

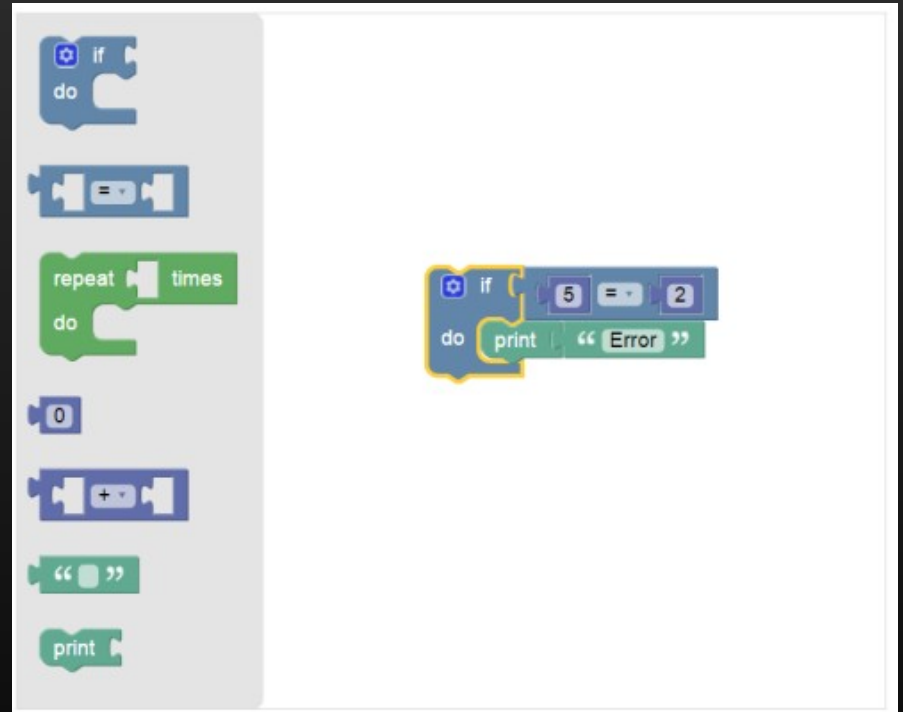
Langage visuel pour un exerciceur

Sommaire

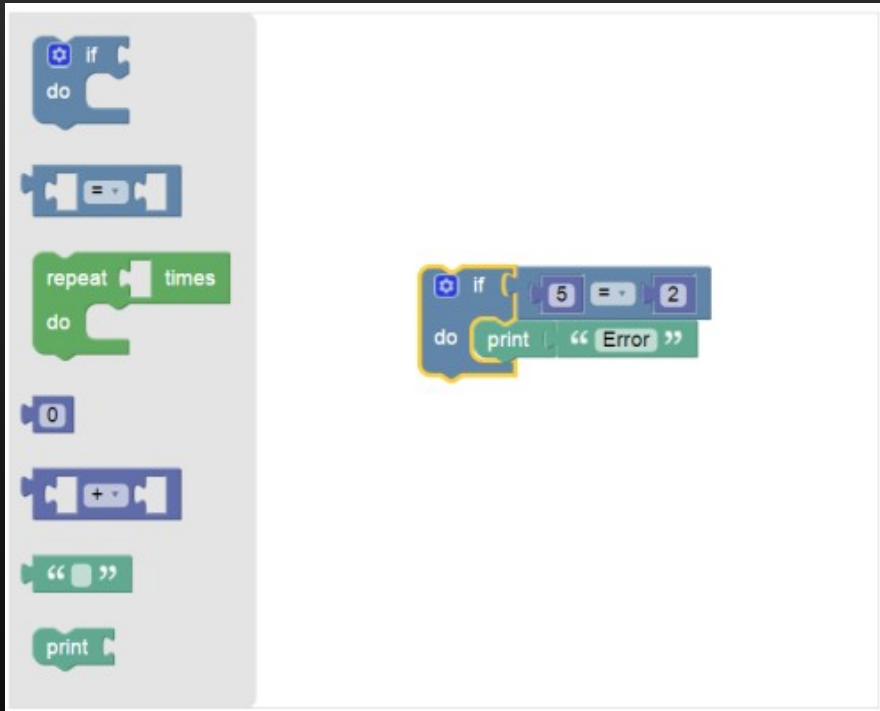
-
- Contexte
-
- Développement
-
- Conclusion

Contexte

- Blockly
-
- Nombreux outils
-
- JavaScript, Python, Dart, PHP
-
- Erreurs de syntaxe impossibles

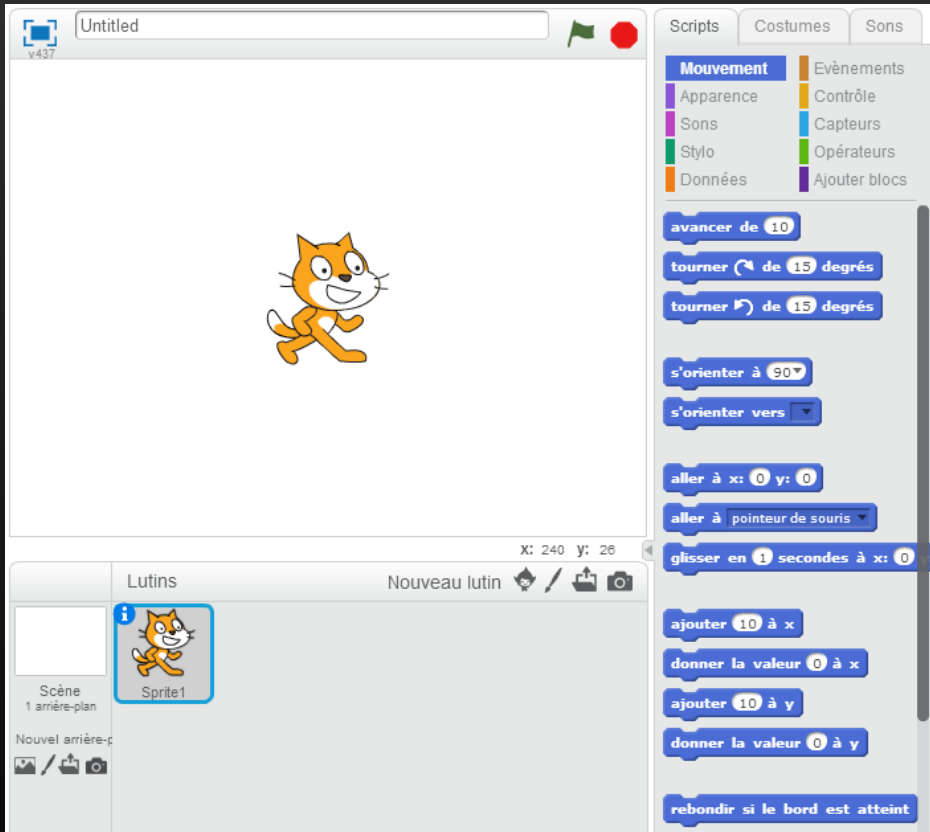


Contexte



- Blockly
-
- Assemblage de blocks
-
- Facile d'utilisation
-
-

Contexte



- Blockly
- Assemblage de blocks
- Facile d'utilisation
- Scratch

Contexte

- Outils, technologie et méthodes
- - Eclipse / Brackets
 -
 - Java / JavaScript
 -
 - Git via GitHub

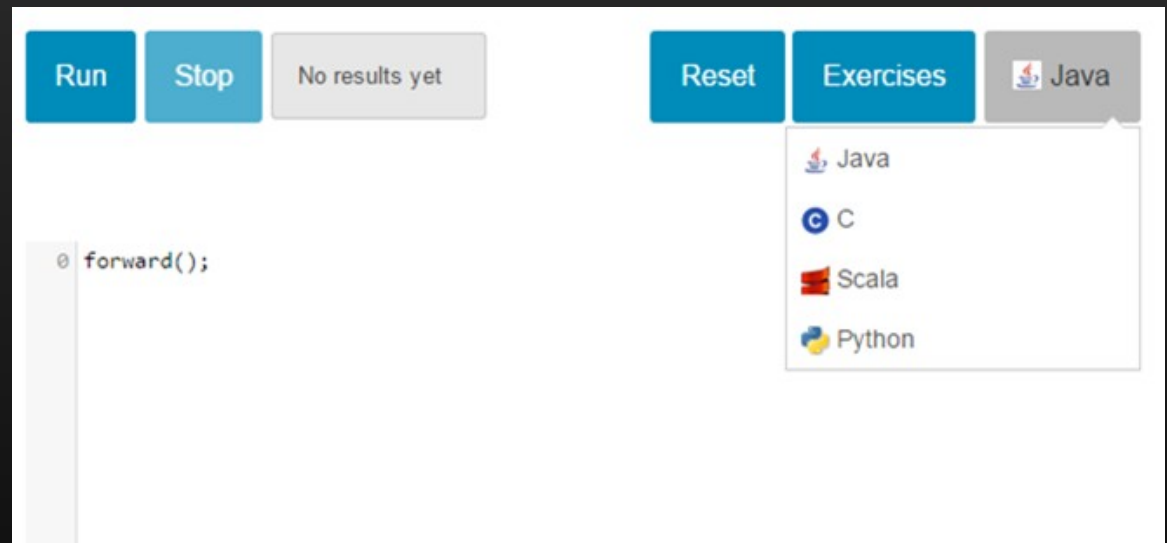


développement

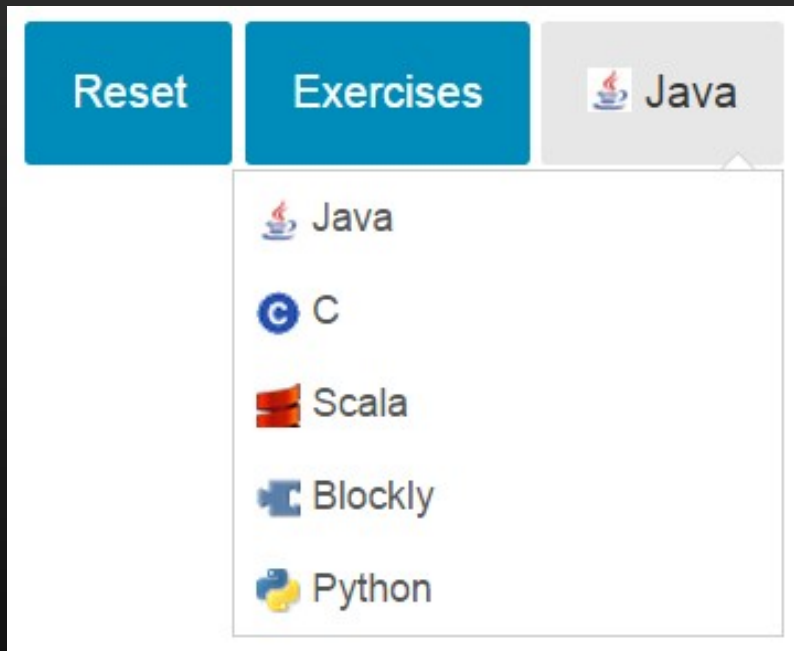
- Découverte
- - Fonctionnement de l'environnement
 -
 - Exploration du code
 -
 - Remise à niveau Framework AngularJS

développement

- Insertion
- - Sélection
 -
 - Affichage
 -
 - Génération



développement



- Sélection
 - Python
 - Deux fichiers
 - LangBlockly,java
 - *ExerciseEntity*,blockly

développement

- Affichage
-
- AngularJS (ng-show)
-
- Directive + Service
-

The image shows a Scratch script for a character named 'Buggle'. The script is organized into a list on the left and a main script area on the right. The list includes: Buggle, Logic, Loops, Move, World. The main script area contains the following code blocks:

```
back
repeat while (not isFacingWall)
do
  repeat while (not isOverBaggle and not isFacingWall)
  do
    forward
  if (isOverBaggle)
  do
    pickupBaggle
    back
    repeat while (not isOverBaggle)
    do
      forward
    backward
    dropBaggle
    back
    forward
  right
  forward
  left
```

A trash can icon is visible in the bottom right corner of the script area.

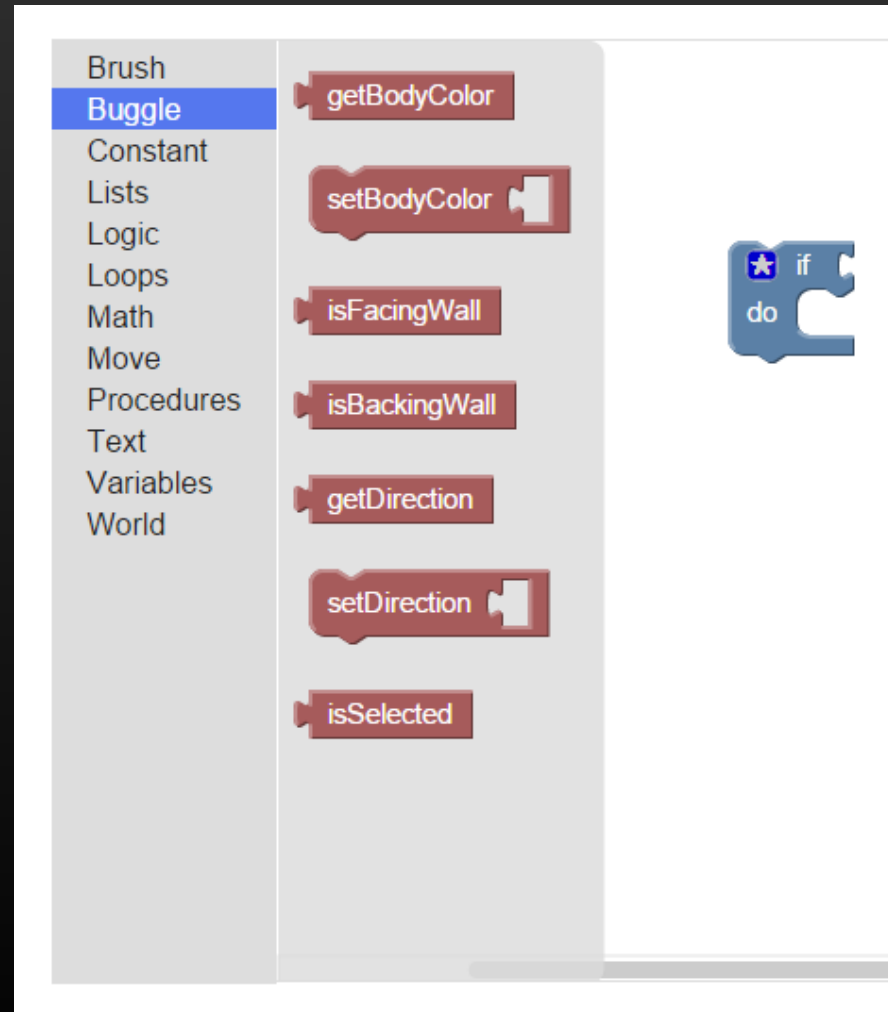
Développement

```
back()
while not (isFacingWall()):
    while not (isOverBaggle()) and not (isFacingWall()):
        forward()
    if isOverBaggle():
        pickupBaggle()
        back()
        while not (isOverBaggle()):
            forward()
        backward()
        dropBaggle()
        back()
        forward()
right()
forward()
left()
forward()
```

- Génération
- - Exercise.controller.js
 -
 - Python
 -
-

développement

- Toolbox
- Bloc
- Liaison



développement



forward()

- Bloc
-
- Nouveaux blocs
-
- Visuel et fonctionnel

développement

Blockly > Demos > Block Factory

Preview: LTR

Help

Input
Field
Type
Colour

name math_foo

inputs dummy input

fields left text forward

automatic inputs

top+bottom connections

top type

bottom type

colour

forward

Language code: JavaScript

```
Blockly.Blocks['math_foo'] = {
  init: function() {
    this.appendDummyInput()
      .appendField("forward");
    this.setPreviousStatement(true);
    this.setNextStatement(true);
    this.setTooltip('');
    this.setHelpUrl('http://www.example.com/');
  }
};
```

Generator stub: Python

```
Blockly.Python['math_foo'] = function(block) {
  // TODO: Assemble Python into code variable.
  var code = '...';
  return code;
};
```

développement

[Blockly](#) > [Demos](#) > Block Factory

Input
Field
Type
Colour

The image shows a Blockly 'Block Factory' interface. On the left, a sidebar lists 'Input', 'Field', 'Type', and 'Colour'. The main area displays a block definition for 'move_forward'. The block is green and has the following fields and inputs:

- name:** move_forward
- inputs:** dummy input (blue block), fields left (dropdown), text forward (text block)
- automatic:** automatic (dropdown)
- inputs:** inputs (dropdown)
- top+bottom connections:** top+bottom connections (dropdown)
- top type:** top type
- bottom type:** bottom type
- colour:** colour

développement

Language code: JavaScript ▾

```
Blockly.Blocks['move_forward'] = {  
  init: function() {  
    this.appendDummyInput()  
      .appendField("forward");  
    this.setPreviousStatement(true);  
    this.setNextStatement(true);  
    this.setTooltip('');  
    this.setHelpUrl('http://www.example.com/');  
  }  
};
```

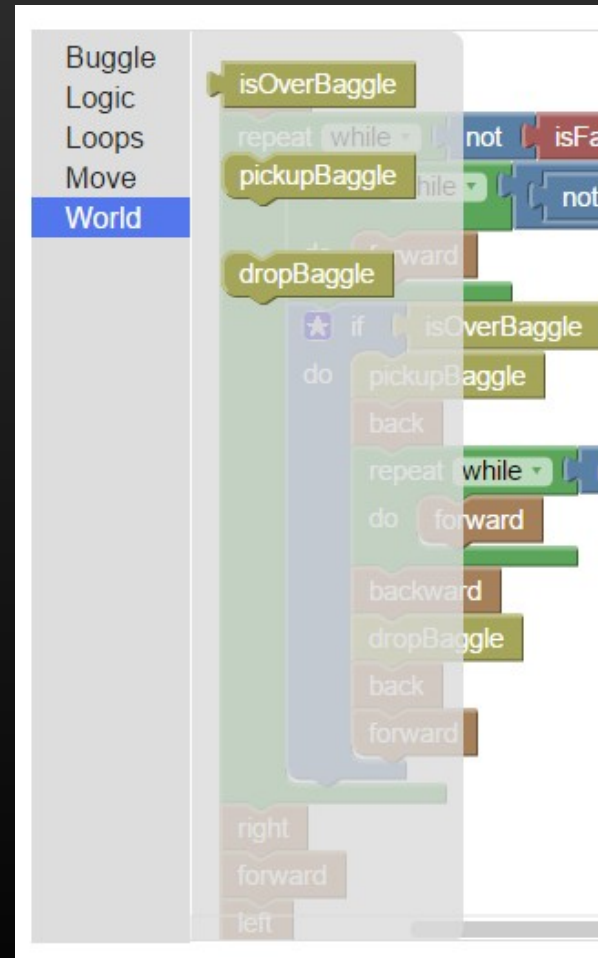
forward

Generator stub: Python ▾

```
Blockly.Python['move_forward'] = function(block) {  
  // TODO: Assemble Python into code variable.  
  var code = '...';  
  return code;  
};
```


développement

- Liaison
- Boite d'outils / exercice
- Non obligatoire
- JSON simple



développement

- Liaison
- Boite d'outils / exercice
- Non obligatoire
- JSON simple

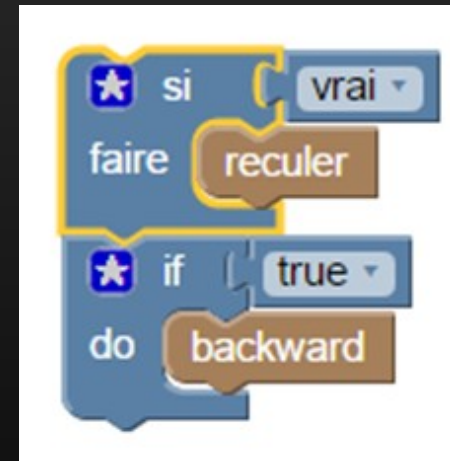
```
1 1 [{"
2     "name": "Buggle",
3     "blocks": [{"
4         "type": "buggle_facingWall"
5     }], {
6     }, {
7     "name": "Logic",
8     "blocks": [{"
9         "type": "controls_if"
10        }, {
11        "type": "newlogic_operation"
12        }, {
13        "type": "logic_negate"
14        }], {
15    }, {
16    "name": "Loops",
17    "blocks": [{"
18        "type": "controls_whileUntil"
19    }], {
20    }, {
21    "name": "Move",
22    "blocks": [{"
23        "type": "move_forward"
24        }, {
25        "type": "move_backward"
26        }, {
27        "type": "turn_right"
28        }, {
29        "type": "turn_left"
30        }, {
31        "type": "turn_back"
32        }], {
33    }, {
34    "name": "World",
35    "blocks": [{"
36        "type": "world_baggle_ground"
37        }, {
38        "type": "world_baggle_pickup"
39        }, {
40        "type": "world_baggle_drop"
41        }], {
42    }
}]
```

développement

- Exemple pour les blocs forward puis backward
-
- ```
<xml xmlns=« http://www.w3.org/1999/xhtml"><block type="move_forward" id="6" x="25"><next><block type="move_backward" id="12"></block></next><block></xml>
```
- Stockage
- 
- Reprise du travail
- 
- Suivi de l'utilisateur
- 
- XML
-

# développement

- Traduction
- - 3 systèmes
  - 
  - Blockly: riche / rechargement
  - 
  - Service



# Conclusion

- 
- Blockly fonctionnel
- 
- Piste de poursuite de développement
- 
- Complément d'expérience vis-à-vis de l'IUT

Merci de votre attention

Des questions ?