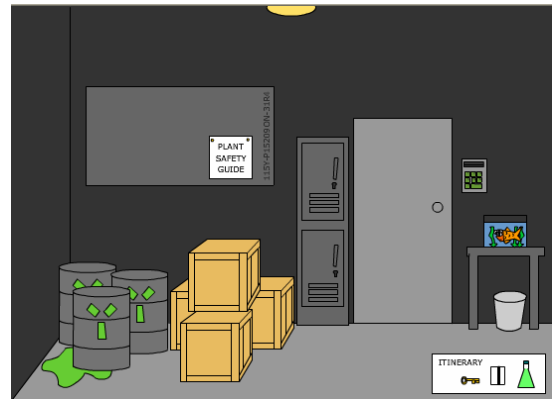




## MAKE AN ESCAPE THE ROOM GAME IN FLASH 8

In this tutorial, you will learn how to create an 'Escape the Room' game in Flash. An 'Escape the Room' game is one where the player finds themselves locked in a mysterious room. They must then explore the room to locate objects that will enable them to escape the room.

The 'Escape the Room' game genre was made popular by the Japanese game 'Crimson Room' that was released on the internet in 2004. This game is available on the internet and can be quite challenging to play.



This tutorial will show you how to create a simplified version of this type of game. You will create a collection of objects that can be moved throughout the room and will hide objects in different locations within the room. Once you have collected the three objects, you will be able to escape the room through the door. On completion of this tutorial, you will have learnt how to:

- create symbols in Flash and give them instance names
- add actionscript to the timeline
- declare and track variables
- change the properties of instances used in the game
- create and set up navigation between multiple scenes

Whilst the game we will build is quite simple, once you have developed these skills, you'll be able to extend the game or create other games with more complexity. You can also use these skills to create adventure games with multiple rooms and locations. You could also enhance your game by building in an interesting narrative, adding sound or using more intricate or detailed graphics. An example of the game you will create is located in the Escape the Room section of the Flash Classroom gallery at [www.flashclassroom.com](http://www.flashclassroom.com).

### PLANNING YOUR GAME

To be able to work through this tutorial successfully, you will need to plan your game carefully. You can be creative in developing a context for your room, however please limit your game to having only the following features at this point to enable you to have a working game at the conclusion of this tutorial. Develop a rough storyboard for your game, ensuring that you include all of the following.

In the main room scene the player must escape from, you will have:

- a background containing the walls and objects that can't be moved
- a locked door
- three objects that the player must locate to be able to open the door
- a collection of objects that the player can drag and drop to other locations
- a space where inventory items (the items the player locates) can be displayed

In addition to this, you will have an introduction scene that contains a short narrative sequence containing four to five lines e.g. You have awoken in a strange room. The door is locked. It is dark. Can you escape the room?

You will also need to design the scene that the player reaches once they escape the room. We will keep this simple at this point by just having some text that says something like 'You've escaped the room' and a button that enables the player to start again.



## FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH 8

Now that you have your plans in place, we'll start building the game in Flash.

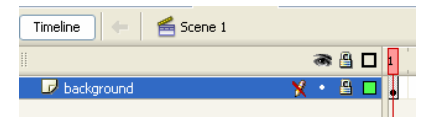
### PART 1—MAKING THE ROOM BACKGROUND

Let's begin by creating the main room for the game that the player must escape from. We will initially create a background layer where we will draw and place all of the features of the room that are not interactive.

1. Open a new document in Flash by selecting **File > New > Flash Document**.
2. In the timeline, double click on the text **Layer 1** and rename the layer **background**.
3. Use the draw tools at the left of the stage to **design** the floor and walls for the room. On this layer, you can also include any objects that will not be interactive. In my example shown above, these objects include a notice board, a locker, a light, a table, a goldfish bowl and a white rectangle for the inventory. The inventory will hold the items the player finds in the room.
4. Once you are happy with your background, **lock the background layer** by clicking on the dot on the background layer that is underneath the lock icon.



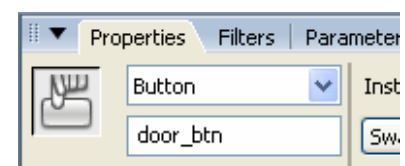
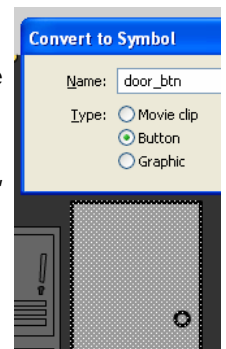
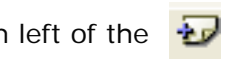
The room containing the inventory.



### PART 2—MAKING THE DOOR FOR THE ROOM

At this point, we are focusing on creating the visual elements or objects in the room. We will add the script that will make these interactive at a later stage. Let's begin by designing the door for our room.

5. Create a new layer by selecting the **Insert Layer** button at the bottom left of the timeline.
6. **Rename** this layer door by double clicking on the text Layer 2 and typing in the word door.
7. Use the **draw** tools to design a door for your room. Once you are happy with the look of your door, select the door and press **F8** to open the **Convert to Symbol** box.
9. Type in the name **door\_btn** and select **Button** for the type. Note that the **\_btn** part of the name is what we call a naming convention. These are used by developers in Flash to easily keep track of what type of symbol the object is. In this case, **\_btn** indicates it is a button. If it was a movieclip, we would have called it **door\_mc**. This isn't essential, but is good practice.
10. Our final step involves allocating an instance name for the door. When we add script to our game to make it interactive, we refer to the objects using their instance name. So even though we have named our objects when we converted them to symbols, we need to give them an instance name. To give the door an instance name, click on the door and in the **instance name cell** of the **properties panel**, type in the name **door\_btn**.





### PART 3—MAKING THE DRAGGABLE OBJECTS FOR YOUR ROOM

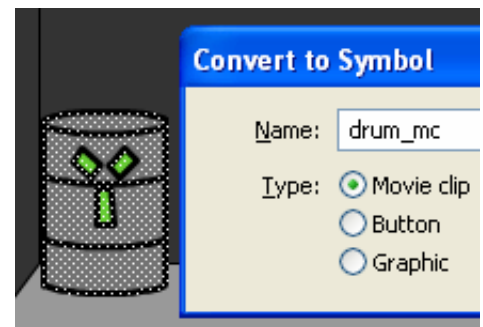
In our simple version of this genre of game, we will be simply hiding different items behind other objects that the player will be able to drag out of the way. In my game, shown on the first page, these objects include drums, crates, a bin and a safety guide that is pinned on the notice board. Note that even though there are three drums, I have only created one drum symbol and then copied it twice. I have then given each drum a different instance name e.g. drum1\_mc, drum2\_mc and drum3\_mc.

To create your draggable objects, follow these steps.

11. Create a new layer and **rename** the layer **objects**.
12. **Draw** a picture of each of the objects you want to have in the room for the player to drag. If you are going to have four crates that are identical, just draw one at this stage, we will copy it after we have converted it to a symbol.
13. Select each object and press **F8** to convert your object to a symbol. Give your object a **name** and select the **Movie Clip** type.

My draggable objects are called drum\_mc, crate\_mc, bin\_mc and guide\_mc.

Ensure you have converted each of your objects into a movie clip symbol prior to moving on to the next step.

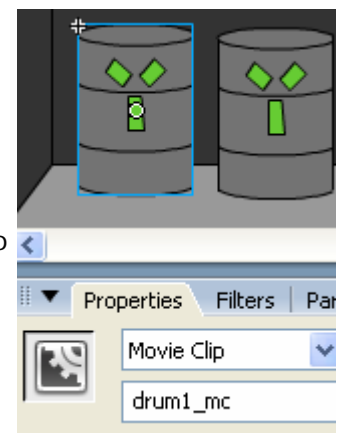


14. It is now time to make copies of any of the objects you want multiple copies of. To do this, either select the object and **Copy** and **Paste** it or simply position your mouse over the symbol, hold down the **Alt** key on the keyboard and drag your mouse to the space next to your object. This creates a second copy of your object and is a very quick and efficient way of copying objects and symbols in Flash.
15. You should now have all of your objects converted to symbols. **Place** them all in the locations you want them in the game.
16. Give each object an instance name by clicking on it and then entering the name in the **instance cell** of the **properties panel**. If you have multiple copies of the one symbol, each symbol will need its own instance name.

For example, in my game I had three drums so these have been given the instance names drum1\_mc, drum2\_mc and drum3\_mc.

Remember that when we write the script for our game, we refer to the instance names that we have given to each instance of a symbol or object on the stage.

**Ensure all of the objects you want to be able to dragged have been given an instance name.**



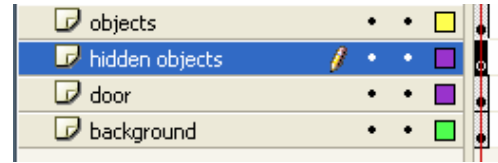
If you are up to this point, you are doing well. We are now going to create the objects that we will hide in the room.



**PART 4 —MAKING THE HIDDEN OBJECTS FOR YOUR ROOM AND INVENTORY**

17. Create a new layer and **rename** the layer **hidden objects**.

18. Click on the layer and drag it so it is under the **objects** layer. This will ensure that your objects are positioned behind the draggable objects in the game. You may want to create additional layers at a later stage to enable the hidden objects to be between draggable objects, however for now, we will stick to just these layers.



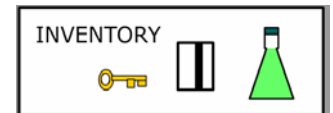
19. Ensure the hidden objects layer is selected and **draw** a picture of each of the objects that the player will have to find in the room. Don't hide these at this stage as we have a few more things to do.

20. Select each object to be hidden and press **F8** to convert your object to a symbol. Give your object a **name** and select the **Movie Clip** type.

My hidden objects are called key\_mc, swipecard\_mc and potion\_mc.

Ensure you have converted each of your objects into a movie clip symbol prior to moving on to the next step.

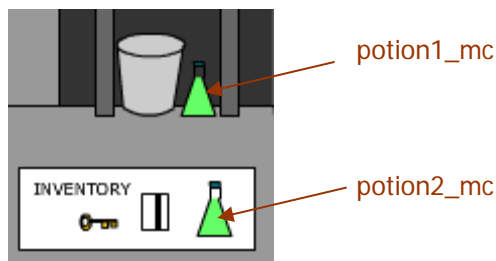
21. Even though you will probably have only one copy of each hidden object for the player to find, you will need to create a second copy of each object that will be hidden. This copy needs to be placed on top of the inventory area you have included in your background design.



22. We are now going to give our hidden objects and the copies we have placed in the inventory **instance names**.

It is essential to name them in the following way as we will be setting up our script to refer to them in this way.

The copy of the symbol that will be hidden in the room needs to have the instance name that contains the 1. The copy of the symbol that is in the inventory, needs to be given the instance name that contains the 2.



For example, in my game the potion is hidden behind the bin. It has the instance name **potion1\_mc**.

The other copy of it in the inventory is called **potion2\_mc**.

To add the instance names, click on each symbol and enter the **instance name** in the **instance name** cell of the **properties panel**.

Now when the player starts the game, we want the inventory items to be empty as the player will not have found any items. We will set the items in the inventory to be invisible when we add the script for our game.



### PART 5 - SETTING UP DYNAMIC TEXT BOXES TO TRACK VARIABLES

We are nearly ready to set up the script for our game that will make the game interactive. Before we do this, we are going to set up some dynamic text boxes that will display the value of some variables that we will create in the next step.

Before we do this, let's quickly run through how our game will work. When the player starts the game, they will view the introduction scene which will contain the narrative for the game. Once this is completed, they will automatically find themselves in the room they must escape from.

Now in this room, we have placed two types of objects. Objects that the player can drag and drop to move out of the way and objects that are hidden behind these objects. To escape the room, the player must locate all of the hidden objects.

In my game, the player must locate the key, the potion and the swipe card. When the player finds each object, the object becomes invisible in the game area but shows up in the player's inventory. This simulates or infers that the object is now in the possession of the player.

Now to make the game work, we will actually be making the door inactive to begin with. The player will not be able to click on it and escape the room until all of the objects are found. When they find all three objects, the door will become enabled and the user will be able to click on it to go to the final scene. We will set this up using script in the next part of this tutorial.

This script will set up three variables, one for each hidden object. In my game, these variables are called *keylocated*, *potionlocated* and *swipe card located*. **Variables** are often described as containers that hold data or values. In this game, these variables will hold text and the text initially will be set to the word *hidden*. We will add script to make the value *hidden* change to *found* when the player finds each item. We will then add what we call an **if statement** to check if the values of each of our three variables are 'found' and if so, we will make the *door\_btn* become enabled.

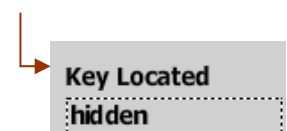
It sounds a bit complicated at first, but you will grasp how it works as we continue working through the tutorial.

#### *Tracking Variables*

To help us see if our game is working, we are going to set up some text boxes at the side of our stage to track if our script is working. These text boxes will track the value of our variables. Many Flash developers do this to help them monitor the value of variables and to ensure their game is working. We are going to do this, so that you can see the value in doing so. It isn't essential though and you can remove these text boxes once your game is complete as the game is not relying on the text boxes to work as we will also declare the variables and values in our script.



23. Create a text box at the side of your stage. In this text box, type in the name of the first hidden object and then type the word Located beside it (as shown below). This is what we call static text. It won't change and we have simply placed it there to enable us to see what hidden object the text box we place below it is referring to.

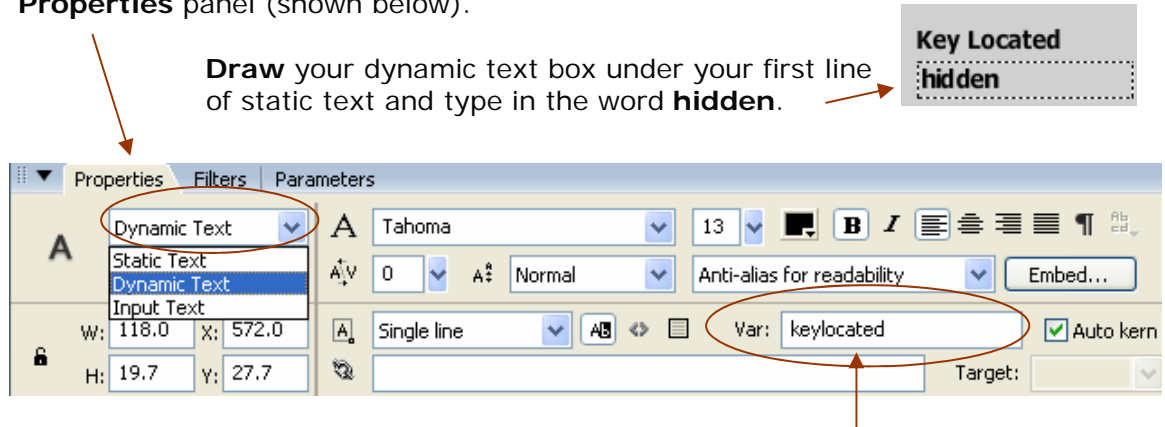




## FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH 8

24. The text box we place below the text we have just added is what we call a **Dynamic Text Box**. This is the type of text box that contains the value of variables. The content of this box or value of the box can be changed when the player interacts with the game. In our case, if they find a hidden object, the value will change from hidden to found. In a shoot-em style game, a dynamic text box could be used to track the score.

To add a dynamic text box, select the **Text Tool** from the tools on the left and then select the **Dynamic Text** option from the text options drop down menu in the **Properties** panel (shown below).



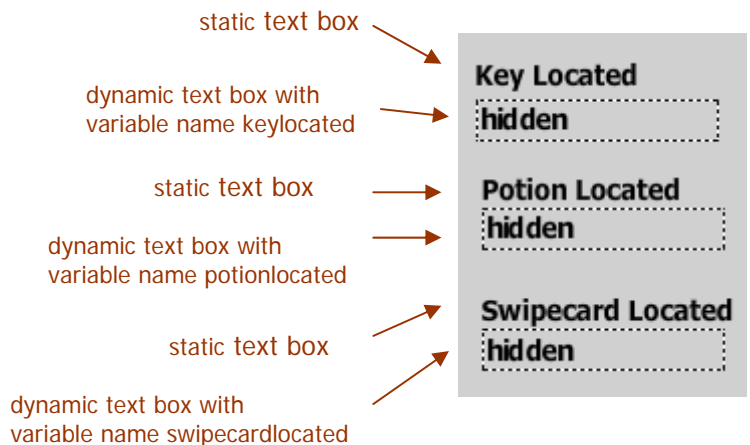
25. To set up a variable name for the dynamic text box, type in a **variable name** in the cell shown here. If your object is a key, give the dynamic text box the name `keylocated`, if it is potion, give it the name `potionlocated`.

Repeat these steps until you have one static text box and one dynamic text box set up for each of your hidden objects.

On completion, your text boxes at the side of the stage should look like this.

If we tested our game at this point, we could drag the edge of the game on that side out to view the text boxes.

At this point, we haven't added any script, so nothing is interactive. We will therefore see the values as we have entered them as 'hidden'.



We have now set up all of the objects, symbols and instance names for our game. It's time to get started on creating the script for the game. If you are new to Flash, you may want to take some time to read through the Flash Classroom 'Actionscript Essentials' tutorial. This will help you understand the 'syntax' of the script we are using.

However, this tutorial is designed to enable you to create your game without knowing all about the script, so it isn't essential to do this. It is just recommended if you are hoping to gain a solid understanding of Actionscript, the scripting language in Flash.



### PART 6 - CREATING THE INTERACTION IN OUR GAME USING ACTIONSCRIPT

We are now ready to add the script that will make our game interactive. Before we get started, I want to mention a couple of things. Firstly, in this tutorial, all of the script we will add will be added to the first keyframe on a layer we called Actionscript on a timeline. Sometimes developers add script directly onto buttons or movieclips and I have done this in the past. If you have completed some of the other games tutorials on the Flash Classroom site, you are probably used to doing this.

We are going to add the actionscript to the timeline for a couple of reasons.

1. It's more efficient as we can go to the one place to change any of the script.
2. The next version of Flash, Flash CS3 doesn't allow you to add script to symbols. The version of actionscript used in Flash CS3 is Actionscript 3 and you are only able to add script to the timeline. Note though that you can open up Flash 8 and other files from previous versions in Flash CS3, however if you create a new CS3 Flash file, you are restricted to the timeline.

I've decided to put all of the script in this game on the keyframe in the timeline for these reasons. It's a good way of preparing us for the changes when we move to the next version.

#### Let's get started with scripting

The following steps will take you step by step through the script used in this tutorial. It will guide you through the process of adding all the script to your game and explain what each part of the script does. The base .fla file for this game is also available in the games tutorial section of the Flash Classroom. You can deconstruct this file to see how it works. I have also added comments to this file to explain how the script works.

26. Add a **new layer** to your timeline and rename this **Actionscript**.
27. On this layer, select the first keyframe and press **F9** to open the **Actions Panel**. This is where we will add the script to our game.
28. To begin with, we are going to add a line of script that tells the Flash Player to stop at this point in the timeline so that the player can play the game. To do this type in the following script. Take care to ensure you use the correct brackets. The ones used in this line are the ones above the 9 and 0 keys on the keyboard.

```
stop();
```

29. The next couple of lines of script we add will declare the door\_btn symbol and set it's enabled property to false. This means that the button will not work until the value is true. The value will be set to change to true once all of the hidden objects are found. Press enter a couple of times to space out your script and then type in:

```
var door_btn:Button;  
door_btn.enabled = false;
```

The text **var** is used in front of the variable name you are declaring. In the first line of the script above, I am setting up the door\_btn symbol as a variable and have stated that it's type is a button. The second line says that of this symbol, the enabled property is false. All of the symbols we create in Flash have properties. These include properties for visibility, alpha, enabled, rotation, scale and x and y locations on the stage.



## FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH 8

30. We are now going to add some more script underneath the script we have added. This script will declare the three variables we want to use to track whether the hidden inventory objects have been found.

As you may have used different objects for your game, I have made the parts of the script you may need to edit to match your own variable names bold. In future script, throughout this tutorial, I will make anything you may need to change **bold**.

```
var keylocated:String;  
var potionlocated:String;  
var swipecardlocated:String;
```

In each of these lines, I am using var to tell Flash I am declaring a variable and am then entering the name of that variable. I am then adding a colon and am following this by the word String. The word String indicates the type of the variable. String is the name of the type used to indicate that the variable contains text.

31. Below these lines of script, we will declare the initial value of these variables. In our case, the value is hidden. The word hidden is surrounded by quotation marks to indicate that it is a string of text. Enter the following, making sure you replace the bolded text with the names used in your own game.

```
keylocated = "hidden";  
potionlocated = "hidden";  
swipecardlocated = "hidden";
```

32. Press enter a few times to space out your script to make it easier to view and edit. We will now declare the status of the movieclip symbols that are part of the game. We are doing this so that in a future step, we can add script to make each of these items draggable.

For each instance on your stage, type in the following line of script and replace the bolded name with the name of your instance.

```
var drum1_mc:MovieClip;
```

Double check to ensure you have added a line for every one of the draggable and hidden objects in your game. Don't forget the instances in your inventory.

33. Press enter again a couple of times and now add the following block of script for each of the instances or objects in your game that you want to be draggable.

```
crate1_mc.onPress = function(){  
    startDrag(this);  
}  
  
crate1_mc.onRelease = function(){  
    stopDrag();  
}
```

Double check to ensure you have added and edited this block for every draggable object.



## FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH 8

We are about to enter into the stage where we add the more intricate part of our script. Before we do this, let's recap and think about the script we have added so far and why we have done so. So far we have:

- added a stop action to make the Flash Player stop at the keyframe containing the room.
- set up the door\_btn instance as a variable, declared it to have a button type and set it's initial enabled property to false. This means that it is on the stage and that it is a button, but it can't be clicked on.
- set up the variables that will track when hidden objects have be found. We have declared that these variables are of a text type and that their initial value is 'hidden'.
- set up all of our draggable objects as variables and declared that they are of movieclip type.
- added a block of script for each draggable object to allow the player to drag and drop the object around the room.

Take a moment now to play your game. To do this select **Control > Test Movie** and then try to drag each of your draggable objects around the stage. Make sure you can drag and drop each of them. If there are problems, double check your instance names and your script to ensure it all matches.

### *Adding more advanced interaction to our game*

- 34.** Ensure your actions panel is open (F9) and add the following lines of script. Once again, replace the parts that are in bold with the instance names you have used.

These lines of script relate to the visibility of the objects in our inventory. As we want the items in our inventory to be invisible to begin with, we are setting the initial value of the visible property of each of these movieclip symbol instances to false. Note that you will only see this when you test or play your game. It won't affect the visibility of the objects in the inventory when you are creating your game.

```
key2_mc._visible = false;  
potion2_mc._visible = false;  
swipecard2_mc._visible = false;
```

- 35.** We are now going to add the blocks of script that will make the hidden objects invisible when clicked on, and the matching items in the inventory visible when this happens. Remember that this is to simulate that the player has found the object and has picked it up. The final line of the script changes the value of the variables we have set up to track the status of each hidden object from 'hidden' to 'found'. Note that the three middle lines of script are only ran or worked through when the player presses their mouse on the hidden movieclip.

```
key1_mc.onPress = function(){  
    key2_mc._visible = true;  
    key1_mc._visible = false;  
    keylocated = "found";  
}
```

Ensure that you have added a block of script like this and have modified it for each of your hidden objects. You should now test your movie by selecting **Control > Test Movie**. In your game, you should be able to click on your hidden objects and see them disappear and then show up in your inventory. If not, check your script and instance names.



## FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH 8

The good news is that we have nearly finished adding the script for the room scene of our game. If you are new to scripting, don't feel intimidated by writing script. After you work with it for a while, you begin to understand what you are writing and how it works. Initially, you will probably be just copying and typing it in and making small changes. This is all part of the learning process so don't feel disheartened.

Let's keep going....

36. Now remember how we made our door and gave it the instance name `door_btn`. In a previous block of script, we set up a variable for the door and told Flash that it had a button symbol type. We also told Flash that the `enabled` property of the door was `false`. This means that it isn't an active button. The user can't click on it to move to a new frame or scene.

We are now going to add three lines of script which will tell Flash the scene that it should take the player when the button is clicked. In this case, we are going to set the button to take the player to a scene called **youescaped**.

If we test our game, will our button do this? Ofcourse not. This is because firstly, we haven't created that scene and secondly, our button is still not enabled. We will change this in upcoming steps.

Add the following script for the button:

```
door_btn.onPress = function(){
    gotoAndStop("youescaped",1);
}
```

Always double check your script and the brackets and instance names you have used. If you think it's right, continue on to the next step.

37. The following five lines of code form the final part of the script for the room scene in our game. These lines contain an if statement that checks if all of the hidden objects have been located. Let's deconstruct the script to see how this works.

The first line of script contains what we call an Event Handler. When the event it handles occurs, it will run the lines of script in the middle section between the parentheses. (I often call the parentheses the squiggly brackets). The event that needs to occur to run the middle lines is related to the object we have given the instance name `key1_mc`. You can infact use any of the instance names you have used in your game. The script says that each time Flash enters a frame this object is in, to run the if statement. Our game is set to 12 frames per second, which means that the condition in our if statement (that all our objects are found) will be checked 12 times each second.

The next lines contain the if statement and the condition. It is checking if our three variables have changed from hidden to found and if so, it will enable the next line to be run. This next line will make the `door_btn` instance enabled so that the player can escape the room. If all of the objects aren't hidden, it will keep testing this block 12 times a second until they have.

```
key1_mc.onEnterFrame = function(){
    if((keylocated eq "found")& (potionlocated eq "found")&
    (swipecardlocated eq "found")){
        door_btn.enabled = true;
    }
}
```



## FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH 8

We've now completed the room for the game. You can test your game now by selecting **Control > Test Movie**. You should be able to drag and drop all of the draggable objects and click on the three hidden objects to see them disappear and become visible in the inventory. Once you have found all three hidden objects, you should also be able to roll your mouse over the door and see your cursor become a hand. This indicates that your door button is now active.

Now at this stage you probably thinking that the game is a bit easy to play. Once you complete this full tutorial, you can go back and modify your game to make it a bit harder to escape the room. For instance, in my game, I could make it harder by making the key invisible to begin with. The player might have to locate and 'drink' the potion first to enable them to have super powers to be able to see the key and other items. This type of interaction can be created using similar script to what you have used so far. You could declare an object such as the key to be invisible to start with and then add an if statement to test if the player has located the potion. If so, the key could become visible.

Don't have a go at this at this stage, but once you've finished your game, save it as another version and try to extend your game by setting some of this more advanced interaction up. These more intricate relationships between objects is what makes an 'Escape the Room' game challenging.

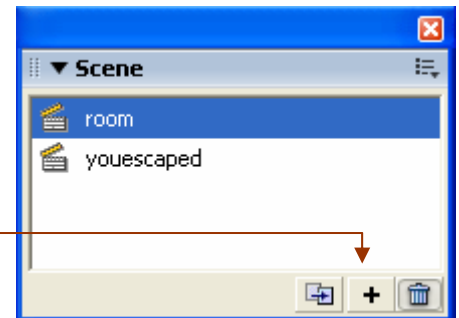
### PART 7 - ADDING THE SCENE THE PLAYER REACHES WHEN THEY ESCAPE

Let's add another scene to our game. This scene will contain some positive feedback for the player and will contain a button that they can press to restart the game.

38. Hold down **Shift + F2** to open the **Scenes Panel**.

39. To begin with you will only have one scene named Scene 1. This is the scene containing your room. Double click on the text **Scene 1** and **rename it room**.

40. Click on the **+ sign** at the bottom of the scenes panel to add another scene.



41. **Rename** this new scene **youescaped**.

Remember that we refer to this scene as this name in the script we added in previous steps so don't get creative. Flash is also case sensitive so ensure you don't add any capitals or spaces. It needs to match the script exactly.

42. Press on the **youescaped** scene to select it.

43. In this scene, **add some text** such as 'Well done! You have escaped the room'. If you have developed a detailed story or narrative for your game, you might want to add more text. However, for the moment, just stick with something simple. You can edit the scene later.

44. Use the **draw tools to create a button**. This is the button that the player will press to restart the game. Draw what you would like your button to look like. Press **F8** to open the **Convert to Symbol** box and then give your button the name **startagain\_btn** and select the **Button** type option. Click OK.




45. Select the button on the stage and in the instance name cell of the properties panel, type in the name **startagain\_btn**. We will refer to this name in the next step.



## FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH 8

To save time, we won't set up the button to change on rollover. If you know how to edit a button and change the button states, feel free to do so. However, in this tutorial, we'll just move on to adding the script to make the button work.

44. If you look at your timeline, we currently only have one layer named **Layer 1**. **Rename** this layer **content**.
45. Now let's add a second layer by selecting the **Insert Layer** button at the bottom left hand corner of the timeline. 
46. Rename this new layer **Actionscript** and select the first keyframe on this layer.
47. Press **F9** to open up the **Actions Panel** and enter the following script. The first line will tell the Flash Player to stop at this point in the scene and the other lines tell Flash to take the player to the first frame of a scene called intro when the start again button is pressed.

```
stop();

startagain_btn.onPress = function(){
    gotoAndPlay("intro",1);
}
```

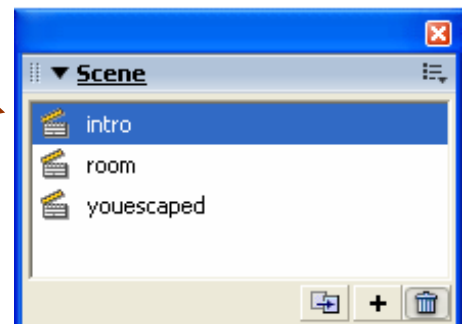
### PART 8 - ADDING THE INTRO SCENE TO THE GAME

OK. Yes. It does seem a little backward, but the final part of making our game involves adding a scene that will form the introduction to the game. The reason I have left this to last is because I found that it is more effective to get your main room done first and also, it's a bit painful running through the introduction each time you want to test the game.

At the beginning of this tutorial you were asked to do some planning. As part of this, you were asked to come up with a short narrative of about 4 - 5 lines in length that could be used as part of the introduction. We are going to add each of these lines to different keyframes in our intro scene so that they appear one at a time. The result will be a short sequence that will create a mood for the game. After viewing this sequence, the player will automatically end up trapped in the room scene.

48. Let's start by setting up a new scene. To do this select **Shift + F2** to reopen the **Scenes Panel**.

49. Add a **new scene** by selecting the plus sign button at the bottom of the panel and **rename** the new scene by double clicking on the text. Name the new scene **intro**.

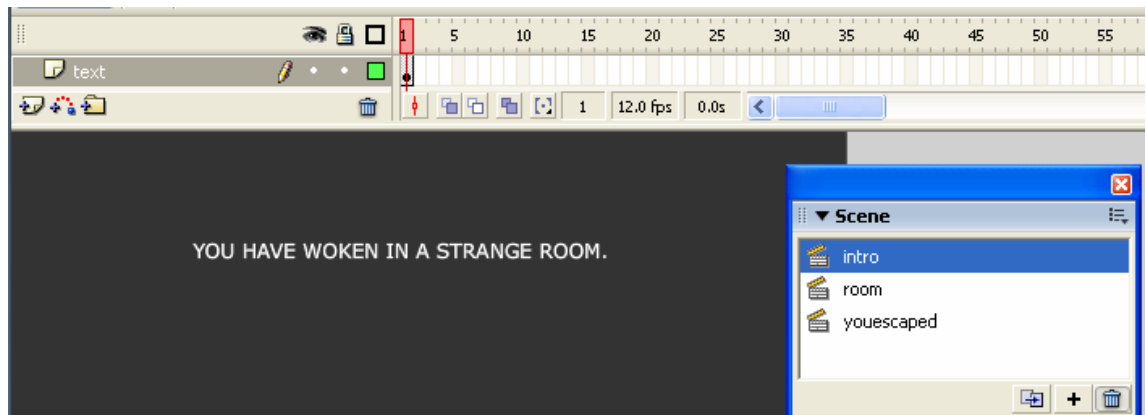


50. Click on this scene and drag the **scene** to the top of the panel. This will mean it will be the **first scene** that the player views.
51. Select this scene and click on the first layer and rename it **text**.
52. Select the **first keyframe** on the timeline and in the centre of the stage, enter the **first line** of your narrative.



## FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH 8

My first line is shown in the screenshot below.



We will now add keyframes at points along the timeline and change the text until we have added all of the lines of our narrative.

53. Click on **Frame 50** and press **F6** to add a new keyframe. Replace your first line of text with your second line.
54. **Repeat this process** by adding another keyframe at say frame 100 and adding the third line, adding another keyframe at 150 and adding the fourth line and so on.
55. Remember to add a final keyframe about 50 frames after your final line. This will ensure this line stays visible long enough for the player to read the text.

You may need to alter the position of the keyframes to enable the text to be read. To do this, you can drag and drop the keyframes to different frames of the timeline to adjust how long each sentence or line is visible. It takes a bit of trial and error and you will need to test your movie to see if the timing is right.

### PART 9 - TESTING YOUR GAME

You've done it. You've completed your game. It's now time to test it. Select **Control > Test Movie** and watch your intro, escape the room and then restart the game.

Does your game work as it should? Are you stuck in the room? Can you drag and drop the objects and locate the hidden items? Are you able to escape once the objects are found?

If so, well done. You have created your first simple 'Escape the Room' game. If not, check through your game, paying particular attention to your instance names and the script. If you have done this and you still have problems, get your teacher to have a look or email me at the address below.

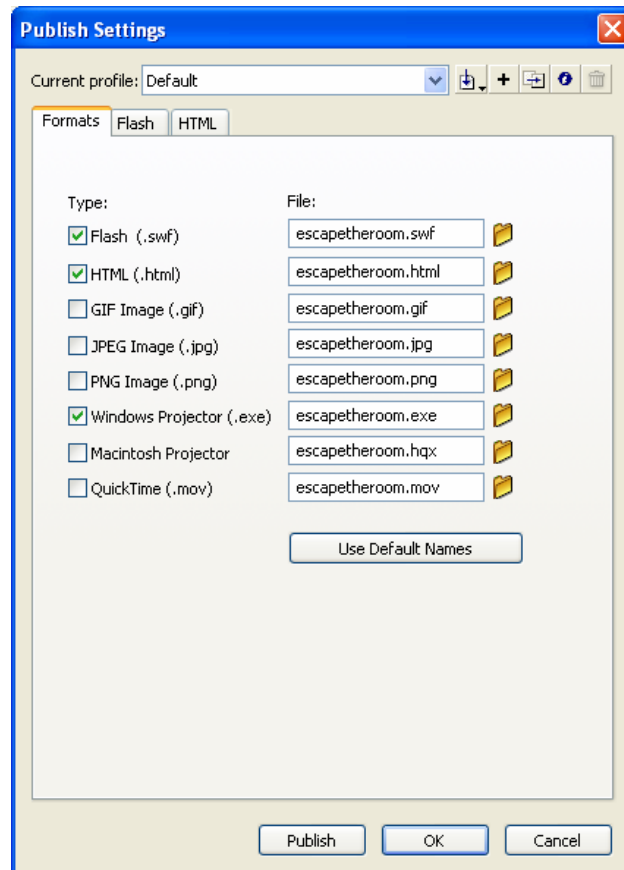
#### So what happens now?

The next page will show you how to save and publish your game so you can share it with friends. Once you've done this, why not make a more intricate Escape the Room game. See if you can adapt the script in this game to extend the game and make it more difficult. This tutorial has helped you develop a range of skills in Flash. Infact, you have developed the skills you need to create an adventure game. So why not begin your own adventure game in Flash and remember we'd love to see any creative 'Escape the Room' or Adventure games that students and teachers make.



## PUBLISH & SHARE YOUR WORK

26. Save your work by selecting **File > Save**.
27. Turn your flash file into a game that can be played on any computer by publishing it in different file formats. To do this select **File > Publish Settings**. The box shown below will appear.



28. Tick the file formats you want and click on the **Publish** button. These files will be saved in the same location you saved your original file. If you want your game to be a standalone file that can be played on Windows or Macintosh machines—ensure you check the Windows Projector (.exe) and Macintosh Projector (.exe) format options.