# 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 | **Sinon\*** | *91* |  | control\_if\_else | CONDITION | SUBSTACK | SUBSTACK2 |  |
| 9 | **Fin de boucle/ Fin Si** | *93* |  | control\_fin |  |  |  |  |
| 10 | **Stop tout** | *103* |  | control\_stop | F | STOP\_OPTION | all |  |
| 11 | **Montrer** | *107* |  | looks\_show |  |  |  |  |
| 12 | **Cacher** | *109* |  | looks\_hide |  |  |  |  |
| 13 | **Dire … pendant … secondes** | *115* |  | looks\_sayforsecs | MESSAGE | SECS |  |  |
| 14 | **Dire** | *117* |  | looks\_say | MESSAGE |  |  |  |
| 15 | **Effacer tout** | *121* |  | pen\_clear |  |  |  |  |
| 16 | **Stylo en position basse (position d’écriture)** | *143* |  | pen\_penDown |  |  |  |  |
| 17 | **Stylo en position haute (relever le stylo)** | *151* |  | pen\_penUp |  |  |  |  |
| 18 | **Avancer de …** | *155* |  | motion\_movesteps | STEPS |  |  |  |
| 19 | **Tourner de … (sens horaire)** | *157* |  | motion\_turnright | DEGREES |  |  |  |
| 20 | **Tourner de … (sens anti-horaire)** | *167* |  | motion\_turnleft | DEGREES |  |  |  |
| 21 | **S’orienter à …** | *171* |  | motion\_pointindirection | DIRECTION |  |  |  |
| 22 | **Aller à x : … y : …** | *173* |  | motion\_gotoxy | X | Y |  |  |
| 23 | **Ajouter … à x** | *179* |  | motion\_changexby | DX |  |  |  |
| 24 | **Mettre x à :** | *181* |  | motion\_setx | X |  |  |  |
| 25 | **Ajouter … à y** | *185* |  | motion\_changeyby | DY |  |  |  |
| 26 | **Mettre y à :** | *199* |  | motion\_sety | Y |  |  |  |
| 27 | **Abscisse x** | *203* |  | motion\_xposition |  |  |  |  |
| 28 | **Abscisse y** | *205* |  | motion\_yposition |  |  |  |  |
| 29 | **Direction** | *211* |  | motion\_direction |  |  |  |  |
| 30 | **Définir Bloc 1** | *213* |  | procedures\_definition |  |  |  |  |
| 31 | **Définir Bloc 2** | *217* |  | procedures\_definition |  |  |  |  |
| 32 | **Définir Bloc 3** | *227* |  | procedures\_definition |  |  |  |  |
| 33 | **Bloc 1** | *229* |  | procedures\_call |  |  |  |  |
| 34 | **Bloc 2** | *233* |  | procedures\_call |  |  |  |  |
| 35 | **Bloc 3** | *241* |  | procedures\_call |  |  |  |  |
| 36 | **Réponse** | *271* |  | sensing\_answer |  |  |  |  |
| 37 | **Touche … pressée ?** | *279* |  | sensing\_keypressed | F | KEY\_OPTION | space |  |
| 38 | **Demander … et attendre** | *283* |  | sensing\_askandwait | QUESTION |  |  |  |
| 39 | **… + …** | *285* |  | operator\_add | NUM1 | NUM2 |  |  |
| 40 | **… - …** | *295* |  | operation\_substract | NUM1 | NUM2 |  |  |
| 41 | **… \* …** | *299* |  | operator\_multiply | NUM1 | NUM2 |  |  |
| 42 | **… / …** | *301* |  | operator\_divide | NUM1 | NUM2 |  |  |
| 43 | **… < …** | *307* |  | operator\_lt | OPERAND1 | OPERAND2 |  |  |
| 44 | **… > …** | *309* |  | operator\_gt | OPERAND1 | OPERAND2 |  |  |
| 45 | **… = …** | *313* |  | operator\_equals | OPERAND1 | OPERAND2 |  |  |
| 46 | **… et …** | *327* |  | operator\_and | OPERAND1 | OPERAND2 |  |  |
| 47 | **… ou …** | *331* |  | operator\_or | OPERAND1 | OPERAND2 |  |  |
| 48 | **… non …** | *333* |  | operator\_not | OPERAND |  |  |  |
| 49 | **Nombre aléatoire entre … et …** | *339* |  | operator\_random | NUM1 | NUM2 |  |  |
| 50 | **Regroupe … et …** | *341* |  | operator\_join | STRING1 | STRING2 |  |  |
| 51 | **Modulo …** | *345* |  | operator\_mod | NUM1 | NUM2 |  |  |
| 52 | **Arrondi de …** | *355* |  | operator\_round | NUM |  |  |  |
| 53 | **Mettre … à ...** | *369* |  | data\_setvariableto | VALUE | F | VARIABLE | VARIABLE |
| 54 | **Ajouter … à …** | *391* |  | data\_changevariableby | VALUE | F | VARIABLE | VARIABLE |
| 55 | **Montrer la variable …** | *395* |  | data\_showvariable | F | VARIABLE | VARIABLE |  |
| 56 | **Cacher la variable …** | *397* |  | data\_hidevariable | F | VARIABLE | VARIABLE |  |
| 57 | **Variable quart\_de\_tour \*\*** | *403* |  | data\_variable |  |  |  |  |
| 58 | **Variable demi\_tour \*\*\*** | *405* |  | data\_variable |  |  |  |  |
| 59 | **Variable var1** | *357* |  | data\_variable |  |  |  |  |
| 60 | **Variable var2** | *361* |  | data\_variable |  |  |  |  |

# \* Sinon => se rattache au premier « Si … Alors … » non finit et le transforme en « Si … Alors … Sinon ... » \*\* Valeur prédéfinie : quart\_de\_tour = 90 \*\*\* Valeur prédéfinie : demi\_tour = 180

# 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,

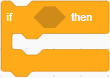
"**x**":ABS,

"**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](https://en.scratch-wiki.info/wiki/Blocks" \l "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.



Prenons par exemple les blocs suivants : ou

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]

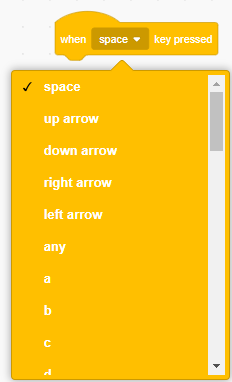
* Dans le cas d’un input pouvant tout prendre en entrée (comme « **wait** ») le champ à entrer dans la liste sera :

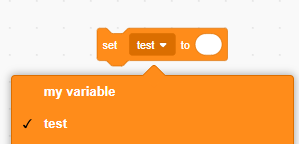
"NOM DU CHAMP":[SHADOW, VALUE]

* + avec **SHADOW** = 1 dans le cas d’une entrée directe
  + 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 ci 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}