

# Programming – Part 1

## Objectives

You will get to know the Scratch programming environment and how to use it in order to create and manage your own programming projects.

You will learn the basic elements of the Scratch programming language and use them to "write" your first simple programs.

You will reflect on your experiences with programming.

You will also learn (without too many theoretical considerations) basic concepts of "professional" programming (object and event orientation, process communication).

---

## Introduction

*Computers* are problem-solving machines. *Programs* contain the commands that instruct a computer to solve a particular problem (e.g. translate a text, control a robot, edit photos). *Programming* means thinking up the commands needed to solve a problem and formulating them in a language that computers can understand (*programming language*).

In order to be able to write and execute our own programs on a computer, we need a "tool", a so-called *programming environment*.

We use the Scratch programming environment (<https://scratch.mit.edu>) in the programming course. Scratch is web-based; all you need to use Scratch is a computer with active internet access and a browser (e.g. Safari, Edge, Chrome, Firefox, Opera, Brave).

If you prefer to use Scratch offline, you can also install the app locally on your own device (download: <https://scratch.mit.edu/download>).

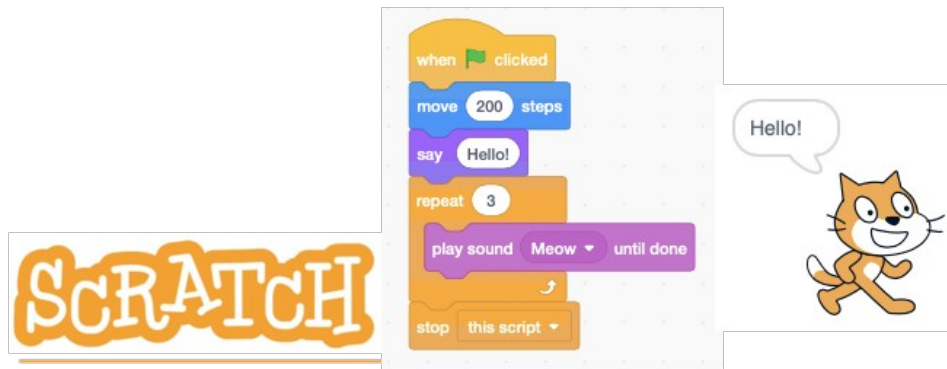


Fig. 1: Programming in Scratch is based on symbols

## Setting up your personal programming environment

To be able to write, manage and share several of your own programs, you need to create an account (Become Scratcher / Join Scratch). To confirm the creation of the account, you need a valid e-mail address.

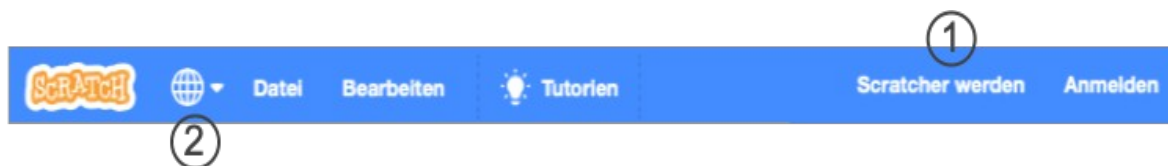
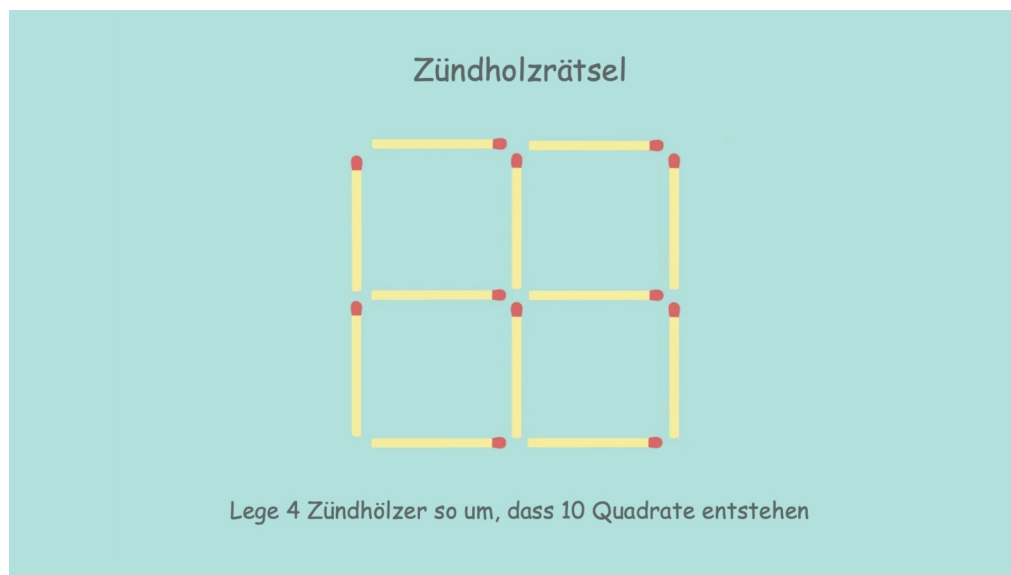


Fig. 2: Menu bar of Scratch

- ⇒ Create a Scratch account (Fig. 2: ①). As a student, always use an email address without your real name and other personal details. As a teacher, however, you are free to use your real name.
- ⇒ As a future teacher, apply for a teacher account so that you can later use all the possibilities it offers (e.g. if you want to publish projects).
- ⇒ Then select the desired language for using Scratch (Fig. 2: ②).

## Get to know Scratch and make a first program yourself

The example program: a match puzzle



*Fig. 3: The puzzle to be programmed*

If you don't have real matches available you can use a program to model the puzzle on the computer. fans of brainteasing can then use the program to move the matches on the screen and try to solve the puzzle.

The following explanations will help you to make such a program yourself step by step and to become familiar with the Scratch programming environment.

⇒ Log in to Scratch with your access data (user name, password) and open a new project.

## The components of the Scratch programming environment

### ----- The stage

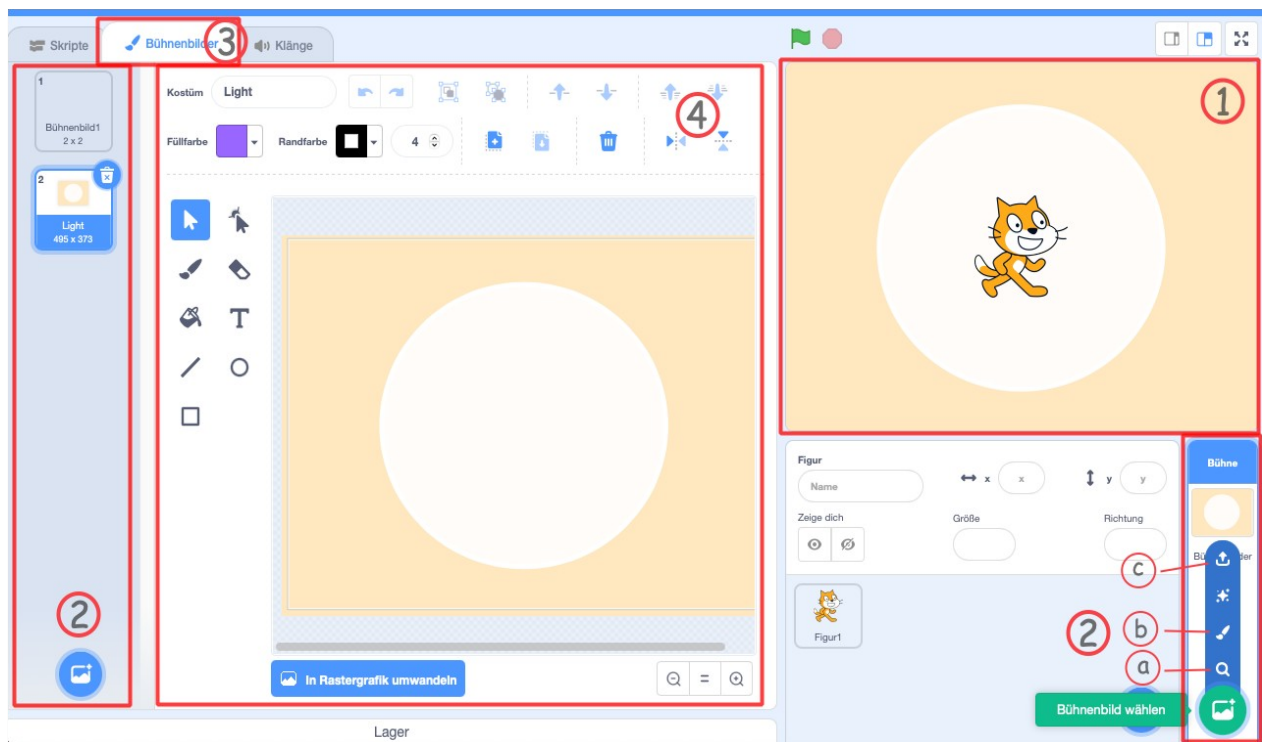


Fig. 4: The Scratch stage and its elements

The events of the program take place on the stage (Fig. 4: ①).

The stage set can be designed yourself using the simple drawing aid (Fig. 4: ②b) / ③ / ④). In addition, numerous templates are available (Fig. 4: ②a). It is also possible to upload image files (Fig. 4: ②c). Within a program, one can also switch dynamically between several stage sets (e.g. in a game or in a story).

Stage sets are background pictures with static elements that do not change while the program is running.

⇒ Select a stage set suitable for the match puzzle and place the program title "Match Puzzle" at the top centre.

### ----- Objects

The objects (characters/sprites) play roles on stage. To do this, they have certain properties (e.g. costume, size, position) and actions (e.g. movements, calculations) that they can perform.

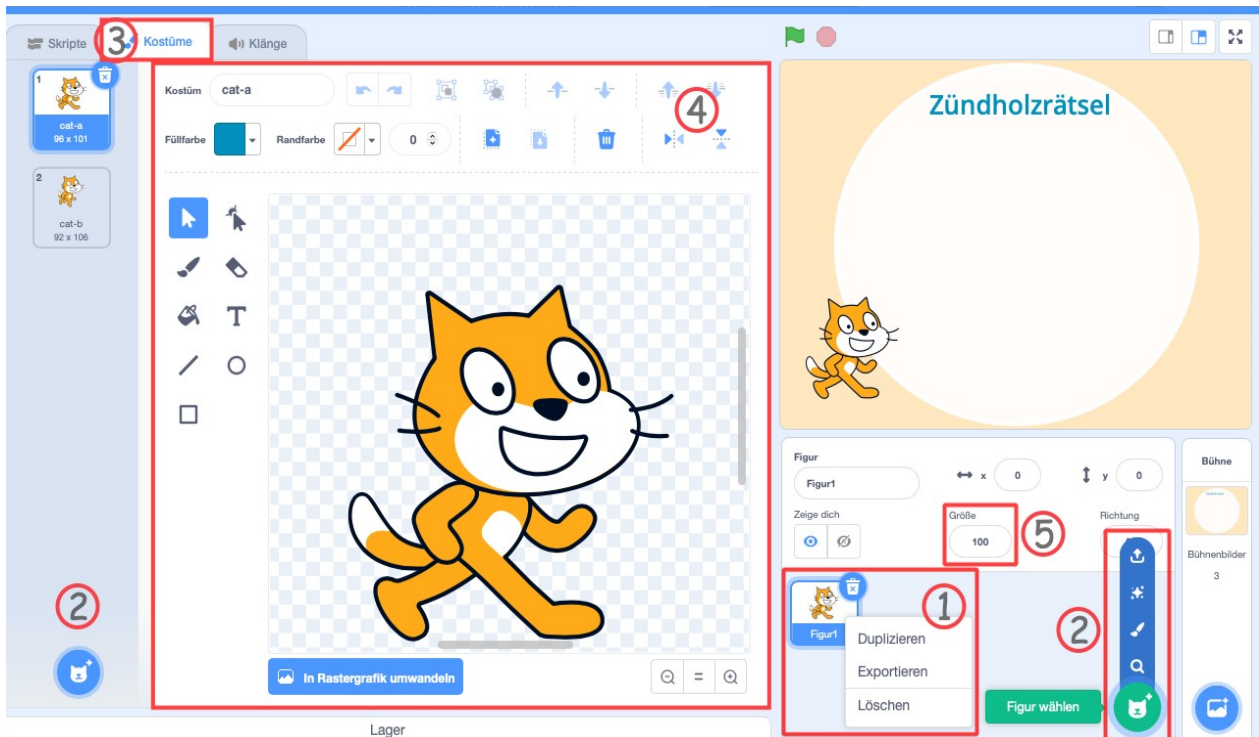


Fig. 5: Properties of an object (a character)

The object "Cat" (character 1 in Scratch) is preset in a new project. However, we need matches for the puzzle. You can therefore delete character 1 (Fig. 5: ①; click on the thumbnail to open the context menu).

We can "create" a new object by selecting a character (Fig. 5: ②), uploading a suitable picture (Fig. 5: ②) or drawing a costume ourselves (Fig. 5: ② / ③ / ④).

- ⇒ Upload the image of a match. You can use the file "Match.svg" provided in the materials section for this purpose. Or draw a match yourself.
- ⇒ If necessary, adjust the size of the object on the drawing area (Fig. 5: ④) or change the relevant property value (Fig. 5: ⑤).

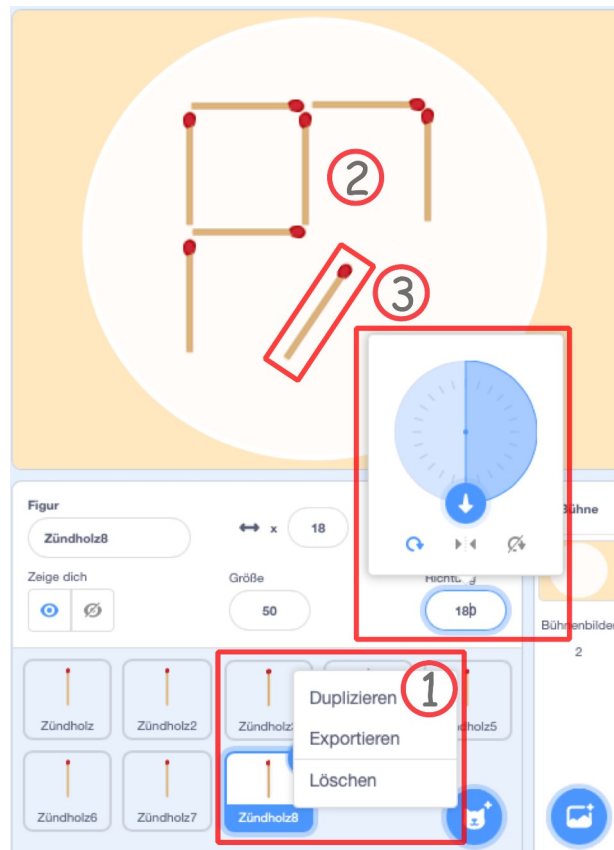


Fig. 6: Placing and rotating objects

We need 12 matches for our program, i.e. 12 objects. We can create several objects of the same kind simply by duplicating one object repeatedly (Fig. 6: ①).

The objects can be placed freely on the stage (fig. 6: ②). They can also be freely rotated (change direction) (fig. 6: ③).

⇒ Create the objects needed for the puzzle (matches) and form the given 5 squares on the stage with the 12 matches.

## ----- Saving, describing and publishing the program

In Scratch, the term "project" is used instead of "program". The project includes the executable program and a description.

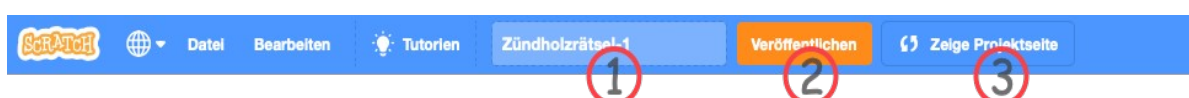
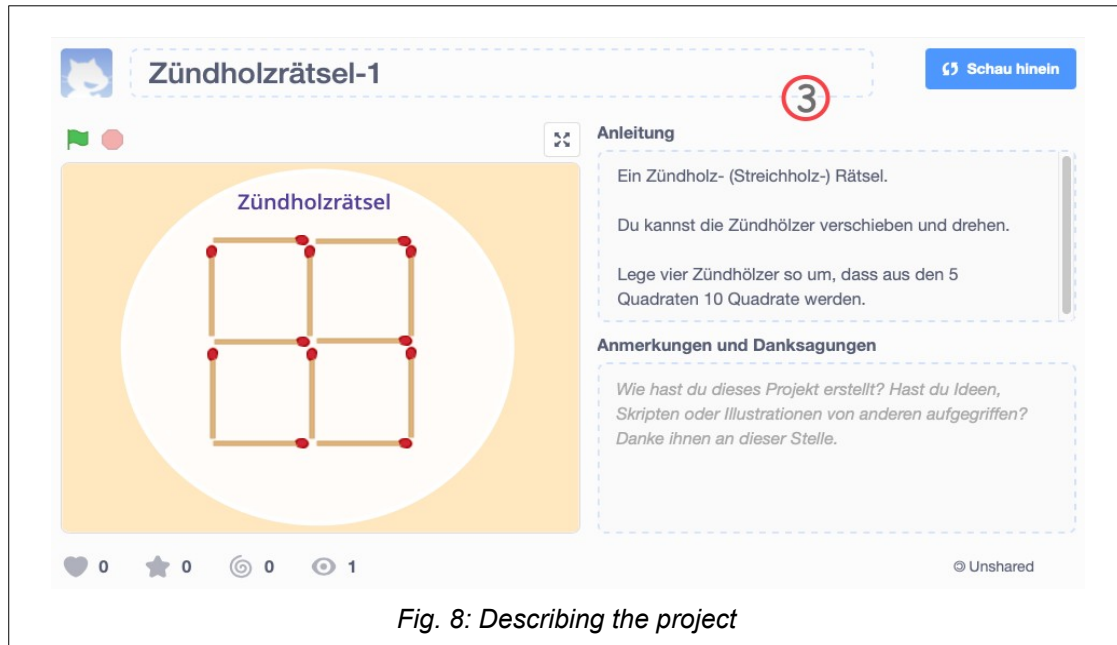


Fig. 7: Naming, describing and publishing a project

⇒ Give your program / project a fitting name (Fig. 7: ①).

The program is always saved automatically. However, it is advisable to save the program yourself from time to time.

If you want other people to be able to use your program, you must publish it (Fig. 7: ②). On the project page (Fig. 7 and Fig. 8: ③) you tell other people what the program is for and how they can use it.



## Reflection

The match puzzle is now ready. You can share it with others or use it yourself to solve the match puzzle.

The focus is on the objects (in this case the matches) with their properties (appearance, size, position, location). This programming concept is called object orientation. However, the objects do not (yet) act in our program.

In the editing mode ("See inside") you can move and rotate the objects on the stage. The necessary commands to the computer are part of the Scratch programming environment. The Scratch programmers developed and realised them.

## Objects and actions

After an attempt to solve the puzzle, the twelve matches are scattered on the stage. Restoring the original arrangement "by hand" is laborious. But by programming, you can automatically set up the starting order.

In order to automatically bring the matches into the "starting position", a "script" must be assigned to each object (match). A script generally comprises the necessary instructions (commands to the computer) to carry out a desired action.

The program start is triggered by an event. In Scratch, this is a click on the "Green Flag" (Fig. 9: ①). The red symbol to the right is used to stop the program.

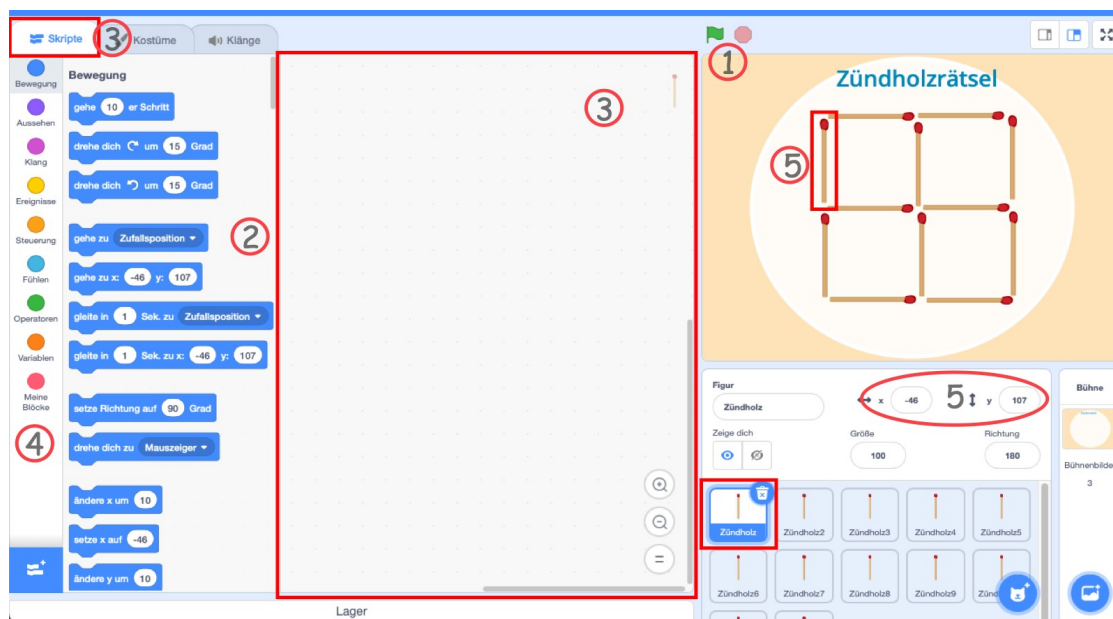


Fig. 9: Assigning scripts (actions) to an object

The commands required for a script can be selected from the list of commands provided (Fig. 9: ②) and placed in the script window (Fig. 9: ③). The commands are listed in the form of symbols and arranged in blocks (movement, appearance, events, control, etc.), which greatly simplifies the programming work (Fig. 9: ④).

### ----- Initialisation

The actions to be performed when a program is started are called "initialisation".

To place a match correctly when starting the program, its orientation and position must be set.





Fig. 10: Initialisation script for a match

The position of an object on the stage is defined with the values x and y in the coordinate system of the stage (Fig. 9: ⑤). The zero point of the coordinate system ( $x=0$ ,  $y=0$ ) is at the centre of the stage (Fig. 11).



Fig. 11: The coordinate system of the Scratch stage

The Video <https://vimeo.com/666437026> shows how a script can be formulated and commented.

⇒ Add the initialisation script to one of the objects of your match puzzle. Check whether the script works (by moving the match to any other position and then clicking on the "Green Flag").

Each individual object (match) must be assigned its own initialisation script so that it is positioned correctly when the "Green Flag" is clicked.

- ⇒ Assign to all objects of your match puzzle the necessary initialisation script. Instead of entering the script "by hand" for each object, you can select, copy and paste the already existing script; however, even then you must enter the correct x and y position "by hand" in each case.
- ⇒ Now test your program. Is the initialisation correct? Adjust the initialisation scripts if necessary.

## ----- Text output

Users would certainly be grateful if the puzzle is briefly explained when the program is started. A short tutorial text would be helpful.

Scratch only supports text output as actions of an object. Therefore, an additional object is needed that displays the instruction text on the stage when the "Green Flag" is clicked.

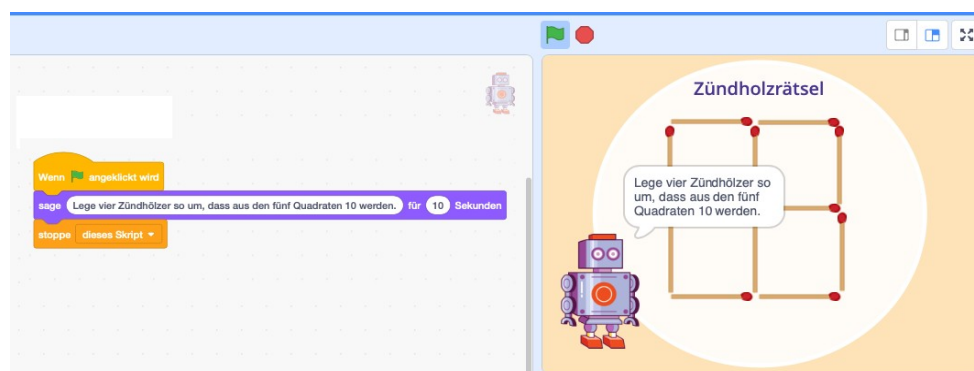


Fig. 12: Script for brief text output

- ⇒ Select an object (a character) from the template collection.
- ⇒ Add the script for the text output you wish to use to the selected object. Check whether the text is started as desired after clicking on the "Green Flag".

Scratch offers the possibility to expand the range of commands with additional blocks. Among others, there is a block for converting a text into spoken language: "Text to Speech" (Fig. 13: ⑥).

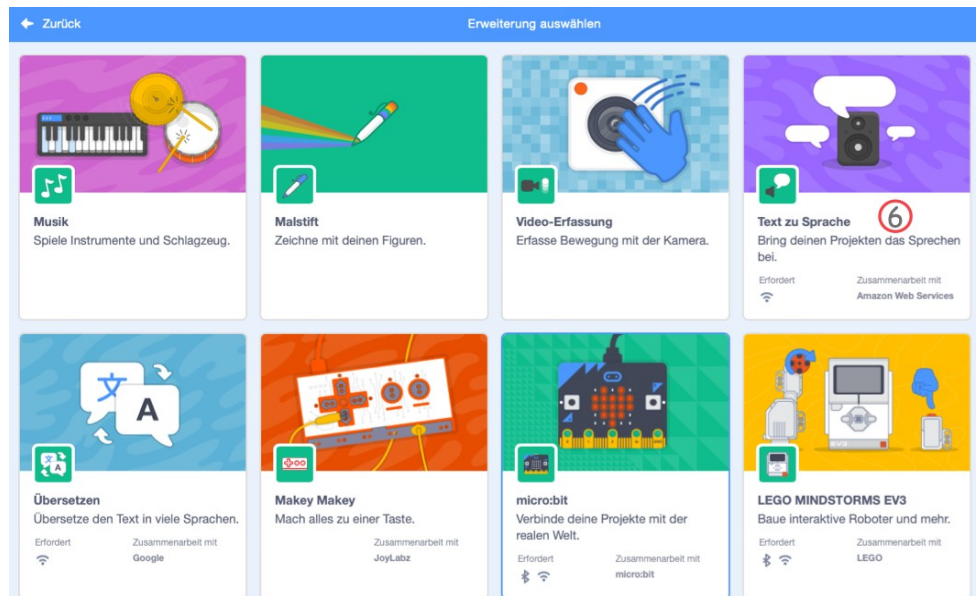


Fig. 13: Blocks for expanding the command scope of Scratch

In order to use the corresponding commands in a script, the block must first be activated.

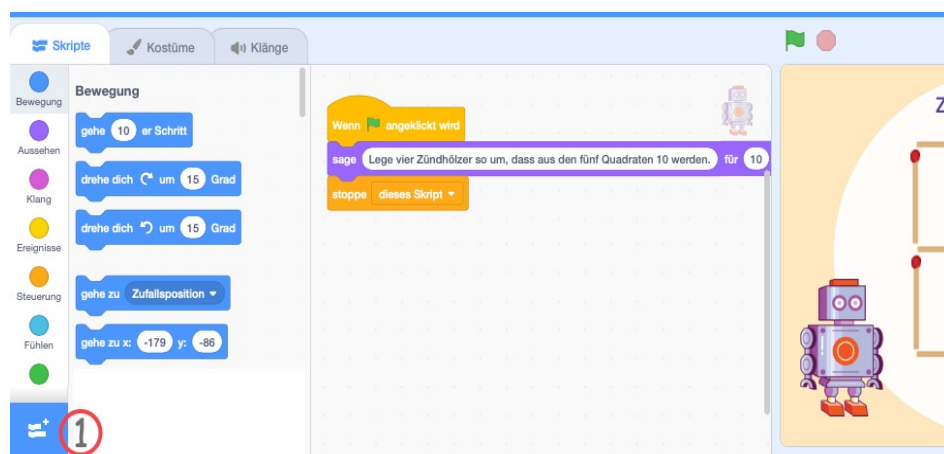


Fig. 14: Activation of extension blocks (①)

⇒ Activate the block (extension) "Text to Speech". (Fig. 14: ①)

After activating the extension (Fig. 15: ①), the commands of the block (Fig. 15: ②) can be used.

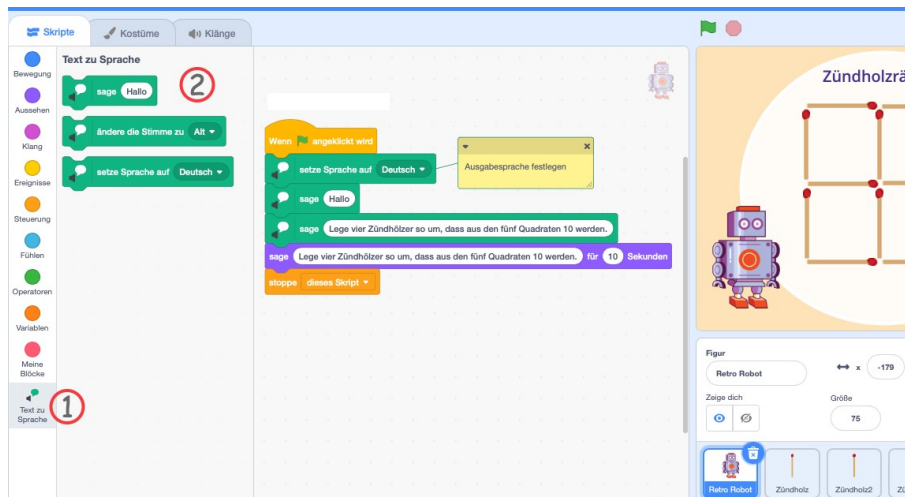


Fig. 15: Using the commands of an extension block

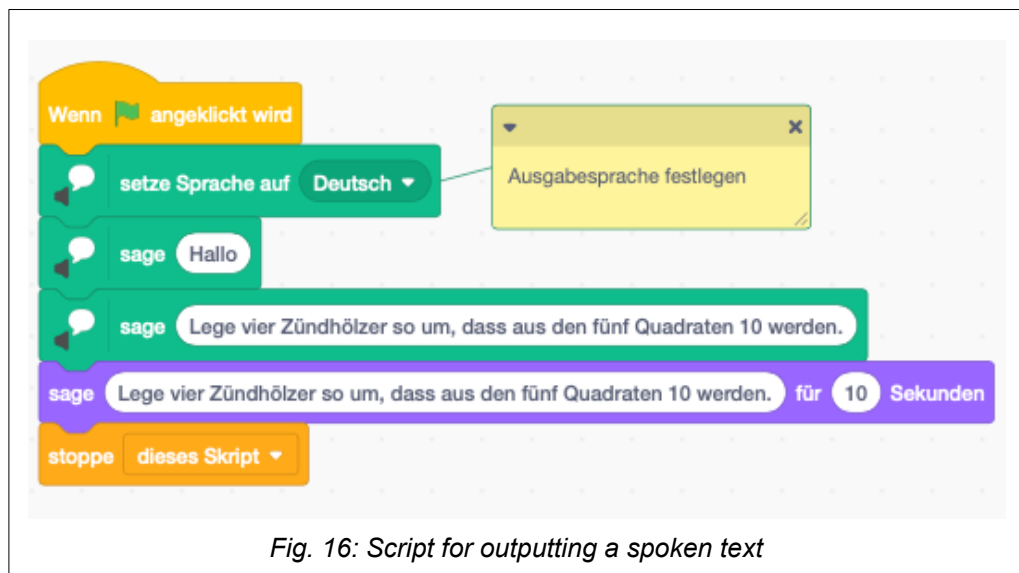


Fig. 16: Script for outputting a spoken text

- ⇒ Complete the script for text output with the corresponding commands for voice output (Fig. 16).
- ⇒ Check the new script. (If you cannot hear the voice output, you may need to adjust the audio settings of your computer).

## Process communication

The users of the match puzzle would probably like to have a sample solution to help them if they cannot find the solution themselves, or to confirm their own solution.

The character you introduced (as an object) for the text output could also prompt the presentation of the sample solution. However, the twelve objects of the puzzle (the matches) would have to move themselves to their respective positions in the pattern solution; just as they do when initialised.

How can one object cause another object to perform a certain action (a certain script)?

Scratch provides commands for communication between objects (sending and receiving messages). For example, the text output object, when clicked, could send a message to all other objects (Fig. 17). The other objects would then each execute a suitable script when receiving the message.



Fig. 17: Script for sending a message to all objects

⇒ Add the script for sending (broadcasting) the message "sample solution" to the object for text output.

The object for text output thus comprises two separate scripts! Depending on the event ("Green Flag", clicking on the object), it can therefore perform two different actions.

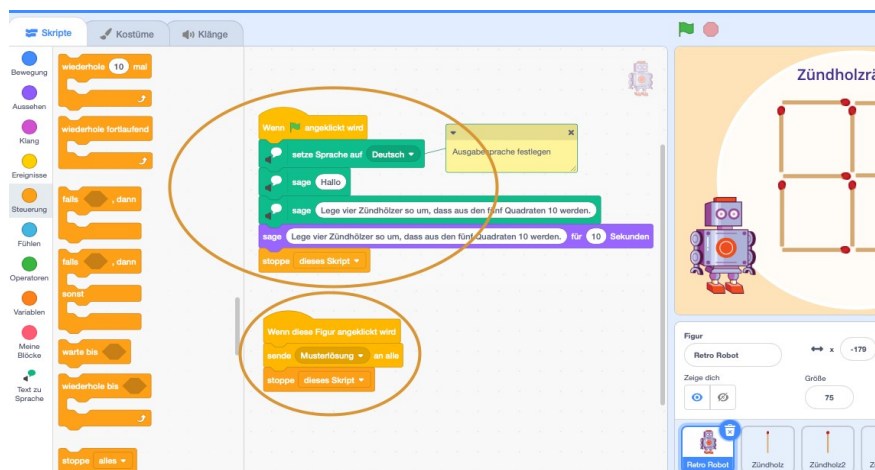


Fig. 18: An object can comprise several scripts

Now all other objects (matches) have to be completed with a script that reacts to the message and moves the respective object to the correct position of the sample solution. For example:



Fig. 19: Example of a script executed when receiving a message

⇒ Complete all match-objects with the script for receiving the message "sample solution" and then moving the match to the correct position in the sample solution.

Now all the objects in the puzzle comprise two separate scripts!

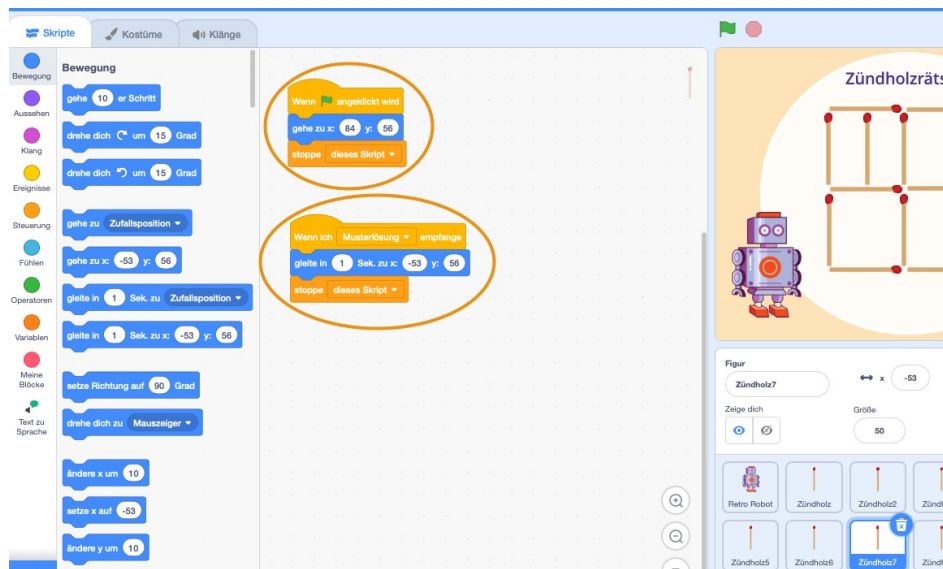


Fig. 20: All objects can comprise more than one script

## Reflection

A program comprises objects. In the example program these are the matches and the character chosen for the text output.

The objects are characterised by their properties (e.g. appearance, size, position, location) and the actions (scripts) assigned to them. This programming concept is called object orientation.

The initially independent objects can interact. The most common interaction is synchronisation (an object performs one of its actions when another object has completed a certain action). In Scratch, synchronisation takes place via exchanged messages (process communication).

---

## Remix

The puzzle with the 12 matches could be varied:

- Position 4 matches so that the five squares become 10 (previous version).
- Move 3 matches so that the five squares become 3 (new version).
- Take away 2 matches so that the five squares become 2 (new version).

Today's diversity of programs would not be possible if every program had to be developed anew, from scratch. Yet numerous programmers nowadays make their programs public and so allow other programmers to change and further develop the programs. Thus, different language versions of a program and programs with an extended range of functions can be created with a reasonable amount of effort. The best known of these are certainly the open source programs, whose program code is publicly accessible.

Scratch also makes it possible to publish one's own programs (projects). Other people can find such programs on the Scratch platform, obtain a copy, change the copy and publish it (under a new name). However, the project page of the modified program should then describe what the original program was and what was changed.

In Scratch, this process is called "remixing".

To take over someone's public project in Scratch and extend or change the program, just click on the Remix button on the project page of the program in question (Fig. 21). However, this is only possible if you have created a Scratch account and are logged in.

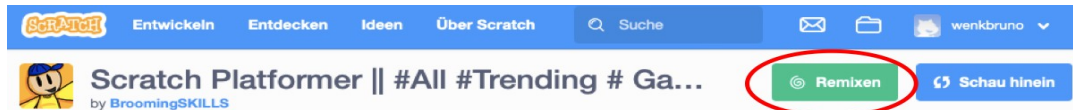


Fig. 21: Remixing a program

To remix your own program, you have to proceed differently (there is no remix button on the project page of your own programs). The video "Scratch Tutorial: How to Remix your OWN Projects" ( [https://youtu.be/dLY\\_MnAMe6o](https://youtu.be/dLY_MnAMe6o)) shows how to do it.

⇒ Realise the version "Flip 3 matches so that the five squares become 3" by remixing your existing match puzzle.

## Editing versus execution mode

To enable a person who is not a Scratcher (i.e. who has not created a Scratch account) to use your program, you must publish the program and send the person the link to your project. You can find the link to your project on the project page.

If you do this with your match puzzle, that person will see the starting line-up, see the text output and hear the voice output when the program starts. They will also see the sample solution by clicking on the additional piece you have chosen. But the person cannot move the matches! Why?

You have assigned the two scripts to the matches that are necessary to bring the matches into the positions of the starting line-up and the positions of the sample solution respectively. But the matches lack a script that allows the users to change the position on the stage with the mouse!

This was not necessary during programming because the move (and rotate) function that you could use in the edit mode is provided by the programming environment (in the background).

A person who is not a Scratcher can only use the program in execution mode. You would therefore still have to add a script to each match for moving it on stage (fig. 22).





Fig. 22: Script for moving an object (a character) with the mouse

---

### Reflection

The strong and rapid spread of computer programs is also related to the fact that new programs can build on existing ones. Open source programs, for example, benefit from this.

In your view, what advantages does open source software have over non-open source programs (e.g. the text processing program LibreOffice Writer over Microsoft Office Word)? What disadvantages do you see?

Could you imagine working on an open source project (e.g. as a translator, a programmer)?

---