

# Writing Classes }

*Advanced Multimedia Technologies*

# Writing Classes

- { Basic Rules
- { Anatomy of a Class
- { Instantiation
- { Write a Class
- { Alternative Instantiation



# Basic Rules

# Basic Rules

**Always** make sure these things have the same name

- { File name
- { Class name
- { Constructor

You must **instantiate** most classes.



# Anatomy of a Class

# Anatomy of a Class

All Classes have at least

- { Package
- { Class Name
- { Constructor

# Anatomy of a Class

```
package{  
    public class ClassTest{  
        public function ClassTest(){  
            // constructor  
        }  
    }  
}
```

# Anatomy of a Class

```
package{
```

```
public class
```

```
public function ClassTest() {
```

```
// constructor
```

```
}
```

```
}
```

```
}
```

a package is the folder  
where the class file lives.

# Anatomy of a Class

```
package{
```

```
public class
```

```
public function ClassTest() {
```

```
// constructor
```

```
}
```

```
}
```

```
}
```

Currently the package is pointing to the same directory as the .fla.

# Anatomy of a Class

```
package{
```

```
    public class ClassTest{
```

```
        public
```

```
        // const
```

```
    }
```

```
    }
```

```
}
```

**Class definition**

# Anatomy of a Class

```
package{  
    public class ClassTest{  
        public function ClassTest(){  
            // constructor  
        }  
    }  
}
```

The constructor runs automatically when the class is instantiated

# Anatomy of a Class

```
package{  
    public class ClassTest{  
        public function ClassTest(){  
            // constructor  
        }  
    }  
}
```

**Don't forget to name  
the .as file ClassTest.as**



Instantiation

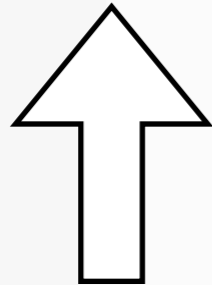
*Instance -*  
A copy of a class

# Instantiation

```
var varName:ClassTest = new ClassTest();
```

# Instantiation

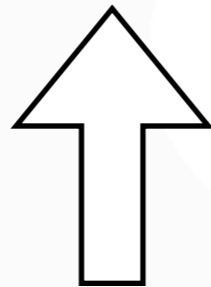
```
var varName:ClassTest = new ClassTest();
```



**Object/Instance Name**

# Instantiation

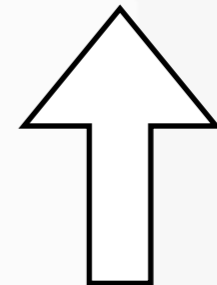
```
var varName:ClassTest = new ClassTest();
```



The Data Type is the  
name of the Class

# Instantiation

```
var varName:ClassTest = new ClassTest();
```



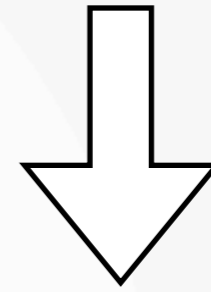
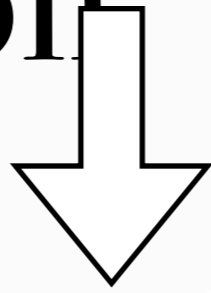
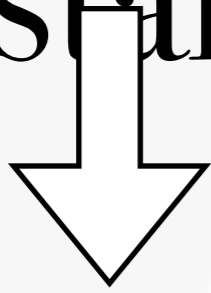
**Creates  
the instance/object**

# Instantiation

```
var varName:ClassTest = new ClassTest();
```

varName is an instance  
of the ClassTest class

# Instantiation



```
var varName:ClassTest = new ClassTest();
```

*object/instance*

*dataType*

*Object Creation*

**Memorize this pattern!**  
It's the same for every  
object in AS 3.0



# Write a Class

# Write a Class

- { Create a new .as file
- { Write the Class
- { Create a new .fla
- { Instantiate the Class
- { Compile

Create new .as file }

# Write the Class

```
package{  
    public class ClassTest{  
        public function ClassTest(){  
            trace("class instantiated")  
        }  
    }  
}
```

Make sure it's saved correctly! }

Create a new *AS 3.0 .fla* }

# Instantiate the Class

```
var varName:ClassTest = new ClassTest();
```

*Write this on the 1st frame of the .fla*

Compile }

You should see the trace message }

Congrats! This is your first Class! }



# Alternative Instantiation

# Doc Class

*Instantiate before the first frame.  
Sometimes you want classes to start  
working as soon as your application  
starts running. It's like having a  
**configuration** file.*

# Doc Class vs Timeline

**timeline  
instantiation**

**During Runtime**

**docClass  
instantiation**

**Before Runtime**

# Doc Class vs Timeline

timeline  
instantiation

During Runtime

Code in the .fla executes  
while the .swf is running.

# Doc Class vs Timeline

Code in an .as file  
executes before the  
1st frame.

`docClass`  
instantiation

Before Runtime

# Creating a Doc Class

```
package{  
import flash.display.MovieClip  
public class ClassTest extends MovieClip{  
    public function ClassTest(){  
        trace("class instantiated")  
    }  
}
```

# Creating a Doc Class

```
package{  
import flash.display.MovieClip  
public class ClassTest extends MovieClip{  
    public fun (...) {  
        trace("cl")  
    }  
}
```

Note the import  
and extension of  
the MovieClip.

**Classes can import functionality  
from other classes.**

**By default a class assumes nothing.**

# Creating a Doc Class

```
package{  
import flash.display.MovieClip  
public class ClassTest extends MovieClip{  
    publ  
    trac  
}  
}
```

**Importing MovieClip allows for the methods and properties of the MovieClip Class to be called/instantiated.**

# Write the Class

```
package{  
import flash.display.MovieClip  
public class ClassTest extends MovieClip{  
    publ  
    trac  
}  
}
```

**ClassTest is extending MovieClip. Just as Spiderman extended Superman.**

# Inheritance Explained

**Class: Spiderman**

**Base Class: superhero.Superman**

**Spiderman and Superman both have powers. Spiderman would “inherit” all of the powers of Superman and have both powers.**

**Anytime we call the Spiderman class we would benefit from the abilities of both Spiderman and Superman.**

# Creating a Doc Class

```
package{  
import flash.display.MovieClip  
public class ClassTest extends MovieClip{  
public  
trac  
}  
}
```

**ClassTest can now do everything  
MovieClip can do. Plus all of the  
functionality from ClassTest.**

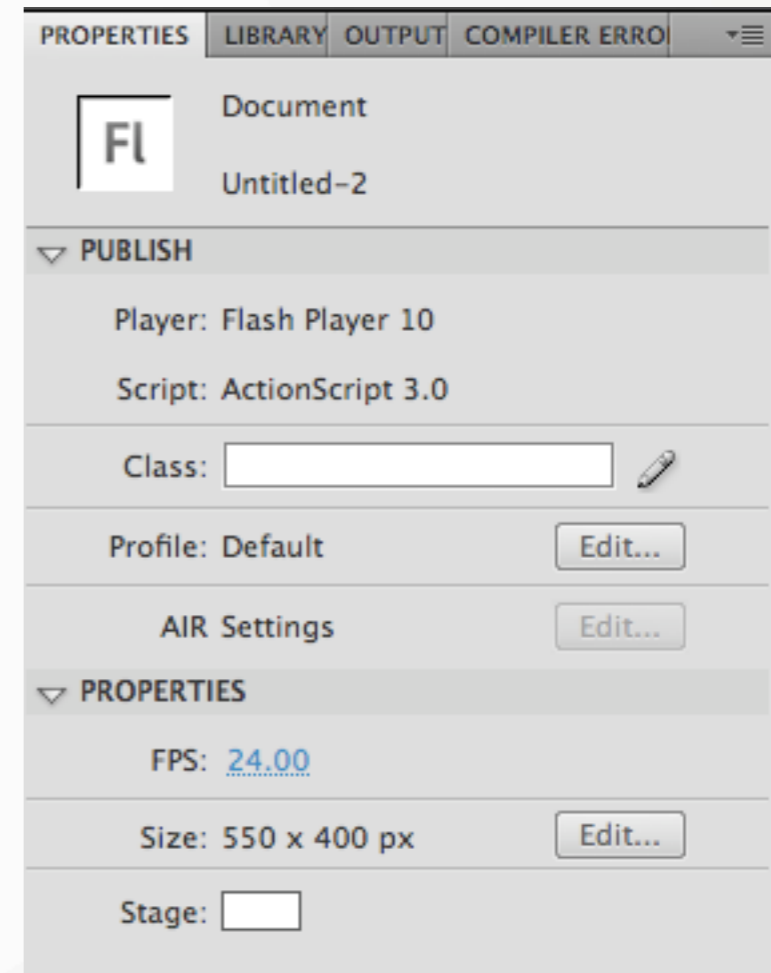
# Creating a Doc Class

```
package{  
import flash.display.MovieClip  
public class ClassTest extends MovieClip{  
public  
trac  
}  
}
```

**A MovieClip or Sprite is the minimum requirement for compiling a .swf when using the Doc Class workflow.**

# Using the Doc Class

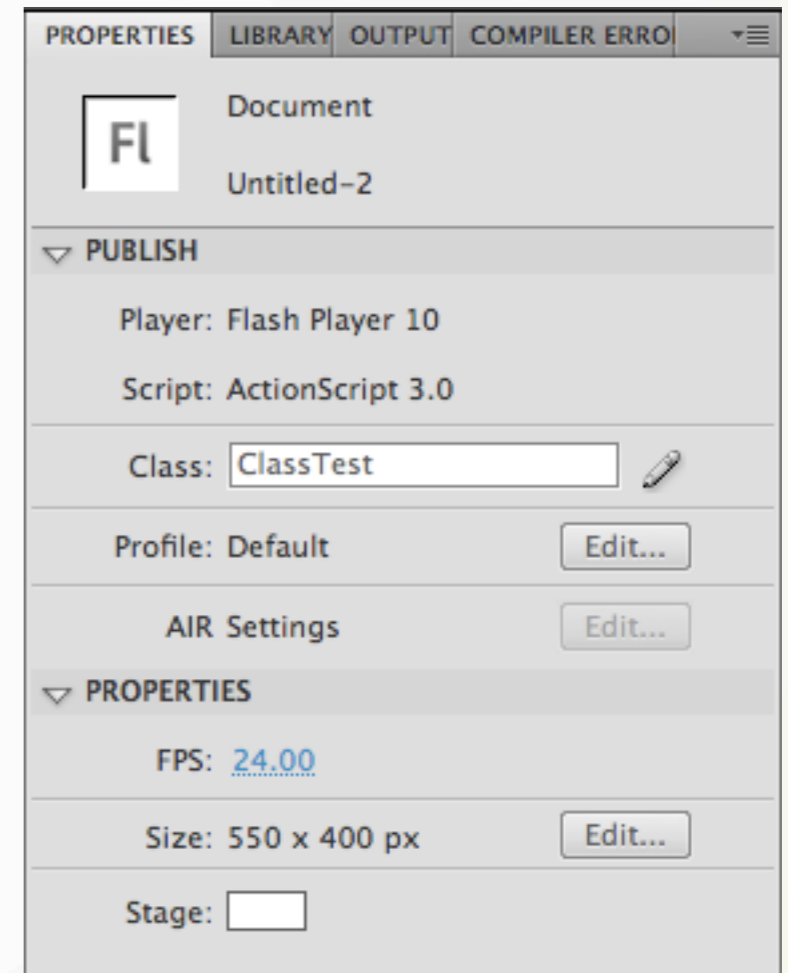
- { Create a .fla
- { Go to the properties window



# Using the Doc Class

- { Create a .fla
- { Go to the properties window

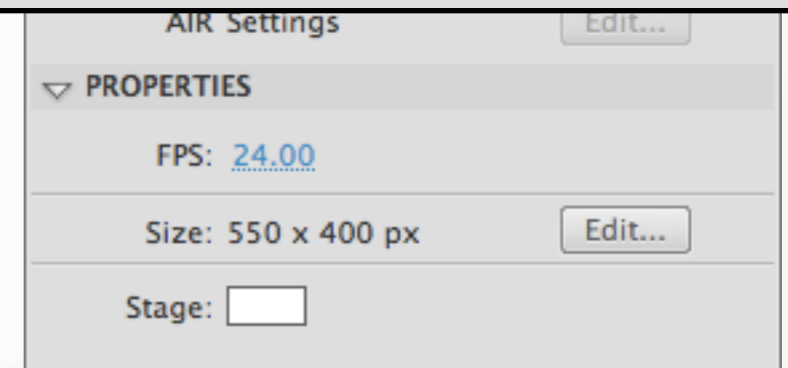
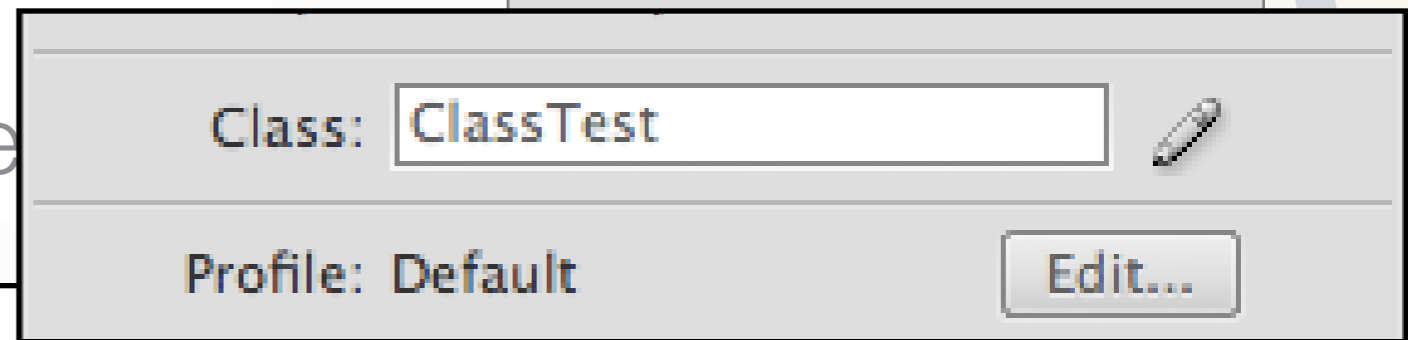
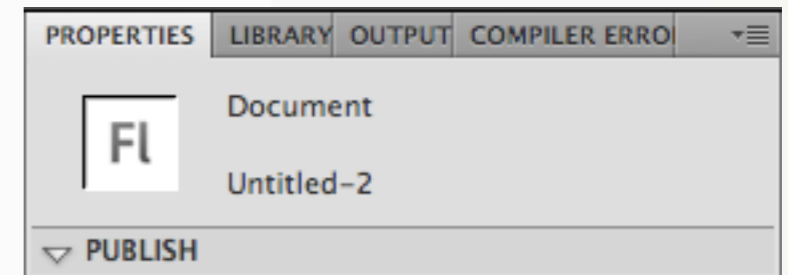
**Enter the name of the class into the Class: field.**



# Using the Doc Class

- { Create a .fla
- { Go to the properties

**Enter the name of the class into the Class: field.**



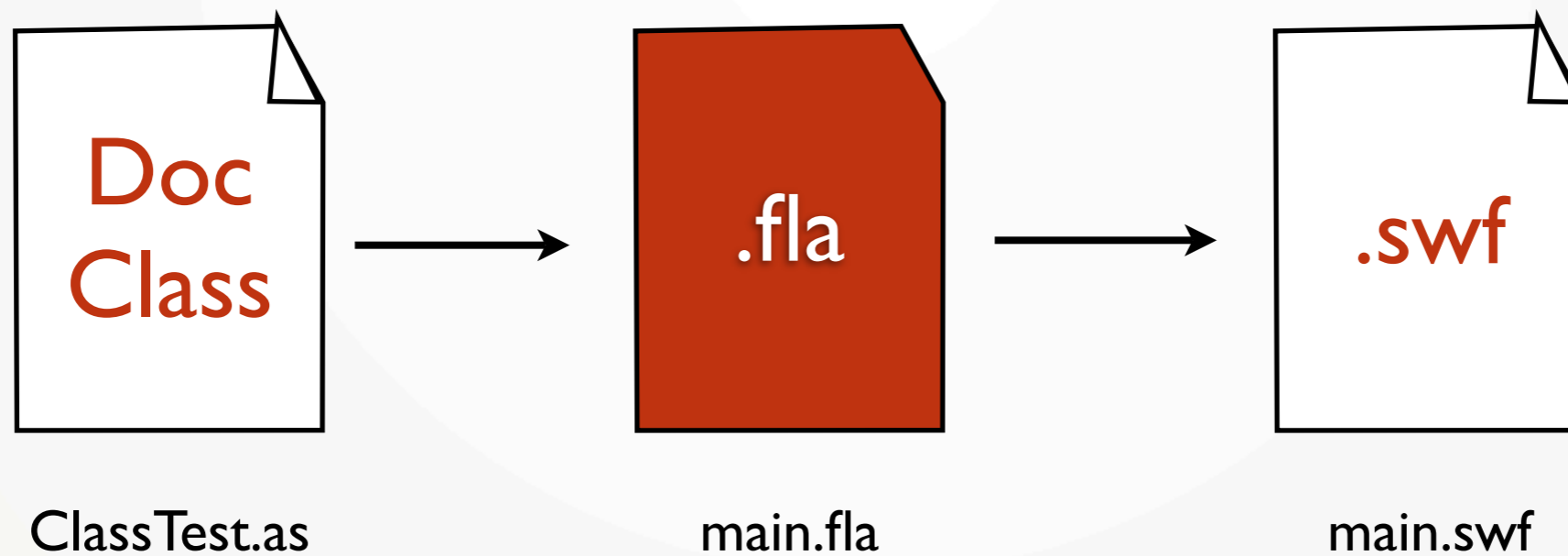


Compile.

Congrats! You have just  
instantiated a Doc Class.

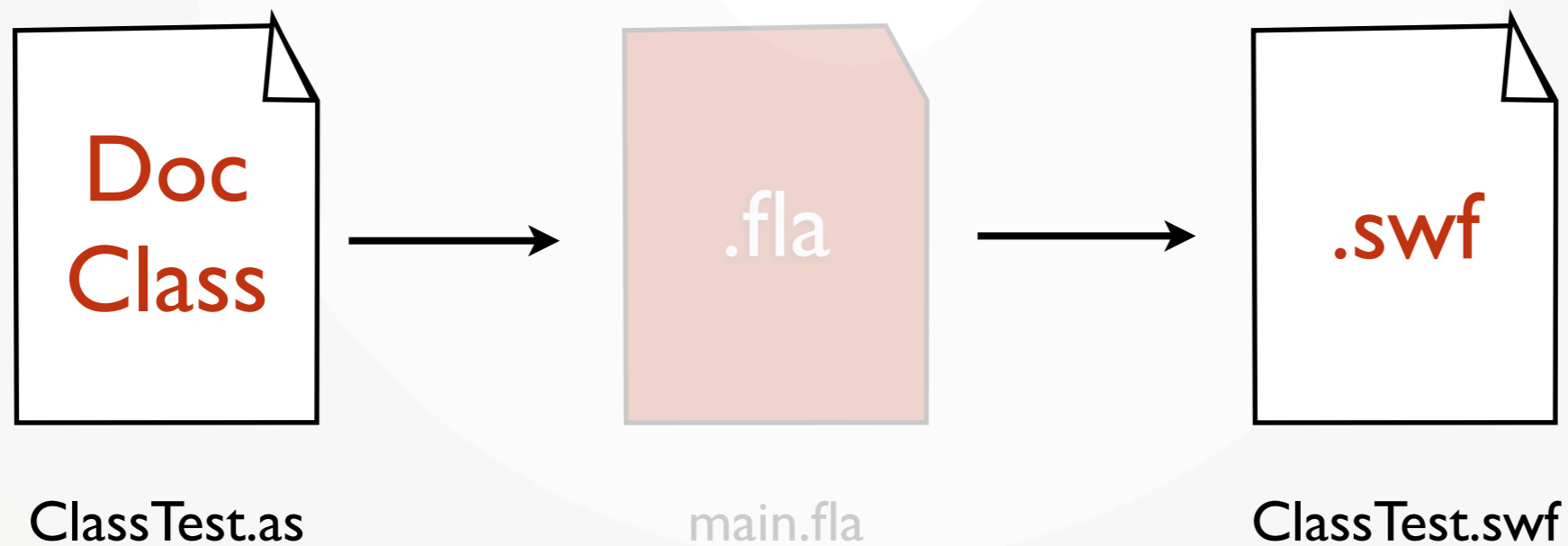
**You can NOT instantiate the  
same class as a Doc Class  
*and* durring runtime.**

# Flash IDE Work Flow

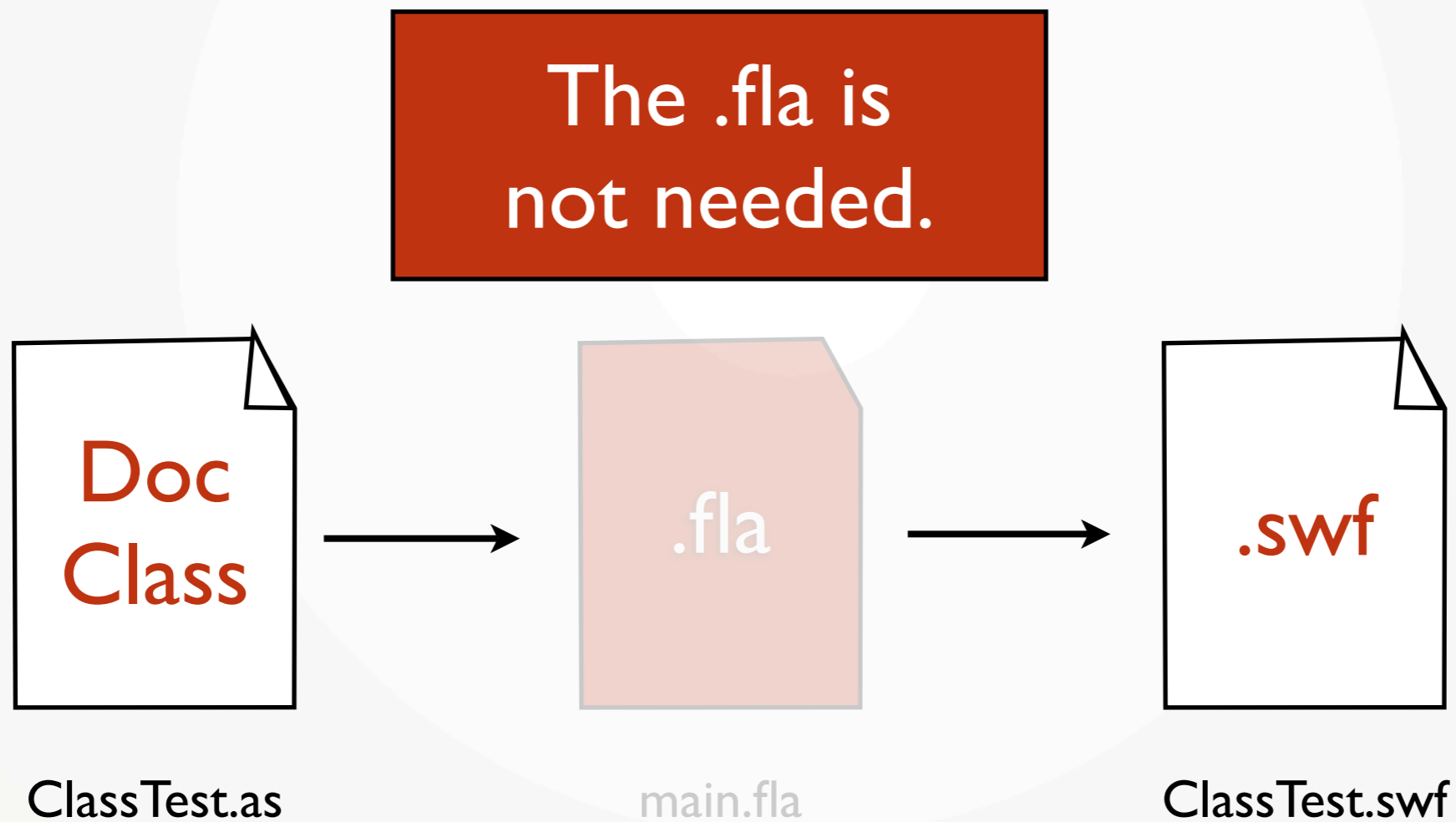


**The Doc Class process is how Flex Builder can compile .swf's without the aide of a .fla.**

# Flex Builder Work Flow



# Flex Builder Work Flow



# Flex Builder

- { Open Flex Builder
- { Create an ActionScript Project
- { Compile (play button)

# Flex Builder

- { Open Flex Builder
- { Create an ActionScript Project
- { Compile (play button)

**This is the same as using  
the Doc Class in the Flash IDE.**

# Writing Classes

- { Basic Rules
- { Anatomy of a Class
- { Instantiation
- { Write a Class
- { Alternative Instantiation