

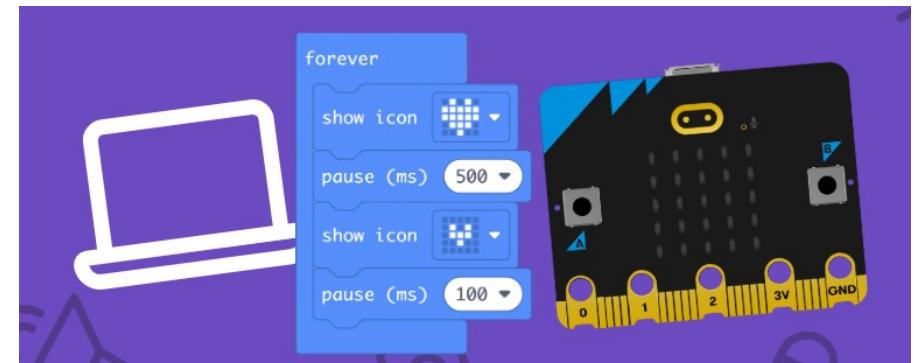


micro:bitにチャレンジ！

micro:bitとは

イギリスのBBC（英国放送協会）が主体となって作成した
教育用マイコンボード（マイクロコンピュータ）

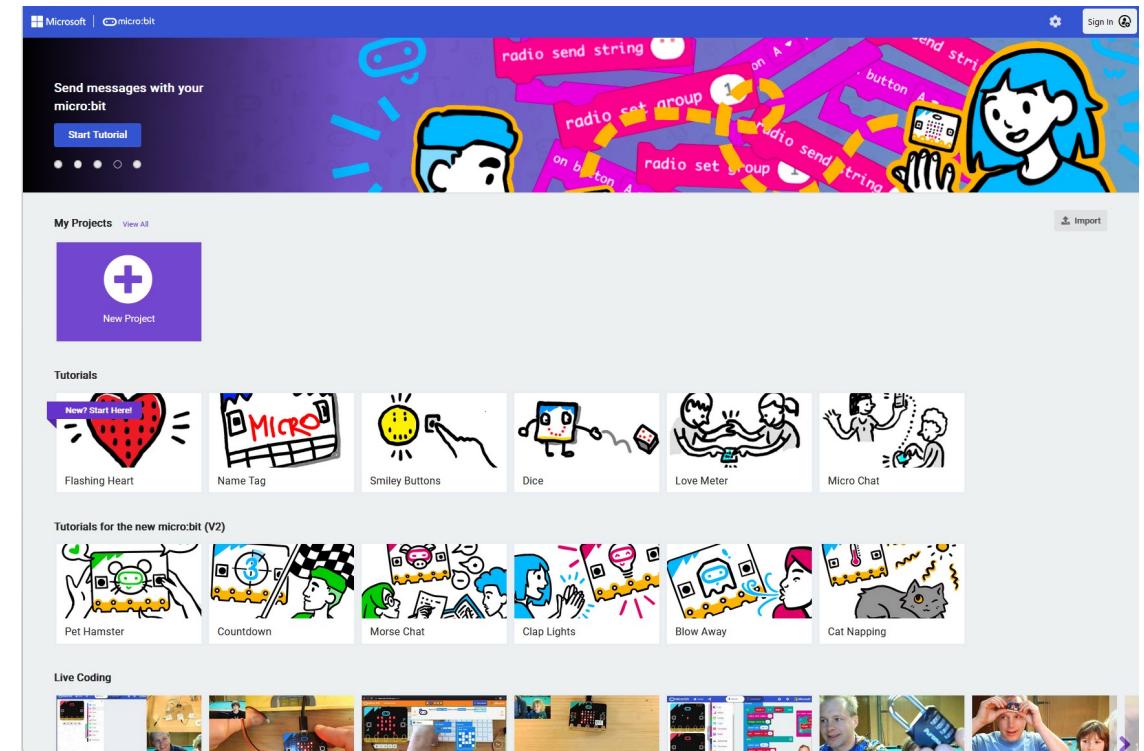
- ・何ができるの？
 - ビジュアルプログラミング、JavaScript、Pythonを使ってプログラミングができるよ
 - Aボタン、Bボタンと5x5のLEDがついているよ
 - 文字や数字、絵を表示できるよ
 - 気温や方角を表示したり、ロボットを動かしたり、
超音波できよりをはかったりもできるよ



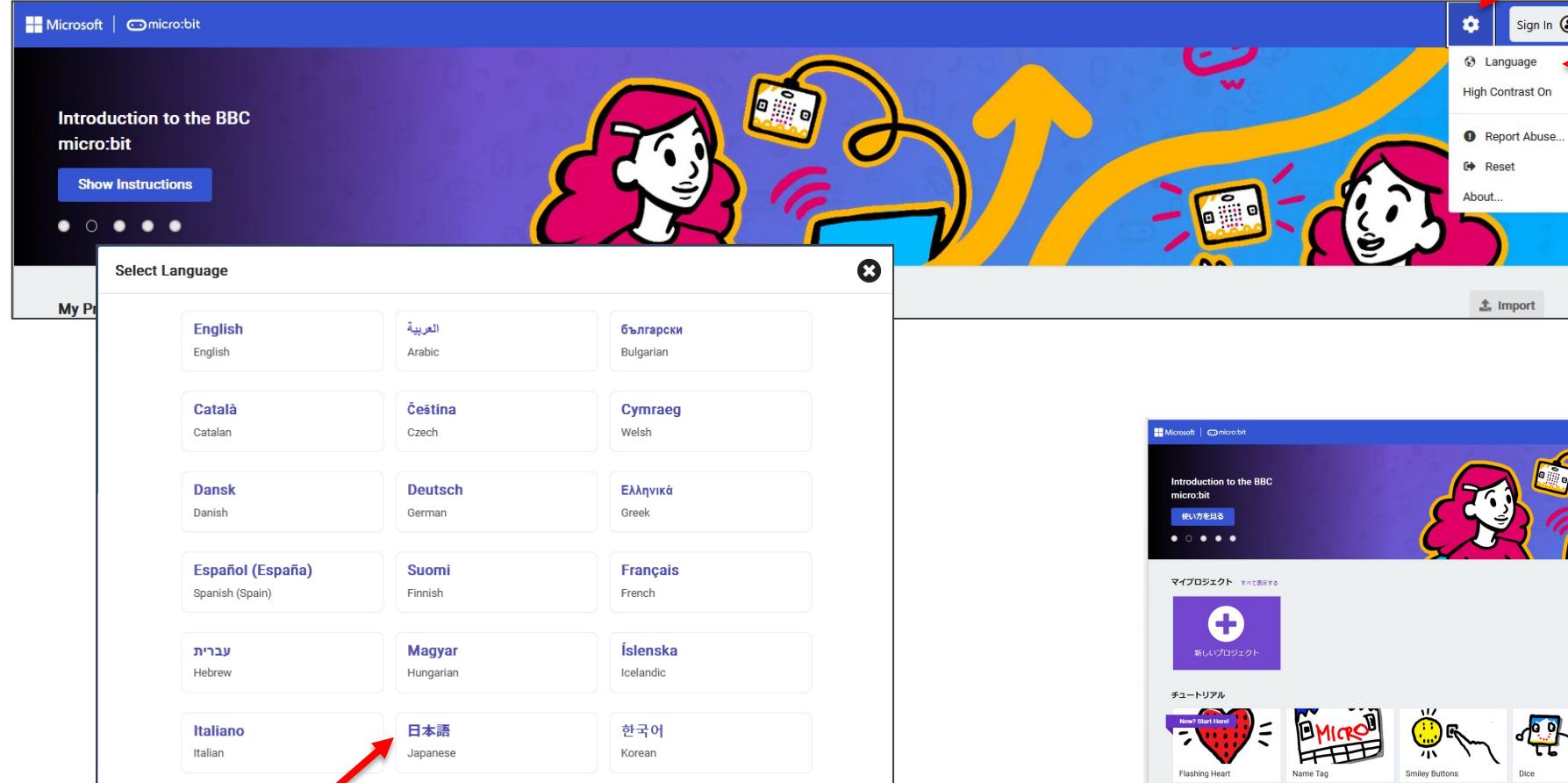
micro:bitをプログラムしてみよう

パソコンのブラウザなどからプログラミングできる

- <https://makecode.microbit.org/>



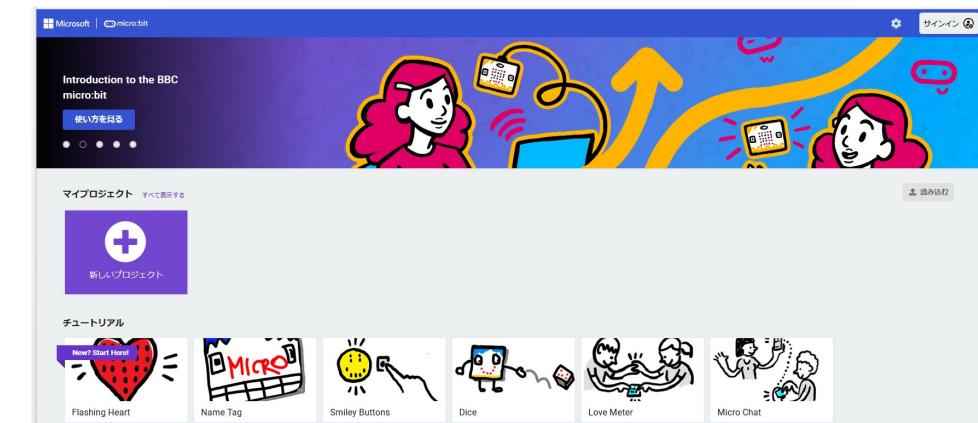
がめんをにほんごにしよう



③日本語をクリック

①はぐるまをクリックしてね

②Languageをクリック



にほんごになった！

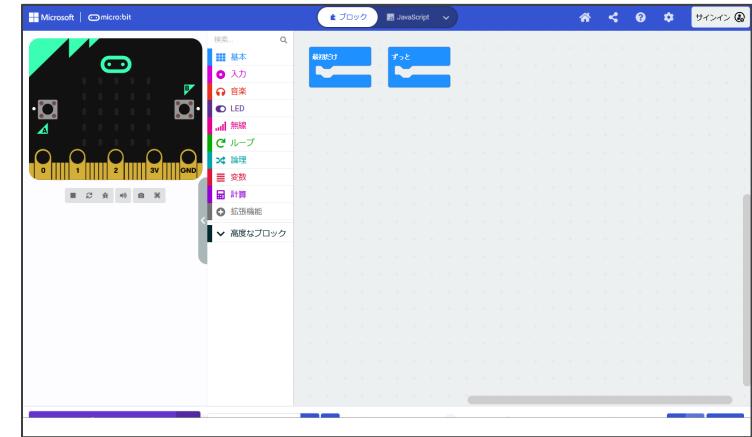
ぷろぐらみんぐのじゅんび



「新しいプロジェクト」
をクリック！



つくるプログラムの
名前を付けてね

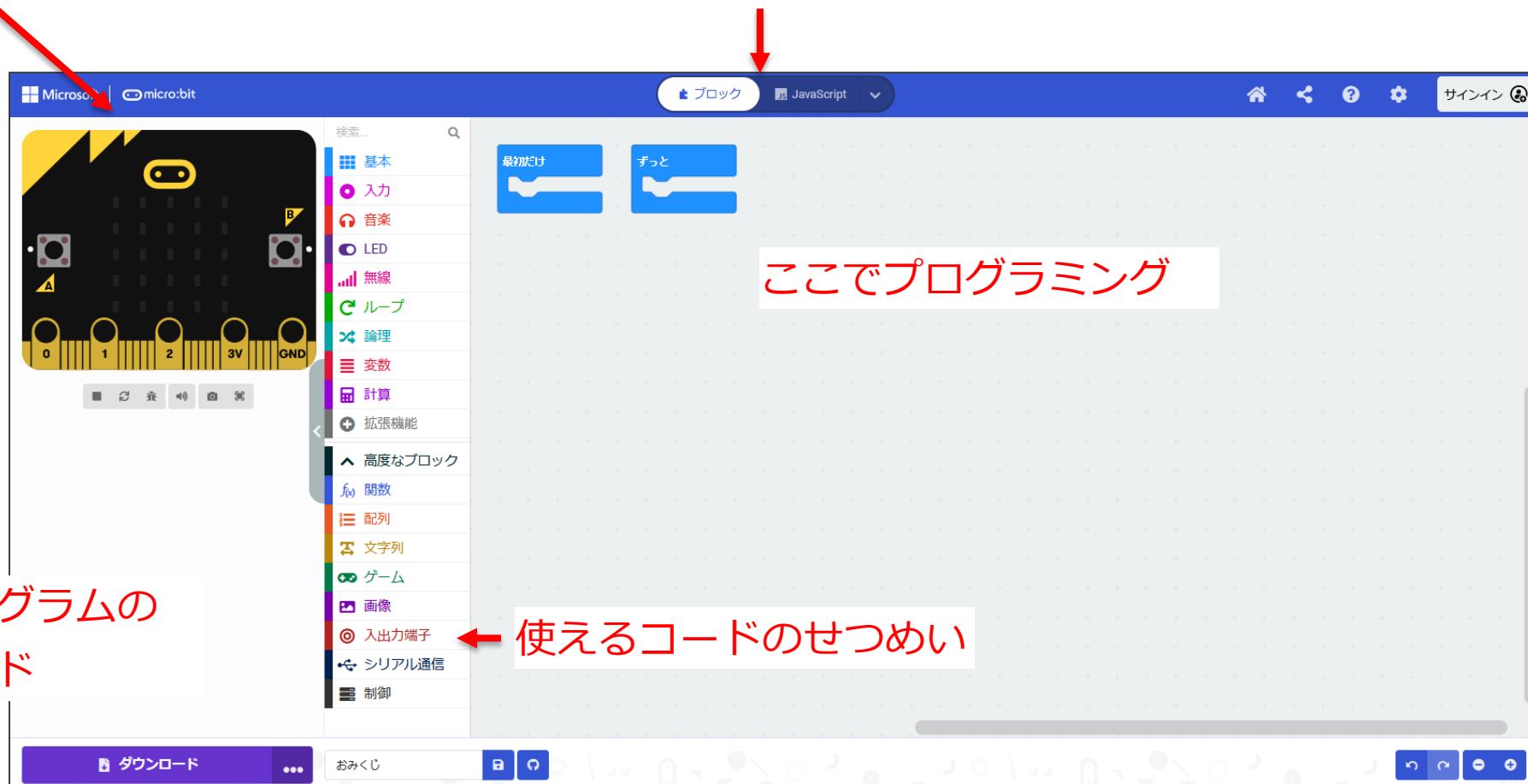


ここでプログラミングしていくよ

がめんのせつめい

micro:bit シミュレーター

プログラミングほうほうのきりかえ



作ったプログラムの
ダウンロード

ダウンロード



CoderDojo Tamachi

© CoderDojo Tamachi@VMware

プログラミングほうほう



ブロックをつかったほうほう
Scratchとおんなじだ

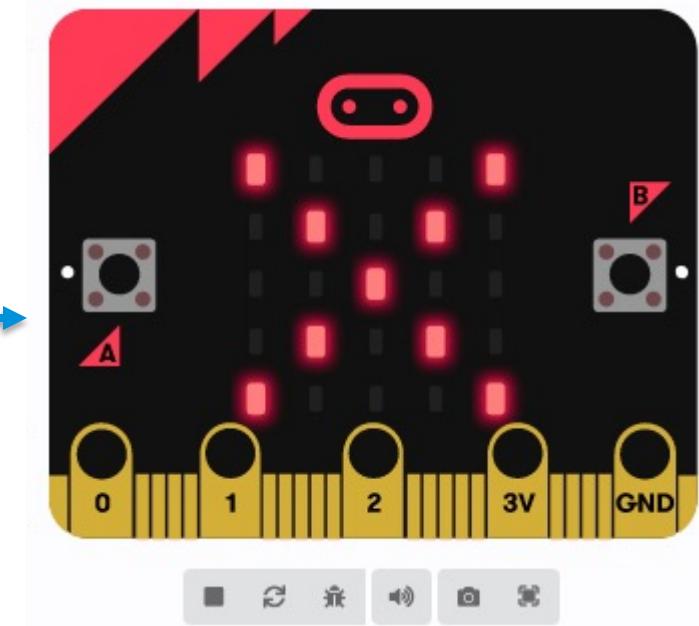
```
1 let omikuji = 0
2 input.onButtonPressed(Button.A, function () {
3     omikuji = randint(0, 2)
4     if (omikuji == 0) {
5         basic.showLeds(`
6             . # # #
7             # . . .
8             # . . .
9             # . . .
10            . # # #
11        `)
12     }
13 })
14 
```

JavaScriptをつかったほうほう
テトリスつくったときとおんなじだ

シミュレーターでためせるよ



Aボタンをおしたら、Xをひょうじするプログラムを作ったよ
上手にできるかな？





おみくじを作ってみよう

JavaScriptではどう考えたかな？

さいしょに「らんすう」を作っているね

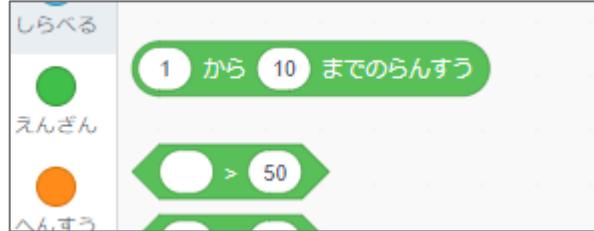
- ・「らんすう」はサイコロと考えよう

つぎに「らんすう」ででた数とおみくじを組み合わせているよ

- ・数が0なら大吉、1なら中吉、2なら小吉

```
1 // おみくじを引く呪文(じゅもん) (参考(さんこう) : 亂数生成(らんすうせいせい))
2 x = Math.floor(Math.random() * 3);
3
4 // ↓↓おみくじの中身 : ○○が出たらxx
5 if (x == 0) {
6   console.log("大吉!");
7 } else if (x == 1) {
8   console.log("中吉!");
9 } else if (x == 2) {
10  console.log("小吉!");
11 }
12 |
```

作ってみよう



けいさんのなかみを
「えんざん」っていうよ



もし、サイコロが1なら
大吉？小吉？

ぜんぶでで何しゅるいかな？

らんすうのけっかは
どこにしまっておけばいいかな？

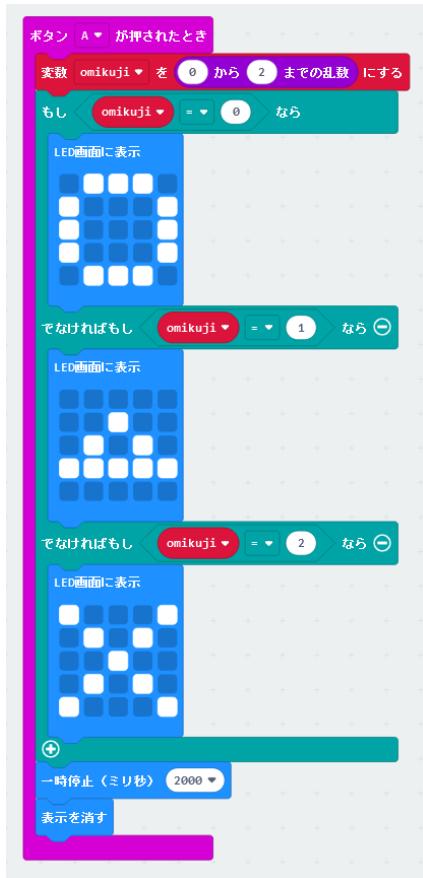


ひょうじ
けっかはどうやって表示しよう？
もじかな？ すうじかな？
それとも絵かな？

こたえを見るまえに考えてみてね



れい さくせい例



○、△、×にしてみたよ

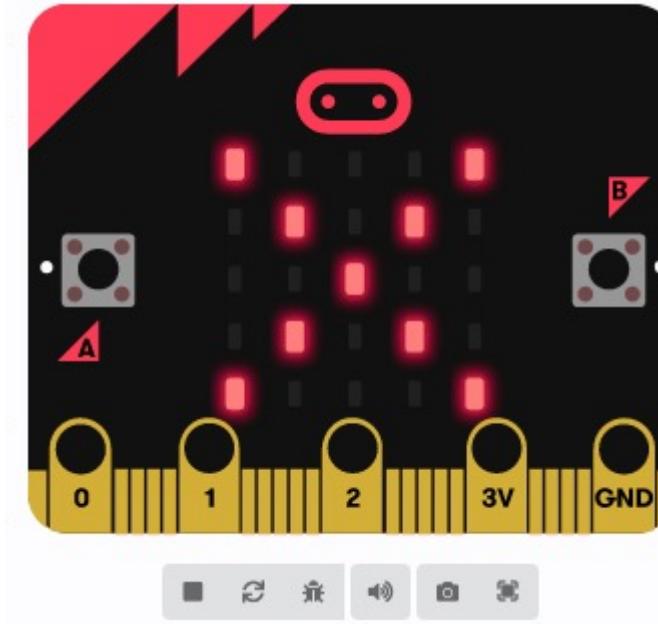
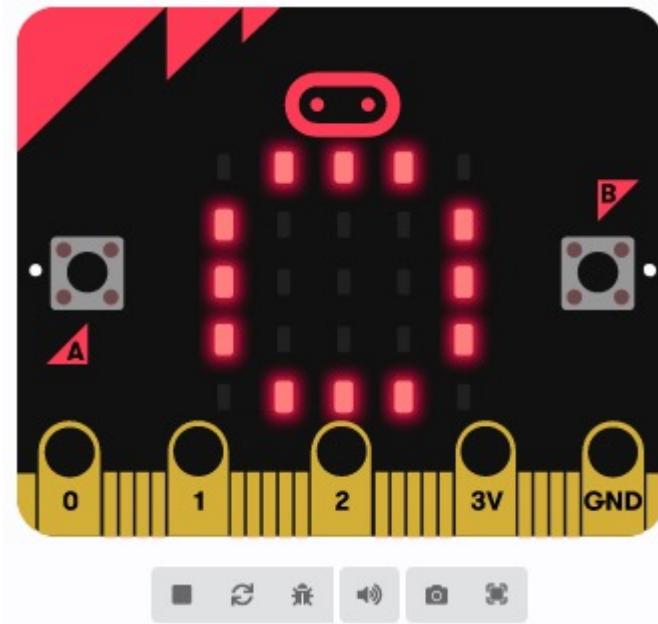
```
1 let omikuji = 0
2 input.onButtonPressed(Button.A, function () {
3     omikuji = randint(0, 2)
4     if (omikuji == 0) {
5         basic.showLeds(`
6             . # # .
7             # . . .
8             # . . .
9             # . . .
10            . # # .
11            `)
12     } else if (omikuji == 1) {
13         basic.showLeds(`
14             . . .
15             . . #
16             . # .
17             # # # #
18             . . .
19             `)
20     } else if (omikuji == 2) {
21         basic.showLeds(`
22             # . . #
23             . # . #
24             . . # .
25             . # . #
26             # . . #
27             `)
28     }
29     basic.pause(2000)
30     basic.clearScreen()
31 })
```

JavaScriptでかいたよ



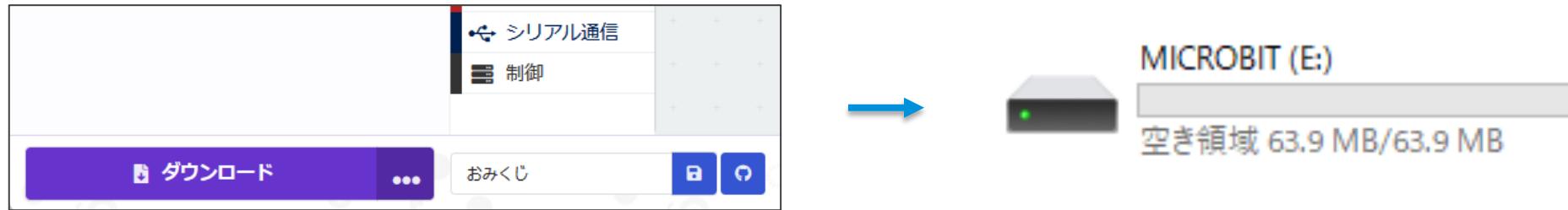
Good, so-so, Bad
のえいごができるようにしたよ

かんせいしたらテストしよう

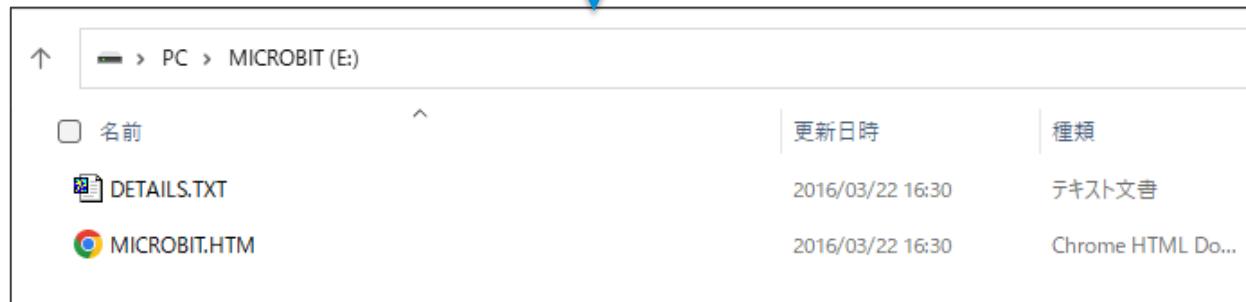


シミュレーターをつかって、うごいているかテストできるよ

micro:bitでためしてみよう



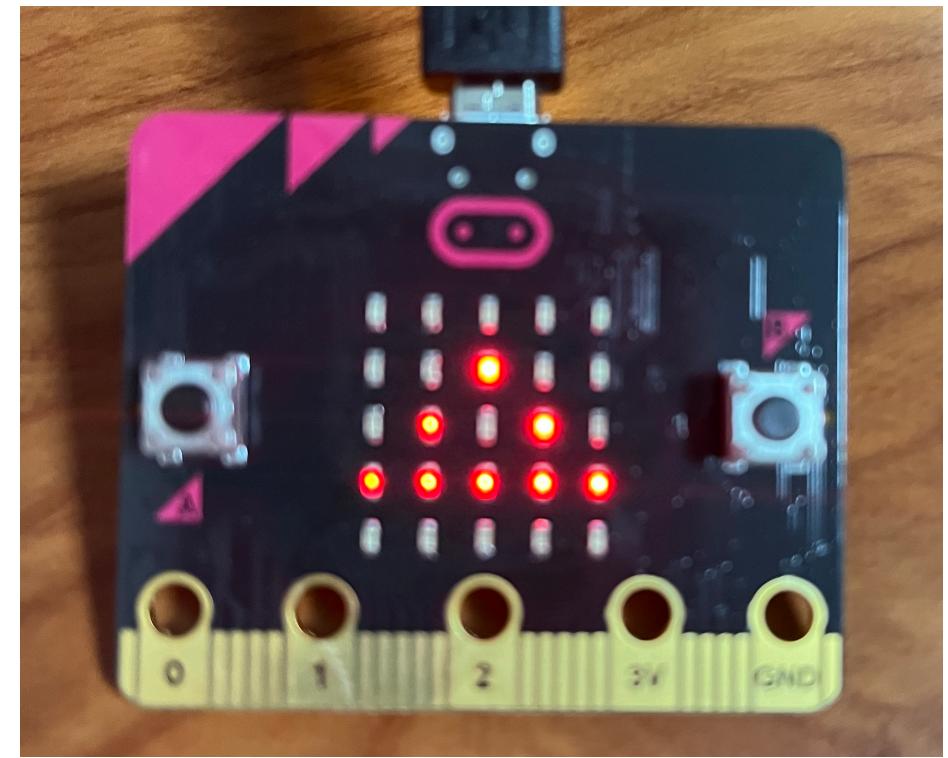
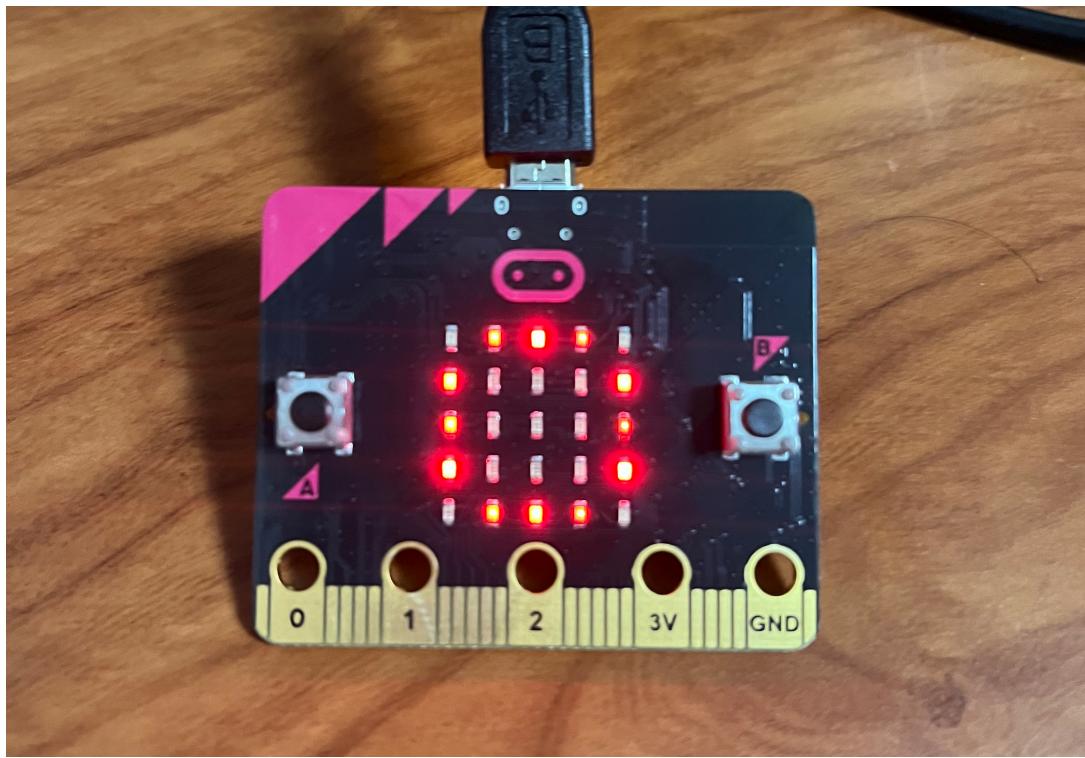
① ダウンロードをクリックして
つくったプログラムをパソコンにほぞんしよう



② micro:bitをUSBケーブルでパソコンにつなごう

③ ほぞんした〇〇.hexファイルを
micro:bitにコピーしよう
ファイルはみえなくなるけどだいじょうぶ！

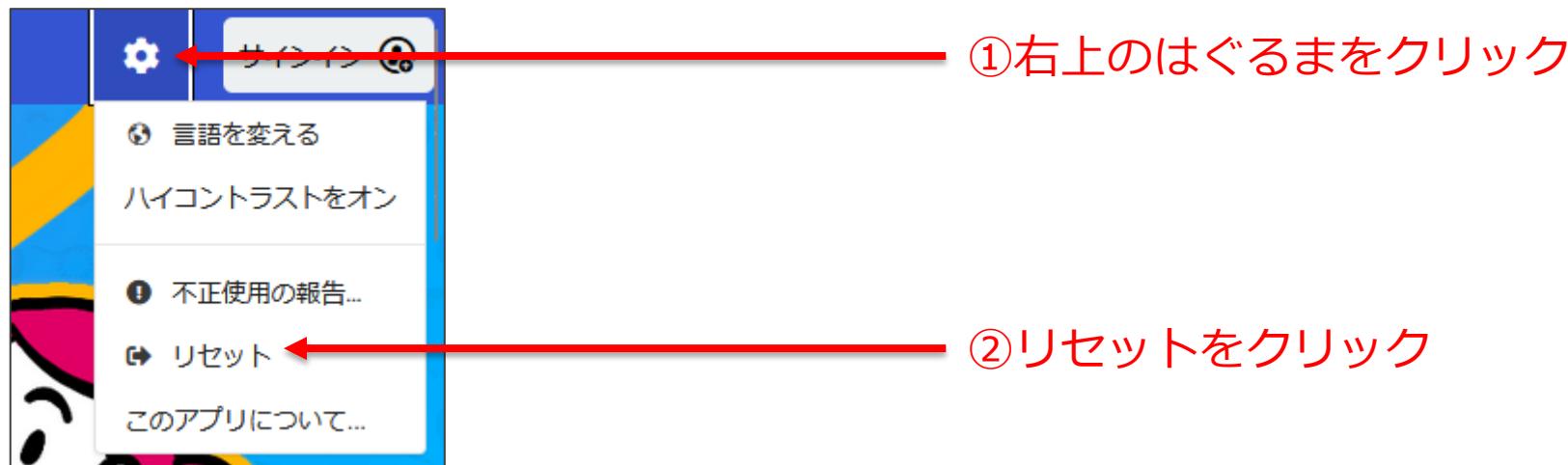
micro:bitでもうございた！





MakeCodeのとくべつなつかいかた

パソコンからデータをけしたい

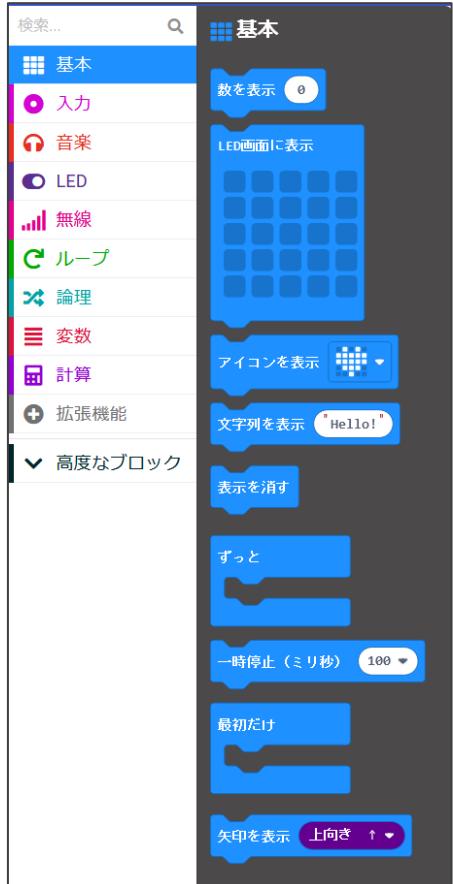


デバッグしたい

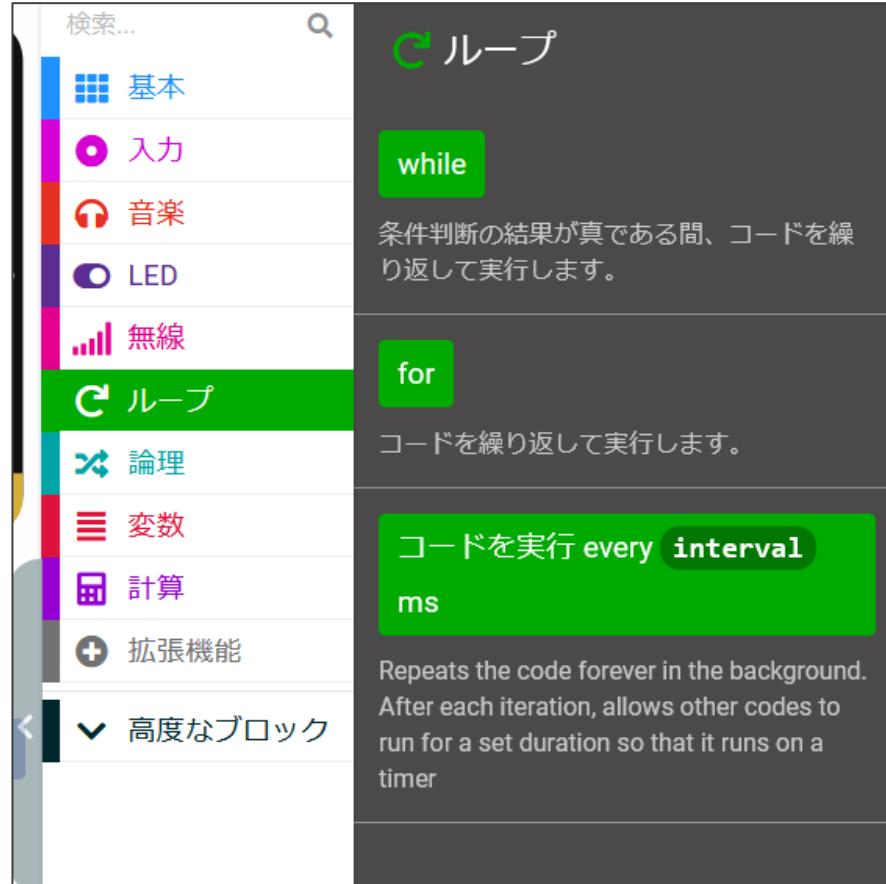


ゆっくりモードでは黄色い
わくで、実行しているめい
れいをかくにんできる

できることはリストからみれるよ



ブロックのしゅるい



JavaScriptのコード



JavaScriptはクリックすれば
書きかたのヒントができる



micro:bitでまなぶ JavaScript入門

micro:bitでまなぶ JavaScript入門 もくじ

1. Hello World!
2. 変数 (へんすう)
3. 定数 (ていすう)
4. データ型 (かた)
5. 演算 (えんざん)
6. 文字列結合 (もじれつけつごう)
7. 演算子 (えんざんし)
8. 条件分岐 (じょうけんぶんき)
9. 比較演算子 (ひかくえんざんし)
10. switch (スイッチ)
11. 配列 (はいれつ) とオブジェクト
12. くり返し (whileループ)
13. くり返し (forループ)
14. breakとcontinue
15. 関数 (かんすう) とreturn
16. 関数と引数 (ひきすう)
17. アルゴリズム
18. 組み込みオブジェクト・ライブラリ



1.Hello World!

LEDにHello World!を出してみよう

JavaScriptのConsoleにHello!を出してみよう

コーディングしてみよう

```
basic.forever(function () {
    basic.showString("Hello World!")
    console.log("Hello!")
})
```

※ConsoleはShow data シミュレーターでみれるよ

 Show data シミュレーター

2.変数(へんすう)

変数(へんすう)はデータのいれもの



- nameのハコをつくって、名前をいれよう
- 「let」をつかってハコをつくるよ
- うまく出せたら、別の名前をいれてみよう

※ConsoleはShow data シミュレーターでみれるよ

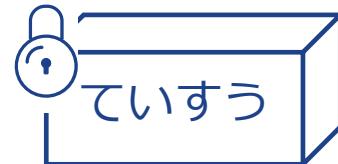
Show data シミュレーター

コーディングしてみよう

```
let name = "山本"  
console.log(name);  
name = "田中"  
console.log(name);
```

3. 定数(ていすう)

定数(ていすう)は1回だけデータを入れられる
いれものだよ



- nameのハコを「const」でつくるよ
- nameのハコに名前をいれよう
- nameのハコの名前をかえてみよう

コーディングしてみよう

```
const name = "田中";
console.log(name);
name = "山本"
console.log(name);
```

あれ？がめんにエラーがでているよ？
「read-only」は「読むだけ」って意味だね

問題 1

行 3: Cannot assign to 'name' because it is a constant or a read-only property.

4.データ型(かた)

いれものにはしゅるいがあるよ



ハコにさいしょに入れたデータでいれものの
しゅるいが決まるよ

文字のハコに小数をいれたら、おこられたよ

問題 1

行 9: Type '8.9' is not assignable to type 'string'.

コーディングしてみよう

```
let name = "田中";
let num = 123;
let dec = 456.7;
```

```
console.log(name);
console.log(num);
console.log(dec);
```

```
name = 8.9;
console.log(name);
```

5.演算(えんざん)

式を使って計算することを演算っていうよ
プログラミングでは足し算、引き算、かけ算、
わり算などを使うよ

演算は数だけじゃなく、数の入った変数と変数
でも計算できるよ

じょうずに演算できるか、ためしてみよう

※ConsoleはShow data シミュレーターでみれるよ

Show data シミュレーター

コーディングしてみよう

```
let plus = 123 + 456  
console.log(plus);
```

```
let num1 = 321  
let num2 = 654  
let add = num1 + num2  
console.log(add);
```

```
let num3 = 3;  
let num4 = 6;  
let div = num3 / num4;  
console.log(div);
```

6. 文字列結合(もじれつけつごう)

+の記号は足し算と文字列結合につかわれるよ

- ・足し算 $1 + 1 = 2$
- ・文字列結合 わたしは+元気=わたしは元気

足し算と文字列結合をいっしょにつかったら
どうなるのかな？ ためしてみよう

※ConsoleはShow data シミュレーターでみれるよ

Show data シミュレーター

コーディングしてみよう

```
let str1 = "今日は";
let str2 = "金";
let str3 = "曜日";

let joinStr1 = str1 + str2 + str3;
console.log(joinStr1);

let num1 = 30;
let joinStr2 = str2 + num1;
console.log(joinStr2);

let num2 = 70;
let joinStr3 = str2 + num1 + num2;
console.log(joinStr3);

let joinStr4 = num1 + num2 + str2;
console.log(joinStr4);

let joinStr5 = str2 + (num1 + num2);
console.log(joinStr5);
```

7. 演算子(えんざんし)

計算につかう記号

- ・足し算 + 引き算 -
- ・かけ算 *わり算 /
- ・わり算のあまり %

特別な計算

- ・1だけ足し算 ++ (インクリメント)
- ・1だけ引き算 -- (デクリメント)
- ・iを足し算 += i (num += i ってかくよ)
- ・iを引き算 -= i (num -= i ってかくよ)

コーディングしてみよう

```
let num = 100;  
  
num = num + 10;  
console.log(num);  
  
num = num - 20;  
console.log(num);  
  
num = num * 3;  
console.log(num);  
  
num = num / 6;  
console.log(num);  
  
num++;  
console.log(num);  
  
num--;  
console.log(num);
```

8. 条件分岐(じょうけんぶんき)

「晴れたら遠足、雨なら遠足中止」のように、
「もし～なら」を条件分岐というよ

条件分岐には3つの書きかたをよく使うよ

- if (もし～なら)
- else (それ以外なら)
- else if (それ以外でもし～なら)

書きかた

- if (条件) {条件がtrue (正しい) ときに行うこと}

コーディングしてみよう

```
let num = 10;  
if(num > 1){  
    console.log(num + "は1より大きい")  
} else {  
    console.log(num + "は1より小さい")  
}
```

9.比較演算子(ひかくえんざんし)

8.条件分岐のときに使う計算記号が比較演算子

① 同じとき	==
② ちがうとき	!=
③ ~より大きい	>
④ ~と同じか大きい	>=
⑤ ~より小さい	<
⑥ ~と同じか小さい	<=
⑦ 4.データ型まで同じ	====

※文字の“48”と数字の48をプログラミングではちがうものです。JavaScriptでは==を使うと、これを同じ(true)としてくれます

コーディングしてみよう

```
let num = 10;  
  
if(num < 10){  
    console.log("10より小さい")  
} else if(num > 10){  
    console.log("10より大きい")  
} else if(num == 10){  
    console.log("10と同じ")  
} else {  
    console.log("計算できませんでした")  
}
```

10.switch(スイッチ)

条件がいくつもあるときはswitchをつかうよ

- case (ケース) ごとにコードをかくよ
- 数字だけではなく、キー入力にもつかえるよ
- break (ブレーク) でcaseをおわりにできるよ
- caseにないもの（サイコロで10がでた！）は default (デフォルト) を行うよ

※randint

- randintはmicro:bitでつかえる乱数（らんすう）
- 亂数は“かずのどれか”をだしてくれるよ
- 他のJavaScriptでは下のようにかくよ

Math.round(Math.random() *6)

←0から5ができるよ

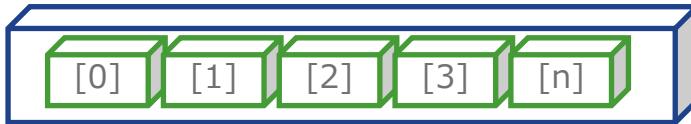
コーディングしてみよう

```
let dice = randint(1,6);
console.log("サイコロは"+ dice);

switch(dice){
    case 1:
        console.log("1歩進む");
        break;
    case 2:
        console.log("2歩進む");
        break;
    case 3:
        console.log("3歩進む");
        break;
    case 4:
        console.log("4歩進む");
        break;
    case 5:
        console.log("5歩進む");
        break;
    default:
        console.log("すきなだけ進む");
        break;
}
```

11.配列(はいれつ)とオブジェクト

配列では”ハコ”にたくさんデータが入るよ



- ・配列は[]でつくれるよ
- ・box[0]で1つめのハコを使えるよ
- ・なかみのないハコはundefinedになるよ
- ・.lengthをつけると、小さいハコの数がわかるよ
- ・.push(データ)で、小さいハコをふやせるよ

{ }をつかうとオブジェクトになるよ

- ・オブジェクトでは小さいハコに名前をつけられるよ
- ・名前は"key:"のように書くよ
- ・オブジェクトは変数.小さいハコの名前でよべるよ

コーディングしてみよう

```
//配列
let box = ["a", "b", "c"];
console.log(box);
console.log(box[0]);
console.log(box[1]);
console.log(box[100]);
console.log("boxの値は" + box.length + "こ");
//配列への追加
box.push("d");
console.log(box);
console.log("boxの値は" + box.length + "こ");
//オブジェクト
let box2 = {
  name: "Yamada",
  age: 10,
  gakunen: "四年生"
}
console.log(box2);
console.log(box2.name);
```

12.くり返し(whileループ)

おなじことをくり返すときはループをつかうよ

- while (ホワイル) をつかうほうほうだよ
- {}のあいだにプログラムをかくよ
- ()にどんなときに{}をおこなうかをかくよ
- むげんループにならないように気をつけよう

右のプログラムはこんなことをやっているよ

- numは1からスタート
- 1000以下のとき、kotae（答え）に回を足しているよ
 - kotae += num;
- 足したあと、回を1ふやしているよ
 - num++;

コーディングしてみよう

```
let num = 1;
let kotae = 0;

while (num <= 1000) {
    kotae += num;
    num++;
}

console.log("1から1000まで足すと
" + kotae);
console.log("numは" + num + "で終わっ
ているよ");
```

13.くり返し(forループ)

くり返しにはfor（フォー）ループもあるよ

- whileとは2つめのletのいちがちがうね
- let iはfor(){}の中だけでつかえるよ
- むげんループにならないように気をつけよう

ちがいをためしてみよう

- console.log(i); を足してみよう
- //をコメントアウトというよ
- コメントアウトは、プログラムで使われないよ

コーディングしてみよう

```
let kotaе = 0
for(let i=1; i<=100; i++){
    kotaе += i;
}
console.log("1から100まで足したら
" + kotaе);
// console.log(i);を足したらどうなるか
な?

//配列をつかったループ
let name_list = ["やまだ", "さとう", "
すずき"]
for (let name of name_list) {
    console.log(name);
}
```



14.breakとcontinue

breakはループやswitchをおわりにするよ

- breakのあとは実行されないよ
- ループもおわっちゃうよ
- 条件分岐でもつかうよ

continue（コンティニュー）はループからぬけるけど、おわらないよ

- continueのあとは実行されないよ
- ループそのものはおわらないよ
- 条件分岐といっしょにつかうよ

コーディングしてみよう

```
//breakでループをおわる
console.log("iのループはじめ");
for (let i = 1; i <= 10; i++) {
    console.log("ループの回数：" + i);
    break;
    console.log("これは見えるかな？：" + i);
}
console.log("iのループおわり");
//continueでループをぬける
console.log("jのループはじめ");
for (let j = 1; j <= 10; j++) {
    console.log("ループの回数：" + j);
    if(j != 5){
        continue;
    } else {
        console.log("これは見えるかな？：" + j);
    }
}
console.log("jのループおわり");
```

15. 関数(かんすう)とreturn

関数は計算してくれるハコ



- function 関数名(){やつてくれること}でかくよ
- 計算けっかはreturnでかえすよ
- returnは変数の他に、文字や計算結果もかえせるよ

条件をくみ合わせるときはAND, ORをつかうよ

- AND (あんど) : AとBのどっちも。"&&"ともかくよ
- OR (おあ) : AかBのどっちか。"||"ともかくよ

コーディングしてみよう

```
let age = 0
let name = ""
function gakunen () {
    let hantei;
    if (age < 6) {
        hantei = name+"は幼児です";
    } else if (age >= 6 && age < 13) {
        hantei = name+"は小学生です";
    } else {
        hantei = name+"は中学生以上です";
    }
    return hantei;
}
input.onButtonPressed(Button.A, function () {
    name = "山本さん"
    age = randint(1, 15)
    basic.showNumber(age)
    basic.pause(2000)
    basic.clearScreen()
})
input.onButtonPressed(Button.B, function () {
    console.log(gakunen());
})
```

16. 関数と引数(ひきすう)

関数にわたす値（あたい）を引数というよ

- 関数の名前(引数1, 引数2) でわたすよ
- 関数は下のようにして引数をうけとるよ
 - function 関数の名前 (引数のハコ1, 引数のハコ2)
- 引数のハコはなまみがなくても良いよ
 - micro:bitプログラミングではエラーとなるので、デフォルト値を設定するよ
 - name=0, name=1.1, name=""のようにデータ型を指定するよ
 - JavaScriptには可変長引数（かへんちょうひきすう）というのもあるよ

引数はどういうときにべんりかな？

- 関数を作るとき、わたされる変数の名前をしらなくて良いよ
- 引数のハコ（変数）は関数のそとに迷惑をかけないよ

コーディングしてみよう

```
let X = 0; //ヨコのLED。左下は0、右下は4
let Y = 4; //タテのLED。左下は4、左上は0
console.log(checkArea(X, Y)); //LEDの場所を
//チェック
led.plot(X, Y); //LEDのスタート地点を点灯

//すすんで良いかのチェック
function checkArea(nextX=0, nextY=0) { //引数
のハコnextX, nextYにおいて、デフォルト値0をいれて
おくよ
  if(nextX >= 0 && nextX <= 4 && nextY >=
0 && nextY <=4){
    return "OK";
  } else {
    return "NG";
  }
}

//Aボタンで右にすすむ
input.onButtonPressed(Button.A, function() {
  if(checkArea(X+1, Y) == "OK"){ //checkA
reaの引数をわたす
    X += 1; //チェック
    //OKなら右に進む
    led.plot(X, Y); //LEDを点
    灯
  }
})

//Bボタンで上にすすむ
input.onButtonPressed(Button.B, function() {
  if (checkArea(X, Y-1) == "OK") {
    Y -= 1;
```

17.アルゴリズム

アルゴリズムはもんだいをとくための方法

- ・計算もやり方もアルゴリズム
- ・背のひくいじゅんばんにならぶにはどうする？
- ・スーパー、本屋さん、花屋さん、どのじゅんにいく？

代表的なアルゴリズム

- | | |
|-------------|---------------|
| ・ソート | じゅんばんをならべかえるよ |
| ・探索（たんさく） | めいろをくりあするよ |
| ・暗号化（あんごうか） | ひみつのてがみをかけるよ |
| ・マシンラーニング | AIがべんきょうにつかうよ |

コーディングしてみよう

```
//配列（はいれつ）に数字を設定
let array = [100,115,90,120,100]
console.log(array);
//バブルソート
function bubble_sort(array = [0,0]){ //ふつうの
JavaScriptは(array)でOK
    for(let I = 0; I < array.length; i++){
        for(let j= array.length -1; j>I; j--){
            if(array[j] < array[j-1]){
                let temp = array[j];
                array[j] = array[j-1];
                array[j-1] = temp;
            }
        }
    }
    //ならべかえ
    bubble_sort(array);
    console.log(array);
}
```



18.組み込みオブジェクト・ライブラリ

ライブラリは図書館（としょかん）のいみ

- ・アルゴリズムをぜんぶつくるのはたいへん
- ・そんなときは、だれかがかいたものをかりよう

組み込みオブジェクト

- ・JavaScriptはさいしょから関数をもっているよ
- ・配列は.sortをつけるとならべかえをしてくれるよ
- ・.sortなどをメソッドというよ
- ・sort（ソート）はならびかえっていみ
- ・乱数につかったMathオブジェクトも組み込んだよ
- ・くわしいせつめいをここを見てね

https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference/Global_Objects

コーディングしてみよう

```
let array = [100, 115, 90, 120, 100]
console.log(array);

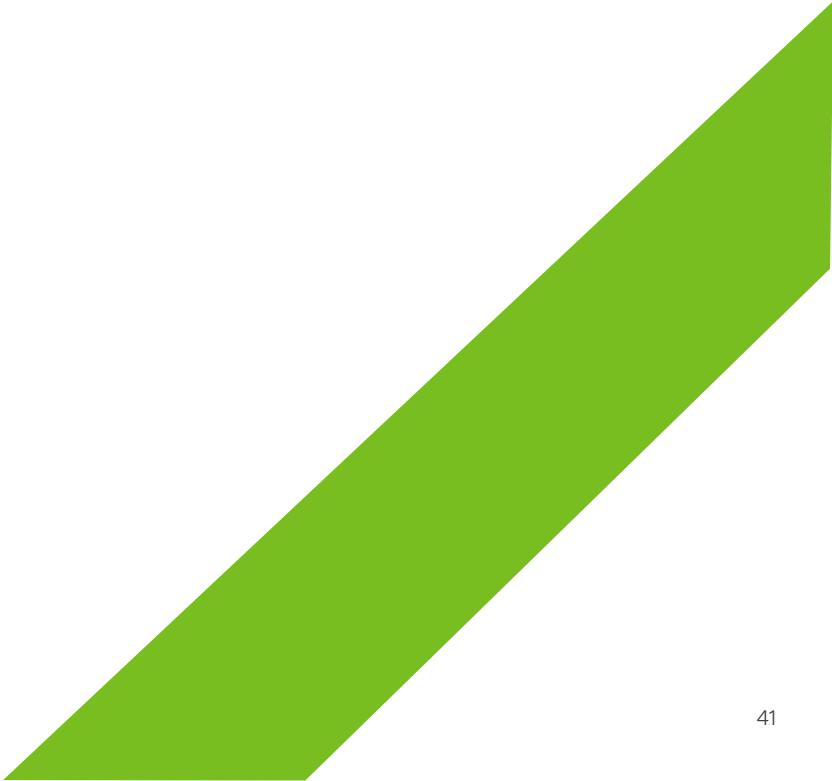
array.sort();    //文字としてソートしてるよ
console.log(array);

let array2 = [3,6,2,1,3,5]
array2.sort((a, b) => a - b);      //数字としてソートしてるよ

console.log(array2);
```



つくってみよう



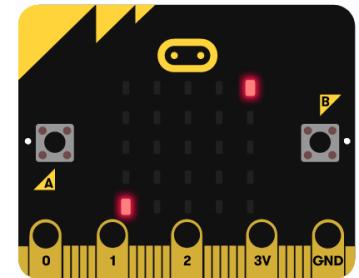
JavaScriptのきほんメモ

✓ ループ	✓ 配列	✓ 計算
✓ While	✓ let item = [1,2,3]	✓ + (足し算)
✓ for	✓ item[]	✓ - (引き算)
✓ 論理	✓ item[0]	✓ * (かけ算)
✓ if	✓ item.length	✓ / (わり算)
✓ else if	✓ item.push(4)	✓ % (わり算のあまり)
✓ else	✓ 関数	✓ 亂数
✓ && (and)	✓ function doSomething(){}	✓ randint(0,6)
✓ (or)	✓ doSomething()	✓ Math.round(Math.random() * 6)
✓ 変数	✓ doSomething(引数)	✓ データの型
✓ let (変数)		✓ let num = 0; //整数
✓ const (定数)		✓ let deci = 0.1; //小数
✓ item += 1, item++		✓ let str = "文字"; //文字
✓ item -= 1, item--		

“からだでプログラミング”をプログラムしてみよう

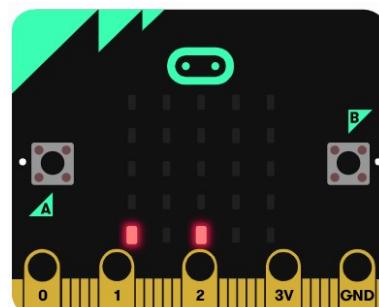
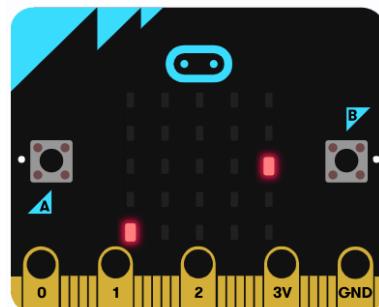
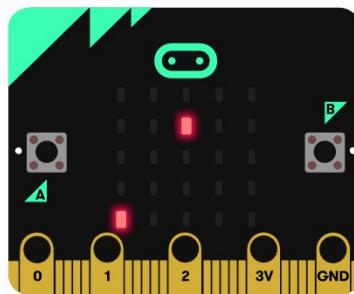
チャレンジ1.左下の点を動かして、右上のお菓子を取って戻ってこよう

- ・ルール1 点のうごきがみえるようにしてね
 - ・ルール2 点は1マスずつうごけるよ、ななめにはうごけないよ
 - ・ルール3 お菓子をとったら、LEDをちかちかさせてね



チャレンジ2.左下の点を動かして、お菓子を取ってこよう

- ・新ルール お菓子のあるところをランダムにしてね



チャレンジ1回答例

```
let mama = [0, 4] // [0]をヨコ、[1]を  
タテのいち  
let boku = [0, 4];  
let okashi = [4, 0] // [0]をヨコ、  
[1]をタテのいち  
led.plot(okashi[0], okashi[1])  
  
//Aボタンでおかしをとりにいく  
input.onButtonPressed(Button.A, function () {  
    led.plot(mama[0], mama[1])  
    basic.pause(1000)  
    while (mama[1] > okashi[1]) {  
        led.unplot(mama[0], mama[1])  
        mama[1] -= 1;  
        led.plot(mama[0], mama[1])  
        basic.pause(1000)  
    }  
    while (mama[0] < okashi[0]) {  
        led.unplot(mama[0], mama[1])  
        mama[0] += 1;  
        led.plot(mama[0], mama[1])  
        basic.pause(1000)  
    }  
    // console.log(mama);  
    // console.log(okashi);
```

```
if(mama[0] == okashi[0] && mama[1] == okashi[1]) {  
    led.unplot(okashi[0], okashi[1])  
    basic.pause(100)  
    led.plot(okashi[0], okashi[1])  
    basic.pause(100)  
    led.unplot(okashi[0], okashi[1])  
    basic.pause(100)  
    led.plot(okashi[0], okashi[1])  
}  
}  
}  
//Bボタンでおかしをもってかかる  
input.onButtonPressed(Button.B, function() {  
    while (mama[1] < boku[1]) {  
        led.unplot(mama[0], mama[1])  
        mama[1] += 1;  
        led.plot(mama[0], mama[1])  
        basic.pause(1000)  
    }  
    while (mama[0] > boku[0]) {  
        led.unplot(mama[0], mama[1])  
        mama[0] -= 1;  
        led.plot(mama[0], mama[1])  
        basic.pause(1000)  
    }
```

```
if(mama[0] == boku[0] && mama[1] == boku[1]) {  
    led.unplot(mama[0], mama[1])  
    basic.pause(200)  
    led.plot(mama[0], mama[1])  
    basic.pause(200)  
    led.unplot(mama[0], mama[1])  
    basic.pause(200)  
    led.plot(mama[0], mama[1])  
    basic.pause(200)  
}  
}  
//AボタンとBボタンでリセットする  
input.onButtonPressed(Button.AB, function() {  
    led.plot(okashi[0], okashi[1])  
    mama[0] = boku[0]  
    mama[1] = boku[1]  
    led.unplot(mama[0], mama[1])  
})
```

LEDをつけけしするべつの方法

```
input.onButtonPressed(Button.A, function () {
    basic.showLeds(`
        . . . .
        . . . .
        . . . .
        . . . .
        . . . .
    `)
})
```

```
input.onButtonPressed(Button.B, function () {
    basic.showLeds(`
        # # # # #
        # . . . #
        # . . . #
        # . . . #
        # # # # #
    `)
})
```

micro:bitチャレンジ

- Helloを出そう
- カタカナを出そう（拡張機能と半角カタカナをつかうよ）
- LEDにハートマークを出そう
- 温度を出そう
- 明るさを出そう
- Aボタンで温度、Bボタンで明るさを出そう
- 方位・方角を出そう
- 加速度を出そう

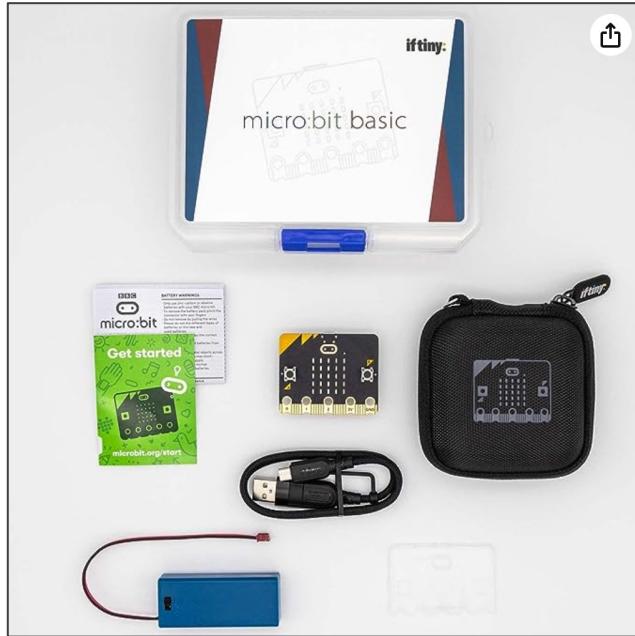


おうちでもやりたい！

おうちでつづきをプログラミングしたい



おうちでmicro:bitをうごかしたい！



micro:bitの標準機能

- Bluetoothアンテナ
- LED/光センサー
- 加速度センサー
- 温度センサー
- 地磁気センサー
- マイク・スピーカー（V2以降）
- タッチセンサー（V2以降）

Iftiny micro:bit basic (V2.21)

<https://www.amazon.co.jp/dp/B09364FXR3/>

- スピーカーも内蔵のV2.21チップ（最新）
- 利用時に必須となる電池ボックスとUSBケーブルを同梱



Tiny:bit じめんの線に沿って走れる

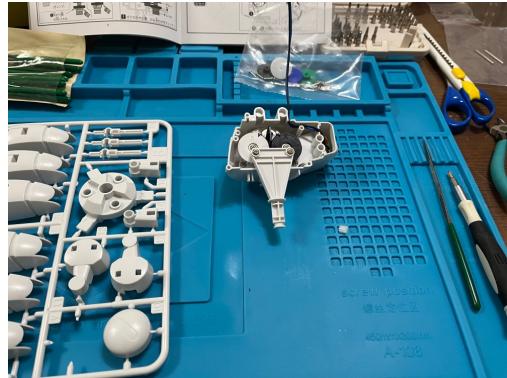


Furo：ロボットをうごかせる

ロボットをうごかしてみよう



プログラミング・フォロ
<https://sedu.link/5482/>



組み立てから自分でやるよ

どのパーツかな？

完成！じょうずにうごくかな？



Thank You