Snap Tips

*Advice on teaching CS and shortcuts when working with the* [*Snap!*](http://snap.berkeley.edu/run) *programming language.*

*This document includes tips from TEALS teachers and the Beauty & Joy (BJC) team. Teachers can feel free to add comments and suggest edits to the BJC provided tips here:* [*bitly.com/snap-advice*](http://bitly.com/snap-advice)*.*

Table of Contents

[TEALS Recommendations 2](#_Toc522196472)

[Tip 0: TEALS Snap! & CS Teaching Tips 2](#_Toc522196473)

[Beauty & Joy of Computing Provided Shortcuts 4](#_Toc522196474)

[Tip 1: Transparent Costume Backgrounds 4](#_Toc522196475)

[Tip 2: Moving blocks of code 6](#_Toc522196476)

[Tip 3: Copy code or costumes between sprites 7](#_Toc522196477)

[Tip 4: Beware the “Stamp” Block! 7](#_Toc522196478)

[Tip 5: Searching for blocks by name 7](#_Toc522196479)

[Tip 6: Finding a sprite on the stage 8](#_Toc522196480)

[Tip 7: Naming sprites, costumes, variables, custom blocks 8](#_Toc522196481)

[Tip 8: Changing Stage Size 8](#_Toc522196482)

[Tip 9: Sprite Rotation 8](#_Toc522196483)

[Tip 10: Sprite Centering 9](#_Toc522196484)

[Tip 11: Sprite Draggability 10](#_Toc522196485)

[Tip 12: Costumes from BYOB 11](#_Toc522196486)

[Tip 13: Costumes for Animation 11](#_Toc522196487)

[Tip 14: Comments 11](#_Toc522196488)

[Tip 15: Touching Color vs Touching Sprite 11](#_Toc522196489)

[Tip 16: Debugging 11](#_Toc522196490)

[Tip 17: Initialization 12](#_Toc522196491)

[Tip 18: Reading from external websites 12](#_Toc522196492)

[Tip 19: Importing lots of text into a list 12](#_Toc522196493)

[Tip 20: The Javascript Block 13](#_Toc522196494)

[Tip 21: Snap Server Mirrors 14](#_Toc522196495)

[Tip 22: Thread Safe Scripts 14](#_Toc522196496)

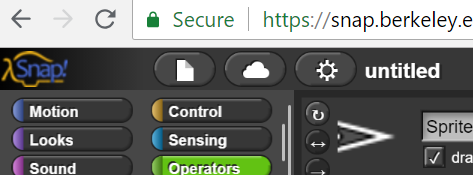
[Tip 23: Script pics 15](#_Toc522196497)

# TEALS Recommendations

### Tip 0: TEALS Snap! & CS Teaching Tips

*These recommendations were provided by TEALS teachers.*

**TEALS Snap! Tips**

* **Local copy of Snap** in case of internet loss
* **Clear** button (in Pen) to clear Stage
* **Save to Cloud** after logging in
* Click and drag blocks to the left (onto Palette Area) to **get blocks off the Scripting Area**
* Can have **students login** name be firstInitial.LastName.birthmonthbirthdate (ex J.Smith.0806); Password: OSIS # --> That way teacher always has student login info and students can easily remember
* Can set **Background Costume to show coordinate plane** (File -> Background -> scroll to and select coordinate plane)
* May want students to **show x/y position** by checking those boxes on the Palette so they show on the Stage as Watchers
* Use **Stepper Tool** to trace the code as the program runs -> great debugging strategy; can slow down the steps
* **Save often** - Snap doesn't auto-save
* **Wait block** can be helpful for debugging
* Right click in the Scripting Area to **"clean up" code blocks** so they are all in one line
* Right click in the Scripting area to **add comments** to code (helps you and students know/remember what code block is supposed to do)
* Add **custom blocks** using the Make a Block at the bottom of each palette
  + To add variable parameters in a custom block - use the %variable name% or use + to add one
* Hold Shift while drawing shapes in the costume editor to **get perfect shapes** (circles, squares, etc)
* Use **crosshair tool** (circle with + through it) in costume editor to **define the center** of a sprite
* Have Sprites **face only left/right** by clicking the 2nd circle on the top under the file (**platformers**; the one with the r/l arrow inside) then they won’t rotate/flip upside down
* When sharing files, you can **ensure students have shared their actual file and not a generic Snap link** by setting up validation on your Learning Management System so the link has to be a certain length.
* Differentiate by creating a **shell/skeleton project** that includes the necessary blocks (Parson's problem)
* Have students **name variables in Snap using Python syntax**
* Explain **Broadcast** function as "like a radio"
* **Change Snap! Language** using the tools bar (to the right of the cloud) and going to the student’s home language.
  + 
* **Anchor vocab to images** (i.e. traversal and traversing a mountain one step at a time)



* Use **pen down to help students see math** for sprite movement
* **Yellkey.com (similar to tinyurl)**

**TEALS CS Teaching Tips**

* **Share final unit projects at the start of the unit** to help students know where they are going and what their lab work builds up to (can also share throughout the unit)
* Differentiate by creating a **shell/skeleton project** that includes the necessary blocks (Parson's problem)
* Have students **name variables in Snap using Python syntax**
* **Examples of Do Now Routines**
  + Each student’s name on a grid
    - Students are timed and must complete the do now and put the post-it note on the chart before the timer goes off. Students are graded based on complete/incomplete. Students who do not post, are late to class, or are absent receive no credit.
  + See, Think, Wonder
    - Use the final project sample as a hook into the beginning of the unit. Try to tie students observations to the upcoming lesson
* 1) Ask students, "What do you see?" *Prompt students to only make observations based on what is plainly apparent, for example an inventory of what items are present, where they are, or a visible change that occurs (such as a costume change)*

# Beauty & Joy of Computing Snap! Shortcuts

*These recommendations were provided by the BJC team.*

### Tip 1: Transparent Costume Backgrounds

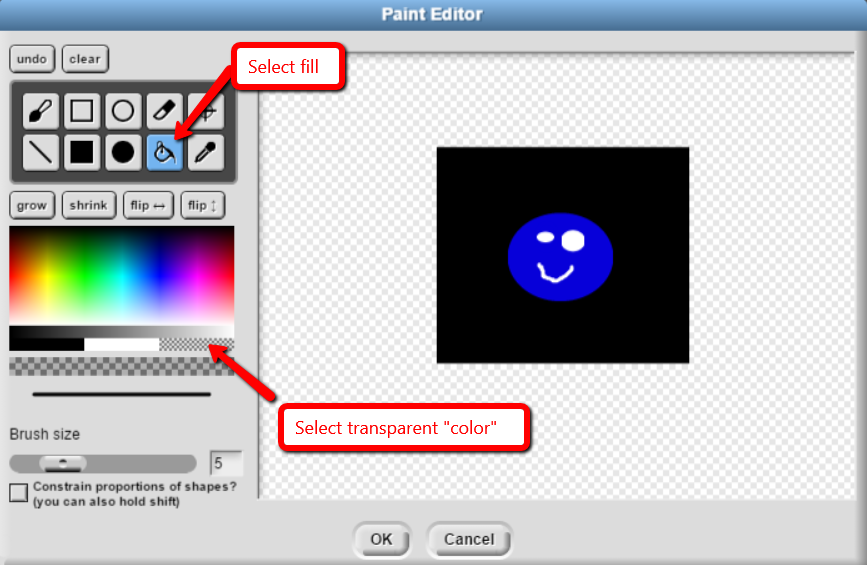
If programmers try to mash-up images they found on the web (with appropriate copyrights of course!) they are probably frustrated when they find the perfect image... and it has a white background which looks ugly in SNAP. There are few options for removing the background.

Warning: If you experience weird behavior with the touching blocks, it could be because your sprite has a non-transparent background!

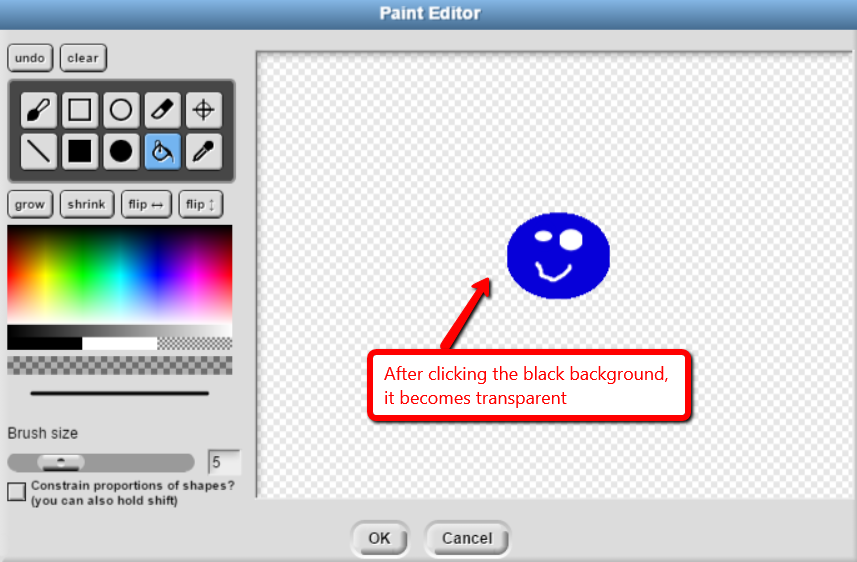
#### A. Using the Snap Image Editor

If the background of an image is a uniform color, and you want to get rid of it all at once, set the pen to the checkerboard pattern, then use the “fill tool” to change the whole background to “empty” with one click. Here are step-by-step directions:

1) Once you import your image, right click on the costume to edit it.

2) From the edit, you can select "fill" and select the somewhat hard-to-spot transparent "color".

3) Click on the background color that you want to make transparent.



You can also use the brush tool with color set to “transparent” to "erase" the background, brushing around the image.

#### B. Use another image editing tool, such as pixlr

[http://www.pixlr.com](http://www.pixlr.com/) has free image editing tools that work in the browser and can remove backgrounds. Users will need to:

-import the image they found to Pixlr

-remove the background

-save as a new file to their computer

-import the file into SNAP

#### C. Search for an image with no background

Google Image search has a filter under "search tools" called "color". If you select "Transparent" as the color, you will only find image results with transparent backgrounds! No editing needed.

### Tip 2: Moving blocks of code

A lot of your time programming in Snap will involve arranging and rearranging blocks, which can be a bit difficult, especially on a laptop where you might not have as fine control over the pointer. Snap tries to make it easier on you by giving you a few shortcuts.

* If you drag a block between two other blocks, it will be inserted between those blocks.
* If you drag a block to the left side and drop it there, it will be removed from the scripting area
* Try to keep the scripting area neat and organized so you can see all the scripts. If you aren’t careful, it’s easy to have a script hidden off screen, which can lead to confusing bugs. You can right-click clean up can also help with this, but it organizes all your blocks of code into one column, which may or may not be the user's preference.

The menus that you get when you right-click on various things in Snap are very useful.

* right-click undo-drop -- This option is not always present, but sometimes after you place a block, if you put it in the wrong place, you can use this option to undo it.
* right-click delete -- very useful if you need to remove a block from between two other blocks
* right-click relabel -- quickly switch between similar blocks.
* right-click duplicate -- duplicates the block that you clicked on, and all the blocks after it in the same script.

### Tip 3: Copy code or costumes between sprites

* You can click and drag blocks (including scripts, or groups of blocks) from the Scripting Area of one sprite onto a different sprite in the Sprite Corral (on the bottom right). This will essentially "copy and paste" those blocks onto that other sprite.
* You can also drag costumes to other sprites in the Sprite Corral to copy the costumes to that sprite.

### Tip 4: Beware the “Stamp” Block!

The “Stamp” block, under the “pen” menu, causes the sprite’s image to appear at that spot on the stage until the stage is cleared. This can be very confusing if you do it by accident or aren’t familiar with the block -- it can look like your sprite is stuck in that spot, or that your sprite can’t change its costume!

### Tip 5: Searching for blocks by name

Press Control-f, then start typing to search for a block by name

If you've ticked the Keyboard Editing option on the Options menu (the 'gear' icon to the right of the cloud) then you can enter blocks using the keyboard by first shift-clicking anywhere in the Scripting Area, then when the horizontal flashing cursor appears start typing.

Just as with Ctrl/F you get an input box into which you can type. Now however as you type the block whose 'name' matches what you've typed appears with a white outline (if the highlighted block isn't the one you want press cursor down to select another). Press enter and the block is inserted into the Scripting Area.

If the block you've just inserted has slots then the Scripting Area highlight moves to the first where you can start typing again (this is very useful with the infix-expression-to-reporter parser discussed below).

You can move the highlight using the cursor keys (left and right move you 'within' the current block, up and down between blocks). Experiment and you'll get a good feel for how it works.

Version 4.0.10 onwards has the infix-expression-to-reporter parser which means that, for example, if you want the equivalent of

if a < b

shift-click, type if, press enter, type a<b and press enter.

This is very quick once you get used to it.

### Tip 6: Finding a sprite on the stage

If your sprite disappears from the screen (such as if you make it go off the right end), you can get them back by right clicking on the sprite icon on the bottom right and clicking "Show"

### Tip 7: Naming sprites, costumes, variables, custom blocks

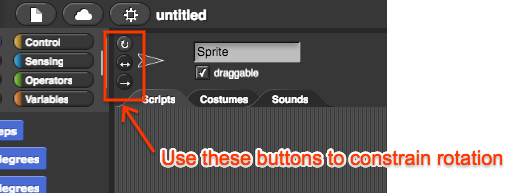
Using meaningful names for sprites, costumes, variables, and custom blocks can greatly improve the readability of your code and help avoid bugs!

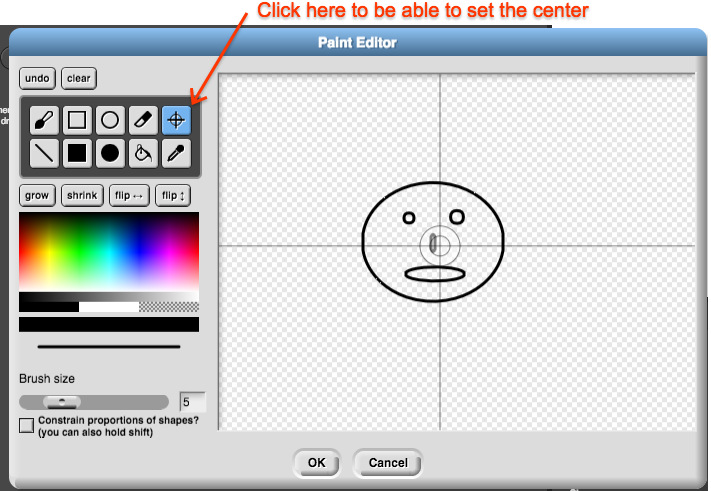
* You can rename sprites (textbox at the top, right above "Draggable" checkbox) which makes it easier to locate the proper sprite under blocks like "Point towards [ ]".
* You can rename costumes, which makes it easier to find them under "Switch to costume [ ]". To rename, right-click on the specific costume under the "costumes" tab.
* Use a naming convention to differentiate different types of variables. For example, you can use ALL CAPS for variables which are “for all sprites”, Capitalize The First Letter Of Each Word for variables which are “for this sprite only” and use lowercase for script variables.

### Tip 8: Changing Stage Size

You can change the dimensions of the scene from the usual 480x360 via the "gear" button (next to "file" and "cloud" buttons), under "Stage Size...".

### Tip 9: Sprite Rotation

Sometimes, you want to constrain the motion of a sprite so that it only faces left or right when rotated, rather than flipping upside down. Snap has a special Sprite property for the mode of rotation. You'll see 3 small circular buttons to the left of the input where you can set a sprite's name. These correspond to free rotation, only face left/right, and no rotation. 



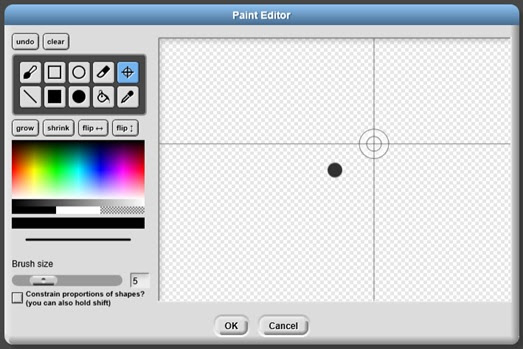
### Tip 10: Sprite Centering

SNAP position sprites based on an "anchor point." Especially when images are imported from the internet, the anchor point may not be in a spot that makes sense. Programmers can customize the anchor point in SNAP's built in image editor. The icon looks like a "target" and will allow them to click on the point within their costume that SNAP should consider the anchor.

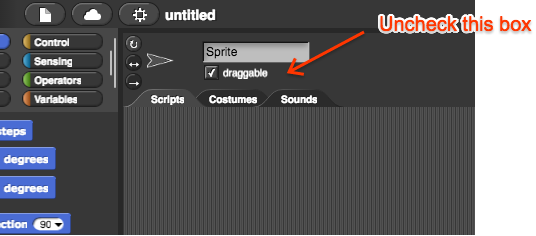
For the default “Turtle” sprite, the center is usually the tip, but you can right click on it to change it to the middle.



Warning: If you experience problems moving your sprite to the right part of the stage, i.e., you tell it to go to a particular (x, y) coordinate, and it seems to go somewhere else, it could be that the costume’s center is not where you think it is. For example, the ball below has a center which is far away, which can be the source of hard to find bugs.



### Tip 11: Sprite Draggability

By default, sprites are marked draggable, which means that both the programmer and the user can click and drag the sprite. Sometimes you want to allow the user to drag the sprite, but often after you have decided where the sprite should be, you want to uncheck the draggable box so that users can’t move it, especially for sprites that represent buttons -- you don’t want users to accidentally move the sprite when they click on it. 

### Tip 12: Costumes from BYOB

Snap comes with several built-in costumes. BYOB, the old, offline version of Snap, had even more. The BYOB costumes can be found at [bitly.com/snap-images](http://bitly.com/snap-images).

### Tip 13: Costumes for Animation

Several of SNAP's built-in costumes represent multiple frames of motion for a continuous animation (dog walking, bat flying, etc). Students will often want this kind of animation with other characters. If the student can find a GIF of the character they want to use, there are some free online tools that will split the GIF into frames that can then be used as costumes:

<http://ezgif.com/split/>

<http://animizer.net/en/gif-apng-splitter>

### Tip 14: Comments

Adding comments to your code is an important way to make sure code readable, which helps avoid bugs. To add comments, right-click on the background of the Scripting Area (not on any block) and choose “add comment”. That will bring up a bubble where you can type your comment. Then you can move the comment around next to the relevant code. If you drag the comment to a block, the comment will attach itself to that block. If you attach a comment to the head (the first block) of a custom block, then when your pointer hovers over the custom block in the palette, that comment will appear to tell users what your block does.

### Tip 15: Touching Color vs Touching Sprite

The touching [color] is quite slow. Try to avoid it if at all possible. The touching [sprite] block performs much better, so it is a reasonable alternative.

### Tip 16: Debugging

It can be a challenge to debug Snap code, especially code involving clones.

* One basic method, which is still quite effective, is to add “say” blocks at various points in your code.
* Check the checkboxes next to variables to display their value on the stage.
* Build up scripts one block at a time to see which block is causing problems.

### Tip 17: Initialization

It’s a good idea to get into the habit that when the user starts your program (usually by clicking on the green flag), the first thing you do is to initialize all the sprites, including their costumes and positions, to wherever you want them to be at the beginning of the program. It may be tempting to skip that initialization, because when you save a Snap program, it remembers many things for you, including the values of the variables and the positions of the sprites. So, you may think that you can just position the sprites by hand, and save your project that way. But explicitly initializing everything with code is a better practice.

* It makes it easier for you to restart the program later (by clicking the green flag again, or with a restart button, etc)
* It is more robust, in case you accidentally save the program in a different state.
* It makes debugging easier because when you debug you often make a small change, then rerun the script or the program. If you’ve initialized things explicitly you know that every time you rerun the script, it will start from the same place and the only difference will be the change that you made.
* It is more similar to what you have to do with other programming languages.

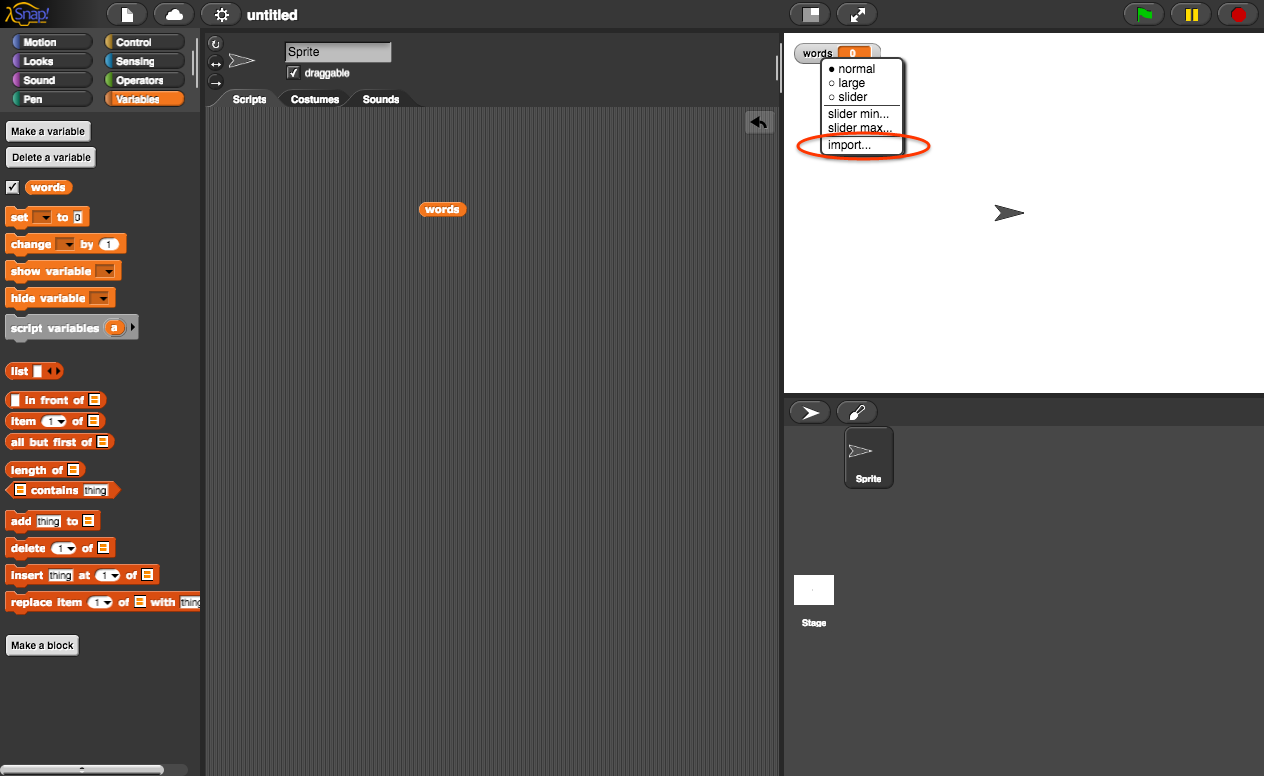
### Tip 18: Reading from external websites

Snap has an “http://” block for reading websites, but it only works for some sites (those that don’t have Cross-Origin-Resource-Sharing, or CORS, enabled). To get around this, you can try to use a “CORS proxy”, which redirects data from other sites. Try <https://crossorigin.me/> or <http://cors.io/>.

### Tip 19: Importing lots of text into a list

This is needed for the hangman project. Here are some ways to do it:

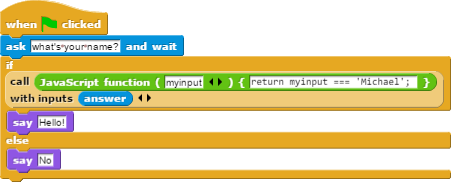
* Copy the text into one long string, paste it into a split block
* Import it into a list variable:
  + Create a variable and assign a blank list.
  + Make sure the variable is visible on the canvas.
  + Right click on the list's image on the canvas to find the "import..." command (this part was really elusive, it's not clear that the import command only appears on the canvas)
  + If the file has one word per line, you are set. Otherwise, after importing the text file, you have to use the "sentence -> list" command to split the words into items.



* Upload the list to github, which allows cross-origin requests, then use the http:// block to read the file.

### Tip 20: The Javascript Block

The javascript block creates a function, so in order to use it, you have to “call” the function by dragging the block into the call block. When you do that, the function will get a gray “ring”. You’ll have to right click on the function to “unringify” it. To pass arguments to the function, click the black arrow on the right end of the “call” block.



### Tip 21: Snap Server Mirrors

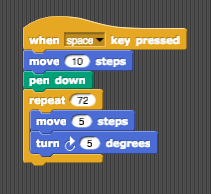
If there are problems with the Snap server, try these mirrors:

<http://bjc.edc.org/snapsource/snap.html>

<https://snap.apps.miosoft.com/>

### Tip 22: Thread Safe Scripts

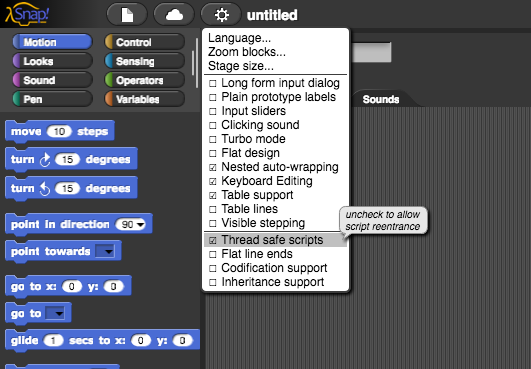
Consider the script:



This script moves the sprite, then draws a circle each time the space key is pressed. However, if you press the space key again before snap has finished drawing the circle, it will stop drawing the first circle and start drawing a second circle, starting wherever the sprite happens to be.

From this example, we can see that Snap scripts start executing when they detect a triggering event. If the triggering event happens again before the script has finished, the script will stop in the middle of whatever it was doing and start handling the new event.

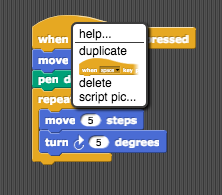
You can change this behavior for a particular Snap project by going to the menu that looks like a gear and checking the box next to “Thread safe scripts”. When that is checked, new events are ignored until the old event has completed.



Going back to the example above, once this box is checked, if you press space again before the first circle has finished, nothing will happen, the second key press will just be ignored.

### Tip 23: Script pics

If you right click on a script and choose the option “script pic”, it will open a new tab of your browser with a picture of that script. This is especially useful for teachers who are creating presentations.



This is currently broken in Chrome, but will be fixed soon: <https://github.com/jmoenig/Snap--Build-Your-Own-Blocks/issues/1840>