

# Advanced Multimedia Technologies

Events, Listeners & Handlers

# Outcomes

- Understand event notification system
- Register targets to listen and respond to events
- Identify and leverage built-in events

# Events

# Event

A notification

**Who does it notify?**

**Anyone that listens, known  
as an Event Listener**

# Event Listener

Registered target that listens for event notification.

```
mc.addEventListener(MouseEvent.CLICK, onClick);
```

**What does the listener do once it gets the event notification?**

# Event Handler

A function that executes when the event listener has received an event notification.

```
private function onClick(e:MouseEvent):void{  
    trace('I have clicked');  
}
```

So `MouseEvent.CLICK`  
is the event?

**Yes**

# MouseEvent.CLICK

This is a public static constant defined in the MouseEvent class.

**CLICK** is one of many  
events defined in the  
**MouseEvent** class.

```
public class MouseEvent(){  
  
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";  
  
}
```

*\*There are more events. Please see Language reference for complete list*

**What does public static  
constant mean?**

# public

Accessible anywhere

# static

Can be used without instantiation.

# constant

A value that never changes.

```
// MovieClip is not static and requires
// instantiation to modify or access properties
// and methods
public var mc:MovieClip = new MovieClip();
trace(mc.x);
// outputs 0

// MouseEvent can be accessed any time (public)
// without instantiation (static) so long as it's
// imported and you will always get the same
// value (constant)

trace(MouseEvent.CLICK);
// outputs click
```

# Quick Review

- Event - a notification
- Event Listener - registered target that listens for event notification
- Event Handler - response from the registered target when the listener receives event notification

**How do events send  
notifications?**

# Dispatch Event

Sends a notification to anyone that is listening.

```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

**But where is the  
dispatchEvent() called?**

**dispatchEvent() is  
strategically placed  
throughout the built in  
Adobe Flash Classes.**

**Deep in the built in  
classes flash player  
understands a  
mouse click.**

# When it happens an event notification

```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

**is sent out.**

# Demo of notification system



**So you have some events**

# Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

So you have some events

# Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

So you have some events  
We want to listen and  
respond to a mouse click.

Event  
Registration

Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

So you have some events  
We want to listen and  
respond to a mouse click.

# Event Registration

# Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

Event  
Registration

Event  
Target

Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

Event  
Registration

Event  
Target

```
mc.addListener(MouseEvent.CLICK, onClick);
```

Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

# Listener

Event  
Registration

Event  
Target

```
mc.addListener(MouseEvent.CLICK, onClick);
```

# Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

# Listener

Event  
Registration

Event  
Target

```
mc.addListener(MouseEvent.CLICK, onClick);
```

# Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

## Listener

Event  
Registration

Event  
Target

```
mc.addListener(MouseEvent.CLICK, onClick);
```

## Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

The target now listens for  
mouse clicks

## Listener

Event  
Registration

Event  
Target

```
mc.addListener(MouseEvent.CLICK, onClick);
```

## Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

The target now listens for  
mouse clicks

# Listener

Event  
Registration

Event  
Target

```
mc.addListener(MouseEvent.CLICK, onClick);
```

# Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

# Listener

Event  
Registration

Event  
Target

```
mc.addListener(MouseEvent.CLICK, onClick);
```

# Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

## Listener

Event  
Registration

Event  
Target

```
mc.addListener(MouseEvent.CLICK, onClick);
```

## Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

An CLICK event has  
been dispatched

## Listener

Event  
Registration

Event  
Target

```
mc.addEventListener(MouseEvent.CLICK, onClick);
```

Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

Event  
Notification

```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

An CLICK event has  
been dispatched

## Listener

Event  
Registration

Event  
Target

```
mc.addEventListener(MouseEvent.CLICK, onClick);
```

## Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

## Event Notification

```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

An CLICK event has  
been dispatched

## Listener

Event  
Registration

Event  
Target

```
mc.addEventListener(MouseEvent.CLICK, onClick);
```

Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

Event  
Notification

```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

An CLICK event has  
been dispatched

# Listener

Event  
Registration

Event  
Target

```
mc.addEventListener(MouseEvent.CLICK, onClick);
```

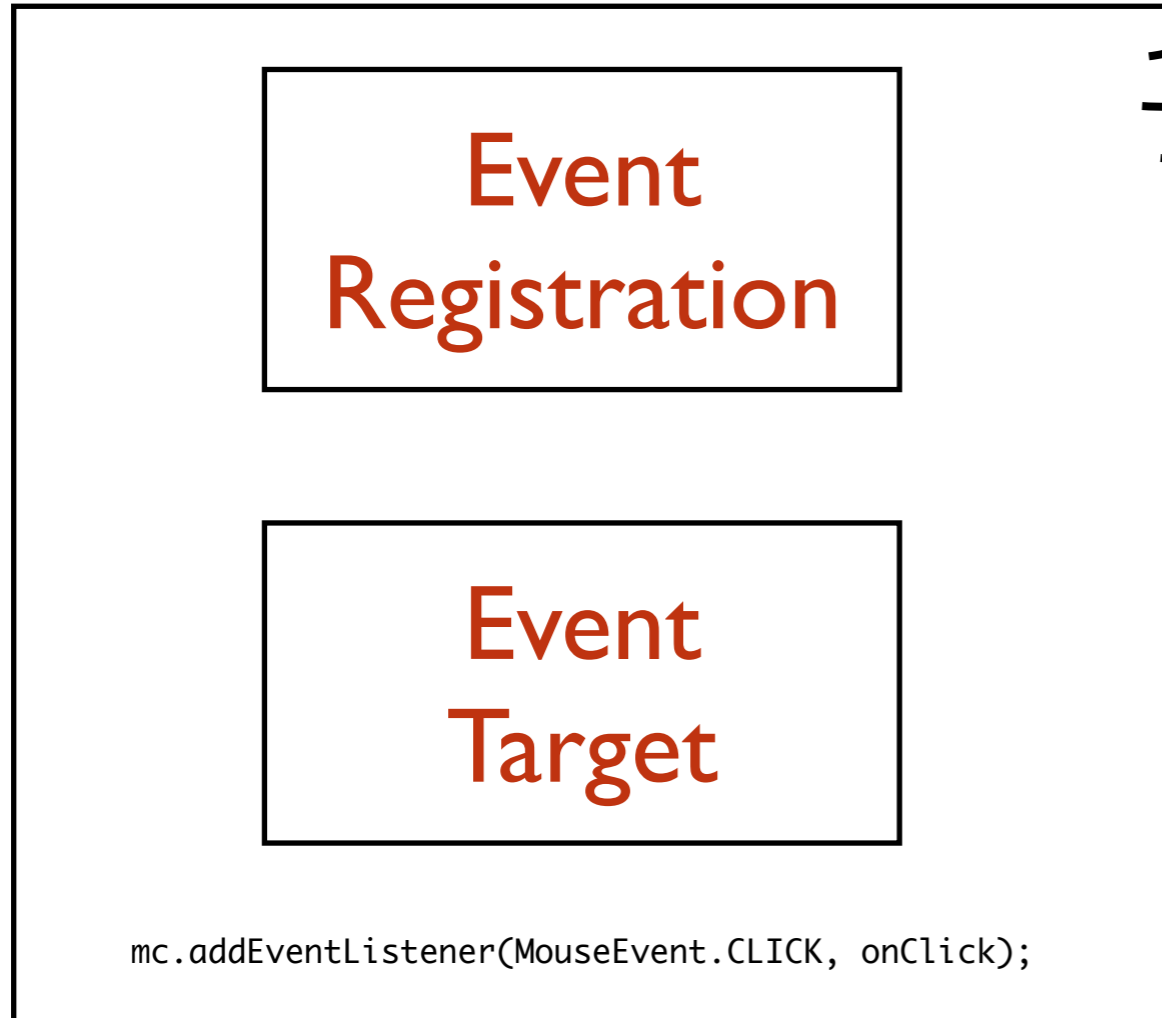
Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

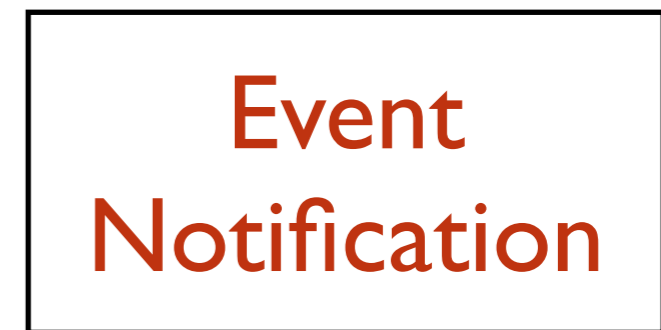
Event  
Notification

```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

# Listener



```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```



```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

Click Event was received

# Listener

Event  
Registration

Event  
Target

```
mc.addEventListener(MouseEvent.CLICK, onClick);
```

# Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

# Event Notification

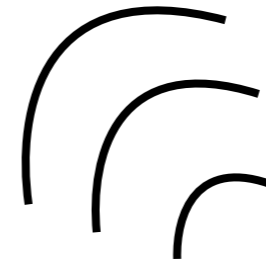
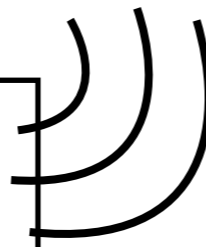
```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

# Listener

Event  
Registration

Event  
Target

```
mc.addEventListener(MouseEvent.CLICK, onClick);
```



# Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

# Event Notification

```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

# Listener



Event  
Registration

Event  
Target

```
mc.addEventListener(MouseEvent.CLICK, onClick);
```

# Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

Event  
Notification

```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

invoke event handler

# Listener

Event  
Registration

Event  
Target

```
mc.addListener(MouseEvent.CLICK, onClick);
```

Event  
Handler

```
private function onClick(e:MouseEvent):void{  
    trace('I have clicked');  
}
```

Event

```
public static const CLICK:String = "click";  
public static const MOUSE_DOWN:String = "mouseDown";  
public static const MOUSE_OUT:String = "mouseOut";  
public static const MOUSE_OVER:String = "mouseOver";  
public static const MOUSE_UP:String = "mouseUp";
```

Event  
Notification

```
dispatchEvent(new MouseEvent(MouseEvent.CLICK));
```

## invoke event handler

# Event Notification

A major part of development for the flash player runtime.

# Types of Events

# Types of Events

- Built in Events
- Custom Events

# Built in Events

Events that are apart of the general ActionScript library.

# Built in Events

- Mouse
- Keyboard
- EnterFrame
- Complete
- Progress
- Focus
- Timer

# Custom Events

Defined by the developer.

# Custom Events

- GAME\_OVER
- END\_LOADING
- CLOSE\_PANEL
- IN
- OUT

# Registering Events in Practice

# Registering Events

Events must be registered in order for the Flash runtime to handle them.

# Registering Events

```
stage.addEventListener(MouseEvent.CLICK,go)
```

# Registering Events

```
stage.addEventListener(MouseEvent.CLICK,go)
```

**stage is the target**

# Registering Events

```
stage.addEventListener(MouseEvent.CLICK,go)
```

**stage wants to register, listen  
and handle a mouse click.**

# Registering Events

```
stage.addEventListener(MouseEvent.CLICK,go)
```

**stage is calling the  
addEventListener method  
which has two arguments.**

# Registering Events

```
stage.addEventListener(MouseEvent.CLICK,go)
```

**The Name of the  
Event Class.**

# Registering Events

```
stage.addEventListener(MouseEvent.CLICK,go)
```

**The event you  
are listening for...**

# Registering Events

```
stage.addEventListener(MouseEvent.CLICK,go)
```

**...and the responding  
function to be invoked  
once the event happens.**

# Registering Events

```
stage.addEventListener(MouseEvent.CLICK,go)
```

**When the mouse clicks  
the stage a function named  
go will execute.**

# Event Handler

# Event Handler

```
stage.addEventListener(MouseEvent.CLICK,go)
```

**go is the Event Handler**

# Event Handler

```
stage.addEventListener(MouseEvent.CLICK,go)
```

**And must be defined.**

# Event Handler

```
stage.addEventListener(MouseEvent.CLICK,go)
function go(e:MouseEvent):void{
    trace("Mouse Click Event")
}
```

# Event Handler

```
stage.addEventListener(MouseEvent.CLICK,go)
function go(e:MouseEvent):void{
    trace("Mouse Click Event")
}
```

**The Event Handler  
declaration.**

# Event Handler

```
stage.addEventListener(MouseEvent.CLICK,go)
function go(e:MouseEvent):void{
    trace("Mouse Click Event")
}
```

**When the Mouse is clicked...**

# Event Handler

```
stage.addEventListener(MouseEvent.CLICK,go)  
function go(e:MouseEvent):void{  
    trace("Mouse Click Event")  
}
```

**...run this function.**

# Event Handler

```
stage.addEventListener(MouseEvent.CLICK,go)
function go(e:MouseEvent):void{
    trace("Mouse Click Event")
}
```

**Resulting in a trace.**

**Questions?**

# Summary

- Understand event notification system
- Register targets to listen and respond to events
- Identify and leverage built-in events