclip 으로 설정할 수 있습니다. 이것은 모든 자식에게 마스크 값을 전달합니다.

4 MovieClip 클래스 확장

hitTest () 메소드

```
public hitTest(x:Number, y:Number, [shapeFlag:Boolean],
            [ignoreInvisibleChildren:Boolean]):Boolean
```

Scaleform version: 2.1.51

표준 3 개짜리 전달인자 hitTest 는 확장된 부울 인자를 하나 더 가지는데, 이는 보이지 않는 자식 클립을 무시할 것인지를 나타낸다. hitTest 의 기본 플래시 속성은 x, y 좌표계에서의 점이 자식 클립 (예, _visible 속성이 false 인 경우 _alpha 속성이 0 으로 설정된 자식 클립은 invisible 로 취급되지 않는다)에 속한 경우라도 "true"를 반환한다.

Parameters

`ignoreInvisibleChildren:Boolean` – 안 보이는 모든 자식 클립을 무시하려면 true 로 설정해야 한다.

hitTestDisable 속성

```
hitTestDisable:Boolean  [read-write]
```

Scaleform version: 1.2.32

true 로 설정하면 `MovieClip.hitTest` 함수는 히트 테스트 탐지 시 이 무비 클립을 무시한다. 또한, 기타 모든 마우스 이벤트가 무비 클립으로 전달되지 않는다.
기본값은 false 다.

관련자료:

`TextField.hitTestDisable`
`Button.hitTestDisable`

noAdvance 속성

```
noAdvance : Boolean
```

Scaleform version: 2.1.52

true 로 설정되면 이 무비 클립 및 모든 자식의 Advance 를 off 시킨다. 이 기능은 성능향상을 위해 사용이 가능하다. 플래시는 무비 클립을 항상 advance 시킴을 기억하라 (타임라인 애니메이션 실행, 프레임의 액션스크립트 코드 등). 그러므로 이 속성을 "true"로 설정하면 Scaleform 와 플래시가 서로 행동이 달라 질 수 있다.

관련자료:

```
_global.noInvisibleAdvance
```

topmostLevel 속성

```
topmostLevel:Boolean  [read-write]
```

Scaleform version: 2.1.50

이 속성을 true 로 설정하면 이 문자는 그 깊이에 상관 없이 다른 모든 글자 앞에 표시된다. 이는 모든 레벨에 있는 객체 위에 커서를 그려야 하는 커스텀 마우스 커서를 구현할 때 유용하다. 기본값은 false 다.

몇몇의 문자들이 "topmostLevel " 로 마킹된 경우, 그리기에 대한 순서는 다음과 같다:

• Scaleform 3.0.71 까지의 버전들에서는 "topmostLevel "로 마크되어 있는 문자들의 그리기 순서는 이 속성을 true 로 설정하는가 그렇지 않은가에 따라 달려있다. (그러므로 "topmostLevel" 로 설정된 문자는 첫 번째로 그려질 것임.)
• Scaleform 3.0.72 버전 부터의 그리기 순서는 문자의 가장 윗부분을 마킹하지 않아도 동일하게 순서대로 그릴 수 있다. 예: 만약 ObjectA가 ObjectB의 아래에 그려졌다면 "topmostLevel"속성을 true로 설정하는 순서에 관계없이 ObjectA가 가장 윗부분으로 마킹이 되어있어도 ObjectB 아래에 위치한다.

주석: 문자를 "topmostLevel"로 설정되면 swapDepth 액션스크립트는 이 문자에 아무런 영향을 미치지 않는다.

관련자료:

```
TextField.topmostLevel
Button.topmostLevel
```

rendererString 속성

```
rendererString:String  [read-write]
```

Scaleform version: 2.2.55

이 속성은 모든 MovieClip 인스턴스에 대해 액션스크립트에서 렌더러로 커스텀 명령을 전송하도록 한다. 속성이 설정되면 문자열 값은 사용자 데이터 형태로 렌더러에 전송된다.

기본값은 없다.

예:

```
myMovieInstance.rendererString = “SHADER_Blur”
```

rendererFloat 속성

```
rendererFloat:Number  [read-write]
```

Scaleform version: 2.2.55

이 속성은 MovieClip 인스턴스에 대해 액션스크립에서 커스텀 명령을 렌더러로 전송하도록 한다. 속성이 설정되면 Float 값은 사용자 데이터 형태로 렌더러에 전송된다.
기본값은 없다.

예:

```
myMovieInstance.rendererFloat = 4; // eg: C++ enum value
```

disableBatching 속성

```
disableBatching:Boolean
```

Scaleform version: 4.0.17

이 속성은 사용자 그리기를 위한 메쉬 생성 배치를 비활성화합니다.

focusGroupMask 프로퍼티

focusGroupMask : Number

Scaleform version: 3.3.84

이 프로퍼티는 스테이지 캐릭터와 **모든** 자식에게 비트마스크를 설정합니다. 이 비트마스크는 캐릭터에 focus group 소유권을 할당하며, 이는 비트마스크에 표시된 controller 만이 focus 를 캐릭터 내부로 이동시킬 수 있음을 의미합니다. Focus group 은 `setController` 과 FocusGroup 확장 메소드를 통해서 controller 와 연결할 수 있습니다.

예를 들어, "button1"이 controller 0 과 1 을 통해서만 focus 를 맞출 수 있다고 가정해 보겠습니다. 이러한 동작을 얻기 위해서는 focus group 과 controller 를 연결해야 합니다.

```
Selection.setControllerFocusGroup(0, 0);
Selection.setControllerFocusGroup(1, 1);

button1.focusGroupMask = 0x1; // bit 0 – focus group 0
movieclip2.focusGroupMask = 0x1 | 0x2 // bits 0 and 1 – focus groups 0 and 1
```

"focusGroupMask" 비트마스크는 부모 movieclip 으로 설정할 수 있습니다. 이것은 모든 자식에게 마스크 값을 전달합니다.

5 Netstream 클래스 확장

Scaleform 는 NetStream 클래스에 자막 조작과 비디오 파일에 내장될 수 있는 추가적인 오디오 트랙을 처리하는 기능을 추가하였다.

onMetaData 이벤트 핸들러

```
onMetaData = function(info:Object) { }
```

Scaleform version: 3.0.63

이 이벤트 핸들러는 현재 재생중인 비디오 파일로부터 정보를 받으며, 또한 2 가지의 추가적인 객체 속성을 포함하고 있다. 이들 속성들은 비디오 파일에 인코딩 되어 있는 자막과 오디오 트랙에 관한 정보를 얻어내는데 쓰인다.

`onMetaData` 이벤트 핸들러에 전달된 객체는 2 가지 추가적인 읽기전용 속성값이 있다.

1. `audioTracks` – 비디오 파일에 인코딩 되어있는 오디오 트랙 배열

   `audioTracks` 배열의 각 객체는 다음과 같은 속성을 가진다.

   `channelsNumber` – 오디오 트랙 채널 수(1-mono, 2- stereo, 6 – 5.1 surround sound)

   `totalSamples` – 트랙에 있는 사운드 샘플 수

   `trackIndex` – 트랙 인덱스 번호. 이 속성은 `NetStream.audioTrack` 속성에 대입함으로써 오디오 트랙 선택에 사용된다. (다음 예 참고)

   `sampleRate` – 사운드 샘플레이트.

2. `subtitleTracksNumber` – 비디오 파일에서 사용 가능한 자막트랙의 개수

audioTrack 속성

```
audioTrack:Number  [read-write]
```

Scaleform version: 3.0.63

이 속성은 현재 재생중인 비디오 파일에 인코딩 된 오디오 트랙에 set/get 하기 위해 사용된다.

사용예:

```
var ns:NetStream = new NetStream(nc);
var audioTracks;

ns.onMetaData = function(info:Object)
{
        audioTracks = info.audioTracks;
}

if (audioTracks != undefined && audioTracks.length > 0)
        ns.audioTrack = audioTracks[0].trackIndex;
```

subtitleTrack 속성

```
subtitleTrack:Number  [read-write]
```

Scaleform version: 3.0.63

이 속성은 현재 자막 트랙을 set 하거나 얻어오도록 해준다. 자막 설정을 끄려면 0 으로 설정한다.

onSubtitle 이벤트 핸들러

```
onSubtitle = function(msg:String) {}
```

Scaleform version: 3.0.63

자막 메시지 출력이 준비되었을 때 호출되는 이벤트 핸들러

예:

```
var ns:NetStream = new NetStream(nc);
var subtitlesNumber = 0;

ns.onMetaData = function(info:Object)
{
        subtitlesNumber = info. subtitleTracksNumber;
 }

 ns.onSubtitle = function(msg:String)
 {
        sbTextField.text = msg;
 }
if (subtitlesNumber > 0)
      ns.subtitleTrack = 1;
```

setNumberOfFramePools 함수

```
public function setNumberOfFramePools(pools : Number) : Void
```

Scaleform version: 3.0.68

이 함수는 내부의 비디오 버퍼들의 수를 설정한다. 비디오 버퍼는 비디오 디코딩 출력을 저장하는데 사용되고, "frame pool"이라고 불린다. CPU의 디코딩 로드가 불안정할 때, frame pool의 수가 많을수록 부드러운 재생에 도움이 된다. 디폴트 값은 1이다.

이 메서드는 비디오가 시작되기 전에 호출되어야 한다.(즉, NetStream.play() 호출 이전).

setReloadThresholdTime 함수

```
public function setReloadThresholdTime(reloadTime : Number) : Void
```

Scaleform version: 3.0.68

본 함수는 다시 로딩할 때의 시기를 초 별로 설정한다. Scaleform Video 라이브러리는 입력 버퍼의 데이터 크기가 재 로딩 threshold(임계점) 보다 작을 경우 다음 파일 읽기 요청을 한다. 이 threshold(임계점) 시간은 비디오 파일의 bitrate에 따라 자동적으로 결정되고, reload 되는 시간이 설정된다. reload 시간의 디폴트 값은0.8(초)이다.

비디오가 재생되고 있고 그 사이 게임 데이터를 읽어 들이는 동안 이 함수를 사용해서 찾기 요청의 수를 줄일 수 있다. 예를 들자면, Netstream 버퍼 시간을 2초로 설정하고 (NetStream.setBufferTime() 메서드) threshold(임계점) 시간을 1초로 설정할 때, 그 게임 데이터는 1초 동안의 시간 동안 끊김 없이 읽어 들일 수 있는 것이다. 하지만 게임 데이터 읽기는 재 로딩 타이밍이 발생할 때까지 혹은 그 이전에 완료되어야 한다. 게임 데이터가 클 때, 그것은 큰 덩어리들을 읽도록 되어있을 것이다. 그리고 그 게임 데이터를 읽는 것은 로딩 타이밍 이전에 끝나지 않는다면, 그 영화의 재생은 비디오 버퍼가 없어 끊김이 발생할 것이다.

이 메서드는 비디오가 시작되기이전에 호출되어야 한다.(즉, NetStream.play() 호출 이전)

6 Selection 클래스 확장

Selection 클래스는 텍스트 필드, 무비 클립, 버튼 사이에서 입력 포커스를 관리할 수 있게 해주는 것이 가장 중요한 용도다.

Scaleform 는 Selection 클래스의 포커스 기능을 확장한다. 액션스크립트 1.0 에선 `Selection.captureFocus()`에서와 마찬가지로 Selection 확장 기능을 직접 참조하는 게 가능하다. 액션스크립트 2.0 은 허용 범위가 작기 때문에 이러한 호출에 대해 컴파일러가 투덜거리는 경우가 있다. ActonScript 2.0 컴파일러를 닥치게 하려면 다음과 같이 접근한다.
`Selection[“captureFocus”]().`

alwaysEnableArrowKeys 정적 속성

`alwaysEnableArrowKeys : Boolean`

Scaleform version: 1.2.34

이 정적 속성은 `"_focusrect"` 속성이 `false` 인 상태에서도 화살표 키로 포커스를 바꿀 수 있도록 한다 (포커스가 캡처중 일 때만 적용됨). 기본적으로 플래시는 `"_focusrect = false;"`로 노란색 포커스 사각형이 사용불가능 할 경우에는 화살표키로 포커스를 바꾸는 것을 허용하지 않는다.
이러한 속성을 변경하려면 `alwaysEnableArrowKeys` 속성을 `"true"`로 설정한다.

예:

```
Selection["alwaysEnableArrowKeys"] = true;
```

alwaysEnableKeyboardPress 정적 속성

`alwaysEnableKeyboardPress : Boolean`

Scaleform version: 3.0.63

`"_focusrect"` 속성이 false(단, 포커스가 캡처중일때)일 때라도 스페이스나 엔터키를 사용해서 onPress/ onRelease 이벤트를 발생시킴. 플래시는 기본적으로 `"_focusrect = false;"`로 노란색

포커스 사각형이 사용 불가능한 상태에서는 키보드로 버튼을 누를 수 없다. 이를 수정하려면 alwaysEnableKeyboardPress 를 true 로 하면 된다.

예:

```
Selection["alwaysEnableKeyboardPress"] = true;
```

captureFocus()정적 메소드

```
public function captureFocus([doCapture:Boolean, controllerIdx:Number]) : void
```

Scaleform version: 1.2.34

이 정적 메소드는 키보드 포커스를 프로그램을 통해서 캡처하거나 해제할 수 있다.
doCapture 값을 true 로 설정해서 (혹은 매개변수 없이) captureFocus 를 호출하면 키보드 포커스를 얻을 수 있으며 화살표 키로 포커스를 변경할 수 있다. 이는 탭 키를 처음 눌렀을 때와 유사하다
doCapture 값을 false 로 설정해서 captureFocus 를 호출하면 키보드 포커스를 해제하며 포커스 사각형이 on 된 것처럼 작동한다.

Parameters

`doCapture:Boolean` – 옵션, 키보드 포커스를 캡처/해제 할 지를 나타낸다. 만약 이 매개변수가 빠진 경우 captureFocus 는 true 로 설정된다.

`controllerIdx:Number` – 옵션, capture 연산에 사용된 키보드/컨트롤러를 표시합니다. 초기값은 controller 0 입니다.

예:

```
Selection["captureFocus"](); // same as passing "true"
Selection["captureFocus"](false);
```

disableFocusAutoRelease 정적속성

```
disableFocusAutoRelease : Boolean
```

Scaleform version: 1.2.34

이 정적 속성은 마우스 움직임이 키보드 포커스를 해제 하는지를 제어하는 데 사용된다 (플래시에서의 표준 속성). 기본적으로 포커스를 획득하고 노란색 사각형을 표시할 때 (화살표 키로 포커스를 변경하도록 함) 마우스 움직임은 포커스를 해제한다. 이러한 행동을 방지하기 위해 정적 속성을 true 로 설정한다.

예:

```
Selection["disableFocusAutoRelease"] = true;
```

disableFocusKeys 정적 속성

```
disableFocusKeys : Boolean
```

Scaleform version: 2.2.58

이 정적 속성은 모든 포커스 키의 처리를 불가능하게 한다 (TAB, Shift-TAB 및 화살표 키). 그러므로 사용자는 자신의 포커스 키 관리를 구현할 수 있다.

예:

```
Selection["disableFocusKeys"] = true;
```

참고자료

```
moveFocus()
```

disableFocusRolloverEvent 정적 속성

```
disableFocusRolloverEvent : Boolean
```

Scaleform version: 2.0.37

이 정적 속성은 키를 통해 포커스가 변경된 경우 롤오버/아웃을 사용할 수 없게 설정하기 위해 사용한다. 기본적으로 화살표 키를 눌러서 포커스가 변경되면 롤오버/롤아웃 이벤트가 시작한다. 이를 막기 위해 이 정적 속성을 true 로 설정한다.

예:

```
Selection["disableFocusRolloverEvent "] = true;
```

modalClip 정적 속성

```
modalClip : MovieClip
```

Scaleform version: 2.2.58

이 정적 속성은 포커스 관리의 관점에서 "modal" 클립으로 특정 무비 클립을 설정한다. 이는 포커스 키 (TAB, Shift-TAB 및 화살표 키)는 특정 무비 클립 내부에서만 포커스 사각형을 움직이며, 예를 들면 무비 클립의 "탭으로 이동 가능한" 자식 사이만을 움직이기도 한다.

예:

```
Selection["modalClip"] = _root.mc;
...
Selection["modalClip"] = undefined;
```

참고자료

```
disableFocusKeys
moveFocus()
```

moveFocus() static 메소드

```
public function moveFocus(keyToSimmulate : String [, startFromMovie:Object,
                includeFocusEnabledChars : Boolean = false,
                controllerIdx :Number]) : Object
```

Scaleform version: 2.2.58

이 정적 메소드는 포커스 키 (TAB, Shift-TAB 또는 화살표 키) 중 하나를 눌러서 키를 작동시킴으로써 포커스 사각형을 움직이는 데 사용한다. `disableFocusKeys` 와 `modalClip` 속성과 이 방법과의 결합은 커스텀 포커스 관리를 구현하는 데 사용할 수 있다.

Parameters

`keyToSimmulate:String` – "up", "down", "left", "right", "tab", "shifttab"과 같이 작동시킬 키 이름.

`startFromMovie:Object` – 문자를 지정하는 추가 매개변수. 시작점으로 현재 포커스 된 것이 아닌 moveFocus 를 사용할 수 있다. 이 값을 null 이나 undefined 로 설정하면 현재 포커스 문자를 시작점으로 사용한다. 이는 세번째 선택 옵션 지정 시에 유용하다.

`includeFocusEnabledChars:Boolean` –`focusEnabled` 속성이 설정되어있고, `tabEnabled/tabIndex` 속성이 설정되어있을 때 moveFocus 가 문자에 지정되도록 해준다. 이 값이 없거나 0 이면 `tabEnabled/tabIndex` 속성이 지정된 문자들은 포커스 이동에만 사용된다.

`controllerIdx:Number` – 옵션, capture 연산에 사용된 키보드/컨트롤러를 표시합니다. 초기값은 controller 0 입니다.

예:

```
Selection["moveFocus"](“up”);
Selection["moveFocus"](“tab”, _root.mc);
Selection["moveFocus"](“tab”, _root.mc, true);
Selection["moveFocus"](“tab”, null, true);
```

반환값

새로 포커스를 받은 문자를 반환하거나 문자를 찾을 수 없을 때는 undefined 를 반환한다.

참고자료

`disableFocusKeys`
`modalClip`

findFocus() 정적 메소드

```
public function findFocus(keyToSimmulate : String [, parentMovie:Object, loop :
                Boolean, startFromMovie:Object, includeFocusEnabledChars : Boolean,
                controllerIndex : Number]) : Object
```

Scaleform version: 3.3.84

이 정적 메소드는 TAB, Shift + TAB 또는 화살표 키의 키 누름을 시뮬레이션하여 다음 focus 항목을 찾는 데 사용됩니다. 이 메소드는 disableFocusKeys 및 setModalClip/getModalClip 익스텐션의 논리곱과 함께 사용자 정의 focus 관리를 실행하는 데 사용할 수 있습니다.

파라미터

`keyToSimulate:String` – 시뮬레이션할 키 이름("up", "down", "left", "right", "tab", "shift + tab")

`parentMovie` - modal clip 으로 사용되는 movie clip 입니다. focus 항목 검색은 이 clip 의 자식 내에서만 수행할 수 있습니다. null 일 수 있습니다.

`loop` - focus 를 반복적으로 수행하는 Boolean 플래그입니다. 예를 들어, 현재 focus 된 항목이 가장 아래에 있고 키가 "down"이면, findFocus 는 "null"(이 플래그가 "false"일 경우) 또는 focus 가능한 항목 중 가장 적합한 것(플래그가 "true"일 경우)을 반환합니다.

`startFromMovie:Object` – 현재 focus 된 항목 대신 findFocus 에서 시작 지점으로 사용할 수 있는 캐릭터를 지정하는 옵션 파라미터입니다. 이 프로퍼티가 null 또는 정의되지 않은 값을 가질 때는 현재 focus 된 캐릭터가 시작 지점으로 사용됨을 의미합니다.

`includeFocusEnabledChars:Boolean` – tabEnabled/tabIndex 프로퍼티 셋을 가진 캐릭터는 물론 focusEnabled 프로퍼티 셋만 가진 캐릭터 위에 moveFocus 를 허용하는 옵션 플래그입니다. 플래그가 false 로 지정 또는 설정되지 않았다면 tabEnabled/tabIndex 프로퍼티 셋을 가진 캐릭터만 focus 무브먼트에 관여합니다.

`controllerIndex` - focus 를 처리 중인 controller 의 zero base index 입니다. 이것은 focus group 과 함께 다중 controller focus 지원에 사용할 수 있습니다.

예:

```
var a = Selection["findFocus"]("up");
```

반환

캐릭터를 찾지 못하는 경우, focus 하거나 null 값으로 만들 다음 캐릭터를 반환합니다.

참조:

```
disableFocusKeys
setModalClip
getModalClip
```

setModalClip() 정적 메소드

```
public function setModalClip(modalClip : Object, controllerIndex : Number)
```

Scaleform version: 3.3.84

이 정적 메소드는 지정된 movie clip 을 focus 관리를 위해 "modal" clip 으로 설정합니다. 이것은 TAB, Shift + TAB 및 화살표 키가 "TAB 사용이 가능한" 모든 자식의 지정된 movie clip 으로만 focus 를 이동시키게 됨을 말합니다.

파라미터

`modalClip`	– A modal clip.
`controllerIndex`	- controller 의 zero base index 입니다.

getModalClip() 정적 메소드

```
public function getModalClip(controllerIndex : Number) : Object
```

Scaleform version: 3.3.84

이 정적 메소드는 지정된 controller 에 대한 modal clip 을 반환합니다.

파라미터

`controllerIndex`	- controller 의 Zero base index 입니다.

반환

찾을 수 없을 경우 modal clip 이나 undefined 를 반환합니다.

setControllerFocusGroup() 정적 메소드

```
public function setControllerFocusGroup (controllerIndex : Number, focusGroupIdx :
                Number) : Boolean
```

Scaleform version: 3.3.84

이 정적 메소드는 `controllerIndex` 를 통해 표시된 controller 를 focus group 과 연결시킵니다. 기본적으로 모든 controller 는 focus group 0 과 연결되어 있으며, 이것은 모두 동일한 focus 를 사용하고 있음을 의미합니다. 하지만, 각각의 controller 가 개별적인 focus 를 할당할 수 있습니다. 예를 들어, 두 controller 가 반드시 별개의 focus 를 가져야 하는 경우(분할 화면을 사용 시)에 `setControllerFocusGroup (1,1)`는 controller 1 에 대해 분할 focus group 을 생성합니다. `setControllerFocusGroup (1,0)`을 호출하면 controller 0 과 1 이 다시 같은 focus 를 공유하도록 합니다.

파라미터

`controllerIndex` – controller 의 zero base index 입니다.

`focusGroupIdx` – focus group 의 zero base index 입니다.

예:

```
Selection["setControllerFocusGroup"](0,0);
Selection["setControllerFocusGroup"](1,1);
Selection["setControllerFocusGroup"](2,1);
```

반환

성공적일 경우 true 를 반환합니다.

getControllerFocusGroup() 정적 메소드

```
public function getControllerFocusGroup (controllerIndex : Number) : Number
```

Scaleform version: 3.3.84

이 정적 메소드는 지정된 controller 와 연결된 focus group index 를 반환합니다.

파라미터

`controllerIndex` – 실제 controller 의 zero base index 입니다.

반환

focus group 의 zero base index 입니다.

getFocusArray() 정적 메소드

```
public function getFocusArray(mc : Object) : Array
```

Scaleform version: 3.3.84

이 정적 메소드는 현재 지정된 movie clip/button/text field 에 대해 focus 를 가지는 controller index 배열을 반환합니다.

파라미터

`mc` - movie clip, button 또는 text field.

반환

zero base index 배열(숫자).

getFocusBitmask() 정적 메소드

```
public function getFocusBitmask(mc : Object) : Number
```

Scaleform version: 3.3.84

이 정적 메소드는 비트마스크를 반환하며, 이것은 각각의 bit 가 현재 지정된 movieclip/button/textfield 에 대해 focus 를 가지는 controller 를 나타내는 위치입니다.

파라미터

`mc` - movie clip, button 또는 text field.

반환

controller 의 비트마스크.

getControllerMaskByFocusGroup() 정적 메소드

```
public function getControllerMaskByFocusGroup (focusGroupIdx : Number) : Number
```

Scaleform version: 3.3.84

이 정적 메소드는 비트마스크를 반환하며, 이것은 각각의 비트가 지정된 focus group 과 연결된 controller 를 나타내는 위치입니다. `setControllerFocusGroup` 함수를 통해 설정한 상태를 반환합니다.

파라미터

`focusGroupIdx` - focus group 의 index 입니다.

반환

controller 의 비트마스트입니다.

numFocusGroups 프로퍼티

```
numFocusGroups : Number
```

Scaleform version: 3.3.84

`setControllerFocusGroup` 함수 호출을 통해 설정한 focus group 의 수를 반환합니다. focus group 0 과 3 이 활성 상태라면, numFocusGroups 은 2 를 반환합니다.

7 TextField 클래스 확장

Scaleform 는 TextField 클래스에 다수의 확장을 도입했으며 이를 통해 더욱 유연하면서 기능도 많아졌고 플래시 8 에서는 내장 TextField 클래스를 제공하게 되었다. 몇몇 제공된 확장은 플래시 9 에서 동일하거나 유사한 속성으로 재사용되고 있다. 추가적인 확장은 IME, 정렬, 텍스트 크기 및 모양, 텍스트 필터 효과 (그림자, 블러 등), 내장 이미지, 선택 및 클립보드 작동 등에서 더 나은 제어를 제공한다.

7.1 일반 기능

여기에선 일반적인 확장 속성 및 메소드를 설명한다.

autoFit 속성

```
autoFit:Boolean  [read-write]
```

Scaleform version: 2.0.39

폰트 자동 힌팅를 켜고 끈다.
기본값은 false 다.

appendText () 메소드

```
public function appendText(newText:String):void
```

Scaleform version: 2.0.37

`newText` 매개변수를 통해 지정된 문자열을 텍스트 필드의 텍스트 끝에 첨부한다. 이 방법은 텍스트 속성에서 추가 할당 (`+=`)에 비해 훨씬 효율적이며 (예, `my_txt.text += appendingText`) 특히 다량의 텍스트를 포함한 텍스트 필드에서 더욱 그렇다.

주석:이 메서드는 style sheet 에 text field 가 적용되었다면 작동하지 않을 것이다.

Parameters

`newText:String` - 기존문자열에 추가할 문자열

참고자료:

`appendHtml()`

appendHtml () 메소드

```
public function appendHtml(newHtml:String):void
```

Scaleform version: 2.0.37

newHtml 매개변수가 정의하는 HTML 을 텍스트 필드의 텍스트 끝에 추가한다. 이 방법은 `htmlText` 속성에서 추가 할당 (+=)보다 훨씬 효과적이다 (예, `txt.htmlText += moreHtml`). `htmlText` 속성에서 일반적인 +=는 HTML 문자열을 생성하며 새로운 HTML 부분을 첨부한 후 전체 HTML 을 처음부터 파싱한다. 이 함수는 단편적인 HTML 파싱, 즉 newHtml 매개변수로부터 HTML 만을 파싱한다 (이는 newHtml 매개변수 내 HTML 을 제대로 구성해야 하기 때문이며 += 연산자에 대해 필요하지 않다). 특히 다량의 문장을 포함하는 텍스트 필드에서 중요하다.

주석:이 메서드는 style sheet 에 text field 가 적용되었다면 작동하지 않을 것이다.

Parameters

`newHtml:String` - 기존 문자열에 추가할 HTML 문자열

참고자료:

`appendText()`

caretIndex 속성

```
caretIndex:Number  [read-only]
```

Scaleform version: 2.0.37

이 속성은 삽입 포인트 (캐럿 또는 커서) 위치의 인덱스를 나타낸다. 텍스트 필드에 포커스가 없고 (깜박이는 커서를 통해)caret 위치가 표시되지 않은 경우 텍스트 필드가 포커스를 잃기 전 마지막 caret 위치를 나타낸다. 또는 텍스트 필드가 포커스를 갖지 않는 경우 0 을 나타낸다. 이 속성은 `Selection.getCaretIndex()`가 반환하는 값과 동일한 값을 갖는다. 그러나 `caretIndex` 는 텍스트 필드에 포커스가 없을 때도 접근이 가능한 반면 `Selection.getCaretIndex()`는 -1 을 반환한다

Caret 인덱스는 0 을 기준으로 한다 (그러므로 첫 번째 위치가 0 이다).

참고자료

`Selection.getCaretIndex()`

getCharBoundaries () 메소드

```
public function getCharBoundaries(charIndex:Number): flash.geom.Rectangle
```

Scaleform version: 2.0.37

이 메소드는 문자의 경계 사각형을 반환한다. 이 메소드는 사각형을 반환하는데, 이때 각 글리프보다 좀 더 큰 값을 사용하여 계산된다는 점을 명심하라. 이는 반환된 사각형이 정확한 크기가 아니라는 점을 의미한다 (그림을 참고할 것, 붉은색 사각형은 'a', 'w' 및 'k'의 경계를 표시한다).

Parameters

`charIndex:Number` - 문자에 대한 0 을 기준으로 하는 인덱스 값 (예를 들어 첫 번째 위치는 0, 두 번째 위치는 1)

반환값

`flash.geom.Rectangle` - 문자의 경계 상자를 정의하는 x 와 y 의 최대, 최소값이 있는 사각형. 좌표계는 텍스트 필드의 좌표 공간이다. 즉, (0,0)은 텍스트 필드의 좌측상단이다.

참고자료

`getExactCharBoundaries()`

getExactCharBoundaries () 메소드

```
public function getExactCharBoundaries(charIndex:Number): flash.geom.Rectangle
```

Scaleform version: 2.0.37

이 메소드는 문자의 정확한 경계 사각형을 반환한다. 이 메소드가 사각형을 반환하고 각 글리프의 실제 폭을 사용해서 계산된다는 점을 명심하라(`getCharBoundaries()`는 더 큰 값을 사용한다). 사각형의 높이는 전체 라인의 높이다. 다음 그림을 참고하다. 붉은색 사각형은 'a', 'w' 및 'k'의 정확한 경계를 나타낸다.

Parameters

`charIndex:Number` - 문자에 대한 0 을 기준으로 하는의 인덱스 값 (예를 들어 첫 번째 위치는 0, 두 번째 위치는 1)

반환값

flash.geom.Rectangle – 문자의 정확한 경계를 정의하는 x 와 y 의 최대, 최소값이 있는 사각형. 좌표계는 텍스트 필드의 좌표 공간이다. 즉, (0,0)은 텍스트 필드의 좌측상단이다.

참고자료

`getCharBoundaries()`

getCharIndexAtPoint () 메소드

```
public function getCharIndexAtPoint(x:Number, y:Number): Number
```

Scaleform version: 2.0.37

이 메소드는 x 와 y 매개변수가 정의하는 지점에서의 문자 인덱스 값을 반환한다(0 을 기준).

Parameters

`x:Number` – 문자의 x 좌표

`y:Number` – 문자의 y 좌표

반환값

Number - 문자의 0 을 기준으로 하는 인덱스 값 (예를 들어 첫 번째 위치는 0, 두 번째 위치는 1). 점이 문자 위에 없을 경우 -1 을 반환한다.

getFirstCharInParagraph () 메소드

```
public function getFirstCharInParagraph(charIndex:Number): Number
```

Scaleform version: 2.0.37

이 메소드는 `charIndex` 인덱스에서 문자를 포함하는 단락 내 첫 번째 문자의 인덱스를 반환한다.

Parameters

`charIndex:Number` - 0 을 기준으로 하는 인덱스 값 (예를 들어 첫 번째 문자는 0, 두 번째 문자는 1)

반환값

Number – 동일한 단락 내 첫 번째 문자의 0 을 기준으로 하는 인덱스 값

getLineIndexAtPoint () 메소드

```
public function getLineIndexAtPoint(x:Number, y:Number): Number
```

Scaleform version: 2.0.37

이 메소드는 x 와 y 매개변수가 정의하는 지점의 라인에서 0 을 기준으로 하는 인덱스 값을 반환한다.

Parameters

`x:Number` – 라인의 x 좌표계

`y:Number` – 라인의 y 좌표계

반환값

Number - 0 을 기준으로 하는 라인의 인덱스 값 (예를 들어 첫 번째 라인은 0, 두 번째 라인은 1). 포인트가 라인에 없는 경우 -1 을 반환한다.

getLineLength() 메소드

```
public function getLineLength(lineIndex:Number): Number
```

Scaleform version: 2.0.37

이 메소드는 특정 텍스트 라인에 있는 글자의 수를 반환한다.

Parameters

`lineIndex:Number` - 0 을 기준으로 하는 라인의 인덱스 값 (첫 번째 라인은 0, 두 번째 라인은 1)

반환값

Number – 라인의 문자 수

getLineMetrics() 메소드

```
public function getLineMetrics(lineIndex:Number): Object
```

Scaleform version: 2.0.43

이 메소드는 주어진 텍스트 라인에 대한 메트릭 정보를 반환한다. 반환 객체는 다음과 같은 멤버 값을 갖는다.

`ascent : Number`

텍스트의 ascent 값은 기준선에서 라인의 상단까지의 높이를 픽셀로 나타낸 값이다.

`descent : Number`

텍스트의 descent 값은 기준선에서 라인 하단까지의 길이를 픽셀로 나타낸 값이다.

`height : Number`

선택한 라인 내 텍스트의 높이값을 픽셀로 나타냄 (완벽한 텍스트일 필요는 없음)

`leading : Number`

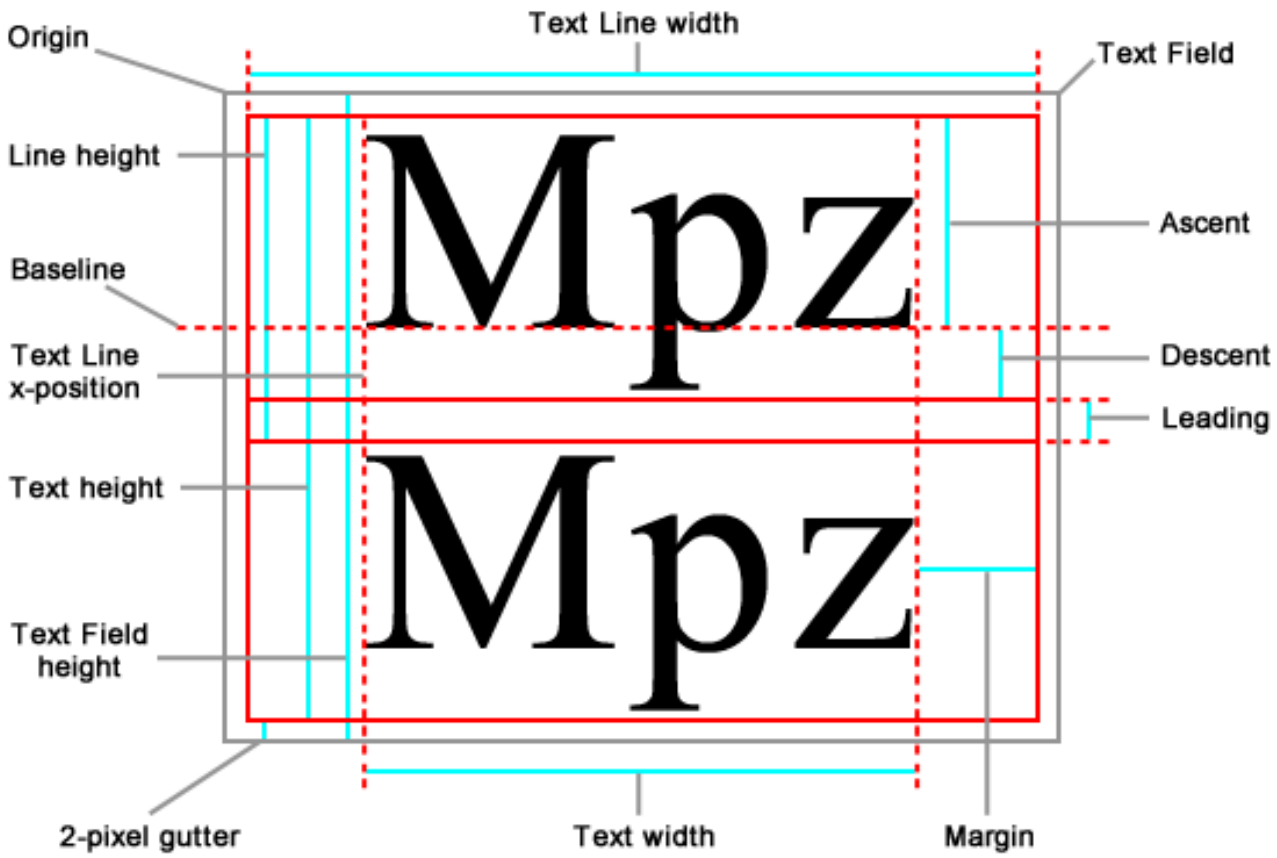
리딩 값은 텍스트 라인 사이의 수직 거리 측정치다

`width : Number`

width 값은 선택한 라인 내 텍스트의 폭을 픽셀로 나타낸 값이다 (완벽한 텍스트일 필요는 없음).

`x : Number`

x 값은 첫 번째 글자의 왼쪽 위치를 픽셀로 나타낸 것이다.



Parameters

`lineIndex:Number` - 0 을 기준으로 하는 라인의 인덱스 값 (예를 들어 첫 번째 라인은 0, 두 번째 라인은 1)

반환값

Number – 메트릭 정보가 있는 객체

getLineOffset () 메소드

```
public function getLineOffset(lineIndex:Number): Number
```

Scaleform version: 2.0.37

이 메소드는 `lineIndex` 매개변수에 설정된 라인의 첫 번째 문자의 인덱스를 반환한다.

Parameters

`lineIndex:Number` - 0 을 기준으로 하는 라인의 인덱스 값 (예를 들어 첫 번째 라인은 0, 두 번째 라인은 1)

반환값

Number - 0 을 기준으로 하는 라인의 인덱스 값

getLineText () 메소드

```
public function getLineText(lineIndex:Number): String
```

Scaleform version: 2.0.43

이 메소드는 `lineIndex` 에 지정된 라인의 텍스트를 반환한다.

Parameters

`lineIndex:Number` - 0 을 기준으로 하는 라인의 인덱스 값 (예를 들어 첫 번째 라인은 0, 두 번째 라인은 1)

반환값

String – 지정된 라인에 포함된 텍스트 문자열

hitTestDisable 속성

```
hitTestDisable:Boolean [read-write]
```

Scaleform version: 1.2.32

`true` 로 설정되면 `MovieClip.hitTest` 함수는 히트 테스트 시에 이 텍스트 필드를 무시한다. 또한, 기타 모든 마우스 이벤트가 텍스트 필드로 전달되지 않는다.

기본값은 false 다.

noTranslate 속성

```
noTranslate:Boolean [read-write]
```

Scaleform version: 1.2.34

true 로 설정되면 GFxTranslator::Translate 콜백이 텍스트 필드에 대해 호출되는 것을 막는다.

기본값은 false 다.

numLines 속성

numLines:Number [read-only]

Scaleform version: 2.0.43

이 속성은 다중 라인 텍스트 필드의 텍스트 라인 수를 나타낸다. wordWrap 속성이 true 로 설정되면 텍스트를 wrap 할 때 라인 수가 증가한다.

topmostLevel 속성

topmostLevel:Boolean [read-write]

Scaleform version: 2.1.50

속성을 true 로 설정하면 이 문자는 그 깊이에 상관없이 모든 다른 문자 위에 표시된다. 이 속성은 모든 레벨에 있는 객체 위에 커스텀 마우스 커서와 팝업을 그릴 필요가 있을 때 유용하다.
기본값은 false 다.

몇몇의 문자들이 "topmostLevel" 로 마킹된 경우, 그리기에 대한 순서는 다음과 같다.:
• Scaleform 3.0.71 까지의 버전들에서는 "topmostLevel" 로 마크되어 있는 문자들의 그리기 순서는 이 속성을 true 로 설정하는가 그렇지 않은가에 따라 달려있다. (그러므로 "topmostLevel" 로 설정된 문자는 첫 번째로 그려질 것임.)
• Scaleform 3.0.72 버전 부터의 그리기 순서는 문자의 가장 윗부분을 마킹하지 않아도 동일하게 순서대로 그릴 수 있다. 예: 만약 ObjectA가 ObjectB의 아래에 그려졌다면 "topmostLevel"속성을 true로 설정하는 순서에 관계없이 ObjectA가 가장 윗부분으로 마킹이 되어있어도 ObjectB 아래에 위치한다.

주석: 문자를 "topmostLevel"로 설정되면 swapDepth 액션스크립트는 이 문자에 아무런 영향을 미치지 않는다.

참고자료:

```
MovieClip.topmostLevel
Button.topmostLevel
```

focusGroupMask 프로퍼티

focusGroupMask : Number

Scaleform version: 3.3.84

이 프로퍼티는 스테이지 캐릭터와 **모든** 자식에게 비트마스크를 설정합니다. 이 비트마스크는 캐릭터에 focus group 소유권을 할당하며, 이는 비트마스크에 표시된 controller 만이 focus 를 캐릭터 내부로 이동시킬 수 있음을 의미합니다. Focus group 은 `setController` 과 FocusGroup 확장 메소드를 통해서 controller 와 연결할 수 있습니다.

예를 들어, "button1"이 controller 0 과 1 을 통해서만 focus 를 맞출 수 있다고 가정해 보겠습니다. 이러한 동작을 얻기 위해서는 focus group 과 controller 를 연결해야 합니다.

```
Selection.setControllerFocusGroup(0, 0);
Selection.setControllerFocusGroup(1, 1);

button1.focusGroupMask = 0x1; // bit 0 – focus group 0
movieclip2.focusGroupMask = 0x1 | 0x2 // bits 0 and 1 – focus groups 0 and 1
```

"focusGroupMask" 비트마스크는 부모 movieclip 으로 설정할 수 있습니다. 이것은 모든 자식에게 마스크 값을 전달합니다.

7.2 선택 및 클립보드 연산

이번 장에서는 선택 및 클립보드 연산에 관한 확장사항을 설명한다.

alwaysShowSelection 속성

```
alwaysShowSelection:Boolean  [read-write]
```

Scaleform version: 2.1.45

기본적으로 텍스트 필드에 포커스가 없을 경우 Scaleform 는 선택된 부분을 강조하지 않는다. 이 속성을 true 로 설정하고 텍스트 필드에 포커스가 없으면 Scaleform 는 회색으로 텍스트 필드 내 선택을 강조한다. 활성 및 비활성 선택 색상은 오버라이드할 수 있다 ("참고자료"부분을 참고할 것).

기본값은 false 다.

참고자료:

```
selectionBkgColor
selectionTextColor
inactiveSelectionBkgColor
inactiveSelectionTextColor
```

copyToClipboard () 메소드

```
public function copyToClipboard([richClipboard:Boolean], [startIndex:Number],
[endIndex:Number]):void
```

Scaleform version: 2.1.45

이 메소드는 텍스트를 클립보드로 복사한다. 모든 매개변수가 선택적이다.

Parameters

`richClipboard:Boolean` – true 인 경우 스타일이 있는 텍스트를 클립보드에 복사한다. 기본값은 `useRichTextClipboard` 속성과 동일하다.

`startIndex:Number` – 복사된 텍스트 세그먼트의 시작 인덱스. 기본값은 `selectionBeginIndex` 와 동일하다.

`endIndex:Number` – 복사된 텍스트 세그먼트의 종료 인덱스. 기본값은 `selectionEndIndex` 와 동일하다.

참고자료:

useRichTextClipboard
selectionBeginIndex
selectionEndIndex
cutToClipboard()
pasteFromClipboard()

cutToClipboard () 메소드

```
public function cutToClipboard([richClipboard:Boolean], [startIndex:Number],
[endIndex:Number]):void
```

Scaleform version: 2.1.45

이 메소드는 클립보드에 텍스트를 복사하고 텍스트 필드에서 이를 제거한다. 모든 매개변수가 선택적이다.

Parameters

`richClipboard:Boolean` – true 인 경우 스타일이 있는 텍스트를 클립보드에 복사한다. 기본값은 `useRichTextClipboard` 속성과 동일하다.

`startIndex:Number` – 복사한 텍스트 세그먼트의 시작 인덱스. 기본값은 `selectionBeginIndex` 와 동일하다.

`endIndex:Number` – 복사한 텍스트 세그먼트의 종료 인덱스. 기본값은 `selectionEndIndex` 와 동일하다.

참고자료:

useRichTextClipboard
selectionBeginIndex
selectionEndIndex
copyToClipboard()
pasteFromClipboard()

inactiveSelectionBkgColor 속성

```
inactiveSelectionBkgColor:Number  [read-write]
```

Scaleform version: 2.1.45

비활성화된 선택 영역의 배경 색을 지정한다. `alwaysShowSelection` 속성이 true 일 경우에만 작동한다. 0xAARRGGBB 형태로 지정되는데, AA 는 알파채널 [0...255]까지의 16 진수값이고, RR 은 적색 [0...255]까지의 16 진수값이고, GG 는 녹색 [0...255]까지의 16 진수값이고, BB 는 청색 [0...255]까지의 16 진수값이다.

알파채널을 0 으로 설정하면 아무것도 그려지지않는 완전 투명상태임을 명심하라. 따라서, 플래시 배경 컬러와 살짝 다를 수 있다. 예를 들어서 완전한 적색을 나타내려면 0xFFFF0000(알파와 적색을 0xFF(255))인데 반해서 일반적인 backgroundColor 속성은 0xFF0000 로 충분하기 때문이다.

참고자료:

```
alwaysShowSelection
inactiveSelectionTextColor
selectionBkgColor
selectionTextColor
```

inactiveSelectionTextColor 속성

```
inactiveSelectionTextColor:Number  [read-write]
```

Scaleform version: 2.1.45

비활성화된 선택 영역의 문자 색을 지정한다. `alwaysShowSelection` 속성이 true 일 경우에만 작동한다. 0xAARRGGBB 형태로 지정되는데, AA 는 알파채널 [0...255]까지의 16 진수값이고, RR 은 적색 [0...255]까지의 16 진수값이고, GG 는 녹색 [0...255]까지의 16 진수값이고, BB 는 청색 [0...255]까지의 16 진수값이다.

알파채널을 0 으로 설정하면 아무것도 그려지지않는 완전 투명상태임을 명심하라. 따라서, 플래시 배경 컬러와 살짝 다를 수 있다. 예를 들어서 완전한 적색을 나타내려면 0xFFFF0000(알파와 적색을 0xFF(255))인데 반해서 일반적인 textColor 속성은 0xFF0000 로 충분하기 때문이다.

참고자료:

```
alwaysShowSelection
```

inactiveSelectionBkgColor
selectionBkgColor
selectionTextColor

noAutoSelection 속성

noAutoSelection:Boolean [read-write]

Scaleform version: 2.1.45

True 로 설정하면 포커스가 텍스트 필드로 전달될 때 자동 선택을 방지한다.

기본값은 false 다.

pasteFromClipboard () 메소드

public function pasteFromClipboard([richClipboard:Boolean], [startIndex:Number], [endIndex:Number]):void

Scaleform version: 2.1.45

클립보드에서 텍스트를 붙인다. 모든 매개변수가 선택적이다.

Parameters

richClipboard:Boolean – true 인 경우 스타일이 있는 텍스트를 클립보드에서 붙여온다. 기본값은 useRichTextClipboard 속성과 동일하다.

startIndex:Number – 클립보드에서 텍스트를 통해 대치될 세그먼트의 시작 인덱스. 기본값은 selectionBeginIndex 와 동일하다.

endIndex:Number – 클립보드에서 텍스트를 통해 대치된 세그먼트의 종료 인덱스. 기본값은 selectionEndIndex 와 동일하다.

참고자료:

useRichTextClipboard
selectionBeginIndex
selectionEndIndex
copyToClipboard()
cutToClipboard()

selectionBeginIndex 속성

`selectionBeginIndex:Number [read-only]`

Scaleform version: 2.1.45

이 속성은 현재 선택 영역 내 첫 번째 문자의 0 을 기준으로 하는 문자 인덱스 값을 나타낸다. 선택한 텍스트가 없으면 이 속성은 caretIndex 값을 갖는다. 이 속성은 `Selection.getBeginIndex()`의 반환값과 동일한 값을 갖는다. 그러나 `selectionBeginIndex` 는 텍스트 필드에 포커스가 없을 때에도 접근이 가능하다. 반면 이 경우 `Selection.getBeginIndex()`는 -1 을 반환한다.

참고자료:

```
caretIndex
selectionEndIndex
Selection.getBeginIndex()
```

selectionEndIndex 속성

`selectionEndIndex:Number [read-only]`

Scaleform version: 2.1.45

이 속성은 현재 선택 영역 내 마지막 문자의 0 을 기준으로 하는 문자 인덱스 값을 나타낸다. 선택한 텍스트가 없을 때 이 속성은 `caretIndex` 의 값을 갖는다. 이 속성은 `Selection.getEndIndex()`가 반환하는 값과 동일한 값을 포함한다. 그러나 `selectionEndIndex` 는 텍스트 필드에 포커스가 없을 때도 접근이 가능하나 이 경우 `Selection.getEndIndex()`는 -1 을 반환한다.

참고자료:

```
caretIndex
selectionBeginIndex
Selection.getEndIndex()
```

selectionBkgColor 속성

```
selectionBkgColor:Number  [read-write]
```

Scaleform version: 2.1.45

활성화된 선택 영역의 배경 색을 지정한다. `alwaysShowSelection` 속성이 true 일 경우에만 작동한다. 0xAARRGGBB 형태로 지정되는데, AA 는 알파채널 [0...255]까지의 16 진수값이고, RR 은 적색 [0...255]까지의 16 진수값이고, GG 는 녹색 [0...255]까지의 16 진수값이고, BB 는 청색 [0...255]까지의 16 진수값이다.

알파채널을 0 으로 설정하면 아무것도 그려지지 않는 완전 투명상태임을 명심하라. 따라서, 플래시 배경 컬러와 살짝 다를 수 있다. 예를 들어서 완전한 적색을 나타내려면 0xFFFF0000(알파와 적색을 0xFF(255))인데 반해서 일반적인 backgroundColor 속성은 0xFF0000 로 충분하기 때문이다.

참고자료:

```
inactiveSelectionTextColor
inactiveSelectionBkgColor
selectionTextColor
```

selectionTextColor 속성

```
selectionTextColor:Number  [read-write]
```

Scaleform version: 2.1.45

활성화된 선택 영역의 문자 색을 지정한다. `alwaysShowSelection` 속성이 true 일 경우에만 작동한다. 0xAARRGGBB 형태로 지정되는데, AA 는 알파채널 [0...255]까지의 16 진수값이고, RR 은 적색 [0...255]까지의 16 진수값이고, GG 는 녹색 [0...255]까지의 16 진수값이고, BB 는 청색 [0...255]까지의 16 진수값이다.

알파채널을 0 으로 설정하면 아무것도 그려지지않는 완전 투명상태임을 명심하라. 따라서, 플래시 배경 컬러와 살짝 다를 수 있다. 예를 들어서 완전한 적색을 나타내려면 0xFFFF0000(알파와 적색을 0xFF(255))인데 반해서 일반적인 textColor 속성은 0xFF0000 로 충분하기 때문이다.

참고자료:

```
inactiveSelectionBkgColor
inactiveSelectionBkgColor
selectionBkgColor
```

useRichTextClipboard 속성

```
useRichTextClipboard:Boolean  [read-write]
```

Scaleform version: 2.1.45

텍스트와 함께 텍스트 포맷정보의 복사 및 붙이기 여부를 정의한다. True 로 설정되면 Scaleform 는 텍스트 필드간 복사 및 붙이기를 할 때 포맷도 복사한다(배열, 볼드 및 이탤릭 등). 원본 텍스트 및 목표 텍스트 모두 `useRichTextClipboard` 가 `true` 로 되어 있어야 한다.

기본값은 `false` 다.

7.3 텍스트 크기 및 정렬

여기에선 텍스트의 크기 및 텍스트 정렬을 제어하는 확장을 다룬다. 표준 정렬뿐만 아니라 (좌 우, 중앙) Scaleform 는 수직정렬 및 수직 자동사이즈 기능도 제공한다.

fontScaleFactor 속성

```
fontScaleFactor:Number  [read-write]
```

Scaleform version: 2.0.43

이 속성은 텍스트 필드 전체에 있는 모든 폰트에 대한 폰트 크기값을 정의한다. 모든 폰트 크기는 `fontScaleFactor` 값을 곱하거나 나눠서 증가 또는 감소시킬 수 있다.

기본값은 1.0 이다.

textAutoSize 속성

`textAutoSize:String [read-write]`

Scaleform version: 2.0.43

`textAutoSize`는 텍스트 폰트 크기의 자동 리사이즈를 가능하게 한다. 이 속성에 대한 유효값은 "`none`", "`shrink`" 및 "`fit`"이다. 만약 이 모드가 켜 있고 ("`shrink`" 또는 "`fit`") 텍스트가 텍스트 필드 크기에 맞지 않는 경우 텍스트 필드의 크기를 비례에 맞춰 감소시켜 텍스트 필드 내 전체 텍스트를 맞추기 때문에 스케일링이 필요 없다. 만약 텍스트 크기가 너무 작아지면 (약 5pt) 기본 스크롤 로직을 사용하며 더 이상의 폰트 크기 축소가 일어나지 않는다.

textAutoSize 를 "`fit`"로 설정하면 텍스트 필드를 텍스트로 채울 때까지 폰트 크기 증가가 가능하다. "`shrink`" 모드는 텍스트 필드 내에서 텍스트의 크기가 맞지 않을 때 그 크기를 줄일 수만 있으며 원래 폰트의 크기가 증가하진 않는다.

기본값은 "none"이다.

verticalAlign 속성

`verticalAlign:String [read-write]`

Scaleform version: 2.0.43

텍스트 상자 내 텍스트의 수직 정렬을 설정한다. 유효한 속성값은 "none" (또는 "none"과 동일한 "top"), "bottom", "center"다. 만약 속성을 "center"로 맞추면 텍스트는 텍스트 상자 내 중앙 정렬을 하고, "bottom"인 경우 텍스트 상자의 바닥에 정렬한다 (아래 그림 참고).

기본값은 "none"이다.

참고자료 :

```
verticalAutoSize
TextField.align
```

verticalAutoSize 속성

`verticalAutoSize:String [read-write]`

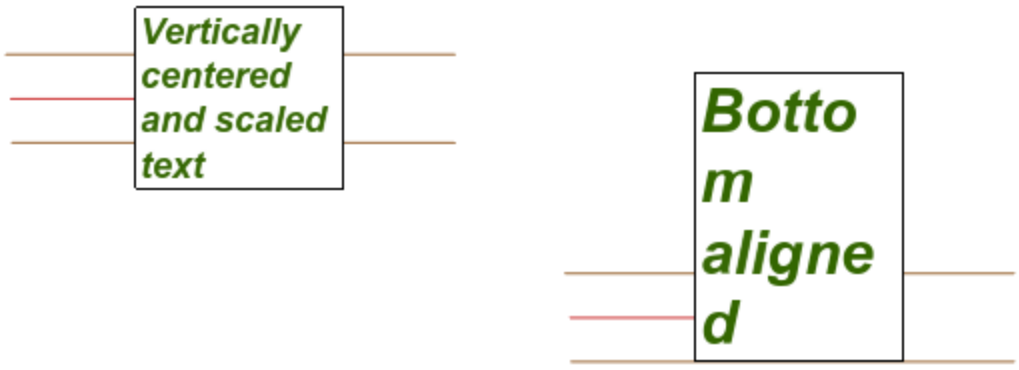
Scaleform version: 2.0.43

텍스트 필드의 자동 수직 사이즈 및 정렬을 제어한다. 지정 가능한 값은 "none" (기본), "top", "bottom", "center"다. `verticalAutoSize` 를 "none"으로 설정하면 (기본값) 리사이즈는 일어나지 않는다.

`verticalAutoSize` 를 "top"으로 설정하면 텍스트 필드는 `wordWrap` 을 `true` 로 설정한 것과 같이 일반적인 `TextFieldautoSize` 와 유사한 속성을 갖는다 (그러므로 텍스트 필드의 바닥만 리사이즈되며 오른쪽은 고정된 상태로 남게 된다).

`verticalAutoSize` 를 "center"로 설정하면 텍스트 필드가 리사이즈되며 텍스트 필드 내 모든 리사이즈는 상단 및 하단 마진 모두에 동등하게 적용된다. 앵커 지점은 원래 텍스트 필드 높이의 가운데 지점에 위치한다 (아래 그림 참고).

`verticalAutoSize` 를 "bottom"으로 설정하면 텍스트 필드는 상단 마진으로 리사이즈된다. 앵커 포인트는 원래 텍스트 상자의 바닥에 위치한다 (아래 그림 참고).



기본값은 "none"이다.

참고자료:

```
verticalAlign
TextField.autoSize
```

7.4 HTML 확장

여기에선 Scaleform 내 HTML 에 대한 확장을 다룬다

<FONT> 태그, ALPHA 속성

ALPHA="#xx"

텍스트에 대한 알파 투명도를 00 에서 FF 범위에서 제어한다 (0 – 255). 0 은 완전한 투명, 255 는 완전한 불투명을 의미한다. COLOR 속성과 결합되는 ALPHA 속성은 HTML 내 텍스트를 반투명하게 만든다.

<IMG> 태그

Scaleform 2.0/2.1 에 있는 <IMG> 태그는 할당된 링크 식별자와 함께 라이브러리에서 불러온 이미지만을 참고할 수 있다. Scaleform 는 웹이나 파일 같은 ("http://", "file://" 그리고 그 외 프로토콜을 지원하지 않음.) 외부 리소스들을 로딩하는 것을 지원하지 않는다. 5.6 장의 이미지 교체를 참고하자. 이미지를 불러오고 링크 식별자를 할당하는 법은 아래를 참고한다.

다음은 "myImage"로 불러온 이미지를 참고해서 <IMG> 태그를 사용하는 HTML 의 예다.

```
t.htmlText = "<p align='right'>abra<img src='myImage' width='20' height='30'
             align='baseline' vspace='-10'>bed232</p>";
```

현재 IMG 태그에 대해 지원이 이뤄지는 속성은 다음과 같다:

src –라이브러리의 이미지 심볼에 대한 연동 식별자(linkage identifier)를 지정합니다. 현재는 이미지만 지원합니다. 이 속성은 필수 항목이며, 다른 모든 속성은 선택적입니다. 외부 파일(JPEG, GIF, PNG, SWF)은 아직 지원하지 않습니다. 사용자 프로토콜 "img://"은 여기서 임베드된 이미지 linkage id 대신 사용 가능합니다. 이 경우, 사용자 정의 가상 메소드 GFx::ImageCreator:: LoadImage 가 호출됩니다.

width - 삽입한 이미지의 폭, 픽셀 단위

height - 삽입한 이미지의 높이, 픽셀 단위

align - 텍스트 필드 내 내장된 이미지의 수평정렬을 정의한다. 현재 지원하는 값은 "baseline" 뿐이다.

vspace –"baseline" 기준으로 상대적인 이미지의 오프셋을 픽셀로 지정한다. 양수 값은 이미지를 기준선 위로, 음수 값은 이미지를 아래로 이동시킴을 의미한다.

지원하지 않는 속성:

id - 내장된 이미지, SWF, 무비클립을 포함하는 무비클립 인스턴스 명 지정

align - "left" 와 "right". 텍스트 필드 내 내장된 이미지의 수평정렬을 정의한다.

hspace - "left"와 "right"에서 문자가 나타나지 않는 이미지를 둘러싼 수평공간의 크기지정

다른 SWF 파일에서 이미지를 불러온 경우 이 이미지는 IMG 태그에서 사용하기 위해 플래시 파일 어딘가에 명시적으로 위치해야 한다. 만약 이미지를 불러왔으나 전혀 사용하지 않는 경우 플래시는 결과로 나오는 SWF 파일에서 해당 이미지에 대한 참고를 완전히 제거한다. 이를 방지하기 위한 한 가지 방법은 무비 클립을 생성하고 IMG 태그에 필요한 모든 불러온 이미지를 드래그 앤 드롭하는 것이다. 이 무비 클립을 엑스포트 해야 한다는 것을 잊지 말라. 그 이유는 명시적으로 사용하지도 않고, 게다가 엑스포트도 되어 있지 않다면 플래시는 생성된 SWF 파일로부터 이를 분리하기 때문이다.

7.5 Shadow 효과 제어

여기에선 서로 다른 셰도우 효과 (블러, 녹아웃 등)를 제어하는 확장을 설명한다.

blurX, blurY 속성

```
blurX:Number  [read-write]
blurY:Number  [read-write]
```

Scaleform version: 2.0.41

텍스트의 블러 반지름을 설정하거나 가져온다. 유효값은 0 에서 15.9 다 (부동 소수점).

기본값은 0 이다.

참고자료:

```
blurStrength
```

blurStrength 속성

```
blurStrength:Number  [read-write]
```

Scaleform version: 2.0.41

텍스트의 블러 강도를 가져오거나 설정한다. 유효값은 0 에서 15.9 다 (부동 소수점).

기본값은 0 이다.

참고자료:

```
blurX
blurY
```

shadowAlpha 속성

```
shadowAlpha:Number  [read-write]
```

Scaleform version: 2.0.41

셰도우 색상에 대한 알파 투명도를 제어한다. 유효값은 0.00 에서 1.00 이다. 예를 들어 .25 는 투명도를 25%로 설정한다는 것을 의미한다.

참고자료:

`shadowColor`

shadowAngle 속성

```
shadowAngle:Number  [read-write]
```

Scaleform version: 2.0.41

`DropShadowFilter` 와 유사하게 셰도우의 각도를 제어한다. 유효값은 0 에서 360˚ (부동 소수점)이다. 기본값은 45 다.

참고자료:

`DropShadowFilter`

shadowBlurX, shadowBlurY 속성

```
shadowBlurX:Number  [read-write]
shadowBlurY:Number  [read-write]
```

Scaleform version: 2.0.41

셰도우의 블러 반지름을 제어한다. 유효값은 0 에서 15.9 다 (부동 소수점).
기본값은 0 이다.

참고자료:

`shadowStrength`

shadowColor 속성

shadowColor:Number [read-write]

Scaleform version: 1.1.31

셰도우의 컬러를 정의한다 (자세한 내용은 shadowStyle 참고). 색상은 0xRRGGBB 포맷으로 정의한다. 여기에서 RR 은 적색의 16 진수 [0..255], BB 는 청색의 16 진수 [0..255] 그리고 GG 는 녹색의 16 진수 [0..255]다.

참고자료

shadowStyle

shadowDistance 속성

shadowDistance:Number [read-write]

Scaleform version: 2.0.41

DropShadowFilter 와 유사하게 셰도우에 대한 오프셋 거리를 픽셀 단위로 제어한다.

참고자료:

DropShadowFilter

shadowHideObject 속성

shadowHideObject:Boolean [read-write]

Scaleform version: 2.0.41

텍스트의 숨김 여부를 나타낸다. True 인 경우 텍스트를 그리지 않았음을 의미한다. 이 경우 셰도우만 보인다. 기본값은 false (텍스트 보임)이다.

shadowKnockOut 속성

shadowKnockOut:Boolean [read-write]

Scaleform version: 2.0.41

녹아웃 효과 (true)를 제어한다. 이는 객체를 투명으로 효과적으로 채우며 문서의 배경색을 나타낸다. 기본값은 false (녹아웃 없음)이다.

shadowQuality 속성

shadowQuality:Number [read-write]

Scaleform version: 2.0.41

셰도우 또는 글로우 필터의 품질을 제어한다. 플래시와는 다르게 1 또는 2 가 사용 가능하다. 1 은 저품질, 2 는 고품질을 나타낸다.

shadowStrength 속성

shadowStrength:Number [read-write]

Scaleform version: 2.0.41

셰도우의 블러 강도를 제어한다. 유효값은 0 에서 15.9 다 (부동 소수점).
기본값은 0 이다.

참고자료:

shadowBlurX
shadowBlurY

shadowStyle 속성

shadowStyle:String [read-write]

Scaleform version: 1.1.31

shadowColor 와 결합하여 텍스트 필드에 대한 셰도우 렌더링을 제어한다. 특히 shadowStyle 포맷 문자열은 좌표계와 함께 적용될 텍스트 레이어 집합을 설명한다. 예를 들면

```
field.shadowStyle = "s{0,-1.5}{-1.4,1.2}{1.4,1.2}t{0,0}";
field.shadowColor = 0xff0000;
```

문자열 내에서 "s"는 shadowColor 색상값으로 그리게 되는 셰도우 레이어의 값을 설정하는 반면 "t"는 표준 TextField.textColor 색상값으로 그려지는 전면 텍스트 레이어의 값을 설정한다 이러한 디리미터(delimiter)는 렌더링이 이뤄지는 해당 텍스트 레이어의 오프셋을 설명하는 좌표쌍 뒤에 따라온다. 각 좌표 단위는 동일 변환이 텍스트에 적용될 때 한 픽셀에 매핑 된다. "t"와 그 좌표쌍을 스타일에서 제거하는 경우 스타일 문자열 텍스트는 "t{0,0}"을 정의할 때와 동일하게 그려진다

현재까지는 셰도우 레이어를 사용하면 추가적인 삼각형을 생성하고 draw primitive 를 호출한다는 점을 명심하자. 즉, 텍스트가 여러 번 렌더링 된다는 것이다. 성능을 테스트해보고 가능하다면 셰도우를 제한할 것을 강력히 권장한다.

참고자료:

- shadowColor
- TextField.textColor

7.6 이미지 교체

여기에선 이미지 교체를 제어하는 확장에 대한 설명을 제공한다.

setImageSubstitutions () 메소드

```
public setImageSubstitutions(substInfoArr:Array) : void
public setImageSubstitutions(substInfo:Object) : void
public setImageSubstitutions(null) : void
```

Scaleform version: 2.0.38

텍스트 필드에 서브 문자열로 이미지 교체를 설정한다.

문자열 교체는 SWF 에 내장된 이미지에 대해서만 작동한다. 이러한 이미지는 또한 엑스포트 명을 갖기 위해 링크도 갖고 있어야 한다. 이미지를 SWF 로 내장시키려면 다음이 필요하다.

1. 비트맵 이미지를 라이브러리에 불러온다.
2. (윈도우에서)라이브러리 내 이미지에서 마우스 우측클릭 혹은 (맥에서는)CTRL 클릭 한 후 컨텍스트 메뉴에서 Linkage 를 선택한다.
3. Export for ActionScript 를 선택하고, 첫번째 프레임 엑스포트에서 원하는 이름을 정한다 (예, myImage)
4. OK 를 클릭하여 링크 식별자를 설정한다.

이미지를 불러오고 링크 식별자를 할당한 후 `BitmapData` 인스턴스를 생성한다.

다음은 액션스크립트 코드 예다.

```
import flash.display.BitmapData;
var imageBmp:BitmapData = BitmapData.loadBitmap("myImage");
```

주: 임포트 문장을 잊지 말라!(`import flash.display.BitmapData;`

혹은 `flash.display.BitmapData` 같은 완벽한 이름 사용).

그렇지 않으면 결과가 "undefined"로 나온다.

하나 이상의 이미지를 교체로 사용하는 경우 각 이미지에 대해 위의 단계를 반복할 필요가 있으며 이를 통해 서로 다른 링크 ID 를 부여한다.

단일 교체 디스크립터 객체는 다음과 같은 멤버를 갖는다.

`subString:String`

이미지에 의해 교체될 서브 문자열을 정의한다. 이는 필수사항이다. 이 서브 문자열의 최대 길이는 15 문자다.

`image : BitmapData`

이미지를 정의한다. 필수

`width : Number`

화면상의 이미지 폭을 픽셀로 정의한다. 선택

`height : Number`

화면상의 이미지 높이를 픽셀로 정의한다. 선택

`baseLineY : Number`

이미지 내 기준선의 Y 오프셋을 원래 이미지의 (변환이 없는)픽셀로 정의한다. 선택. 기본적으로 이 값은 이미지의 높이와 동일하므로 이미지 바닥이 기준선으로 나타난다.

`id : String`

"updateImageSubstitution" 호출에 대한 첫 번째 매개변수로 사용하는 id 를 정의한다. 선택

"substInfoArr"는 이러한 객체의 배열이어야 하며 또한 "substInfo"는 객체의 인스턴스여야 한다. 버전 `setImageSubstitutions(substInfo:Object)`는 단일 개체만을 설정할 수 있는 반면 버전 `setImageSubstitutions(substInfoArr:Array)`은 여러 개체를 설정할 수 있다.

`setImageSubstitutions` 에 대한 모든 호출은 내부 목록에 교체를 추가한다는 점을 기억하라. 이를 삭제하려면 `setImageSubstitutions(null)`을 호출한다.

교체될 배열의 참조를 유지하거나 `setImageSubstitutions` 을 호출한 후 액션스크립트 코드 내 단일 설명자 객체에 대한 참조를 유지할 필요는 없다. 그러나 액션스크립트에서 어느 정도 참조할 필요가 있다면 보관해두자. 왜냐하면 텍스트 필드로부터 교체 배열 되가져오는 방법이 없기 때문이다.

Parameters

`substInfoArr:Array –`	디스크립터 객체 교체 베열 (위를 참고)
`substInfo:Object –`	단일 교체 디스크립터 객체 (위를 참고).
`null –`	모든 교체 소거

참고자료:

updateImageSubstitution()

예:

```
var b1 = BitmapData.loadBitmap("smile1");
var b2 = BitmapData.loadBitmap("smile2");
var b3 = BitmapData.loadBitmap("smile3");
var a = new Array;
a[0] = { subString:"=)", image:b1, baseLineY:35, width:20, height:20, id:"sm=)" };
a[1] = { subString:":-)", image:b2, baseLineY:20, id:"sm:-)" };
a[2] = { subString:":\\", image:b3, baseLineY:35, height:100 };
a[3] = { subString:":-\\", image:b1 };
t.setImageSubstitutions(a);
```

텍스트 필드가 따옴표 없이 서브 문자열 "=)"을 포함하자마자 이 서브 문자열은 "smile1" 링크 식별자가 있는 이미지로 교체된다.

updateImageSubstitution () 메소드

```
public updateImageSubstitution(id:String, image:BitmapData) : void
```

Scaleform version: 2.0.38

`setImageSubstitutions` 함수로 생성된 텍스트 교체 이미지를 바꾸거나 제거한다.

Parameters

`id:String –`	교체 ID, `setImageSubstitutions` 호출에 사용된 디스크립터 객체와 동일한 id 번호
`image:BitmapData –`	새로운 이미지를 정의. Null 인 경우 교체 완전제거

참고자료:

`setImageSubstitutions()`

예:

```
t.updateImageSubstitution("sm=)", b3);
```

다음은 내장 이미지의 애니메이션 예다. onEnterFrame 핸들러 또는 setInterval 을 사용해서 업데이트가 이뤄질 수 있다. `updateImageSubstitution` 을 호출할 때 텍스트 재포매팅이 이뤄지지 않음을 명심한다. 따라서, 새로운 이미지의 크기는 기존의 크기와 동일해야 한다.

```
var phase = 0;
var b1a = BitmapData.loadBitmap("smile1a");
var b2a = BitmapData.loadBitmap("smile2a");

onEnterFrame = function()
{
   if (phase % 10 == 0)
   {
      if (phase % 20 == 0)
      {
         _root.t.updateImageSubstitution("sm=)", b1);
         _root.t.updateImageSubstitution("sm:-)", b2);
      }
      else
      {
         _root.t.updateImageSubstitution("sm=)", b1a);
         _root.t.updateImageSubstitution("sm:-)",b2a);
      }
   }
   ++phase;
}
```

7.7 IME 지원

이 부분은 IME 지원을 위한 확장을 설명한다.

disableIME 속성

```
disableIME:Boolean  [read-write]
```

Scaleform version: 2.1.50

True 로 설정되면 이 텍스트 필드에서 IME 가 활성화되는 것을 막는다.

기본값은 false 다.

getIMECompositionStringStyle () 메소드

```
public function getIMECompositionStringStyle(categoryName:String): Object
```

Scaleform version: 2.1.50

IME 색상 설정의 지정 범위에 대한 색상 설정 디스크립터를 반환한다. categoryName 과 디스크립터 객체에 대한 자세한 설명은 setIMECompositionStringStyle 을 참고한다.

Parameters

categoryName:String – IME 색상 설정 카테고리 (setIMECompositionStringStyle 참고).

반환값

Object – 카테고리에 대한 색상 설정 디스크립터 (\참고).

참고자료:

setIMECompositionStringStyle
disableIME

setIMECompositionStringStyle () 메소드

```
public function setIMECompositionStringStyle(categoryName:String,
                                    styleDescriptor:Object): Number
```

Scaleform version: 2.1.50

IME 색상 설정의 적절한 카테고리에 대해 스타일을 설정. categoryName 매개변수는 다음과 같은 값을 포함할 수 있다.

"compositionSegment" – 전체 컴포지션 문자열을 위한 색상 설정을 설정

"clauseSegment"– 절 세그먼트에 대한 색상 설정을 설정

"convertedSegment" – 변환된 텍스트 세그먼트에 대한 색상 설정을 설정

"phraseLengthAdj" – 구 길이 조정에 대한 색상 설정을 설정

"lowConfSegment" – 신뢰도가 낮은 문자에 대한 색상 설정을 설정

styleDescriptor 는 다음과 같은 멤버를 갖는 객체 인스턴스다.

textColor : Number
텍스트 색상

backgroundColor : Number
배경색

`underlineColor : Number`

밑줄색

`underlineStyle : String`

밑줄 스타일. 유효값은 다음과 같다.

```
"single"
"thick"
"dotted"
"ditheredSingle"
"ditheredThick"
```

모든 색상은 0xRRGGBB 포맷으로 정의한다. 여기에서 RR 은 적색 컴포넌트의 16 진수 [0..255], BB 는 청색 컴포넌트의 16 진수 [0..255] 그리고 GG 는 녹색 ` 16 진수 [0..255]이다.

Parameters

`categoryName:String` – IME 색상 설정 카테고리(설명 참고)

`styleDescriptor:Object` – 색상 설정 디스크립터 객체 (설명 참고)

참고자료:

`getIMECompositionStringStyle`
`disableIME`

8 TextFormat 클래스 확장

TextFormat 클래스는 색상, 폰트 크기, 밑줄 등과 같은 문자 포맷 정보를 나타낸다.

alpha 속성

`alpha:Number [read-write]`

Scaleform version: 2.0.41

텍스트에 대한 알파 투명도 값을 퍼센트로 제어한다. 유효값은 0 – 100 (%)이다. 표준 플래시의 `TextFormat.color`는 알파 투명도 설정이 불가능하므로 이 확장은 텍스트를 부분적 또는 완전히 투명하게 만드는 데 사용될 수 있다.

참고자료:

TextFormat
HTML <FONT ALPHA='xx'>

9 Stage 클래스 확장

Stage 클래스 속성 전체를 지원하는 것뿐만 아니라 Scaleform 는 스테이지 크기 추적을 향상시키는 확장도 제공한다. 이러한 확장은 "visibleRect", "safeRect" 및 "originalRect"와 같은 속성을 포함할 뿐만 아니라, "onResize" 핸들러에 대한 추가적인 매개변수를 포함하는데 이는 현재 눈에 보이는 프레임 사각형을 나타낸다. 보다 자세한 사항은 아래를 참고한다.

액션스크립트 1.0 에선 `Stage.visibleRect` 에서와 같이 확장을 직접 참조하는 것이 가능하다.

액션스크립트 2.0 은 허용 범위가 좁아서 이러한 호출에 대해 컴파일러가 투덜거리게 된다.

ActonScript 2.0 컴파일러를 닥치게 만들려면 다음과 같은 접근문장을 사용해야 한다.

```
Stage["visibleRect"]..
```

visibleRect 속성

```
visibleRect:flash.geom.Rectangle  [read-only]
```

Scaleform version: 2.2.56

이 속성은 현재 보이는 사각형을 갖고 있다. 이 사각형은 종횡비, 스케일 모드, 정렬 및/또는 뷰포트의 크기를 변경하고 해당 순간 어떤 영역이 보이는 지를 알고 싶은 경우 변경된다. 이 사각형은 음수 좌측상단 좌표를 가질 수 있다.

참고자료:

```
safeRect
originalRect
```

safeRect 속성

```
safeRect:flash.geom.Rectangle  [read-only]
```

Scaleform version: 2.2.56

이 속성은 현재 설정된 안전한 사각형을 갖고 있다. 이 사각형은 `GFx::Movie::SetSafeRect` 를 사용해서 설정해야 한다. 설정되지 않은 경우 "`visibleRect`" 속성과 동일한 사각형을 갖는다.

참고자료:

```
visibleRect
```

originalRect

originalRect 속성

```
originalRect:flash.geom.Rectangle  [read-only]
```

Scaleform version: 2.2.56

이 속성은 원본 SWF 의 크기가 있는 원본 SWF 사각형을 항상 갖고 있다. 그러므로 플래시 스튜디오 내 "Document properties" 의 크기를 (600px by 400px)로 설정하면 이 사각형은 {0, 0, {600, 400} } 값을 갖는다.

참고자료:

visibleRect
safeRect

onResize 이벤트 핸들러

```
onResize = function([visibleRect:flash.geom.Rectangle]) {}
```

Scaleform version: 2.2.56

onResize 이벤트 핸들러는 추가 매개변수를 갖는 확장이다. 이 매개변수는 현재 보이는 사각형을 나타내며 Stage.visibleRect 확장 속성을 통해 반환된 값과 같은 값을 갖는다.

참고자료:

visibleRect

translateToScreen() 메소드

```
public function translateToScreen(pt:Object): Point
```

Scaleform version: 3.3.84

이 메소드는 화면 좌표 시스템에 매핑된 Stage 좌표 시스템 내의 point 를 반환합니다.

파라미터

`pt:Object` – 변경될 point 의 좌표를 나타내는 x, y 멤버를 가지는 Object 입니다. flash.geom.Point 은 제네릭 Object 대신 사용할 수 있습니다.

반환

Point – 화면 좌표 시스템에 매핑된 입력 point 입니다.

10 배열 클래스 확장

로케일 지정 정렬

플래시 버전의 Array.sort 와 Array.sortOn 메소드는 유니코드에 기반해서 값을 소팅 한다. 하지만, 이 값이 항상 옳은 것은 아니다. 특히 프랑스어, 스페인어, 독일어 등에서는 말이다. 유니코드 소팅은, 모든 문자열의 발음구분기호(umlaut, accent, acute 등)가 z 다음에 나타나므로 플래시에서 ['á', 'z', 'a']를 소팅 하면 현재 로케일과 무관하게 ['a', 'z', 'á']이 반환되는 것이다. 하지만, 프랑스어로 로케일 지정을 하면 ['a', 'á', 'z']로 소팅 될 것이다.

Array.LOCALE:이 문제를 해결하기 위해서 Scaleform 는 Array.sort/sortOn 메소드에 대하여 옵션을 제공한다. 그것이 바로 Array.LOCALE 이다.

```
var arr = [‘á’, ‘z’, ‘a’];
var newArr1 = arr.sort(); // returns [‘a’, ‘z’, ‘á’]
var newArrLoc = arr.sort(Array.LOCALE); // returns [‘a’, ‘á’, ‘z’]
var newArrRev = arr.sort(Array.LOCALE | Array.DESCENDING); // [‘z’, ‘á’, ‘a’]
```

대소문자 구분없는 로케일 지정 소팅도 가능하다.

```
var arr = [‘Á’, ‘z’, ‘a’];
var a = arr.sort(Array.LOCALE | Array.CASEINSENSITIVE); // returns [‘a’, ‘Á’, ‘z’]
```

이러한 기능이 특정 시스템에서는 작동하지 않을 수도 있다. 이러한 기능을 커널에서 지원하지 않는다면 Array.LOCALE 소팅을 해도 유니코드의 결과와 동일할 것이다.

Scaleform version: 3.0.65

참고자료:

String.localeCompare

11 문자열 클래스 확장

localeCompare() 메소드

이 함수는 2 개 이상의 문자열의 소팅 순서를 배교해서 그 값을 숫자로 반환한다. 이 메소드는 로케일 지정 상태로 비교를 한다. 문자열이 동일하다면 0, 원본 문자열 값이 매개변수로 지정된 값보다 앞선다면 음수값, 원본 문자열 값이 매개변수보다 나중에 나온다면 양수값을 반환한다. "배열 확장 클래스"를 보면 더 상세한 로케일 지정 문자열 연산에 대하여 알 수 있을 것이다. 이러한 기능이 특정 시스템에서는 작동하지 않을 수도 있다. 이러한 기능을 커널에서 지원하지 않는다면 Array.LOCALE 소팅을 해도 유니코드의 결과와 동일할 것이다.

```
public localeCompare(other:String [, caseInsensitive:Boolean]) : Number
```

Scaleform version: 3.0.65

Parameters

`other:String` – 비교할 문자열

`caseInsensitive:Boolean` – 대소문자를 무시한 비교를 할 것인지 지정하는 추가 인자. 인자값이 없으면 대소문자 구분비교를 수행한다.

참고자료:

`Array.sort`

12 Video 클래스 확장

표준 플래쉬에서 하나의 비디오 오브젝트는 한번에 하나의 NetStream 오브젝트로부터 데이터를 받을 수 있다. Scaleform 는 이 제한적인 기능을 제거하였고, 다양한 비디오 오브젝트들을 같은 NetStream 오브젝트에 붙여 넣을 수 있고 같은 비디오 데이터를 받을 수 있도록 했다

clipRect 속성

```
clipRect: flash.geom.Rectangle [read-write]
```

Scaleform version: 3.0.70

이 속성은 하나의 비디오 오브젝트가 본래의 비디오 프레임의 일부분에 출력하도록 해준다.
예:

```
video.clipRect = new flash.geom.Rectangle(10, 20, 100, 100);
```

13 전역 확장

noInvisibleAdvance 속성

```
noInvisibleAdvance : Boolean
```

Scaleform version: 2.1.52

True 로 설정되면 이 속성은 보이지 않는 모든 무비 클립의 진행을 중지한다. 숨겨진 무비 클립이 많은 SWF 의 성능 향상을 위해 사용할 수 있다. 플래시는 보이지 않는 무비 클립도 진행시킨다 (타임라인 애니메이션을 실행, 프레임의 액션스크립트 호출 등). 그러므로 이 속성을 "true"로 설정하면 Scaleform 와 플래시 사이의 속성 차이를 유발할 수 있다.

참고자료:

```
MovieClip.noAdvance
```

imecommand 함수

```
imecommand(command:String, parameters:String) : Void
```

Scaleform version: 2.1.50

이 함수는 fscommand 와 유사하나 Scaleform IME 구현하고만 통신한다. Scaleform 내부에서 사용된다.

getIMECandidateListStyle () 함수

```
public function getIMECompositionStringStyle(categoryName:String): Object
```

Scaleform version: 2.1.50

IME 후보 목록에 대한 색상 설정 디스크립터 객체를 반환한다. 디스크립터 객체에 대한 보다 자세한 사항은 `setIMECandidateListStyle` 을 참고한다.

반환값

Object – 후보 목록에 대한 색상 설정 디스크립터

참고자료:

```
setIMECandidateListStyle
TextField.disableIME
```

setIMECandidateListStyle () 함수

```
public function setIMECandidateListStyle(styleDescriptor:Object)
```

Scaleform version: 2.1.50

전달인자로 전달된 객체의 후보 목록에 후보 목록 스타일을 설정한다.

`styleDescriptor` 는 다음과 같은 멤버를 갖는 객체 인스턴스다.

```
styleObject.textColor                          = 0x5EADFF;
styleObject.selectedTextColor            = 0xFFFFFF;
styleObject.fontSize                           = 20;
styleObject.backgroundColor                    = 0x001430;
styleObject.selectedTextBackgroundColor  = 0x2C5A99;
styleObject.indexBackgroundColor         = 0x12325A;
styleObject.selectedIndexBackgroundColor       = 0x2C5A99;
styleObject.readingWindowTextColor       = 0xFFFFFF;
styleObject.readingWindowBackgroundColor = 0x001430;
styleObject.readingWindowFontSize        = 20;
```

`textColor : Number`

텍스트 색상

`selectedTextColor : Number`

선택한 텍스트 색상

`fontSize : Number`

열 및 열 인덱스에 있는 텍스트의 폰트 크기

`backgroundColor : Number`

선택하지 않은 열의 텍스트 부분 배경색

`seletedTextBackgroundColor : Number`

선택한 열의 텍스트 부분 색상

`indexBackgroundColor : Number`

선택하지 않은 열의 인덱스 부분 배경색

`selectedIndexBackgroundColor : Number`

선택한 열의 인덱스 부분 배경색

`readingWindowTextColor : Number`

판독창 내 텍스트 색상

`readingWindowBackgroundColor : Number`

판독창 배경색

`readingWindowFontSize : Number`

판독창 내 텍스트의 폰트 크기

모든 색상은 0xRRGGBB 포맷으로 정의한다. 여기에서 RR 은 적색 컴포넌트의 16 진수 [0..255], BB 는 청색 컴포넌트의 16 진수 [0...255] 그리고 GG 는 녹색 컴포넌트의 16 진수 [0..255]이다.

참고자료:

`getIMECandidateListStyle`
`TextField.disableIME`

14 3Di 클래스 확장

Scaleform 3Di 는 Scaleform 에 포함된 AS2 익스텐션 기본 셋을 사용하여 실행됩니다. 3Di 에는 다음 ActionScript 익스텐션이 추가되었습니다.

Scaleform version: 3.2.82

_z

오브젝트의 Z 좌표(깊이)를 설정합니다. 초기값은 0 입니다.

_zscale

오브젝트의 스케일을 Z 방향 쪽으로 해서 백분율로 표시합니다. 초기값은 0 입니다.

_xrotation

X(수평)축을 중심으로 오브젝트의 회전을 설정합니다. 초기값은 0 입니다.

_yrotation

Y(수직)축을 중심으로 오브젝트의 회전을 설정합니다. 초기값은 0 입니다.

_matrix3d

16 개의 float 로 구성된 배열(4 x 4)를 사용하여 오브젝트의 3D 변환을 설정합니다. 값이 NULL 로 설정되면 모든 3D 변환이 제거되고 오브젝트가 2D 로 바뀝니다.

_perspfov

오브젝트의 원근 시야(Field Of View)각을 설정하며, 올바른 각도 값은 0 보다 크고 180 보다 작아야 합니다. 설정하지 않으면, 오브젝트는 FOV 각도(초기값 55)를 상속받습니다.

이러한 새 익스텐션을 사용하려면 전역 변수인 gfxExtensions 가 ActionScript 에서 true 로 설정되어야 합니다.

15 표준 메소드 및 이벤트 핸들러 익스텐션

Key 클래스

```
Key.getCode(controllerIdx:Number)
Key.getAscii(controllerIdx:Number)
Key.isDown(controllerIdx:Number)
Key.isToggled(controllerIdx:Number)
```

키보드/컨트롤러의 옵션 파라미터를 취득하는 Key 클래스 메소드입니다. 지정하지 않은 경우, `controllerIdx` 의 초기값은 0 입니다.

```
Key.onKeyDown(controllerIdx:Number)
```

Key 리스너 메소드인 onKeyDown 은 이벤트를 발생시킨 키보드/컨트롤러의 추가 파라미터를 받을 수 있습니다.

Selection 클래스

```
Selection.setFocus(ch:Object, controllerIdx:Number)
Selection.getFocus(controllerIdx:Number)
Selection.getBeginIndex(controllerIdx:Number)
Selection.getEndIndex(controllerIdx:Number)
Selection.setSelection(start:Number, end:Number, controllerIdx:Number)
Selection.getCaretIndex(controllerIdx:Number)
```

키보드/컨트롤러의 옵션 파라미터를 취득하는 Selection 클래스 메소드입니다. 지정하지 않은 경우, controllerIdx 의 초기값은 0 입니다.

```
Selection.onSetFocus(old:Object, new:Object, controllerIdx:Number)
```

Selection 리스너 메소드인 onSetFocus 는 이벤트를 발생시킨 키보드/컨트롤러의 추가 파라미터를 받을 수 있습니다.

MovieClip/Button/TextField

```
MovieClip.onSetFocus(old:Object, controllerIdx:Number)
MovieClip.onKillFocus(new:Object, controllerIdx:Number)
Button.onSetFocus(old:Object, controllerIdx:Number)
Button.onKillFocus(new:Object, controllerIdx:Number)
TextField.onSetFocus(old:Object, controllerIdx:Number)
TextField.onKillFocus(new:Object, controllerIdx:Number)
TextField.onChanged(controllerIdx:Number)
```

이러한 MovieClip/Button/TextField 리스너 메소드는 이벤트를 발생시킨 키보드/컨트롤러의 추가 파라미터를 받을 수 있습니다.

System.capabilities

`System.capabilities.numControllers` – 시스템에서 감지한 controller 의 수를 반환합니다.