

# Introduction to Scratch

## Problem solving and Scratch

---

Produced      Mairead Meagher  
by:            Dr. Siobhán Drohan



Waterford Institute *of* Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics  
<http://www.wit.ie/>

# Topics list

---

- Problem Solving
- Scratch: Offline Editor and Online Editor
- Flow of Control in a Program
- SomethingFishy Examples:
  - Example1: Sequence
  - Example2: Sequence, Selection and Iteration.
  - Example3: Sequence, Selection and Iteration.

# Problem Solving

---

- Programming **IS** problem solving.
- We will learn about problem solving **BEFORE** learning how to write Java code!
- You should **ALWAYS** think about how you are going to solve the programming problem **BEFORE** jumping in and starting to code.
- For this reason, we are going to start the module by using Scratch.

# Scratch

---

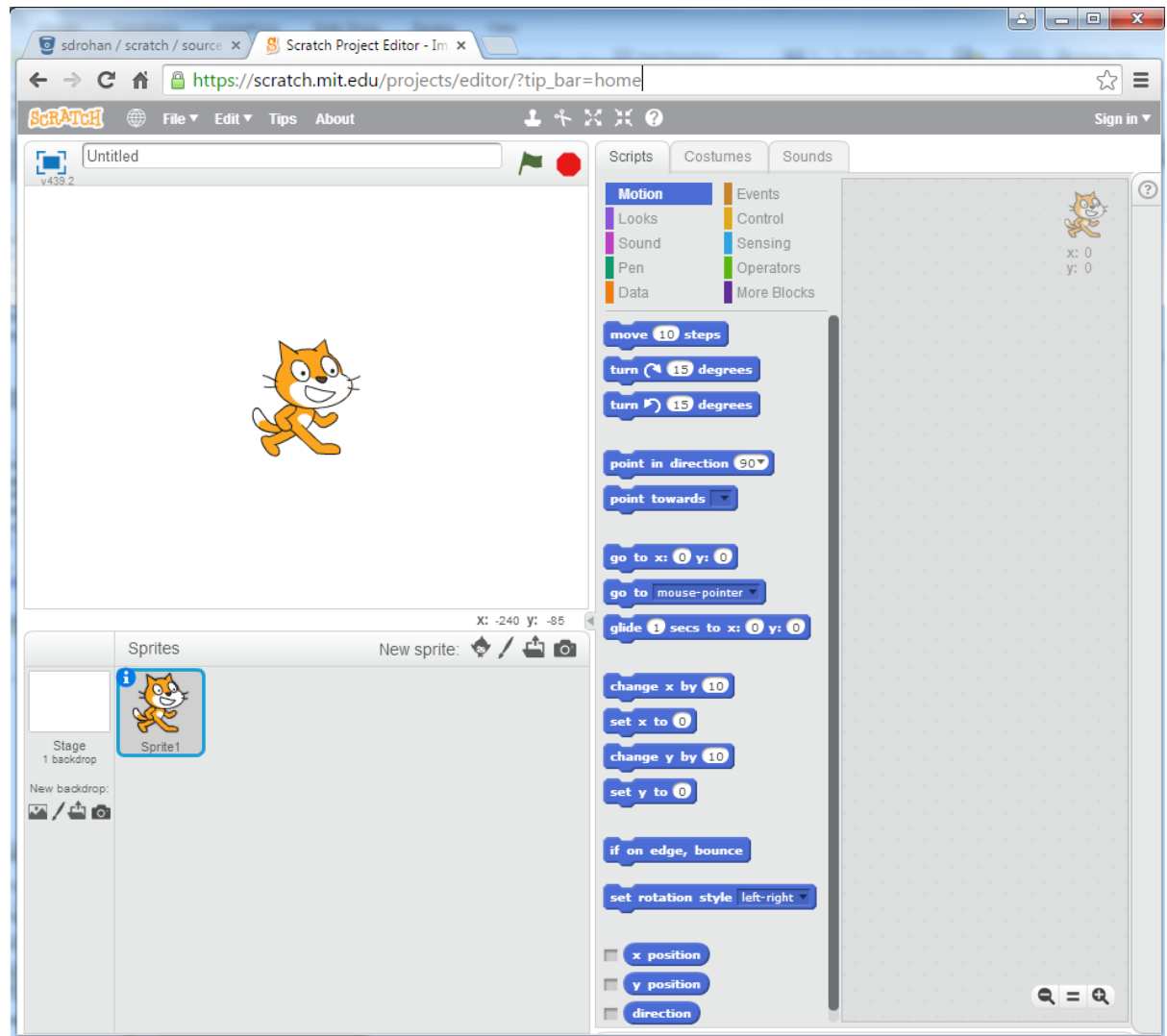
- Scratch is a graphical programming language.
- It was developed to help students to understand programming fundamentals and concepts.
- You write code in a Scratch editor.
- You can download the Offline Editor:  
<https://scratch.mit.edu/download>

Or

- You can use the online editor:  
[https://scratch.mit.edu/projects/editor/?tip\\_bar=home](https://scratch.mit.edu/projects/editor/?tip_bar=home)

# Scratch Online Editor

Demo



# Flow of Control in a Program

---

- Each program you write will typically have:

<b>Sequence</b>	Things that will be done in a particular order
<b>Selection</b>	Things that will be done conditionally
<b>Iteration</b>	Things that will be done repetitively

- By using examples, we will explore what each of these mean.

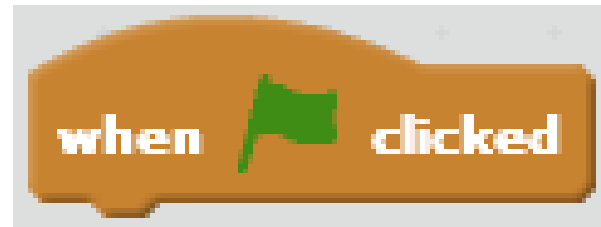
# Example 1

Demonstrating SEQUENCE

# Events

---

Runs the  
program when  
the green flag is  
clicked





# Statements

---

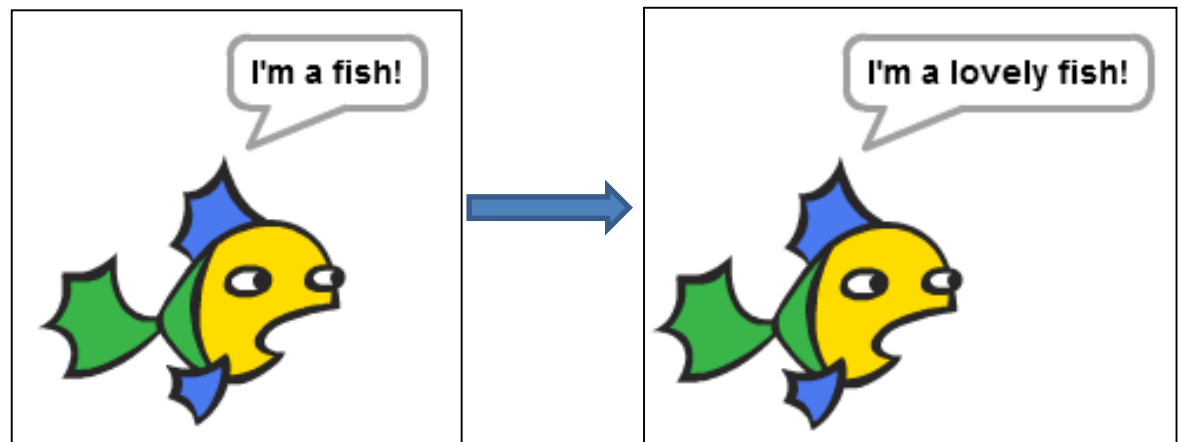
You can type in any text. The words will appear in a speech bubble for the sprite.



You can type in any words to say. The number of seconds tells the speech bubble how long to show. The program waits that long before continuing.



# SomethingFishy1

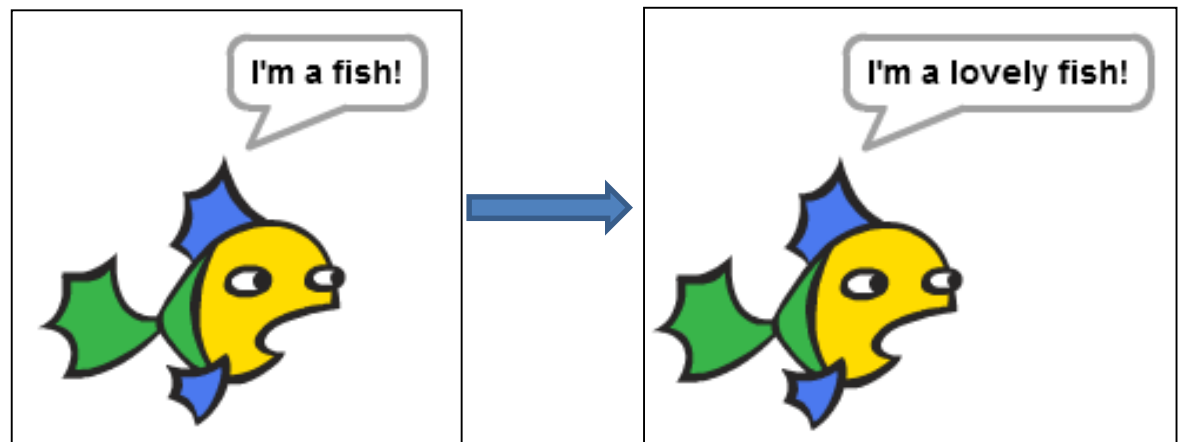


# SomethingFishy1



This example demonstrates  
**SEQUENCE**  
in a program.

The statements are executed in  
sequential order.





# Example 2

Demonstrating  
SEQUENCE, SELECTION and  
ITERATION

# Selection / Conditions

---

<p>If condition is true, runs the statements inside it.</p>	 A yellow Scratch 'if-then' block. It features a small hexagonal condition slot at the top, followed by the word 'if' on the left and 'then' on the right. Below the condition slot is a single long rectangular slot for a statement.
<p>If condition is true, runs the statements inside the <b>if</b> portion; if not, runs the statements inside the <b>else</b> portion.</p>	 A yellow Scratch 'if-then-else' block. It features a small hexagonal condition slot at the top, followed by the word 'if' on the left and 'then' on the right. Below the condition slot are two long rectangular slots for statements, with the word 'else' positioned to the left of the second slot.

# Example of a condition

---

Reports true if  
specified key is  
pressed.



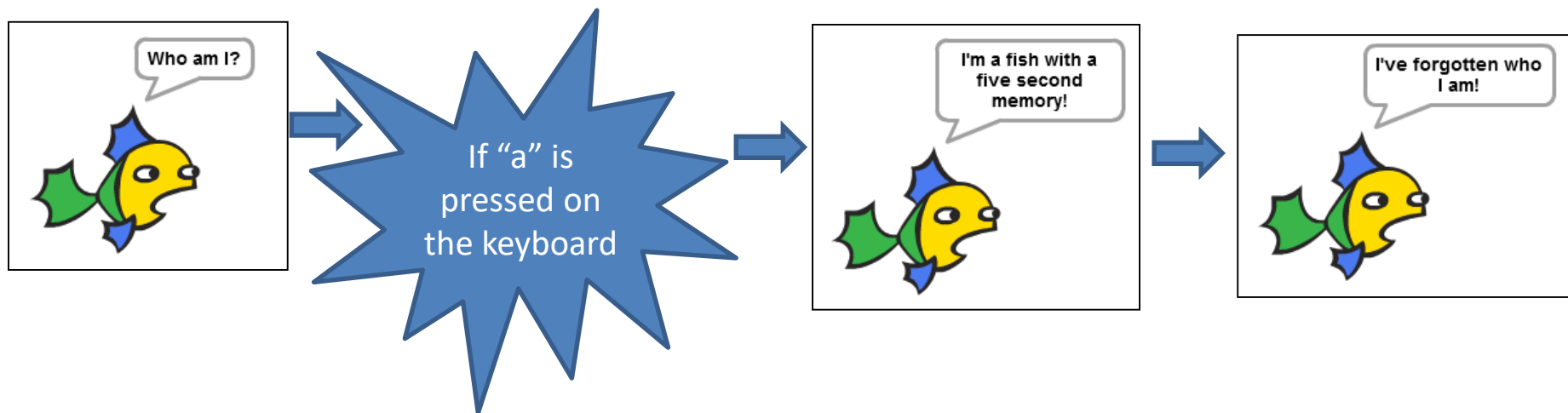
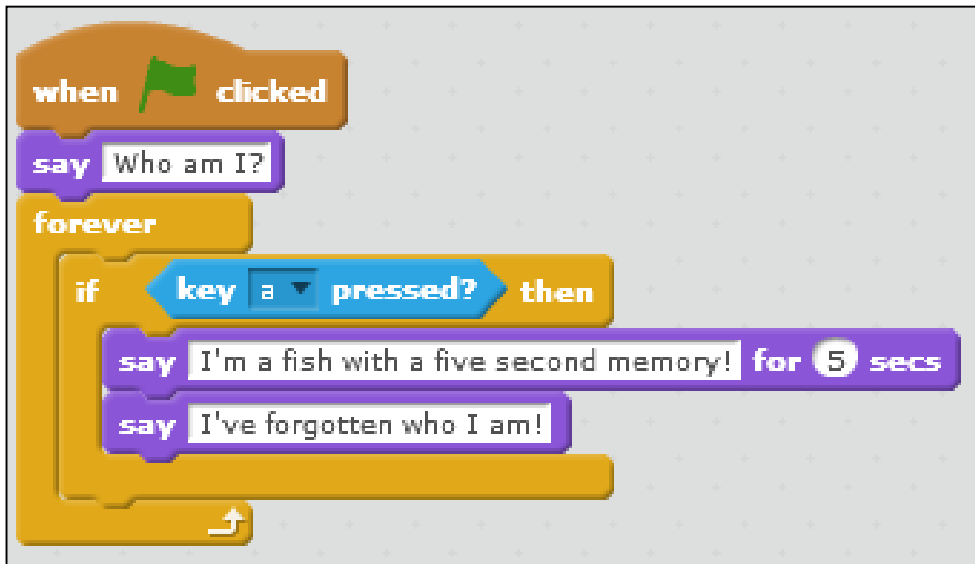
# Iteration / Loops

---

Runs the  
statements  
inside over and  
over again

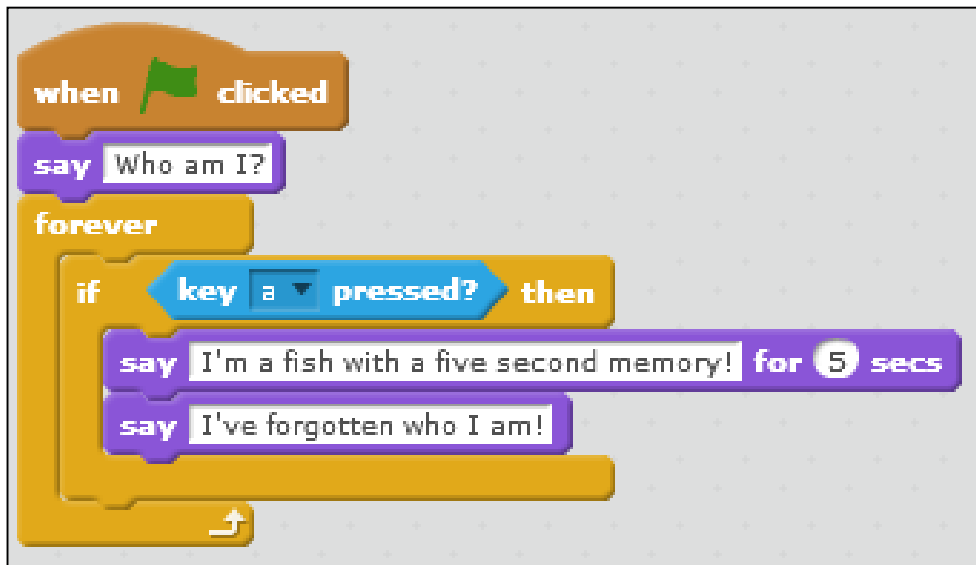


# SomethingFishy2





# SomethingFishy2



This example demonstrates:

**SEQUENCE** (The statements are executed in sequential order)

**SELECTION** (if “a” is pressed on the keyboard, the messages are printed to the speech bubbles. If “a” is not pressed, nothing happens)

**ITERATION** (The program is continually listening/waiting for the “a” key to be pressed).

# Example 3

More on  
SEQUENCE, SELECTION and  
ITERATION

# More Control / Loops

---

We saw this one in an earlier slide...it runs the statements inside over and over.



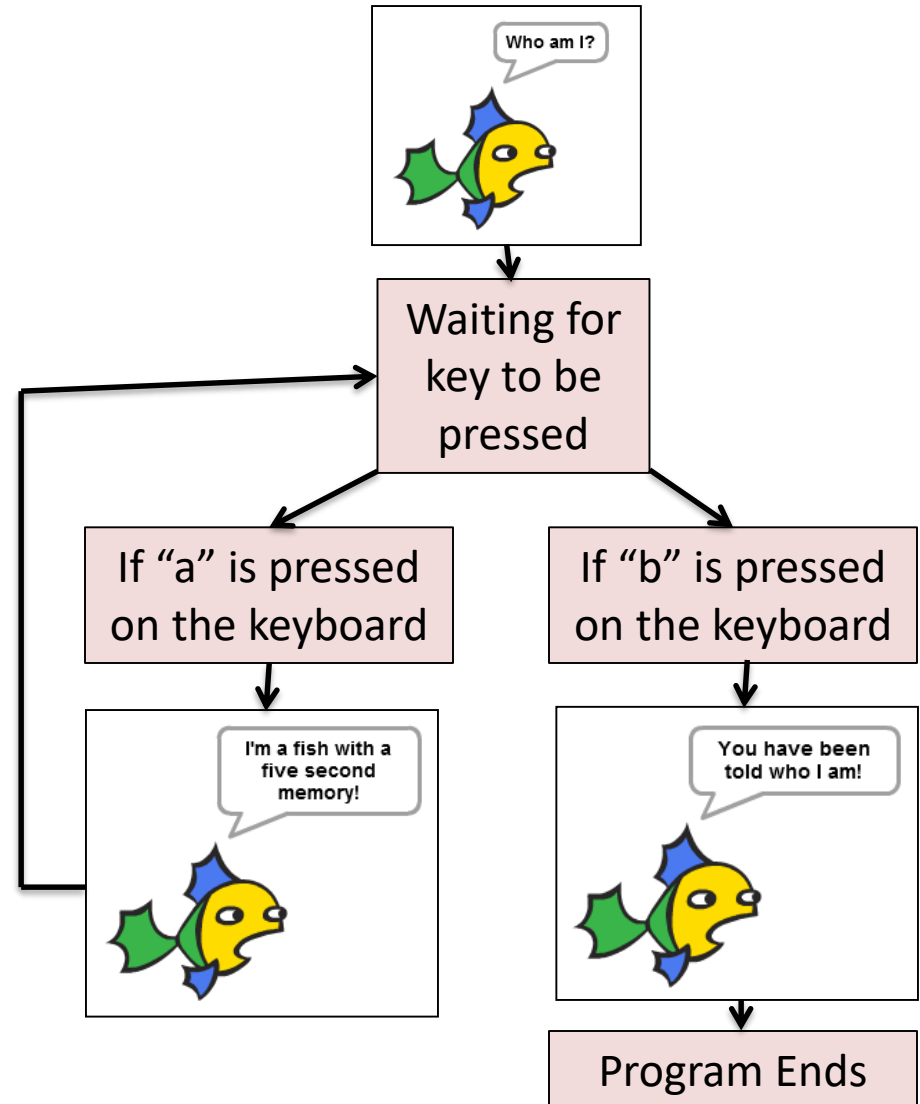
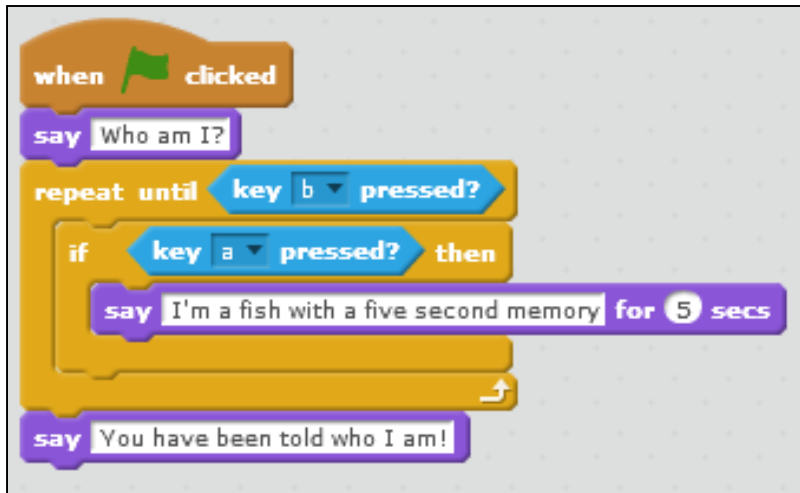
Repeat the statements inside until condition is true.

Checks to see if condition is false:

- if so, runs the statements inside and checks condition again.
- If condition is true, goes on to the statements that follow.



# SomethingFishy3



# Questions?

---



# References

---

- Vickers, P. (2008) *How to Think Like a Programmer: Problem Solving for the Bewildered*, Cengage Learning EMEA.