Eckart Modrow
emodrow@informatik.uni-goettingen.de

SQLsnap-manual

(2014-03-04)

SQLsnap is a little modification of Jens Moenigs great *snap!*-beta-sourcecode from December 2013. I added some extensions to fit *snap!* a bit better to the German CS-curriculum for schools. SQLsnap is meant as a toolbox, not as an interface to mySQL. If you need this, better use phpMyAdmin or other tools. SQLsnap has some new features to expand the creativity of snap! by subjects like databases, SQL and image processing. The combination of databases and image-processing should open new chances to let students build applications with topics of social relevance.

SQLsnap can be found on http://snapextensions.uni-goettingen.de.

User “snapexuser” (read-only) with password “snap!user” is listed on the connection-block.

1. New features

There are new blocks …

- to change the size of the workspace. This seems necessary to display larger lists or images (variables-palette).

- to get and set RGB-values of a point, and to convert RGB to HSV (sensing-palette).

- to connect to a database-server, read databases, tables and columns and to execute a query.

- to create variables with table- and column-names to avoid mistakes, and to delete them.

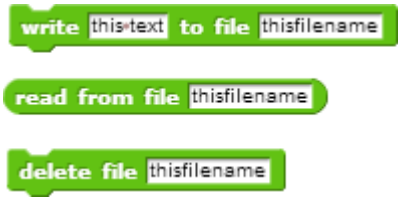
- to build predicates for the queries.

- to uses aggregation-functions.

- to create SQL-queries.

With this toolkit most SQL-queries can be constructed.

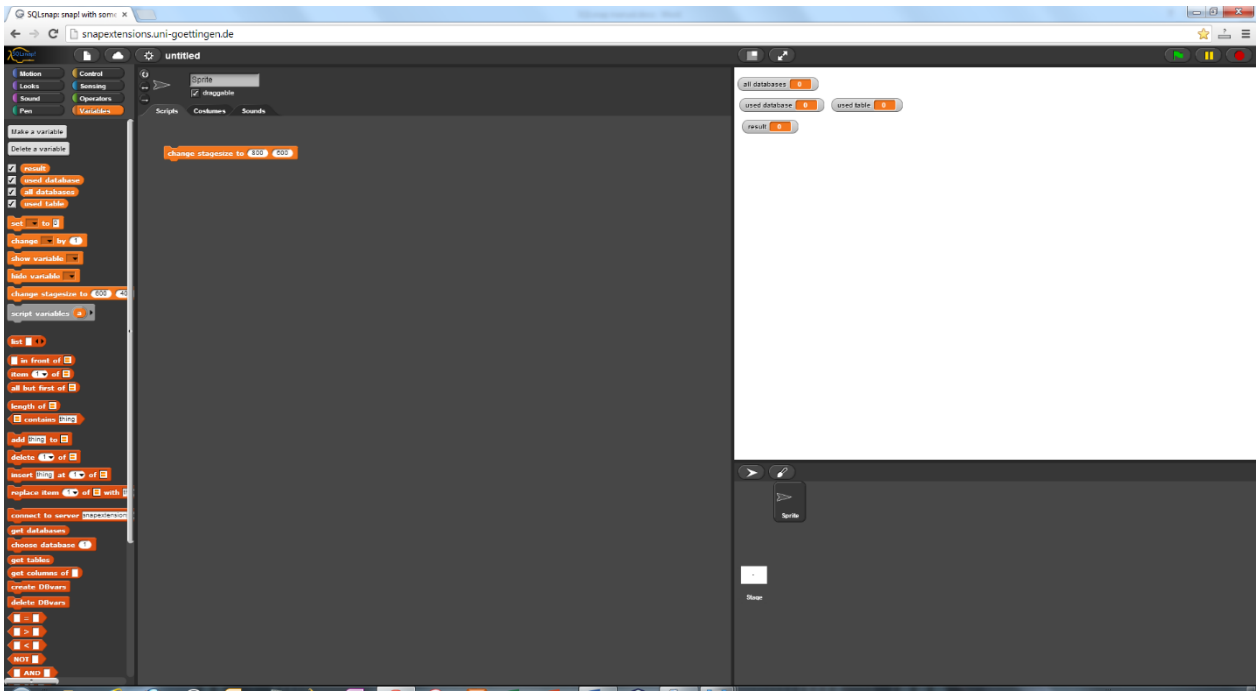
- to use text files on the server, available from any computer (limited to one string at time).



- to execute JavaScript-code generated by code-mapping.

2. Using a database

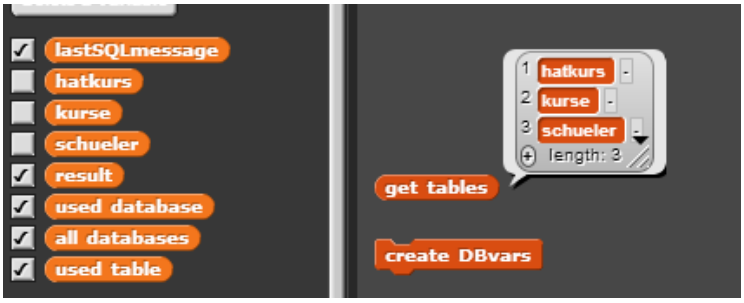
We change the stage size to 800x600 and create the variables “result”, “all databases”, “used database” and “used table”.



Then we choose a database …

… and get a different stage.

We read the tables and create appropriate database-(DB)-variables. Because till now only the tables of the actual database are known, we get three DBvariables “hatkurs”, “kurse” and “schueler”. DBvariables have their name as value, so we can use them to construct queries.



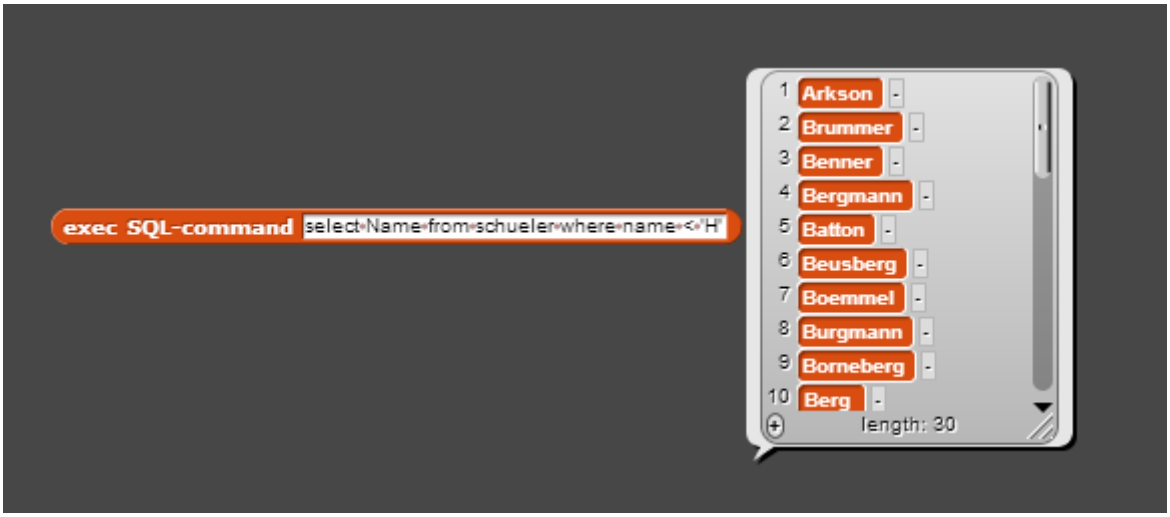
In addition the “create DBvars”-block has created a DBvariable “lastSQLmessage”. This variable is always updated with the last internal SQL-message.

If we want more information about the tables, we use the “get columns of”-block with the wanted tablename (a DBvariable) and get the columns of this table. They are stored internally, so we can create the appropriate DBvariables by using the “create DBvars”-block again.

If we need the DBvars of other tables, we read the columns and create the next variable-crowd.

3. SQL-queries

The simplest way is to use the “exec SQL-command”-block directly: type a SQL-string and click.

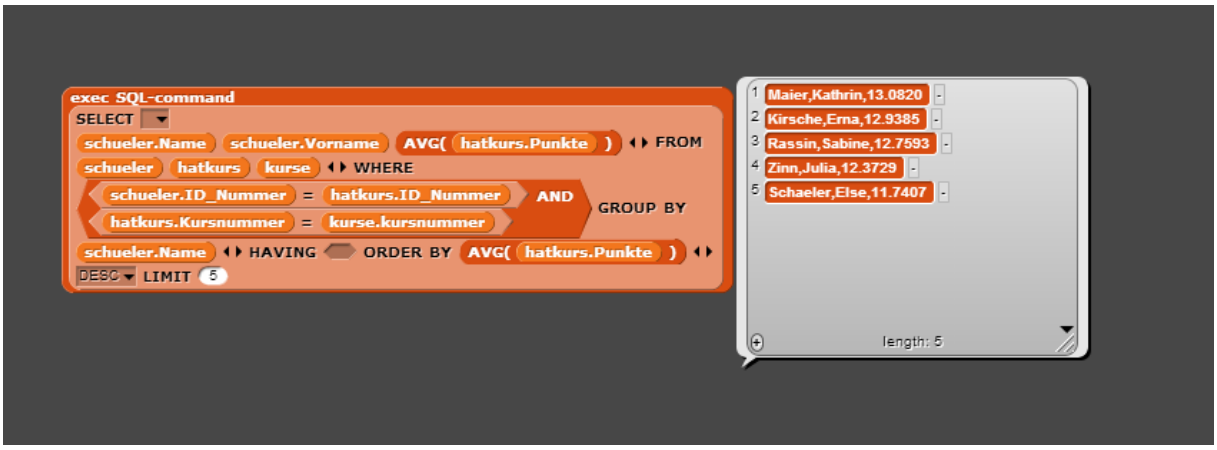


The strings for a SQL-query can also be constructed using SQL-blocks. There are two versions:

The first one “simple SQL” creates strings with the wanted query.

If we put it in the slot of the “exec SQL-command”-block, the query is executed.

The second version allows constructing almost complete SQL-queries.



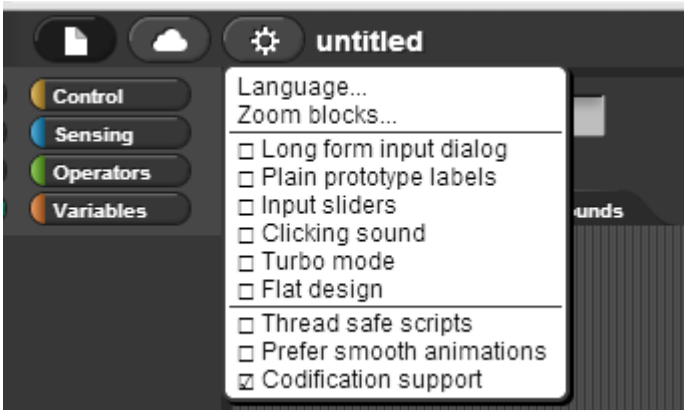
4. Manipulation of Pixels

You can read the RGB-value of a point. getRGB reads the RBG-value of the stage- or pentrails-point at (x|y).

You can write a RGB-value to a point (x|y) either on stage or on pentrails-image. On pentrails you can clear the painted pixels with “clear” from the pen-palette.

If necessary you can convert RGB-values to HSV.

5. JavaScript with Code-mapping



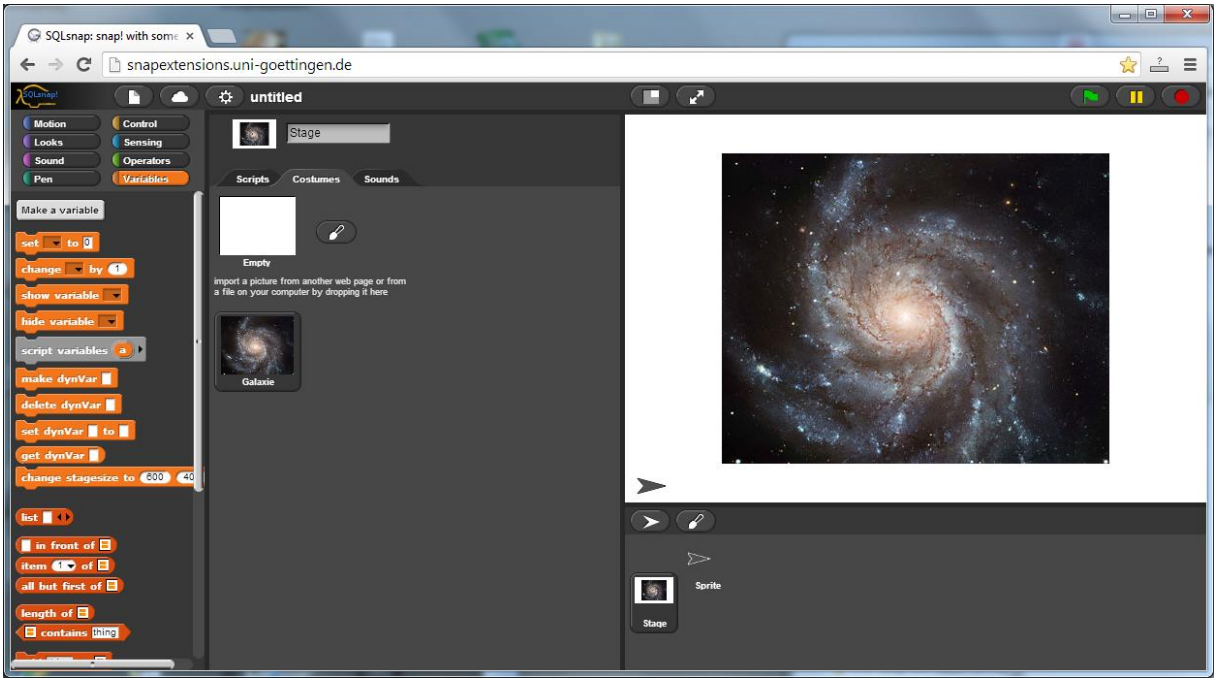
Pixel-manipulation on this way is rather slowly. So in SQLsnap codemapping-on is the default status (see codification-support).

Most command-blocks and variable/list-blocks can be mapped to JavaScript-code as well as the pixel-blocks and operator-blocks. The code can be produced with the “code-of”-block (variables-palette) and can be executed with “execute JScode of” (variables-palette).

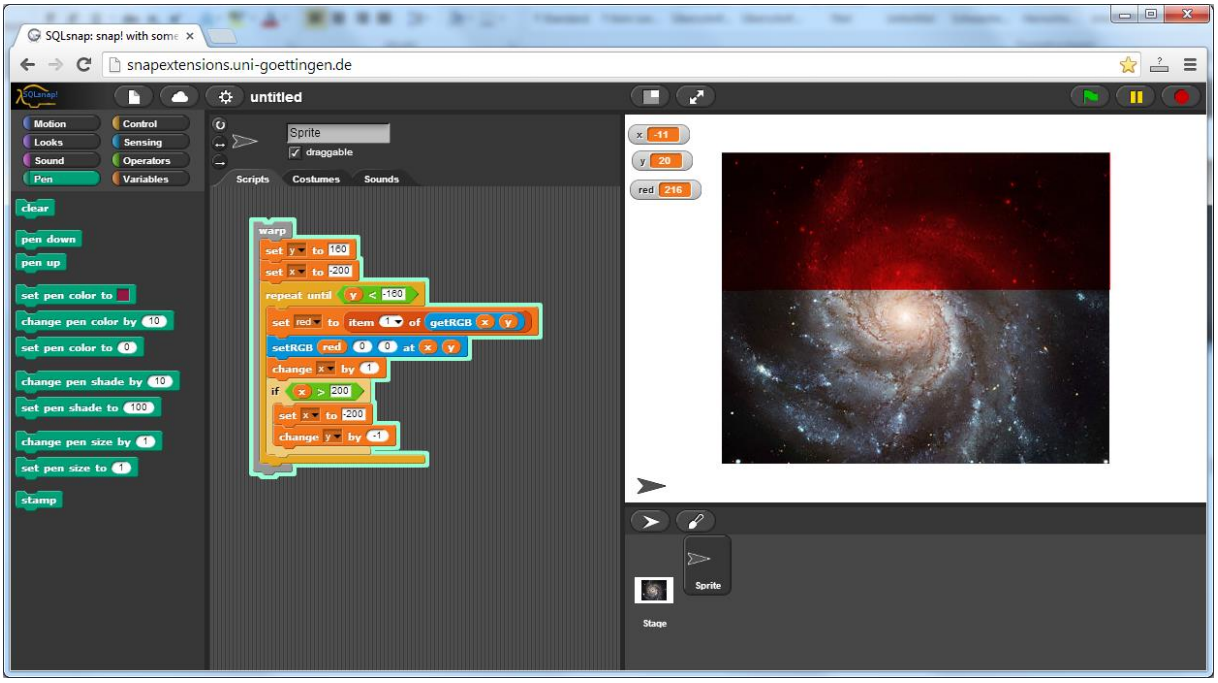
Because on this way unstoppable infinite-loops (e.g.) are possible, you should first test the scripts without code-mapping. Afterwards you can execute them directly with JavaScript.

Example:

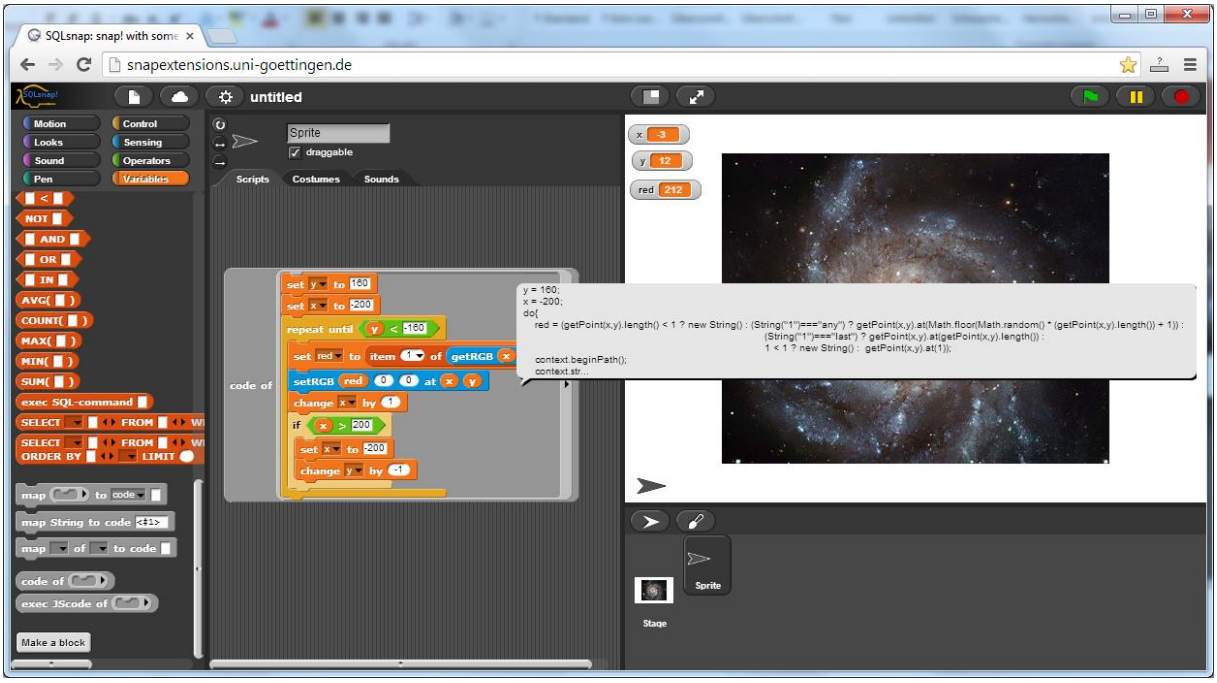
1. Load an image as new stage-costume. (here: a galaxy)



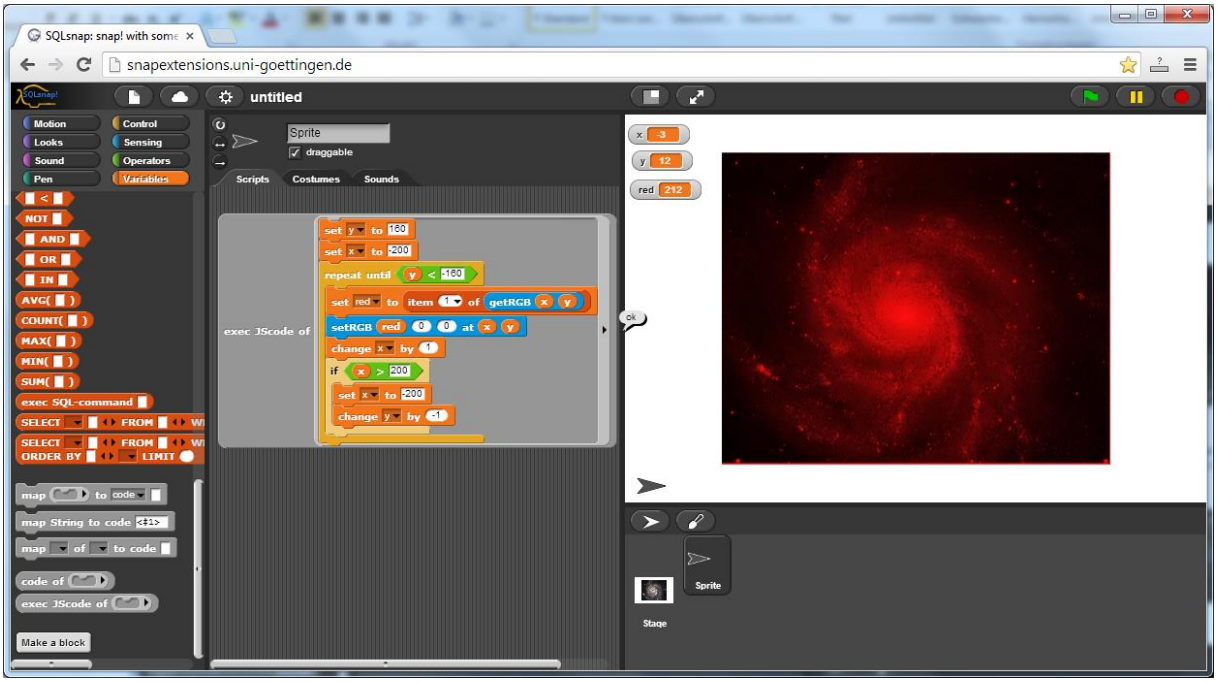
2. Write a script to transform the image to its RED-values. (These are the regions with old stars.) Test it in Snap.



3. Produce the appropriate JavaScript-code by code-mapping. Have a look on it.

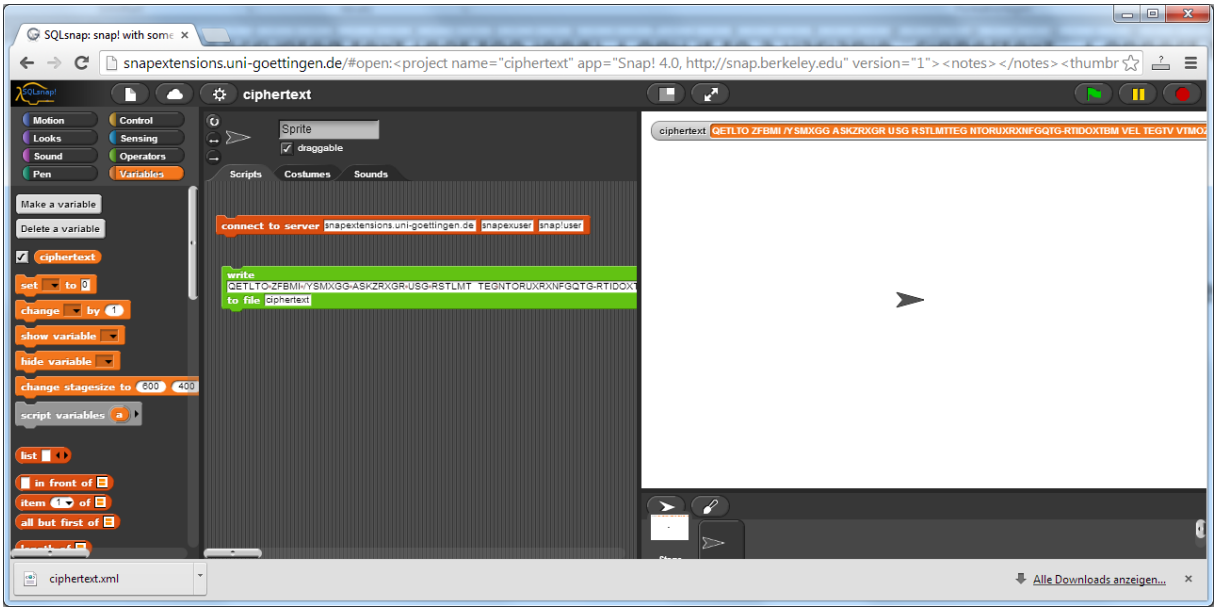


4. Execute JavaScript-code with the new Block.

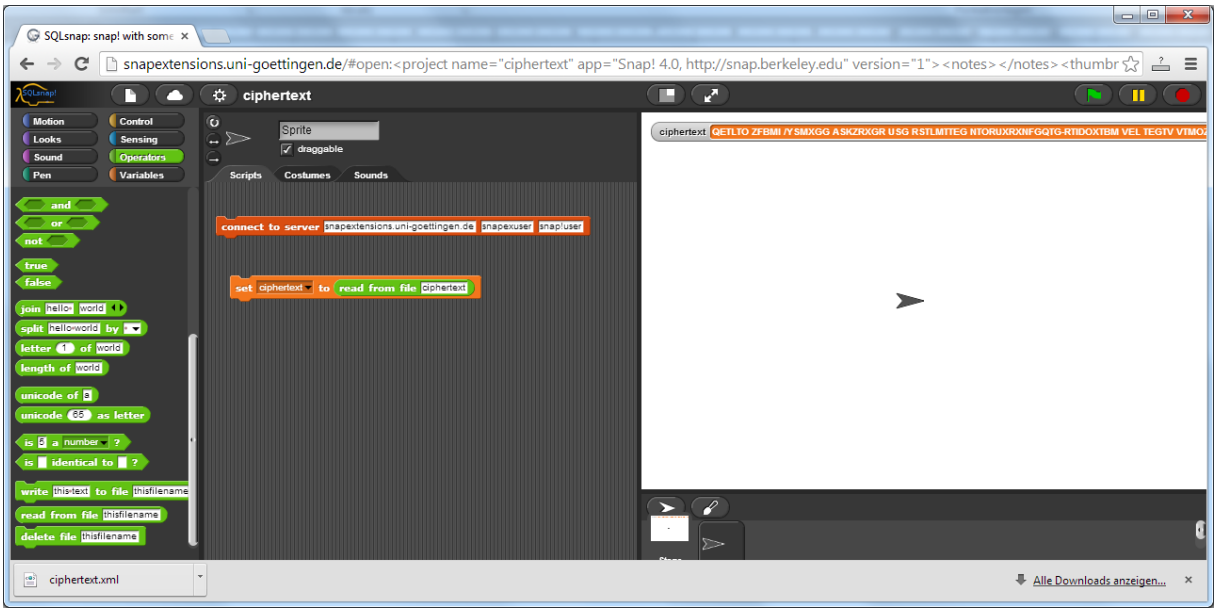


6. Using text files

Find an encrypted text (not too long). Copy it to a variable “ciphertext”. Connect to an appropriate server. Write the text in a text file.



Start SQLsnap on the same or another computer. Connect to the server. Read the ciphertext from the text file.



Try to identify the language of the ciphertext.

Try to decrypt it.

7. Installing SQLsnap

Load the code from the server (ZIP file).

Find an appropriate server or install one on your computer (XAMPP, ….).

Copy the PHP-script to the correct directory (XAMPP: htdocs).

Unzip the SQLsnap files to an appropriate directory, set the path in index.html.

Run SQLsnap.