TRALALERE

Notions de programmation

Avec exemples propres à DataDecode

DataDecode

Programmer et comprendre **la data**

Bibliothèques Sans Frontières
Libraries Without Borders

V1.0
Auteur : Tralalere

Sommaire

Introduction 3

Les zones de manipulation 4
- La zone de texte 4
- La base de données 5
  - Les mots 6
  - Les images 7
  - Les textes 8
  - Les tags 8
- La zone de code 9

Les variables 10
- Déclarer et modifier une variable 10
- Type de contenu d'une variable 10
- Manipuler une variable 11

Opérations 12

Les comparateurs 13

Les Conditions 13

Règles logiques 16

Les boucles 17
- La boucle for 17
- La boucle while 18

Événement 18
- La validation d'une entrée texte 18
- Le clic sur une image 19
- La lecture d'un mot 20

La recherche 21
- La recherche simple 21
- La recherche filtrée et triée 21

Introduction

Ce document s'adresse aux acteurs éducatifs désireux de faire des ateliers de programmation avec l'application DataDecode aussi bien en langage javascript qu'en blocs.

DataDecode est un éditeur d'exercices permettant la manipulation d'une base de données. On y trouve trois zones de manipulation :

- la zone de texte, dans laquelle on va pouvoir écrire ce que l'on veut ;
- la zone de code, qui va permettre de manipuler et transformer le texte à l'exécution du programme ;
- la base de données, dans laquelle peuvent être stockés des mots, des images et des textes.

DataDecode permet de s'initier au langage objet. Ce document vous aidera à appréhender et comprendre les spécificités de cette application pour ainsi mieux expliquer chaque ligne de code aux élèves.

Prérequis pour pouvoir utiliser ce guide :

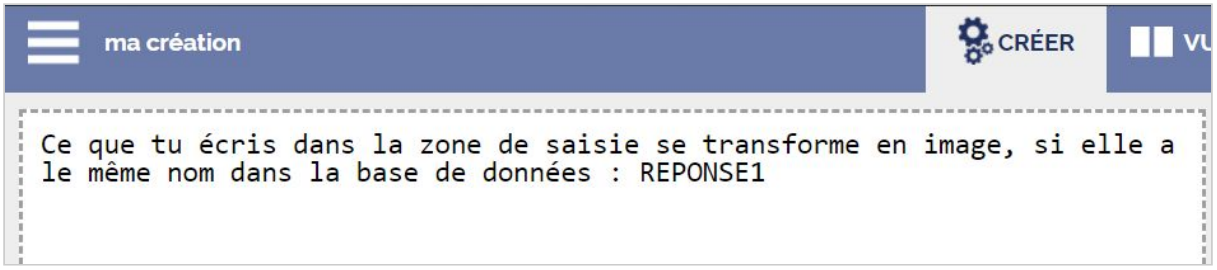
- **Connaître l'application DataDecode.** Nous vous invitons à découvrir l'application au travers du parcours « DataDecode - Tutos ».
- **Avoir des notions en programmation.** Nous vous invitons à consulter le document « Notions générales de programmation » disponible sur le site.

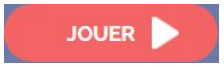
Pour approfondir :

- Découvrez de nombreux exemples de code au travers des documents « DataDecode - blocs : exemples de code » et « DataDecode - script : exemples de code » disponibles sur le site. Ces documents vous donneront des idées de comportements pour aider les élèves à concevoir leurs créations.

Les zones de manipulation

La zone de texte



Dans cette zone, on peut écrire un texte qui sera transformé au moment de l'exécution du script 

.

Cette zone est capable d'interpréter certains mots-clés :

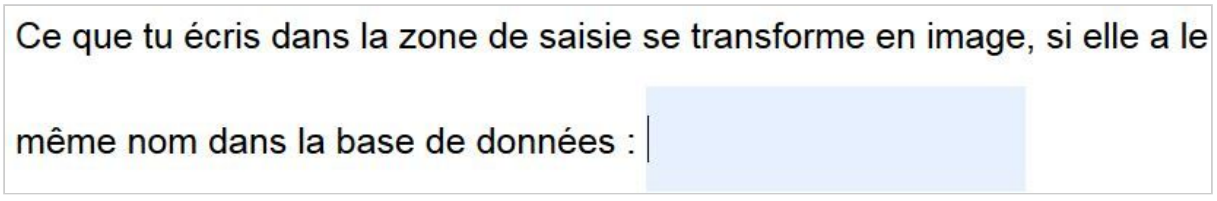
- REPONSEX ;
- POSITIONX.

Le mot-clé REPONSEX, dans lequel X est un chiffre, permet de symboliser l'emplacement d'une zone de saisie.

Ainsi,

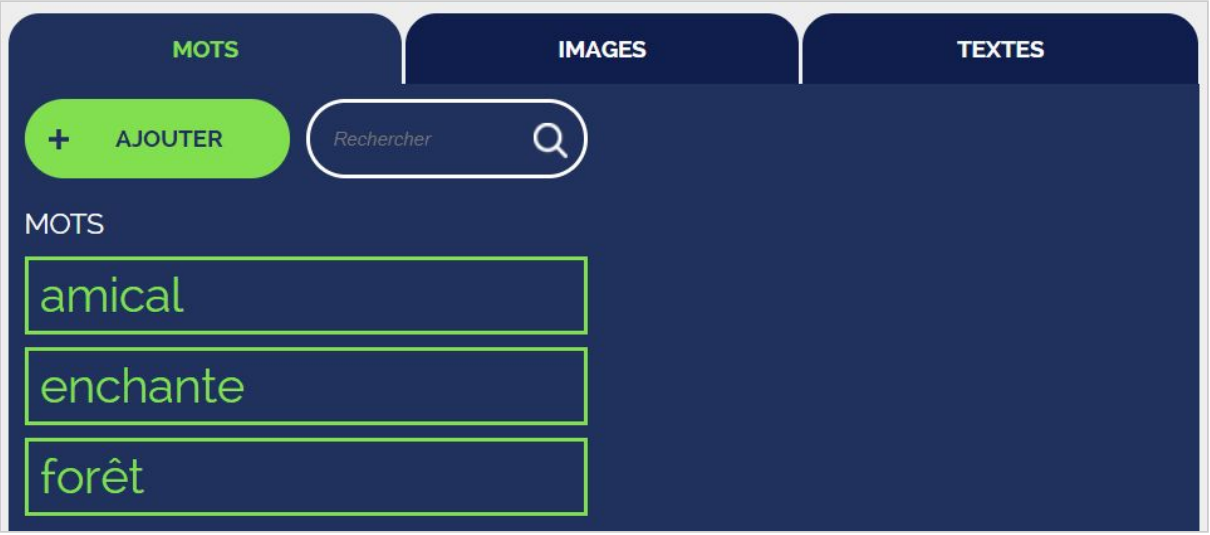
```
Ce que tu écris dans la zone de saisie se transforme
en image, si elle a le même nom dans la base de
données : REPONSE1
```

À l'exécution devient :



De la même façon, le mot-clé POSITIONX sera remplacé à l'exécution. Il s'agit d'une ancre dont on se sert pour positionner une donnée que l'on ajoute avec le code.

La base de données



La base de données est un espace de stockage propre à chaque création. Elle est découpée en trois collections :

- une collection de mots ;
- une collection d'images ;
- une collection de textes.

Pour chercher un élément de cette base de données, on se sert du code. On appelle cela effectuer une requête.

Les mots

Les mots stockés dans la base sont caractérisés par leur nature et des tags :

- La nature permet de différencier les mots selon leur fonction grammaticale : nom, verbe, adjectif, connecteur.
- Le tag permet de lier plusieurs mots en leur attribuant la même étiquette. On peut aussi associer plusieurs tags à un même mot pour opérer une classification précise.

Grâce à ces indicateurs, on pourra faire une recherche dans cette collection. Exemples de requêtes possibles :

- Récupérer tous les mots dont la nature est « sujet ».
- Récupérer tous les mots qui possèdent le tag « fantastique ».

Les images

Les images stockés dans la base sont caractérisées par leur nom, le fichier image et des tags :

- On peut rechercher une image directement par son nom.
- Le système de tags est le même que pour les mots.

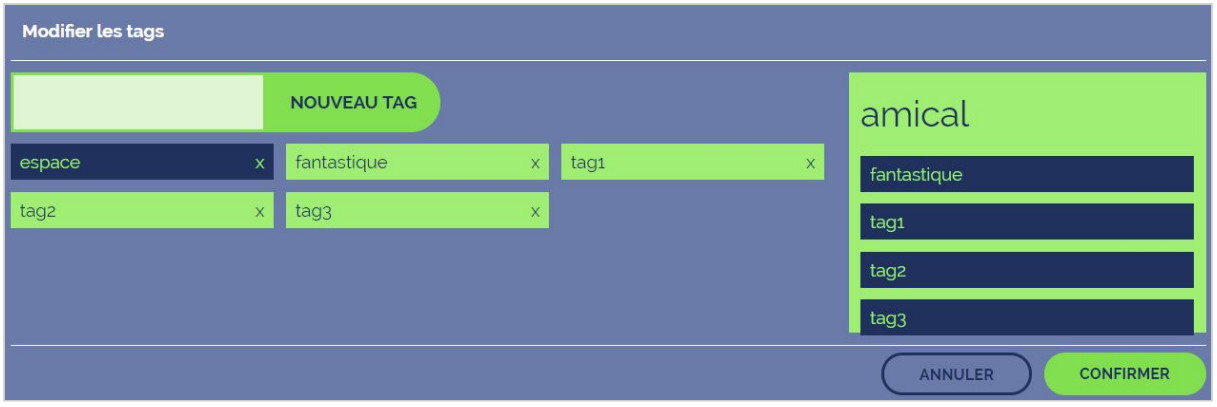
Les textes

Les textes stockés dans la base sont caractérisés par un titre, un texte et des tags :

- On peut rechercher un texte directement par son titre.
- Le système de tags est le même que pour les mots.

Les tags

La collection de tags apparaît lorsque l'on clique sur le bouton « gestion des tags », après avoir sélectionné un élément de la base de données.



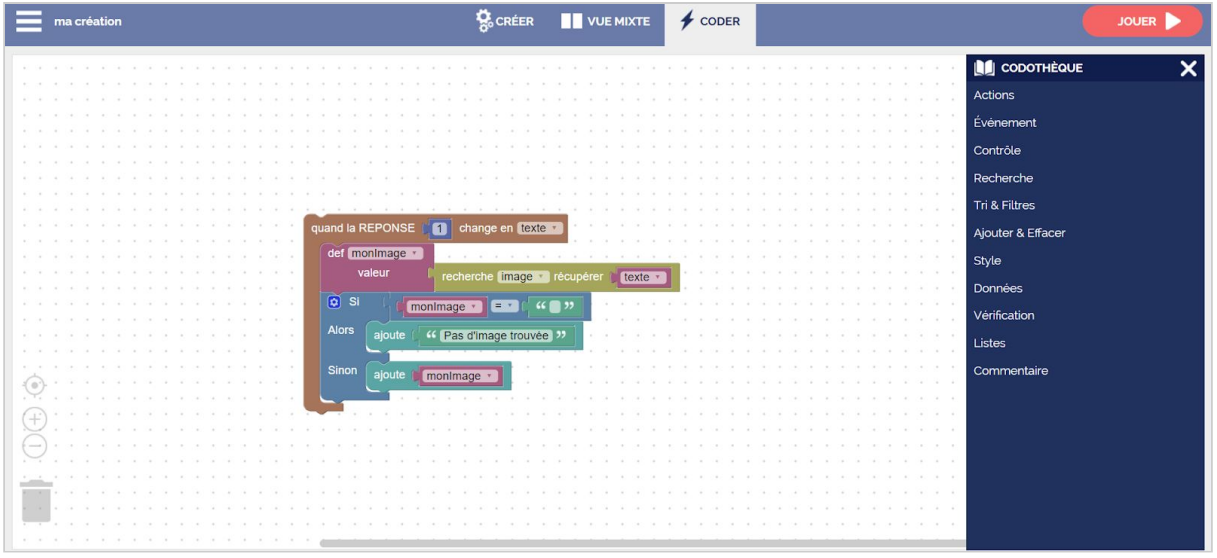
Dans le compartiment de droite, on observe l'élément sélectionné et les tags associés.
À gauche, on retrouve l'ensemble de tous les tags. On distingue :

- ceux sur fond vert, qui sont associés à l'élément sélectionné ;
- ceux sur fond bleu, qui sont disponibles mais pas associés.

On peut :

- ajouter de nouveaux tags : saisie puis clic sur « nouveau tag » ;
- supprimer des tags : clic sur la croix du tag à supprimer ;
- sélectionner ou désélectionner les tags associés à l'élément de la base de données : clic sur le tag, ce qui a pour effet de l'ajouter à la liste de l'élément s'il en était absent ou de le retirer s'il était présent.

La zone de code



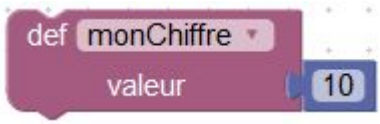
La zone de code est la zone qui va permettre de construire le programme pour transformer le texte. Disponibles avec Blockly ou du javascript, toutes les briques de code utilisables sont regroupées dans la « Codothèque ».

Les variables

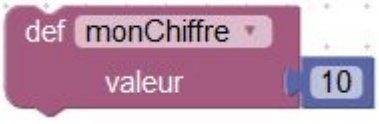
Une variable est un contenant, dans lequel on peut stocker une valeur. La valeur contenue dans une variable peut changer au cours de l'exécution du programme.

Déclarer et modifier une variable

Avec Blockly, pour déclarer et initialiser une variable, on utilise le bloc **def**.
En javascript, on déclare une variable à l'aide du mot-clé **var** ; à ce moment, on peut déjà lui attribuer une valeur à l'aide du signe **=**.

Bloc	Script
	`var monChiffre = 10;`

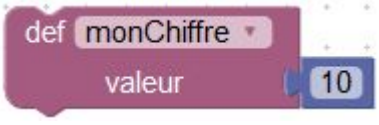
Pour actualiser la valeur de la variable avec Blockly, on utilise le même bloc de définition, alors qu'en script le mot-clé **var** n'est plus nécessaire.

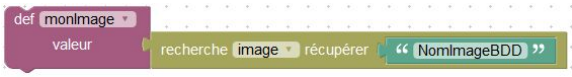
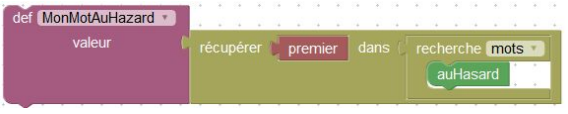
Bloc	Script
	`monChiffre = 10;`
	`monChiffre = monChiffre + 2;`

Type de contenu d'une variable

Dans DataDecode, la variable peut contenir différents types de données : nombre, texte, élément de la base de données.

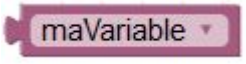
Exemples :

Bloc	Script
	`var monChiffre = 10;`

	var monTexte = 'ceci est un texte';
	var monImage = recherche.image('NomImageBDD');
	var monTexte = recherche.texte('NomTexteBDD');
	var MonMotAuHasard = recherche.mots.auHasard.premier;

Manipuler une variable

Pour pouvoir tester ou utiliser cette variable, on utilise son nom :

Bloc	Script
	maVariable

Exemple 1 :

Bloc	Script

	si (monImage == '')
	{
	ajoute('Pas d\'image trouvée');
	}

Dans cet exemple on teste si la variable contient un texte vide car c'est ce qu'une recherche renvoie lorsqu'elle ne trouve aucun résultat.
Ici, si aucune image n'a été trouvée, on ajoute le texte « pas d'image trouvée » à l'écran.

Exemple 2 :

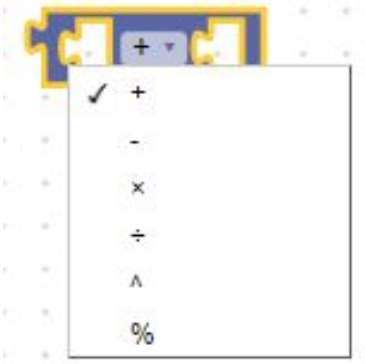
Bloc	Script
	ajoute(monImage)

Ici, on ajoute à l'écran lors de l'exécution du programme, le contenu de la variable « **monImage** » : si elle contient une image, elle sera alors affichée.

Opérations

En code, nous trouvons les opérateurs suivants :

- « **+** » : addition ;
- « **-** » : soustraction ;
- « ***** » : multiplication ;
- « **/** » : division ;
- « **^** » : exposant. Il permet d'élever un nombre à une puissance ;
- « **%** » : « modulo ». Il permet d'obtenir le reste d'une division entière.



Exemples :
3 / 2 = 1,5
6^3 = 6*6*6 = 216
17%7 = 3 car 17 divisé par 7 donne 2 et il reste 3
15%5 = 0 car 15 divisé par 5 donne 3 et il reste 0

Pour modifier une variable par le calcul, il existe quelques raccourcis :

Exemple :

```
3 / 2 = 1,5
3 % 2 = 1
4 % 2 = 0
```

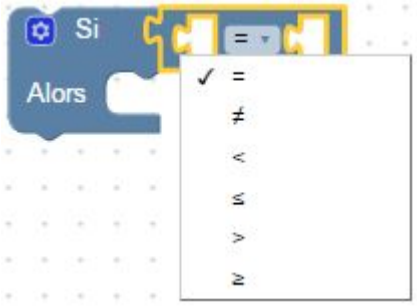
Pour modifier une variable par le calcul il existe, en script, quelques raccourcis :

Exemples :

```
i = i + 1 peut également s'écrire i += 1 ou encore i++
i = i + 10 peut également s'écrire i += 10
```

Les comparateurs

Pour comparer des valeurs, nous utilisons les signes mathématiques suivants :




```
==    /* renvoie vrai si les deux éléments comparés sont égaux */

!=    /*renvoie vrai si les deux éléments comparés sont différents */

<     /* renvoie vrai si la valeur de l'élément de gauche est strictement inférieure à celle de l'élément de droite */

<=    /* renvoie vrai si la valeur de l'élément de gauche est inférieure ou égale à celle de l'élément de droite */

>     /* renvoie vrai si la valeur de l'élément de gauche est strictement supérieure à celle de l'élément de droite */

>=    /* renvoie vrai si la valeur de l'élément de gauche est supérieure ou égale à celle de l'élément de droite */
```

En javascript, on peut aussi observer des comparaisons utilisant le signe « **===** ».
Ce test compare les valeurs de deux éléments mais aussi leur type.

Exemple :

```
(i === 0) /* Cette écriture compare les valeurs ainsi que le type */
```

En programmation, le type permet de différencier les variables. Par exemple **var test = 0** et **var test = "0"** n'ont pas le même type : la première est un chiffre et la deuxième est un texte, ou chaîne de caractères.

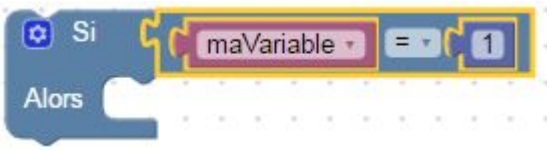
Les Conditions

Une condition permet de définir une tâche à effectuer lorsqu'une hypothèse est vérifiée.

À l'aide des comparateurs, les conditions permettent d'écrire du code qui sera mis dans un bloc SI/ALORS. SI la condition située après le terme « si » est vraie, ALORS le code contenu dans la partie « alors » du bloc sera exécuté. En javascript, cette partie correspond à celle encadrée par les accolades « **{ }** ».

Exemple 1 :

Bloc	Script

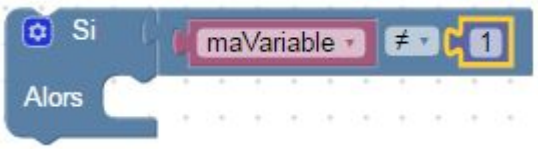
	si (maVariable == 1)
	{
	
	}

Cette condition évalue l'égalité entre « maVariable » et la valeur 1. Il y a deux cas de figure :

- « maVariable » vaut 1. Cette comparaison renvoie la valeur « vrai » (true). Le code contenu après ALORS est donc exécuté.
- « maVariable » ne vaut pas 1. Cette comparaison renvoie la valeur « faux » (false). Le reste du bloc « si » est ignoré, le code contenu après ALORS n'est donc pas exécuté.

Exemple 2 :

Bloc	Script

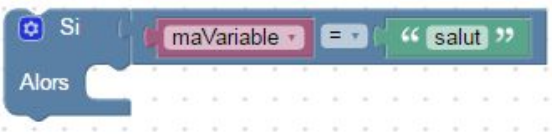
	si (maVariable != 1)
	{
	
	}

Cette condition évalue l'inégalité entre « maVariable » et la valeur 1. Il y a deux cas de figure :

- « maVariable » ne vaut pas 1. Cette comparaison est vraie, « maVariable » est différente de 1. La comparaison renvoie la valeur « vrai » (true). Le code contenu après ALORS est donc exécuté.
- « maVariable » vaut 1. Cette comparaison renvoie la valeur « faux » (false). Le reste du bloc « si » est ignoré, le code contenu après ALORS n'est donc pas exécuté.

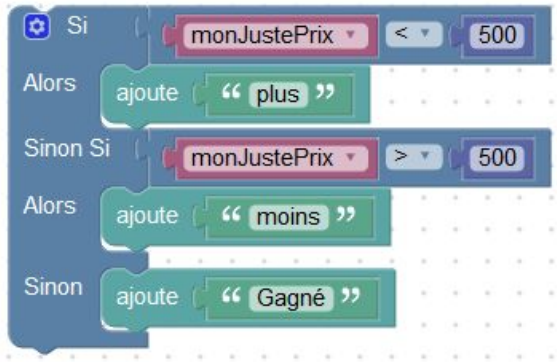
Autres exemples :

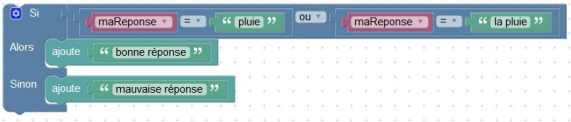
Bloc	Script

	si (maVariable == 'salut')
	{
	
	}
	
	/*Compare « maVariable » à la chaîne de

	caractère « salut »*/

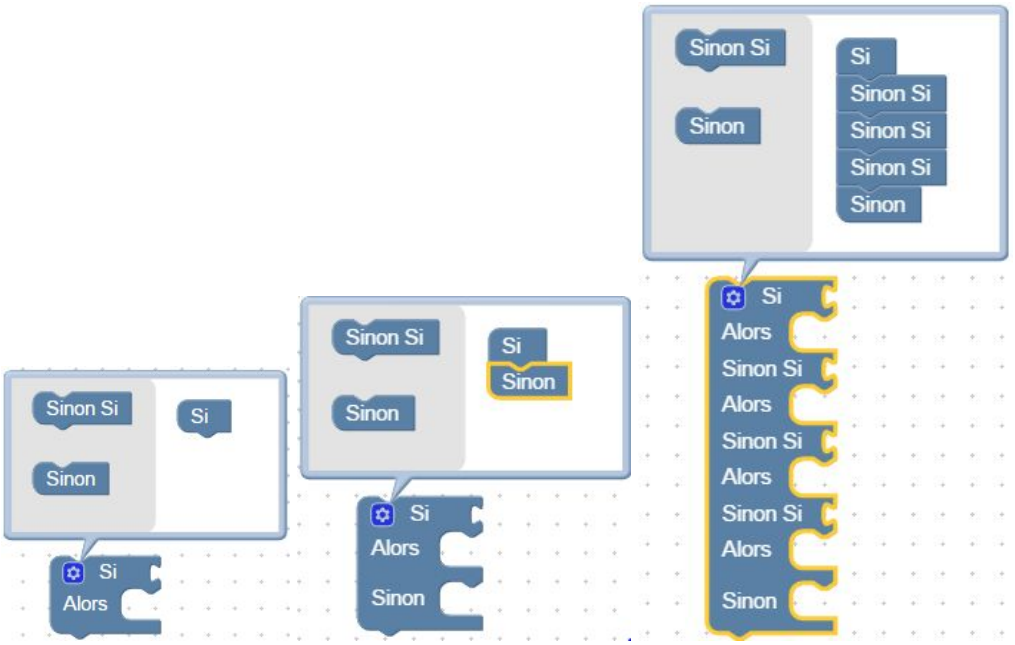
	si (premierChoix == ‘spaghettis’)
	{
	ajoute(‘tu aimes les pâtes’);
	}

	si (monJustePrix < 500)
	{
	ajoute(‘Plus’);
	}
	sinon si (monJustePrix > 500)
	{
	ajoute(‘Moins’);
	}
	sinon
	{
	ajoute(‘Gagné’);
	}

	si (maReponse == ‘pluie’ ou maReponse == ‘la pluie’)
	{
	ajoute(‘Bonne réponse’);
	}
	sinon
	{
	ajoute(‘Mauvaise réponse’);
	}

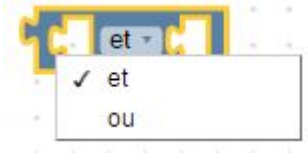
Comme indiqué dans l’exemple, il est possible d’utiliser des comparateurs et règles logiques dans les conditions.

Dans Blockly, on utilise la roue dentée présente en haut à gauche du bloc « si » pour modifier la condition :



Règles logiques

Les symboles logiques ET et OU permettent de donner un caractère inclusif ou exclusif à une comparaison.

Bloc	Script
	`&&` /* et */ `||` /* ou */

Exemples :

Bloc	Script

	si ((maVariable >= 0) && (maVariable <= 5))
	{
	
	}

Cette condition est vérifiée si la valeur de « maVariable » est à la fois supérieure ou égale à 0, et inférieure ou égale à 5. En d'autres termes, cette condition est vraie si la valeur de « maVariable » est comprise entre 0 et 5.

Bloc	Script

	si (maVariable <= 0) || (maVariable >= 5)
	{
	
	}

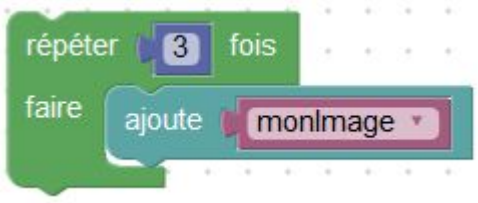
Les deux éléments de cette vérification sont maintenant traités séparément et le résultat, ici, sera vrai si la valeur de « maVariable » est inférieure ou égale à 0 OU si la valeur de « maVariable » est supérieure ou égale à 5.

Si jamais on se trompe et que l'on met ET, le code présent dans la condition ne sera jamais exécuté, car il faudrait que « maVariable » soit à la fois plus grande que 5 et plus petite que 0. Il faut donc bien faire attention aux conditions que l'on utilise, et au choix entre ET et OU.

Les boucles

Une boucle est une instruction permettant de répéter du code plusieurs fois de suite.

Dans DataDecode, il est possible d'utiliser le bloc « répéter _ fois ».

Bloc


En script, il existe deux équivalents : la boucle « for » et la boucle « while ».

La boucle for

Script

for (var index = 0; index < 3; index = index + 1)
{
    ajoute(monImage);
}

La boucle while

Script

```
var index = 0;
while (index < 3)
{
    ajoute(monImage);
    index = index + 1;
}
```

Ces deux boucles sont configurées de la même façon et ont le même effet.
On utilise la variable **index** comme compteur, que l'on initialise à 0.
On définit comme condition de sortie de boucle d'atteindre la valeur 3. Ainsi, tant que la variable **index** est inférieur à 3, le code présent dans la boucle est exécuté.
On exécute le code et on incrémente la valeur de la variable **index**.

Événement

La programmation d'événements permet de déclencher l'exécution d'un bout de code uniquement lorsque l'événement se produit.

Ce code ne s'exécute pas tant que le déclencheur n'est pas atteint. Une fois que ce dernier est atteint, les actions définies dans l'événement se réalisent.

Dans DataDecode on compte trois catégories d'événements :

- la détection de la validation d'une entrée texte ;
- la détection du clic sur une image ;
- la détection d'un mot à la lecture.

La validation d'une entrée texte

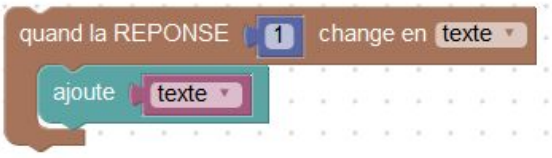
Pour créer une entrée texte, on utilise le mot-clé REPONSEX, dans lequel X est un chiffre, ce qui permet de définir plusieurs entrées texte en fonction du besoin.

```
écrit ici ta réponse :
REPONSE1
```

Lors de l'exécution du programme, ce mot-clé devient :

Avec le code suivant, on peut récupérer le contenu de la zone de saisie lorsque l'utilisateur appuie sur la touche « entrée » :

Bloc	Script

	quand('REPONSE1 change', fonction (text)
	{
	ajoute(text);
	})

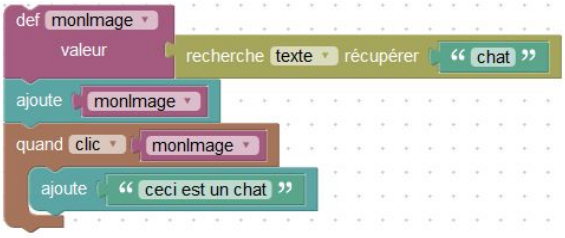
Dans cet exemple, le texte écrit par l'utilisateur sera affiché à l'écran.

Le clic sur une image

La détection du clic de l'utilisateur se fait à l'aide du bloc « quand clic ».

Exemple :

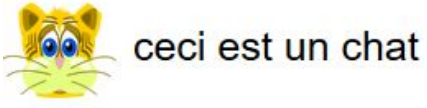
Bloc	Script

	var monImage = recherche.image('chat');
	ajoute(monImage);
	quand('clic', monImage, fonction ()
	{
	ajoute('ceci est un chat');
	})

Dans l'exemple ci-dessus, on affiche l'image dont le nom est « chat » dans la base de données. Quand on détecte que l'utilisateur clique sur cette image, on affiche le texte « ceci est un chat ».

À l'exécution du programme, on obtient dans la zone de rendu :

Et après le clic de l'utilisateur, on obtient :



La lecture d'un mot

Avec la fonctionnalité de lecture, il est possible de détecter un mot dans un texte et d'exécuter un comportement lorsque ce mot est « lu » par le programme.

Pour cela, nous avons besoin :

- d'un texte ;

```
Dans mon histoire, il y a un pirate, un trésor, et du soleil
```

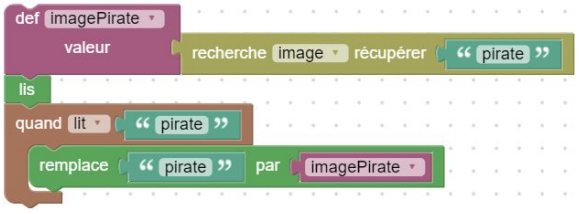
- de la fonction de lecture ;

Bloc	Script
	lis();

- de l'événement « quand lit ».

Exemple :

Bloc	Script

	var imagePirate = recherche.image('pirate');
	lis();
	quand('lit', 'pirate', fonction ()
	{
	remplace('pirate', imagePirate);
	})

Dans l'exemple ci-dessus, « imagePirate » est une variable dans laquelle on stocke une image. À la lecture du mot « pirate », on remplace le mot par l'image.

On obtient dans la zone de rendu :

Dans mon histoire, il y a un pirate, un trésor, et du soleil

La tête de lecture parcourt la phrase. Ici, elle est sur le mot « il ».

Dans mon histoire, il y a un  , un trésor, et du soleil

Une fois que la tête de lecture passe sur le mot « pirate », celui-ci est remplacé par l'image de pirate.

La recherche

Dans DataDecode, il est possible d'aller chercher, grâce au code, des informations contenues dans la base de données. C'est ce que l'on appelle effectuer une requête.

Deux types de requêtes peuvent être émises :

- une requête simple, par nom, qui permet d'aller chercher un élément de la base de données ;
- une requête avec des options de tri et de filtres permettant de sortir un ou plusieurs résultats.

La recherche simple

La recherche simple se fait avec un seul bloc de recherche qui permet de trouver un élément par son nom.

Bloc	Script
	`var imagePirate = recherche.image('pirate');`
	`var textePirate = recherche.texte('pirate');`

Ici, on recherche une image ou un texte dont le titre est « pirate ». Le résultat est stocké dans une variable pour être réutilisé par la suite.

La recherche filtrée et triée

La recherche filtrée et triée permet de récupérer un à plusieurs éléments, organisés en fonction des options de recherche. Dans DataDecode, cela permet de récupérer un ou plusieurs résultats, au hasard ou en fonction de leurs tags.

Pour effectuer une recherche filtrée nous avons besoin de deux blocs :

- Le bloc de recherche avec option de tri.

Ce bloc permet à la fois de sélectionner la collection dans laquelle chercher et de déterminer des options de tri. Le résultat est un objet contenant la collection de mots mélangés et n'est pas encore utilisable en cet état.
Les options de tri ne sont pas nécessaires pour utiliser ce bloc.

- Le bloc de récupération et filtre de recherche.

Ce bloc filtre la collection contenue dans l'objet renvoyé par le bloc de recherche. C'est le résultat renvoyé par ce bloc qui sera exploitable. Le filtre est nécessaire pour utiliser ce bloc afin de déterminer le nombre de résultats à renvoyer, et cela même si la recherche ne concerne qu'un seul élément.

On les assemble comme ceci :

Bloc	Script
	recherche.mots.auHasard.premier;

Dans cet exemple, on récupère le premier résultat de la collection de mots qui a été mélangée.