


A Twine story with variables and an image

2022-06-16

Last Updated 2022-11-20 with Twine 2.5.1

Author: Chris Kerslake (chris@xmodus.com).

Home page for variables.
The key question:
Look for the key!
You have visited the rooms 1 times.
The last page you visited was Room 1.
Num visits testing:
Keep trying
Visit [Room 1](#)
Visit [Room 2](#)
[Restart Game](#)

Using the key, you unlock the door and ... find the Twine logo!

Twine
Return to [Variables Home](#)
[Restart Game](#)

Copyright:



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/)

Contents

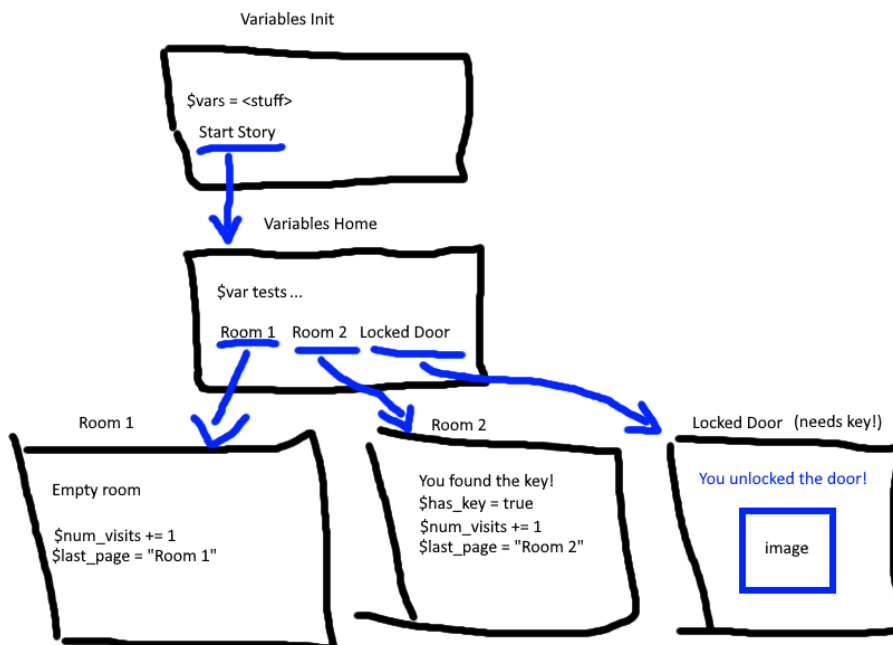
A Twine story with variables and an image	3
Step: Planning	3
Step: Create a new story	4
Step: Add Variables Init passage	4
Step: Add Variables Home passage	4
Step: Disabling the Twine Undo link (sidebar links)	5
Step: Add text to Room 1	5
Step: Add text to Room 2 (and the key)	6
Step: Test \$has_key in Variables Home	6
Step: Adding text and an image to the Locked Door passage	7
Image Gotchas	8
Adding More Variables	9
Step: Add \$last_page and \$num_visits to Variables Init	9
Step: Display \$last_page and \$num_visits on the Variables Home passage	9
Step: Add new variables to Room 1	9
Step: Add new variables to Room 2 (too)	10
Step: Testing \$num_visits and “else-if”	10
Step: Add the page variables to the Locked Door passage	11
Appendix: Full Story	12
Passage: Variables Init:	12
Passage: Variables Home:	12
Passage: Room 1	13
Passage: Room 2	13
Passage: Locked Door	13
Story Stylesheet	14

A Twine story with variables and an image

The title says it all. In this document, we will be creating a simple story that includes variables (coding/programming) and adding an image. You might be thinking that adding an image shouldn't be that big of a deal, and it isn't, but there is a big catch that you need to be aware of. If you are only interested in the image, skip to that section.

Step: Planning

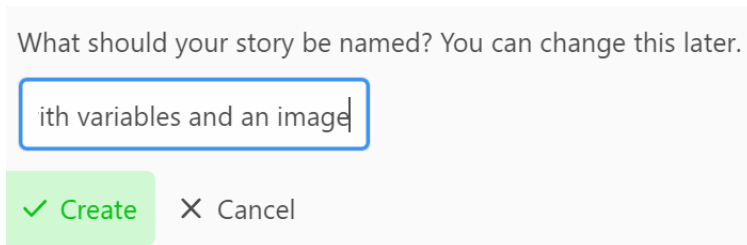
In this story, we will need to find the yellow key, which is located in Room 2. We will prevent access to the Locked Door room until the player finds the key (`$has_key`) and returns to the Story Home room. We will also keep track of the number of rooms that they entered (called `$num_visits`) and the last page that they visited (`$last_page`). An important element to this design is that we need a new starting page named Variable Init. This page is where we initialize and reset the values for the game variables. Typically, this is where your game starts or restarts to. It is on this page that we will also add our image.



Before we start adding variables, it should be noted that in Twine (Harlowe), variables start with a dollar sign like this `$var`. Also, be aware that the default Twine undo link that appears on the left side of the screen when you visit a page will undo any variable changes you make when clicked. As a result, it is suggested that you disable the undo button (see below) and replace it with links to the pages you need to return to.

Step: Create a new story

We'll start by creating a new story. I will name mine "A story with variables and an image" and then clicking the Create button.



What should your story be named? You can change this later.

✓ Create ✕ Cancel

Step: Add Variables Init passage

For this story, our starting page will be called "Variables Init" which is short for variable initialization. When creating variables, it is important to set their starting value explicitly. This is called initialization, or init for short, and hence the passage title "Variables Init".

Click on your first passage and rename it to Variable Init. Then add the following text:

```
Start the story... [[Variables Home]]
```

```
Initialize the variables.  
{  
    (set: $has_key = false)  
}
```

The story starts with some text that has been abbreviated for the sake of time. There is a link to our story's home page, "Variables Home". This is where the player will return to after visiting each page until they finish the story. Next, we add a section where we initialize our variables. Note that the title "Initialize the variables" is for our purposes and isn't required to actually initialize the variables. The curly braces surround our code block and the (set:) macro is used to set the value for the variable \$has_key to the value false.

Build and play your story and you should note that the text inside the curly braces is not visible.

Step: Add Variables Home passage

Now that we have our Variables Init page, we will add text to our story's home page, "Variables Home". Click on the Variables Home passage and add the following text:

```
Home page for variables.
```

```
Visit [[Room 1]]
```

```
Visit [[Room 2]]
```

```
[[Restart Game->Variables Init]]
```

Twine will create two more passages, Room 1 and Room 2. Also, note that we have linked back to the Variables Init passage but given the link an alias “Restart Game”. Note that there is no space between the arrow (“->”), the alias text, and the passage name. If you misspell the passage name, including adding a space, Twine will create a new passage or give you an error.

Step: Disabling the Twine Undo link (sidebar links)

By default, Twine will add a navigation bar on the left side of the story with two links: undo and redo. These can be useful, but they can also be harmful when using variables. As their names imply, they undo and redo the actions taken. When using variables, this means that any changes to the variables will be lost when the undo link is clicked, which is not what we want to have happen. As a result, we typically want to disable these when using variables and add explicit links to control the flow.

To disable the undo button in Twine, we hide the link by changing the Cascading Style Sheet (CSS) under Story > # Stylesheet. Type the following into the stylesheet page.

```
tw-sidebar {display: none;}
```

This is a CSS entry that says to set the display attribute for the element “tw-sidebar” (Twine’s name for its sidebar) to none. In other words, don’t display the tw-sidebar.

Step: Add text to Room 1

Now that we have Room 1, we need to add some text to it. In this story, we will add some placeholder text and a link back to our Variables Home passage. We will add some code for changing variables later.

```
Just an empty room.
```

```
Return to [[Variables Home]]
```

After making this change, test that Room 1 is working, and the link takes you back to the Variables Home passage.

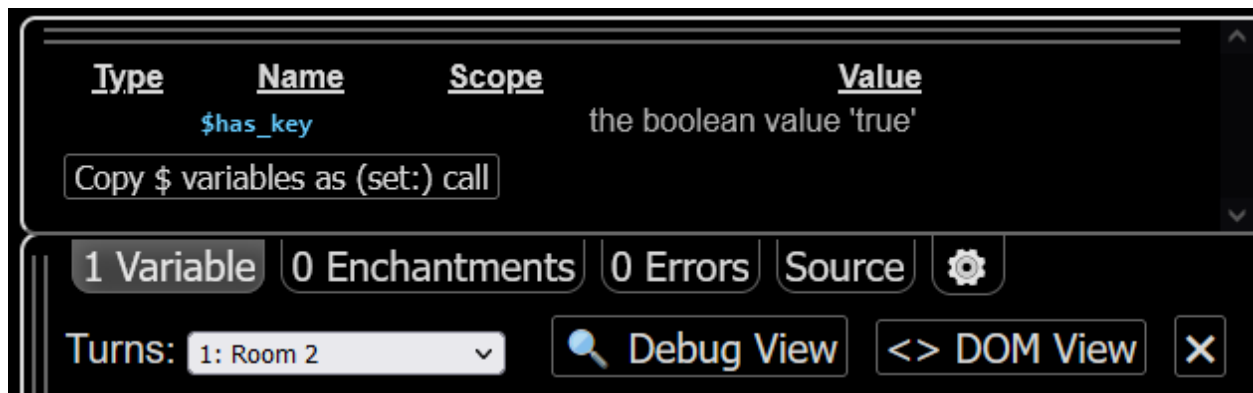
Step: Add text to Room 2 (and the key)

For Room 2, we will add placeholder text as well, but we will also change the value of our \$has_key variable from false to true. Add the following to the Room 2 passage:

```
You found the key!  
  
{  
  
    (set: $has_key = true)  
  
}  
  
Return to [[Variables Home]]
```

The location of the curly braces is not important on this page and is in the middle just because our focus is on teaching about variables. Therefore, they could be at the top or the bottom as well. After adding this text, test the page. Note that nothing should change, but nothing should break, i.e., you shouldn't have any errors.

It's worth noting that if you click the "Test From Here" for the Room 2 passage that the debug console will appear in the bottom right of the Twine browser window, like this:



Note that we are in Room 2 (Turns) and that the \$has_key variable has the Boolean value "true". This viewer will only appear during debugging/testing.

Step: Test \$has_key in Variables Home

Now that we have our Room 2 where our \$has_key variable will be set to true, we will add a test to our Variables Home page for this variable. To do that, let's add the following if-macro code to Variables Home.

```
The key question:  
  
(if: $has_key)[You have the key, open the [[Locked  
Door]]](else:)[[Look for the key!]]
```

The “The key question:” pun is not required but is always appreciated.

The if-else-macro is structured like this:

```
(if: <Boolean statement>) [<text when true>] (else:) [<text when false>]
```

Note that an (else:) macro is not required for an (if:), but in this case we want to change the text both when the player has the key and when they do not.

Now that we have our conditional text, note that Twine added our new “Locked door” passage, so let’s add text and an image (the treasure).

Step: Adding text and an image to the Locked Door passage

Double click on the new Locked Door passage and add the following text:

```
Using the key, you unlock the door and ... find the Twine logo!
```

```
Twine 
```

```
Return to [[Variables Home]]
```

```
[[Restart Game->Variables Init]]
```

On this page, we will finally add an image. In this case, we will add the Twine logo image. The logo is found at this URL:

<https://twinery.org/icons/twine.svg>

I found this image by visiting the Twine home page (<https://twinery.org>) and the right-clicking on their logo and choosing “Open image in new tab”. This then showed me the image and also gave me the full URL. You need a full URL for images.

Images in Twine are added using the HTML tag which has a “src” (source) attribute that is set to the URL for the image and optional tags for width, height, and title.

Once you have this text, test from this page to verify that the image displayed as expected.

Using the key, you unlock the door and ... find the Twine logo!



Return to Variables Home

Restart Game

Image Gotchas

The gotcha with images is that you can reference local images on a computer but when you publish your story, Twine will not include the images. Instead, you must manually create a zip file that contains both the .html file and the images (or image folders) yourself. This is certainly doable but is not a single click and is prone to mistakes. As a result, I suggest you start with publicly available images. Of note, you can also self-publish images using Dropbox, OneDrive, or Google Drive and then put a link to that image in your story.

If you came here just for the image, then we're done. The rest of the document adds more variables, one numeric and one string.

Adding More Variables

We've already seen how to add a Boolean (`$has_key`) variable, set it, and test it as well. Next, we will add two more variables, `$last_page` and `$num_visits`. The `$last_page` variable will be set by the room passages and displayed on the Variables Home page. The `$num_visits` variable will be a numeric (integer) that is incremented each time we visit a room, and the total number of visits will be shown on the Variables Home page.

Step: Add `$last_page` and `$num_visits` to Variables Init

To start, we will initialize our two new variables on our Variables Init passage. Open that passage and add the following to the variables section of the code.

```
(set: $num_visits = 0)

(set: $last_page = "Variables Init")
```

Note that the `$last_page` variable is a string (text) and thus its assigned value must be enclosed in matching double quotation marks, as shown above.

I would suggest testing your Variables Init page to ensure that adding the new variables didn't introduce any errors (it shouldn't).

Step: Display `$last_page` and `$num_visits` on the Variables Home passage

Now that we have our two new variables, we will display them on our Variables Home passage. To do that, add the following:

```
You have visited the rooms $num_visits times.

The last page you visited was $last_page.
```

Notice that the variables are just embedded as-is within our text. Twine will insert the variables into the text automatically. To verify, play your story.

Step: Add new variables to Room 1

Now that we have our variables displaying on the Variables Home passage, we will add code to Room 1 (and Room 2 later) to change these variables. Open up Room 1's passage and add the following:

```
Increase the page count and set the last visited page name.

{

(set: $num_visits to it + 1)
```

```
(set: $last_page to "Room 1")
}
```

Again, the sentence describing what we are doing is not required but is included for reference later. On this page, we see the (set:) macro as well as the “to” assignment operator and the keyword “it”. The “it” keyword refers to whatever variable you are referencing. You can put the explicit variable name instead or you can be lazy and let Twine figure it out by using “it”. The “to” assignment operator is the same as “=”.

Once you’ve made these changes, run your game, visit Room 1, and then return to Variables Home to see if the variables changed.

Step: Add new variables to Room 2 (too)

Add the same text to Room 2, but make sure that \$last_page is set to “Room 2”.

```
Increase the page count and set the last visited page name.
{
  (set: $num_visits to it + 1)
  (set: $last_page to "Room 2")
}
```

Again, run your game and visit Room 2 to verify that the \$num_visits and \$last_page variables work as expected.

Step: Testing \$num_visits and “else-if”

Now that we have our page variables working, we will add one more test to our Variables Home passage to demonstrate testing the \$num_visits numeric variable using the if, else-if, and else macros. Add the following code to the Variables Home passage:

```
Num visits testing:
  (if: $num_visits > 3)[Three is enough](else-if: $num_visits <
  1)[Try a room](else:)[Keep trying]
```

This (if:) macro will show different text based on the number of room visits. First when the number of visits is more than 3, it will print “Three is enough”. However, if the visits are less than 1, it will prompt the player to try a room. Finally, if the number is between 1 and 4, then it will tell them to Keep trying.

Run your game and verify that this is working as expected.

Step: Add the page variables to the Locked Door passage

For completeness, we will add the page variables to the Locked Door passage as well. Add the following:

```
{  
  (set: $num_visits to it + 1)  
  (set: $last_page to "Locked Door")  
}
```

As always, test your game to make sure that the Locked Door page works as expected.

We are done. We have added three types of variables, changed them, tested them, and embedded them into our game.

Appendix: Full Story

The following are the passages (and stylesheet) for the entire project.

Passage: Variables Init:

```
Start the story... [[Variables Home]]  
Initialize the variables.  
{  
  (set: $has_key = false)  
  (set: $num_visits = 0)  
  (set: $last_page = "Variables Init")  
}
```

Passage: Variables Home:

```
Home page for variables.  
  
The key question:  
(if: $has_key)[You have the key, open the [[Locked  
Door]]](else:)[Look for the key!]  
  
You have visited the rooms $num_visits times.  
The last page you visited was $last_page.  
  
Num visits testing:  
(if: $num_visits > 3)[Three is enough](else-if: $num_visits < 1)[Try  
a room](else:)[Keep trying]  
  
Visit [[Room 1]]  
Visit [[Room 2]]  
  
[[Restart Game->Variables Init]]
```

Passage: Room 1

Just an empty room.

Increase the page count and set the last visited page name.

```
{  
(set: $num_visits to it + 1)  
(set: $last_page to "Room 1")  
}
```

Return to [\[\[Variables Home\]\]](#)

Passage: Room 2

You found the key!

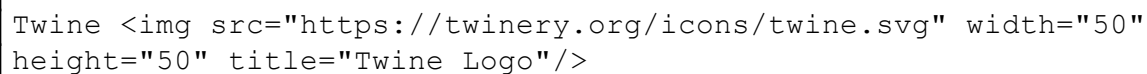
Increase the page count and set the last visited page name.

```
{  
(set: $has_key = true)  
(set: $num_visits to it + 1)  
(set: $last_page to "Room 2")  
}
```

Return to [\[\[Variables Home\]\]](#)

Passage: Locked Door

Using the key, you unlock the door and ... find the Twine logo!

Twine  Twine

```
{  
(set: $num_visits to it + 1)  
(set: $last_page to "Locked Door")  
}
```

Return to [\[\[Variables Home\]\]](#)

[\[\[Restart Game->Variables Init\]\]](#)

Story Stylesheet

```
tw-sidebar {display: none;}
```