






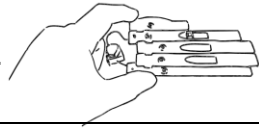
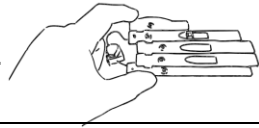


Liste des blocs Scratch reconnus

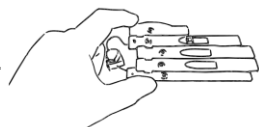
# ID	Signification du bloc	topcode		opcode	[inputs]		
1	Quand drapeau cliqué	31		event_whenflagclicked			
2	Quand ... est pressé	47		event_whenkeypressed	F	KEY_OPTION	space
3	Si ...alors	55		control_if	CONDITION	SUBSTACK	
4	Répéter Jusqu'à ce que	59		control_repeat_until	CONDITION	SUBSTACK	
5	Répéter ... fois	61		control_repeat	TIMES	SUBSTACK	
6	Répéter indéfiniment	79		control_forever	SUBSTACK		
7	Attendre ... secondes	87		control_wait	DURATION		



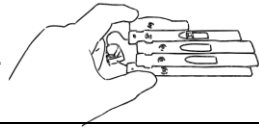
8	Si ... alors ... sinon	91		control_if_else	CONDITION	SUBSTACK	SUBSTACK2
9	Fin de boucle	103		control_stop	F	STOP_OPTION	all
10	Montrer	107		looks_show			
11	Cacher	109		looks_hide			
12	Dire ... pendant ... secondes	115		looks_sayforsecs	MESSAGE	SECS	
13	Dire	117		looks_say	MESSAGE		
14	Effacer tout	121		pen_clear			
15	Stylo en position basse (position d'écriture)	143		pen_penDown			



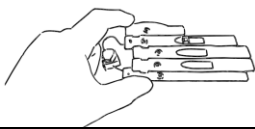
16	Stylo en position haute (relever le stylo)	151		pen_penUp			
17	Avancer de ...	155		motion_movesteps	STEPS		
18	Tourner de ... (sens horaire)	157		motion_turnright	DEGREES		
19	Tourner de ... (sens anti-horaire)	167		motion_turnleft	DEGREES		
20	S'orienter à ...	171		motion_pointindirection	DIRECTION		
21	Aller à x : ... y : ...	173		motion_gotoxy	X	Y	
22	Ajouter ... à x	179		motion_changexby	DX		
23	Mettre x à :	181		motion_setx	X		



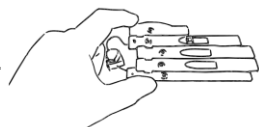
24	Ajouter ... à y	185		motion_changeyby	DY		
25	Mettre y à :	199		motion_sety	Y		
26	Abscisse x	203		motion_xposition			
27	Abscisse y	205		motion_yposition			
28	Direction	211		motion_direction			
29	Définir Bloc 1	213		procedures_definition			
30	Définir Bloc 2	217		procedures_definition			
31	Définir Bloc 3	227		procedures_definition			



32	Bloc 1	229		procedures_call			
33	Bloc 2	233		procedures_call			
34	Bloc 3	241		procedures_call			
35	Réponse	271		sensing_answer			
36	Touche ... pressée ?	279		sensing_keypressed	F	KEY_OPTION	space
37	Demander ... et attendre	283		sensing_askandwait	QUESTION		
38	... + ...	285		operator_add	NUM1	NUM2	
39	... - ...	295		operation_subtract	NUM1	NUM2	



40	... * ...	299		operator_multiply	NUM1	NUM2	
41	... / ...	301		operator_divide	NUM1	NUM2	
42	... < ...	307		operator_lt	OPERAND1	OPERAND2	
43	... > ...	309		operator_gt	OPERAND1	OPERAND2	
44	... = ...	313		operator_equals	OPERAND1	OPERAND2	
45	... et ...	327		operator_and	OPERAND1	OPERAND2	
46	... ou ...	331		operator_or	OPERAND1	OPERAND2	
47	... non ...	333		operator_not	OPERAND		



48	Nombre aléatoire entre ... et ...	339		operator_random	NUM1	NUM2	
49	Regroupe ... et ...	341		operator_join	STRING1	STRING2	
50	Modulo ...	345		operator_mod	NUM1	NUM2	
51	Arrondi de ...	355		operator_round	NUM		
52	Variable 1	369		data_setvariableto	VALUE		
53	Mettre ... à ...	391		data_changevariableby	VALUE		
54	Montrer la variable ...	395		data_showvariable			
55	Cacher la variable ...	397		data_hidevariable			

Annexe - Structure d'un bloc Scratch

Un programme Scratch (.sb3) est en fait un dossier zippé contenant un fichier au format JSON et des ressources

```
"ID DU BLOC" :  
  {"opcode": "OPCODE DU BLOC",  
   "next": "ID DU BLOC SUIVANT CE BLOC",  
   "parent": "ID DU BLOC PARENT DE CE BLOC",  
   "inputs": {LISTE DES INPUTS DU BLOC},  
   "fields": {LISTE DES FIELDS DU BLOC},  
   "shadow": BOOLEAN,  
   "topLevel": BOOLEAN,  
   "x": ABS,  
   "y": ORD}
```

A propos des formes de blocs

Les formes de chaque bloc sont disponibles ici : https://en.scratch-wiki.info/wiki/Blocks#Block_Shapes

Remplir la partie inputs

La partie **inputs** est une liste construite entre des accolades dont les champs sont séparés par des virgules. {Champ1, Champ2, ..., ChampN}

Les **inputs** d'un bloc sont des entrées qui ne sont pas dans un menu déroulant dans Scratch.



Ces deux blocs montrent les principaux types d'inputs qui existent. Dans le cas du bloc « **if** » on ne peut qu'entrer un bloc (que cela soit en condition ou dans le corps de la partie après le « **then** »), on ne peut ni taper une entrée ni y mettre une variable.

Dans le cas du bloc « **wait** » on peut mettre y mettre un bloc, y entrer des données directement ou y mettre une variable.

Voici comment construire chacun de ces types d'inputs :

- Dans le cas d'un input ne prenant en entrée qu'un bloc (comme le « **if** »), le champ à entrer dans la liste sera :
"NOM DU CHAMP": [TYPE DE L'ENTRÉE, ID DU BLOC]
- Dans le cas d'un input pouvant tout prendre en entrée (comme « **wait** ») le champ à entrer dans la liste sera :
"NOM DU CHAMP": [SHADOW, VALUE]
 - avec **SHADOW** = 1 dans le cas d'une entrée directe

Projet TaBGO –

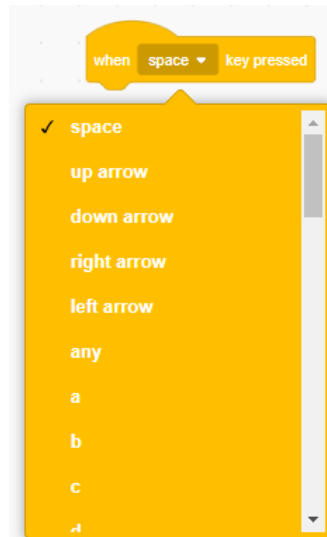
- et `SHADOW = 3` dans le cas d'un bloc ou d'une variable
- et **VALUE** = `[TYPE DE L'ENTRÉE, "ENTRÉE"]` dans le cas d'une entrée directe
- `"ID DU BLOC", [TYPE DE L'ENTRÉE, "ENTRÉE PAR DÉFAUT"]` dans le cas d'un bloc
- `[12, "NOM DE LA VARIABLE", "ID DE LA VARIABLE"], [TYPE DE L'ENTRÉE, "ENTRÉE PAR DÉFAUT"]` dans le cas d'une variable

Remplir la partie fields

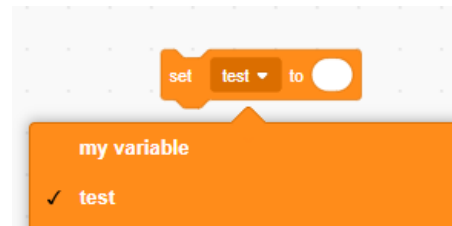
La partie **fields** est une liste construite entre des accolades dont les champs sont séparés par des virgules. {Champ1, Champ2, . . . , ChampN}

Les **fields** d'un bloc sont des entrées qui se trouvent dans un menu déroulant sur le bloc Scratch.

Prenons par exemple les blocs :



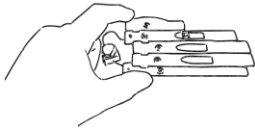
:ou



Ces deux blocs montrent les deux types de **fields** qui existent, le premier a une liste définie et immuable d'options dans lesquelles choisir alors que le deuxième a une liste qui dépend de l'état actuel de notre environnement Scratch.

Voici comment construire un champ de la liste des **fields** d'un bloc :

- Dans le cas du premier type de blocs le champ à entrer sera : `"NOM DU CHAMP": [OPTION, null]`
- Dans le cas du deuxième type de blocs, les seuls que nous avons dans la liste demandée sont ceux qui concernent les variables (à voir s'il y en a d'autres). Pour ces blocs la syntaxe sera : `VARIABLE: [nom de la variable, ID de la variable]`



Cas d'un bloc personnalisé

La création d'un bloc personnalisé ressemble à celle de n'importe quel bloc et a un champ d'input qui a pour syntaxe "custom_block": [1, "ID DU BLOC"] avec ID DU BLOC = l'ID qu'on utilisera pour créer le prototype de ce bloc et le champ **next** sera le premier bloc du corps de ce bloc personnalisé. (l'opcode à utiliser est « **procedures_definition** »).

Quand on crée un bloc personnalisé un bloc prototype est donc créé, ce bloc a pour opcode « **procedures_prototype** » et a une syntaxe un peu différente des autres blocs :

```
"ID DU BLOC" :
  {"opcode":"procedures_prototype"
  "next":"ID DU BLOC SUIVANT CE BLOC",
  "parent":"ID DU BLOC PARENT DE CE BLOC"
  "inputs":{},
  "fields":{},
  "shadow":BOOLEAN,
  "topLevel":BOOLEAN,
  "x":ABS,
  "y":ORD}
  "mutation":
    {"tagName":"mutation",
    "children":[],
    "proccode":"NOM DU BLOC",
    "argumentids":"[]",
    "argumentnames":"[]",
    "argumentdefaults":"[]",
    "warp":"BOOLEAN"}
```

L'utilisation d'un bloc personnalisé a aussi le champ **mutation** à remplir pour l'utiliser.

```
"ID DU BLOC" :
  {"opcode":"procedures_call",
  "next":"ID DU BLOC SUIVANT CE BLOC",
  "parent":"ID DU BLOC PARENT DE CE BLOC"
  "inputs":{},
  "fields"
  " shadow"BOOLEAN,
  "topLevel":BOOLEAN,
  "x":ABS,
  "y":ORD}
```