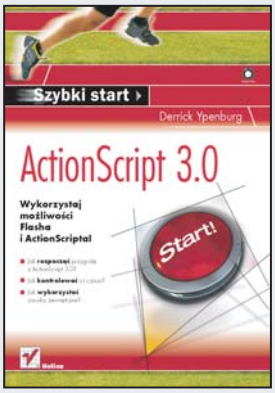
ActionScript 3.0. Szybki start



Autor: Derrick Ypenburg
Tłumaczenie: Łukasz Schmidt
ISBN: 978-83-246-2164-4
Tytuł oryginału: ActionScript 3.0: Visual QuickStart Guide
Format: 170×230, stron: 336

Wykorzystaj możliwości Flasha i ActionScript!

- Jak rozpocząć przygodę z ActionScript 3.0?
- Jak kontrolować oś czasu?
- Jak wykorzystać zasoby zewnętrzne?

ActionScript jest obiektowym językiem programowania, wykorzystywanym w Adobe Flash. Obecnie stosowana składnia tego języka pojawiła się po raz pierwszy we Flashu 5. ActionScript pozwala na sterowanie aplikacją oraz animacją, a jego możliwości mogą zaskoczyć niejednego programistę. To dzięki niemu stworzysz jeszcze bardziej interaktywne i atrakcyjne strony WWW!

Książka ta pozwoli Ci błyskawicznie opanować podstawy pracy z ActionScript, a po jej przeczytaniu stworzenie aplikacji na platformie Flash nie powinno stanowić dla Ciebie żadnego problemu. Dzięki niej poznasz składnię ActionScript 3.0, sposób wykorzystania zmiennych, właściwości oraz funkcji i metod. Nauczysz się pracować z obiektami wyświetlania oraz reagować na zdarzenia. Ponadto dowiesz się, jak kontrolować oś czasu, a także używać pól tekstowych oraz klas Math i Date. „ActionScript 3.0. Szybki start" jest obowiązkową pozycją dla wszystkich osób chcących rozpocząć przygodę z ActionScript 3.0!

- Składnia ActionScript 3.0
- Zmienne, właściwości, funkcje i metody
- Praca z klasami i obiektami
- Reagowanie na zdarzenia
- Kontrolowanie osi czasu
- Operacje na łańcuchach znaków
- Używanie pól tekstowych
- Zastosowanie klas Math i Date
- Instrukcje warunkowe oraz iteracje i powtórzenia
- Komunikacja zewnętrzna z wykorzystaniem protokołu HTTP
- Wykorzystanie zasobów zewnętrznych
- Dynamiczne sterowanie animacją
- Praca z wideo i dźwiękiem

Opanuj możliwości ActionScript szybko i przyjemnie!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Komunikacja i zdarzenia

7

Komunikacja w ActionScript jest w pełni oparta na zdarzeniach (*events*). Model zdarzeń w ActionScript 3 pozwala na całkowitą enkapsulację obiektów klas. **Enkapsulacja** oznacza, że obiekty klas „zajmują się same sobą" i nie dbają o to, jak funkcjonują inne obiekty. Po prostu robią, co do nich należy, i nie przejmują się niczym innym. Kiedy obiekty chcą porozumieć się ze światem zewnętrznym, wyzwalają zdarzenia. Do obiektów przypisywane są funkcje nasłuchujące, które czekają na wyzwolenie zdarzenia i podejmują odpowiednie działania.

Zdarzenia mogą być wyzwalane na wiele sposobów, zależnie od klasy. Jednak proces obsługi zdarzeń jest w całym ActionScript API standardowy. W tym rozdziale zapoznasz się z modelem zdarzeń ActionScript 3 poprzez zdefiniowanie jego faz oraz pracę ze zdarzeniami klas `Event`, `MouseEvent` i `KeyboardEvent`. Nauczysz się także korzystać z informacji przekazywanych przez wyzwolony obiekt zdarzenia, aby uprościć programowanie w ActionScript.

Kiedy skończysz czytać ten rozdział, będziesz swobodnie posługiwał się modelem i klasami zdarzeń.

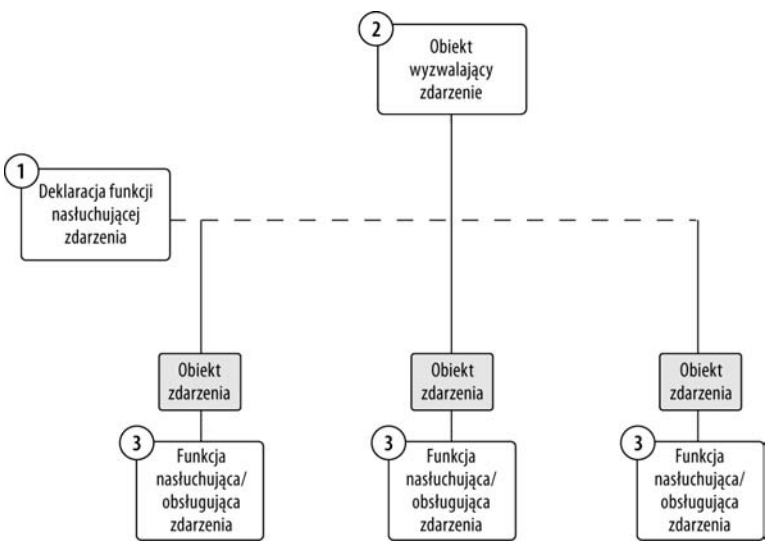
Model zdarzeń w ActionScript 3

Model zdarzeń przyjęty w ActionScript 3 sprawia, że komunikacja pomiędzy obiektami jest prosta i spójna. W poprzednich wersjach ActionScript modele zdarzeń nie były tak przejrzyste; istniało ich kilka i należało użyć odpowiedniego dla wykorzystywanej klasy. W ActionScript 3 model zdarzeń jest tylko jeden.

Szczegóły procesu zdarzeniowego

Istnieją trzy etapy procesu zdarzeniowego:

- **Deklaracja funkcji nasłuchującej** — deklaracja przypisuje funkcję nasłuchującą zajścia zdarzenia.
- **Wyzwalanie zdarzenia** — kiedy wewnątrz klasy zachodzi zdarzenie, zostaje ono wyzwolone przez obiekt. Wraz ze zdarzeniem przekazywane są obiekty zdarzenia. Obiekty zdarzenia zawierają informacje o zdarzeniu (rysunek 7.1).



Rysunek 7.1. *Trzy etapy procesu obsługi zdarzenia. Do jednego obiektu można przypisać wiele funkcji nasłuchujących, ponieważ obiekt może wyzwalać różne rodzaje zdarzeń*

Tabela 7.1. *Właściwości klasy Event*

Właściwość	Opis
`bubbles`	Wartość typu `Boolean` (`true`/`false`) wskazująca, czy zdarzenie rozchodzi się podobnie do bąbelka
`cancelable`	Wartość typu `Boolean`, która określa, czy zachowanie przypisane do zdarzenia może być anulowane
`currentTarget`	Obiekt i funkcja nasłuchująca przetwarzająca obiekt zdarzenia w danej chwili
`phase`	Aktualna faza przepływu zdarzenia
`target`	Obiekt będący celem zdarzenia
`type`	Typ wyzwalanego zdarzenia

Wskazówka

- Możesz programować zdarzenia własne dla klas własnych, jednak jest to zagadnienie poza zakresem tej książki. Więcej na ten temat dowiesz się, wyszukując w Pomocy Flash hasło `EventDispatcher class`.

- **Funkcja nasłuchująca lub obsługująca** — funkcja, która zawiera listę instrukcji (kod) wywoływanych w odpowiedzi na „usłyszenie" zdarzenia.

Funkcja **nasłuchująca zdarzenia** i **obsługująca zdarzenie** są jednym i tym samym — funkcją, która nasłuchuje zdarzenia, a także je obsługuje. Od tej pory będę nazywał ją albo funkcją nasłuchującą, albo obsługującą zdarzenia, powinieneś więc pamiętać, że jest to ta sama funkcja.

Klasy wyzwalające zdarzenia

Wszystkie klasy, które wyzwalają zdarzenia, używają metody `dispachEvent()` dziedziczonej po klasie `EventDispacher` lub stosują interfejs `IEventDispacher`. Metoda `dispachEvent()` jest wywoływana, kiedy wewnątrz klasy dzieje się coś, co klasa chce ogłosić pozostałej części aplikacji. Informacje mające związek ze zdarzeniem oraz inne parametry, które należy rozesłać, przekazywane są (do funkcji nasłuchującej) wraz z obiektem zdarzenia. Wyzwolenie zdarzenia rozpoczyna proces zdarzeniowy.

Klasa `Event` jest klasą bazową dla obiektów zdarzeń. Kiedy zdarzenie jest wyzwalane metodą klasy `EventDispacher` (na przykład w wyniku użycia `addEventListener`), obiekt zdarzenia zostaje przekazany do metody (funkcji) nasłuchującej jako parametr. Obiekt ten zawiera dane związane ze zdarzeniem, które zaszło. Klasa `Event` zawiera standardowe właściwości, metody i stałe, które są używane przez większość klas. Tabela 7.1 wymienia właściwości klasy `Event`, które zostają przekazane wraz z obiektem zdarzenia. Do tych właściwości można uzyskać dostęp w funkcji obsługującej zdarzenie.

W tej książce będziemy zajmować się głównie najbardziej podstawowymi właściwościami obiektów zdarzeń — są to właściwości `type`, `target` oraz `currentTarget`.

Rejestrowanie funkcji nasłuchującej

Funkcje nasłuchujące są przypisywane (rejestrowane) do obiektów w celu nasłuchiwania zajścia wyzwalanych przez te obiekty zdarzeń. Obiekt będzie wyzwalał zachodzące zdarzenia niezależnie od tego, czy została zarejestrowana funkcja nasłuchująca, która czeka na „usłyszenie" lub „odebranie" zdarzenia.

Funkcje nasłuchujące muszą zostać „podłączone" do obiektu, aby móc usłyszeć jego zdarzenia.

Funkcje nasłuchujące są przypisywane do obiektywów, kiedy określone zdarzenie, wyzwalane przez obiekt, jest ważne dla funkcjonalności aplikacji — na przykład, kiedy użytkownik klika przycisk myszy (wyzwalane jest zdarzenie `mouseDown`) lub kiedy kończy się ładowanie obrazu (w takim przypadku jest do zdarzenie `complete`).

Każda funkcja nasłuchująca przypisana (zarejestrowana) do obiektu nasłuchuje *zdarzenia określonego typu*. Typ wyzwalanego zdarzenia reprezentowany jest przez ciąg znaków, który przechowuje **stała klasy**. Stałe to właściwości tylko do odczytu, które się nie zmieniają i są zwyczajowo zapisywane wielkimi literami. Jak powiedzieliśmy już wcześniej, referencje do właściwości statycznych są tworzone bezpośrednio poprzez nazwę klasy. Klasa zdarzenia powiązana z obiektem zdarzenia także przechowuje stałe typów zdarzeń. Przykładowo, typem zdarzenia `MouseEvent` może być `mouseDown`, czyli `MouseEvent.` ↪`MOUSE_DOWN`. Powodem użycia stałej zdarzenia (tutaj `MOUSE_DOWN`) zamiast ciągu znaków (`"mouseDown"`) dla określenia typu zdarzenia jest umożliwienie automatycznego uzupełniania kodu w edytorze ActionScript oraz wychwycenia przez kompilator wszelkich literówek, które mogą wystąpić przy wpisywaniu typu zdarzenia.

Tabela 7.2 pokazuje niektóre najpopularniejsze typy zdarzeń klasy Event i odpowiadające im zmienne

Tabela 7.2. *Typy zdarzeń klasy Event*

Stała	Typ zdarzenia
`ADDED_TO_STAGE`	Obiekt zdarzenia `"addedToStage"`
`CHANGE`	Obiekt zdarzenia `"change"`
`COMPLETE`	Obiekt zdarzenia `"complete"`
`ENTER_FRAME`	Obiekt zdarzenia `"enterFrame"`
`FULL_SCREEN`	Obiekt zdarzenia `"fullScreen"`
`INIT`	Obiekt zdarzenia `"init"`
`REMOVED`	Obiekt zdarzenia `"removed"`
`REMOVED_FROM_STAGE`	Obiekt zdarzenia `"removedFromStage"`
`RESIZE`	Obiekt zdarzenia `"resize"`
`SOUND_COMPLETE`	Obiekt zdarzenia `"soundComplete"`
`UNLOAD`	Obiekt zdarzania `"unload"`

Składnia służąca do dodawania funkcji nasłuchującej do obiektu zaczyna się od nazwy obiektu, po której następuje metoda `addEventListener()`:

```
objectName.addEventListener();
```

Do metody `addEventListener()` należy przekazać dwa obowiązkowe parametry:

- Typ zdarzenia, którego będzie nasłuchiwać.
- Funkcję, która będzie funkcją obsługującą zdarzenie.

Oto przykład:

```
objectName.addEventListener(EventClass.EVENT_TYPE,
↪EventHandlerFunction);
```

W taki sposób należałoby zapisać nasłuchiwanie zdarzenia `MouseEvent` powiązanego z przyciskiem:

```
buttonName.addEventListener(MouseEvent.MOUSE_UP,
↪mouseUp);
```

Funkcje nasłuchujące

Funkcja nasłuchująca zdarzenia (nazywana także obsługującą zdarzenie) to funkcja lub metoda, która została przypisana do obiektu, aby „nasłuchiwać" zajścia zdarzenia określonego typu. Jest deklarowana tak, jak każda inna funkcja, jednak jako parametr takiej funkcji musisz wskazać obiekt zdarzenia:

```
function eventHandler(evt:Event):void {
}
```

Do właściwości obiektu zdarzenia można uzyskać dostęp, odwołując się do parametru przekazywanego do funkcji nasłuchującej.

Wskazówka

- Użycie ciągu znaków (np. `"mouseDown"`) zamiast stałej zdarzenia (`MOUSE_DOWN`) jest sposobem poprawnym, ale niezalecanym.

Praca z funkcjami nasłuchującymi

Funkcje nasłuchujące są kanałami obiegu wszystkich informacji w ActionScript. Zdarzenie jest wyzwalane, funkcja nasłuchująca „słyszy" je i obsługuje.

Kiedy mówimy „obsługuje zdarzenie", mamy na myśli, że funkcja nasłuchująca otrzymuje obiekt zdarzenia jako wartość parametru i przechodzi do wykonania zawartego w sobie kodu.

Klasa Event

Klasa `flash.events.Event` obsługuje najpopularniejsze zdarzenia ActionScript API. Na przykład, zdarzenie `addedToStage` jest wyzwalane przez każdy obiekt wyświetlania, który zostanie dodany do stołu montażowego w czasie wykonywania.

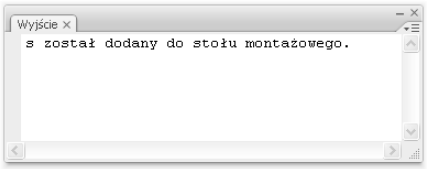
Jak nasłuchiwać zdarzenia klasy Event?

1. Umieść poniższy kod ActionScript w pierwszej klatce kluczowej nowego pliku. Kod tworzy nową instancję obiektu klasy `Sprite` i przypisuje do niej funkcję nasłuchującą zdarzenia `addedToStage` klasy `Event`:

```
var s:Sprite = new Sprite();
s.addEventListener(Event.ADDED_TO_STAGE,
↪spriteAddedHandler);
```

2. Dodaj kolejny fragment kodu, który tworzy funkcję nasłuchującą zdarzenia `ADDED_TO_STAGE`:

```
function spriteAddedHandler(evt:Event):void {
  trace("s został dodany do stołu montażowego.");
}
```



Rysunek 7.2. *Funkcja nasłuchująca spriteAddedHandler() jest wykonywana po wyzwoleniu zdarzenia Event.ADDED_TO_STAGE przez obiekt Sprite, co ma miejsce po dodaniu tego obiektu do stołu montażowego*

Listing 7.1. *Rejestrowanie funkcji nasłuchującej zdarzenia REMOVED_FROM_STAGE*

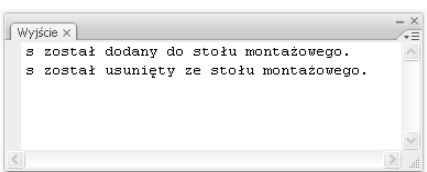
Listing

```
var s:Sprite = new Sprite();
s.addEventListener(Event.ADDED_TO_STAGE,
↪spriteAddedHandler);
s.addEventListener(Event.REMOVED_FROM_STAGE,
↪spriteRemovedHandler);
```

Listing 7.2. *Dodawanie funkcji nasłuchującej spriteRemoveHandler() i usuwanie sprajta z listy wyświetlania*

Listing

```
function spriteAddedHandler(evt:Event):void {
   trace("s został dodany do stołu
   ↪montażowego.");
}
function spriteRemovedHandler(evt:Event):void {
   trace("s został usunięty ze stołu
   ↪montażowego.");
}
this.addChild(s);
this.removeChild(s);
```



Rysunek 7.3. *Funkcja nasłuchująca spriteRemoveHandler() jest wywoływana, kiedy obiekt Sprite jest usuwany ze stołu montażowego, co wyzwala zdarzenie REMOVED_FROM_STAGE*

3. Wpisz następujący kod, który doda obiekt `Sprite` jako obiekt-dziecko do stołu montażowego:

```
addChild(s);
```

4. Obejrzyj działanie pliku. Wiadomość z instrukcji `trace()` powinna zostać wyświetlona w panelu *Wyjście* (rysunek 7.2).

Do tego samego obiektu można dodać kilka funkcji nasłuchujących (jeden obiekt będzie wyzwalał zdarzenia różnych typów do różnych funkcji nasłuchujących). Kilka obiektów można połączyć z jedną funkcją nasłuchującą (standardowa funkcja nasłuchująca będzie identyfikować typ wyzwolonego zdarzenia oraz obiekt, z którego zdarzenie to pochodzi).

Jak nasłuchiwać kilku typów zdarzeń klasy Event?

1. Dodaj kod wyróżniony w listingu 7.1, który zarejestruje funkcję nasłuchującą zdarzenia typu `REMOVED_FROM_STAGE` klasy `Event`.
2. Dodaj kod wyróżniony w listingu 7.2, który tworzy funkcję obsługującą `spriteRemoveHandler` i usuwa obiekt `Sprite` ze stołu montażowego.
3. Obejrzyj działanie pliku. W panelu *Wyjście* powinna zostać wyświetlona widomość funkcji `spriteRemovedHandler()` (rysunek 7.3).

Powyższe ćwiczenie pokazało, że do jednego obiektu można przypisać kilka funkcji nasłuchujących — po jednej dla każdego typu wyzwalanego zdarzenia.

Praca z obiektami zdarzeń

Referencje do obiektów klasy `Event` tworzone są poprzez wywoływanie obiektu `Event` przekazywanego jako parametr do funkcji nasłuchującej. Właściwości przekazywane z obiektem tego rodzaju znajdziesz w tabeli 7.1.

Jak uzyskać dostęp do właściwości obiektu zdarzenia?

1. Wróć do pliku z poprzedniego ćwiczenia.
2. Zmodyfikuj kod ActionScript tak, aby wyświetlał typ zdarzenia oraz cel zdarzenia (listing 7.3).
3. Obejrzyj działanie pliku. Właściwości obiektu `Event` powinny zostać wyświetlone w panelu *Wyjście* (rysunek 7.4).

Wskazówka

- Dobrym sposobem wykorzystania właściwości obiektu zdarzenia jest sprawdzenie kilku typów zdarzeń w pojedynczej funkcji obsługującej zamiast tworzenia osobnych funkcji dla każdego z typów nasłuchiwanych zdarzeń. Przyjrzymy się temu bliżej w kolejnym podrozdziale.

Klasa MouseEvent

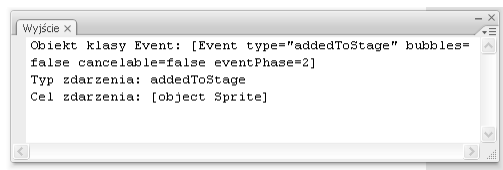
Niektóre obiekty będą wymagały wyzwolenia dodatkowych typów zdarzeń, których klasa `Event` nie potrafi już obsłużyć. Będziemy musieli użyć „wzbogaconego" obiektu zdarzenia. Takie wzbogacone klasy rozszerzają klasę `Event`, aby móc obsłużyć dodatkowe typy zdarzeń i właściwości. Przykładem jest klasa `flash.events.MouseEvent`.

Klasa `MouseEvent` jest jedną z najczęściej używanych klas zdarzeń w ActionScript. Bez możliwości obsługiwania zdarzeń myszy interakcje z użytkownikiem byłyby bardzo ograniczone. Tabela 7.3 wymienia wszystkie typy zdarzeń klasy `MouseEvent` dostępne w ActionScript API.

Listing 7.3. *Wywoływanie właściwości obiektu klasy Event i wyświetlanie ich w panelu Wyjście*

Listing

```
function spriteAddedHandler(evt:Event):void {
   //trace("s został dodany do stołu
   ↪montażowego.");
   trace("Obiekt klasy Event: " + evt);
   trace("Typ zdarzenia: " + evt.type);
   trace("Cel zdarzenia: " + evt.target);
}
function
spriteRemovedHandler(evt:Event):void {
   //trace("s został usunięty ze stołu
   ↪montażowego.");
}
this.addChild(s);
this.removeChild(s);
```



Rysunek 7.4. *Wyświetlanie właściwości obiektu klasy Event*

Tabela 7.3. *Typy zdarzeń klasy MouseEvent*

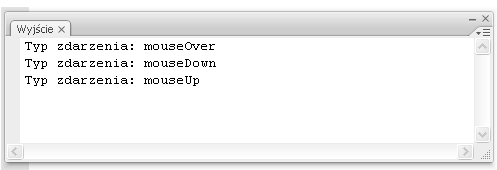
Stała	Typ zdarzenia
`CLICK`	Obiekt zdarzenia `"click"`
`DOUBLE_CLICK`	Obiekt zdarzenia `"doubleClick"`
`MOUSE_DOWN`	Obiekt zdarzenia `"mouseDown"`
`MOUSE_MOVE`	Obiekt zdarzenia `"mouseMove"`
`MOUSE_OUT`	Obiekt zdarzenia `"mouseOut"`
`MOUSE_OVER`	Obiekt zdarzenia `"mouseOver"`
`MOUSE_UP`	Obiekt zdarzenia `"mouseUp"`
`MOUSE_WHEEL`	Obiekt zdarzenia `"mouseWheel"`
`ROLL_OUT`	Obiekt zdarzenia `"rollOut"`
`ROLL_OVER`	Obiekt zdarzenia `"rollOver"`

Listing 7.4. *Dodawanie funkcji nasłuchującej zdarzenia typu MOUSE_DOWN*

```
Listing
btnButton1.addEventListener(MouseEvent.MOUSE_
↪DOWN, mouseEventHandler);
function
mouseEventHandler(evt:MouseEvent):void {
   trace("Typ zdarzenia: " + evt.type);
}
```

Listing 7.5. *Obiekt wyzwala trzy zdarzenia klasy MouseEvent do tej samej funkcji nasłuchującej/obsługującej*

```
Listing
btnButton1.addEventListener(MouseEvent.MOUSE_
↪DOWN, mouseEventHandler);
btnButton1.addEventListener(MouseEvent.MOUSE_
↪UP, mouseEventHandler);
btnButton1.addEventListener(MouseEvent.MOUSE_
↪OVER, mouseEventHandler);
function mouseEventHandler(evt:MouseEvent):
↪void {
   trace("Typ zdarzenia: " + evt.type);
}
```



Rysunek 7.5. *Pojedyncza funkcja obsługująca używa właściwości type obiektu klasy MouseEvent do wyświetlania informacji o różnych zdarzeniach myszy w panelu Wyjście*

Jak nasłuchiwać zdarzeń myszy?

1. Narysuj na stole montażowym prostokątny kształt.
2. Zaznacz kształt i skonwertuj go na symbol przycisku.
3. Nadaj instancji przycisku na stole montażowym nazwę `btnButton1`.
4. Utwórz nową warstwę `Actions`.
5. Wpisz kod z listingu 7.4, dodający do przycisku `btnButton1` funkcję nasłuchującą, która będzie nasłuchiwać zdarzenia typu `MOUSE_DOWN` klasy `MouseEvent`.
6. Obejrzyj działanie pliku. Wiadomość z instrukcji `trace()` powinna zostać wyświetlona w panelu `Wyjście` po kliknięciu przycisku.
7. Zmodyfikuj kod ActionScript tak, aby do tej samej funkcji nasłuchującej przypisać zdarzenia `MOUSE_OVER` oraz `MOUSE_UP` klasy `MouseEvent` (listing 7.5).
8. Obejrzyj działanie pliku. W panelu *Wyjście* powinieneś zobaczyć informacje o trzech typach zdarzeń (rysunek 7.5).

Wykrywanie zdarzeń kilku typów za pomocą tej samej funkcji obsługującej

Zwyczajem każdego zaawansowanego programisty jest pisanie tak małej ilości kodu, jak to możliwe. Dzięki temu wzrasta wydajność aplikacji, a zarządzanie i aktualizowanie kodu staje się łatwiejsze. Dobrą okazją do zoptymalizowania ilości kodu jest zmniejszenie liczby funkcji obsługujących zdarzenia. Tworząc referencje do właściwości `type` obiektu zdarzenia, można określić typ zdarzenia i na tej podstawie wykonać odpowiednie działania w funkcji obsługującej.

Jak sprawdzić wystąpienie kilku typów zdarzeń w jednej funkcji nasłuchującej?

1. Wróć do pliku z poprzedniego ćwiczenia.
2. Tuż poniżej instrukcji `trace()` w funkcji `mouseEventHandler` obsługującej zdarzenia dodaj kod z listingu 7.6.

Instrukcje warunkowe zostaną omówione w rozdziale 12.

Klasa KeyboardEvent

Klasa `flash.events.KeyboardEvent` to kolejna klasa rozszerzająca klasę `Event`. Klasa `KeyboardEvent` zawiera dwa nowe zdarzenia, które dodają funkcjonalność klawiatury do aplikacji: `keyDown` i `keyUp`. Ich stałe to odpowiednio `KEY_DOWN` oraz `KEY_UP`.

Aby nasłuchiwać globalnie zdarzeń klawiatury na poziomie aplikacji, można przypisać funkcję nasłuchującą do stołu montażowego: `stage.addEventListener()`. Taka funkcja może nasłuchiwać zdarzeń `KEY_DOWN` i `KEY_UP` i wywoływać odpowiednie dla nich procedury.

Aby móc wykrywać i reagować na naciśnięcie klawisza, należy zadeklarować funkcję nasłuchującą zdarzenia klasy `KeyboardEvent`. Kombinacje klawiszy, na przykład z klawiszem *Ctrl*, *Alt* lub *Shift*, także mogą zostać wykryte. Te właściwości wymienia tabela 7.4.

Listing 7.6. *Użycie instrukcji warunkowej if w celu wyboru odpowiedniego typu zdarzenia klasy MouseEvent w pojedynczej funkcji obsługującej zdarzenie. Kiedy literalna nazwa typu zdarzenia będzie odpowiadać ciągowi znaków instrukcji if, ta instrukcja zostanie wykonana*

Listing

```
function mouseEventHandler(evt:MouseEvent):
↪void {
trace("Typ zdarzenia: " + evt);
if (evt.type == "MOUSE_DOWN") {
   trace("Zdarzenie mouseDown. Zrób coś.");
} else if (evt.type == "MOUSE_UP") {
   trace("Zdarzenie mouseUp. Zrób coś.");
} else if (evt.type == "MOUSE_OVER") {
   trace("Zdarzenie mouseOver. Zrób coś.");
}
}
```

Tabela 7.4. *Właściwości klasy KeyboardEvent*

Właściwość	Opis
`charCode`	Kod znaku dla naciśniętego lub zwolnionego klawisza
`keyCode`	Kod naciśniętego lub zwolnionego klawisza
`keyLocation`	Położenie naciśniętego lub zwolnionego klawisza na klawiaturze
`ctrlKey`	`True`, jeśli klawisz jest naciśnięty, `false`, jeśli nie
`altKey`	`True`, jeśli klawisz jest naciśnięty, `false`, jeśli nie
`shiftKey`	`True`, jeśli klawisz jest naciśnięty, `false`, jeśli nie

Listing 7.7. *Wyświetlanie kodu klawisza w panelu Wyjście*

```
Listing
stage.addEventListener(KeyboardEvent.KEY_DOWN,
↪downEventHandler);
function downEventHandler(evt:KeyboardEvent):
↪void {
   trace("Wartość keyCode: " + evt.keyCode);
}
```

Listing 7.8. *Nasłuchiwanie naciśnięcia klawisza W*

```
Listing
stage.addEventListener(KeyboardEvent.KEY_DOWN,
↪downEventHandler);
function
downEventHandler(evt:KeyboardEvent):void {
   if (evt.keyCode == 87) {
      trace("Wartość keyCode: " +
      ↪evt.keyCode);
   }
}
```

Wykrywanie rodzaju klawisza osiągane jest poprzez porównanie wartości `keyCode` aktualnie naciśniętego klawisza ze standardowymi wartościami służącymi do identyfikacji klawiszy w ActionScript.

Jak wykrywać zdarzenia klawiatury i wartości klawiszy?

1. Dodaj do stołu montażowego kod z listingu 7.7, aby nasłuchiwać zdarzenia `KEY_DOWN` i użyć instrukcji `trace()` do wyświetlania wartości właściwości `keyCode` w panelu *Wyjście*.
2. Obejrzyj działanie pliku. Naciśnij klawisz *W*; w panelu *Wyjście* powinna zostać wyświetlona wiadomość *Wartość keyCode: 87*.
3. Umieść instrukcję `trace()` funkcji `downEventHandler()` wewnątrz bloku instrukcji warunkowej `if`, która zostanie wykonana tylko wtedy, kiedy użytkownik naciśnie klawisz *W* (listing 7.8).
4. Obejrzyj działanie pliku. Instrukcja `trace()` zostanie wykonana tylko wtedy, kiedy wartość `keyCode` będzie odpowiadała liczbie 87 wskazanej w instrukcji `if`.

Pełną listę kodów klawiszy można znaleźć na stronie Adobe LiveDocs: *http://livedocs.adobe.com/flash/9.0/main/00001136.html*.

Wskazówka

- Oglądanie działania pliku bezpośrednio w środowisku Flash może spowodować, że naciśnięcia niektórych klawiszy nie będą rejestrowane przez odtwarzacz, ponieważ samo środowisko będzie traktować je jako własne skróty klawiszowe i będzie zakłócać pracę odtwarzacza.

Usuwanie powiązania z funkcjami nasłuchującymi

Tak samo ważne, jak dodawanie funkcji nasłuchujących do obiektów, jest ich usuwanie, kiedy chcemy usunąć obiekt ze stołu montażowego lub nie musimy już nasłuchiwać zdarzeń tego obiektu. Mimo że obiekt nie będzie już potrzebny, jego powiązania z funkcjami nasłuchującymi pozostaną i będą dalej pochłaniać zasoby komputera.

Aby usunąć przypisanie funkcji nasłuchującej, użyjemy składni zbliżonej do tej, która posłużyła do dodania tej funkcji, z wyjątkiem wywoływanej metody, którą będzie tym razem `removeEventListener()`. Pierwszym krokiem przy usuwaniu funkcji nasłuchującej jest wywołanie obiektu, do którego została przypisana ta funkcja. Kolejny krok to wywołanie metody `removeEventListener()`, a ostatnim jest wskazanie jako parametrów typu zdarzenia i funkcji nasłuchującej:

```
objectName.removeEventListener(EventClass.EVENT_TYPA,
↪eventHandler);
```

Jak usunąć powiązanie z funkcją nasłuchującą?

1. Dodaj kod z listingu 7.9, który deklaruje funkcję nasłuchującą zdarzenia `MOUSE_DOWN` klasy `MouseEvent` dla stołu montażowego.
2. Za instrukcją `trace()` dodaj instrukcję `removeEventListener()`, która usunie przypisanie funkcji nasłuchującej zdarzenia `MouseEvent.MOUSE_DOWN` ze stołu montażowego (listing 7.10).
3. Obejrzyj działanie pliku. Kliknij stół montażowy — zobaczysz komunikat *Kliknięto stół montażowy*. Ponownie kliknij stół montażowy. Tym razem nie zobaczysz komunikatu, ponieważ powiązanie z funkcją nasłuchującą zdarzenia `MOUSE_DOWN` zostało usunięte.

Listing 7.9. *Nasłuchiwanie zdarzenia MOUSE_DOWN dla stołu montażowego*

Listing

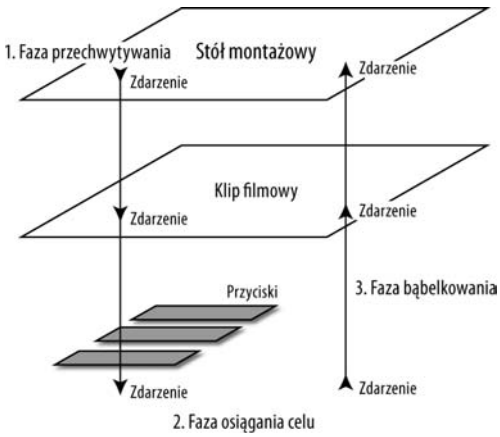
```
stage.addEventListener(MouseEvent.MOUSE_DOWN,
↪stageClickHandler);
function
stageClickHandler(evt:MouseEvent):void {
   trace("Kliknięto stół montażowy.");
}
```

Listing 7.10. *Usuwanie funkcji nasłuchującej zdarzenia MOUSE_DOWN*

Listing

```
stage.addEventListener(MouseEvent.MOUSE_DOWN,
↪stageClickHandler);
function stageClickHandler(evt:MouseEvent):
↪void {
   trace("Kliknięto stół montażowy.");

stage.removeEventListener(MouseEvent.MOUSE_
↪DOWN, stageClickHandler);
}
```



Rysunek 7.6. *Trzy fazy przepływu zdarzeń*

Przepływ zdarzeń

Przepływ zdarzeń to koncepcja nowa w ActionScript 3, mająca zastosowanie głównie w połączeniu z obiektami `MouseEvent` i `KeyboardEvent`. Kiedy wyzwalane jest zdarzenie, na przykład z przycisku, takie jak zdarzenie klasy `MouseEvent`, w istocie nie jest ono wyzwalane tylko przez sam przycisk, ale także przez obiekty wyświetlania-pojemniki, w których znajduje się ten przycisk (rysunek 7.6).

Ta zasada może okazać się przydatna, na przykład, kiedy dysponujesz obiektem wyświetlania-pojemnikiem klasy `MovieClip`, w którym znajduje się kilka przycisków. Zamiast przypisywać funkcje nasłuchujące do każdego przycisku w pojemniku `MovieClip`, możesz przypisać pojedynczą funkcję nasłuchującą do samego pojemnika. Następnie możesz sprawdzać, który z przycisków w pojemniku został kliknięty, poprzez odczytywanie właściwości `target` i `currentTarget` obiektu `MouseEvent`.

Przepływ zdarzeń dzieli się na trzy fazy:

- **Faza przechwytywania** — zdarzenie rozpoczyna przepływ na stole montażowym i przechodzi przez wszystkie obiekty wyświetlania, aż dotrze do obiektu, w którym ma powstawać (zostać wyzwolone).
- **Faza osiągania celu** — zdarzenie jest wyzwalane przez obiekt docelowy.
- **Faza bąbelkowania** — jeśli w chwili tworzenia obiektu zdarzenia właściwość `Bubbles` określona została jako prawdziwa, zdarzenie zaczyna poruszać się jak bąbelek ponownie w górę do kolejnych nadrzędnych obiektów-pojemników, aż do osiągnięcia stołu montażowego.

Różnica pomiędzy fazami przechwytywania i bąbelkowania polega na tym, że możesz pracować ze zdarzeniem, kiedy porusza się w dół w fazie przechwytywania lub (i) kiedy porusza się w górę po fazie osiągania celu. Jeśli będziesz chciał używać tylko fazy przechwytywania, ustaw wartość `true` dla dodatkowego parametru `useCapture` po wskazaniu obiektu zdarzenia w funkcji obsługującej:

```
function eventHandler (evt:EventObject,
↪useCapture=true);
```

Domyślnie wartością `useCapture` jest `false`, a zdarzenia są obsługiwane zarówno w fazie osiągania celu, jak i bąbelkowania.

Chcąc pracować ze zdarzeniami w różnych fazach przepływu, użyj właściwości `target` i `currentTarget` obiektu zdarzenia.

Właściwość `currentTarget` zawsze wskazuje na obiekt, do którego przypisano funkcję obsługującą, właściwość `target` natomiast na obiekt będący celem w bieżącej chwili przepływu zdarzenia.

Powiedzmy, że przycisk o nazwie `button1` znajduje się na liście wyświetlania klipu filmowego `buttonContainer`, do którego przypisano funkcję o nazwie `clickHandler` obsługującą zdarzenie `MOUSE_DOWN`. Jeśli wskaźnik myszy będzie umieszczony nad przyciskiem `button1` w chwili kliknięcia klipu `buttonContainer`, właściwość `currentTarget` obiektu klasy `MouseEvent` zwróci `buttonContainer`, a właściwość `target` zwróci `button1`.

Jak pracować z przepływem zdarzeń?

1. Narysuj na stole montażowym prostokąt, który będzie mógł posłużyć za przycisk.
2. Skonwertuj prostokąt na symbol przycisku o nazwie `Button`.
3. Przeciągnij na stół montażowy dodatkowe dwie instancje symbolu `Button`.
4. Nadaj instancjom symbolu `Button` nazwy `button1`, `button2` i `button3`.
5. Zaznacz wszystkie instancje symbolu `Button` na stole montażowym i skonwertuj na symbol klipu filmowego o nazwie `ButtonContainer`. Wszystkie trzy instancje przycisków znajdą się wewnątrz instancji `ButtonContainer`.
6. Nadaj instancji symbolu `ButtonContainer` nazwę `buttonContainer`.
7. Utwórz nową warstwę `Actions`.

Listing 7.11. *Wyświetlanie właściwości target i currentTarget obiektu klasy MouseEvent*

```
function buttonEventHandler(evt:MouseEvent):
↪void {
   trace("Właściwość target:: " +
   ↪DisplayObject(evt.target).name);
   trace("Właściwość currentTarget: " +
   ↪DisplayObject(evt.currentTarget).name);
}
```

```
Właściwość target:: button1
Właściwość currentTarget: buttonContainer
Właściwość target:: button2
Właściwość currentTarget: buttonContainer
Właściwość target:: button3
Właściwość currentTarget: buttonContainer
```

Rysunek 7.7. *Wartością właściwości currentTarget jest zawsze buttonContainer, ponieważ funkcję nasłuchującą przypisano do właśnie tego obiektu. Wartości właściwości natomiast zmieniają się dla każdego przycisku i są nimi obiekty docelowe pod kursorem myszy w chwili zajścia zdarzenia mouseDown, czyli kliknięcia.*

8. W warstwie `Actions` wpisz następujący kod, który doda funkcję nasłuchującą do klipu `buttonContainer` — będzie nią funkcja o nazwie `buttonEventHandler`:

```
buttonContainer.addEventListener(MouseEvent.M
OUSE_DOWN, buttonEventHandler);
```

9. Dodaj do skryptu kod z listingu 7.11, aby stworzyć funkcję nasłuchującą i wyświetlić w panelu *Wyjście* właściwości `target` i `currentTarget` obiektu `klasy MouseEvent`.

Zauważ, że wewnątrz instrukcji `trace()` właściwości `evt.target` i `currentTarget` są konwertowane na obiekty `DisplayObject` przed wywołaniem ich właściwości `name`. Zrobiliśmy to, ponieważ właściwości `target` i `currentTarget` to referencje będące łańcuchami znaków do nazw obiektów `DisplayObject`, a nie do samych obiektów. Konwersja zapobiega wywołaniu ich bezpośrednio jako obiektów `DisplayObject`. Metoda `DisplayObject()` nakazuje programowi traktować właściwości `target` i `currentTarget` jako obiekty wyświetlania, które można wywoływać i kontrolować z kodu ActionScript.

10. Obejrzyj działanie pliku. Kliknij kolejne przyciski i zobacz, jakie informacje wyświetla instrukcja `trace()` w panelu *Wyjście* (rysunek 7.7).

Wskazówka

- Rozróżnianie pomiędzy fazami bąbelkowania i przechwytywania jest użyteczne w przypadku złożonych interakcji pomiędzy komponentami, gdzie pewne zdarzenia muszą zostać przechwycone, zanim osiągną określone obiekty, lub kod musi być wykonany dokładnie w fazie bąbelkowania. Na tym etapie nauki nie musisz przejmować się obiegiem zdarzeń, chociaż wiedza na jego temat będzie przydatna przy wykrywaniu błędów w kodzie związanym z obsługą interaktywności myszy lub wszędzie, gdzie spodziewasz się, że błąd jest wynikiem problemu z przepływem zdarzeń.

Podklasy klasy Event

W tym rozdziale omówiliśmy klasy `Event`, `MouseEvent` oraz `KeyboardEvent`. W następnych rozdziałach będziemy pracowali z wieloma innymi podklasami klasy `Event`. Tabela 7.5 zawiera zestawienia popularnych podklas klasy `Event`. Jeśli będziesz chciał zapoznać się z pełną listą podklas i opisem każdej z nich, wyszukaj w Pomocy Flash wyrażenie `Event class`.

Tabela 7.5. *Podklasy klasy Event*

Klasa	Opis
`ColorPickerEvent`	Obiekt zdarzenia powiązany z komponentem `ColorPicker`
`FocusEvent`	Obiekt zdarzenia używany, kiedy fokus przenosi się z jednego obiektu na drugi
`KeyboarEvent`	Obiekt zdarzenia używany do rozpoznawania klawiszy klawiatury
`MetadataEvent`	Obiekt zdarzania używany w chwili otrzymania pakietu metadanych z FLV wideo
`MotionEvent`	Obiekt zdarzenia wyzwalany przez klasę `fl.motion.Animator`
`MouseEvent`	Obiekt zdarzenia wyzwalany w chwili zajścia zdarzenia myszy
`NetStatusEvent`	Obiekt zdarzenia wyzwalany przez klasy `NetConnection`, `Netstream` i `SharedObject` w chwili zmiany ich statusu
`ProgressEvent`	Obiekt zdarzenia wyzwalany przez obiekty `loader` w regularnych odstępach czasu po rozpoczęciu operacji wczytywania
`SoundEvent`	Obiekt zdarzenia wyzwalany przez obiekt `Sound` w chwili zmiany ustawień
`TextEvent`	Obiekt zdarzenia wyzwalany podczas interakcji z obiektem `textField` (polem tekstowym) lub zmiany jego zawartości
`TimeEvent`	Obiekt zdarzenia wyzwalany przez obiekt `Timer` w określonych odstępach czasu
`TweenEvent`	Obiekt zdarzenia wyzwalany przez klasę `fl.transitions.Tween`
`VideoEvent`	Obiekt zdarzenia wyzwalany w chwili zmiany statusu wideo