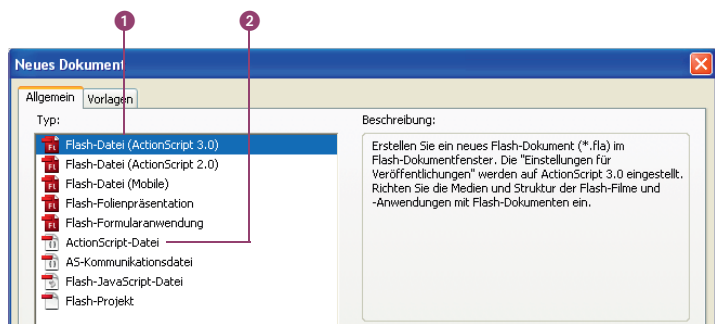


11 ActionScript 3

Dieses Kapitel bietet Ihnen einen ersten Einstieg in ActionScript 3. Sie lernen grundlegende Merkmale der Skriptsprache kennen und lernen, mit ActionScript-3-basierten Flash-Filmen umzugehen.

11.1 Einführung

Im Vergleich zu früheren ActionScript-Versionen bietet ActionScript 3 eine optimierte Leistungsfähigkeit und einen erweiterten Funktionsumfang. Zur Ausführung von Flash-Filmen, die auf ActionScript 3 basieren, wird eine neue so genannte **Virtual Machine** (AVM2) verwendet, die im Flash Player 9 integriert ist. Im Flash Player 9 ist neben der neuen Virtual Machine auch die alte Virtual Machine (AVM1) integriert, sodass auch ActionScript-1- und -2-basierte Flash-Filme im Player abspielbar sind.



Um einen Flash-Film mit ActionScript 3 zu erstellen, klicken Sie im Menü auf DATEI • NEU und wählen im Reiter ALLGEMEIN den Typ FLASH-DATEI (ACTIONSCRIPT 3.0) ❶ aus.

Sollten Sie Klassen in ActionScript 3 erstellen, können Sie per Mausclick auf ACTIONSCRIPT-DATEI ❷ eine externe ActionScript-Datei erzeugen.

ActionScript 3 in Flex

Die Skriptsprache ActionScript 3 wird nicht nur in Flash verwendet. Im Adobe Flex Builder, einer IDE (Integrated Development Environment) zur Entwicklung von RIAs (Rich Internet Applications) wird ActionScript 3 ebenso eingesetzt. Mehr zu Flex erfahren Sie in Kapitel 19, »Ein Blick über den Tellerrand«.

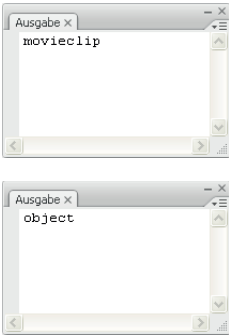
◀ Abbildung 11.1

Flash-Datei (ActionScript 3.0) erstellen

Hinweis

Um die Erläuterungen in diesem Kapitel nachzuvollziehen, sollten Sie zuvor das Kapitel 10, »Einführung in die objektorientierte Programmierung«, lesen.

11.2 Dokumentklasse



▲ **Abbildung 11.2**

Oben: Datentyp der Hauptzeitleiste in AS 1/2, unten: Datentyp der Hauptzeitleiste in AS 3

Vererbung der Dokumentklasse

Die Dokumentklasse muss die Klasse `flash.display.MovieClip` erweitern, wenn der Flash-Film bildbasierte Skripte enthält, die Dokumentklasse den Ablaufvorgang eines Flash-Films steuern soll oder die Bühne Elemente enthält, die mehrere Bilder besitzen. Sollten diese Fälle nicht gegeben sein, kann die Dokumentklasse optional auch die Klasse `flash.display.Sprite` erweitern.

Jeder Flash-Film, der auf ActionScript 3 basiert, besitzt eine so genannte **Dokumentklasse**. Die Hauptzeitleiste eines Flash-Films kann als eine Instanz oder ein Objekt der Dokumentklasse bezeichnet werden. Sichtbar wird das z.B., wenn Sie folgende Anweisung auf einem Schlüsselbild der Hauptzeitleiste platzieren und die Ausgabe in ActionScript 3 mit einem ActionScript-1/2-basierten Flash-Film vergleichen.

```
trace(typeof(this));
```

In ActionScript 1 und 2 ist die Hauptzeitleiste vom Datentyp `movieclip` und verhält sich dementsprechend wie ein `Movieclip`. In ActionScript 3 ist die Hauptzeitleiste vom Datentyp `object`, ein Objekt der Dokumentklasse.

11.2.1 Dokumentklasse definieren

Mithilfe einer eigenen Dokumentklasse können Sie z. B. Methoden und Eigenschaften festlegen, die dann direkt im Flash-Film über das Schlüsselwort `this` referenziert werden können. Wenn Sie keine eigene Dokumentklasse definieren, erzeugt Flash automatisch eine entsprechende Dokumentklasse.

Um eine eigene Dokumentklasse anzulegen, müssen Sie zunächst über **DATEI • NEU...** im Reiter **ALLGEMEIN** eine **ACTIONSCRIPT-DATEI** erzeugen, in der Sie die Klasse definieren. Im folgenden Beispiel wird eine Klasse `MainClass` mit einer entsprechenden Konstruktor-Funktion definiert, die automatisch vom Flash-Film aufgerufen wird, wenn dieser initialisiert wird.

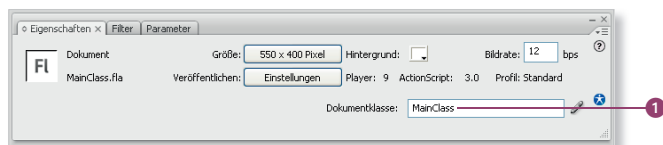
```
package {
    import flash.display.MovieClip
    public class mainClass extends MovieClip{
        public function mainClass() {
            trace("init");
        }
    }
}
```

Speichern Sie das ActionScript-Dokument in einem beliebigen Verzeichnis ab. Achten Sie darauf, dass der Dateiname dem Klassennamen entspricht – in diesem Fall also `MAINCLASS.AS`.

Nachdem die Dokumentklasse definiert ist, müssen Sie dem Flash-Film die Klasse als solche zuweisen. Öffnen Sie dazu zunächst über **DATEI • NEU...** im Reiter **ALLGEMEIN** und **FLASH-**

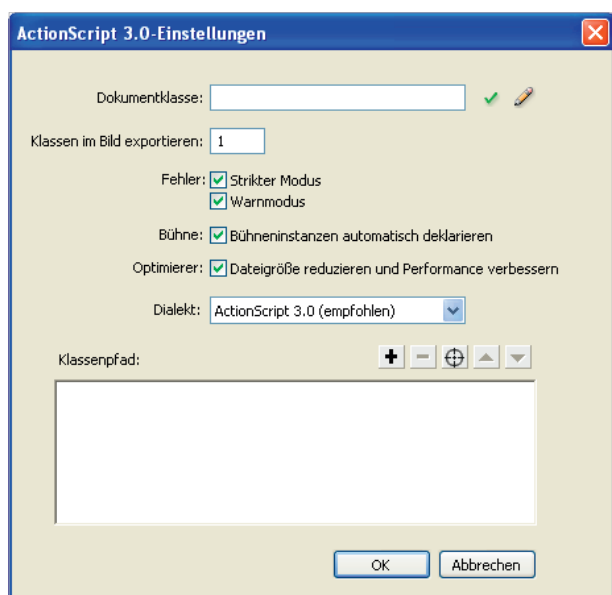
DATEI (ACTIONSCRIPT 3.0) einen neuen Flash-Film. Öffnen Sie anschließend das EIGENSCHAFTEN-Fenster, und geben Sie im Feld DOKUMENTKLASSE den Namen der Klasse, in diesem Beispiel MainClass, ein. Achten Sie darauf, dass Sie nur den Bezeichner ohne die Dateiergung eingeben.

Per Mausklick auf das Stiftsymbol rechts neben dem Eingabefeld ❶ wird die ActionScript-Datei zur angegebenen Dokumentklasse übrigens automatisch in Flash geöffnet.



◀ **Abbildung 11.3**
Dokumentklasse definieren

Alternativ können Sie die Dokumentklasse auch über das Menü DATEI • EINSTELLUNGEN FÜR VERÖFFENTLICHUNGEN... im Reiter FLASH über die Schaltfläche EINSTELLUNGEN unter ACTIONSCRIPT-VERSION einstellen.



◀ **Abbildung 11.4**
Dokumentklasse definieren im
Bereich ACTIONSCRIPT 3.0-EIN-
STELLUNGEN

Speichern Sie den Flash-Film im selben Verzeichnis wie die zuvor erstellte ActionScript-Datei, falls Sie keinen Klassenpfad verwenden möchten. Wenn Sie den Film veröffentlichen, wird die Konstruktor-Methode automatisch aufgerufen. Methoden und Eigenschaften des Objekts der Dokumentklasse lassen sich auf der Hauptzeitleiste des Flash-Films dann direkt ansprechen.

Angenommen, Sie definieren in einer Dokumentklasse eine Methode `openPage`. Diese lässt sich dann über `this.openPage()` ;

aus der Hauptzeitleiste des Flash-Films ansprechen. Der Bezeichner `this` bezieht sich also immer auf das Objekt der Dokumentklasse.

11.3 Anzeigebjekte

Jeder auf ActionScript 3 basierende Flash-Film besitzt eine Hierarchie von so genannten Anzeigebjekten, die auch als **Anzeigeliste** bezeichnet wird. Diese Anzeigeliste enthält alle sichtbaren Elemente eines Flash-Films. Dabei entspricht die Struktur der eines Baumdiagramms. Das oberste Element der Anzeigeliste des Flash-Films ist ein Objekt der Klasse `Stage`. Das Objekt dient als Container für alle Anzeigebjekte. Darunter befindet sich ein Objekt der zuvor beschriebenen Dokumentklasse. Unterhalb des Objekts finden sich dann Anzeigebjekte und so genannte Anzeigebjekt-Container, die zusätzlich zu ihrer eigenen visuellen Darstellung weitere untergeordnete Objekte, wie z. B. andere Anzeigebjekte, enthalten können.

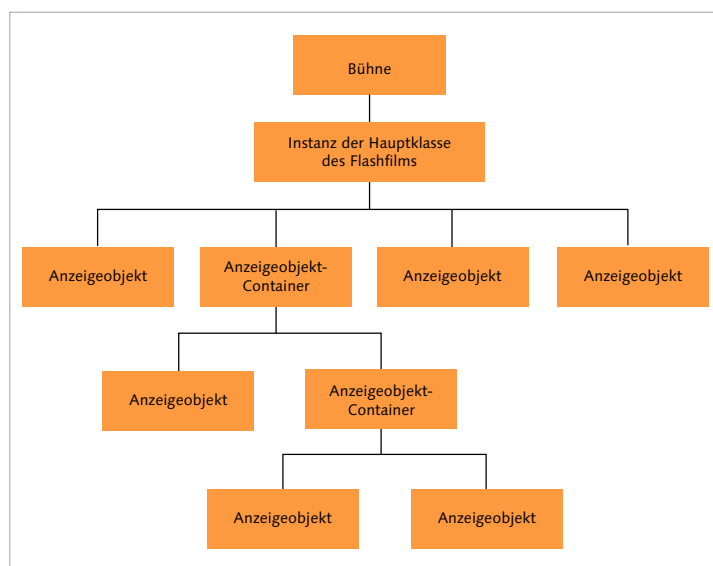


Abbildung 11.5 ►
Baumstruktur: Anzeigebjekte

In ActionScript 1 und 2 werden grafische Elemente, die über ActionScript gesteuert werden sollen, meist in einen Movieclip konvertiert. In ActionScript 3 hingegen gibt es neben Movieclips viele weitere Anzeigebjekte, die sich über ActionScript steuern lassen. Zu den wichtigsten Hauptanzeigeklassen gehören:

- **MovieClip (Anzeigebjekt-Container):** Ein Objekt der Klasse `MovieClip` gleicht einer Instanz eines `Movieclip-Symbols`.

- **Shape:** Mithilfe der Shape-Klasse können Sie Vektorgrafiken wie Kreise, Rechtecke etc. erstellen. Ein Shape-Objekt besitzt im Gegensatz zu einem Movieclip keine Zeitleiste. Da Shape-Objekte grundsätzlich einfacher strukturiert sind als Movieclips, benötigen sie dementsprechend weniger Speicher. Shape-Objekten können keine Maus- oder Tastaturereignisse zugewiesen werden.
- **Loader (Anzeigeobjekt-Container):** Mithilfe der Loader-Klasse lassen sich externe Inhalte wie Flash-Filme (SWF) oder Bitmaps laden.
- **SimpleButton:** Ein Objekt der SimpleButton-Klasse ähnelt einem Schaltflächensymbol. Es besitzt drei Zustände: AUF, GEDRÜCKT und DARÜBER.
- **Sprite (Anzeigeobjekt-Container):** Ein Sprite-Objekt kann Grafiken und untergeordnete Anzeigeobjekte enthalten. Ein Sprite-Objekt besitzt keine Zeitleiste. Sprite-Objekten können Maus- und Tastaturereignisse zugewiesen werden.
- **Bitmap:** Mithilfe eines Bitmap-Objekts lassen sich sowohl externe Bitmaps als auch über ActionScript erzeugte Bitmaps darstellen und steuern.
- **TextField:** Klasse zur Darstellung von Text
- **Video:** Video-Klasse zur Darstellung von Videos
- **AVM1Movie:** Die Klasse dient zur Darstellung von geladenen ActionScript-1/2-basierten Flash-Filmen.
- **Stage (Anzeigeobjekt-Container):** Jeder Flash-Film besitzt eine Instanz (Objekt) der Klasse Stage, über die u.a. auf Eigenschaften der DisplayObject-Instanzen zugegriffen werden kann. Die Bühne ist der grundlegende Anzeigeobjekt-Container an oberster Stelle eines Flash-Films.
- **DisplayObjectContainer:** Eine Basisklasse, die als Container dient. Die Klasse wird von den Klassen Loader, Stage und Sprite erweitert, d.h., Eigenschaften und Methoden der DisplayObjectContainer-Klasse werden von den genannten Klassen geerbt.
- **InteractiveObject:** Eine Basisklasse, die zur Interaktion über Maus und Tastatur verwendet wird.
- **MorphShape:** Wenn Sie in der Entwicklungsumgebung ein Formtweening erstellen, lässt sich über ein Objekt der MorphShape-Klasse darauf zugreifen.

Die folgende Tabelle zeigt die wichtigsten Eigenschaften der DisplayObject-Klasse, die von allen Anzeigeobjekten geerbt werden und deshalb auch mit ihnen verwendet werden können.

DisplayObject-Klasse

Alle Anzeigeobjekte erweitern die so genannte DisplayObject-Klasse. So werden auch Eigenschaften und Methoden der DisplayObject-Klasse vererbt. Zum Beispiel besitzen alle Anzeigeobjekte die Eigenschaft `x` und `y`, über die die Position des Anzeigeobjekts auf der Bühne gesteuert werden kann.

Anzeigeobjekt-Container

Klassen, die die DisplayObject-Container-Klasse erweitern, können neben eigenen grafischen Elementen auch untergeordnete Anzeigeobjekte beinhalten. Sie werden auch als Anzeigeobjekt-Container bezeichnet. So können einem Objekt der Klasse Sprite z. B. mehrere Objekte der Klasse Shape als untergeordnete Objekte zugewiesen werden.

InteractiveObject-Klasse

Die InteractiveObject-Klasse wird von vielen anderen Klassen, wie z. B. der Sprite-Klasse, geerbt. Über Methoden der InteractiveObject-Klasse können dem Objekt dann Maus- und Tastaturereignisse zugewiesen werden.

Eigen-schaft	Beispiel	Beschreibung
alpha	<code>myShape.alpha = 0.5;</code>	Der Alphawert (Transparenz) des Anzeigeobjekts. Wertebereich 0 bis 1 (100 %).
height	<code>mc.height = 100;</code>	Höhe des Anzeigeobjekts
width	<code>mc.width = 50;</code>	Breite des Anzeigeobjekts
name	<code>trace(myShape.name);</code>	Instanzname des Anzeigeobjekts (kann nur für Anzeigeobjekte definiert werden, wenn diese über ActionScript erzeugt wurden)
parent	<code>trace(mc.parent);</code>	Eine Referenz auf das übergeordnete <code>DisplayObjectContainer</code> -Objekt.
rotation	<code>mySprite.rotation = 90;</code>	Rotation (Drehung) des Anzeigeobjekts
scaleX	<code>mySprite.scaleX = 2,0;</code> (200 % Skalierung nach rechts, wenn der Registrierungspunkt rechts ist.) <code>mySprite.scaleX = -2,0;</code> (200 % Skalierung nach links, wenn der Registrierungspunkt links ist.)	Horizontale Skalierung des Anzeigeobjekts, ausgehend vom Registrierungspunkt
scaleY	<code>mySprite.scaleX = 2,0;</code> (200 % Skalierung nach unten, wenn der Registrierungspunkt unten ist.) <code>mySprite.scaleX = -2,0;</code> (200 % Skalierung nach oben, wenn der Registrierungspunkt unten ist.)	Vertikale Skalierung des Anzeigeobjekts, ausgehend vom Registrierungspunkt
stage	<code>trace(mySprite.stage)</code>	Stage-Objekt, in dem sich das Anzeigeobjekt befindet
visible	<code>mc.visible = false;</code>	Gibt an, ob das Anzeigeobjekt sichtbar (<code>true</code>) oder unsichtbar (<code>false</code>) ist.
x	<code>mc.x = 100;</code>	die x-Koordinate des Anzeigeobjekts, relativ zum übergeordneten <code>DisplayObjectContainer</code> -Objekt
y	<code>mc.y = 200;</code>	die y-Koordinate des Anzeigeobjekts, relativ zum übergeordneten <code>DisplayObjectContainer</code> -Objekt

Tabelle 11.1 ►
Eigenschaften der
`DisplayObject`-Klasse

11.3.1 Anzeigeeobjekte in der Anzeigeliste

Einer der vielen Vorteile von ActionScript 3 gegenüber ActionScript 1 und 2 ist, dass alle Anzeigeeobjekte, die in der Entwicklungsumgebung erstellt wurden, über ActionScript ansteuerbar sind. Angenommen, Sie zeichnen in einem neuen Flash-Film ein Rechteck und möchten die Position des Rechtecks über ActionScript ändern.

Zunächst können Sie dazu das Rechteck über die Methode `getChildAt` referenzieren.

Die Methode erwartet als Argument den Index des Objekts in der Anzeigeliste eines Anzeigeeobjektcontainers. Anschließend können Sie z. B. die Position eines Anzeigeeobjekts über die `x`- und `y`-Eigenschaft ändern:

```
var myRect:DisplayObject = this.getChildAt(0);
myRect.x = 200;
myRect.y = 100;
```

Alternativ können Sie ein Anzeigeeobjekt auch über die Methode `getChildByName` referenzieren. Jedes Anzeigeeobjekt besitzt eine `name`-Eigenschaft, die Sie selbst zuweisen können. Sollten Sie die Eigenschaft nicht definieren, wird dem Objekt automatisch ein Name, wie z. B. `instance1`, zugewiesen.

Angenommen, Sie haben einen Movieclip erstellt und dem Movieclip im EIGENSCHAFTEN-Fenster den Instanznamen `mc` zugewiesen.

Anschließend können Sie den Movieclip wie folgt referenzieren:

```
var meinMovieClip:flash.display.DisplayObject = this.
  getChildByName("mc");
meinMovieClip.x = 300;
```

Dabei wird der Eigenschaft `name` des Anzeigeeobjekts automatisch der Instanzname zugewiesen. Der folgende Code führt z. B. zur Ausgabe `mc`.

```
trace(this.getChildAt(0).name);
```

11.3.2 Anzeigeeobjekte hinzufügen und entfernen

Wenn Sie ein Anzeigeeobjekt auf der Bühne platzieren möchten, müssen Sie das Anzeigeeobjekt einem Anzeigeeobjekt-Container hinzufügen. Angenommen, Sie möchten auf der Bühne ein Textfeld erstellen. Zunächst müssen Sie dazu das Anzeigeeobjekt initialisieren. Dazu müssen Sie die Klasse `TextField` des Pakets

[X] Anzeigeeobjekt-Eigenschaften

Beachten Sie, dass Eigenschaften von Anzeigeeobjekten in ActionScript 3 ohne den Unterstrich »_« geschrieben werden. Der folgende Vergleich zeigt die unterschiedliche Schreibweise:

ActionScript 1 und 2:

```
mc._x = 300;
```

ActionScript 3:

```
mc.x = 300;
```

Datentyp überprüfen

Über den so genannten `is`-Operator können Sie überprüfen, ob ein Anzeigeeobjekt kompatibel zu einem bestimmten Datentyp ist. Der folgende Code prüft z. B., ob das Element mit dem Index 0 in der Anzeigeliste kompatibel zum Datentyp `MovieClip` ist:

```
if(this.getChildAt(0) is
  MovieClip) {
  trace("Ich bin ein
  MovieClip");
}
```

flash.display importieren. Anschließend können Sie ein Textfeld referenzieren und über die Methode addChild einem Anzeigebereich-Container zuweisen. Durch diese Zuweisung wird das Textfeld dann auf der Bühne platziert.

```
import flash.text.TextField;
var meinTextfeld:TextField = new TextField();
meinTextfeld.text = "Hallo Welt";
this.addChild(meinTextfeld);
```

Klassen importieren

Bevor Sie ein Objekt einer Klasse initialisieren können, müssen Sie die Klasse oder das gesamte Paket, das die Klasse enthält, über die import-Anweisung importieren. Um z. B. ein Objekt der Klasse TextField erzeugen zu können, müssen Sie die Klasse zunächst importieren:

```
import flash.text.TextField;
```

Alternativ können Sie auch alle Klassen des Pakets importieren:

```
import flash.text.*;
```

Beachten Sie, dass das Textfeld ohne die letzte Zeile nicht auf der Bühne platziert werden würde. Es existiert zwar, übrigens auch im Speicher, befindet sich jedoch außerhalb der sichtbaren Anzeigeliste des Flash-Films.

Alternativ können Sie über die Methode addChildAt ein Anzeigebereichsobjekt an einem bestimmten Index in der Anzeigeliste eines Anzeigebereichscontainers einfügen. Angenommen, Sie möchten ein Textfeld oberhalb eines Sprites, das z. B. einen Hintergrund für das Textfeld beinhaltet, platzieren. Das Textfeld wird allerdings als Erstes instanziiert. Sie können dann das Sprite-Objekt nachträglich auf dem Index des Textfeldes positionieren. Folgender Code sorgt dafür, dass das Sprite-Objekt meinSprite auf dem Index 0 positioniert wird:

```
import flash.display.*;
import flash.text.TextField;
var meinTextfeld:TextField = new TextField();
meinTextfeld.text = "Hallo Welt";
this.addChild(meinTextfeld);
var meinSprite:Sprite = new Sprite();
this.addChildAt(meinSprite, 0);
```

Indexposition bestimmen

Über die Methode getChildIndex() können Sie den Index eines Elements in der Anzeigeliste eines DisplayObjectContainers bestimmen. Über folgende Anweisung würden Sie den Index eines Sprite-Objekts meinSprite bestimmen:

```
trace(this.
getChildIndex(meinSprite));
```

Anzeigeobjekt entfernen | Über die Methode removeChild lässt sich ein Anzeigeobjekt aus der sichtbaren Anzeigeliste auch wieder entfernen:

```
this.removeChild(meinTextfeld);
```

Dabei wird der Index jedes Elements der Anzeigeliste, das über dem entfernten Element liegt, automatisch um 1 reduziert.

Alternativ können Sie ein Element an einem bestimmten Index über die Methode removeChildAt entfernen. Die folgende Anweisung entfernt z. B. das Element mit dem Index 3:

```
this.removeChildAt(3);
```

11.3.3 Reihenfolge in der Anzeigeliste ändern

Der Index eines Anzeigeobjekts in einer Anzeigeliste eines Anzeigeobjekt-Containers bestimmt, ob das Objekt über- oder unterhalb anderer Objekten der Anzeigeliste dargestellt wird. Angenommen, Sie haben die beiden Movieclips »mc0« und »mc1« auf der Bühne platziert **1** und möchten die Reihenfolge entsprechend ändern **2**.

Sie könnten die Reihenfolge über zwei Methoden entsprechend ändern:

- **setChildIndex:** Über die Methode `setChildIndex` können Sie den Index eines Elements neu setzen. So würde folgende Anweisung dazu führen, dass der Movieclip »mc0« auf dem Index 1 positioniert wird. Dem Movieclip »mc1« wird dann automatisch der Index 0 zugewiesen.

```
this.setChildIndex(mc0,1);
```

- **swapChildren:** Über die Methode `swapChildren` können Sie den Index zweier Elemente vertauschen.

```
this.swapChildren(mc0,mc1);
```

11.3.4 Movieclips

Viele Flash-Nutzer, die von ActionScript 1/2 auf ActionScript 3 umsteigen, fragen sich zu Beginn oft, wie sie eine Instanz eines Movieclip-Symbols aus der Bibliothek erzeugen und anschließend über ActionScript auf die Instanz zugreifen können.

Angenommen, Sie möchten einen Movieclip, der in der Bibliothek liegt, zur Laufzeit auf der Bühne positionieren. Wählen Sie dazu den Movieclip in der Bibliothek aus, öffnen Sie über die rechte Maustaste das Kontextmenü, und wählen Sie den Menüpunkt VERKNÜPFUNG **1**.

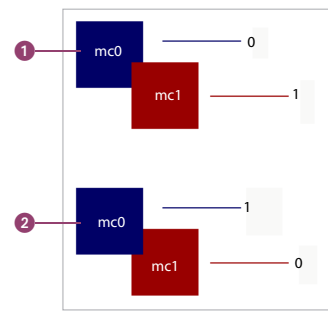
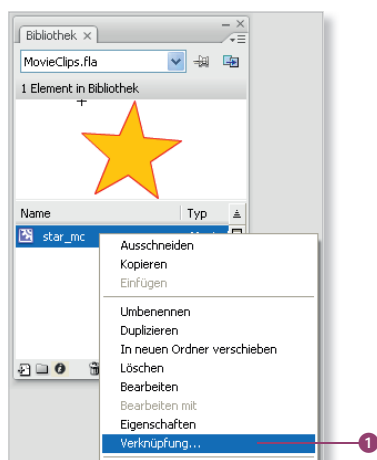


Abbildung 11.6 ▲
Indexreihenfolge ändern

◀ Abbildung 11.7
VERKNÜPFUNG ...



Klassen als public definieren

Wenn Sie in ActionScript 3 eine Klasse definieren, müssen Sie die Klasse über das Schlüsselwort `public` kennzeichnen, wenn sie öffentlich zugänglich sein soll.

Abbildung 11.8 ►

Verknüpfungseigenschaften

Klassenpakete

Klassenpakete werden in ActionScript 3 über das Schlüsselwort `package` definiert. Angenommen, Sie erstellen eine eigene Klasse `com.username.MyClass`.

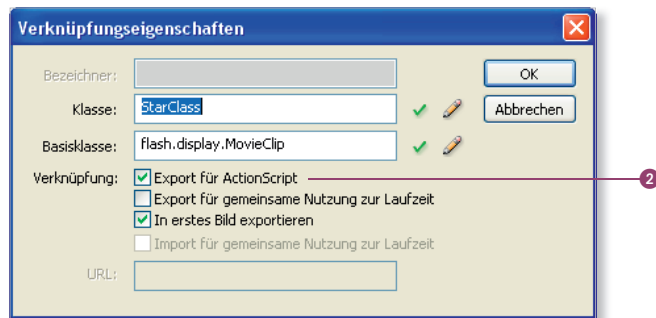
Innerhalb der Klasse wird dann der Bezeichner des Pakets wie folgt angegeben:

```
package com.username {  
    public class MyClass {  
        ...  
    }  
}
```

Diese Schreibweise unterscheidet sich von ActionScript 2. Nähere Erläuterungen zu Klassenpaketen finden Sie in Kapitel 10, »Einführung in die objektorientierte Programmierung«.

Aktivieren Sie anschließend die Option **EXPORT FÜR ACTIONSCRIPT**. Die Felder **KLASSE** und **BASISKLASSE** werden automatisch aktiviert. Unter **KLASSE** geben Sie den Bezeichner der Klasse ein, die mit dem Movieclip verknüpft wird. Über die angegebene Klasse können Sie den Movieclip in ActionScript 3 referenzieren, indem Sie ein Objekt der Klasse erzeugen. Dazu mehr in Kürze.

Unter **BASISKLASSE** wird automatisch der Wert `FLASH.DISPLAY.MOVIECLIP` eingetragen. Sie können hier optional auch eine eigene Klasse, die die `Sprite`- oder `MovieClip`-Klasse erweitert, eingeben. Durch eine eigene Klasse könnten Sie die Funktionalität des Movieclips beliebig erweitern. Sollten Sie die Instanz des Symbols in ActionScript erstellen, ohne weitere Funktionen einzufügen, können Sie den vorgegebenen Klassennamen beibehalten.



Sollten Sie alle Einstellungen so belassen, erscheint ein Warnhinweis. Die Meldung weist Sie darauf hin, dass es bisher keine Klasse mit dem angegebenen Namen gibt und diese automatisch in den Flash-Film integriert wird. Sie können diesen Hinweis mit einem Klick auf **OK** erst mal ignorieren. Wenn Sie keine eigene Klasse erstellt haben, wird automatisch eine Klasse mit folgendem Aufbau in den Flash-Film integriert:

```
package {  
    import flash.display.MovieClip;  
    public class StarClass extends MovieClip {  
        public function StarClass() {  
        }  
    }  
}
```

Anschließend können Sie eine neue Movieclip-Instanz durch die Initialisierung eines Objekts der dazugehörigen Klasse erzeugen und wie folgt auf der Bühne platzieren:

```
var myStar_mc:StarClass = new StarClass();
this.addChild(myStar_mc);
```

Mehrere Instanzen | Natürlich können Sie auch mehrere Instanzen der eigenen Klasse zur Anzeigeliste hinzufügen. Dazu folgen des Beispiel.

```
for (var i:int; i<10; i++) {
var myStar_mc:StarClass = new StarClass();
this.addChildAt(myStar_mc,i);
var stageWidth:int = myStar_mc.stage.stageWidth;
var stageHeight:int = myStar_mc.stage.stageHeight;
myStar_mc.x = randomExt(75,stageWidth-75);
myStar_mc.y = randomExt(75,stageHeight-75);
}
```

Das vollständige Beispiel finden Sie auf der DVD unter KAPITEL_11 • MOVIECLIPS • MOVIECLIPS_02.FLA.

package-Schlüsselwort

Das package-Schlüsselwort muss in ActionScript-3-Klassen immer angegeben werden, selbst wenn die Klasse zu keinem Paket gehört. Verwenden Sie die Anweisung dann einfach ohne einen Bezeichner, z. B.:

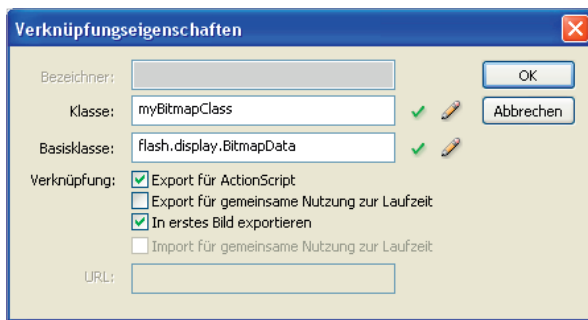
```
package {
public class TestClass {
...
}
}
```



KAPITEL_11 • MOVIECLIPS •
MOVIECLIPS_02.FLA

11.3.5 Bitmaps

Ähnlich wie auf Movieclips können Sie auch auf Bitmaps in der Bibliothek zugreifen. Dazu weisen Sie über das Kontextmenü VERKNÜPFUNG... der Bibliothek der Bitmap zunächst im Feld KLASSE eine Klasse zu. Im Feld BASISKLASSE wird automatisch die Klasse FLASH.DISPLAY.BITMAPDATA eingetragen.



◀ **Abbildung 11.9**
Verknüpfungseigenschaften einer Bitmap

Anschließend können Sie ein Objekt der Klasse im Flash-Film wie folgt erzeugen:

```
var myBitmap:BitmapClass = new BitmapClass(0, 0);
```

Wenn Sie eine Bitmap in ActionScript erzeugen, müssen Sie die Breite und die Höhe der Bitmap bei der Erzeugung angeben. Da die Breite und die Höhe in diesem Beispiel der Breite und der Höhe der Bitmap entsprechen, müssen Sie die Felder nur pro



forma ausfüllen. Bei einer über ActionScript erzeugten Bitmap kann die Größe zur Laufzeit variieren. Sie können hier die Werte 0,0 verwenden.

Anschließend können Sie ein neues Bitmap-Objekt erzeugen und dem Objekt das zuvor erzeugte Objekt als Quelle zuweisen. Das Bitmap-Objekt wird anschließend zur Anzeigeliste hinzugefügt.

```
var pic:Bitmap = new Bitmap(myBitmap);
addChild(pic);
```

11.3.6 Anzeigebereich-Container

Anzeigebereich-Container können zur Strukturierung eines Flash-Films verwendet werden. Der grundlegende Anzeigebereich-Container eines Flash-Films ist ein Objekt der Klasse Stage.

Die Klasse für Anzeigebereich-Container – DisplayObjectContainer – gehört zum Paket flash.display. Die Klassen Loader, Sprite, MovieClip und Stage sind Unterklassen der DisplayObjectContainer-Klasse und können als solche eingesetzt werden.

Angenommen, Sie möchten zwei Objekte der Klasse Shape erzeugen und beide Objekte in einem Anzeigebereich-Container platzieren, z. B. um diese gemeinsam über den Anzeigebereich-Container steuern zu können. Dazu könnten Sie die Methode addChild des Anzeigebereich-Containers, z. B. eines Sprite-Objekts, wie folgt nutzen:

```
import flash.display.*;
var shape0:Shape = new Shape();
var shape1:Shape = new Shape();
var container:Sprite = new Sprite();
container.addChild(shape0);
container.addChild(shape1);
```

Das Stage-Objekt

Der Geltungsbereich des Stage-Objekts ist nicht, wie in ActionScript 1/2, global. Wenn Sie auf Eigenschaften und Methoden des Stage-Objekts zugreifen möchten, können Sie das Objekt über die Eigenschaft stage eines Anzeigebereichs, das sich in der Anzeigeliste befindet, referenzieren. Beachten Sie, dass die Eigenschaft stage kleingeschrieben wird.

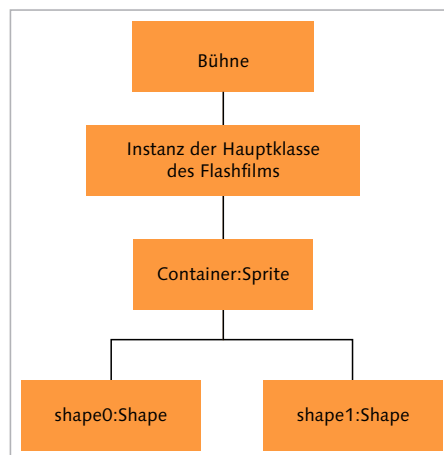
```
var myTextField:TextField =
new TextField();

myTextField.text = "Textbeispiel";

this.addChild(myTextField);

trace(myTextField.stage.
stageWidth+"x"+myTextField.
stage.stageHeight);
```

Abbildung 11.10 ►
Die gewünschte Struktur



11.4 Loader-Klasse

Mithilfe der Loader-Klasse können Sie externe Inhalte in ein Anzeigeobjekt laden und darstellen. Folgende Inhalte können mit einem Objekt der Klasse Loader geladen werden.

- Flash-Filme: Ein auf ActionScript 1 und 2 basierender Flash-Film oder ein auf ActionScript 3 basierender Flash-Film.
- Bitmap-Grafiken: Dazu gehören JPEG-, PNG-, und GIF-Dateien.

Um eine Bitmap-Grafik zu laden, benötigen Sie zunächst ein Objekt der Klasse Loader. Anschließend definieren Sie einen so genannten URLRequest, der an die Methode load des Loader-Objekts übergeben wird. Damit die Bitmap-Grafik im Flash-Film dargestellt wird, wird das Loader-Objekt zur Anzeigeliste hinzugefügt:

```
var picLoader:Loader = new Loader();
var picRequest:URLRequest = new URLRequest("bild.png");
picLoader.load(picRequest);
this.addChild(picLoader);
```

11.4.1 Ladevorgang kontrollieren

Der Ladevorgang kann über ein so genanntes LoaderInfo-Objekt kontrolliert werden. Sobald der Ladevorgang gestartet wurde, wird automatisch ein LoaderInfo-Objekt erzeugt, das sich über die Eigenschaft contentLoaderInfo des Loader-Objekts ansprechen lässt. Das Objekt contentLoaderInfo kann mithilfe von Ereignis-Listenern auf verschiedene Ereignisse reagieren.

Angenommen, Sie möchten ein Bild auf der Bühne positionieren, sobald das Bild geladen wurde. Dazu lässt sich das Ereignis complete wie folgt nutzen:

```
var picLoader:Loader = new Loader();
var picRequest:URLRequest = new URLRequest("bild.png");
picLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, posPic);
picLoader.load(picRequest);
function posPic(event:Event):void {
    this.addChild(picLoader);
    picLoader.content.x = 10;
    picLoader.content.y = 10;
}
```



KAPITEL_11 • LOADER-
KLASSE • BILD_LADEN.FLA

Hinweis

Beachten Sie, dass Sie den Inhalt des Loader-Objekts, in diesem Fall die Bitmap-Grafik, über die Eigenschaft content referenzieren können. Alternativ könnten Sie z. B. auch ein Sprite-Objekt, das als Container dient, verwenden.

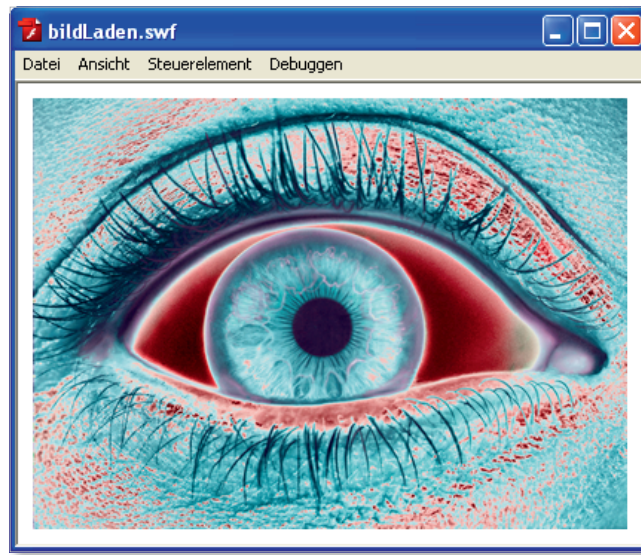


Abbildung 11.11 ►
Bild positionieren

▼ Tabelle 11.2

Ereignisse des LoaderInfo-Objekts

In der folgenden Tabelle werden die wichtigsten Ereignisse des LoaderInfo-Objekts erläutert, über die Sie den Ladestatus kontrollieren können.

Ereignis	Beispiel	Beschreibung
complete	<pre>myLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, completeHandler); myLoader.load(picRequest); function completeHandler(event:Event):void { trace("Inhalt geladen"); }</pre>	Wird aufgerufen, wenn der Inhalt vollständig geladen wurde.
init	<pre>myLoader.contentLoaderInfo.addEventListener(Event.INIT, initHandler); myLoader.load(picRequest); function initHandler(event:Event):void { trace("Inhalt kann angesteuert werden"); }</pre>	Wird aufgerufen, sobald auf Eigenschaften des geladenen Inhalts zugegriffen werden kann. Kann z. B. auch zur Positionierung verwendet werden.
ioError	<pre>myLoader.contentLoaderInfo. addEventListener(IOErrorEvent.IO_ERROR, errorHandler); myLoader.load(picRequest); function errorHandler(event:IOErrorEvent):void { trace("Inhalt konnte nicht geladen werden"); }</pre>	Wird aufgerufen, wenn ein Eingabe- oder Ausgabefehler auftritt, z. B. wenn der geladene Inhalt nicht existiert.
open	<pre>myLoader.contentLoaderInfo.addEventListener(Event.OPEN, openHandler); myLoader.load(picRequest); function openHandler(event:Event):void { trace("Ladevorgang gestartet"); }</pre>	Wird aufgerufen, sobald der Ladevorgang gestartet wird.

Ereignis	Beispiel	Beschreibung
progress	<pre>myLoader.contentLoaderInfo. addEventListener(ProgressEvent.PROGRESS, progressHandler); myLoader.load(picRequest); function progressHandler(event:ProgressEvent):void { var geladen:uint = event.bytesLoaded; var total:uint = event.bytesTotal; var prozent:uint = Math.round((geladen/total)*100); trace(prozent + "% geladen"); }</pre>	Wird regelmäßig aufgerufen, sobald Daten empfangen werden.
unload	<pre>myLoader.contentLoaderInfo.addEventListener(Event. UNLOAD, unloadHandler); myLoader.contentLoaderInfo.addEventListener(Event. COMPLETE, completeHandler); myLoader.load(picRequest); function unloadHandler(event:Event):void { trace("Inhalt wurde entfernt"); } function completeHandler(event:Event):void { myLoader.unload(); }</pre>	Wird aufgerufen, wenn der Inhalt des Loader-Objekts über die Methode unload entfernt wird oder wenn ein geladener Inhalt durch einen anderen ersetzt wird.

▲ **Tabelle 11.2**

Ereignisse des LoaderInfo-Objekts (Fortsetzung)

11.5 Shape-Objekt: Zeichnungs-API

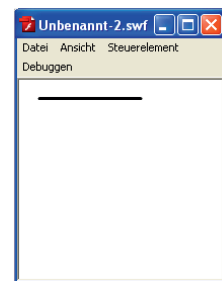
Über die Zeichnungs-API lassen sich Vektorgrafiken über ActionScript zur Laufzeit erzeugen. Zur Erzeugung von Vektorgrafiken können Sie in ActionScript 3 die Graphics-Klasse nutzen. Jedes Shape-, Sprite-, und MovieClip-Objekt besitzt eine Eigenschaft graphics, bei der es sich um eine Instanz der Klasse Graphics handelt.

11.5.1 Linien zeichnen

Bevor Sie in einem Anzeigebereich eine Linie zeichnen können, müssen Sie den Linienstil über die Methode `lineStyle` definieren. Die Methode hat folgenden vereinfachten formellen Aufbau:

```
lineStyle(Strichstärke, Farbe, Alpha);
```

Angenommen, Sie möchten in einem Shape-Objekt `myShape` eine schwarze Linie mit Alpha 50% und einer Strichstärke von drei Pixeln zeichnen. Zunächst initialisieren Sie dazu ein Shape-Objekt und setzen die Eigenschaft des Linienstils entsprechend.



▲ **Abbildung 11.12**

Eine mit ActionScript erzeugte Linie

[x] Wertebereich

Beachten Sie, dass der Wertebereich der Eigenschaften `scaleX`, `scaleY` und `alpha` etc. von Anzeigebildern in ActionScript 3 von 0 bis 1 und nicht, wie bei ActionScript 1/2, von 0 bis 100 geht.

Gezeichnete Formen löschen

Um alle über ActionScript gezeichneten Formen innerhalb eines Anzeigebilds zu löschen, können Sie die Methode `clear` der `Graphics`-Klasse nutzen. Angenommen, Sie haben eine Form in einem `Shape`-Objekt `myShape` erstellt und möchten die Form entfernen:

```
myShape.graphics.clear();
```

Rechteck mit Eckrundungen

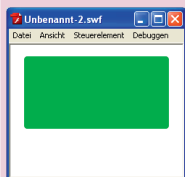
Über die Methode `drawRoundRect` können Sie ein Rechteck mit Eckrundungen zeichnen. Eine vereinfachte formelle Syntax der Methode lautet wie folgt:

```
drawRoundRect(x,y,Breite,
Höhe,Höhe der Eckrundung,
Breite der Eckrundung);
```

Dabei entspricht die Höhe bzw. die Breite der Eckrundungen der Höhe bzw. der Breite der Ellipse, die zur Definition der Eckrundungen angegeben wurde.

Folgender Code führt zum Rechteck in Abbildung 11.13.

```
myShape.graphics.
drawRound-
Rect(0,0,200,100,10,10);
```



▲ **Abbildung 11.13**
Rechteck mit Eckrundung

```
import flash.display.*;
var myShape:Shape = new Shape();
myShape.graphics.lineStyle(3,0x000000,0.5);
```

Anschließend können Sie über die Methode `moveTo` die Anfangsposition eines fiktiven Zeichenstifts, der die Linie zeichnet, bestimmen. Über die Methode `lineTo` zeichnen Sie dann eine Linie, ausgehend von der Anfangsposition bis hin zu den angegebenen Koordinaten.

Damit das `Shape`-Objekt bzw. die Linie auf der Bühne dargestellt wird, müssen Sie das Objekt zur Anzeigeliste hinzufügen. Die Position des `Shape`-Objekts können Sie dann nachträglich über die `x`- und `y`-Eigenschaft bestimmen:

```
myShape.graphics.moveTo(0,0);
myShape.graphics.lineTo(100,0);
this.addChild(myShape);
myShape.x = 20;
myShape.y = 20;
```

11.5.2 Füllungen und Formen erzeugen

Über die Methode `beginFill` können Sie eine Form mit einer einfarbigen Füllung versehen. Dazu müssen Sie, ähnlich wie bei `lineStyle`, zunächst entsprechende Eigenschaften definieren. Angenommen, Sie möchten eine Form mit einer roten Füllfarbe mit einem Alphawert von 70% füllen. Zunächst werden dazu entsprechende Eigenschaften definiert:

```
import flash.display.*;
var myShape:Shape = new Shape();
myShape.graphics.beginFill(0xCC0000,0.7);
```

Rechteck zeichnen | Anschließend können Sie z. B. über die integrierte Methode `drawRect` ein Rechteck zeichnen, das mit der angegebenen Füllung gefüllt wird.

Eine vereinfachte formelle Syntax der Methode `drawRect` lautet wie folgt:

```
drawRect(x,y,Breite,Höhe);
```

Beachten Sie dabei, dass Sie die Füllung über die Methode `endFill` zum Schluss beenden sollten, um zu vermeiden, dass in der Folge erzeugte Formen ebenfalls mit der angegebenen Füllung gefüllt werden.

```
import flash.display.*;
var myShape:Shape = new Shape();
myShape.graphics.beginFill(0xCC0000,0.7);
myShape.graphics.drawRect(10,10,200,150);
myShape.graphics.endFill();
this.addChild(myShape);
```

Kreis zeichnen | Ähnlich können Sie über die Methode `drawCircle` einen Kreis zeichnen. Eine vereinfachte formelle Syntax der Methode lautet wie folgt:

```
drawCircle(x,y,Radius);
```

Durch folgenden Code würde ein Kreis mit einem Radius von 100 Pixeln, einer schwarzen Strichfarbe und einer grünen Füllfarbe erzeugt. Beachten Sie, dass sich die x- und y-Koordinaten auf den Mittelpunkt des Kreises beziehen:

```
import flash.display.*;
var myShape:Shape = new Shape();
myShape.graphics.lineStyle(2,0x000000,1);
myShape.graphics.beginFill(0x00CC00,1);
myShape.graphics.drawCircle(100,100,50);
myShape.graphics.endFill();
this.addChild(myShape);
```

Ellipse zeichnen | Mithilfe der Methode `drawEllipse` können Sie sehr einfach auch Ellipsen zeichnen. Die Methode besitzt folgende vereinfachte formelle Syntax:

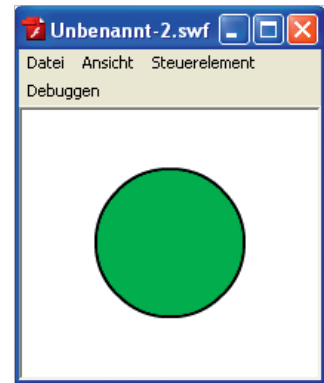
```
drawEllipse(x,y,Breite,Höhe);
```

Durch folgenden Code würden Sie z.B. eine Ellipse mit einer Breite von 150 Pixeln und einer Höhe von 100 Pixeln erzeugen:

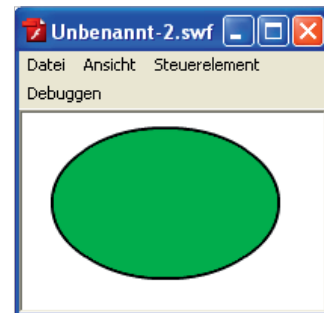
```
import flash.display.*;
var myShape:Shape = new Shape();
myShape.graphics.lineStyle(2,0x000000,1);
myShape.graphics.beginFill(0x00CC00,1);
myShape.graphics.drawEllipse(20,20,150,100);
myShape.graphics.endFill();
this.addChild(myShape);
```



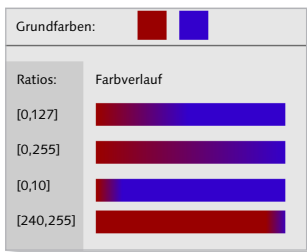
▲ **Abbildung 11.14**
Ein gefülltes Rechteck



▲ **Abbildung 11.15**
Kreis zeichnen



▲ **Abbildung 11.16**
Ellipse zeichnen



▲ **Abbildung 11.17**
Lineare Farbverläufe und entsprechende Ratios

Zeichnen mit ActionScript 1/2

In ActionScript 1/2 können Sie einige der hier genannten Zeichenfunktionen auf ähnliche Weise nutzen. Eigenschaften und Methoden für das Zeichnen gehören in ActionScript 1/2 allerdings zur MovieClip-Klasse.

Das folgende Beispiel zeigt die Anwendung in ActionScript 1/2:

```
this.createEmptyMovieClip(
    "container",1);
container.
lineStyle(1,0x000000,100);
container.
beginFill(0xFFCC00,100);
container.moveTo(0,0);
container.lineTo(100,0);
container.lineTo(100,100);
container.lineTo(0,100);
container.lineTo(0,0);
container.endFill();
```

11.5.3 Farbverlaufslinien und -füllungen erzeugen

Die Erzeugung einer Farbverlaufslinie oder -füllung ist vergleichsweise etwas komplizierter. Über die Methode `lineGradientStyle` legen Sie Eigenschaften für eine Farbverlaufslinie fest. Über die Methode `beginGradientFill` legen Sie Eigenschaften für einen Farbverlauf fest. Beide Methoden erwarten mindestens folgende Argumente:

- ▶ **Fülltyp:** Legt fest, ob ein linearer (`GradientType.LINEAR`) oder ein radialer Farbverlauf (`GradientType.RADIAL`) verwendet wird.
- ▶ **Farben:** Ein Array mit hexadezimalen RGB-Farbwerten, die im Farbverlauf verwendet werden, wie z. B. `0x000000` (Schwarz) und `0xFFFFFF` (Weiß).
- ▶ **Alphas:** Ein Array mit Alphawerten für die unter Farben angegebenen Farbwerte.
- ▶ **Ratios:** Ein Array mit Farbverteilungsverhältnissen der angegebenen Farbe. Der Wertebereich liegt zwischen 0 und 255. Der Wert 0 entspricht der linken Seite des Farbfeldes. Der Wert 255 entspricht der rechten Seite des Farbfeldes (siehe Abbildung 11.17).
- ▶ **Matrix:** Eine Transformationsmatrix, die sich vereinfacht über die Methode `createGradientBox` der Matrixklasse erzeugen lässt. Über die Matrix bestimmen Sie u. a. den Winkel des Farbverlaufs.

Matrix für Farbverläufe

Bevor Sie eine Matrix für Farbverläufe erzeugen können, müssen Sie die Matrixklasse importieren. Anschließend können Sie über die Methode `createGradientBox` einen Matrixstil erstellen, der sich für Farbverläufe eignet. Die Methode erwartet mindestens folgende Argumente:

- ▶ Breite des Farbverlaufsfelds
- ▶ Höhe des Farbverlaufsfelds
- ▶ Rotation des Farbverlaufs (im Bogenmaß (Radiant))

Führen Sie folgende Schritte durch, um einen Matrixstil für einen Farbverlauf zu erstellen.

1. Initialisieren Sie ein Objekt der Klasse `Matrix`.

```
import flash.geom.Matrix;
var myMatrix:Matrix = new Matrix();
```

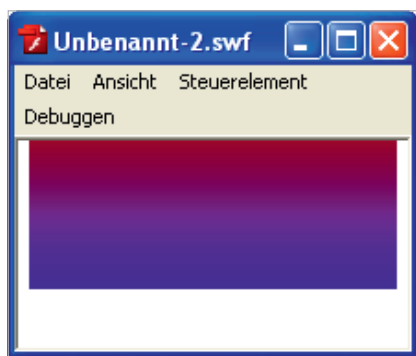
2. Erzeugen Sie einen Matrixstil über die Methode `createGradientBox`. Im Folgenden wird ein Matrixstil für einen 200 × 100 Pixel großen Farbverlauf erzeugt. Dabei wird der Farbverlauf um 90 Grad gedreht:

```
var radiant:Number = 90*(Math.PI/180);
myMatrix.createGradientBox(200,100,radiant);
```

Angenommen, Sie möchten ein 200 × 100 Pixel großes Rechteck mit einem linearen Farbverlauf füllen. Der Verlauf soll sich aus einem Rotton (0x990000) und einem Blauton (0x3300CC) zusammensetzen. Das Farbverteilungsverhältnis wird so eingestellt, dass beide Farben den gleichen Anteil besitzen (0,255). Über die Matrix soll der Farbverlauf um 90 Grad gedreht werden, sodass er von oben (Rot) nach unten (Blau) verläuft.

Folgender Code würde ein Rechteck erzeugen und es mit dem genannten Farbverlauf füllen:

```
import flash.display.*;
import flash.geom.*;
var typ:String = GradientType.LINEAR;
var colors:Array = new Array(0x990000, 0x3300CC);
var alphas:Array = new Array(1, 1);
var ratios:Array = new Array(0, 255);
var myMatrix:Matrix = new Matrix();
var radiant:Number = 90*(Math.PI/180);
myMatrix.createGradientBox(200,100,radiant);
var myShape:Shape = new Shape();
myShape.graphics.beginGradientFill(typ,colors,
    alphas,ratios,myMatrix);
myShape.graphics.drawRect(0,0,200,100);
myShape.graphics.endFill();
this.addChild(myShape);
```



◀ **Abbildung 11.18**

Rechteck mit linearem und um 90 Grad gedrehtem Farbverlauf

11.6 InteractiveObject-Klasse

Wie zuvor erwähnt, ist die `InteractiveObject`-Klasse eine Basisklasse für Anzeigeobjekte, die über die Maus oder die Tastatur gesteuert werden können. Über ein Ereignis lässt sich so z. B. einem `Sprite`-Objekt ein `Button`-Ereignis wie ein Mausklick zuweisen und abfragen.

[x] Shape-Objekt

Beachten Sie, dass die Klasse Shape die Basisklasse zur Steuerung von Maus- und Tastatureingaben – InteractiveObject – nicht erbt. Es ist deshalb nicht direkt möglich, Tastatur- oder Mausabfragen für Shape-Objekte abzufragen. Verschachteln Sie deswegen das Shape-Objekt in einem Sprite-Objekt, das Methoden und Eigenschaften der Klasse InteractiveObject erbt.

Schreibweise: void

Beachten Sie, dass der erste Buchstabe des void-Operators in ActionScript 3 kleingeschrieben wird. In ActionScript 2 wird der erste Buchstabe großgeschrieben.

Angenommen, Sie möchten prüfen, ob der Mauszeiger über ein Sprite-Objekt bewegt wird, das ein Objekt vom Typ Shape beinhaltet, z. B. ein Rechteck. Zunächst müssen Sie das Sprite-Objekt zur Anzeigeliste hinzufügen und anschließend ein entsprechendes Shape-Objekt definieren und dem Sprite-Objekt als Unterelement zuweisen.

```
var myButton:Sprite = new Sprite();
this.addChild(myButton);
var myForm:Shape = new Shape();
myForm.graphics.beginFill(0x00CC00,100);
myForm.graphics.drawRect(0,0,140,30);
myForm.graphics.endFill();
myButton.addChild(myForm);
```

Anschließend können Sie dem Sprite-Objekt über die Methode addEventListener eine Ereignisprozedur zuweisen, die aufgerufen wird, wenn das entsprechende Ereignis auftritt. In diesem Fall, wenn die Maus über das Sprite-Objekt bewegt wird.

```
myButton.addEventListener(MouseEvent.CLICK,
myClickHandler);
function myClickHandler(event:MouseEvent):void
    trace("over");
}
```

In der folgenden Tabelle sind die wichtigsten Ereignisse der InteractiveObject-Klasse aufgeführt.

Ereignis	Beispiel	Beschreibung
click	myButton.addEventListener(MouseEvent.CLICK, clickHandler);	Wird ausgelöst, wenn der Benutzer die Haupttaste des Eingabegeräts (Maus) über dem InteractiveObject drückt.
double-click	myButton.doubleClickEnabled = true; myButton.addEventListener(MouseEvent.DOUBLE_CLICK, dbclickHandler);	Wird ausgelöst, wenn der Benutzer einen Doppelklick über dem InteractiveObject ausführt. Zuvor muss die Eigenschaft doubleClickEnabled des Objekts auf true gesetzt werden.

▲ **Tabelle 11.3**
Ereignisse der InteractiveObject-Klasse

Ereignis	Beispiel	Beschreibung
focusIn	<pre> var meinTextfeld:TextField = new TextField(); meinTextfeld.text = "Hallo Welt"; this.addChild(meinTextfeld); meinTextfeld.addEventListener(FocusEvent.FOCUS_IN, focusInHandler); function focusInHandler(event:FocusEvent):void { trace("Fokus erhalten"); } </pre>	Wird ausgelöst, nachdem ein Anzeigebjekt den Fokus erhalten hat.
focusOut	<pre> ... meinTextfeld.addEventListener(FocusEvent. FOCUS_OUT, focusOutHandler); function focusOutHandler(event:FocusEvent):void { trace("Fokus verloren"); } </pre>	Wird ausgelöst, nachdem ein Anzeigebjekt den Fokus verloren hat.
keyDown	<pre> meinTextfeld.addEventListener(KeyboardEvent. KEY_DOWN, keyDownHandler); function keyDownHandler(event:KeyboardEvent): void { trace("Taste gedrückt"); } </pre>	Wird ausgelöst, wenn eine Taste der Tastatur gedrückt wird und das Anzeigebjekt, z. B. ein Textfeld, den Fokus besitzt.
keyUp	<pre> meinTextfeld.addEventListener(KeyboardEvent. KEY_UP, keyUpHandler); function keyUpHandler(event:KeyboardEvent):void { trace("Taste losgelassen"); } </pre>	Wird ausgelöst, wenn eine Taste der Tastatur losgelassen wird und das Anzeigebjekt, z. B. ein Textfeld, den Fokus besitzt.
mouseDown	<pre> myButton.addEventListener(MouseEvent.MOUSE_DOWN, mouseDownHandler); function mouseDownHandler (event:MouseEvent):void { trace("Maustaste gedrückt"); } </pre>	Wird ausgelöst, wenn die Taste des Zeigegeräts über einem Anzeigebjekt gedrückt wird.
mouseUp	<pre> myButton.addEventListener(MouseEvent.MOUSE_UP, mouseUpHandler); function mouseUpHandler(event:MouseEvent):void { trace("Maustaste losgelassen"); } </pre>	Wird ausgelöst, wenn die Taste des Zeigegeräts über einem Anzeigebjekt losgelassen wird.
mouseOut	<pre> myButton.addEventListener(MouseEvent.MOUSE_OUT, mouseOutHandler); function mouseOutHandler(event:MouseEvent):void { trace("Maustaste wegbewegt"); } </pre>	Wird ausgelöst, wenn das Zeigegerät aus einer InteractiveObject-Instanz wegbewegt wird.
mouseOver	<pre> myButton.addEventListener(MouseEvent.MOUSE_OVER, mouseOverHandler); function mouseOverHandler(event:MouseEvent):void { trace("Maus über mir"); } </pre>	Wird ausgelöst, wenn das Zeigegerät über eine InteractiveObject-Instanz bewegt wird.

▲ **Tabelle 11.3**
Ereignisse der InteractiveObject-Klasse (Fortsetzung)

Ereignis	Beispiel	Beschreibung
rollOver	<pre>myButton.addEventListener(MouseEvent.ROLL_OVER, rollOverHandler); function rollOverHandler(event:MouseEvent):void { trace("rollOver"); }</pre>	Wird ausgelöst, wenn das Zeige- gerät über eine Instanz der InteractiveObject-Klasse bewegt wird. Im Gegensatz zu mouseOver ist das Ereignisziel die übergeordnete Instanz.
rollOut	<pre>myButton.addEventListener(MouseEvent.ROLL_OUT, rollOutHandler); function rollOutHandler(event:MouseEvent):void { trace("rollOut"); }</pre>	Wird ausgelöst, wenn das Zeige- gerät aus einer Instanz der InteractiveObject-Klasse bewegt wird. Im Gegensatz zu mouseOut ist das Ereignisziel die übergeordnete Instanz.
enterFrame	<pre>mc.addEventListener(Event.ENTER_FRAME, enterFrameHandler); function enterFrameHandler(event:Event):void { trace("enterFrame"); }</pre>	Wird ausgelöst, wenn der Abspielkopf ein neues Bild erreicht.
mouseMove	<pre>mc.addEventListener(MouseEvent.MOUSE_MOVE, mouseMoveHandler); function mouseMoveHandler(event:MouseEvent):void { trace("Maus wurde bewegt"); }</pre>	Wird ausgelöst, wenn die Maus sich über dem Anzeigebereich befindet und bewegt wird.

▲ **Tabelle 11.3**
Ereignisse der InteractiveObject-Klasse (Fortsetzung)

11.7 Unterschiede ActionScript 1/2 und 3

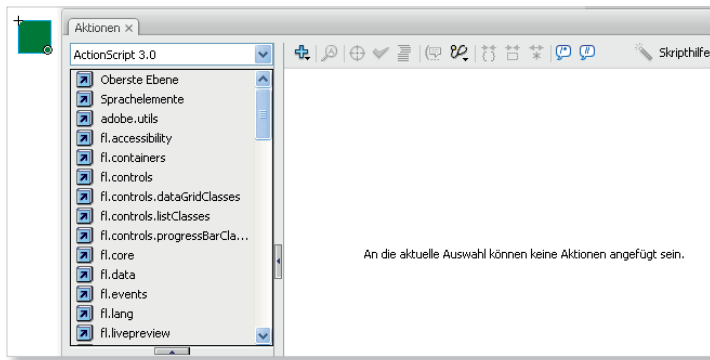
Textfelder in ActionScript 3
Dynamische und Eingabetextfelder besitzen in ActionScript-3-basierten Flash-Filmen kein Variablenfeld mehr, über das auf den Text des Textfeldes zugegriffen werden kann. Um auf den Text eines dynamischen oder eines Eingabetextfeldes zuzugreifen zu können, müssen Sie die text-Eigenschaft nutzen:

```
var myText:TextField = new TextField();
myText.text="Hallo Welt";
this.addChild(myText);
```

Da sich ActionScript 1/2 und ActionScript 3 sehr deutlich voneinander unterscheiden, fällt vielen Flash-Nutzern der Umstieg auf ActionScript 3 vergleichsweise schwer. In diesem Abschnitt möchte ich Sie noch auf einige Unterschiede hinweisen, die Ihnen den Einstieg bzw. den Umstieg auf ActionScript 3 erleichtern sollen.

11.7.1 ActionScript-Code

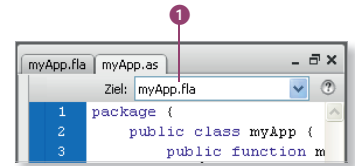
Grundsätzlich können Sie ActionScript-3-Code entweder nur in Bildern einer Zeitleiste definieren oder in einer separaten ActionScript-Datei. In ActionScript 3 können Sie Symbolinstanzen, wie Movieclips, keinen Code mehr direkt zuweisen.



▲ **Abbildung 11.19**

Einem Movieclip können Sie in ActionScript 3 keinen Code zuweisen.

ActionScript-Datei mit Flash-Film verknüpfen | Wenn Sie eine ActionScript-Datei erzeugen, können Sie die Datei im ActionScript-Editor mit einem geöffneten Flash-Film verknüpfen ¹. So können Sie über `[Strg]/[⌘] + [↩]` den verknüpften Flash-Film direkt aus dem ActionScript-Editor testen und müssen nicht vorher zum Flash-Film wechseln.



▲ **Abbildung 11.20**

ActionScript-Datei mit Flash-Film verknüpfen

11.7.2 Variablen und Datentypen

Variablen müssen in ActionScript 3 über das Schlüsselwort `var` definiert werden.

```
var myString:String = "Stringwert";
```

Variablen ohne bestimmten Geltungsbereich oder mit einem globalen Geltungsbereich sind nicht mehr zulässig. Folgende Anweisungen sind in ActionScript 3 ungültig:

```
myString = "Stringwert"; und _global.myString =  
"Stringwert";
```

Einmalige Variablendefinition | Variablen in einem bestimmten Geltungsbereich, z.B. einer Zeitleiste, können nur einmal über das Schlüsselwort `var` definiert werden. Eine mehrmalige Zuweisung ist unzulässig. Folgende Anweisungen wären demnach in verschiedenen Bildern einer Zeitleiste ungültig.

Code auf Bild 1:

```
var myString:String = "Wert 0";
```

Code auf Bild 10:

```
var myString:String = "Neuer Wert";
```

Das hat den Nebeneffekt, dass der Datentyp einer Variablen nicht geändert werden kann. In ActionScript 3 können Sie einer Variablen jedoch einen undefinierten Datentyp zuweisen. Dazu folgendes zulässige Beispiel:

```
var myVar:* = "String";  
myVar = 0;
```

Konstanten | Über das Schlüsselwort `const` können Sie in ActionScript 3 eine Konstante definieren. Eine Konstante ist ein Wert, der sich zur Laufzeit nicht ändern kann. Eine Konstante ist damit das Gegenstück zur Variable, deren Wert sich zur Laufzeit ändern kann.

```
const startValue:Number = 0;
```

Datentyp eines MovieClip-Objekts

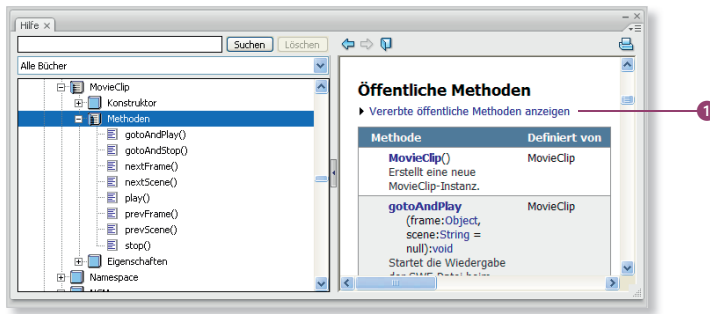
Da die `MovieClip`-Klasse von der `DisplayObject`-Klasse erbt, kann der Datentyp jedes `MovieClip`-Objekts auch als `DisplayObject` gekennzeichnet werden. Die folgende Definition ist demnach gültig: `var mc:DisplayObject = this.getChildAt(0);`

Neue Datentypen | In ActionScript 3 gibt es zwei neue numerische Datentypen `int` und `uint`, die vorzugsweise für Ganzzahlenwerte eingesetzt werden und deutlich schneller verarbeitet werden können als Werte vom Datentyp `Number`.

- ▶ `int`: Eine Variable vom Datentyp `int` kann eine 32-Bit-Ganzzahl mit Vorzeichen (Wertebereich von -2.147.483.648 bis 2.147.483.647) besitzen. Der Datentyp eignet sich z.B. für Zählerwerte und andere Werte, bei denen keine Gleitkommazahl benötigt wird.
- ▶ `uint`: Eine Variable vom Datentyp `uint` kann eine 32-Bit-Ganzzahl ohne Vorzeichen (Wertebereich von 0 bis 4.294.967.295) besitzen. Der Datentyp eignet sich besonders gut für Pixelfarbwerte aus dem ARGB- oder RGBA-Farbraum.

Klassenhierarchie | Alle Elemente in ActionScript 3, wie z.B. Movieclips, Textfelder etc., basieren auf einer Klassenhierarchie. Wenn Sie mit ActionScript 3 arbeiten möchten, ist es sehr wichtig, diese Hierarchie nach und nach verstehen zu lernen.

Wenn Sie sich z.B. die Methoden der `MovieClip`-Klasse ansehen, wird Ihnen vielleicht auffallen, dass die Klasse vergleichsweise wenig Methoden bereitstellt. In der Hilfe fällt erst auf den zweiten Blick die Option **VERERBTE ÖFFENTLICHE METHODEN ANZEIGEN**  auf, die, per Mausklick aktiviert, alle verfügbaren Methoden anzeigt. Viele Klassen erben Methoden, Eigenschaften und Ereignisse von übergeordneten Klassen.



◀ **Abbildung 11.21**
Vererbte öffentliche Methoden anzeigen

Diese Klassenhierarchie zieht sich durch ActionScript 3 und basiert auf der objektorientierten Programmierung. Übrigens können Sie die Beziehungen einer Klasse im oberen Bereich der Hilfe nachvollziehen **2**.



ActionScript-3-Klassen- diagramm

Die komplexe Klassenhierarchie von ActionScript 3 lässt sich auf einen Blick mithilfe eines Klassendiagramms nachvollziehen. Sie finden ein solches Diagramm z. B. unter http://flex.org/download/AS3API_01.pdf.

◀ **Abbildung 11.22**
Movieclip-Vererbung