

遊びで学んで
デジタルコンテンツ
の裏側を見てみよう



まずは遊んで見よう！

準備する事

1. プログラムの準備 「ゲームファイルをダウンロード」
2. コントローラの準備 1 「micro:bitとPCをUSBケーブルでつなぐ」
3. コントローラの準備 2 「micro:bitにプログラムを入れる」
4. Gameの開始 「Gameファイルをクリックする」

プログラムの準備

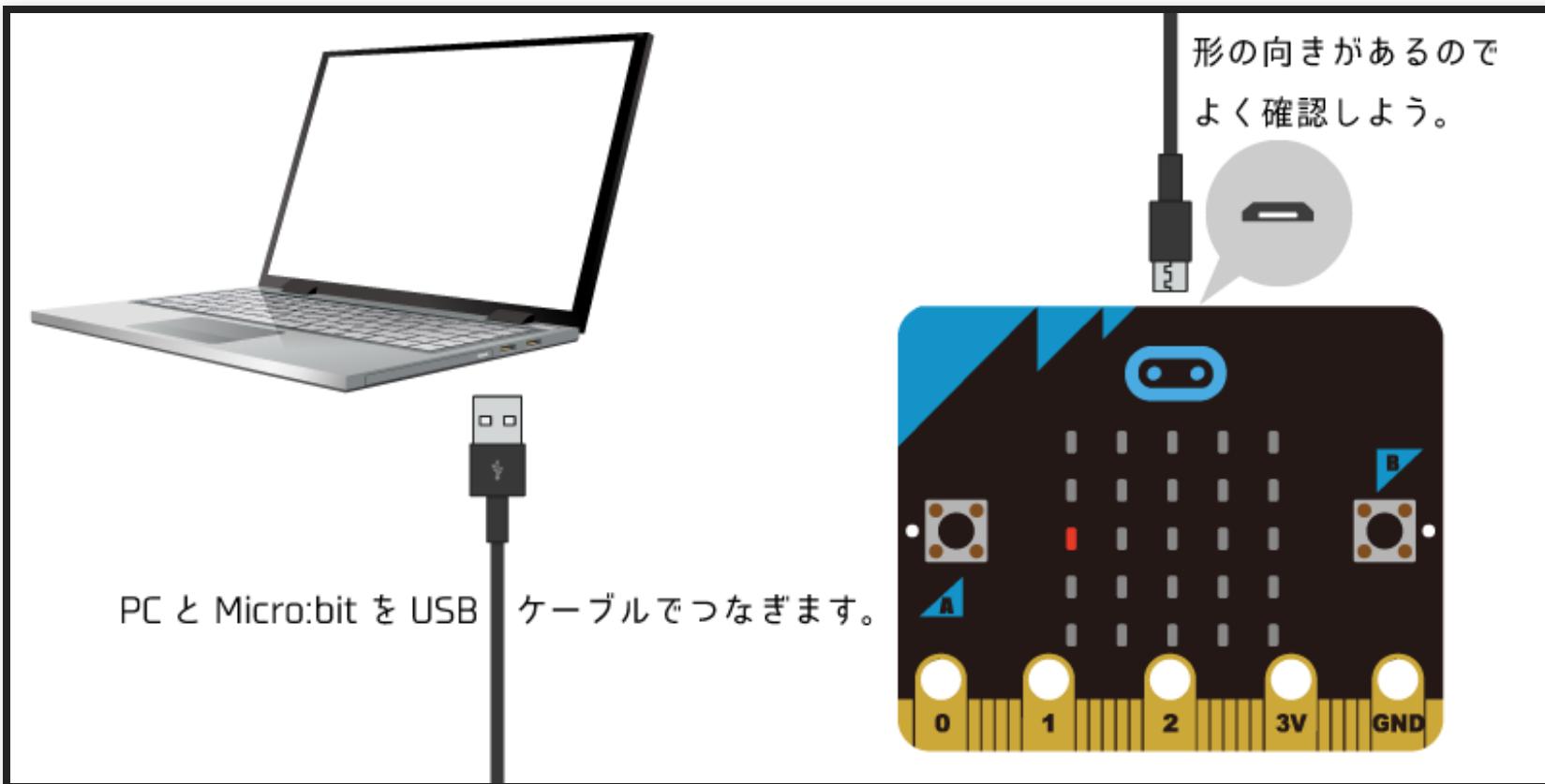
ファイルのダウンロードお使いのPCのOSに合わせてダウンロードしてください。

下にある「ダウンロード」をWindowsかMacを選んでクリック

Windows版: [ダウンロード](#)

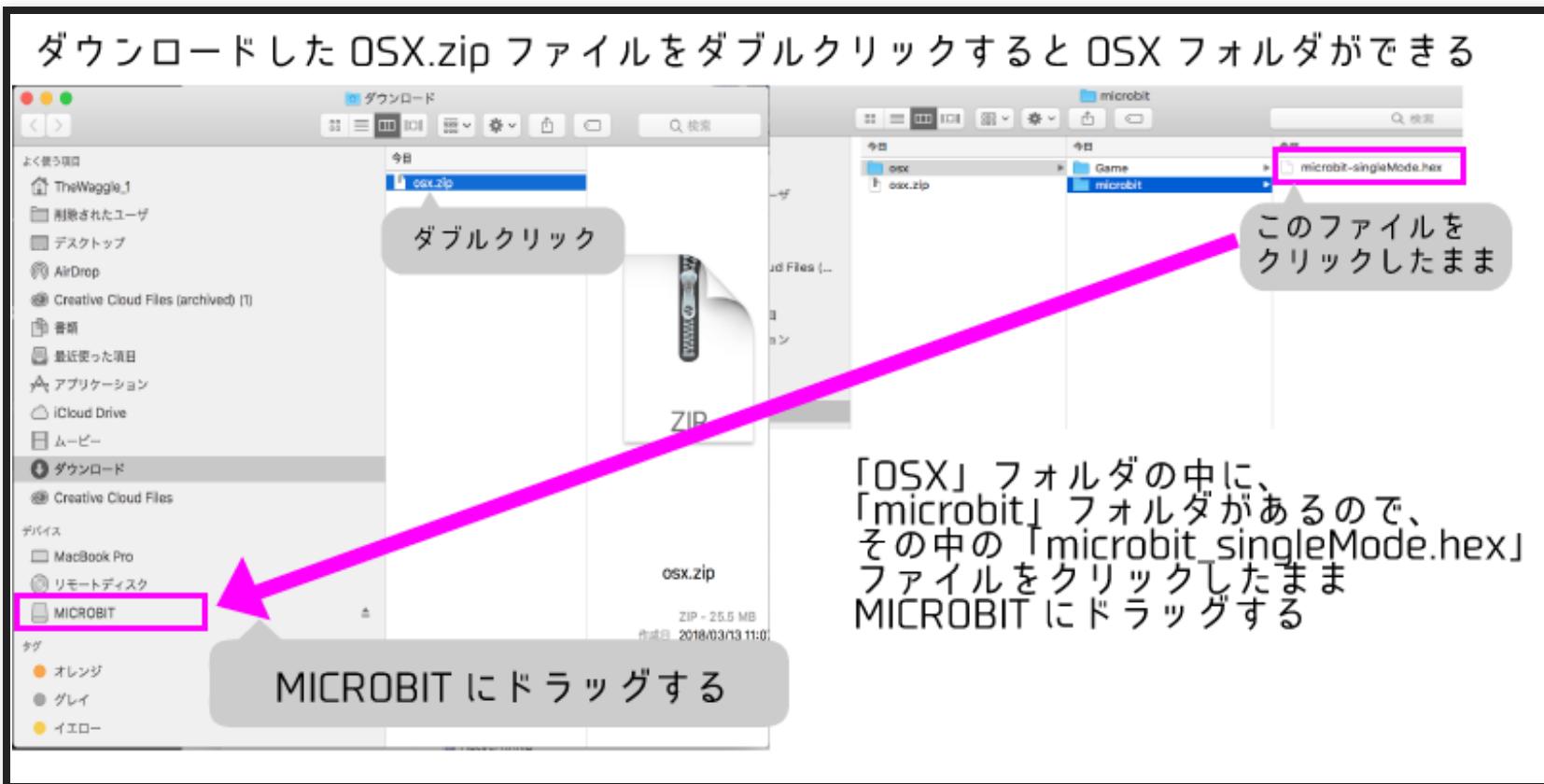
Mac版: [ダウンロード](#)

コントローラの準備 1 [MICROBITの準備]



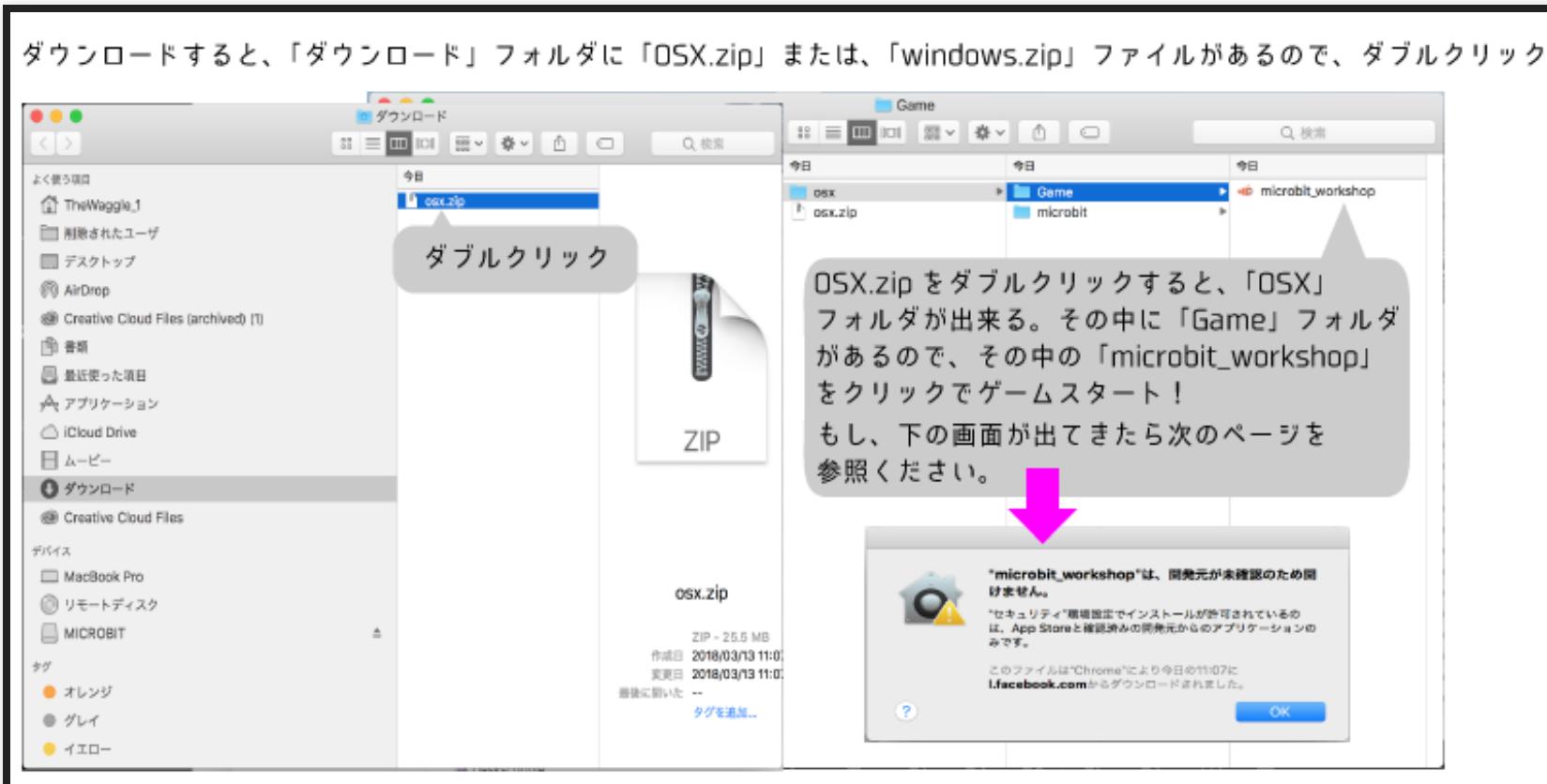
コントローラの準備2

Microbitにプログラムを入れる



GAMEの開始

ファイルの場所(画面はMac版)



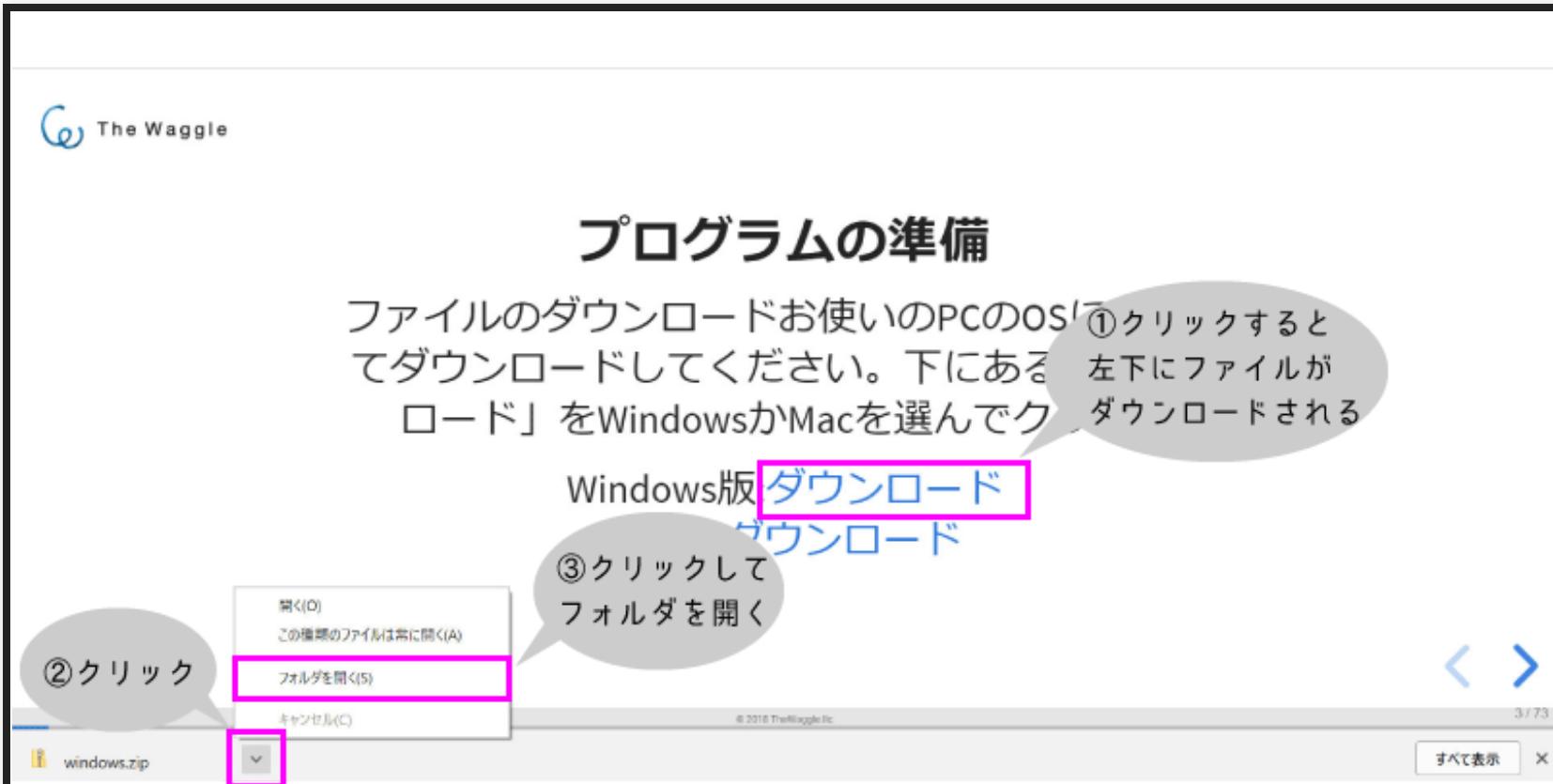
セキュリティメッセージが出た場合の回避方法 (Mac版)

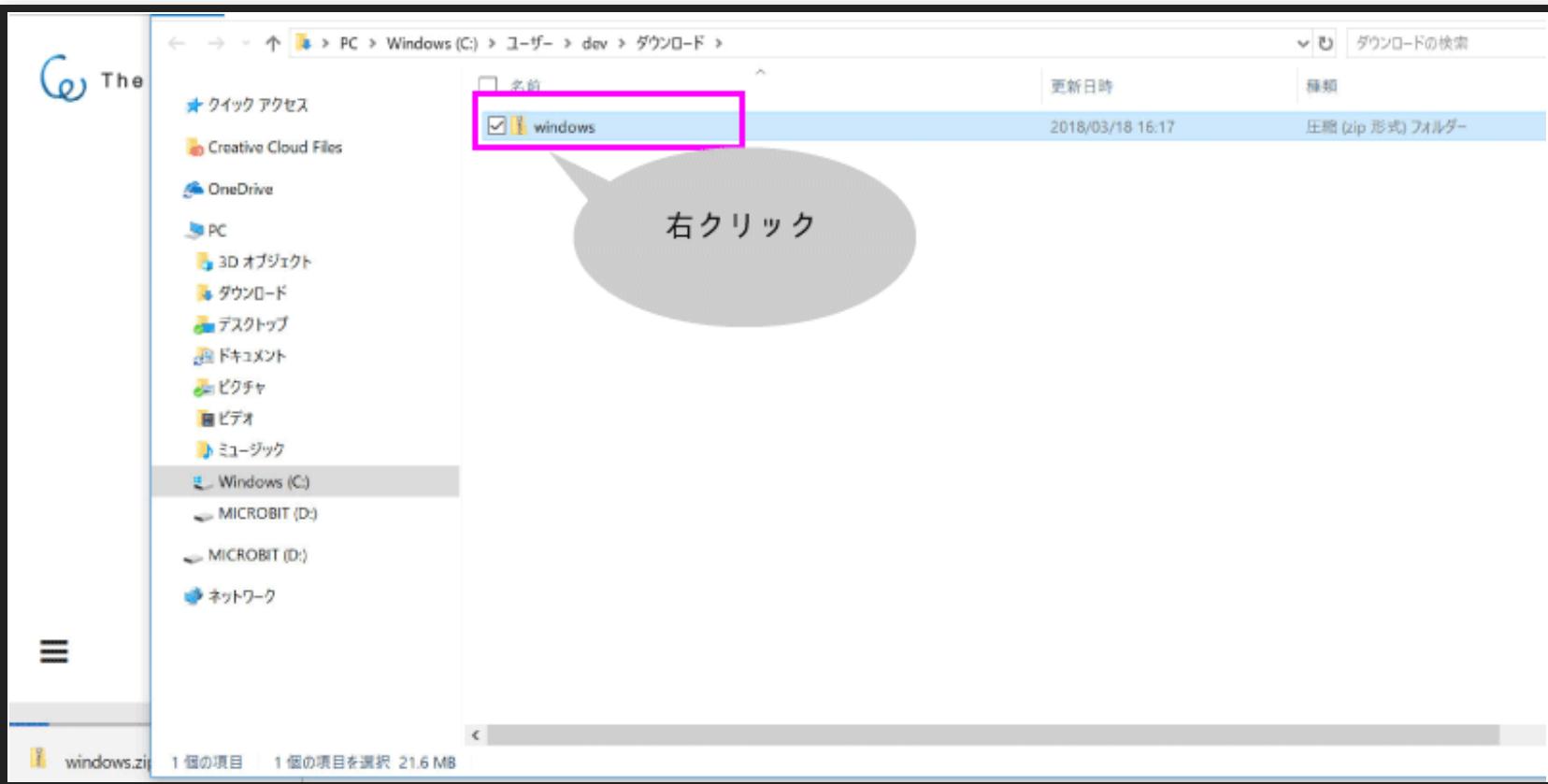


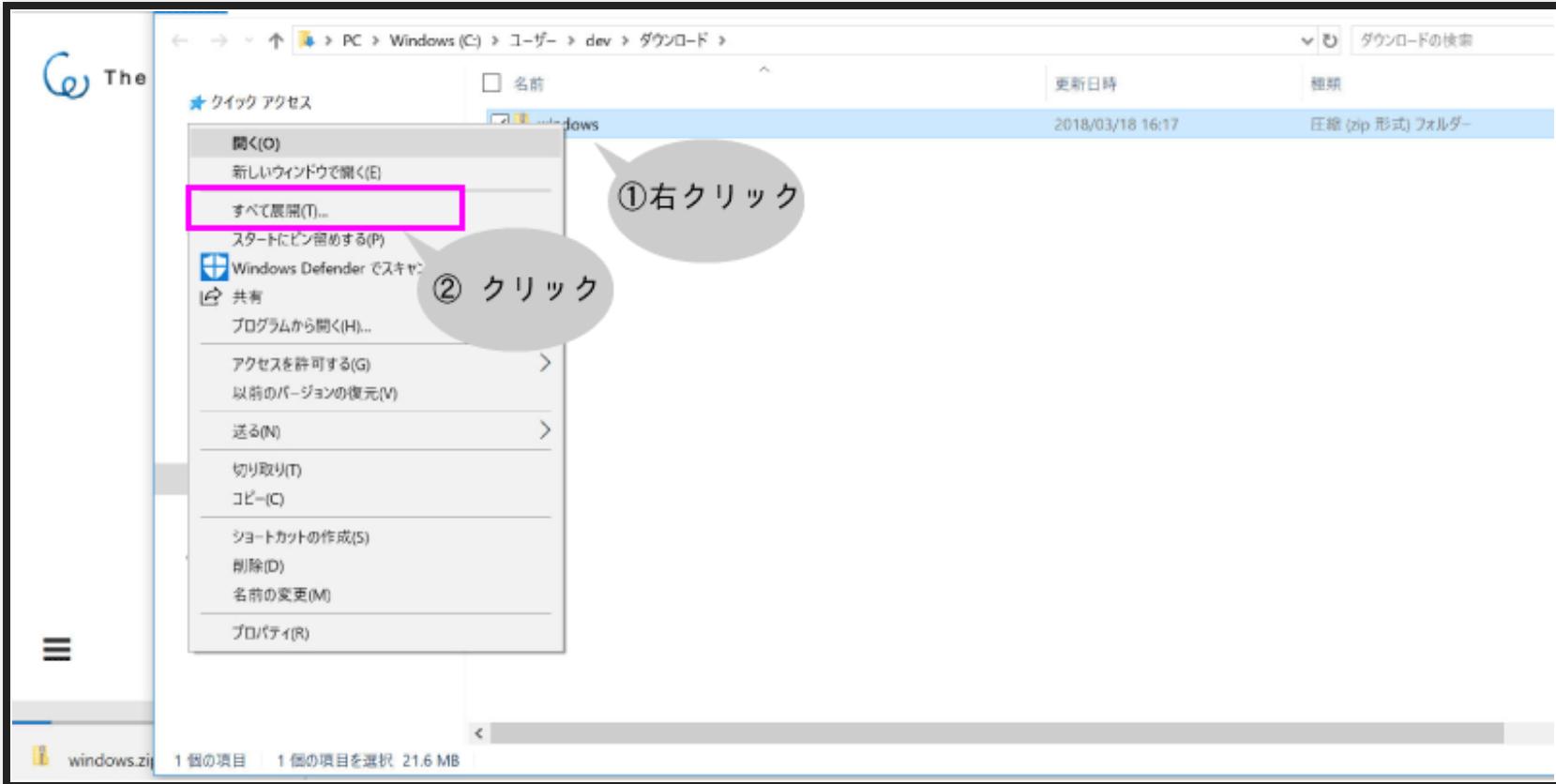
Game画面スタート(Mac版)

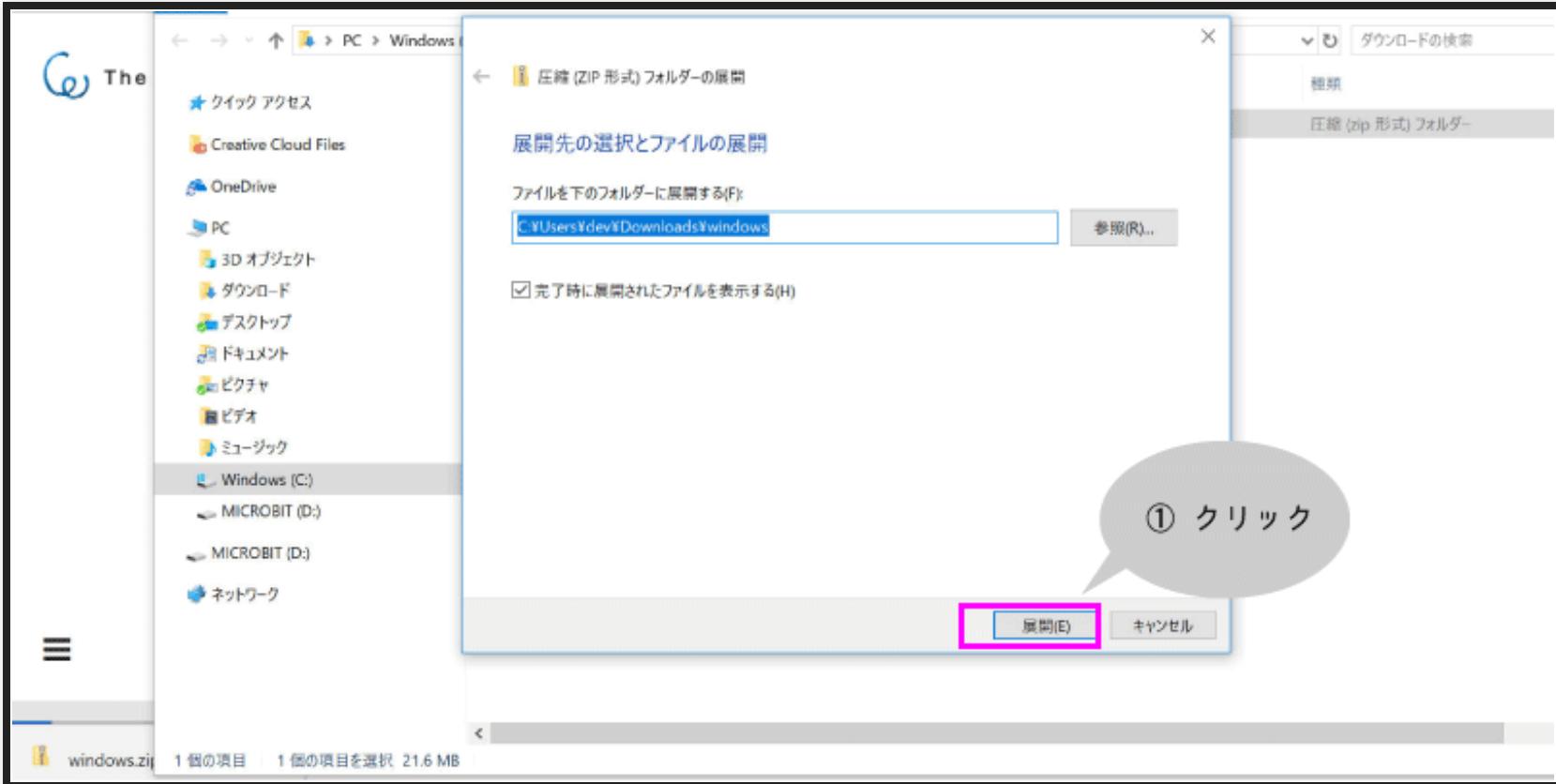


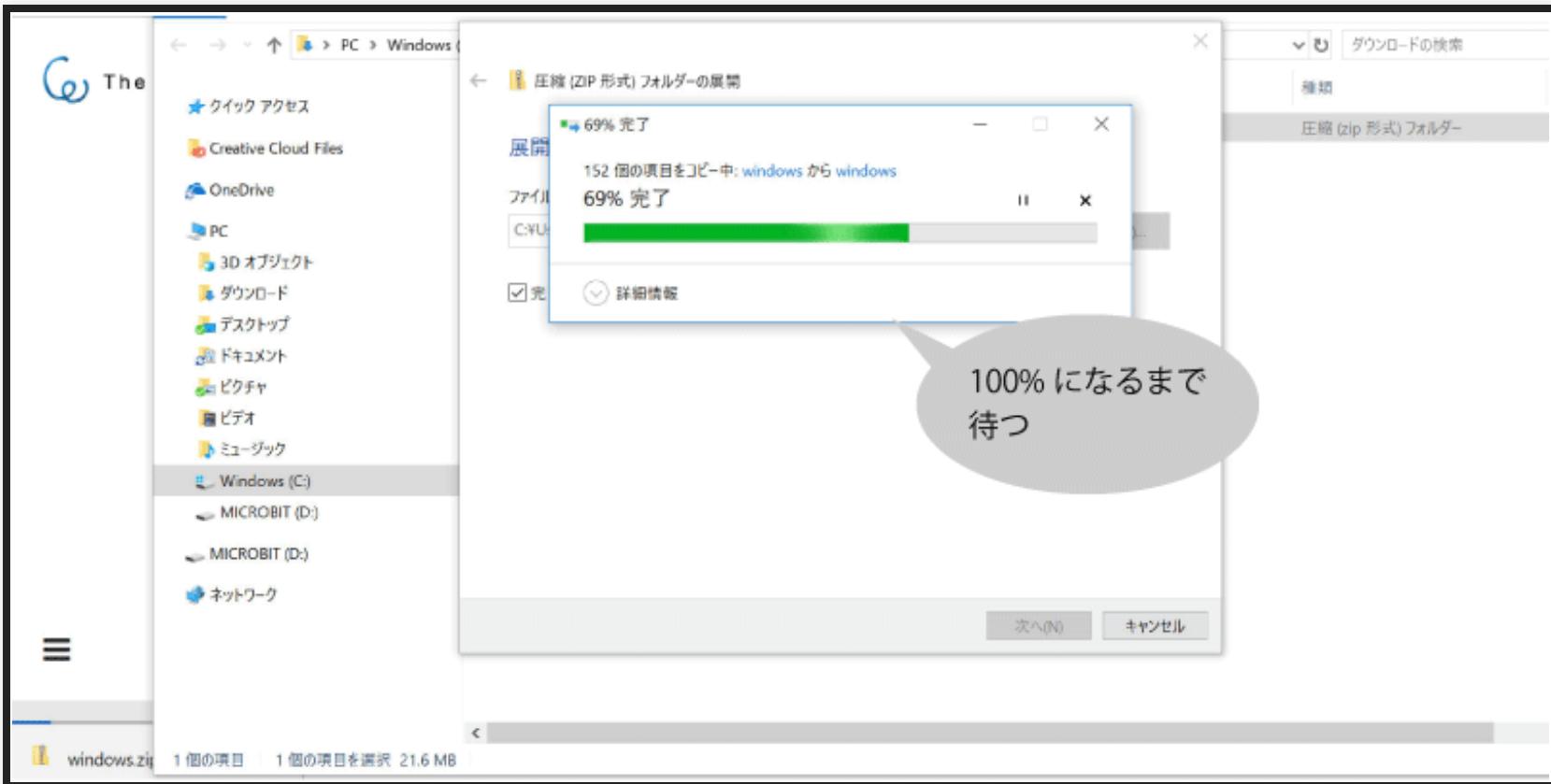
コントローラの準備2(WINDOWS)

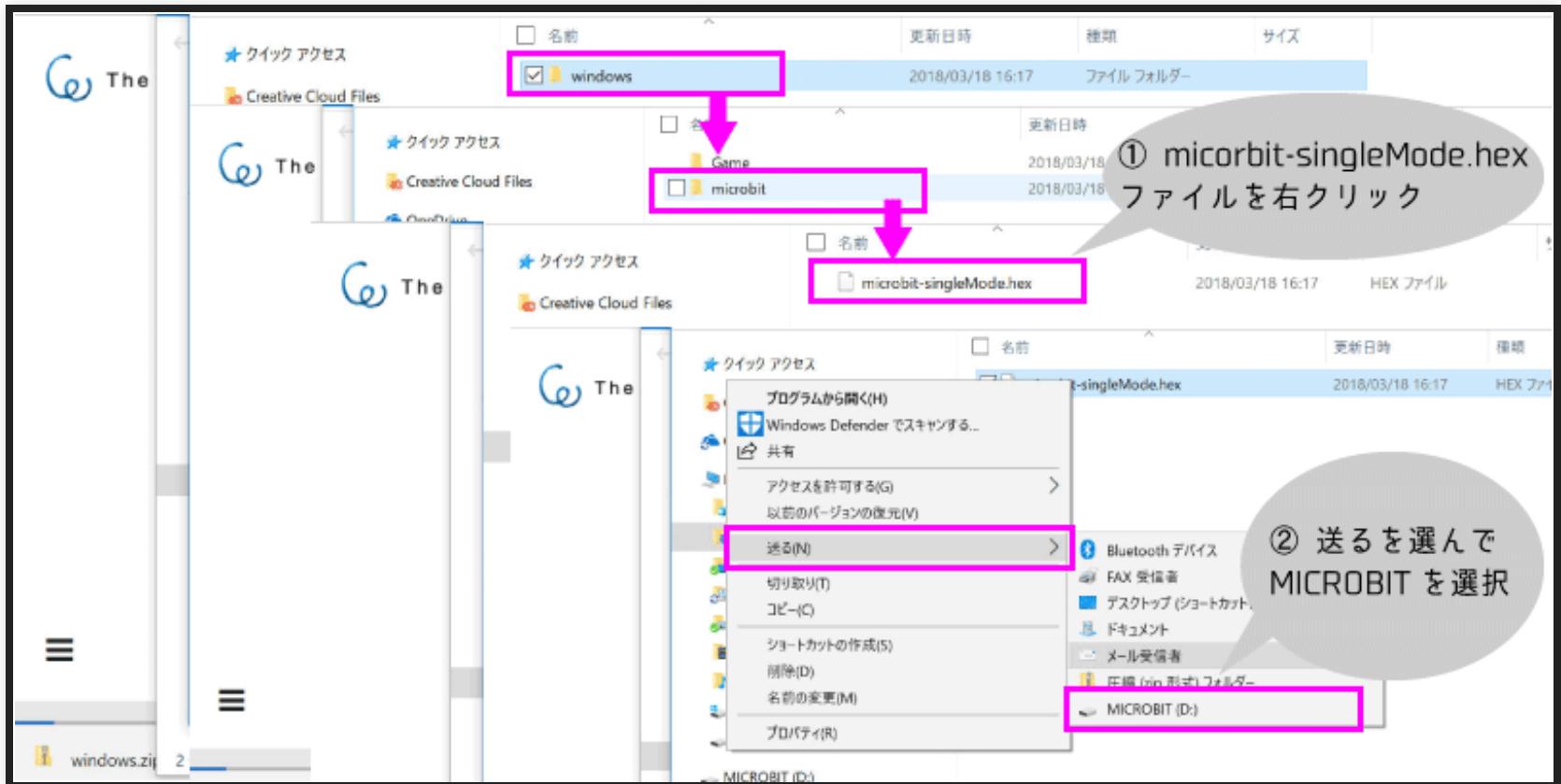








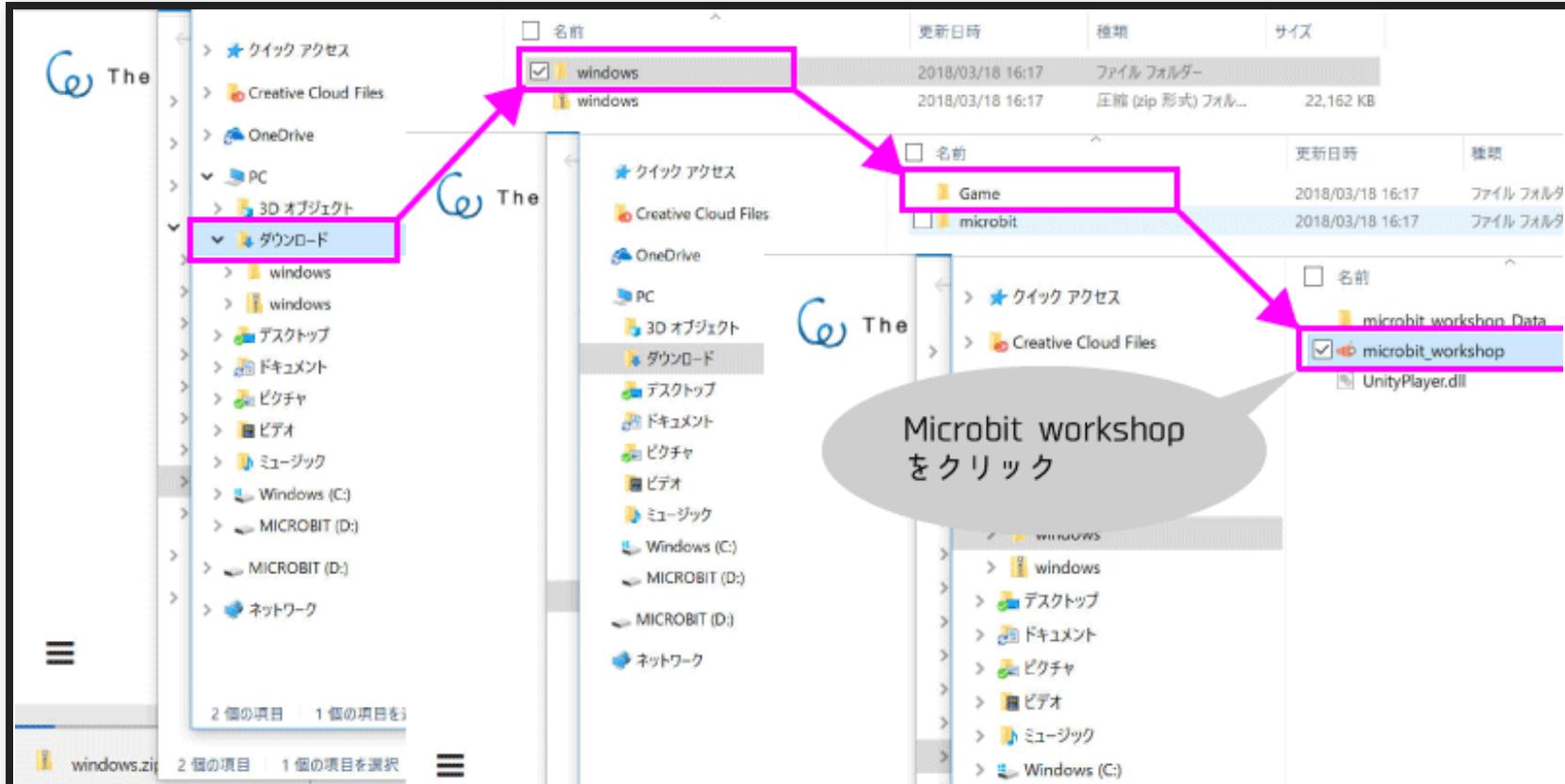




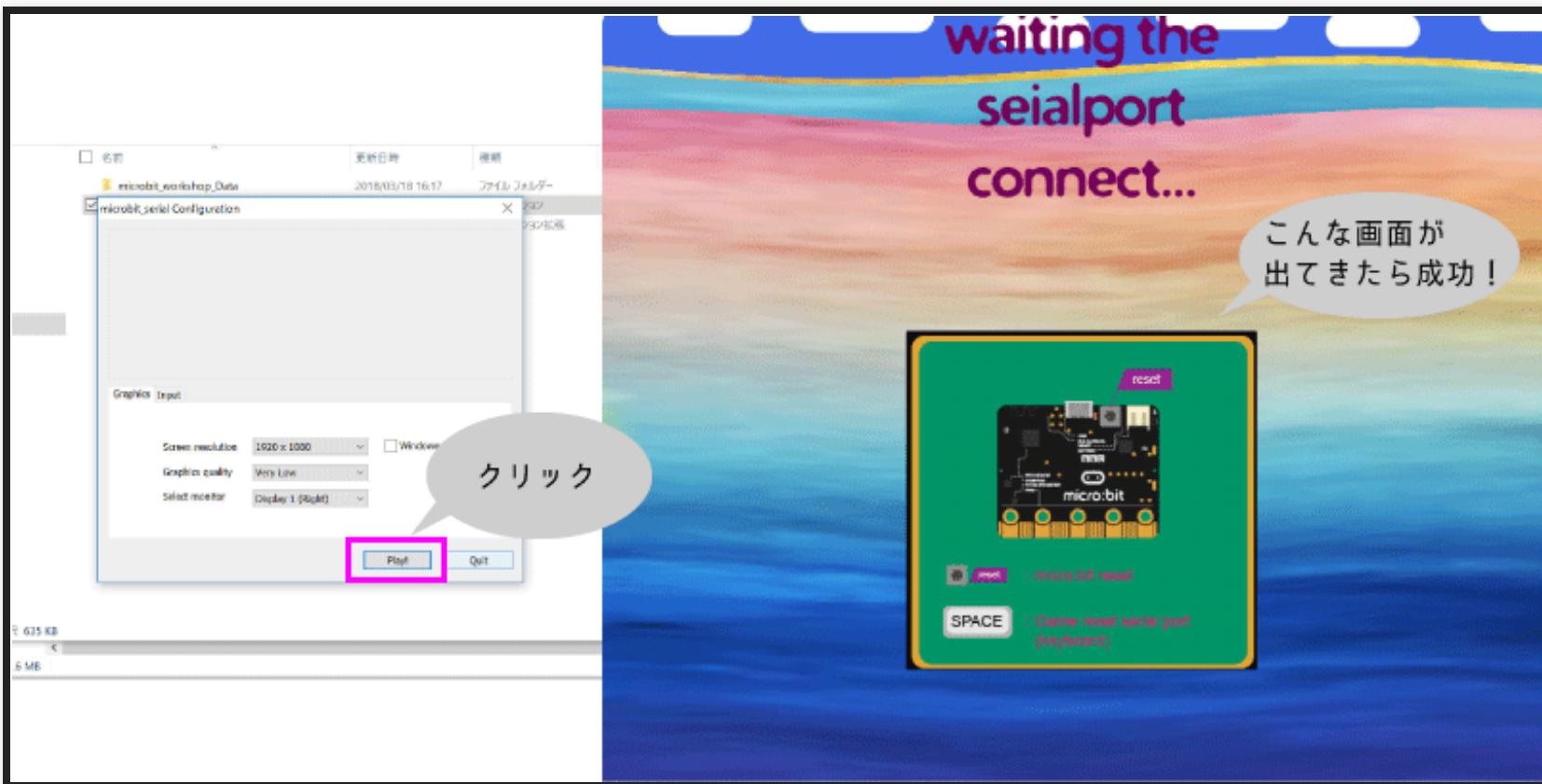
MICROBITにプログラムを入れたら次は
ゲームスタート！

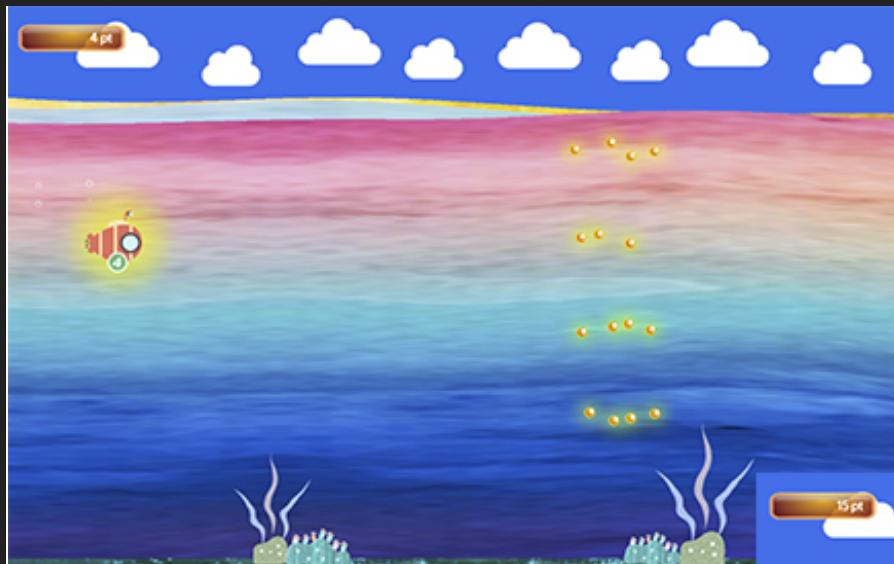


MICROBIT WORKSHOPファイルをクリック









A screenshot from the mobile game 'The Waggle'. The player character is a small, colorful fish-like creature with a red body and blue fins. It is swimming in an ocean with a gradient from blue at the bottom to yellow and orange at the top. There are several small, glowing yellow spheres scattered throughout the water. The fish is currently positioned near some green coral reefs at the bottom. A yellow progress bar at the top left shows '4 pt'. The background features a blue sky with white clouds.

Aボタンで上に
Bボタンで下に動きます。
光の玉を沢山の集めましょう。



A screenshot from the mobile game 'The Waggle'. The player character is now a small, colorful fish-like creature with a red body and blue fins. It is swimming towards the right side of the screen. A large, dark blue shark is swimming towards it from the right. The background shows a gradient ocean from blue to yellow. A yellow progress bar at the top left shows '15 pt'. The background features a blue sky with white clouds.

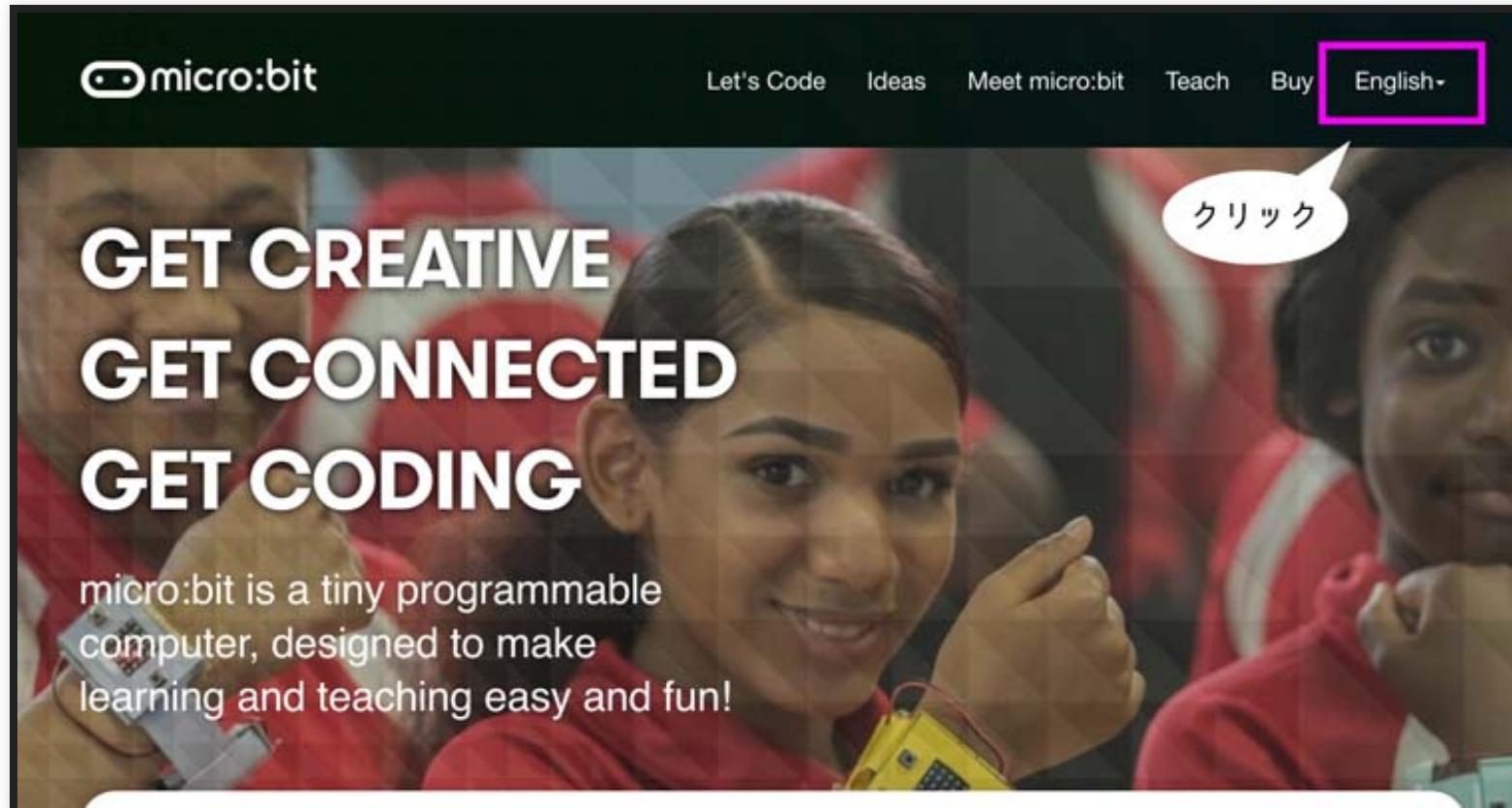
魚に当たったら GameOver !
RELEASEPOWER が表じされ
ている時に A+B ボタンを押し
ながら、光センサーを隠すと
...

早速！Micro:bitでプログラムをして見よう！

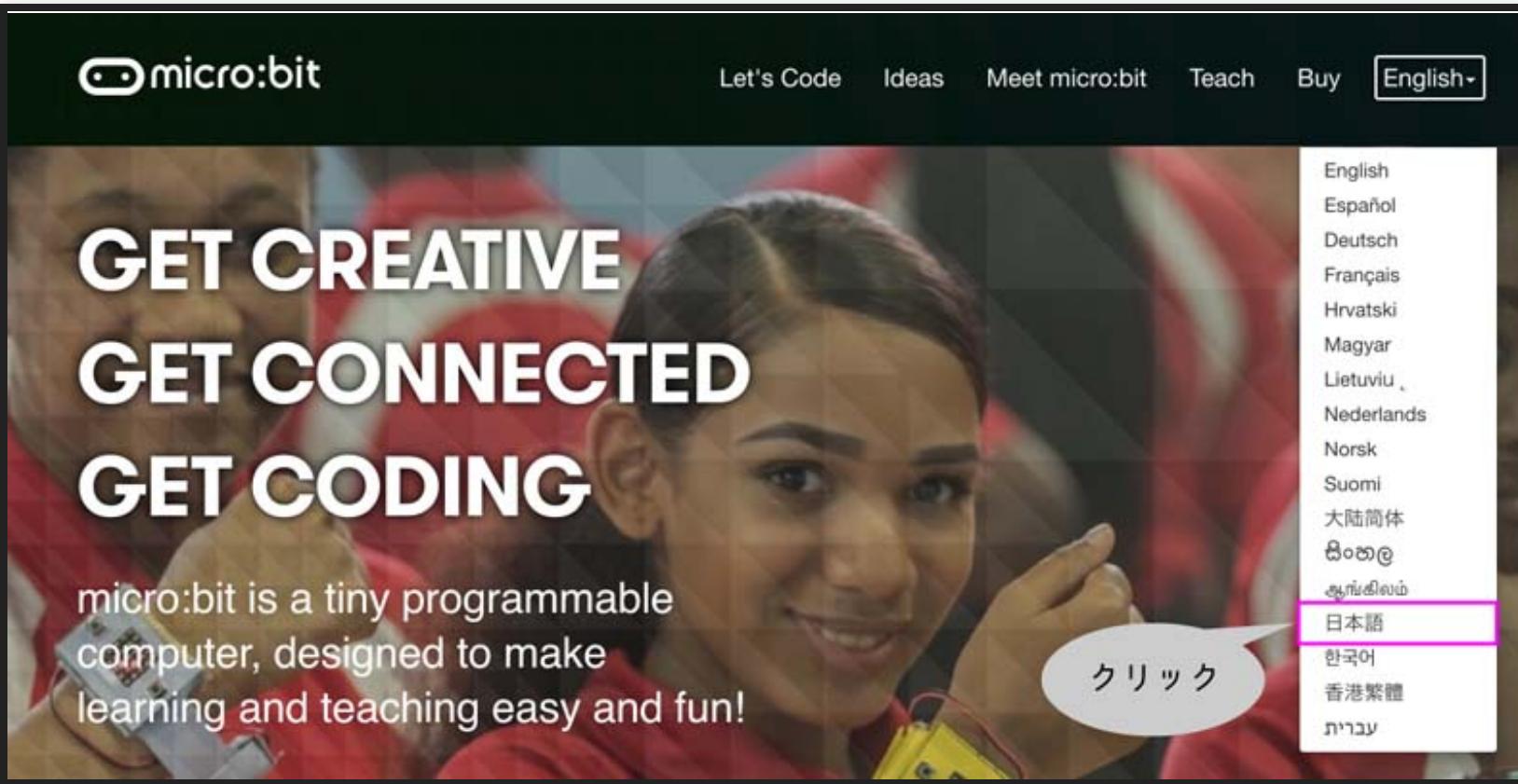
Micro:bit



チュートリアル TUTORIAL(まずははじめに使い方の説明)



[HTTP://MICROBIT.ORG/](http://microbit.org/)



The screenshot shows the official micro:bit website homepage. At the top left is the micro:bit logo. The top navigation bar includes links for "Let's Code", "Ideas", "Meet micro:bit", "Teach", "Buy", and a language dropdown set to "English". The main banner features a young girl holding a micro:bit board with the text "GET CREATIVE", "GET CONNECTED", and "GET CODING" overlaid. Below the banner, a subtitle reads: "micro:bit is a tiny programmable computer, designed to make learning and teaching easy and fun!". A speech bubble contains the Japanese text "クリック". To the right of the banner is a vertical list of language options: English, Español, Deutsch, Français, Hrvatski, Magyar, Lietuvių, Nederlands, Norsk, Suomi, 大陸简体, ສිංහල, മലയാളം, 日本語 (which is highlighted with a pink border), 한국어, 香港繁體, and עברית.







クリック

JavaScript ブロックエディタ

マイクロビット JavaScript ブロックエディターを使えば、BBC micro:bit を、ブロックまたは JavaScript で簡単にプログラミングできます。

MakeCode で開発。もしアクセスに問題があるなら、あなたの学校でエディターが ブロック されていないか確認してください。アイデアやひらめきが必要なら、これらの プロジェクト を参考にしてください。

[プログラムしましょう](#)

[リファレンス](#)

[Lessons](#)



きほん

「**基本**」をクリックしましょう。



さいしょ
「最初だけ」をクリックしましょう。



きほん
次に、また「基本」をクリックしましょう。



もじれつひょうじ
「文字列を表示」をクリックしましょう。



もじれつひょうじ
「文字列を表示」をドラッグして「最初だけ」の
中に入れよう。

さいしょ



一回だけ「HELLO!」という文字が表示されること
かくにん
を左のシミュレーターで確認しよう。



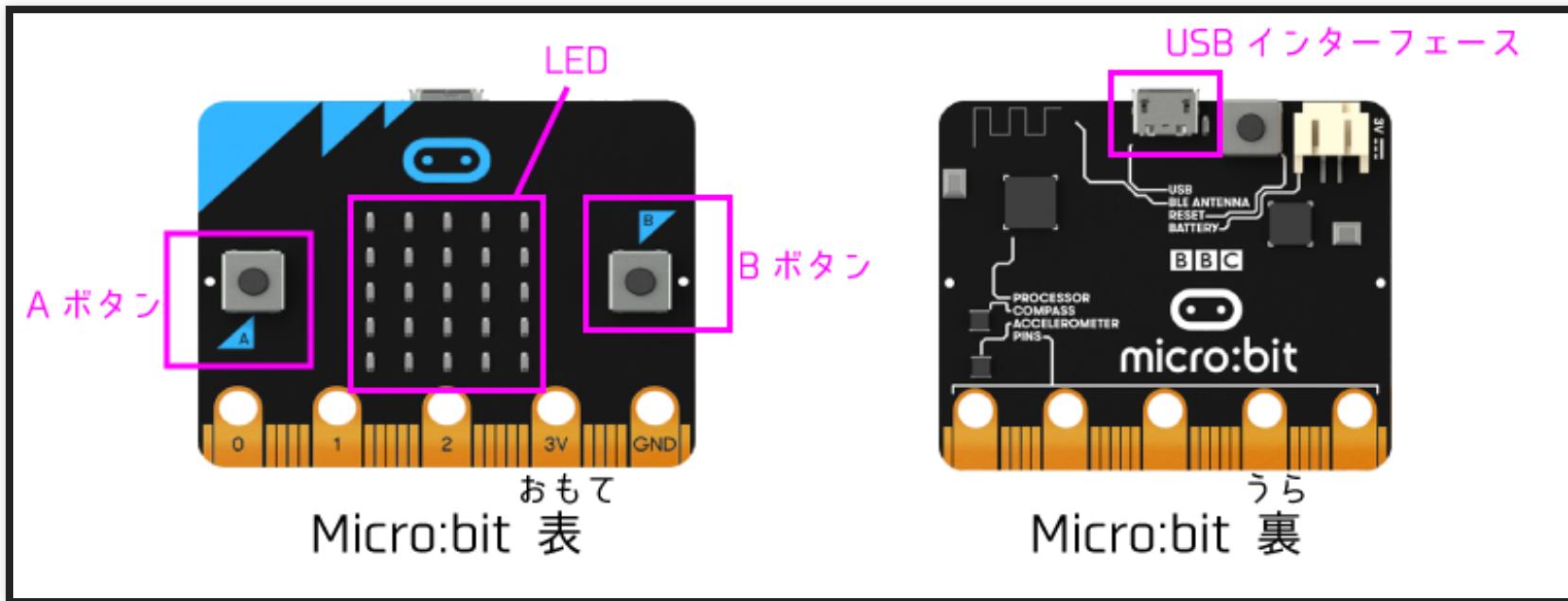
左のエリアにドラッグしてブロックを削除しよう。

レッスン
LESSON

そうさほうほう りかい
操作方法は理解しましたか？

理解ができたところで、実際にmicrobitを自分の
思い通りに動かせるようにプログラムを組んで
いきましょう！

ハードウェアの紹介 :MICRO:BIT

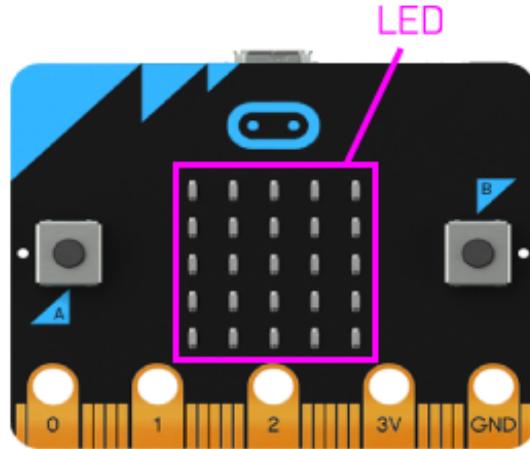


にゅうりょく しゅつりょく そな
マイクロコンピュータと入力や 出力を 備えた
きばん
基盤です。

とりあつか
ちゅういてん
取 扱いの注意点！ 濡れた手でさわらない



マイクロビット きのうしょうかい
MICRO:BIT の 機能紹介 LED



エル イー ディー はっこう
LED : は 発行ダイオード (LIGHT EMITTING DIODE)
という意味です。MICRO:BITには25個のLEDがあり、
これを使って数字や文字を表現します。

LESSON1 「出力」

Micro:bitのLEDを自由に表示できるようにしよう。



LESSON1 の目的

- LEDの表示を自分の意図した通りに表現する。
- 「最初だけ」 「ずっと」 「ミリ秒」 を理解する。
- プログラムをmicro:bitに入れて動かす。

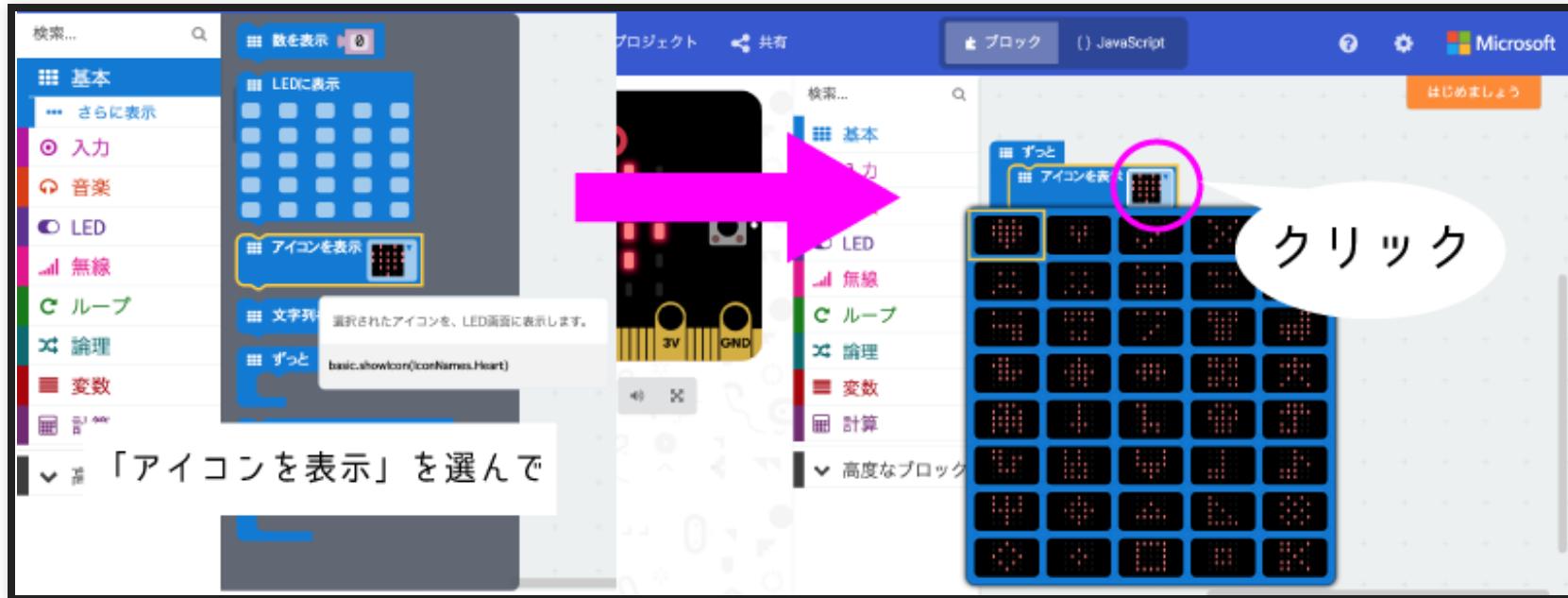
≡

MICRO:BITのLEDを制御して色々な表示をしよう



えら
「基本」→「ずっと」を選んでクリックします。

アイコンを表示しよう



「アイコンを表示」を選んで「ずっと」ブロックの中に入れ、アイコンをクリックして好きなアイコンを表示するプログラムを作りましょう。

プログラムとは

プログラムは、コンピュータを動かす為に「いつ」「どうする」を書いた命令書みたいなものだよ。

「いつ」を表すブロック

最初だけ

最初だけ
(この中にある「どうする」)
を動かしてね。

ずっと

ずっと
(この中にある「どうする」)
をくりかえし動かしてね。

「どうする」を表すブロック

アイコンを表示



選んだアイコンを表示してね

一時停止(ミリ秒)

100

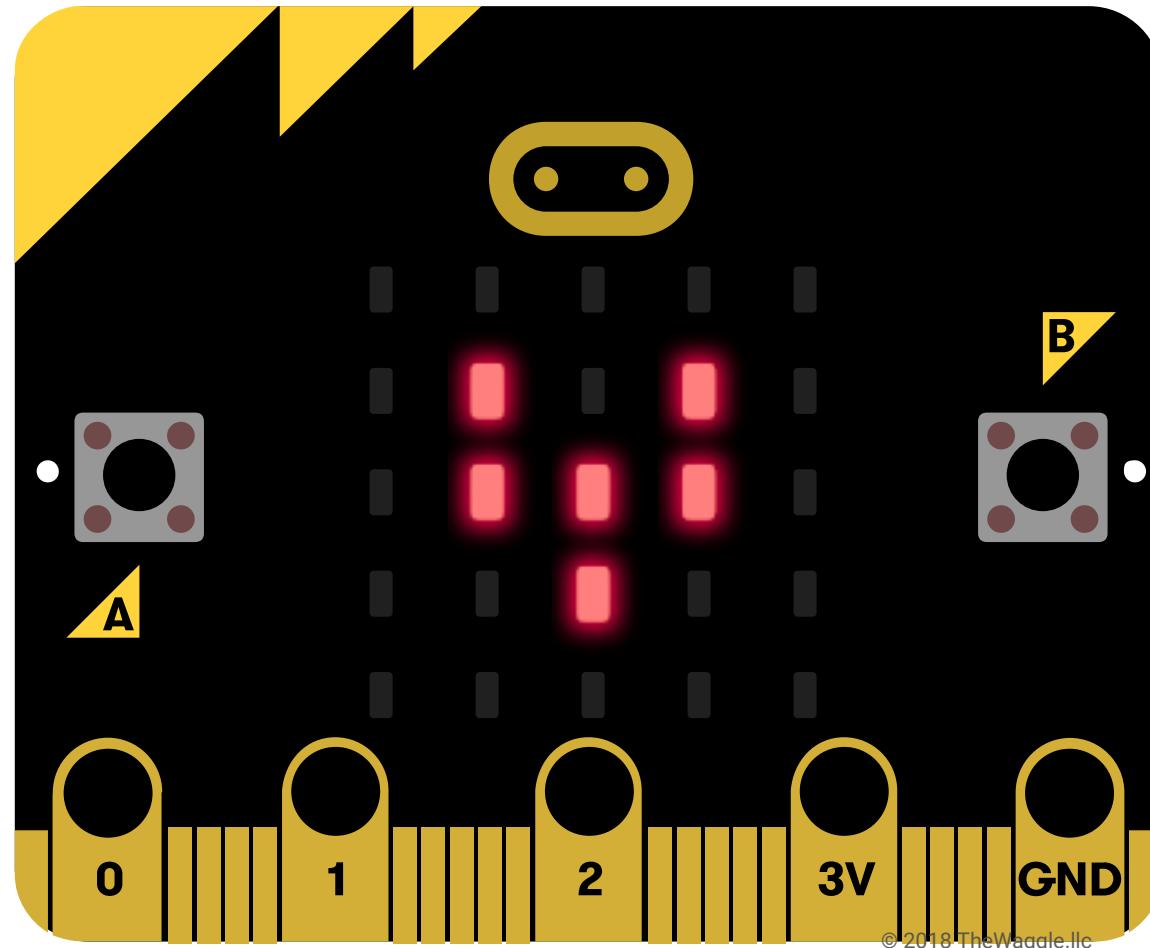
100ミリ秒動くの
止めてね

+

ブロックは「いつ」 + 「どうする」の組み合わせで命令を作る

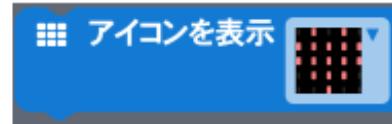
やってみよう!

ハートがドキドキするアニメーションを作ってみよう！



ヒント

利用するブロックの種類



大きいハートのアイコン

小さいハートのアイコン

プログラムに名前をつけよう。



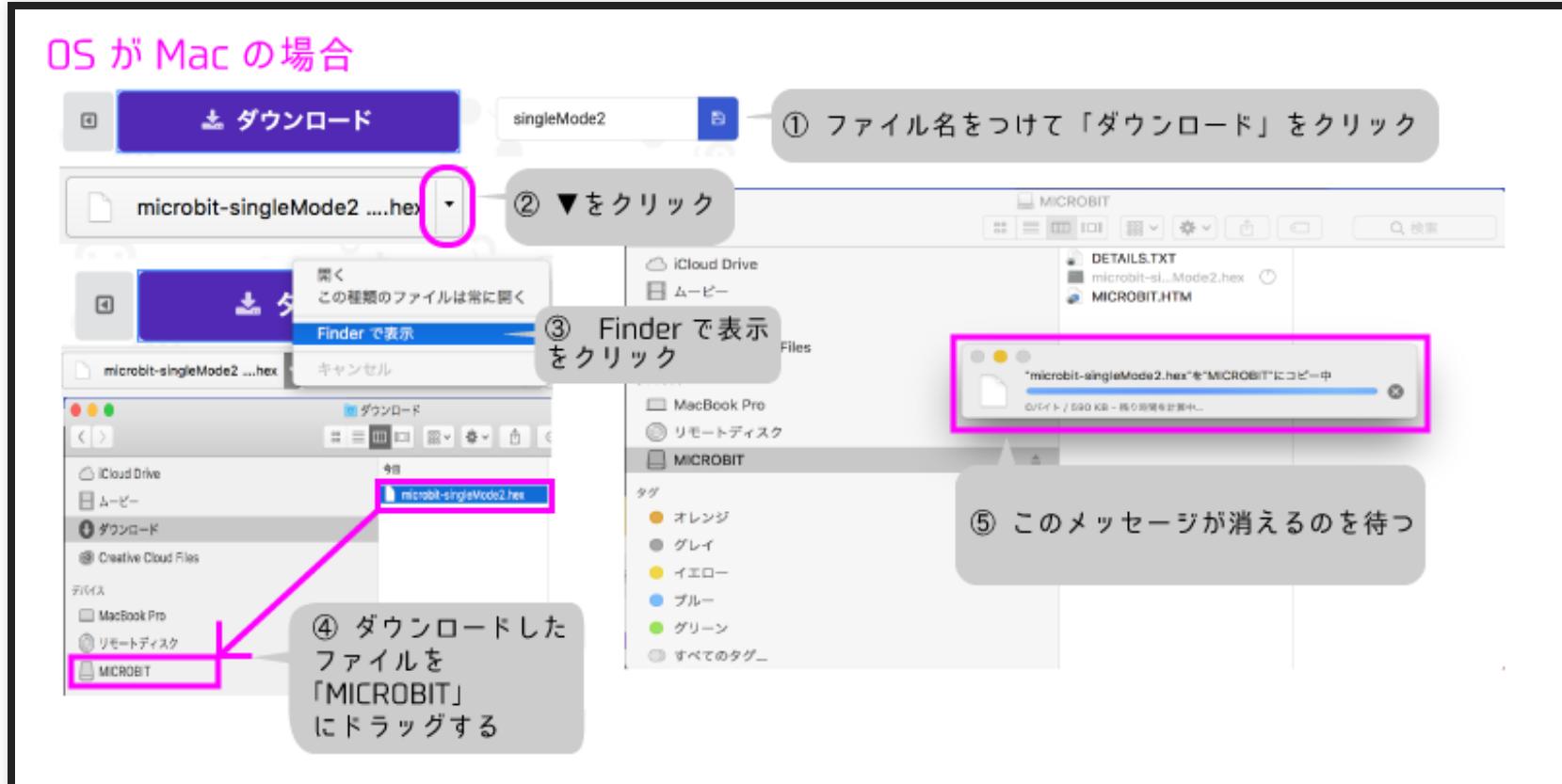
- 「クリック」
- 「delete」キーで「題名未設定」を削除
- 「Heart」と入力します。

プログラムを保存しよう。

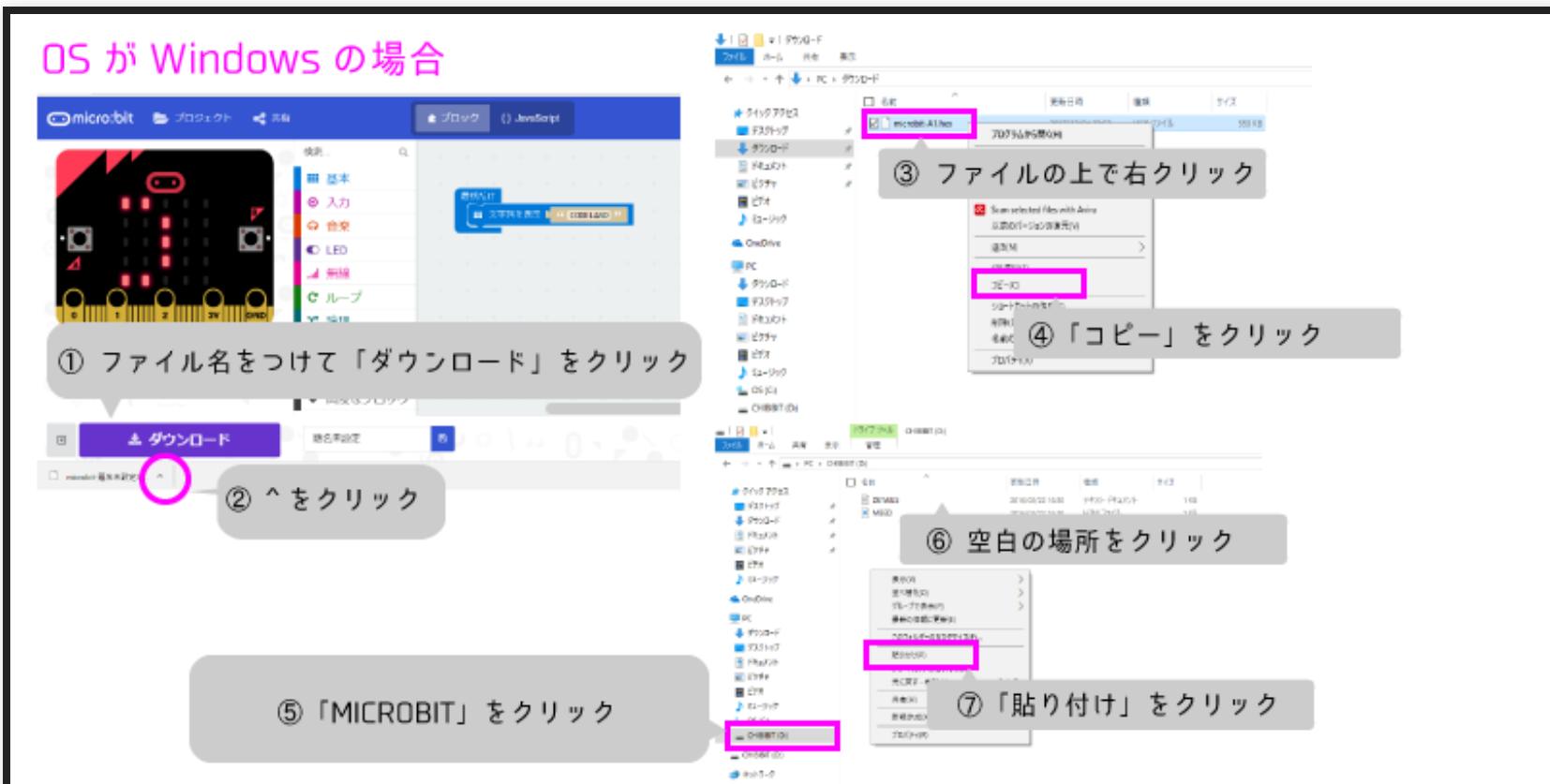


- 上をクリックするとプログラムファイルが
ダウンロードフォルダに保存される。

プログラムをMICRO:BITに入れてみよう！MACの場合



プログラムをMICRO:BITに入れてみよう！ WINDOWSの場合



LESSON2 [入力]

Micro:bitのLEDに色々な表示ができるようになりましたか？
LESSON1では、Micro:bitの出力を変更する事をやりました。

それでは、続いてMicro:bitの「入力」を使って
「出力」する物を変える事をやって見ましょう。

LESSON2 の目的

- 入力条件について理解する
- 変数の使い方が分かる

≡

ボタンを使って、表示する数値を 変えてみよう。



Aボタンを押すと「0を表示する」を作る



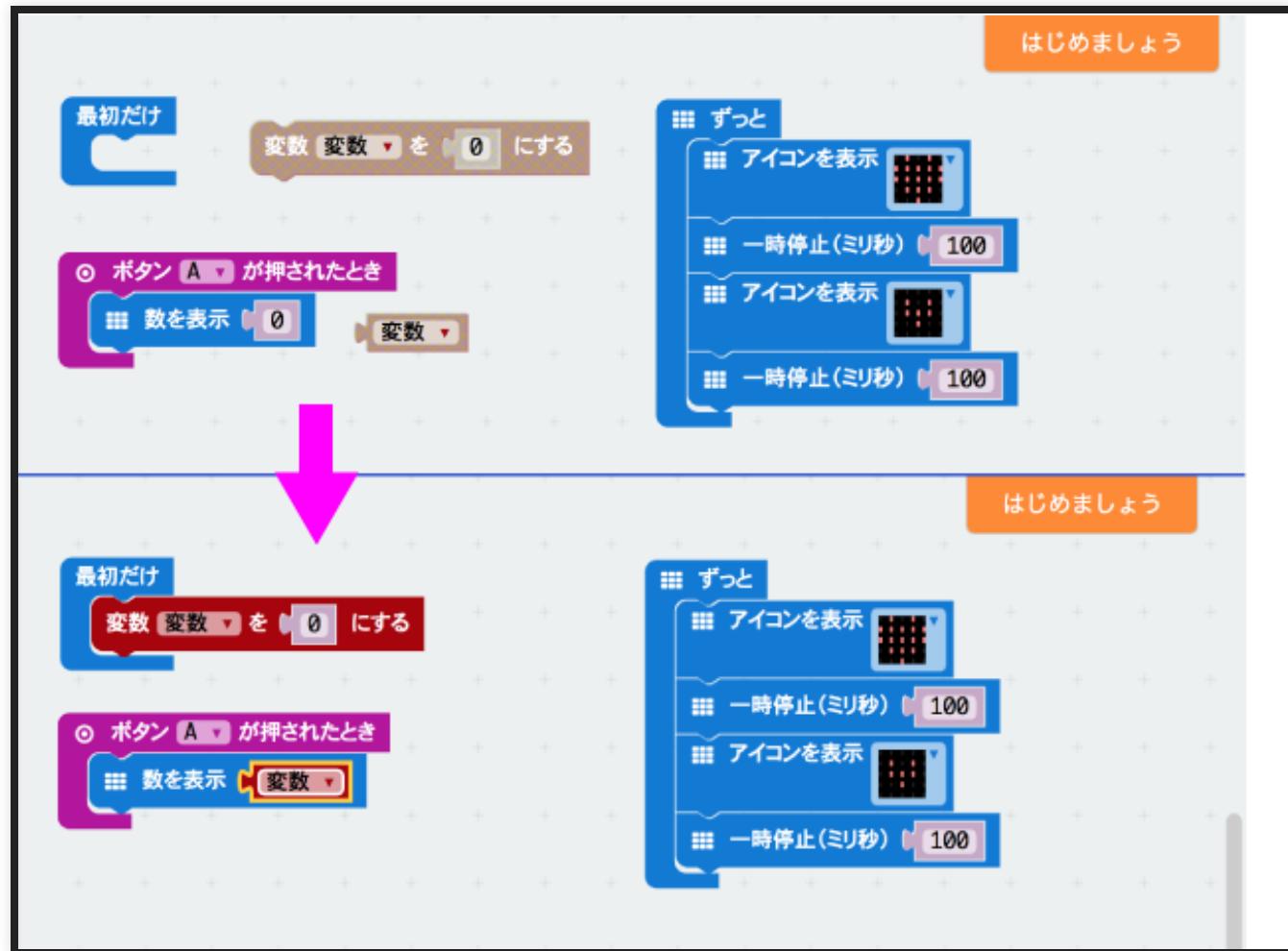
変数を使ってみよう [変数] をクリック



「変数を 0 にする」をクリック



「変数」を使って数字を表示しよう



Aボタンを押して「0」が表示された



変数のイメージ

「変数」はこんなイメージで使います。



「変数」という名前のカラッポの箱を作りました。



「変数」に「0」を入れる

最初だけ

変数 **変数** を **0** にする



「変数」に1を足す

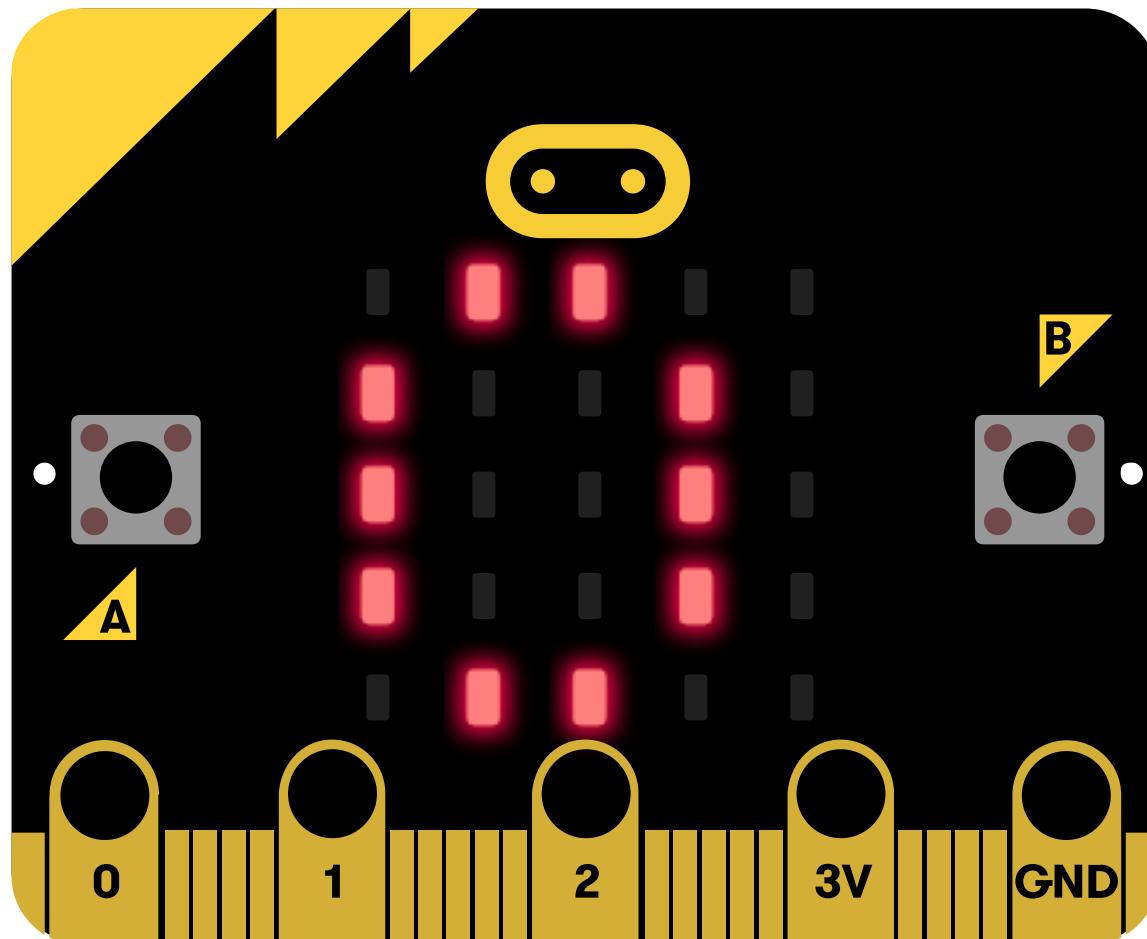
変数 **変数** を **+1** だけ増やす

Bボタンを押して表示する数字を増やそう



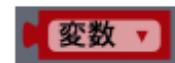
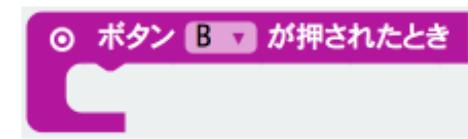
やってみよう！

Aボタンを押すと、数字が増えて Bボタンを押すと、
数字が0になる



ヒント

利用するブロックの種類



LESSON3 [制御]

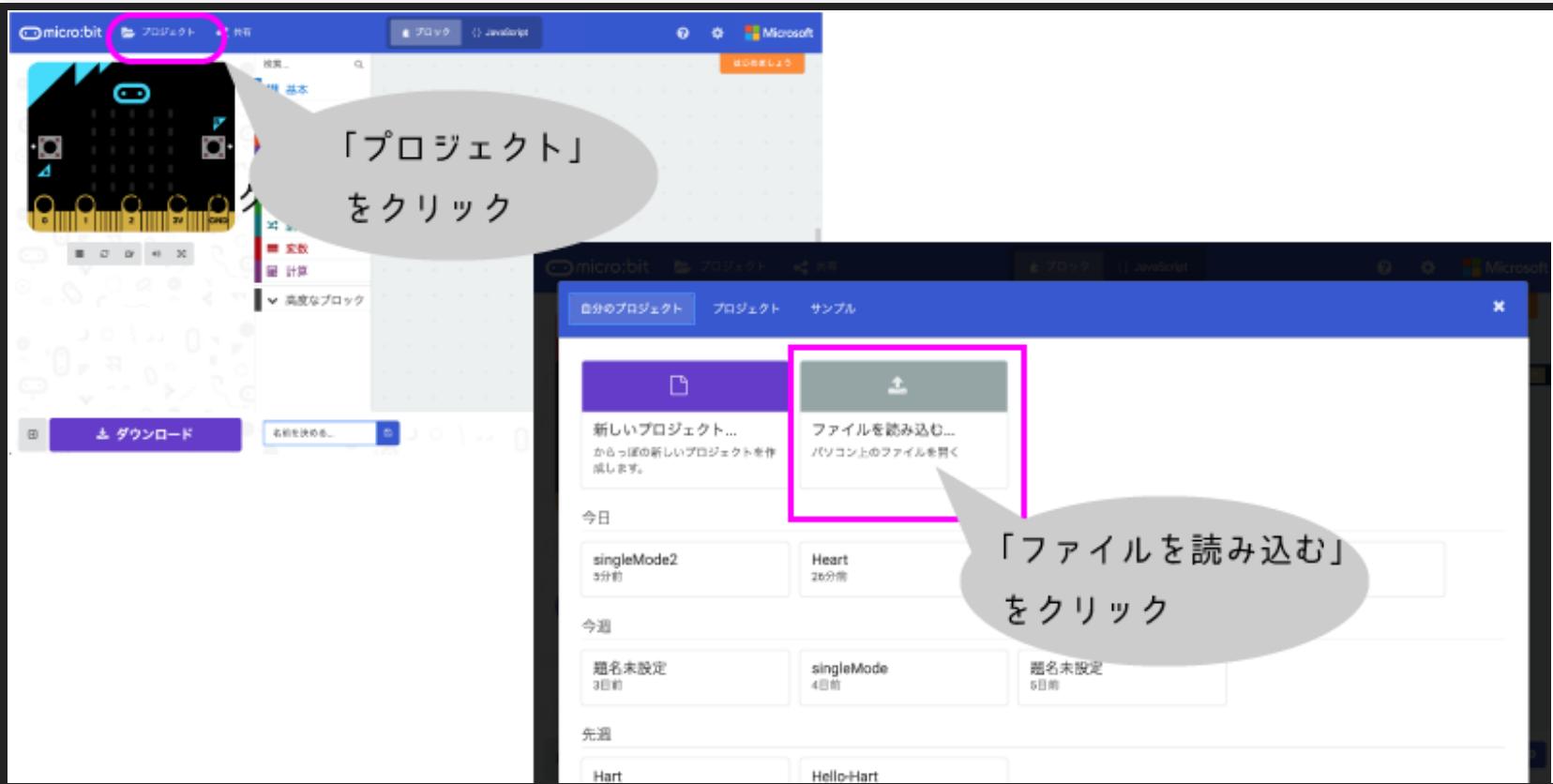
それでは、いよいよ、プログラムの制御について学びます。

実際に体験したゲームコントローラーのプログラムを改造しながら、学んで行きましょう。

LESSON3 の目的

- 条件分岐を理解する
- 複数の変数を利用する

プログラムの読み込み



プログラムを選択する



ボタンの機能を変更しよう

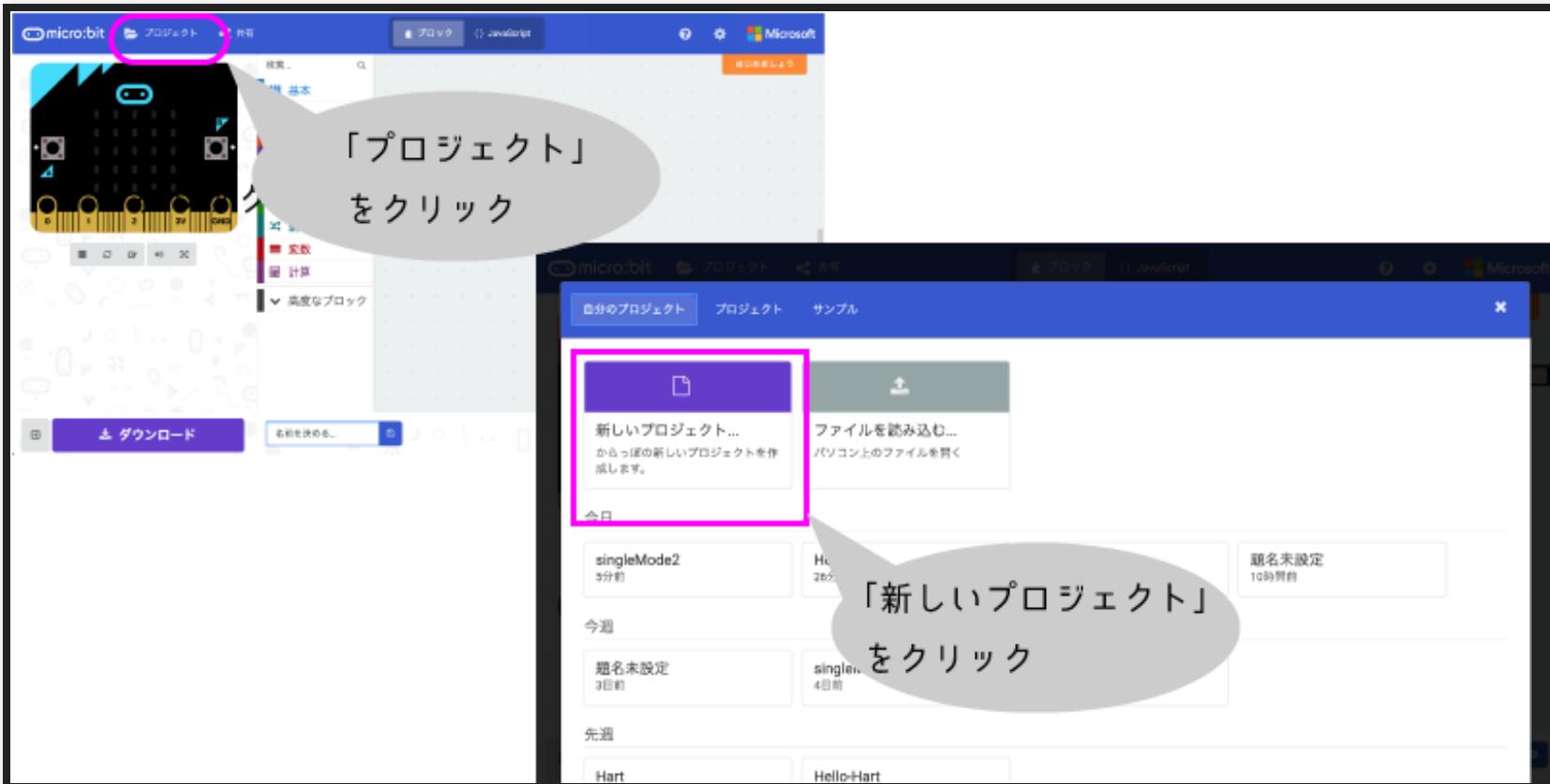


A+Bボタンの中を見てみよう



[lightVal]という変数に「明るさ」の中の数値を入れている。

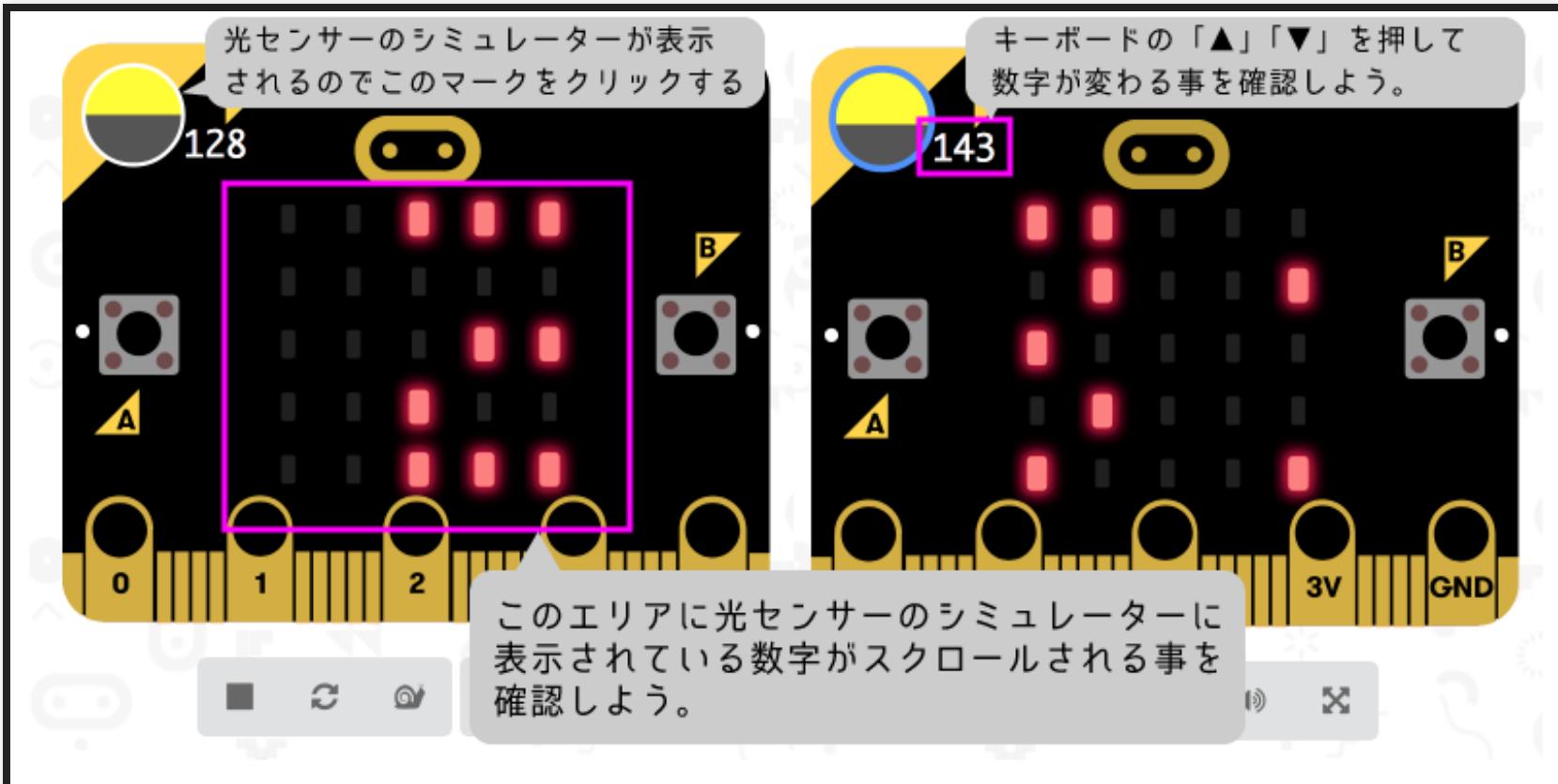
明るさの数値を表示してみよう。



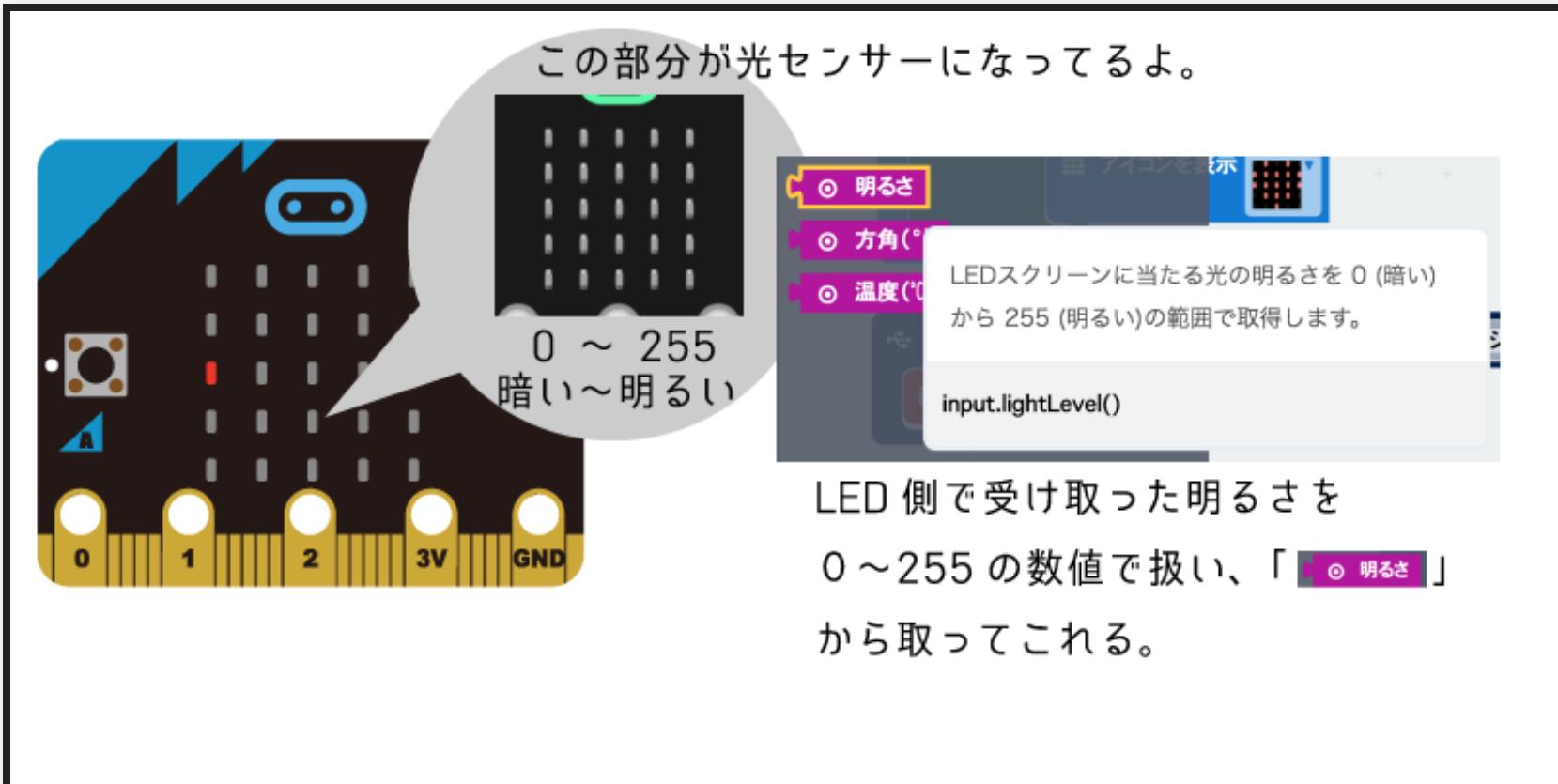
明るさを数字表示に入れてみよう



明るさの数値を確認



「明るさ」の仕組み



じょうけんぶんき

条件分岐



The image shows a Scratch interface with Japanese labels. On the left, there's a search bar and a category sidebar with the following items:

- 検索...
- クリック
- 音楽
- LED
- 無線
- ループ
- 論理** (highlighted in teal)
- 変数
- 計算

The main workspace displays two Scratch scripts side-by-side. Both scripts use a "もし...なら..." (If...then...) conditional block. The first script has the condition set to "真" (True) and the second to "偽" (False). Both scripts then branch into two "でなければ..." (unless...) blocks, each with a "数を表示" (Display number) block set to either 1 or 0.

Text in the workspace asks: 「真」と「偽」どちらが1を表示するでしょうか？ 試してみよう！

Below the workspace, a micro:bit board is shown with its pins and digital pins labeled 0, 1, 2, 3V, and GND. The board has red LED lights on its digital pins.

もっと詳しく条件分岐を知ろう



最初だけ
もし
なら
でなければ
数を表示 1
数を表示 0

最初だけ
変数 変数名 を 真にする
もし
なら
でなければ
数を表示 1
数を表示 0

最初だけ
もし
なら
でなければ
数を表示 1
数を表示 0

最初だけ
もし
なら
でなければ
 $0 = 0$
数を表示 1
数を表示 0

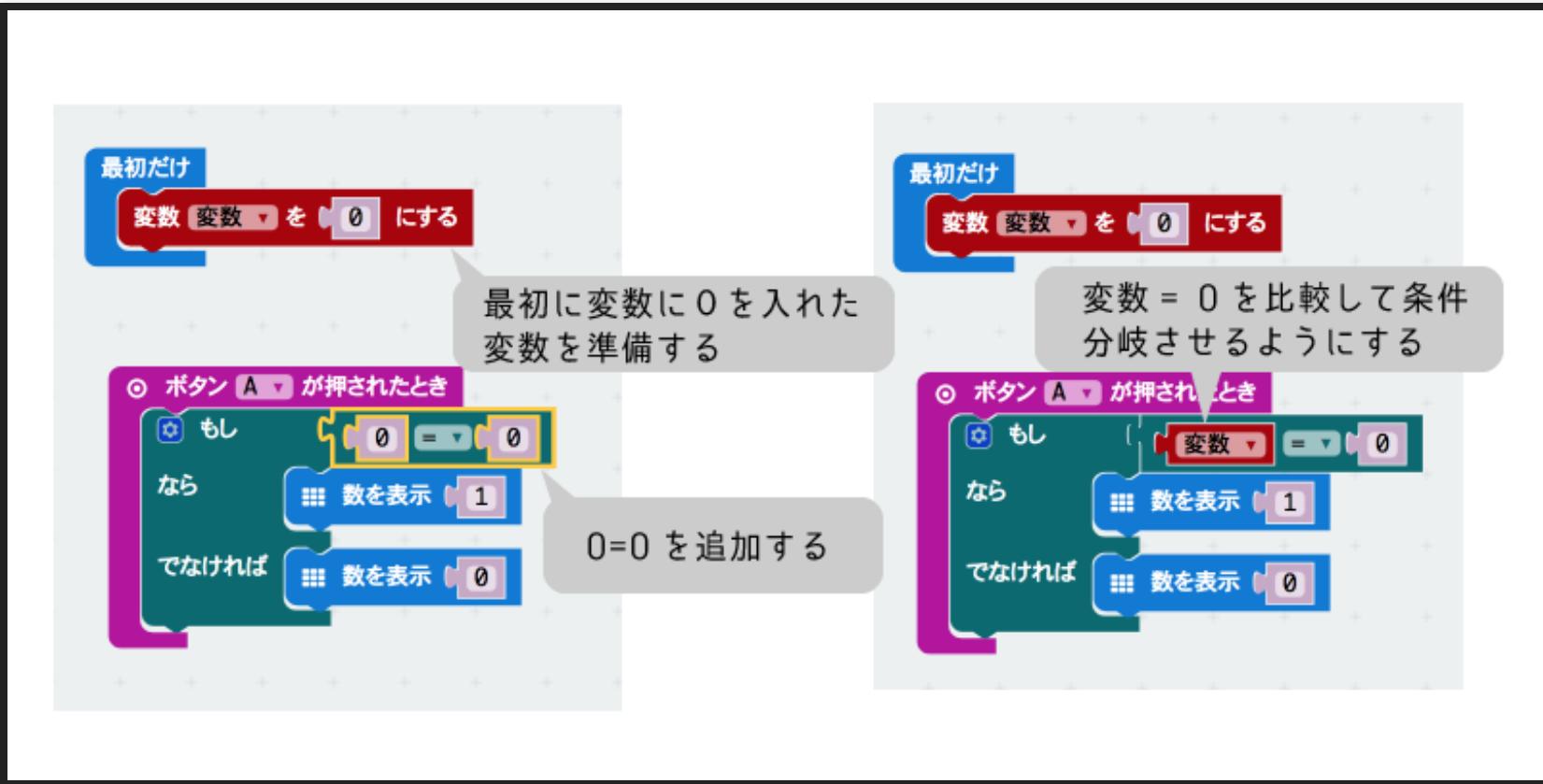
もし「真」なら 1 を表示する

変数も使える

もし「偽」なら 0 を表示する

条件式も使える 「0」 = 「0」
この場合は、= が成り立つので 1 を表示

変数を使った条件分岐



The image shows two Scratch scripts side-by-side, both starting with a "最初だけ" (Only once) hat block followed by a "変数 [変数 v] を [0] にする" (Set variable [variable v] to [0]) block.

The left script is titled "最初に変数に0を入れた変数を準備する" (Prepare a variable with value 0). It has a control loop "◎ ボタン A が押されたとき" (When A key pressed) containing an if-then-else block:

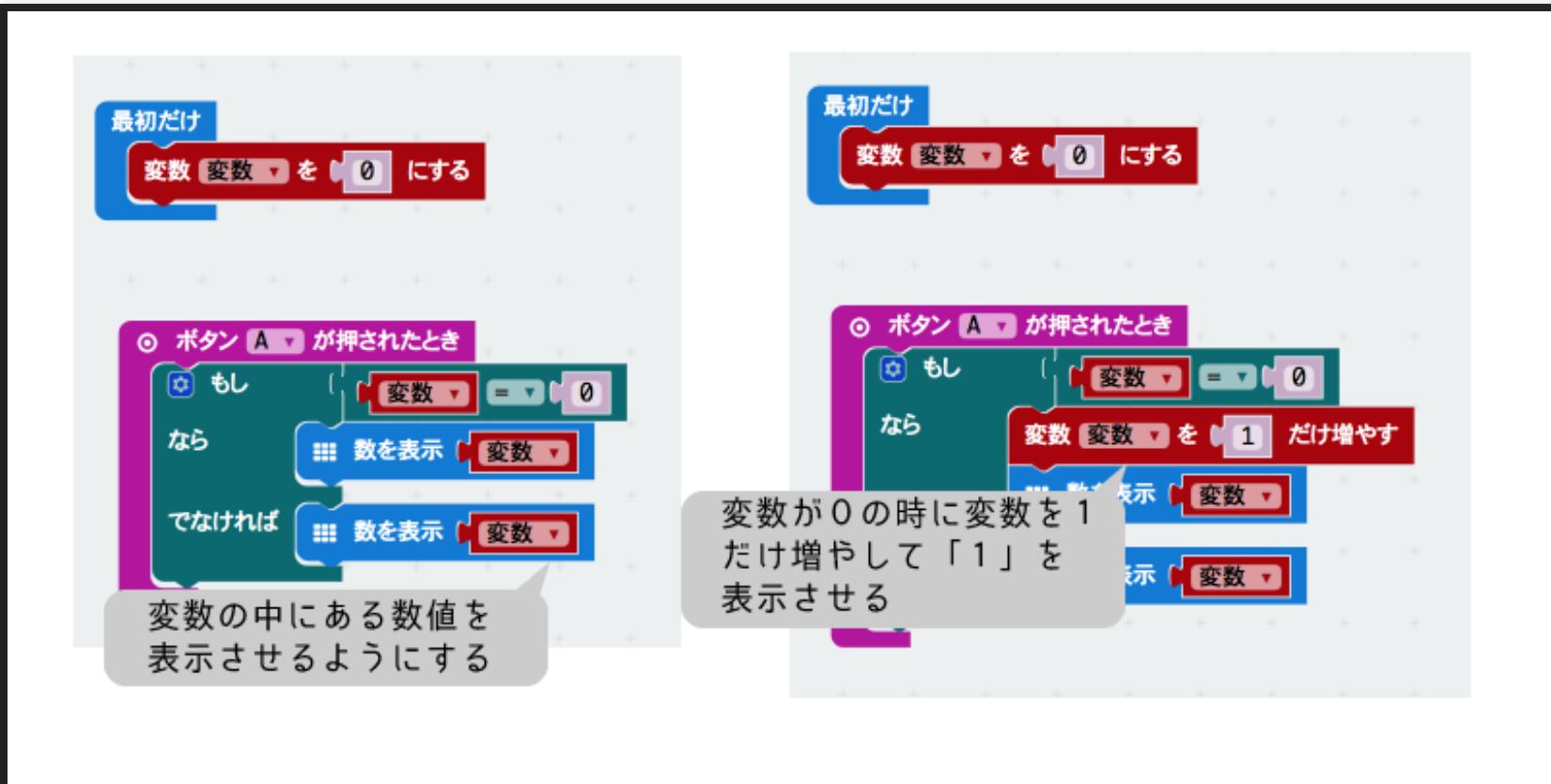
- もし [0] = [0] なら [数を表示 1]
- でなければ [数を表示 0]

The right script is titled "変数 = 0 を比較して条件分岐させるようにする" (Create a conditional branch by comparing variable = 0). It also has a control loop "◎ ボタン A が押されたとき" (When A key pressed) containing an if-then-else block:

- もし [変数 v] = [0] なら [数を表示 1]
- でなければ [数を表示 0]

Both scripts include a callout box "0=0 を追加する" (Add 0=0) pointing to the condition in the if-then-else blocks.

変数を操る



変数と条件分岐の活用

最初だけ
変数 [変数] を [0] にする

変数と条件分岐を使う事で、A ボタンを押す動作で
2 つの異なる結果を操る事が出来るようになりました。



```
when green flag clicked
  set [num v] to [0]
  if [num > 0] then
    set [num v] to [1]
    say [0]
  else
    set [num v] to [0]
    say [1]
  end
end
```

○ ボタン A が押されたとき

- もし [変数] = [0]
 - 変数 [変数] を [1] だけ増やす
 - 数を表示 [変数]
- でなければ [変数] を [0] にする

変数が 1 の時に変数を 0 に変えて「0」を表示させる

再び、コントローラーのプロジェクトを開こう



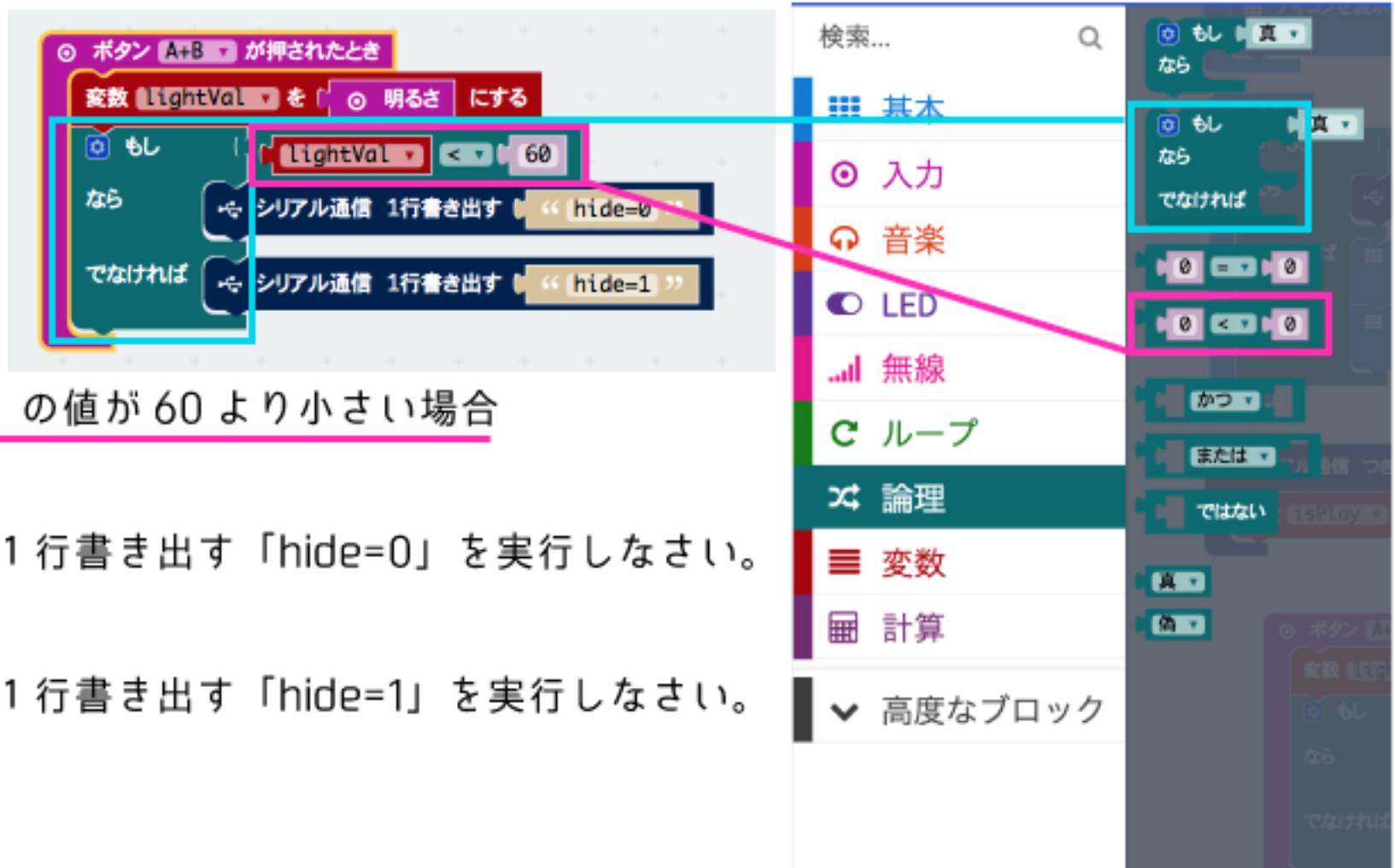
A+Bボタンのプログラムを確認しよう



条件分岐を使ってみよう

じょうけんぶんき 条件分岐

もし「lightVal」の値が 60 より小さい場合
が成り立つなら
 シリアル通信 1 行書き出す「hide=0」を実行しなさい。
でなければ
 シリアル通信 1 行書き出す「hide=1」を実行しなさい。

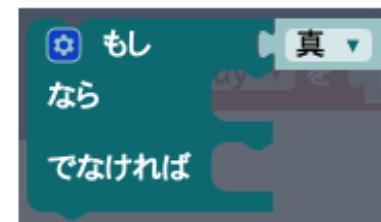
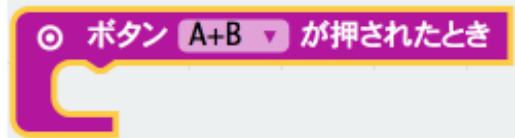


やってみよう！

- A+Bボタンをおすと、LEDの画面に1と表示されて、
- もう一度押すと0と表示される。
- 1と表示されている間は、プレイヤーが消えて、
- 0と表示されている間はプレイヤーが表示される
- flagという変数を作ってやってみよう！

ヒント

利用するブロックの種類



変数を追加する

変数 [変数 ▾] を 0 にする

flag ▾

→ シリアル通信 1行書き出す “hide=0”

→ シリアル通信 1行書き出す “hide=1”

数を表示 0

さあ、他にも改造してオリジナルの
コントローラーを作ろう！

≡