



ようこそ!

CoderDojoたまち@ヴィエムウェア



vmware®

しゃいん しゃかいこうけんかつどう

CoderDojoたまち@VMwareは、VMware社員の社会貢献活動として

かつどう むりょう

どうじょう

活動する無料の小中学生向けプログラミング道場です。



第4回プログラミング教室

～みんなでゲームを作ろう！
テトリス風パズルゲーム編～

2023/7/22 (土曜日) 13:30 – 16:30

スケジュール

- 13:30 – 13:40 あいさつ、ねが
ご挨拶、お願ひ
- 13:40 – 14:00 カラダでプログラミング！
- 14:00 – 14:20 おみくじプログラミング！

～きゅうけい（10分）～



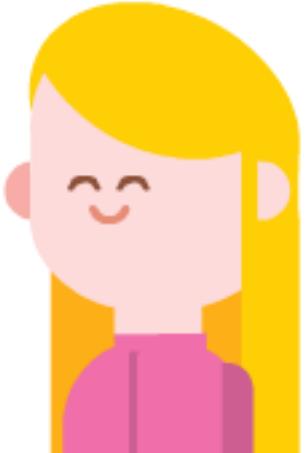
- 14:30 – 16:20 テトリス・プログラミング！
- 16:20 – 16:30 はっぴょうかい、しゅうごうしゃしん
発表会、集合写真



どうじょう よ かた

道場での呼び方

ニンジャ



さんか
参加いただいた
みなさん

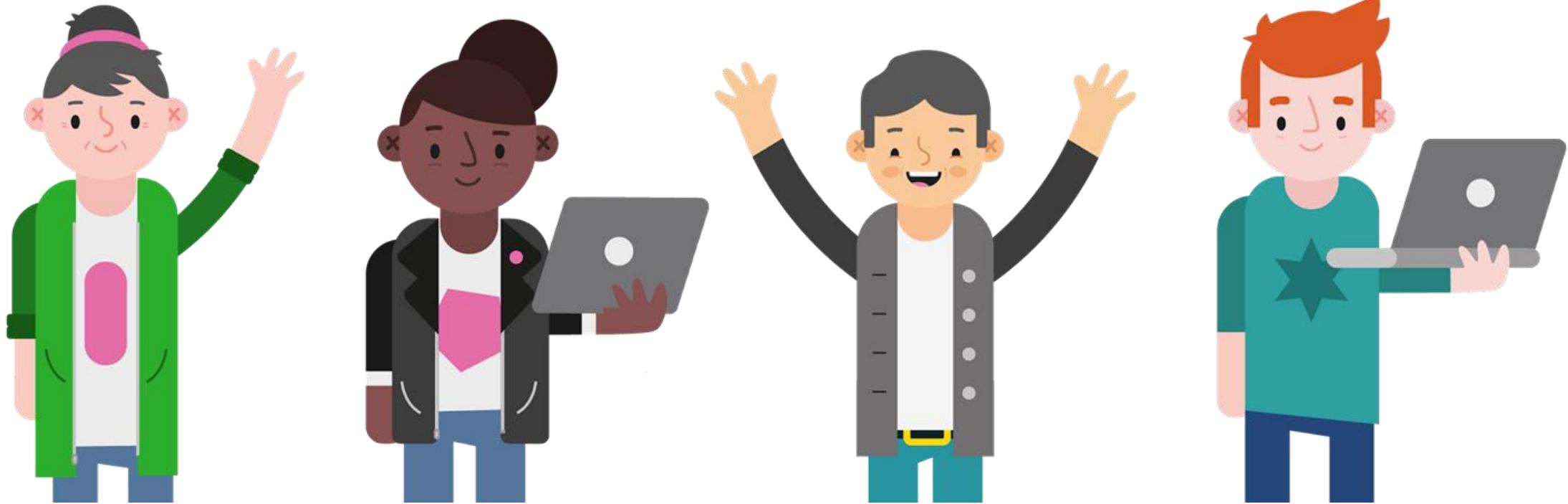
メンター



せんせい
先生

じこしょうかい
“ニンジャ”の自己紹介

お名前と何年生か教えてね！





おねがい
こどもむけ

おねがい

の もの

- 飲み物がこぼれないように、キャップ（ふた）をしめてね

たいちょう わる ばあい て おし

- 体調が悪い場合は、手をあげてすぐにメンターに教えてね



おねがい

● トイレに行きたいときはお父さんやお母さんと一緒に行ってね

かわ の もの
● のどが乾いたら飲み物をのんでもいいよ

ほか ひと
● 他の人のパソコンやマウスには触らないように
しようね

はっぴょう しず き
● 発表は静かに聞こうね



おねがい

● 困ったことがあれば、手をあげてメンターに教えてね



● 緊急時はメンターがみんなを安全な場所に案内するから、
落ち着いて指示に従ってね

● みんなとはすこし間をあけてね

● 名札シールは胸や腕など見える場所に貼っておいてね

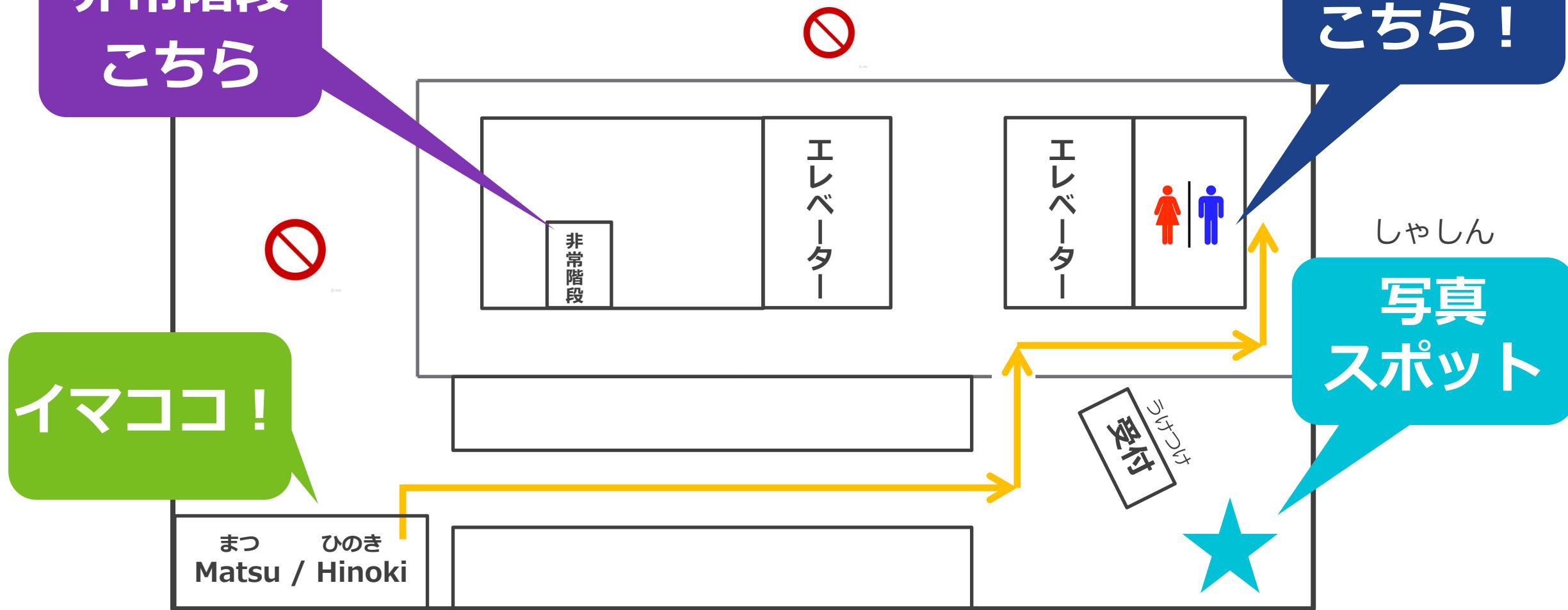
ひじょうかいだん

非常階段
こちら

かいじょうあんないす

かい

会場案内図(18階)



トイレは
こちら！

しゃしん

写真
スポット



ご連絡

保護者の方向け

保護者の方へのお願い

- ・ トイレに行かれる際は、玄関の開閉が必要なため社員がアテンドさせていただきます。お手数ですが、お気軽にお声がけくださいませ。
- ・ もし、お部屋以外に移動される場合はお声がけくださいませ。
- ・ 写真やビデオ撮影をして、VMwareの社外SNSや社内イントラ上で活動を公開する場合があります
- ・ 公開に同意されない方はメンターにご連絡ください
- ・ 皆様が写真やビデオ撮影する場合は、SNS投稿時に他の方の顔が映らないようにお願いします。
- ・ 緊急時はメンターが避難場所にご案内しますので、落ち着いて指示に従ってください。
- ・ プログラミングはお子様主体で実施をお願いいたします。

避難場所

芝浦公園が避難場所です（下図青枠）

赤枠は現在地/田町ステーションタワーN棟です





プログラミングとは？

コンピュータはどうやって動く？

ゲーム機きのコンピュータは、何をしたら動く？ 考えてみよう。



1) ボタンをおす

2) スティックを動かす

3) コントローラをかたむける

なぜボタンをおしたら、勇者は動くのかな？

コンピュータはコードという命令で動く めいれい

「ボタンをおしたら、勇者が攻撃する」という命令（コード）を送れるよ。



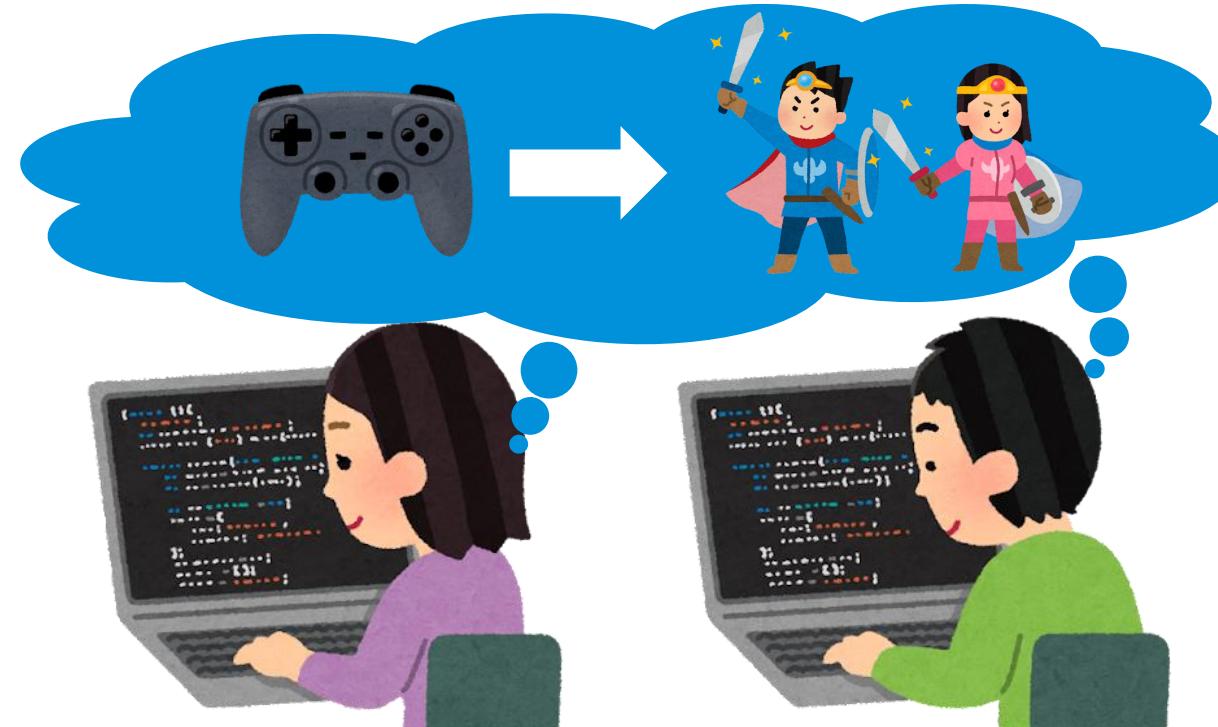
ないよう

コードの内容をかえれば、コンピュータの動きもかわるよ



プログラミング = コードを書く

勇者^{ゆうしゃ}が何の操作^{そうさ}で、どのように動かすか考えて、コードを書く
= プログラミング



プログラミングでコンピュータを動かそう

プログラミングをする役
めいれい
(命令をする役)



コンピュータの役
めいれい
(命令をされる役)

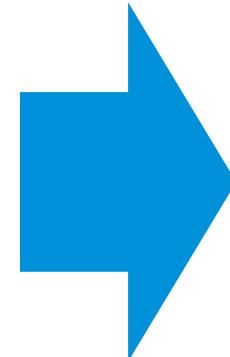


コンピュータを動かしてみよう

プログラミングをする役
めいれい
(命令をする役)

コンピュータの役
めいれい
(命令をされる役)

イスから立ち上がって、
歩いて



れいだい:のみものをとりに行ってもらう

コンピューター役にのみものをとりに行ってもらいましょう

のみものを正しくとりに行くにはどんな動きが必要かな？
ひつよう



れいだい:のみものをとりに行ってもらう

コンピューター役にのみものをとりに行ってもらいましょう

のみものを正しくとりに行くにはどんな動きが必要かな？

たとえば……

1. イスをひく
2. 立ち上がる
3. イスの前からテーブルまで歩く
4. 立ち止まる
5. 飲み物をつかむ
6. 後ろを向く
7. テーブルから自分のイスの前にもどる
8. 立ち止まる

9. イスにすわる
10. 飲み物をテーブルの上におく



もんだい：おかしをたべてもらう

村上
再考

1. イスをひく
2. 立ち上がる
3. イスの前からからテーブルまで歩く
4. 立ち止まる

この先にどんな動きが必要か考えて、書いてみよう！
書いたとおりに大人に動いてもらうよ



もんだい：おかしをたべてもらう

たとえば……

1. イスをひく
2. 立ち上がる
3. イスの前からからテーブルまで歩く
4. 立ち止まる
5. おかしをつかむ
6. 後ろを向く
7. テーブルから自分のイスの前にもどる
8. 立ち止まる
9. イスにすわる
10. おかしのふくろを開ける

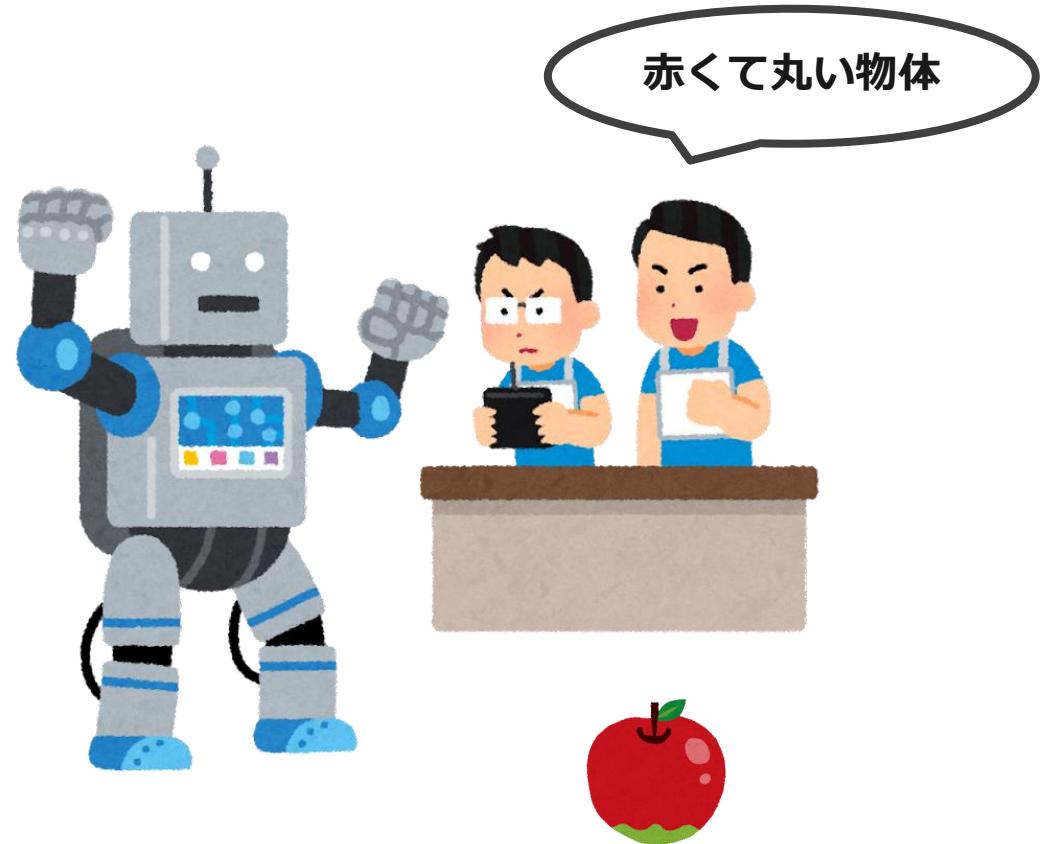
11. おかしを取り出す
12. おかしを口に入れる



しじ

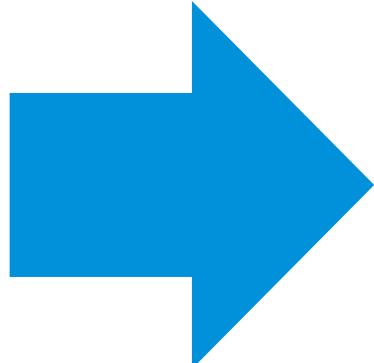
ひつよう

ロボットにはくわしい指示が必要

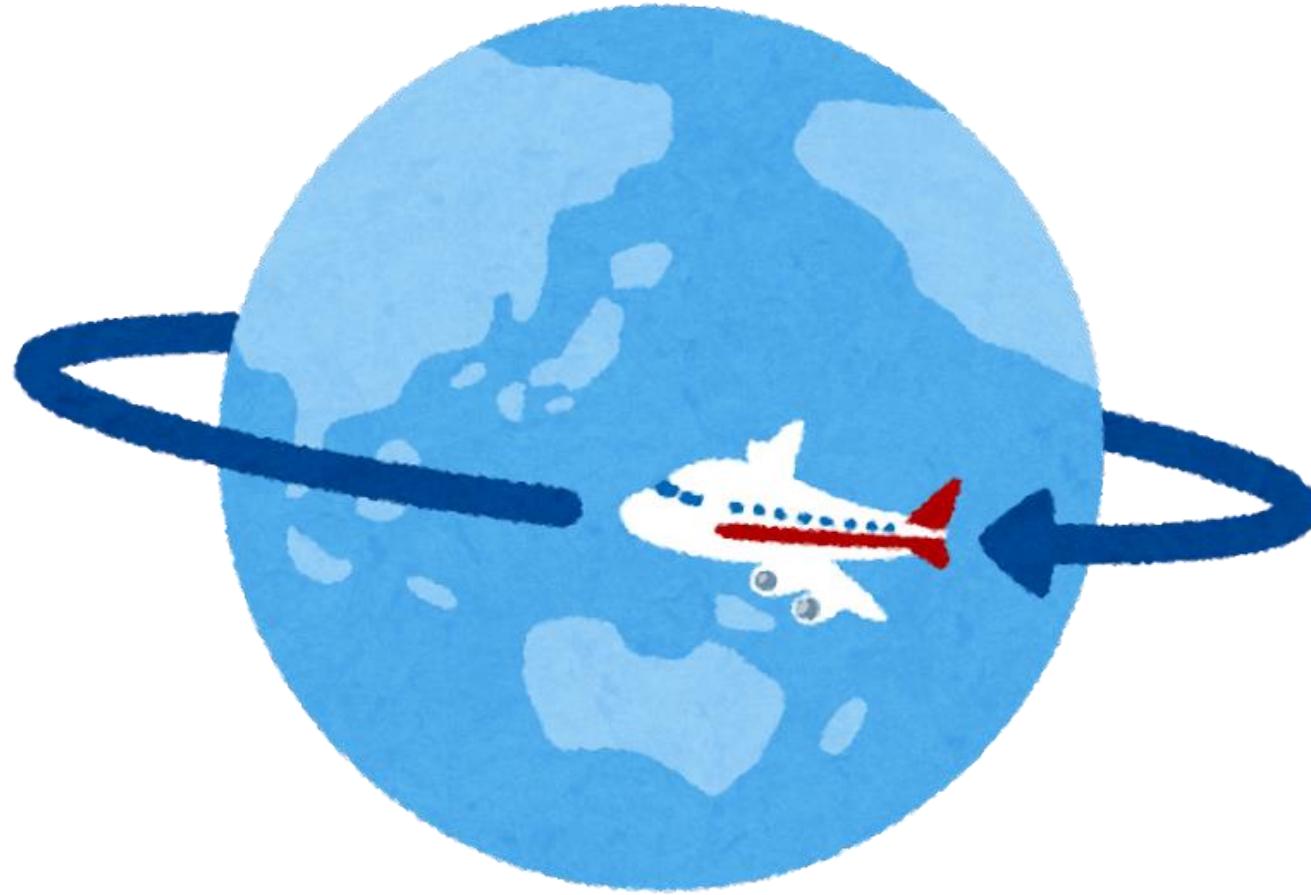


プログラミングを学べばよくなること

おそうじロボットもプログラミングで自動でうごくよ！



プログラミングで世界がもっとべんりに





じゅんび
準備をしよう

じゅんび
パソコンの準備をしよう

パソコンにログインしよう

はじめにパソコンにログインしよう

Ninja^{もじ} という文字の下に**パスワード**を入力してみよう

<パスワード>

vmware123

*パソコンからログアウトしたときも同じパスワードで入りなおせるよ

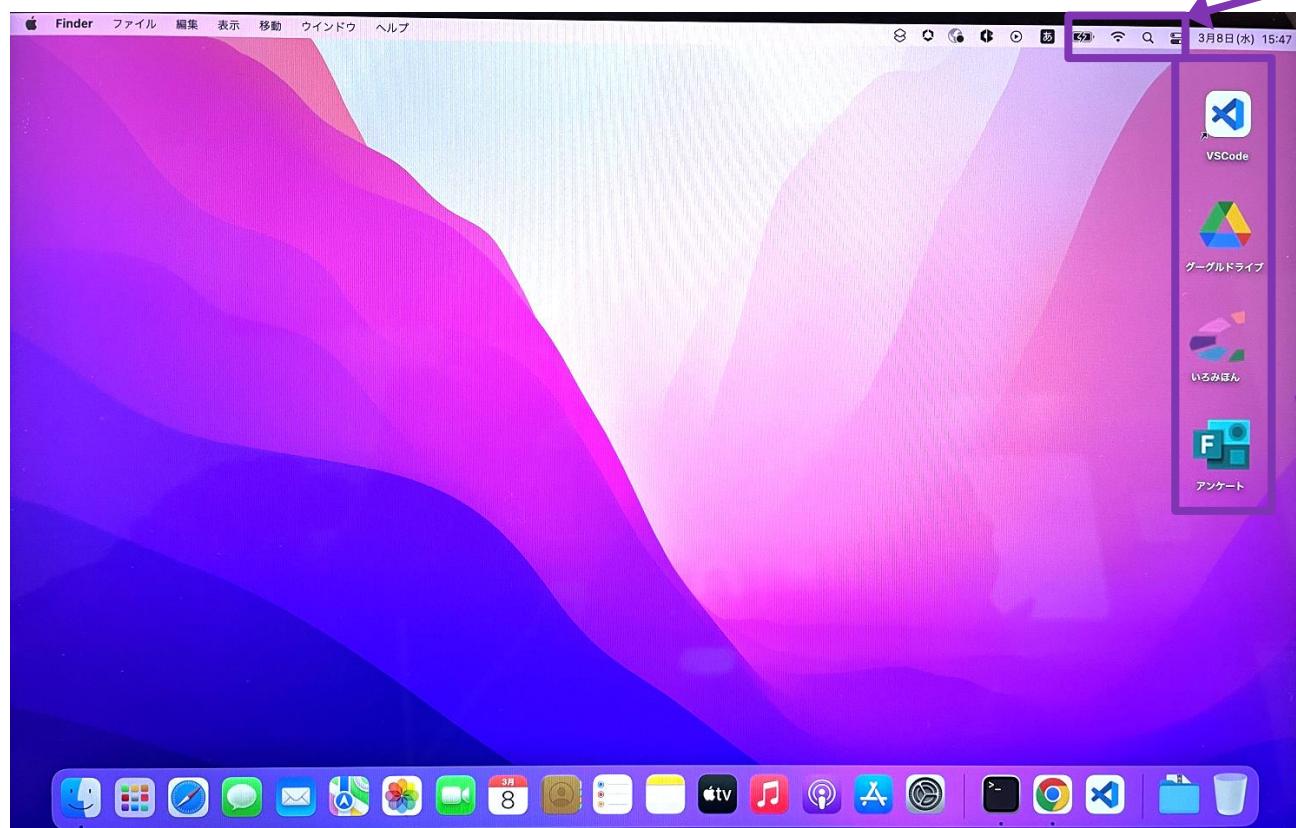


パソコンにログインしよう

がめん

パソコンにログインすると下のような画面になるよ

ログインしたときに
このメッセージが
ひょうじされても
もんだいはないよ



ワイファイ
Wi-Fiのマークはここだよ

ショートカット
アイコンはここにあるよ

ワイファイ

かくにん

Wi-Fiを確認しよう

画面の右上にWi-Fiのマーク  があるか見てみよう

Wi-Fiのマークが白色や黒色であればOK

もしグレー  になっていたらWi-Fiに接続しなおそう

よい例



わるい例



接続のしかたはつぎのページを見てね

ワイファイ

せつぞく

Wi-Fiに接続しよう

Wi-Fiマークがグレー  のときは、
Wi-Fiマークをクリックしてね

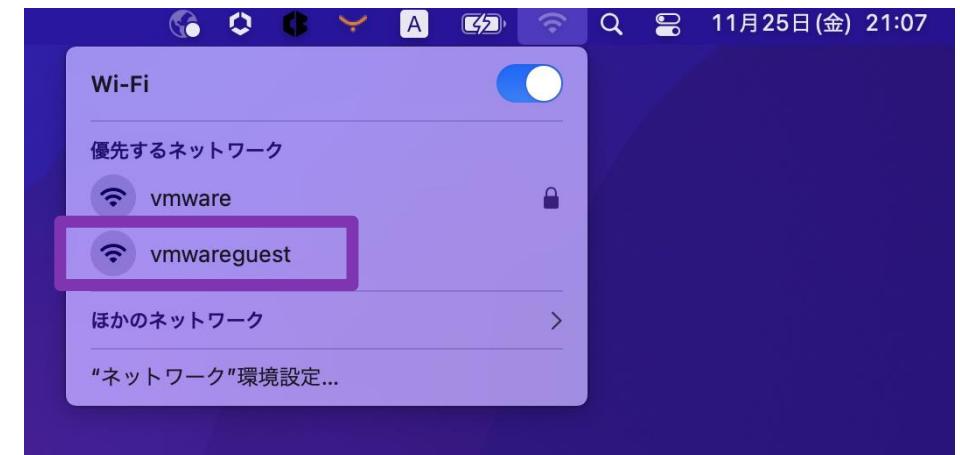
右の画面で **vmwareguest** をえらんで
アカウントとパスワードを入力しよう

<アカウント> <パスワード>

Event

vmware

Wi-Fiマークが白色や黒色  になればOK





そうさ

パソコンの操作について

ブラウザやデスクトップの操作について

つか

キーボードの入力 - よく使うキー

エスケープ [ESC]
とりけし
(取り消し)

タブ [TAB]
くぎ
(区切り)

シフト [Shift]
おおもじ・きごうにゅうりょく
(大文字・記号入力)

バックスペース [BS]
もど
もじけ
(戻る・1文字消す)

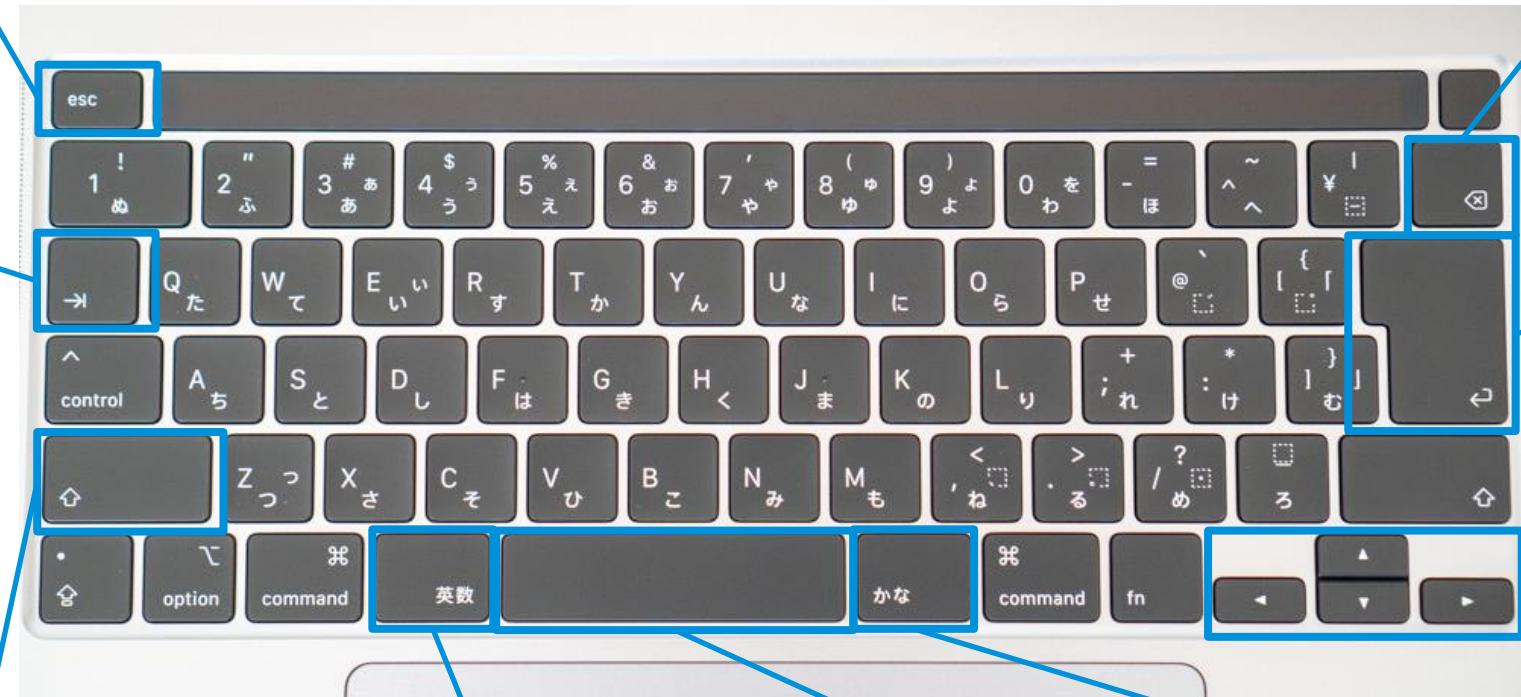
リターン [↵]
けってい
(決定)

十時キー [↑ ↓]
いどう
(移動する)

英数
えいすう にゅうりょく
(英数の入力)

スペース
もじ
(1文字あける)

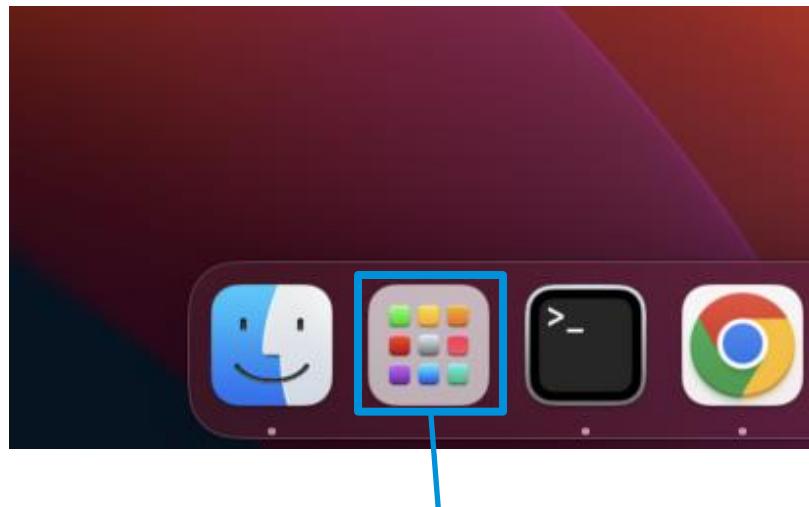
かな
にほんご にゅうりょ
(日本語の入力)



よく使うキー

もじ にゅうりょく
文字を入力してみよう

えら きどう
PC (Mac)の左下のLaunchpadから“メモ”を選んで起動



ここ(Launchpad)をクリック



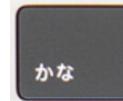
“メモ”をクリック

もじ にゅうりょく

文字の入力

かんじ にほんご にゅうりょく

漢字や日本語を入力するためには、かなキー



お にゅうりょく

押すと入力できるようになるよ！

えいご にゅうりょく

英語を入力するためには、英語キー



お にゅうりょく

押すと入力できるようになるよ！

さくせい
「メモを作成します」をクリック

じぶん なまえ えいご
ここに自分の名前を英語とひらがな、漢字で入力しましょう

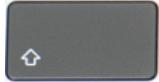
かんじ にゅうりょく
漢字で入力しましょう



きごういちらん

記号一覧

たとえば、”=” イコール記号を入力したい場合は、シフトキー



お押しながら - キー



お押します

記号	読み方	キー	記号	読み方	キー	記号	読み方	キー
!	びっくり (エクスクラメーション)	Shift+1	^	ベキ	^	*	アスタリスク	Shift+ :
”	ダブル・ クオテーション	Shift+2	~	チルダ	Shift+ ^	】	右中かっこ	】
#	シャープ	Shift+3	¥	イエン (バックスラッシュ) シユ)	¥	｝	右大かっこ	Shift+]
\$	ドル (ダラー)	Shift+4		パイプ	Shift+ ¥	,	アット	@
%	パーセント	Shift+5	@	アット	@	<	小なり	Shift+ @
&	アンド	Shift+6	`		Shift+ @	.	ピリオド	.
'	シングル・ クオテーション	Shift+7	[左中かっこ	[>	大なり	Shift+ .
(左かっこ	Shift+8	{	左大かっこ	Shift+ [/	スラッシュ	/
)	右かっこ	Shift+9	;	セミコロン	;	?	クエスチョン	Shift+ /
-	マイナス (ハイフン)	-	+	プラス	Shift+ ;	-	アンダスコア	-
=	イコール	Shift+ -	:	コロン	:			

参考 : JIS X 6002-1980, <https://ja.wikipedia.org/wiki/JIS%E3%82%AD%E3%83%BC%E3%83%9C%E3%83%BC%E3%83%89>
<https://www602.math.ryukoku.ac.jp/Prog1/charnames.html>

こゆう

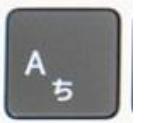
とくしゅ

Mac固有の特殊なキー

“⌘-a”と書かれている場合は、コマンドキー

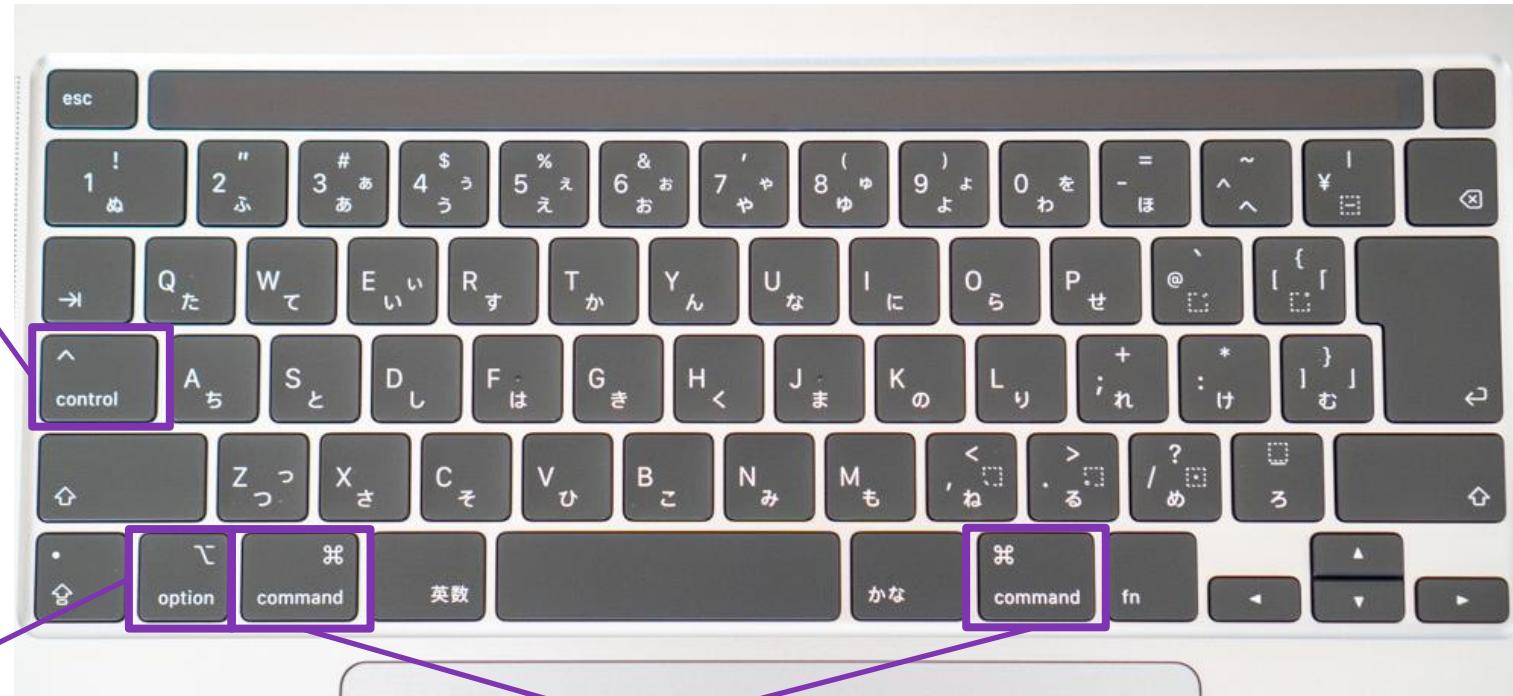


とAキー



同時に押します

コントロール [Ctrl]
(コントロール・キー)



オプション [⌘:Option]
(オプション・キー)

コマンド [⌘:Command]
(コマンド・キー)*左右どちら同じ

□ 他のキーと組み合わせて使うキー

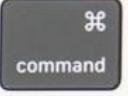
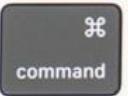
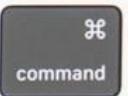
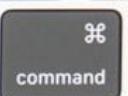
べんり

便利なショートカット

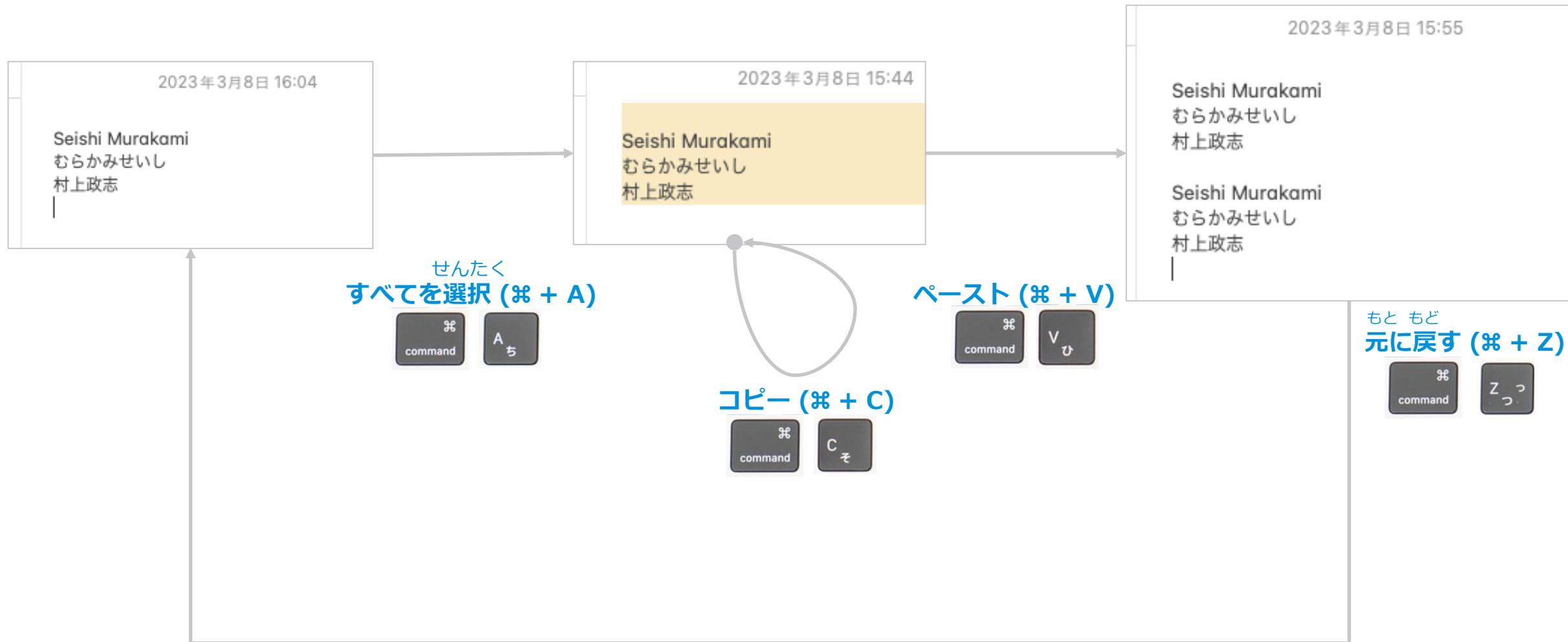
もじにゅうりょく べんり

おぼえ

文字入力に便利なショートカットを覚えよう！

ショートカット	実際のキー
すべてを選択 せんたく	⌘ + A  + 
コピー (複写) ふくしや	⌘ + C  + 
ペースト (貼り付け) はりつけ	⌘ + V  + 
カット (切り取り) きりとり	⌘ + X  + 
元に戻す もと もど	⌘ + Z  + 

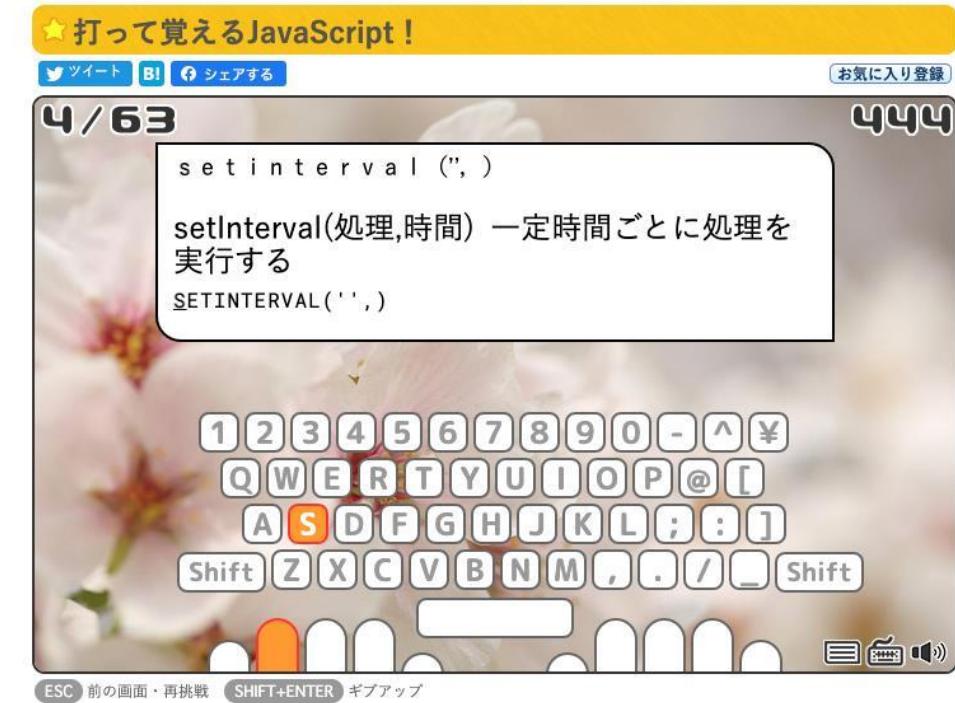
コピー & ペースト



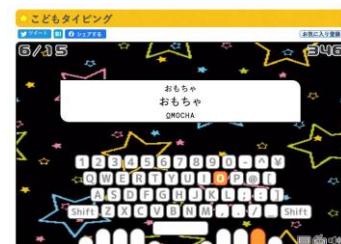
れんしゅう

タイピング練習してみよう！

- 打って覚えるJavascript！（記号入力）
 - デスクトップの”タイピング”というショートカットをクリックしてみましょう！
 - <https://typing.twi1.me/game/30503>



- 参考：こどもタイピング（ローマ字入力）
 - <https://typing.twi1.me/game/3880>





ヴィジュアル スタディオ コード

Visual Studio Codeについて

ブイエスコード

きどう

VSCodeを起動しよう

まずはVSCodeを起動してみよう

画面の右上にある VSCode の
アイコンを探してダブルクリックしてね



ブイエスコード

きどう

VSCodeを起動しよう

きどう

ソフトウェアが起動して

がめん

ひょうじ

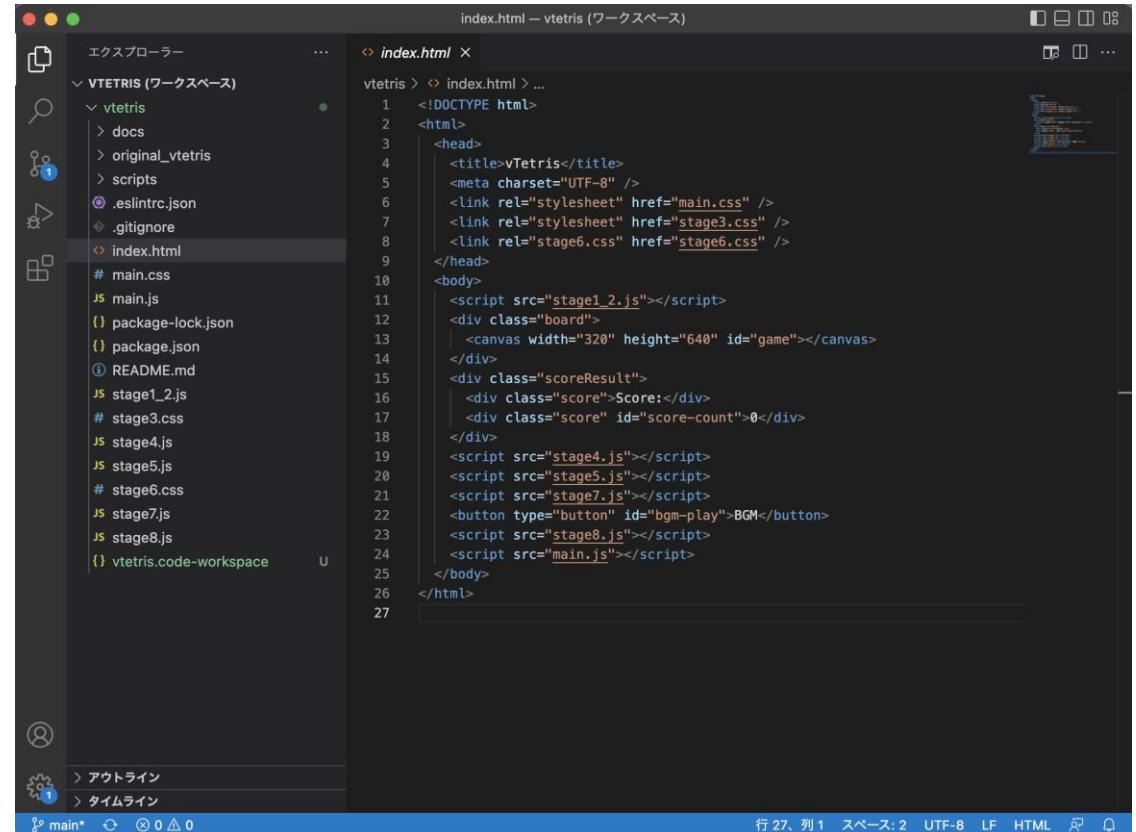
右のような画面が表示されることを

かくにん

確認してね

ブイエスコード

* このソフトウェアを VSCode というよ



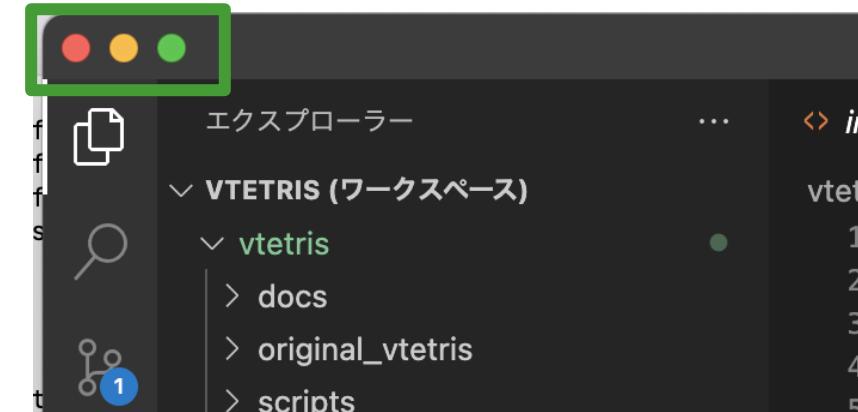
```
<!DOCTYPE html>
<html>
<head>
<title>vTetris</title>
<meta charset="UTF-8" />
<link rel="stylesheet" href="main.css" />
<link rel="stylesheet" href="stage3.css" />
<link rel="stylesheet" href="stage6.css" href="stage6.css" />
</head>
<body>
<script src="stage1_2.js"></script>
<div class="board">
<canvas width="320" height="640" id="game"></canvas>
</div>
<div class="scoreResult">
<div class="score">Score:</div>
<div class="score" id="score-count">0</div>
</div>
<script src="stage4.js"></script>
<script src="stage5.js"></script>
<script src="stage7.js"></script>
<button type="button" id="bgm-play">BGM</button>
<script src="stage8.js"></script>
<script src="main.js"></script>
</body>
</html>
```

ブイエスコード

きどう

VSCodeを起動しよう

ブイエスコード
VSCodeを大きく表示したい
ときは左上の緑色の ○ を
シングルクリックしてね



ブイエスコード
VSCodeを終了したいときは
キーボードの「Command」キーと
「Q」キーを同時にクリックしてね



ブイエスコード

きどう

VSCodeを起動しよう

ブイエスコード
VSCodeを一時的に最小化する

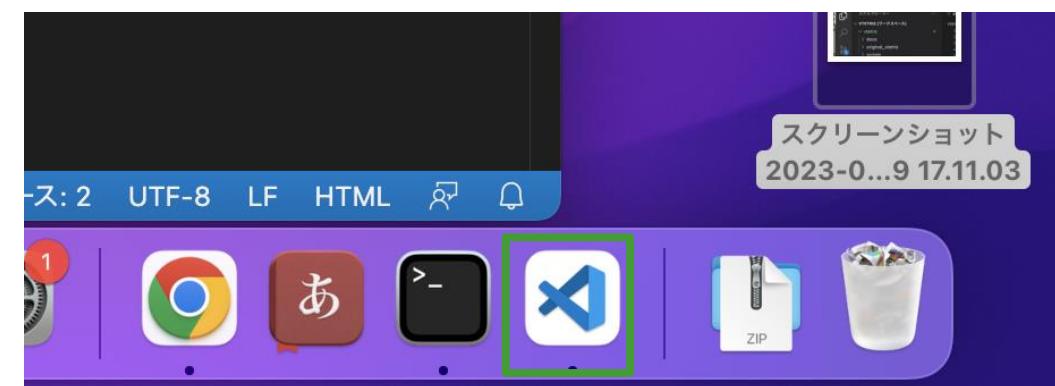
ときは左上の黄色の ○  を

シングルクリックしてね

これはデスクトップ画面を表示させたい

ときなどに使用するよ

最小化したVSCodeは画面の
右下のほうにいるよ



ブイエスコード

ひょうじ

ばいりつ

VSCodeの表示倍率について

ブイエスコード
がめん
VSCODEの画面の倍率を

か
変えることもできるよ

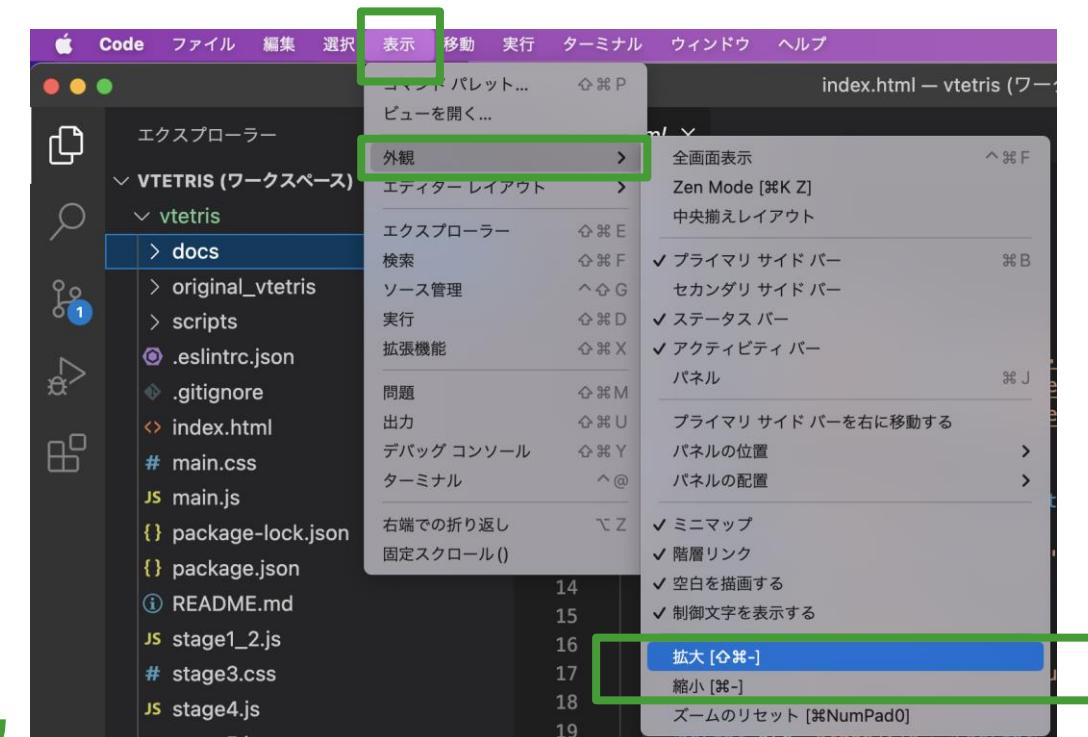
もじ
文字が小さかったり

がめん
ゲーム画面が大きすぎて画面から
しよう
はみ出しているときに使用してね

がめん
画面上の表示をクリックして

がいかん
かくだい
外観 の **拡大** または **縮小** をクリック

へんこう
すると変更できるよ



おんりょう

パソコンの音量について

おと
ゲームの音などが大きいときは
おんりょう
パソコンの音量を下げるね

画面の右上の  のマークを
シングルクリックして
サウンドのバーを**左にずらせば**
音が小さくなるよ





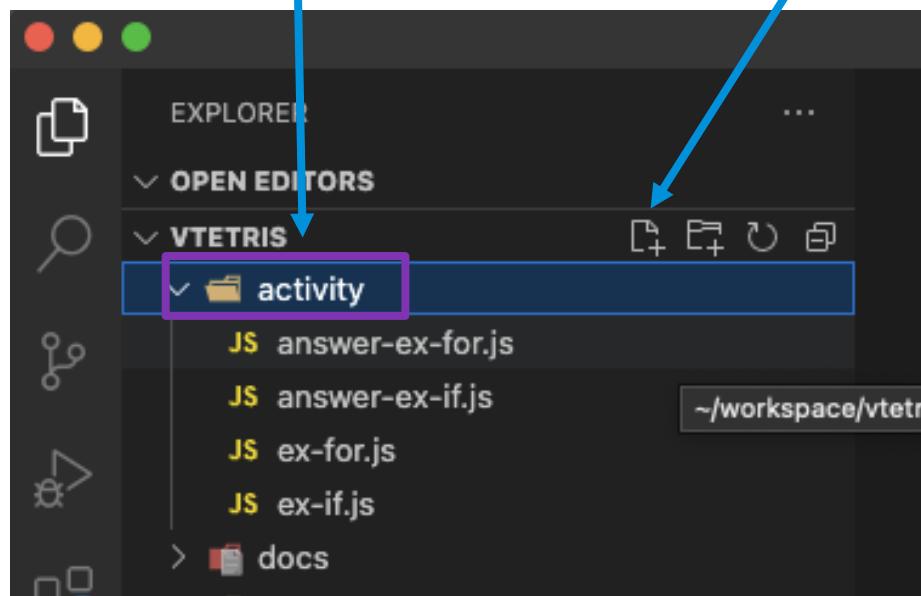
プログラミングはじめ！

プログラムを書いて！うごかしてみよう！

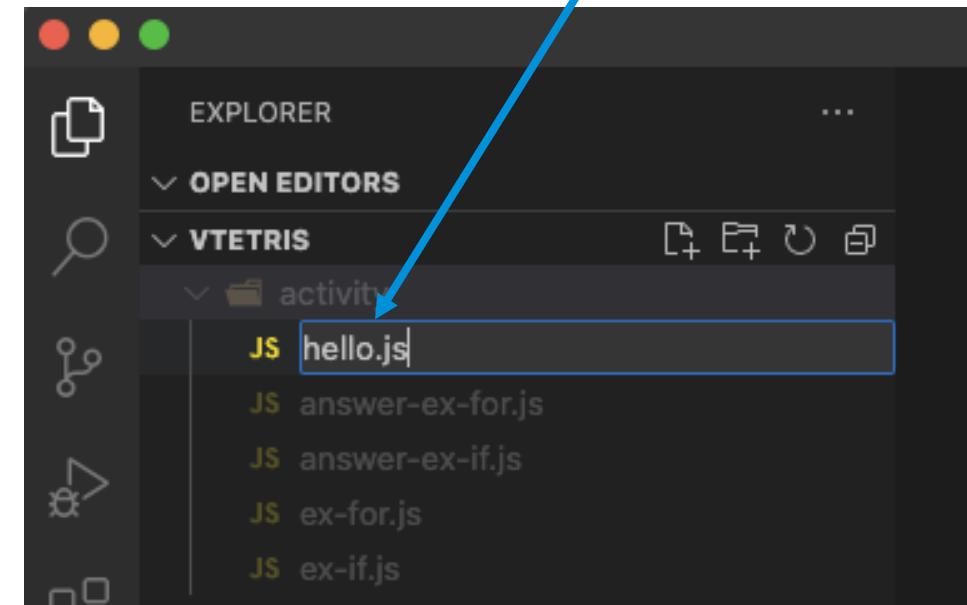
プログラムをつくる！

ブイエスコード
VSCodeからNew Fileをクリックして、新規ファイルを
作成するよ

アクティビティ
① 「activity」 をクリック

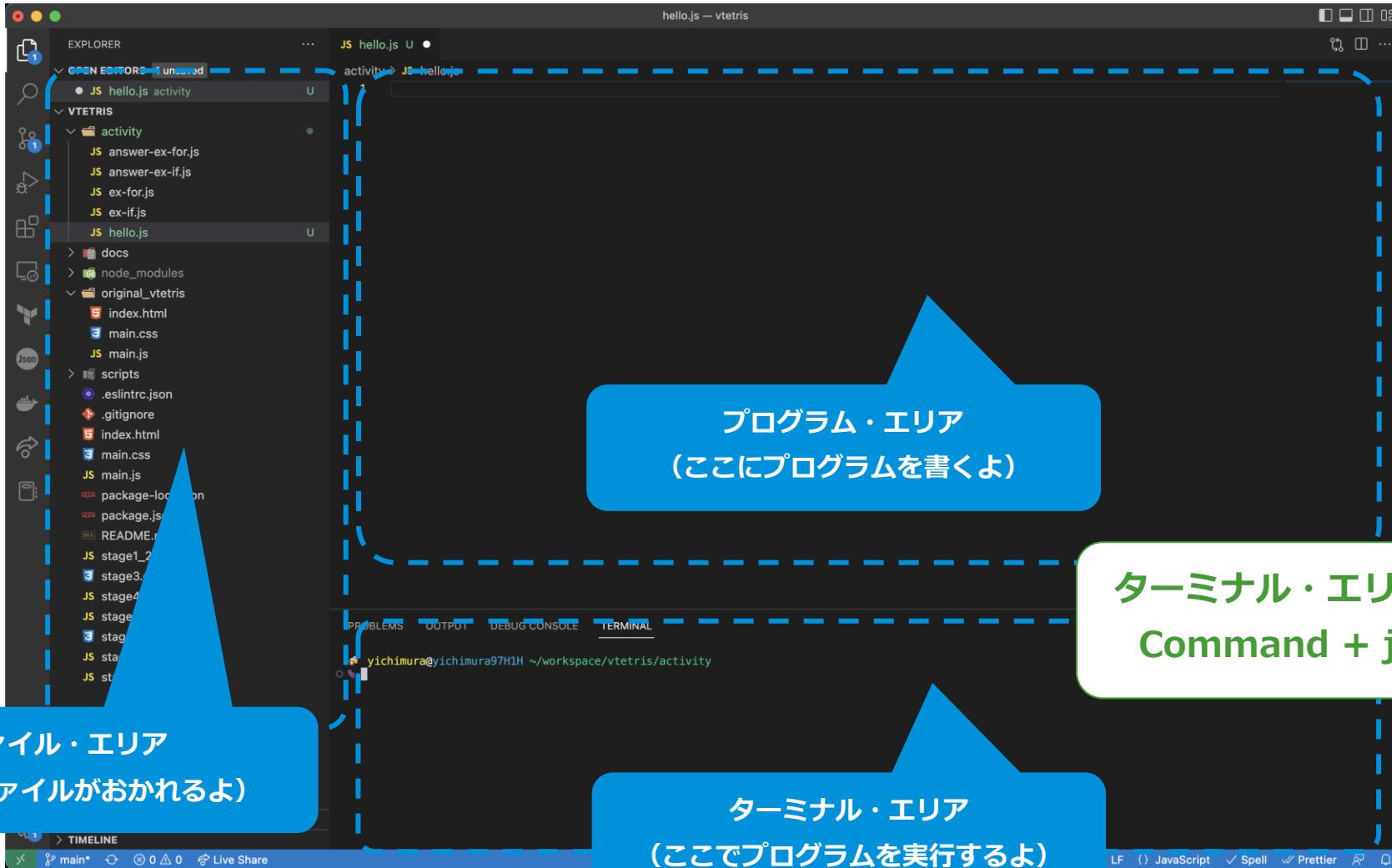


ニューファイル
② 「New File」 をクリック

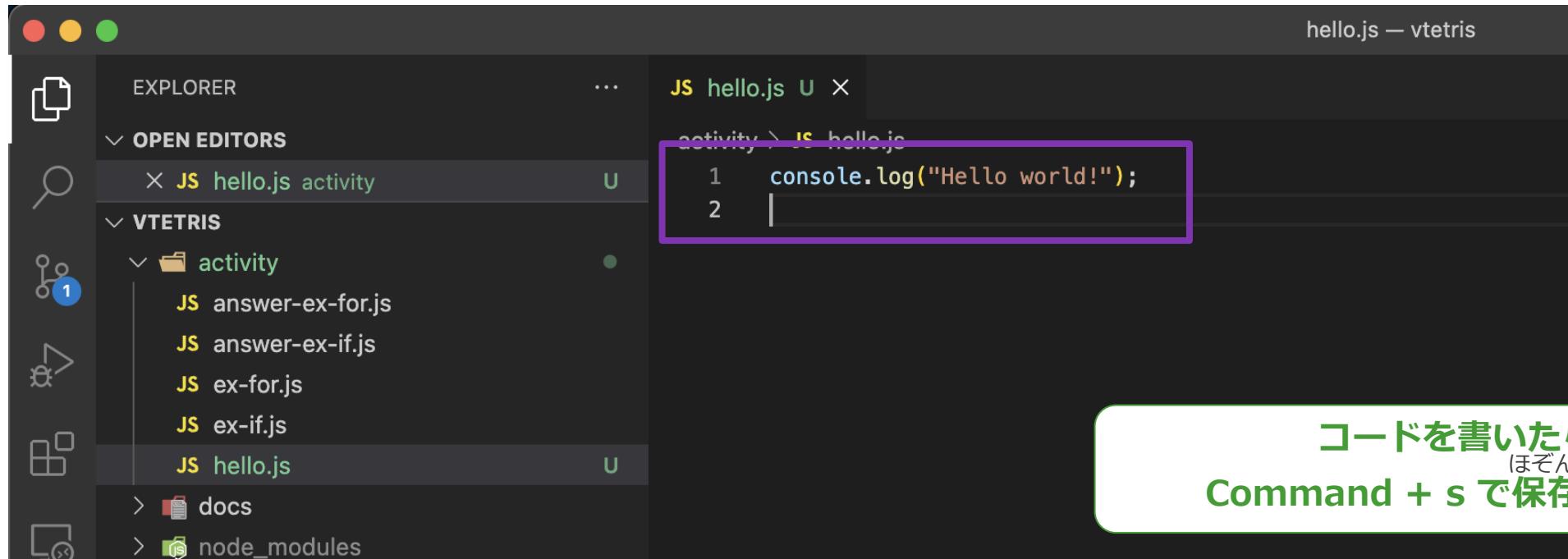


③ 「hello.js」 を入力しEnter

プログラムを書いたり動かしたりするところ



さあ、プログラムを書くよ！



```
JS hello.js U X
activity > JS hello.js
1 console.log("Hello world!");
2 |
```

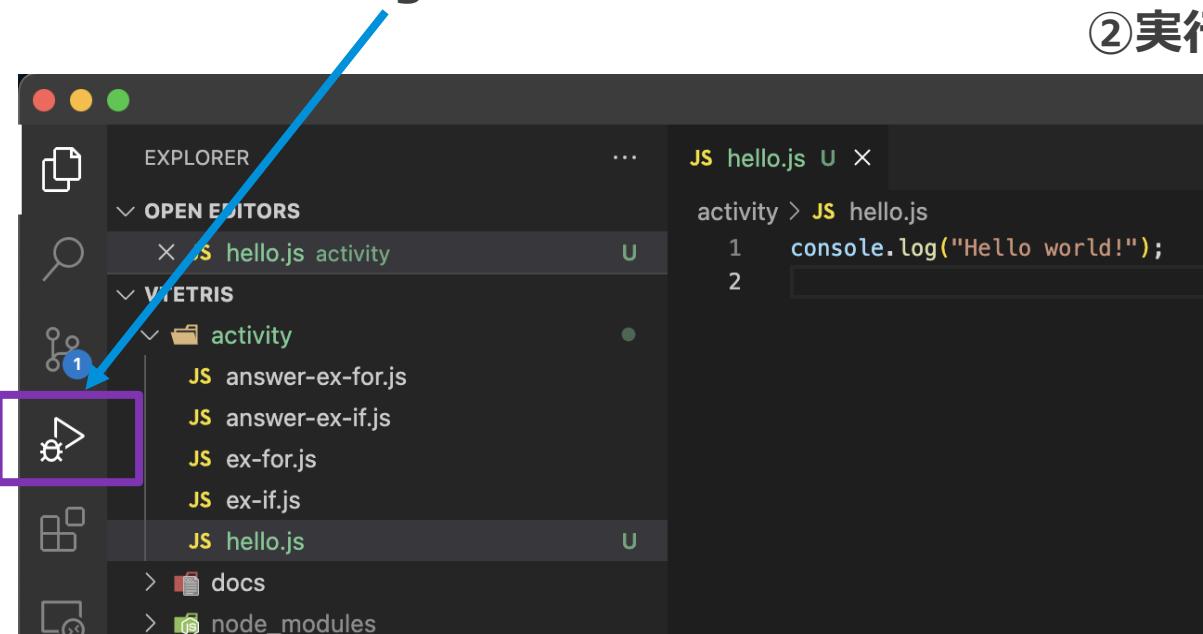
コードを書いたら
Command + s で保存できるよ

プログラムエリアの1行目にコードを書いてみよう！

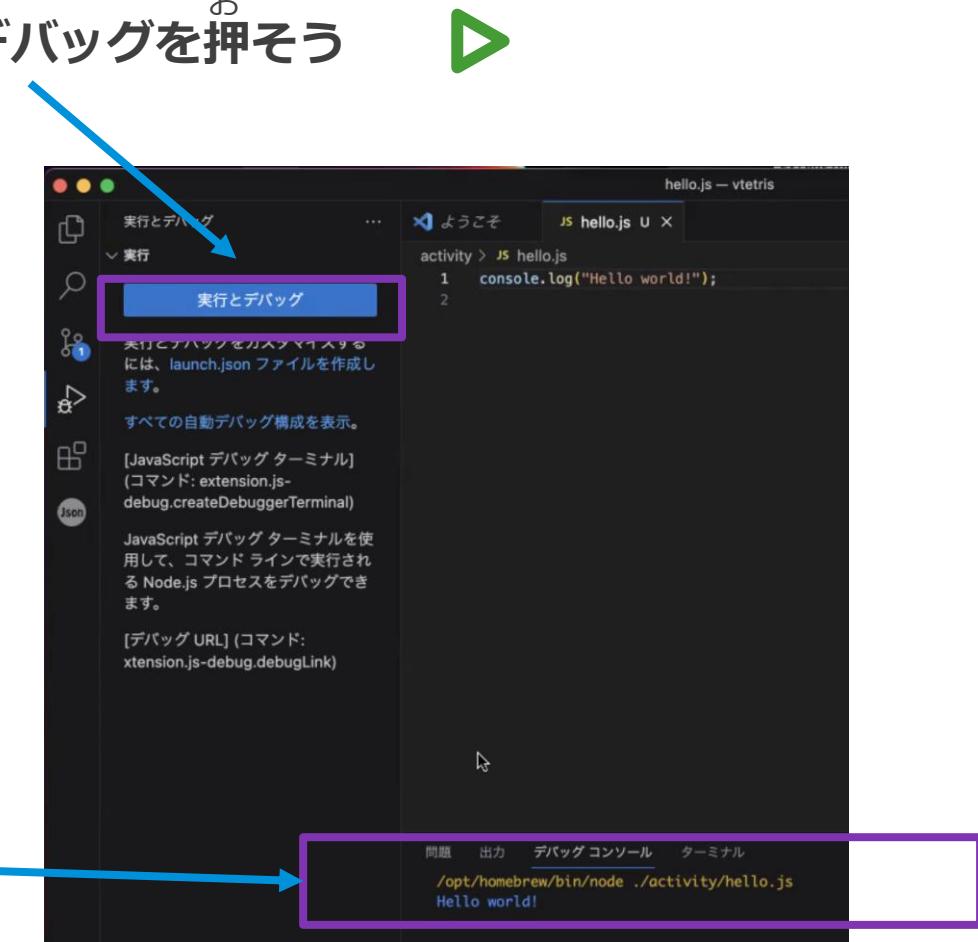
`console.log("Hello world!");`

さあ、プログラムを動かそう

① Run and Debug ボタンを押そう



②実行とデバッグを押そう



③実行結果が表示される

もしエラーがでたら！？

A screenshot of a Node.js development environment. The code editor shows a file named `hello.js` with the following content:

```
const hello = "Hello world!";
console.log(hellp);
```

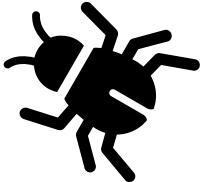
The word `hellp` is highlighted in red, indicating a spelling error. The terminal window below shows the output of running the script:

```
/opt/homebrew/bin/node ./activity/hello.js
Uncaught ReferenceError ReferenceError: hellp is not defined
at <anonymous> (/Users/yichimura/workspace/vtetris/activity/hello.js:2:13)
at Module._compile (internal/modules/cjs/loader:1246:14)
at Module._extensions..js (internal/modules/cjs/loader:1103:3)
at Module.load (internal/modules/cjs/loader:942:1)
at executeUserEntryPoint (internal/modules/run_main:1:4)
at <anonymous> (internal/main/run_main_module:23:4)
Process exited with code 1
```

Two blue callout boxes highlight specific parts of the interface:

- A box on the left points to the error message in the terminal: "こんなエラーがでたら、なにかプログラムがまちがつてるとかも" (If an error like this appears, does it mean there's a mistake in the program?).
- A larger box on the right points to the error message in the terminal and the variable name in the code editor: "hellp という変数がないと 言ってるよ" (It says there is no variable called hellp). It also includes a comparison: "hello ○" (correct) and "hellp ×" (incorrect).

プログラムをなおす（デバッグ）



hello.js — vtetris

Run Cur ▾

VARIABLES

```
activity > JS hello.js > ...
1 const hello = "Hello world!";
2 console.log(hellp);
3
```

PROBLEMS DEBUG CONSOLE ... Filter (e.g. text, !exclude)

/opt/homebrew/bin/node ./activity/hello.js
Hello world!

WATCH

なおしたらもう一回 をクリックしてみよう

2行目がのhellpが
せってい
設定されていないってことは、、、、

まちが
helloとhellpで打ち間違えてたんだ！！



おみくじプログラミング

いろいろなうごきをまなぼう！



もう少し練習してみよう！

おみくじを作つてみよう

まず一步目

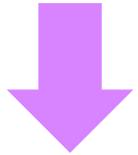
おみくじの仕組みを
考えてみよう



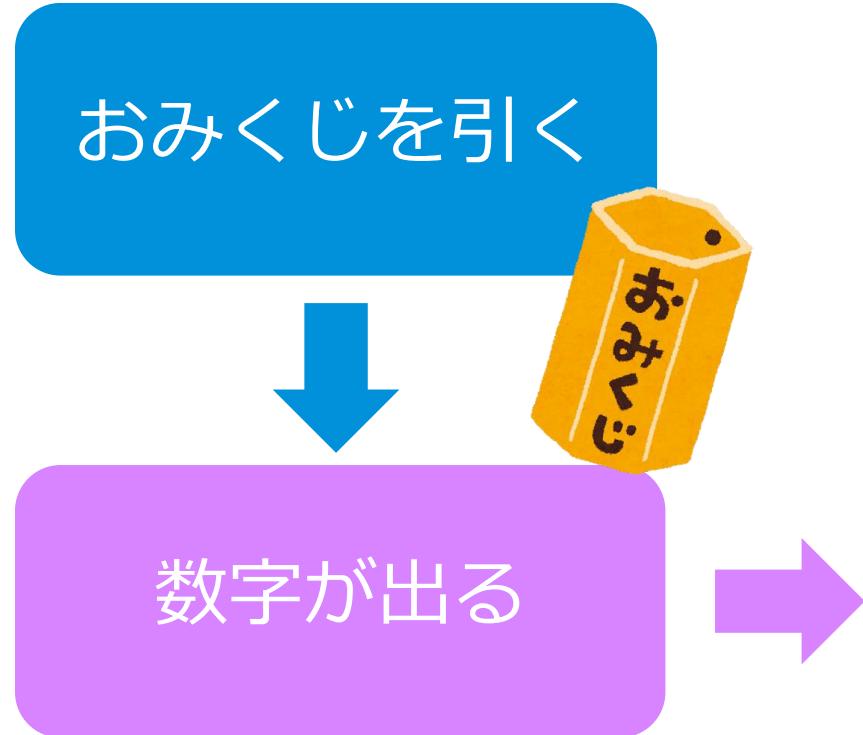
おみくじの仕組み



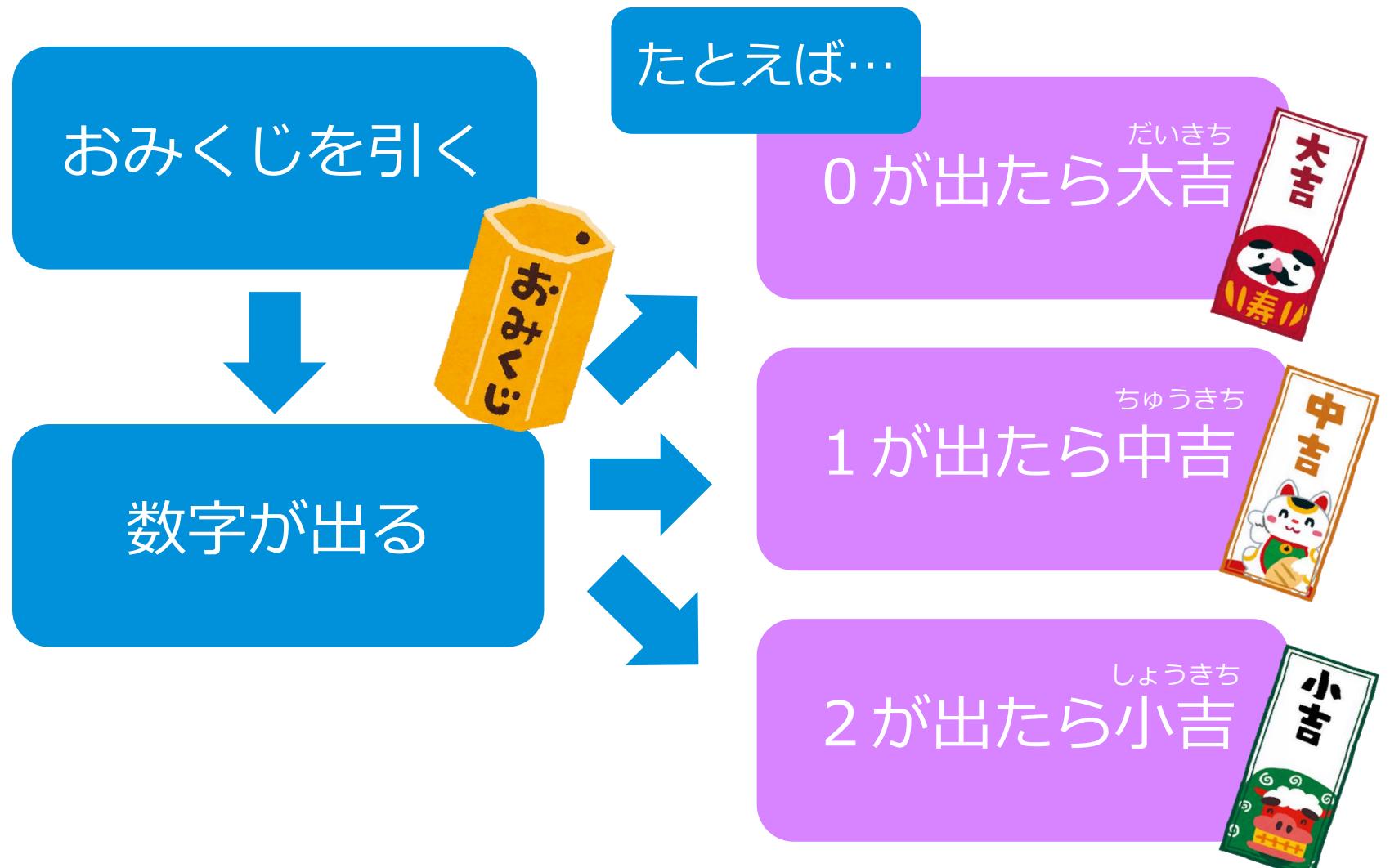
おみくじを引く



おみくじの仕組み



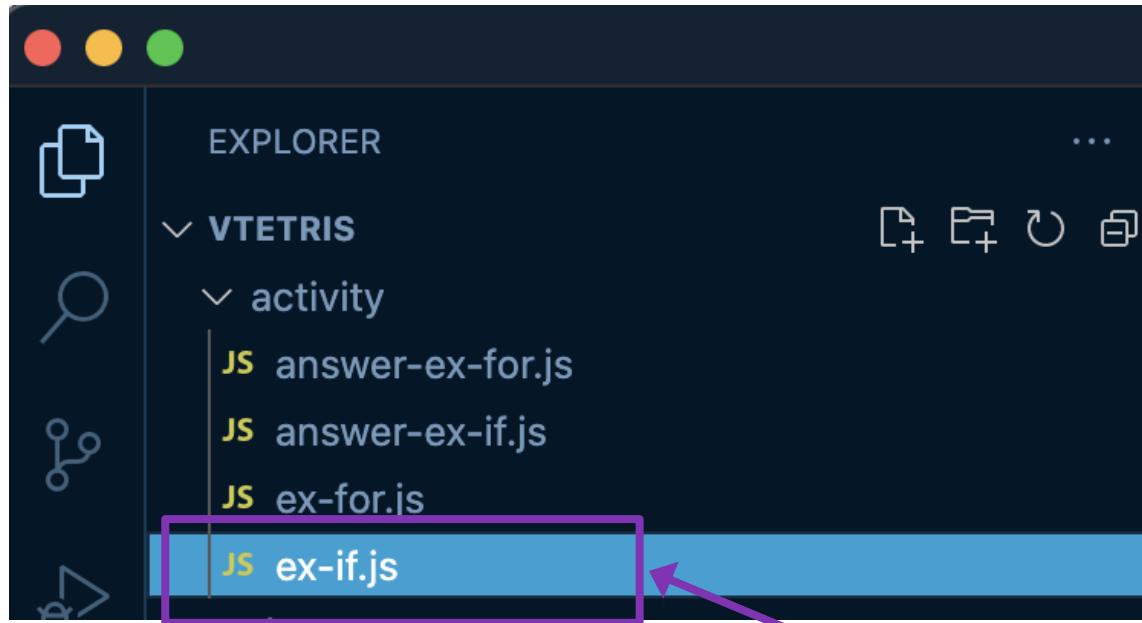
おみくじの仕組み



おみくじを作つてみよう

「ex-if.js」の
ファイルをクリックして
開こう！

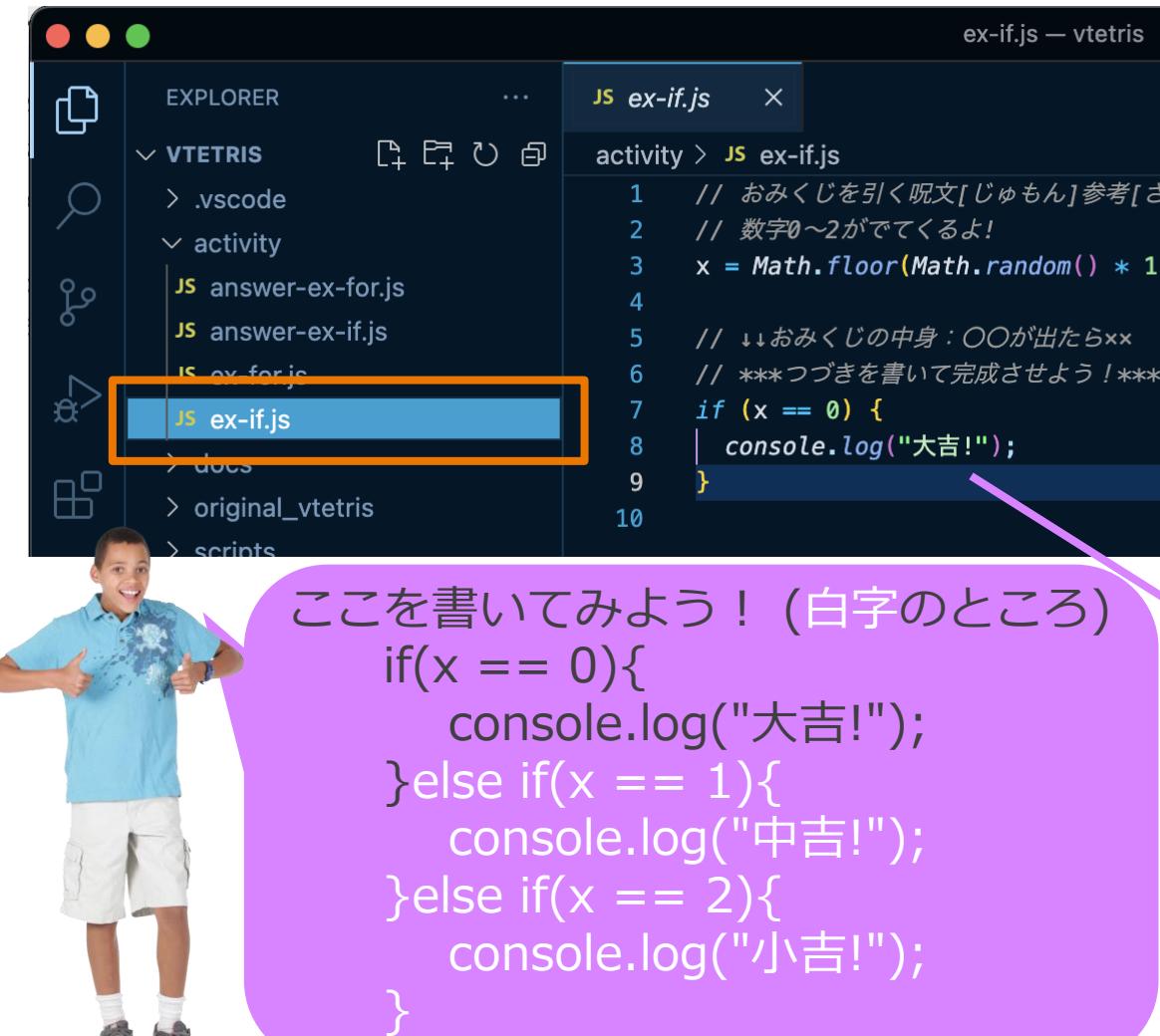
「activity」フォルダの中を見てみよう！



クリックしよう！



おみくじを作つてみよう



EXPLORER

activity > JS ex-if.js

```
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 // 数字0~2がでてくるよ!
3 x = Math.floor(Math.random() * 1);
4
5 // ↓おみくじの中身: ○○が出たらxx
6 // ***つづきを書いて完成させよう! ***
7 if (x == 0) {
8 | console.log("大吉!");
9 }
```

JS ex-if.js

```
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 |
```

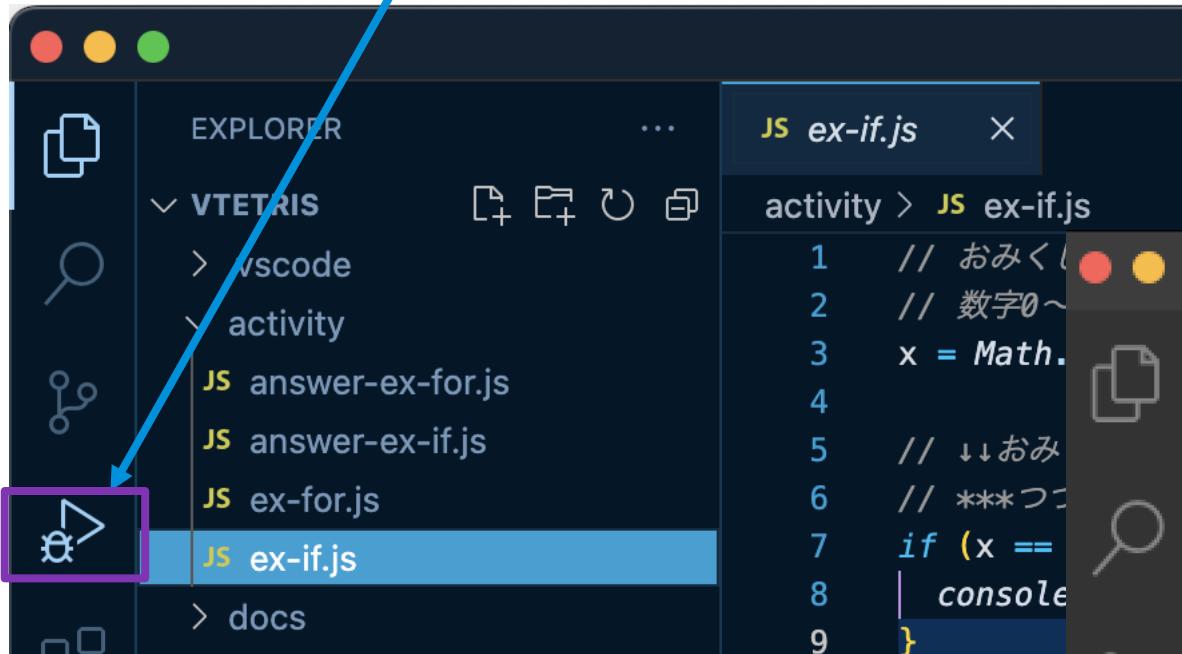
※せつめい書きなどは
//
使えるよ!

ここを書いてみよう！（白字のところ）

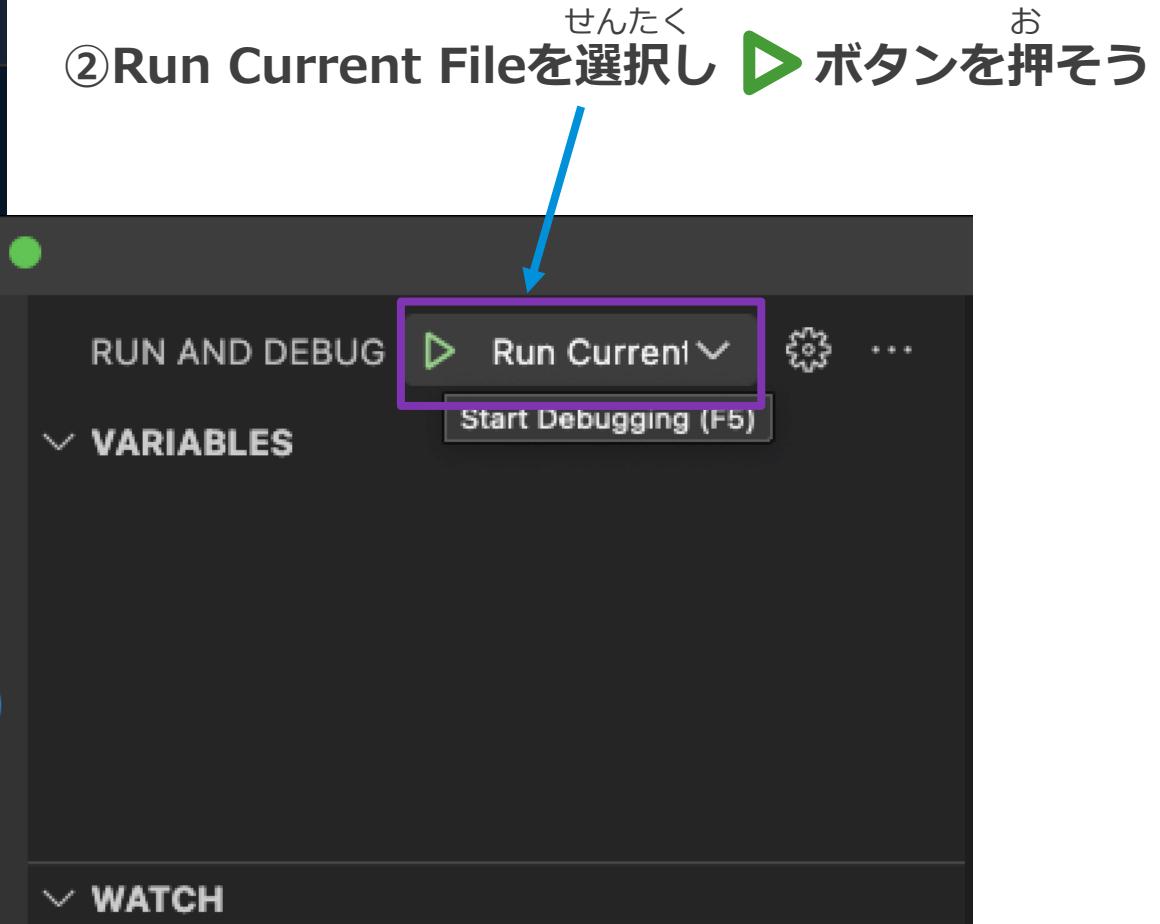
```
if(x == 0){
    console.log("大吉!");
}else if(x == 1){
    console.log("中吉!");
}else if(x == 2){
    console.log("小吉!");
}
```

おみくじを引いてみよう

①Run and Debug ボタンを押そう



②Run Current Fileを選択し ▶ ボタンを押そう



おみくじを引いてみよう

しうきち
“小吉”が出た！
みんなはどんな運勢だった？

```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL
/opt/homebrew/bin/node ./activity/answer-ex-if.js
小吉!
```



if : ○○○だったら△△△する

ふり返ってみよう！

```
if ( 0が出たら ){
    だいきち      ひょうじ
    “大吉！”と表示する;

}else if ( 1が出たら ){
    ちゅうきち      ひょうじ
    “中吉！”と表示する;

}else if ( 2が出たら ){
    しょうきち      ひょうじ
    “小吉！”と表示する;

}
```

“if”と“else if”
ってなに？

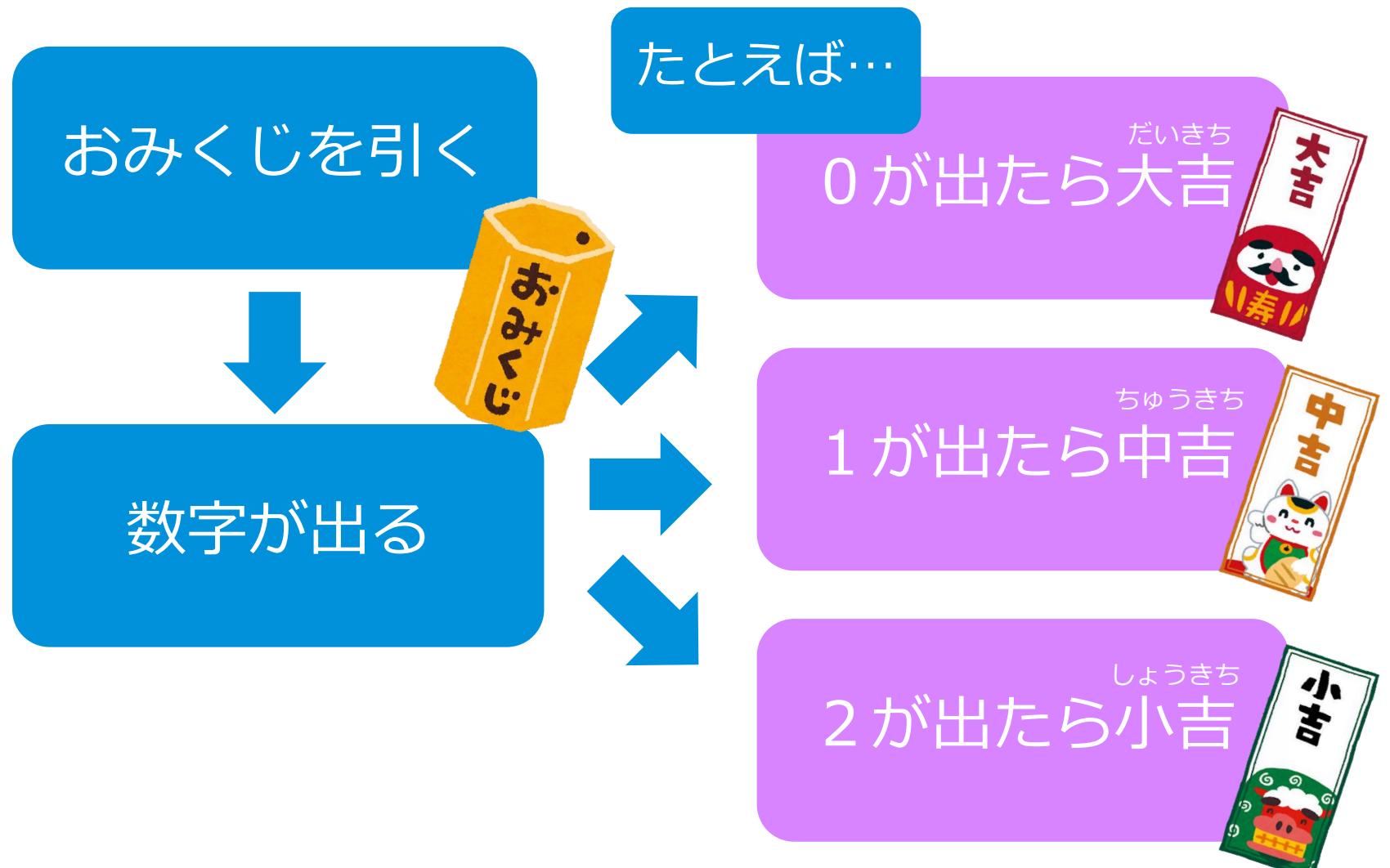


おみくじをたくさん引こう



友だちの分もおみくじを
引いてあげよう！

おみくじの仕組み





おみくじの仕組み

おみくじを引く

たとえば…

0が出たら大吉

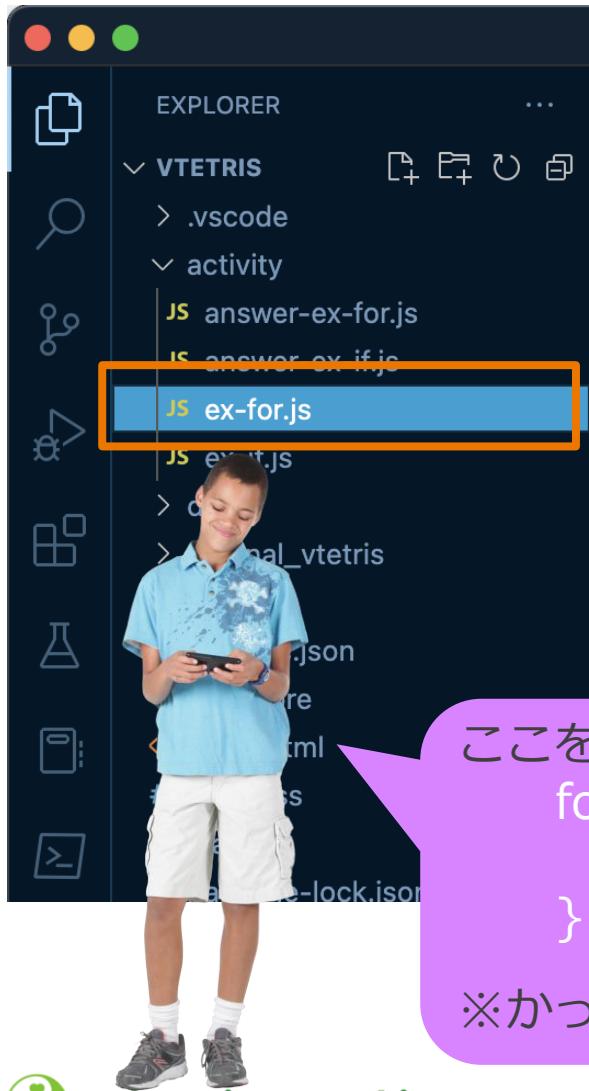
これをみんなの分だけくり返そう
数字が出る

1が出たら中吉

2が出たら小吉



おみくじを作つてみよう



The image shows a screenshot of the Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar with a file tree. A file named "JS ex-for.js" is selected and highlighted with an orange rectangle. In the center is the code editor with a file titled "ex-for.js". The code is written in JavaScript and generates lottery results for three friends. A pink arrow points from a callout bubble at the bottom left to the "for" loop in the code editor. Another pink arrow points from the same callout bubble to the closing brace of the loop in the code editor.

```
// 友だちリスト
let friends = ["Aさん", "Bさん", "Cさん"];
// ***ここにforを使って書いてみよう***
// おみくじを引く呪文(参考(さんこう)):乱数生成
// 数字0~2がでてくるよ!
x = Math.floor(Math.random() * 3);
// ↓さっき完成させたifの部分
if (x == 0) {
  console.log(friend, ":", "大吉!");
} else if (x == 1) {
  console.log(friend, ":", "中吉!");
} else if (x == 2) {
  console.log(friend, ":", "小吉!");
}
```

JS ex-for.js — vtetris

activity > JS ex-for.js > ...

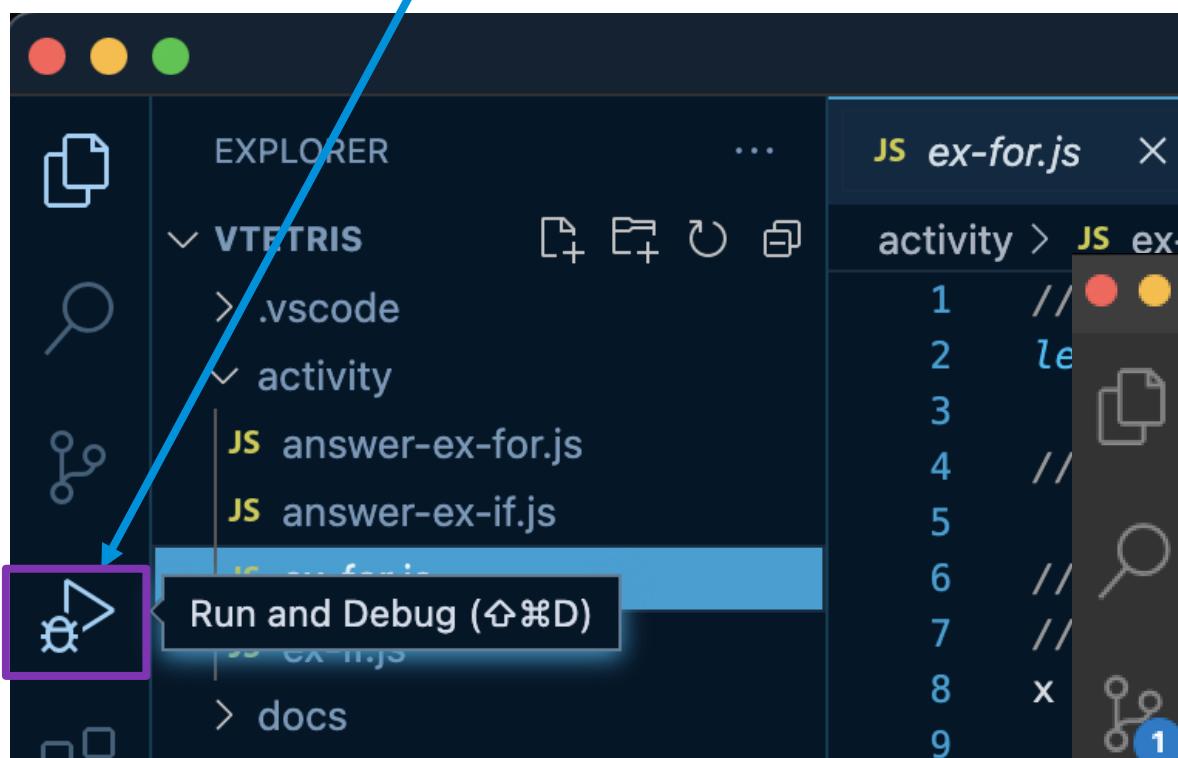
```
let friends = ["Aさん", "Bさん", "Cさん"];
// forで繰り返してみよう
for (let friend of friends) {
  // おみくじを引く呪文(参考(さんこう)):乱数生成(らんすうせいせい))
  x = Math.floor(Math.random() * 3);
  // ↓おみくじの中身:〇〇が出たら××
  if (x == 0) {
    console.log(friend, ":", "大吉!");
  } else if (x == 1) {
    console.log(friend, ":", "中吉!");
  } else if (x == 2) {
    console.log(friend, ":", "小吉!");
  }
}
```

ここを書いてみよう！
for(let friend of friends){
 . . .
}

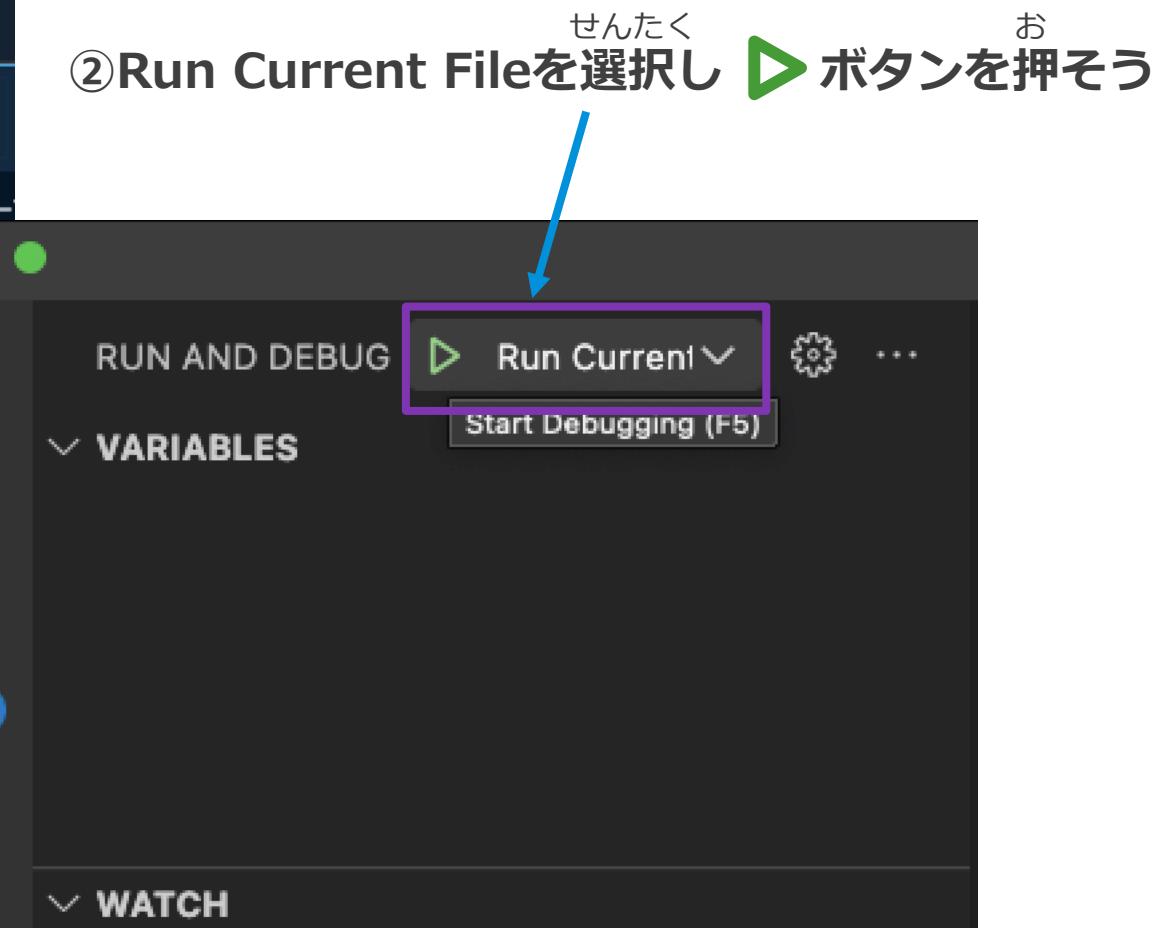
いち
※かっこことじの位置に気をつけよう

おみくじを引いてみよう

①Run and Debug ボタンを押そう
お



②Run Current Fileを選択し ▶ ボタンを押そう
せんたく お



おみくじを引いてみよう

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL

```
/opt/homebrew/bin/node ./activity/answer-ex-for.js
```

Aさん : 中吉!
Bさん : 中吉!
Cさん : 小吉!

みんなの分ができた！



for : ○○○をくり返す

くわしくみてみよう！

```
1 let friends = ["Aさん", "Bさん", "Cさん"];
2
3 // forで繰り返してみよう
4 for (let friend of friends) {
5     // おみくじを引く呪文(参考(さんこう):乱数生成(らんすうせいせい))
6     x = Math.floor(Math.random() * 3);
7
8     // ↓おみくじの中身: ○○が出たらxx
9     if (x == 0) {
10         console.log(friend, ":", "大吉!");
11     } else if (x == 1) {
12         console.log(friend, ":", "中吉!");
13     } else if (x == 2) {
14         console.log(friend, ":", "小吉!");
15     }
16 }
17 |
```

"friends"

で友だちのリスト
をじゅんびしている

"friend"

で友だちリストの
じゅんばんに一人ず
つえらんで全員分く
り返している



for : ○○○をくり返す

ふり返ってみよう！

```
for (友だちの全員の分){  
    おみくじを引く;  
}
```



1回書いたら
くり返してくれ
るんだ！

練習おわり！



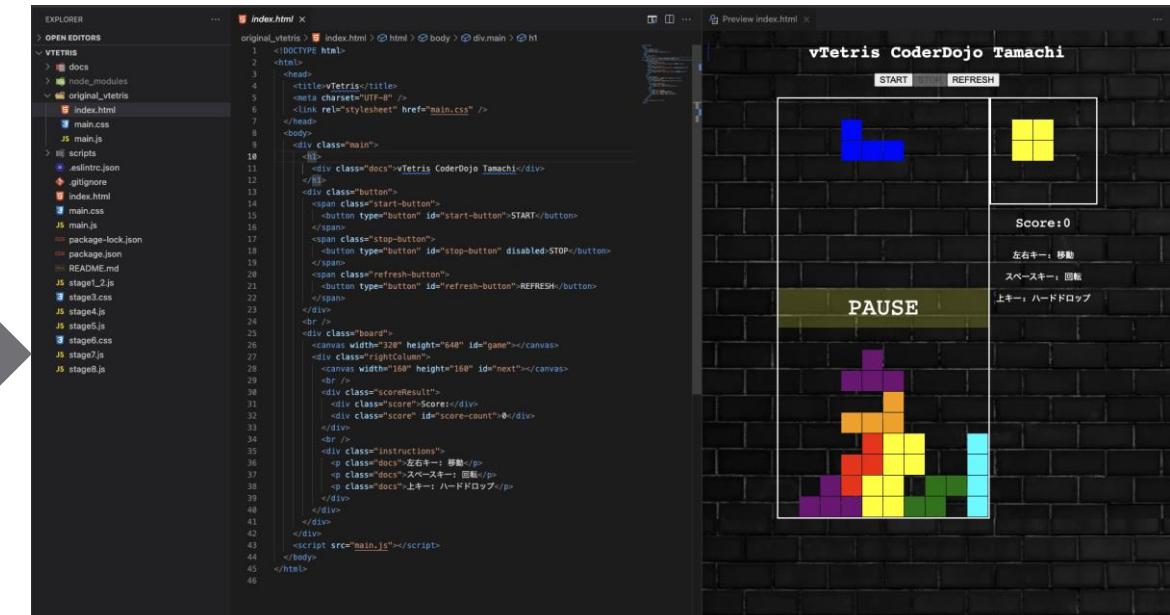
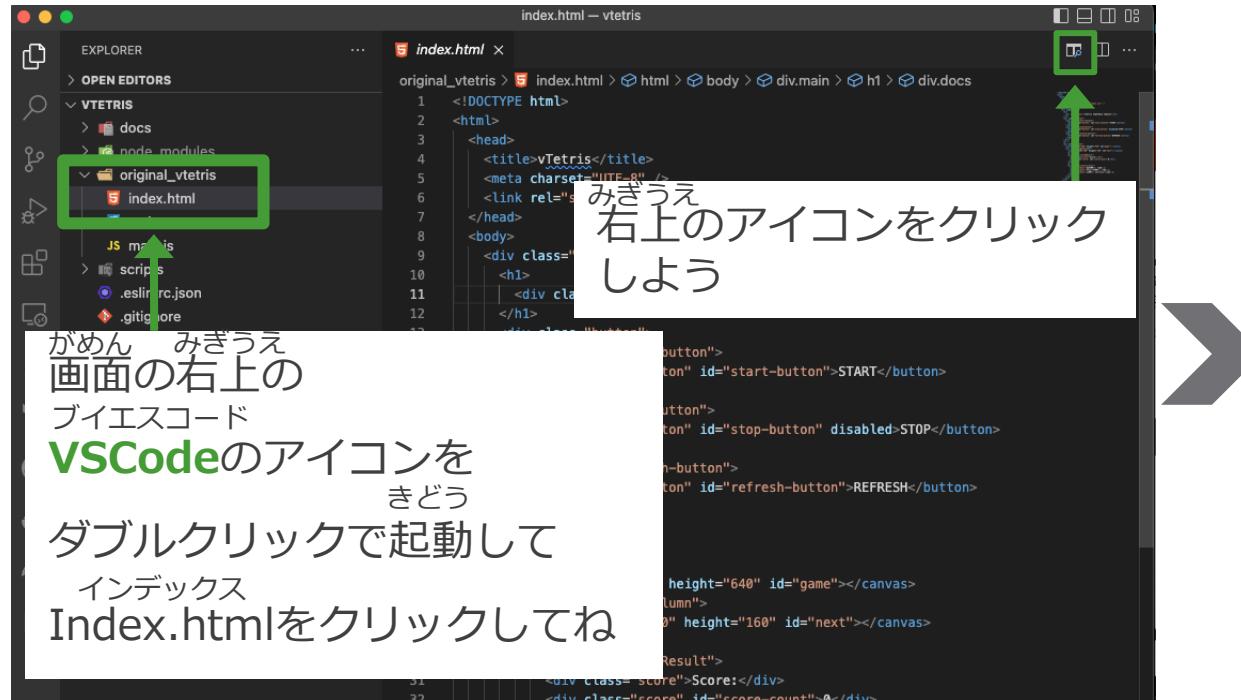


まずはテトリスで遊んでみよう！

テトリス風パズルゲーム - vテトリス

まずはvテトリスで遊んでみよう！

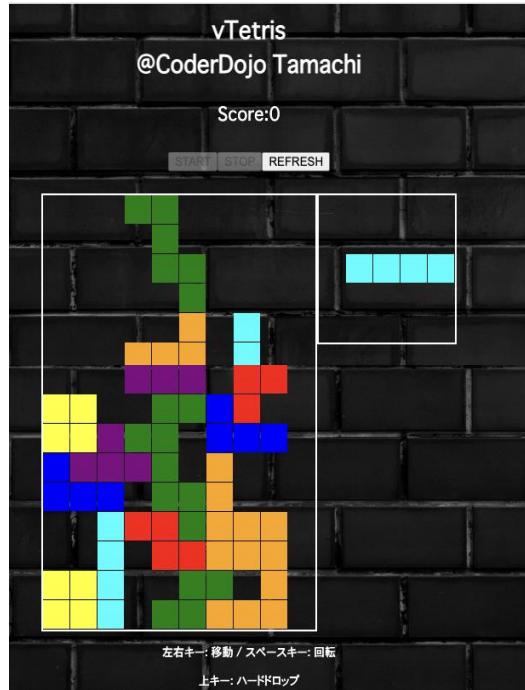
メンターのおにいさん、おねえさんに遊び方を教えてもらってね。





テトリス・プログラミング

かんせい
テトリス風パズルゲームを完成させよう！





じゅんび

ゲームを表示してみよう

ひょうじ



ようこそ ゲームの要素

vTetrisは主にJavaScript / HTML / CSS で作られてるよ
index.htmlにwebブラウザからアクセスすることで遊べるよ

The screenshot shows a code editor with the following details:

- EXPLORER:** Shows the project structure:
 - OPEN EDITORS: index.html
 - VTETRIS: docs, node_modules, original_vtetris, .eslintrc.json, .gitignore, index.html, main.css, main.js, package-lock.json, package.json, README.md, stage1_2.js, stage3.css, stage4.js, stage5.js, stage6.css, stage7.js, stage8.js
- index.html:** The code editor displays the HTML file content. It includes the following code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>vTetris</title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="main.css" />
    <link rel="stylesheet" href="stage3.css" />
    <link rel="stylesheet" href="stage6.css" href="stage6.css" />
  </head>
  <body>
    <script src="stage1_2.js"></script>
    <div class="board">
      <canvas width="320" height="640" id="game"></canvas>
    </div>
    <div class="scoreResult">
      <div class="score">Score:</div>
      <div class="score" id="score-count">0</div>
    </div>
    <script src="stage4.js"></script>
    <script src="stage5.js"></script>
    <script src="stage7.js"></script>
    <button type="button" id="bgm-play">BGM</button>
    <script src="stage8.js"></script>
    <script src="main.js"></script>
  </body>
</html>
```
- Content:** The code editor highlights several files with red boxes:
 - index.html
 - main.css
 - main.js
 - stage1_2.js
 - stage3.css
 - stage4.js
 - stage5.js
 - stage6.css
 - stage7.js
 - stage8.js

Annotations on the left side of the code editor:

- HTML points to the index.html file.
- CSS points to the main.css and stage3.css files.
- JavaScript points to the main.js and stage1_2.js files.
- チャレンジするステージ (Challenge Stage) points to the stage1_2.js through stage8.js files.

The screenshot shows the vTetris game interface with the following details:

- Title:** vTetris @CoderDojo Tamachi
- Score:** 0
- Controls:** START, STOP, REFRESH
- Game Area:** A 10x20 grid where Tetrominoes (I, O, L, S, Z, T) are falling.
- Instructions:** 左右キー: 移動 / スペースキー: 回転 (Left/Right keys: Move / Space key: Rotate)
- Drop Key:** 上キー: ハードドロップ (Up key: Hard Drop)

ひょうじ ほうほう ゲームの表示方法

ゲームを動かすには、
みんなが書いたプログラミングコードを

- ✓ コンピュータに置く
 - ✓ コンピュータでじっこうする

ひつよう 必要があるよ



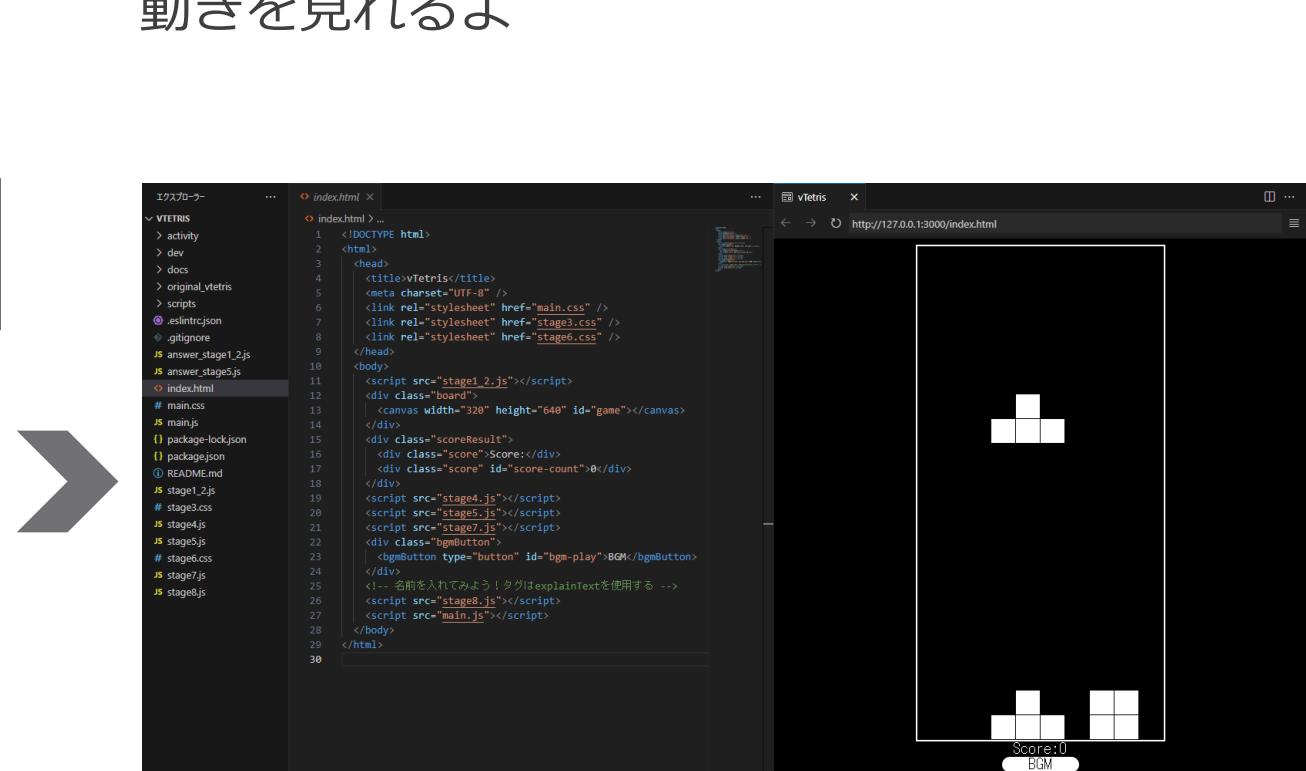
ひょうじ ほうほう ゲームの表示方法

index.html を開いて VSCode画面右上のアイコンをクリックしよう



```
エクスプローラー ... index.html x
V TETRIS
> activity
> dev
> docs
> original_vtetris
> scripts
.eslintrc.json
.gitignore
JS answer_stage1_2.js
JS answer_stage5.js
JS index.html
# main.css
JS main.js
{ package-lock.json
{ package.json
 README.md
JS stage1_2.js
# stage3.css
JS stage4.js
JS stage5.js
# stage6.css
JS stage7.js
JS stage8.js
4 <title>vTetris</title>
5 <meta charset="UTF-8" />
6 <link rel="stylesheet" href="main.css" />
7 <link rel="stylesheet" href="stage3.css" />
8 <link rel="stylesheet" href="stage6.css" />
9 </head>
<body>
10 <script src="stage1_2.js"></script>
11 <div class="board">
12   <canvas width="320" height="640" id="game"></canvas>
13 </div>
14 <div class="scoreResult">
15   <div class="score">Score:</div>
16   <div class="score" id="score-count">0</div>
17 </div>
18 <script src="stage4.js"></script>
19 <script src="stage5.js"></script>
20 <script src="stage7.js"></script>
21 <div class="bgmButton">
22   <bgmButton type="button" id="bgm-play">BGM</bgmButton>
23 </div>
24 <!-- 名前を入れてみよう！タグはexplainTextを使用する -->
25 <script src="stage8.js"></script>
26 <script src="main.js"></script>
27 </body>
28 </html>
```

みぎがわ
右側にゲーム画面が表示されて、実際に動きを見れるよ



きほんのテトリスを見てみよう

さいしょ

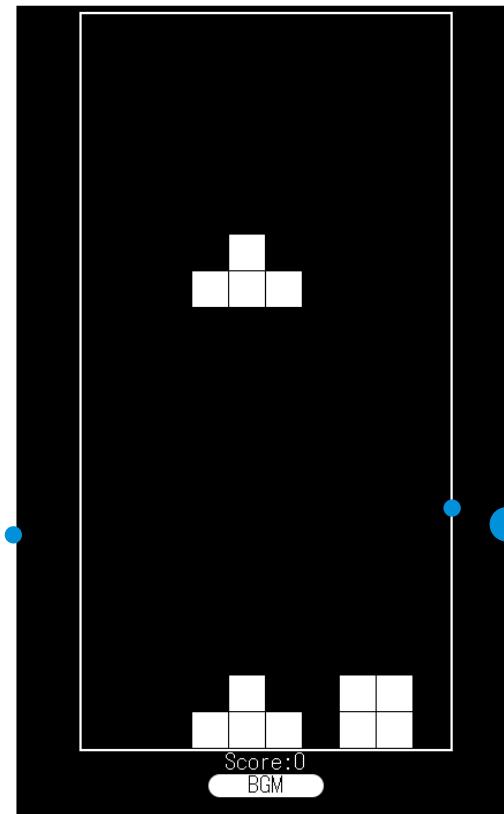
最初のじょうたいは、なんだかさみしいね 😊

色がすくないなあ …

ブロックのしゅるい
がすくないなあ…

ブロックが回転
しないぞ？

いろいろ
アレンジしたいな！





ステージのすすめかた

- プリントにゲームを完成させる **かんせい Stage 1** から **Stage 6** までの問題があるよ
すきな Stage をえらんでチャレンジしてみよう！
- もんだいは別冊の資料みてね
- メンターのおねえさん、おにいさんにどんどんしつもんしよう！
- 自分なりのくふうをしてみよう！



ステージ

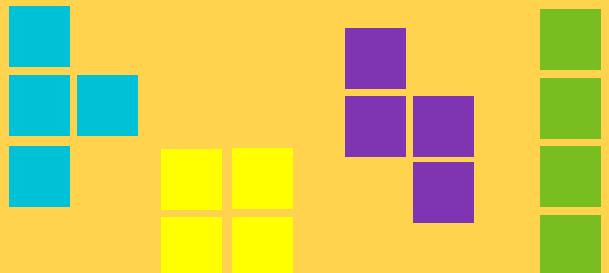
Stage1

～色をかえてみよう～



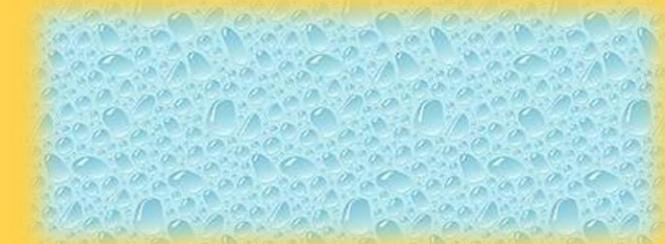
Stage2

～ブロックをふやそう～



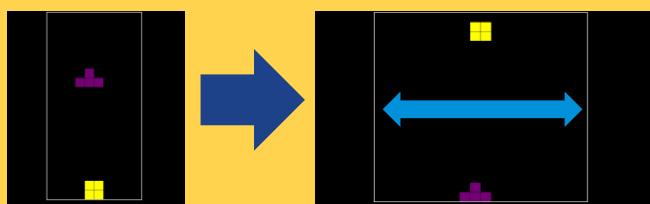
Stage3

～背景をアレンジしよう～



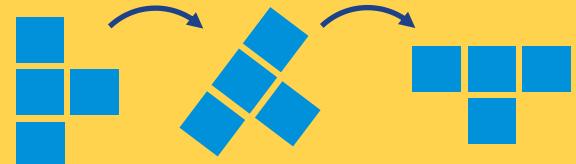
Stage4

～フィールドを広げよう～



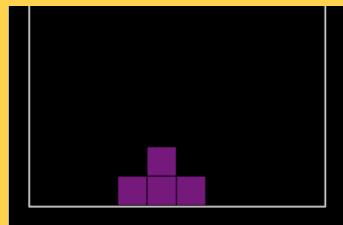
Stage5

～回転させてみよう～



Stage6

～文字を入れてみよう～



Coded by マイケル





Stage 1

ブロックに色をつけよう

stage 1 – 色の変更
へんこう

Stage 1でつかうソースコード

このStageでは『stage1_2.js』をつかうよ！

The screenshot shows a code editor interface with a sidebar containing file icons and a list of files. The list includes: docs, original_vtetris, .eslintrc.json, .gitignore, index.html, main.css, main.js, package-lock.json, package.json, README.md, stage1_2.js (which is highlighted with an orange rectangle), stage3.css, stage4.js, stage5.js, stage6.css, stage7.js, and stage8.js.

The main editor area displays the content of stage1_2.js:

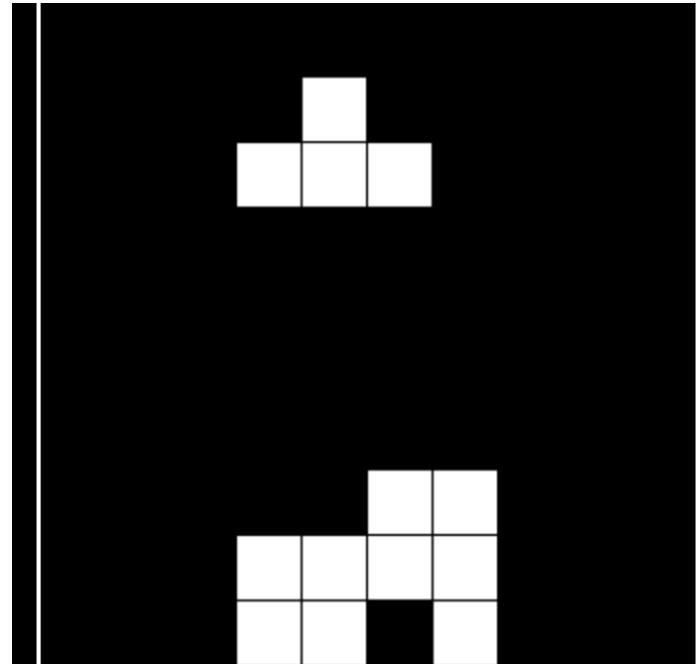
```
// テトロミノの色を設定
const colors = {
  I: "white",
  T: "white",
};

// 参考サイト: https://www.colordic.org/

// テトロミノを二次元配列で設定
// https://tetris.fandom.com/wiki/SRS
const tetrominos = {
  I: [
    [1, 1],
    [1, 1],
  ],
  T: [
    [0, 1, 0],
    [1, 1, 1],
    [0, 0, 0],
  ],
};
// 参考サイト: https://www.javadrive.jp/javascript/array/index5.html
```

ブロックに色をつけよう

はじめは、ブロックは白いよ
どうやって色をつけるのかな？
いっしょにやってみよう！！



色をえらぼう①



さんこう
参考サイトはDesktopの
ショートカットから飛べるよ

まずは下の9つから色をえらぼう 🔎 ✨

もっとこだわりたい人は、参考サイトをのぞいてみてね 🐰 💕 🌸

yellow

pink

blue

purple

powder
blue

orange

green

red

green
yellow

さんこう
*参考サイト

<https://www.colordic.org/>

どの色がいいかな？



どれもいいなあ

色をえらぼう②

色の名前を “ ” のなかに書こう！

lightpink
#ffb6c1

palevioletred
#db7093

これはどのブロックかな？
Stage2のP.80をみてみよう！

JS stage1_2.js > [?] colors > 🔑 O

```
1 // テトロミノの色を設定
2 const colors = {
3   O: "#ffb6c1",
4   T: "palevioletred",
5 }
```

“ ” の中に色を書くよ

💡 NestStep :
色は"#ffb6c1"などのcolor codeでも指定できるよ！
「#～(color code)」は
RGBを0～255で
あらわしているよ！
気になったら「16進数」を
調べてみよう 🧑‍💻 💫

みんなの大好きな色を、たくさん使おう♡

色がいっぱいだと
ジェリービーンズみたいで
たのしいね *



もちろん
色がなくても
たのしいよ♪



ここからは自由に作ってみよう！





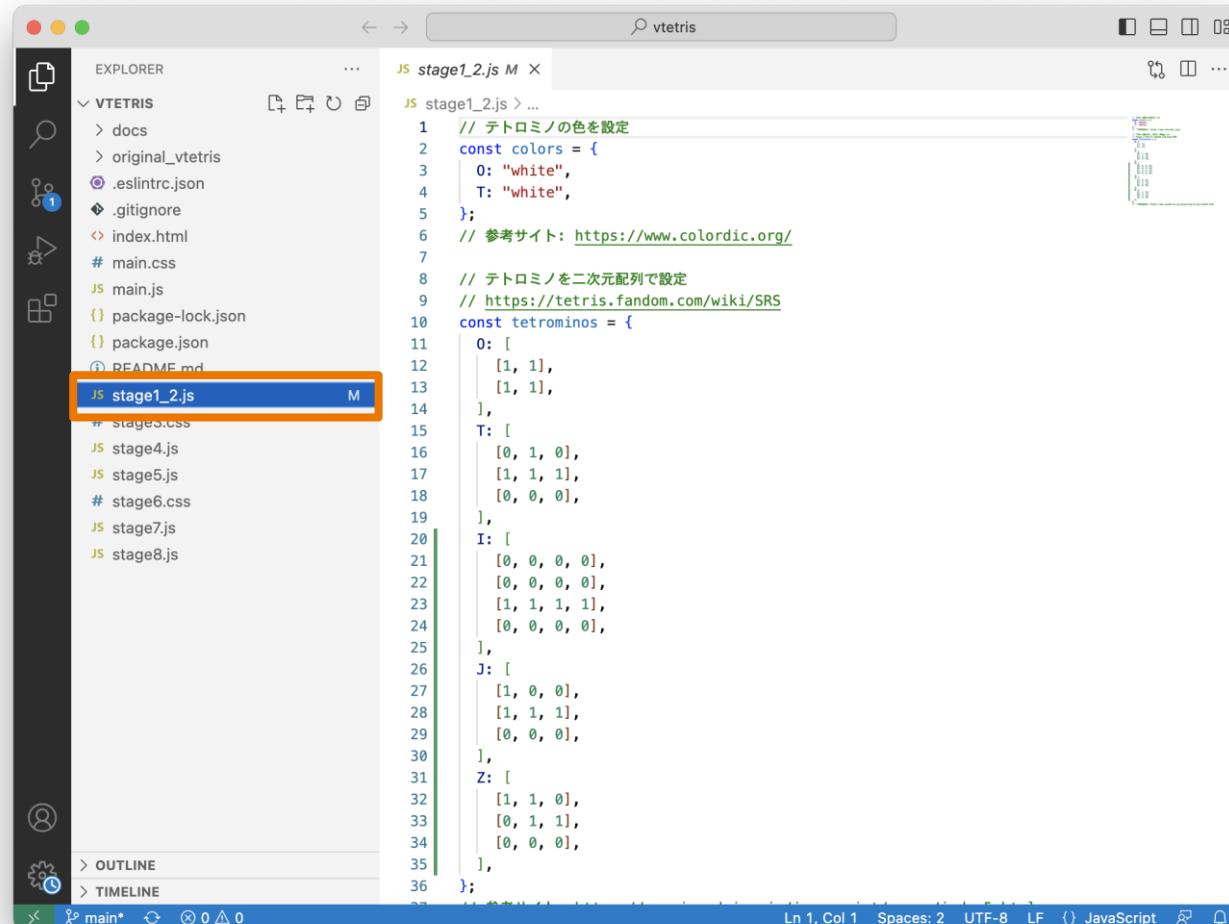
Stage 2

ブロックをふやそう

stage 2 – ブロックの形を作つてみよう ♦

Stage 2でつかうソースコード

このStageでは『Stage1_2.js』をつかうよ！



The screenshot shows a code editor window with the title bar "vtetris". The left sidebar is labeled "EXPLORER" and lists several files: "docs", "original_vtetris", ".eslintrc.json", ".ignore", "index.html", "# main.css", "main.js", "package-lock.json", "package.json", and "README.md". The file "stage1_2.js" is highlighted with a red box in the sidebar. The main editor area shows the following JavaScript code:

```
// テトロミノの色を設定
const colors = {
  O: "white",
  T: "white",
};

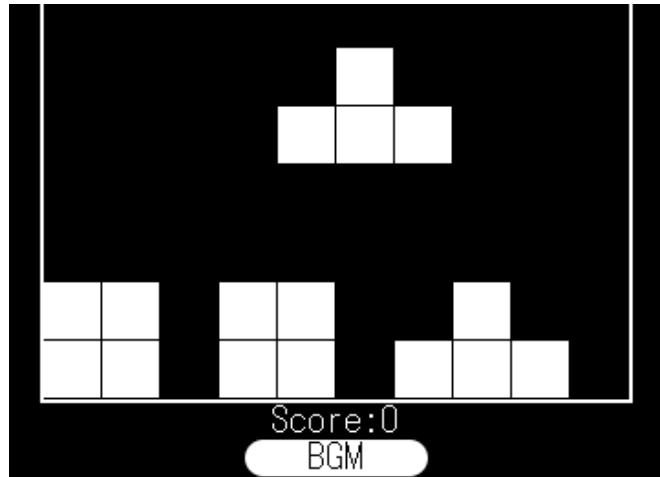
// 参考サイト: https://www.colordic.org/

// テトロミノを二次元配列で設定
// https://tetris.fandom.com/wiki/SRS
const tetrominos = {
  O: [
    [1, 1],
    [1, 1],
  ],
  T: [
    [0, 1, 0],
    [1, 1, 1],
    [0, 0, 0],
  ],
  I: [
    [0, 0, 0, 0],
    [0, 0, 0, 0],
    [1, 1, 1, 1],
    [0, 0, 0, 0],
  ],
  J: [
    [1, 0, 0],
    [1, 1, 1],
    [0, 0, 0],
  ],
  L: [
    [1, 1, 0],
    [0, 1, 1],
    [0, 0, 0],
  ],
  Z: [
    [1, 1, 0],
    [0, 1, 1],
    [0, 0, 0],
  ],
};
```

The status bar at the bottom indicates "Ln 1, Col 1" and "Spaces: 2".

まずはブロックを作ってみよう

さいしょ
最初の じょうたい だとブロックが 2 種類しかないね …



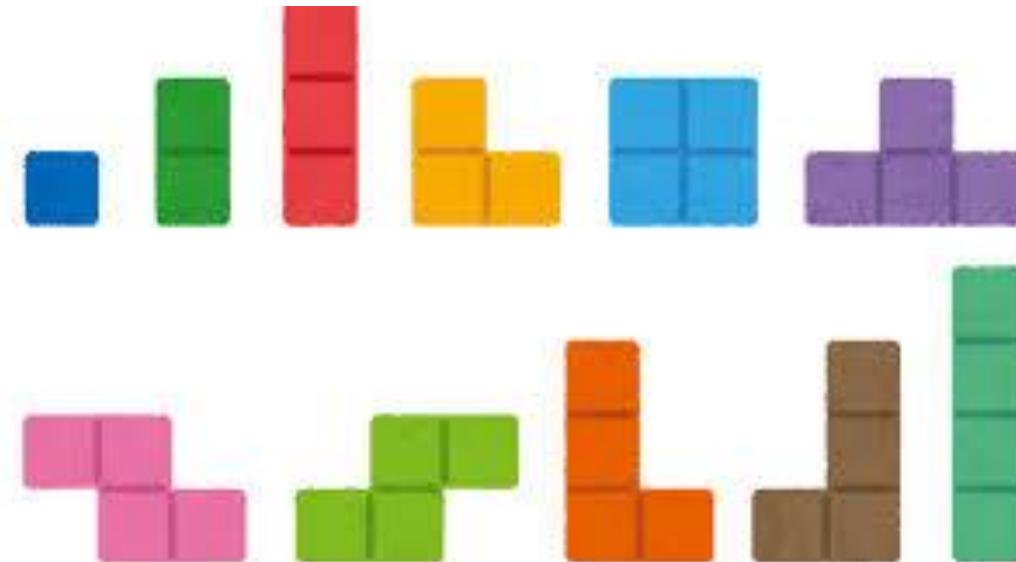
しゅるい
種類がすくないな。。。
他のブロックもふやせないかな 🤔

まずはブロックを作ってみよう

ゲームの主役になるブロックをまずは作ってみよう！

ひとつずつ落ちてくるブロックにはいろんな形があるね👉

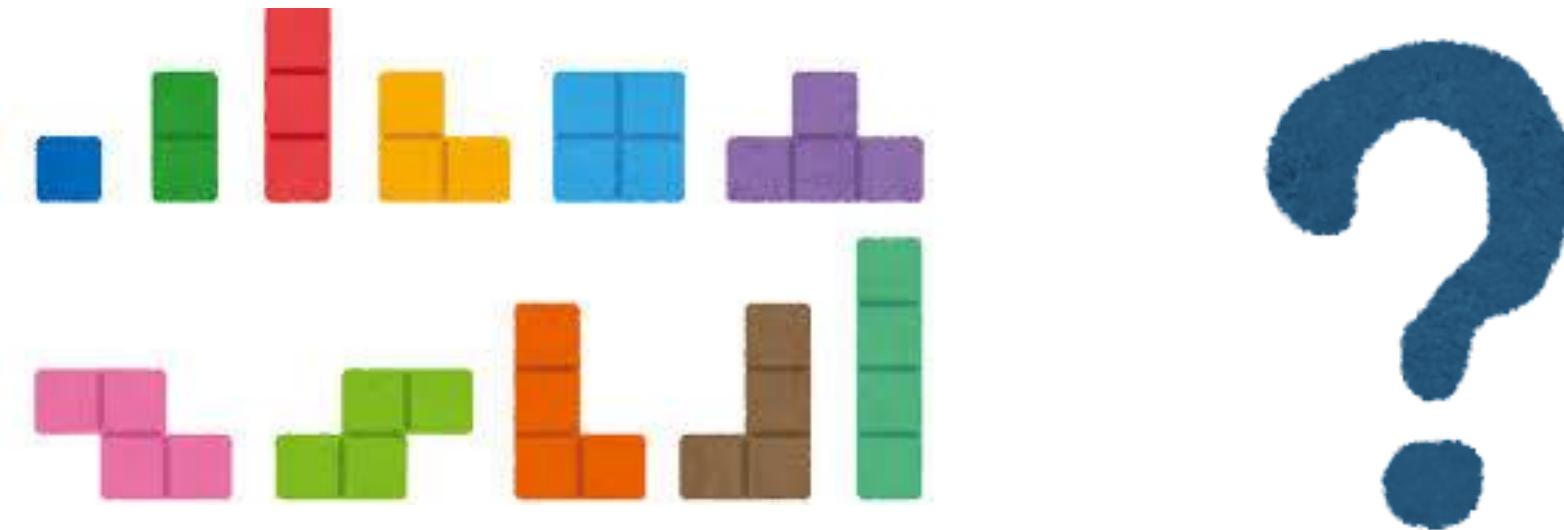
その形を作っていくのがさいしょのステップになるよ👉



かんさつ

ブロックの形を作るために、まずは観察してみよう

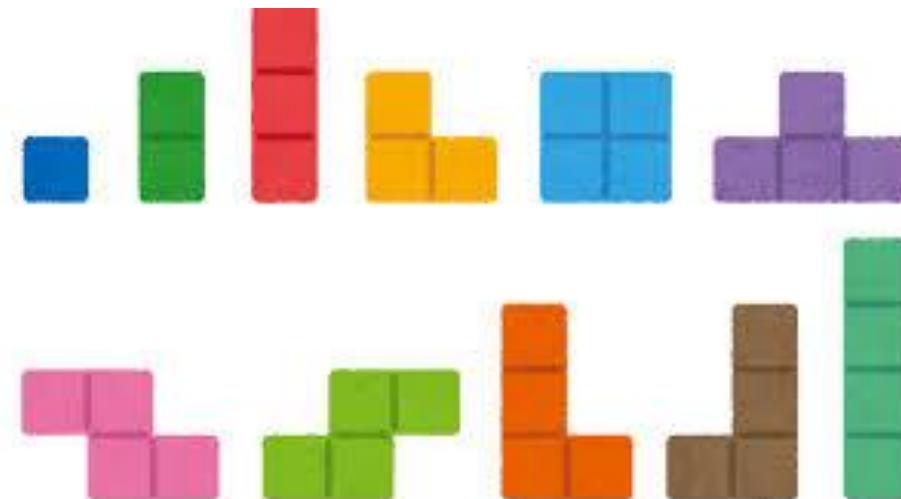
1つのブロックがどんなふうにできているか、どんなかたちなのかまずは見てかんがえてみよう 😊



ポイント1: ブロックはこうできている！

よく見るとどのブロックも小さい正方形(四角形)のパーツが組み合
わさせてできている！

このこと とそれぞれのブロックの形に注目しながらプログラミング
をやっていこう💡



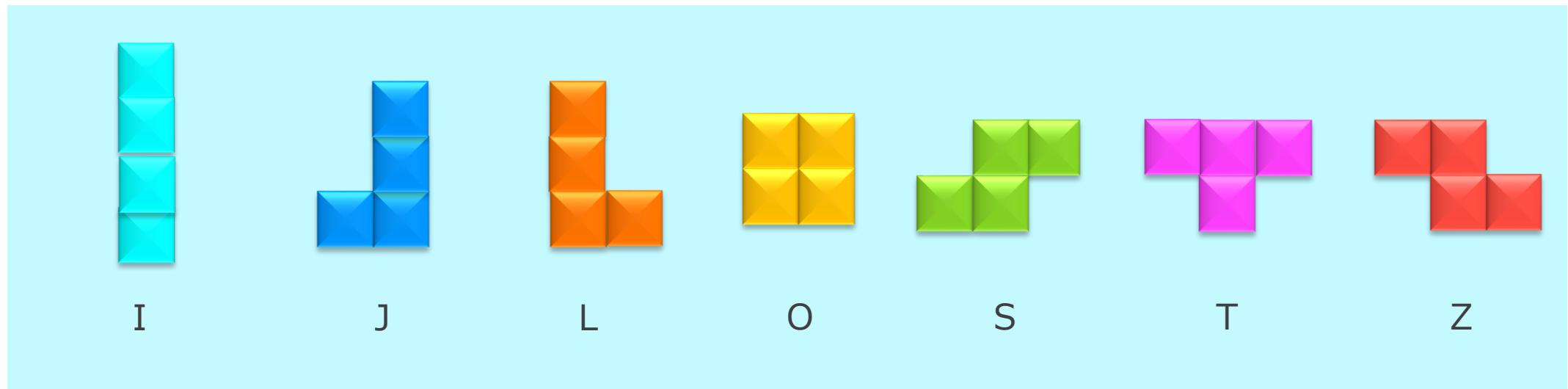
ポイント2：ブロックの形が何かにしている！

いろんな形のブロック、実は。。

よくみるとアルファベットの形にしている！

(ちょっと見づらいのもあるけれど😊)

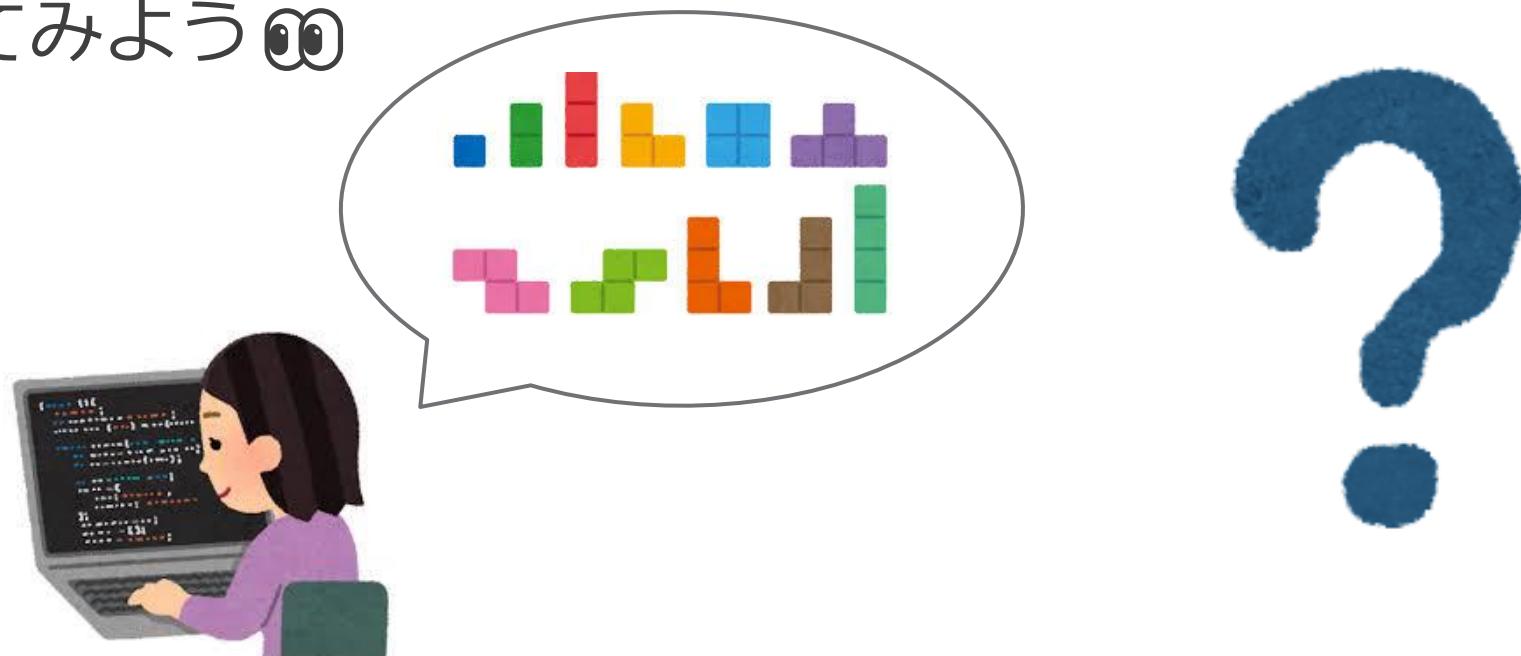
ブロックのかたちはこのようなアルファベットでよびます！



どうやってプログラミングでブロックの形を表す？

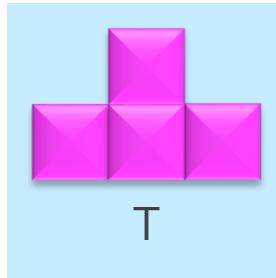
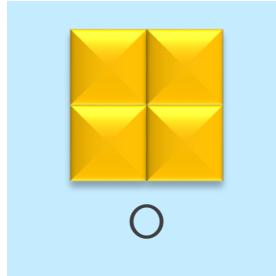
ブロックが正方形のパーツでできていることと、アルファベットみたいなかたちなことがわかったね！

そこで、プログラミングではどんなふうにブロックの形をあらわすのか見てみよう@@



まずはOとTのブロックの表し方を見て考えてみよう

```
O: [  
  [1, 1],  
  [1, 1],  
],  
  
T: [  
  [0, 1, 0],  
  [1, 1, 1],  
  [0, 0, 0],  
],
```

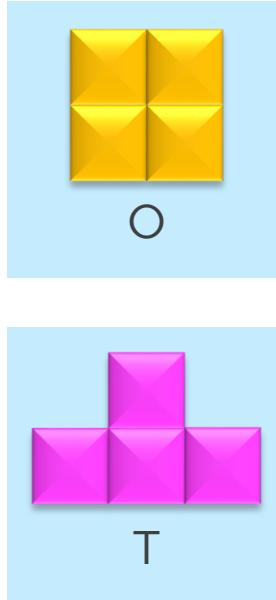


OとTはこんな風にコードを書くよ 
かんさつ
コードとブロックをあわせて観察してみよ
う。 

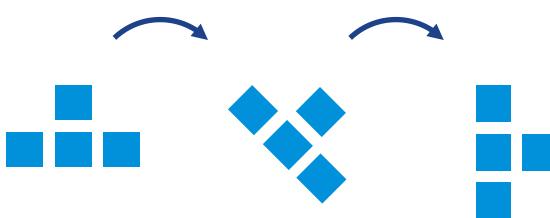
- ✓ 使われている数字はどんなのだろう？
- ✓ その数字はどんなふうにならんでいる？
- ✓ もしかしてなにか決まったルールがありそう？

コードはこう書く！

```
0: [
  [1, 1],
  [1, 1],
],
T: [
  [0, 1, 0],
  [1, 1, 1],
  [0, 0, 0],
],
```



- ✓ 使う数字は0と1
- ✓ ブロックの形をなぞるように1が書かれている！空きマスは0
- ✓ タテヨコの列が同じマス数
 - ✓ いちばん長いところに合わせよう
 - ✓ 回転させる時にきいてくるポイントだよ！



回転はStage 5でチャレンジしてみよう！

問題をやってみよう！

Stage 2 の問題を見てみてね！



Stage 3

はいけい

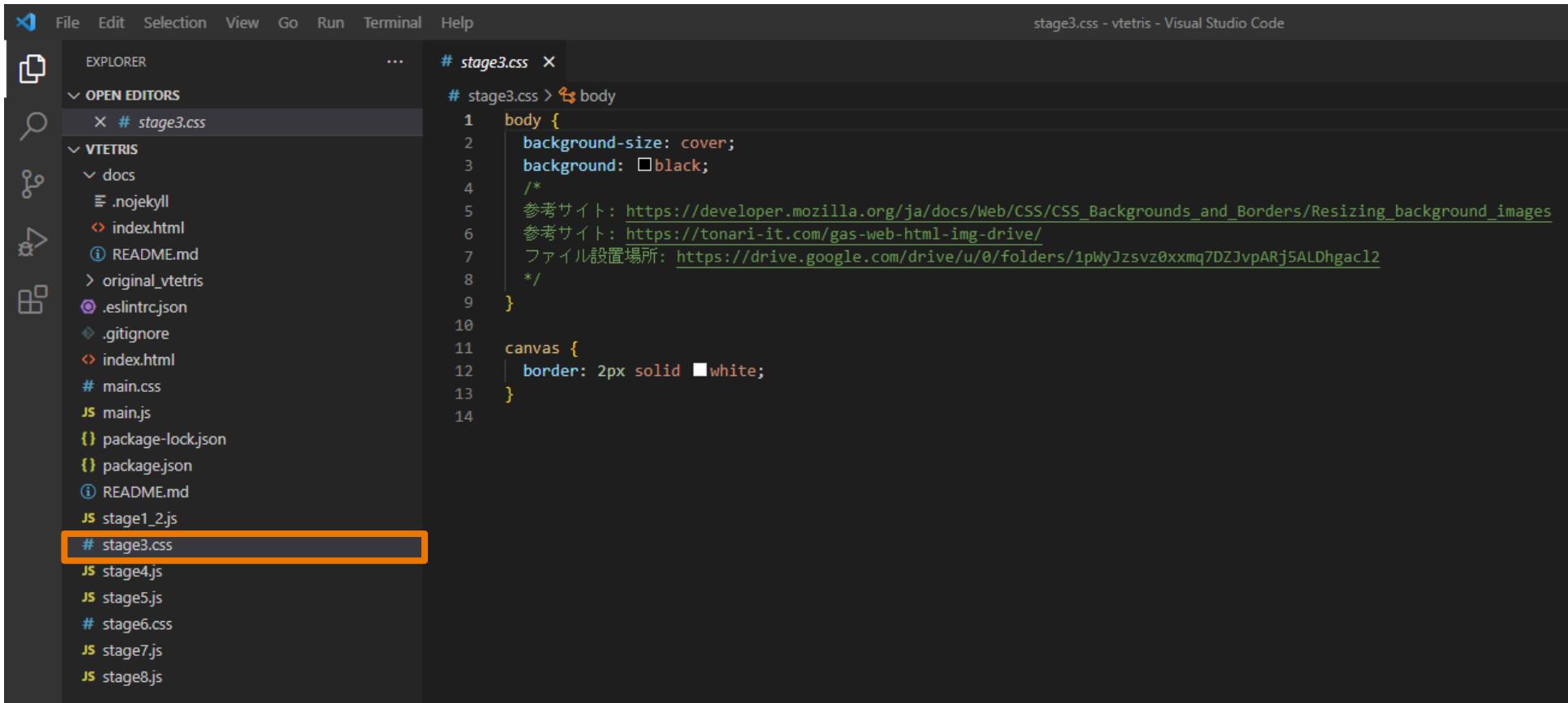
背景をアレンジしてみよう

はいけい

stage 3 – 背景の色をかえてみよう

Stage 3でつかうソースコード

このStageでは『stage3.css』をつかうよ！



The screenshot shows the Visual Studio Code interface. The left sidebar displays a file tree with the following structure:

- File
- Edit
- Selection
- View
- Go
- Run
- Terminal
- Help

EXPLORER

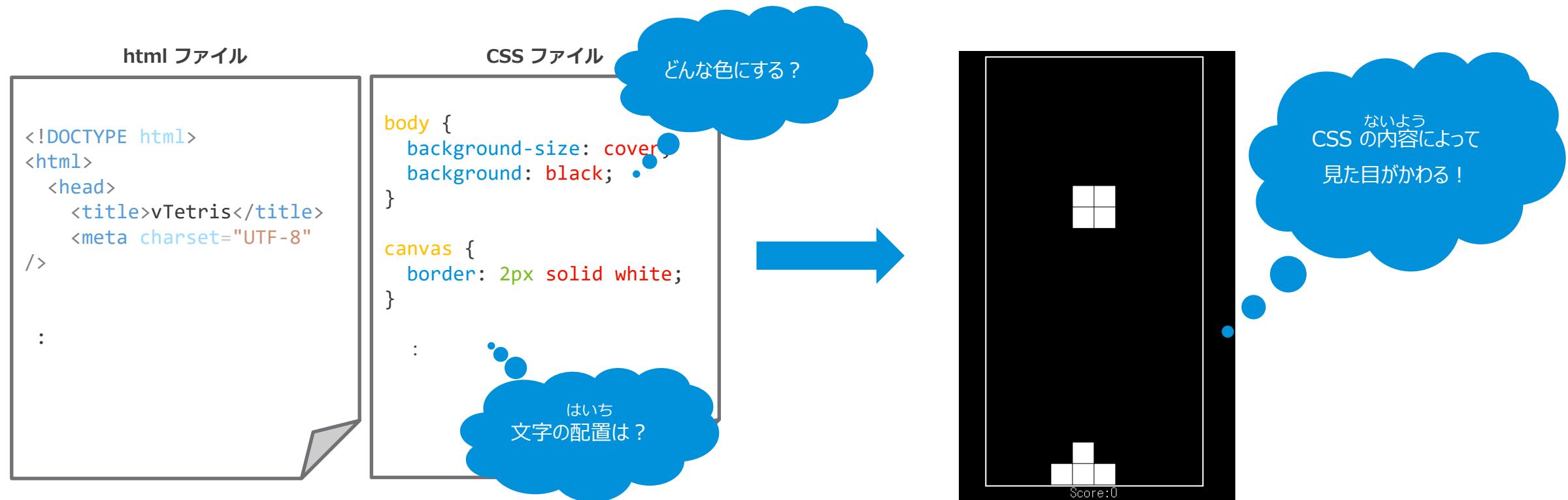
- OPEN EDITORS
 - # stage3.css
- VTETRIS
 - docs
 - .nojekyll
 - index.html
 - README.md
 - original_vtetris
 - .eslintrc.json
 - .gitignore
 - index.html
 - # main.css
 - main.js
 - package-lock.json
 - package.json
 - README.md
 - stage1_2.js
 - # stage3.css
 - stage4.js
 - stage5.js
 - # stage6.css
 - stage7.js
 - stage8.js

```
# stage3.css - vtetris - Visual Studio Code
# stage3.css > body
1 body {
2     background-size: cover;
3     background: black;
4     /*
5     参考サイト: https://developer.mozilla.org/ja/docs/Web/CSS/CSS\_Backgrounds\_and\_Borders/Resizing\_background\_images
6     参考サイト: https://tonari-it.com/gas-web-html-img-drive/
7     ファイル設置場所: https://drive.google.com/drive/u/0/folders/1pWyJzsvz0xxmq7DZJvpARj5ALDhgac12
8 */
9 }
10 canvas {
11     border: 2px solid white;
12 }
13
14
```

はいけい

背景の色をかえてみよう！

はいけい
背景の色や線の太さなど、「見え方」にかんするものは
CSS というファイルに書くこともできるよ！



はいけい

背景の色をかえてみよう！

はいけい
stage3.cssを見てみよう！どこで背景の色を決めていそうか？

えいご　はいけい　バックグラウンド

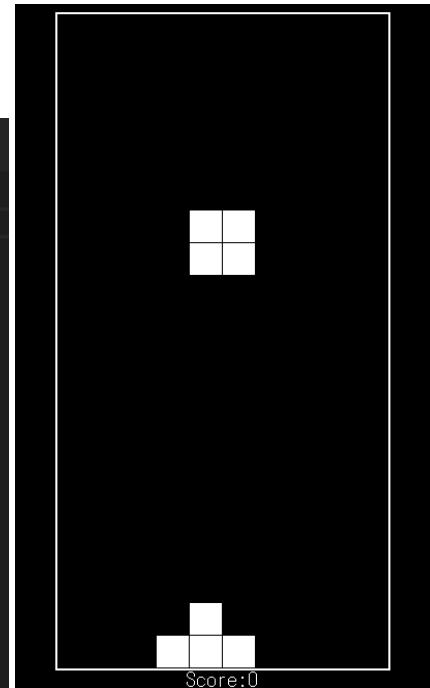
英語で背景は“Background”、黒は“black”

いまはこんな感じ

```
# stage3.css X
# stage3.css > body
1  body {
2    background-size: cover;
3    background: black;
4    /*
5      参考サイト: https://developer.mozilla.org/ja/docs/Web/CSS/CSS\_Backgrounds\_and\_Borders/Resizing\_background\_images
6      参考サイト: https://tonari-it.com/gas-web-html-img-drive/
7      ファイル設置場所: https://drive.google.com/drive/u/0/folders/1pWyzsvz0xxmq7DZJvpARj5ALDhgac12
8    */
9  }
10
11 canvas {
12   border: 2px solid white;
13 }
14
```

ここかな？

それともこっち？



じっさい

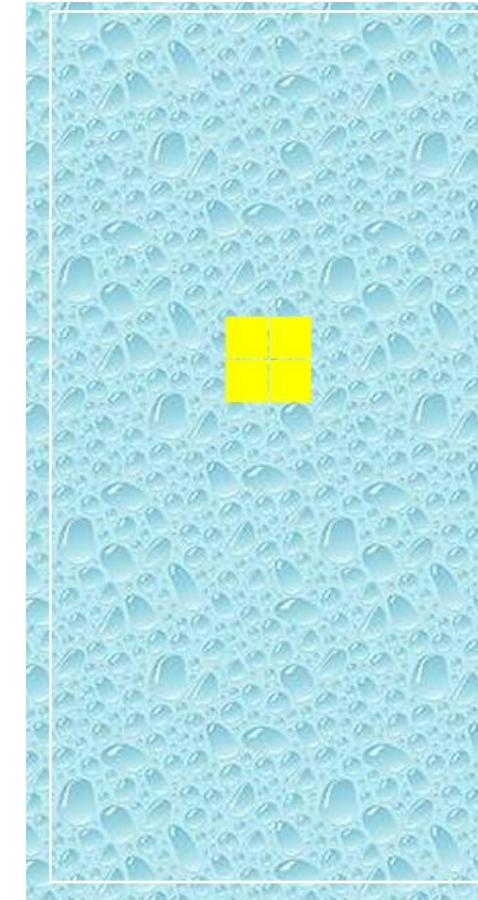
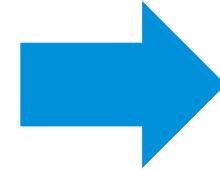
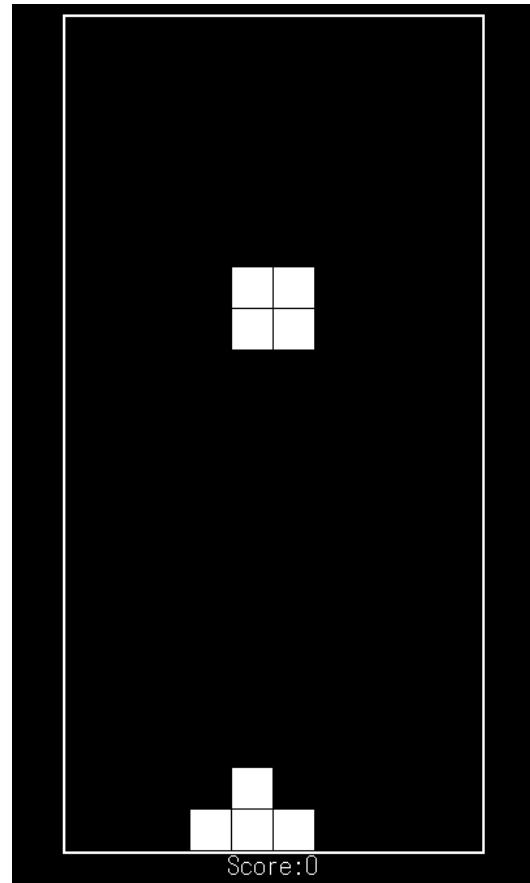
実際にやってみよう！

Stage 3 の問題を見てみてね！

はいけい がぞう

背景を画像にしてみよう！

たとえばこんなかんじ



はいけい

がぞう

背景を画像にしてみよう！

stage3.cssをみてみよう

1. なにを用意したらよさそうか？
2. どう書いたら画像を背景にできそうか？

```
body {  
    background-size: cover;  
    background: black;  
}  
  
canvas {  
    border: 2px solid white;  
}
```

はいけい がぞう
背景を画像にしてみよう！

stage3.cssをみてみよう

ようい
1. なにを用意したらよさそうか？

はいけい がぞう
→ **背景にしたい画像が必要！**

ようい がぞう
まずは用意してある画像（サンプル： water.jpg ）を背景にしてみよう



はいけい
水の背景

今日はインターネット
(グーグルドライブ)上に
ある画像を使います

はい けい がぞう
背景を画像にしてみよう！

がぞう
サンプル画像を見てみよう！

グーグル ドライブ
Google Driveへはデスクトップ上のアイコンをダブルクリックしてアクセスできます



はいけい

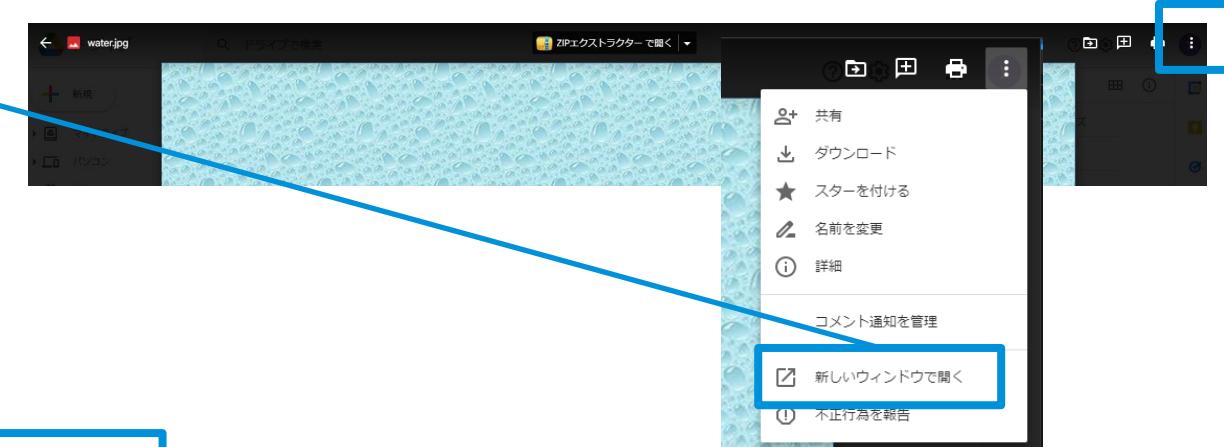
がぞう

背景を画像にしてみよう！

がぞう
サンプル画像を見てみよう！

がぞう せんよう ばんごう アイディー
画像にはそれぞれ専用の番号（ID）があります

- がぞう 画像名（water.jpg）をダブルクリックし、右上の「...」をクリック
- 新しいウィンドウで開く、をクリック



- がぞう アイディー
この部分が画像のIDです

drive.google.com/file/d/10Ub.../view

はいけい がぞう
背景を画像にしてみよう！

stage3.cssをみてみよう

がぞう はいけい
2. どう書いたら画像を背景にできそうか？

はいけい してい
→ 背景は **background-image** をつかって指定できる！

```
body {  
  background-size: cover;  
  background: black;  
  background-image:url( 画像がおいてある ばしょ );  
}  
  
canvas {  
  border: 2px solid white;  
}
```

じっさい

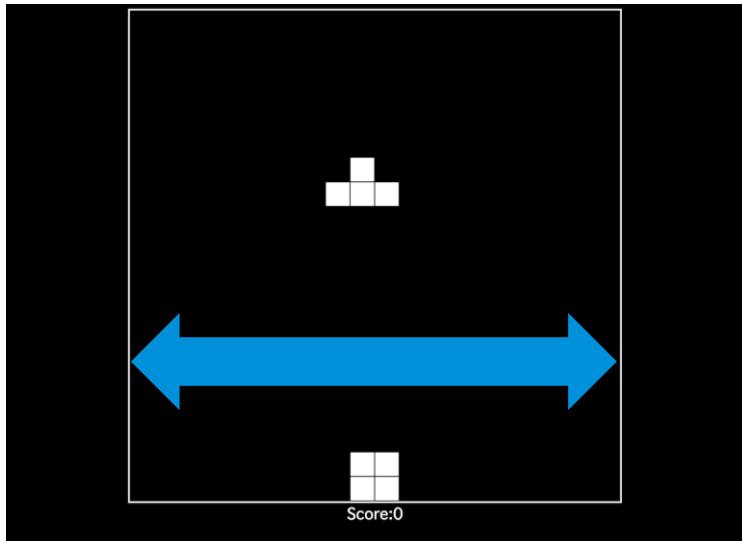
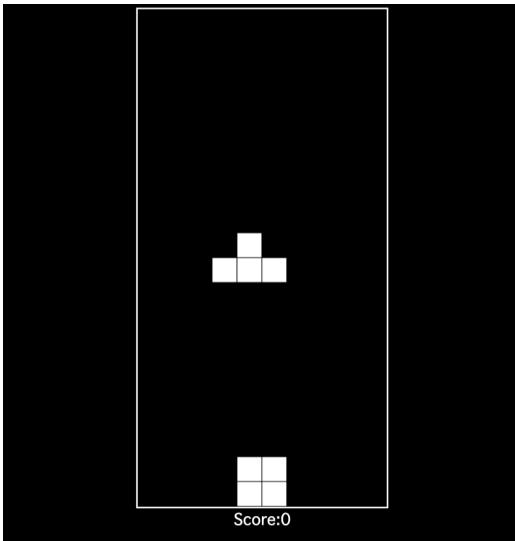
実際にやってみよう！

Stage 3 のチャレンジ問題を見てみてね！



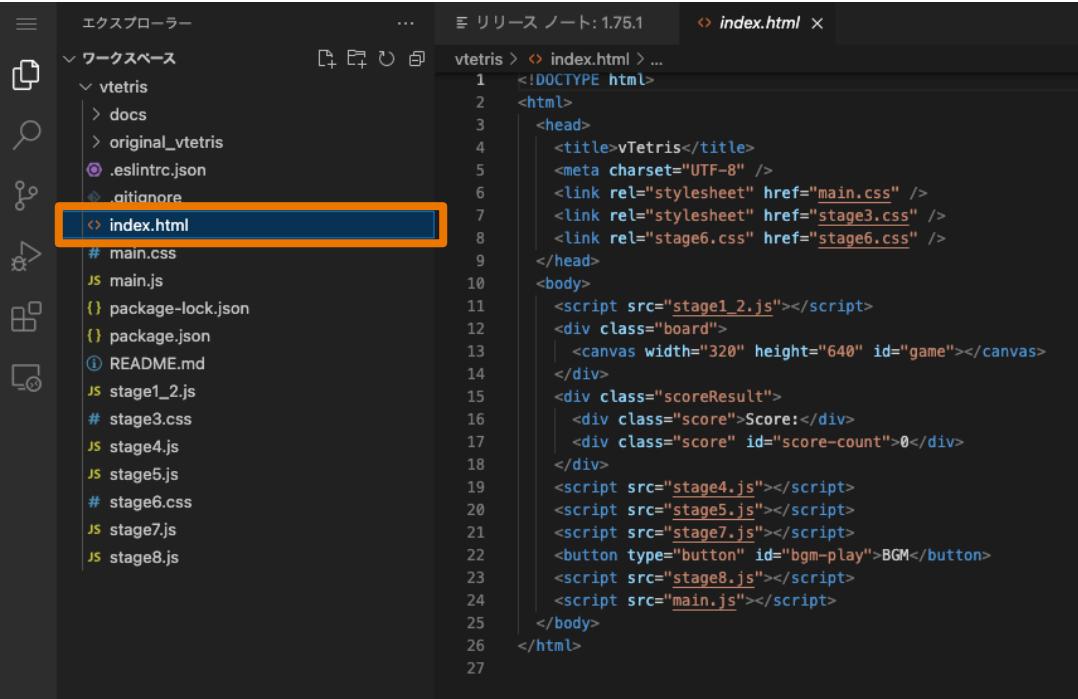
Stage 4

プレイフィールドを広げてみよう
stage 4 - プレイフィールド拡張

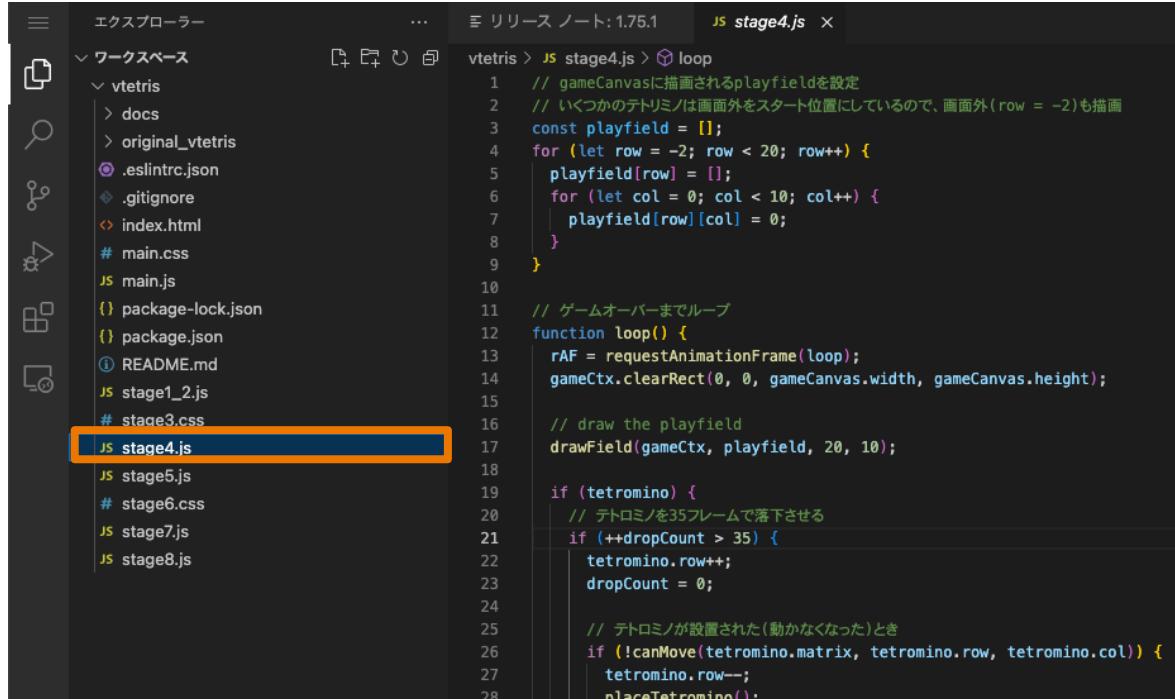


Stage 4でつかうソースコード

このStageでは『index.html』と『Stage4.js』をつかうよ！



```
<!DOCTYPE html>
<html>
  <head>
    <title>vTetris</title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="main.css" />
    <link rel="stylesheet" href="stage3.css" />
    <link rel="stage6.css" href="stage6.css" />
  </head>
  <body>
    <script src="stage1_2.js"></script>
    <div class="board">
      <canvas width="320" height="640" id="game"></canvas>
    </div>
    <div class="scoreResult">
      <div class="score">Score:</div>
      <div class="score" id="score-count">0</div>
    </div>
    <script src="stage4.js"></script>
    <script src="stage5.js"></script>
    <script src="stage7.js"></script>
    <button type="button" id="bgm-play">BGM</button>
    <script src="stage8.js"></script>
    <script src="main.js"></script>
  </body>
</html>
```



```
// gameCanvasに描画されるplayfieldを設定
// いくつかのテトリミノは画面外をスタート位置にしているので、画面外(row = -2)も描画
const playfield = [];
for (let row = -2; row < 20; row++) {
  playfield[row] = [];
  for (let col = 0; col < 10; col++) {
    playfield[row][col] = 0;
  }
}

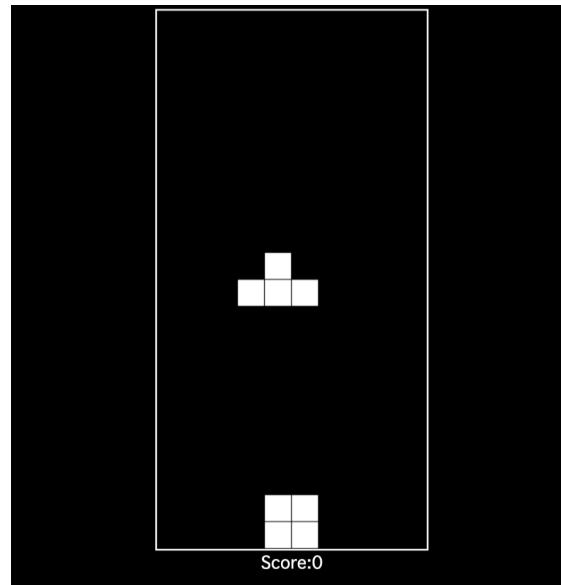
// ゲームオーバーまでループ
function loop() {
  RAF = requestAnimationFrame(loop);
  gameCtx.clearRect(0, 0, gameCanvas.width, gameCanvas.height);

  // draw the playfield
  drawField(gameCtx, playfield, 20, 10);

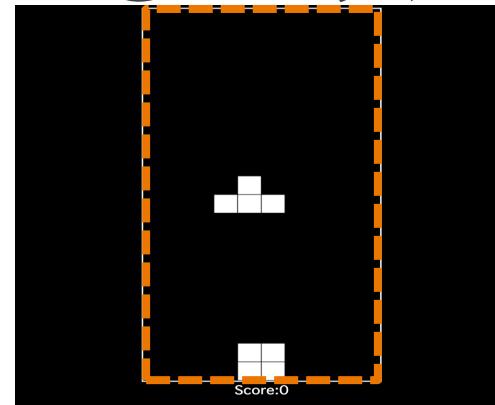
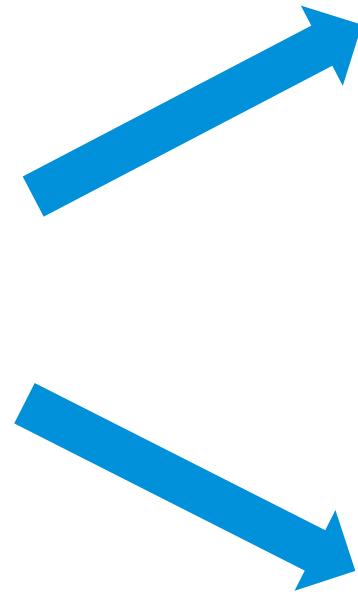
  if (tetromino) {
    // テトリミノを35フレームで落下させる
    if (++dropCount > 35) {
      tetromino.row++;
      dropCount = 0;

      // テトリミノが設置された(動かなくなった)とき
      if (!canMove(tetromino.matrix, tetromino.row, tetromino.col)) {
        tetromino.row--;
        placeTetromino();
      }
    }
  }
}
```

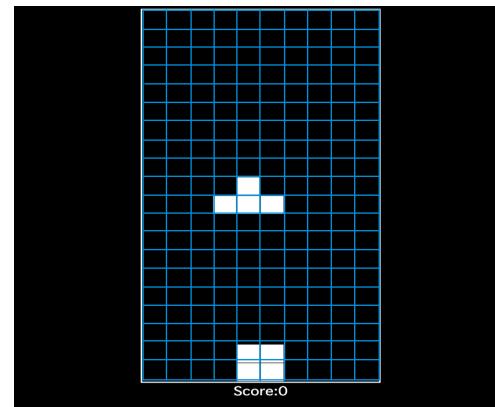
『見えるエリア』と『遊べるエリア』



なにげなく遊んでいるけど、実は・・

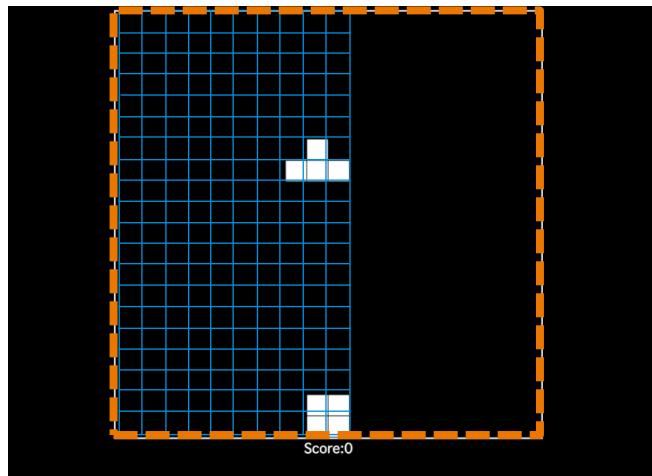


見えるエリア (Canvas) と、

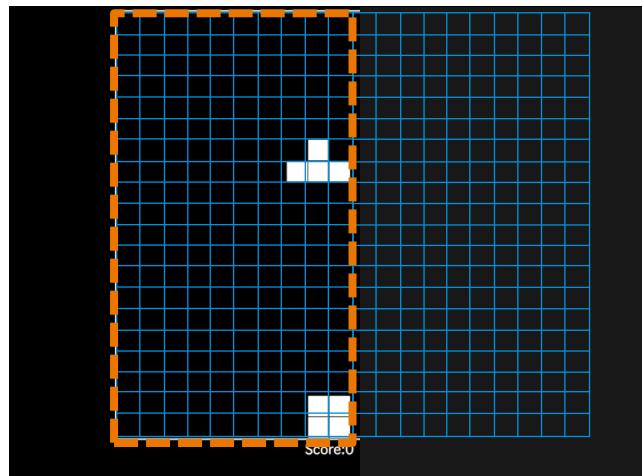


遊べるエリア (Playfield) でできているよ

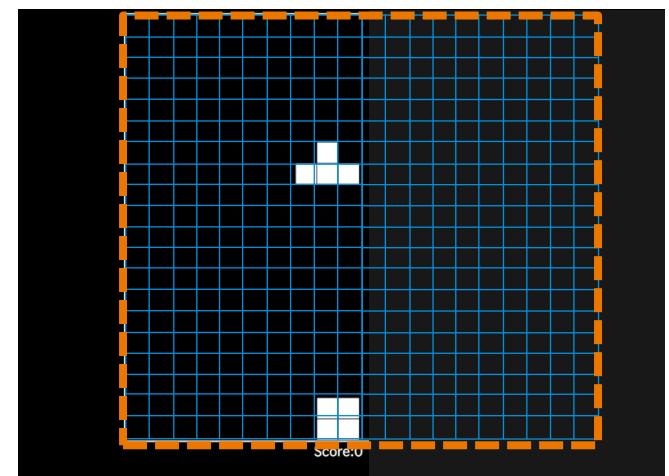
エリアがかわるとどうなるかな？



見えるエリア (Canvas)
だけを広げると・・



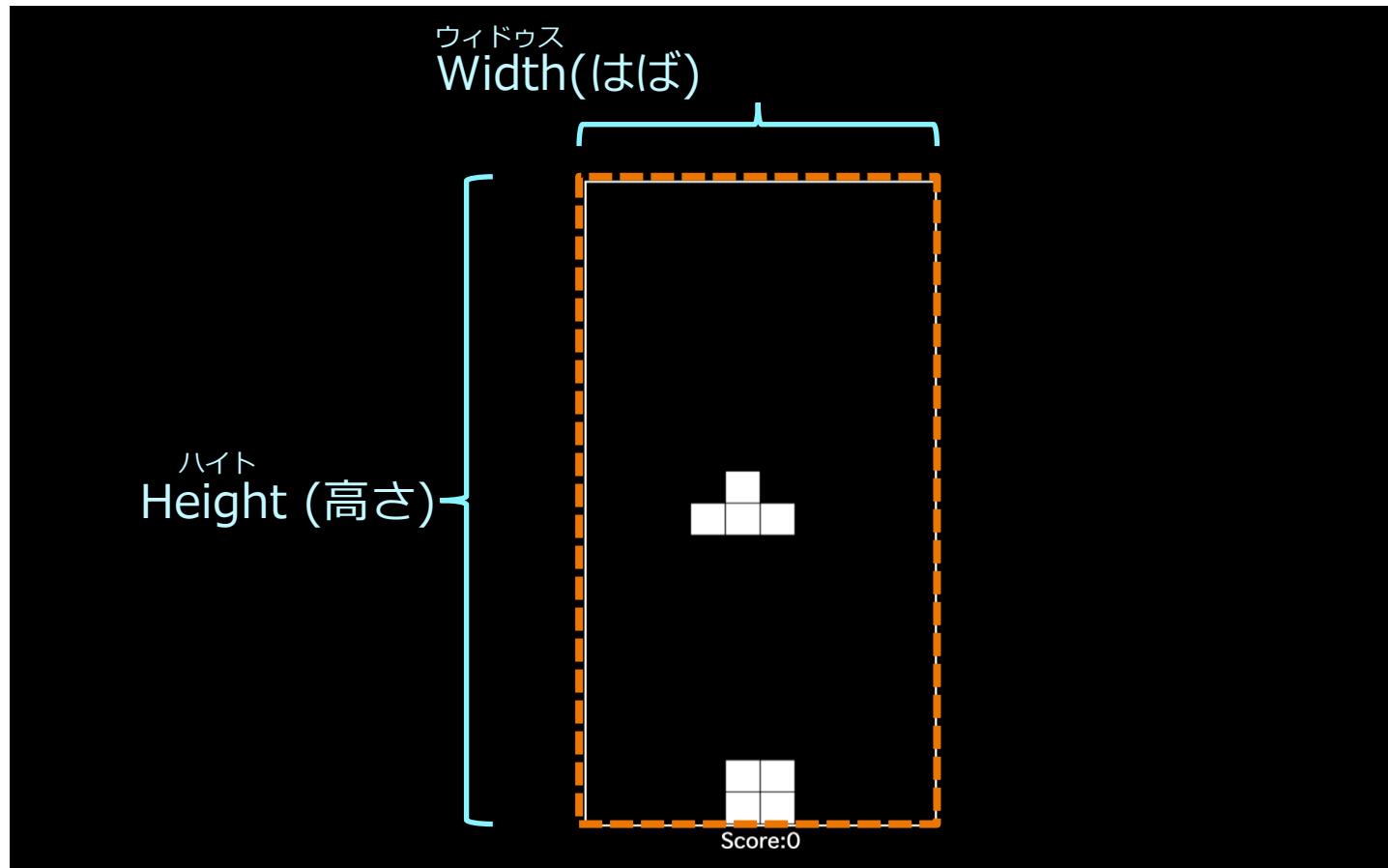
遊べるエリア (Playfield)
だけを広げると・・



両方を広げて合わせると・・

『見えるエリア』の決め方

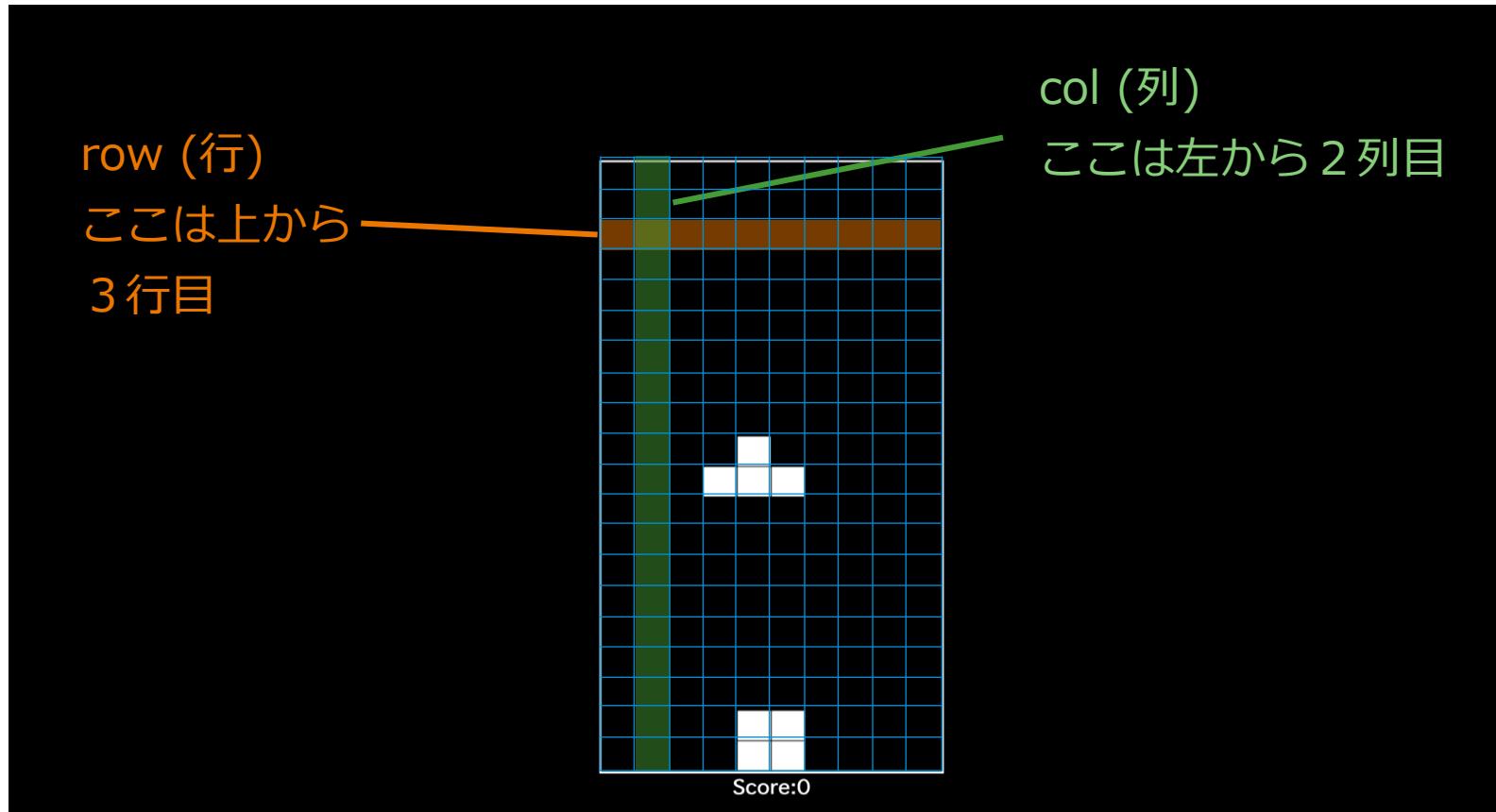
『index.html』のどこかに、見えるエリア(canvas)の大きさ(HeightとWidth)が書かれています
書かれている場所を見つけたら、数字をいじってデプロイしてみよう



『遊べるエリア』の決め方 1

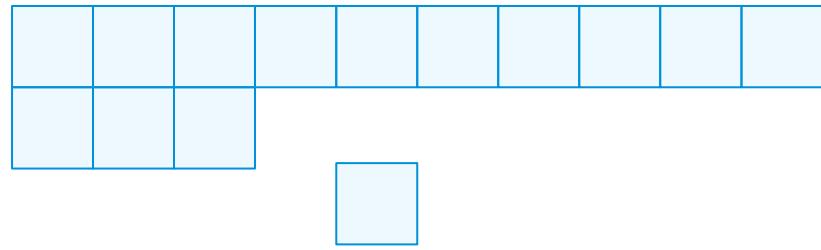
遊べるエリアは 「行:row」^{ロウ} と 「列:column」^{カラム} で決められているよ
にじげんはいれつ

このようなマス目を「行列」、プログラムの中では「二次元配列」という言い方もするよ



『遊べるエリア』の決め方 2

プログラムの中では、行列（マス）に「0」を入れて「遊べるエリア」を決めているよ
でも、1つ1つのマスに0を入れるプログラムを書くのはたいへん・・



遊べるエリア を作ろう!

プログラム
の中では

	0	1	2	3	4	5	6	7	8	9	col (列)
0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0								
2											
3											
row (行)	0	1	2	3	4	5	6	7	8	9	

まず、0行目の0列目に0、
次は、0行目の1列目に0、
次は、0行目の2列目に0、
次は・・・



『遊べるエリア』の決め方 3

プログラムでは、くり返し文 (for) を使って、行列(マス)に0を入れる作業をくり返しているよ

①のくり返し作業

0列目から9列目まで

0を入れる作業をくり返してね

```
for (let col = 0; col < 10; col++) {  
    0を入れる作業;  
}
```



col (列)									
0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0
1									
2									
3									
4									
5									
6									
7									
8									
9									

②のくり返し作業

0行目から 3行目まで

①の作業をくり返してね

```
for (let row = 0; row < 4; row++) {  
    ①のくり返し作業;  
}
```



col (列)									
0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0

『遊べるエリア』の決め方 4

①と②を組み合わせると、こんな風に書けるよ

『Stage4.js』から書かれているところを見つけて、

遊べるエリアを広げるにはプログラムをどうかえればよいか、考えてみよう

②のくり返し作業

0行目から3行目(4より小さい数)まで

①の作業をくり返してね

①のくり返し処理
しより

0列目から9列目(10より小さい数)まで

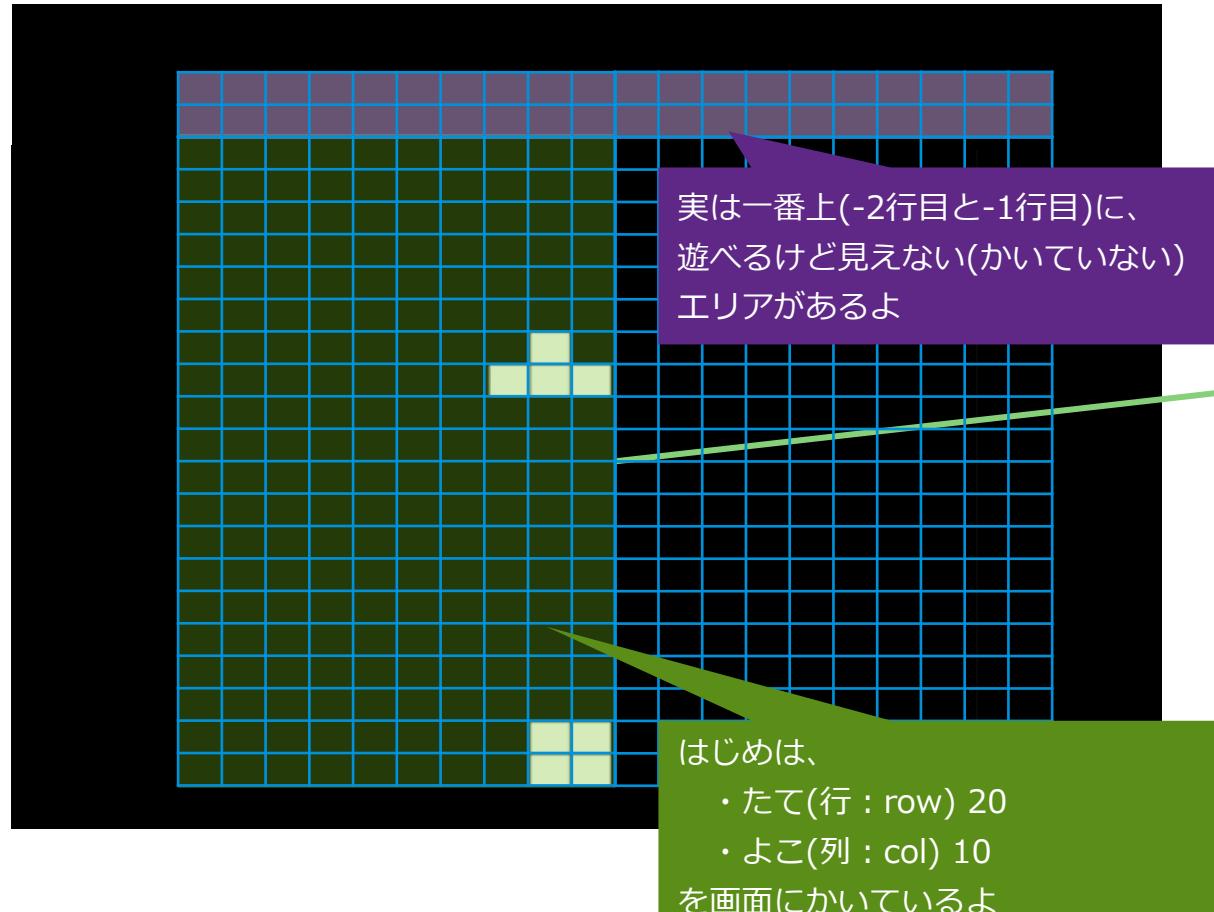
0を入れる作業をくり返してね

```
for (let row = 0; row < 4; row++) {  
    playfield[row] = [];  
  
    for (let col = 0; col < 10; col++) {  
        playfield[row][col] = 0;  
    }  
}
```

『遊べるエリア』の決め方 5

えが
ドロー

さいごに、遊べるエリア の中で、どの部分を画面に描く(draw) かを 決めます



えが
画面に描くプログラムはこんな風に書かれているよ
どのように直すか、考えてみよう

```
// draw the playfield  
ドローフィールド  
drawField(gameCtx, playfield, 行, 列);
```

問題をやってみよう！

Stage 4 の問題を見てみてね！



上級編！^{へん}



テトロミノを回転させよう



stage 5 – 回転

Stage 5でつかうソースコード

このStageでは『Stage5.js』をつかうよ！

The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar:
 - OPEN EDITORS: JS stage5.js (highlighted)
 - GITHUB: vtetris
 - docs
 - original_vtetris
 - .eslintrc.json
 - .gitignore
 - index.html
 - main.css
 - JS main.js
 - package-lock.json
 - package.json
 - README.md
 - JS stage1_2.js
 - # stage3.css
 - JS stage4.js
 - JS stage5.js (highlighted)
 - # stage6.css
- stage5.js** editor tab: JS stage5.js
- Content of stage5.js:

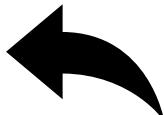
```
vtetris > JS stage5.js > ...
ichisuke55, 1 minute ago | 1 author (ichisuke55)
1 // 回転処理
2 // 参考サイト: https://www.bugbugnow.net/2019/04/array-rotate.html
3
4 // キーボード入力イベントのリッスン
5 document.addEventListener("keydown", function (e) {
6     if (isGameOver) return;
7     switch (e.code) {
8         // 左、右キー (移動)
9         case "ArrowLeft":
10            const col =
11                e.code === "ArrowLeft" ? tetromino.col - 1 : tetromino.col + 1;
12                if (canMove(tetromino.matrix, tetromino.row, col)) {
13                    tetromino.col = col;
14                }
15                break;
16        case "ArrowDown": // 下キー(落下)
17            const row = tetromino.row + 1;
18            if (!canMove(tetromino.matrix, row, tetromino.col)) {
19                tetromino.row = row - 1;
20            }
21        case "ArrowRight":
22            const col =
23                e.code === "ArrowRight" ? tetromino.col - 1 : tetromino.col + 1;
24                if (canMove(tetromino.matrix, tetromino.row, col)) {
25                    tetromino.col = col;
26                }
27                break;
28    }
29}
```

『回転』は何をしているのかな？

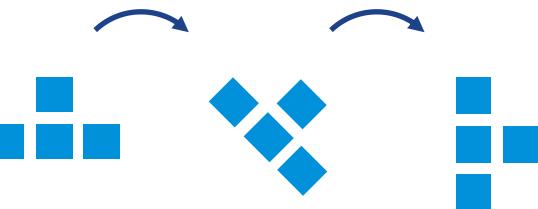
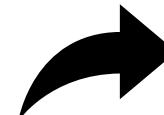
クリック！

space

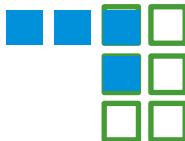
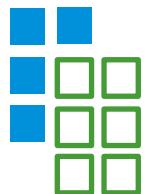
キーをクリックするよ



回転って
何してる？

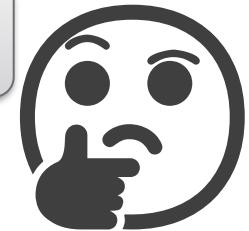


コロコロころがるなあ



回したら、かべにめりこんじゃった！

1つずつ、考えるんだな



① キー入力方法を作ろう

ゼロからプログラミング? すでにある機能を使える?



←→キーは使えるぞ！



どのキーがどの文字？

```
4 // キーボード入力イベントのリッスン
5 document.addEventListener("keydown", function (e) {
6     if (isGameOver) return;
7     switch (e.code) {
8         // 左、右キー (移動)
9         case "ArrowLeft":
10        case "ArrowRight":
11            const col =
12                e.code === "ArrowLeft" ? tetromino.col - 1 : tetromino.col + 1;
13                if (!isAvailableMove(col)) break;
```

ArrowLeftがあやしいなあ
Arrowは “やじるし”
Leftは “ひだり”
←キーのことかな？

キーボードのことが何か書かれているぞ

①キー入力方法を作ろう(つづき)

どのキーがどの文字？

ArrowLeftが怪しいなあ
Arrowは“やじるし”
Leftは“ひだり”
←キーのことかな？

キーボードイベントのコード (e.code)

backspace	Tab	Enter	Shift	Ctrl
“Backspace”	“Tab”	“Enter”	“ShiftLeft”	“ControlLeft”
alt	Esc	Space	Insert	Del
“AltLeft”	“Escape”	“Space”	“Insert”	“Delete”
←	↑	→	↓	Fキー
“ArrowUp”	“ArrowLeft”	“ArrowRight”	“ArrowDown”	“KeyF”

それぞれのキーにはなまえ（コード）があるようだ
使いたいキーのなまえをさがしてみよう

- ・JavaScript | キーボードの入力イベントを実装（じっそう）する方法（ほうほう） (<https://1-notes.com/javascript-addeventlistener-key-ivent/>)
- ・キーボードイベントのコードを調べられるページ (<https://developer.mozilla.org/ja/docs/Web/API/KeyboardEvent/code>)

②回転のしくみを作ろう

ゼロからプログラミング?
きのう
すでにある機能を使える?



仕組みづくり
アルゴリズムを考えよう



ブロックをおぼえて
いるのはだれ?



アルゴリズムって
ほうほう
計算方法のことだよね

回転させるは英語でrotate
えいご ローテート
图形はrotate()で回転できる
でも、テトロミノは图形かな？

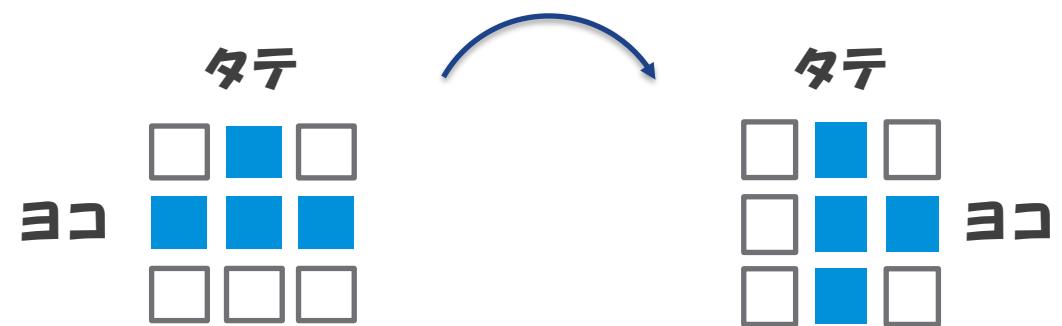
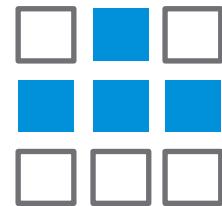
まずはテトロミノを
1つ回転してみましょう



②回転のしくみを作ろう(つづき)

```
15   T: [  
16     [0, 1, 0],  
17     [1, 1, 1],  
18     [0, 0, 0],  
19   ],
```

テトロミノは0と1だ!
図形じゃなくて、ブロック
を組み合わせているんだね

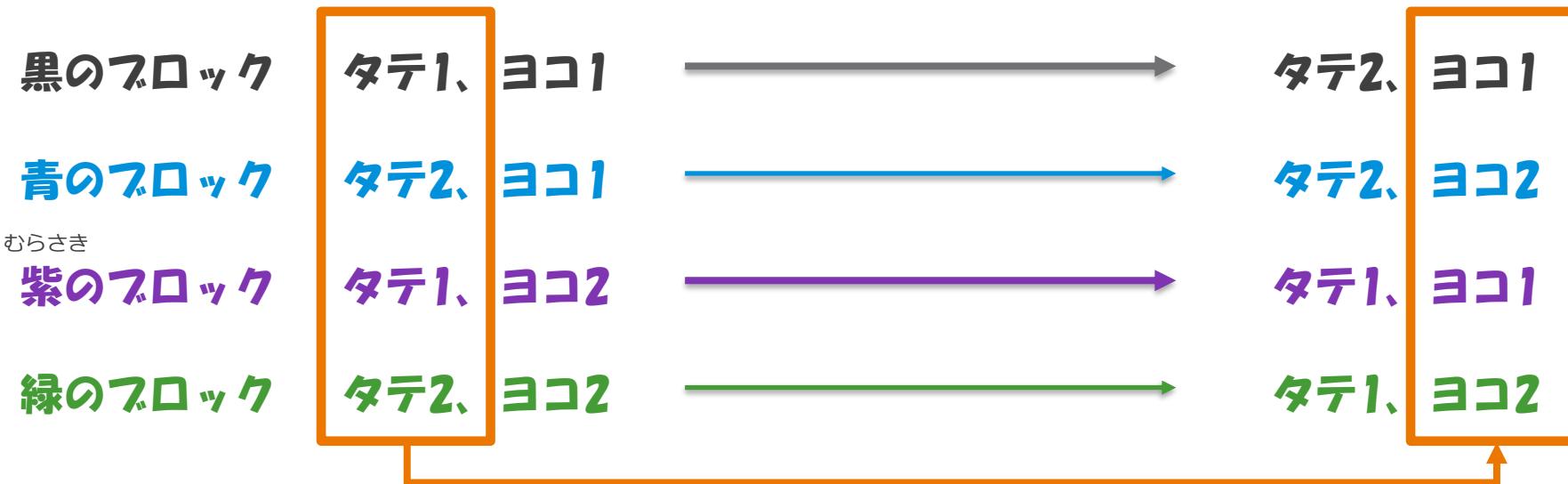
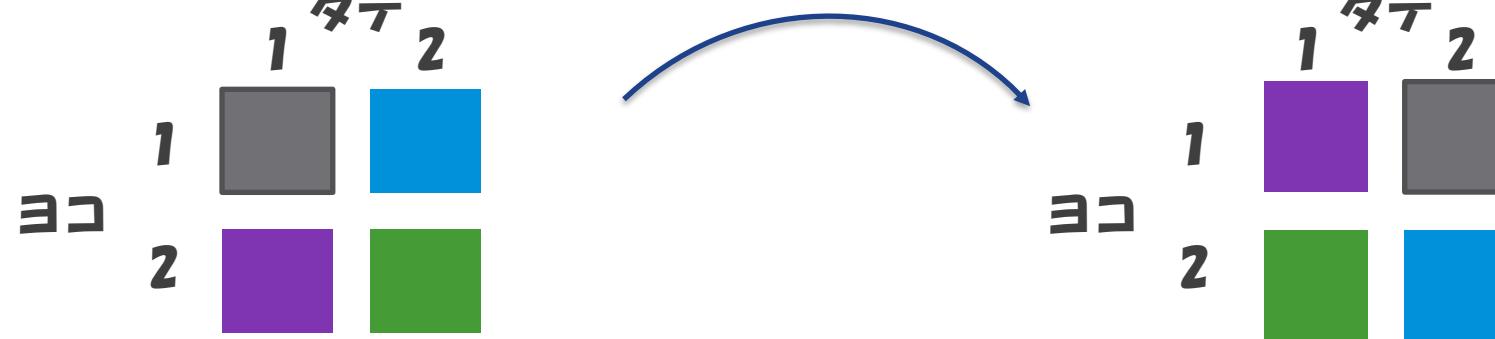


ヨコがタテにいって、
タテがヨコに行ったわ

なにかルールがありそう
分かりやすくしてみよう

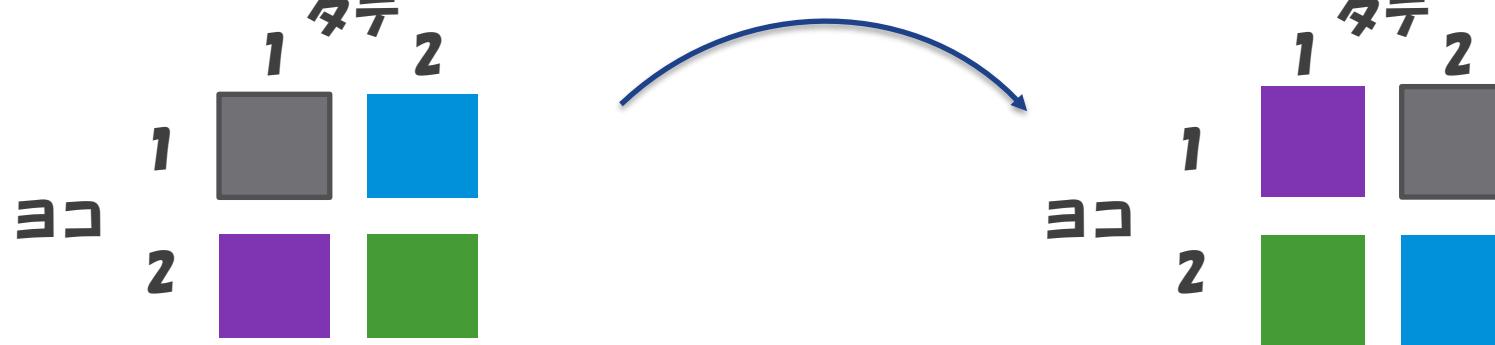


②回転のしくみを作ろう(つづき)



タテとヨコが入れかわってるって、いってたなあ 😱

②回転のしくみを作ろう(つづき)



笑笑こっちはどうかな？入れかわっているかな？

②回転のしくみを作ろう(つづき)

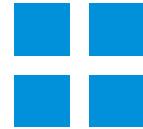


1が2、2が1に入れかわっている

でもテトロミノは3つや4つのものもあるね
4つのだと1が4、4が1。2が3、3が2かな？

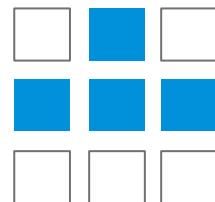
かくにん
ノートに書いて確認してみよう！

②回転のしくみを作ろう(つづき)



ブロックが2この場合

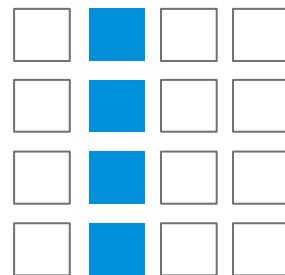
$1 \rightarrow 2$
 $2 \rightarrow 1$



ブロックが3この場合

$1 \rightarrow 3$
 $2 \rightarrow 2$
 $3 \rightarrow 1$

足し算、引き算、かけ算、わり算
答えが同じになるものはあるかな？



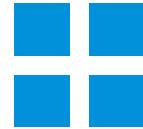
ブロックが4この場合

$1 \rightarrow 4$
 $2 \rightarrow 3$
 $3 \rightarrow 2$
 $4 \rightarrow 1$

あれなら全部おなじだ！



②回転のしくみを作ろう(つづき)

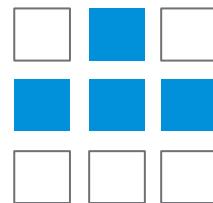


ブロックが2この場合

$1 \rightarrow 2$
 $2 \rightarrow 1$



足したら3
3-元の数で右の数になるぞ

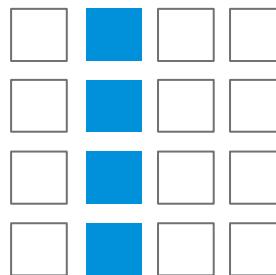


ブロックが3この場合

$1 \rightarrow 3$
 $2 \rightarrow 2$
 $3 \rightarrow 1$



足したら4
4-元の数で右の数になるぞ



ブロックが4この場合

$1 \rightarrow 4$
 $2 \rightarrow 3$
 $3 \rightarrow 2$
 $4 \rightarrow 1$



足したら5
5-元の数で右の数になるぞ

②回転のしくみを作ろう(つづき)

ゼロからプログラミング?
きのう
すでにある機能を使える?

仕組みづくり
アルゴリズムを考えよう

ブロックをおぼえて
いるのはだれ?

まとめ

1. タテの数はヨコの数になるぞ
2. ヨコの数からタテの数にするには、引き算を使うぞ
3. 引き算に使う元の数はブロックの長さより1つ多いぞ
『(ブロックの長さ+1) - ヨコの数』になるんだね

じゅんばん
配列は0, 1, 2の順番
プログラミングでは
(ブロックの長さ-1) - ヨコの数
になるぞ

②回転のしくみを作ろう(つづき)

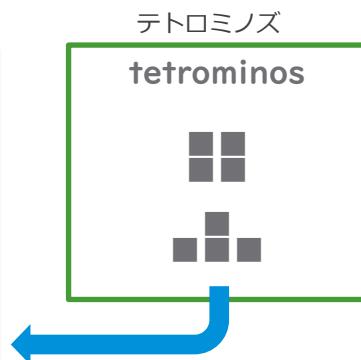
ゼロからプログラミング?
きのう
すでにある機能を使える?



仕組みづくり
アルゴリズムを考えよう



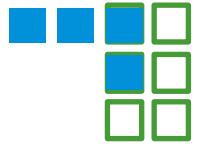
ブロックをおぼえて
いるのはだれ?



ソースコードからわかったわ



③ブロックがめりこまないようにしてみよう



回転したあとにチェック！



ゼロからプログラミング?
すでにある機能を使える?

```
7  switch (e.code) {
8    // 左、右キー(移動)
9    case "ArrowLeft":
10   case "ArrowRight":
11     const col =
12       e.code === "ArrowLeft" ? tetromino.col - 1 : tetromino.col + 1;
13     if (canMove(tetromino.matrix, tetromino.row, col)) {
14       tetromino.col = col;
15     }
16     break;
17   case "ArrowDown": // 下キー(落下)
18     const row = tetromino.row + 1;
19     if (!canMove(tetromino.matrix, row, tetromino.col)) {
20       tetromino.row = row - 1;
21       placeTetromino();
22       return;
23     }
24     tetromino.row = row;
25     break;
26   case "ArrowUp": // 上キー(ハードドロップ)
27     while (canMove(tetromino.matrix, tetromino.row + 1, tetromino.col)) {
28       tetromino.row++;
29     }
30     break;
```

キャン ムーブ

他のキーではcanMoveを使っているみたい

その場で回転だと、+1とか-1はいるのかな？

さあ、書いてみよう！

書くときのポイント！

- キーをおす → 回転したかたちを出す → チェックする → かたちを画面にもどす
テトロミノ マトリックス
- 今のtetromino.matrixを読みこむための箱はどう作ればよいかな？
 - ヒント：配列の中に、配列を入れるよ。二次元配列っていうんだ
レングス
 - ヒント：テトロミノによってブロックの長さ (**length**) がかわるけど、どうしたらいいんだろう？
- 何回もくり返しをするときは、何を使うのかな？
 - くり返しをくり返すときはどうしたらよいか考えよう

こたえを見るのもOK！

- 『どうしてこう書くのかな？』が分かることが大切
- どこかを書きかえたり、デバッグしたりして、「何をしようとしているのか」を知ろう！

さあ、書いてみよう！

Stage 5 の問題を見てみてね！



仕上げ！

文字を入れよう！

stage 6 – 名前を入れよう

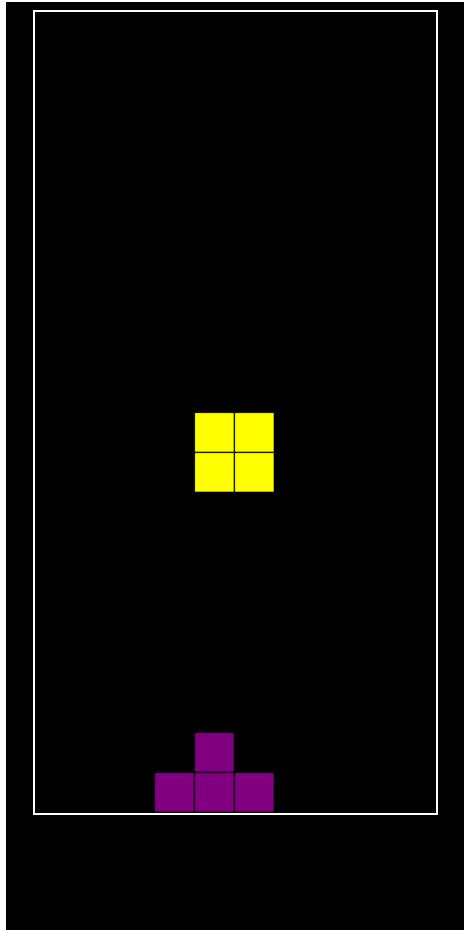
Stage 6でつかうソースコード

このStageでは『index.html』と『stage6.css』をつかうよ！

The screenshot shows a code editor interface with a dark theme. On the left is a file tree for a project named "VTETRIS". The tree includes files like "activity", "docs", "original_vtetris", "scripts", ".eslintrc.json", ".gitignore", "answer_stage1_2.js", "answer_stage5.js", "index.html", "main.css", "main.js", "package-lock.json", "package.json", "README.md", "stage1_2.js", "stage3.css", "stage4.js", "stage5.js", "stage6.css", "stage7.js", and "stage8.js". The file "stage6.css" is currently selected and its content is displayed in the main editor area. The content of stage6.css is:

```
# stage6.css > /* explainText classを変更して、文字を装飾してみよう */
.explainText {
  font-family: monospace;
  color: #0000;
  display: flex;
  align-items: center;
  justify-content: left;
}
```

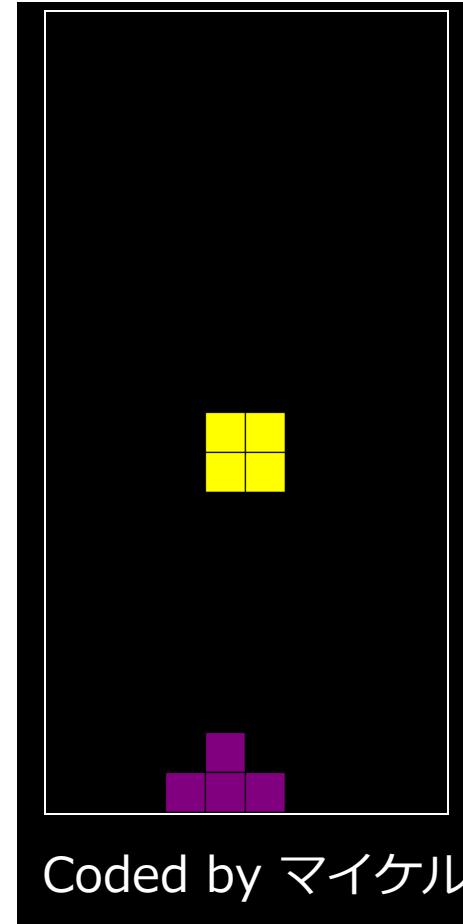
The file "index.html" is also visible in the editor, though its content is not shown in the screenshot.



ゲーム画面だけだとさびしいなあ

自分の名前も入れたいなあ

よし、文字を入れよう！



Coded by マイケル

Canvasの外にHTMLの<Body>がある



ゲームはHTMLのBodyの中に書かれ
ているのね

せいてき
静的なコンテンツ（へんかしない
じょうほう）を見せるには、HTMLに
そのまま書くとかんたんね

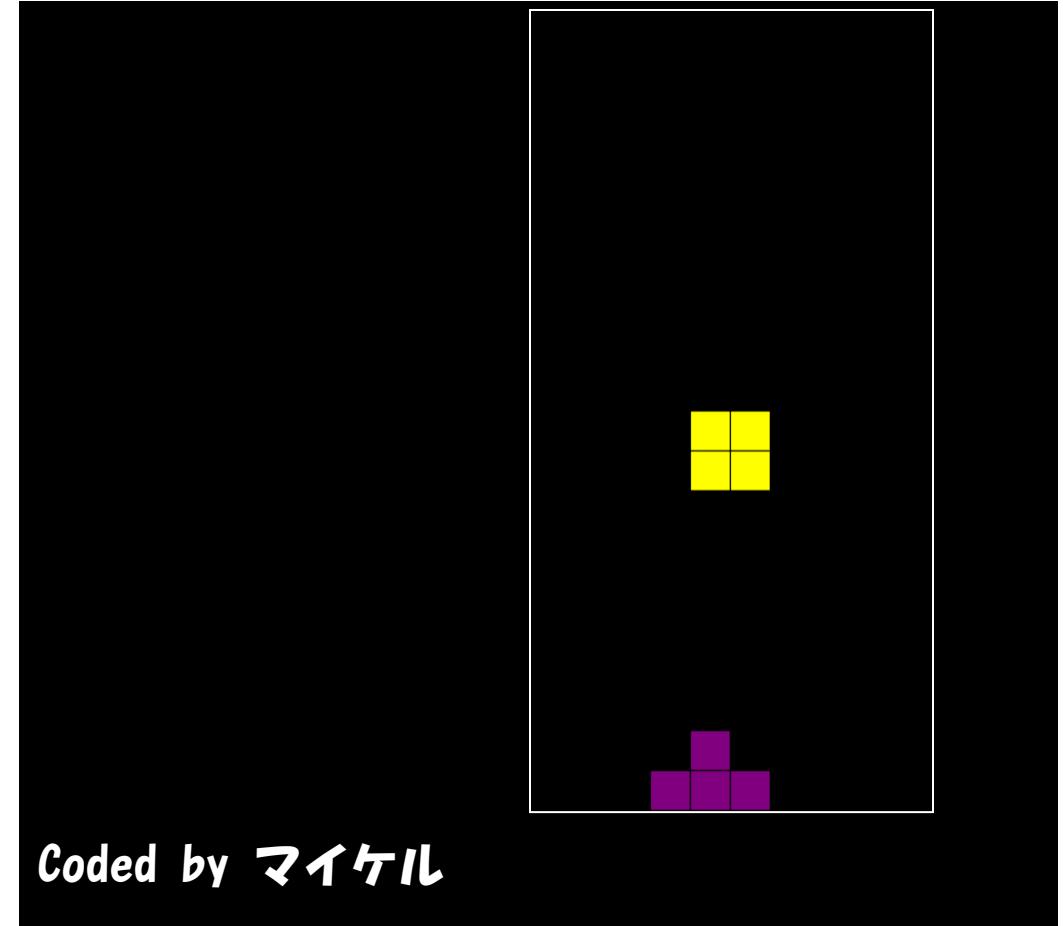


HTMLとCSS



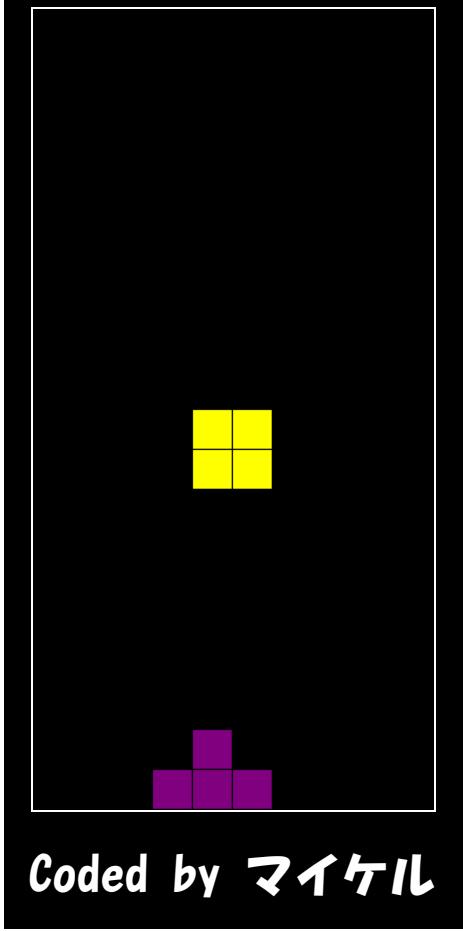
文字が左はしにいっちゃった
真ん中におきたいなあ

“HTMLの書き方”があるみたい
CSSを使うとわかりやすくできるって
聞いたわ



Coded by マイケル

やりたいことをまとめよう

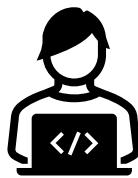


- ① 文字をいれたい
- ② 文字は真ん中に表示したい^{ひょうじ}
- ③ 文字の色は黒じゃない色がいいな

· · · · ·

HTMLでの書き方

HTMLの世界では、文字をはさむことで色々な見た目にできる



表示する文字



- ① 文字をいれたい
 - ② 文字は真ん中に表示したい
 - ③ 文字の色は黒じゃない色がいいな
-
-
- 真ん中は英語でcenter
えいご センター^{ひょうじ}
カラー ホワイト
色 (color) は白 (white) にしよう

CSSで読みやすくしよう！

CSSは「書き方のルールをまとめたもの」

プログラミング

きのう
ブロックを消す機能

きのう
ブロックを回転する機能

CSS
(スタイルシート)

Stage 3 でも
つかったね！

- 文字の見た目のルール
- 文字のしゅるい
 - 文字のいろ
 - 文字のばしょ

CSSの基本（きほん）

https://developer.mozilla.org/ja/docs/Learn/Getting_started_with_the_web/CSS_basics

CSSで書くとどうなるかな？

htmlファイル

```
<p class = "myClass">ブロック落としゲーム</p>
```

クラス
CSSのclassのルールでへんかするよ

クラス
classを書くときはあたまに
“.”をわすれずにつけよう

CSSのclass

```
.myClass {  
    文字のぼしよ  
    文字の色  
    :  
}
```

問題をやってみよう！

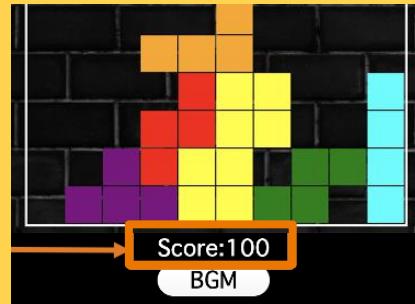
Stage 6 の問題を見てみてね！



Next ステージ

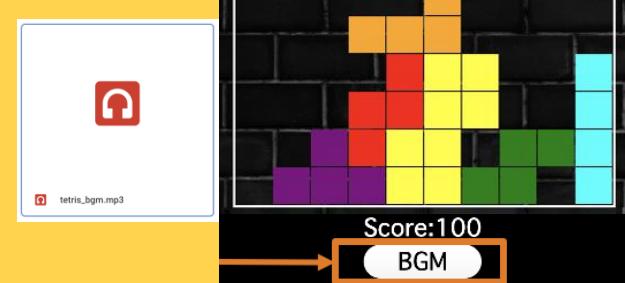
Stage7

～スコアをつけよう～



Stage8

～BGMをつけよう～



Stage9

～オリジナルを作ろう～





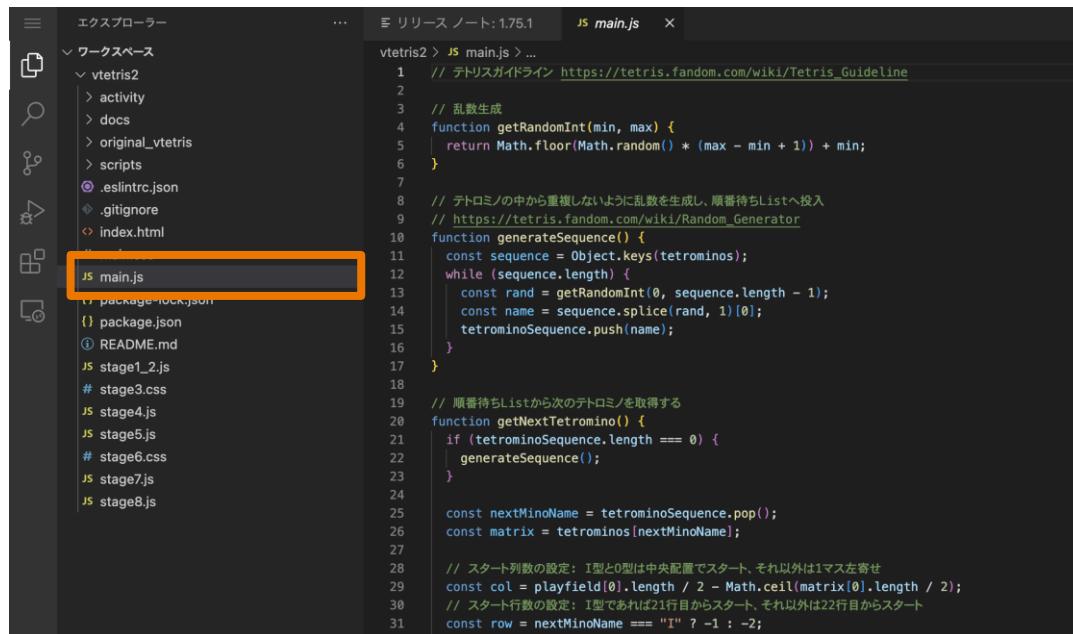
Stage 7

スコアをつけよう

stage 7 – スコアの計算

Stage 7でつかうソースコード

このStageでは『main.js』と『stage7.js』をつかうよ！

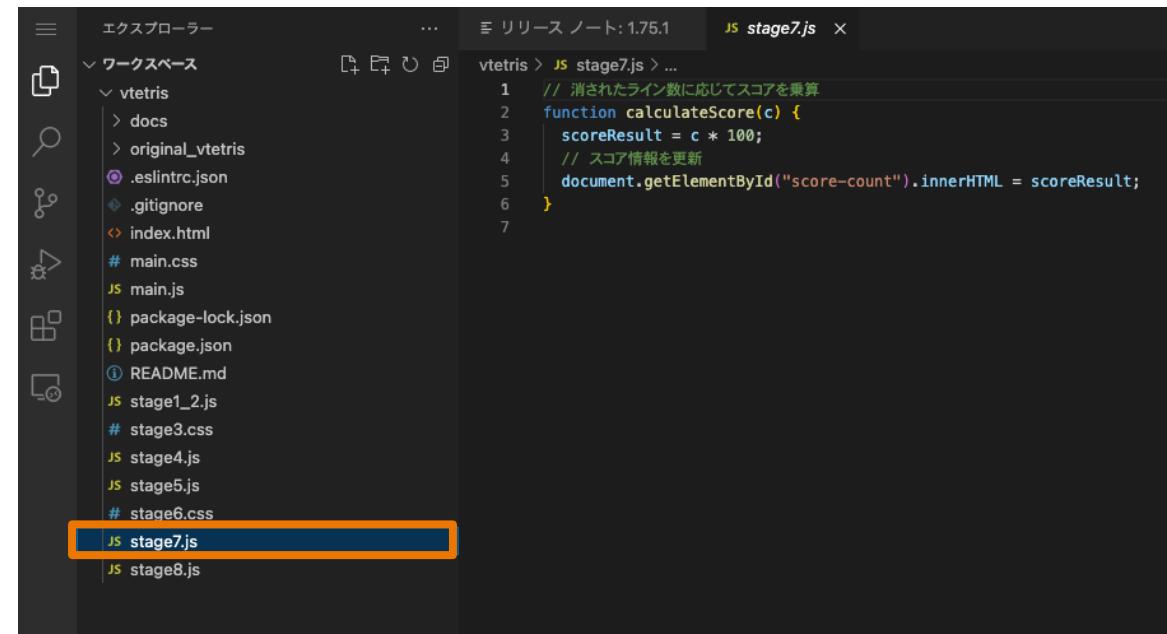


```
// テトリスガイドライン https://tetris.fandom.com/wiki/Tetris_Guideline
// 亂数生成
function getRandomInt(min, max) {
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

// テトロミノの中から重複しないように乱数を生成し、順番待ちListへ投入
// https://tetris.fandom.com/wiki/Random_Generator
function generateSequence() {
  const sequence = Object.keys(tetrominos);
  while (sequence.length) {
    const rand = getRandomInt(0, sequence.length - 1);
    const name = sequence.splice(rand, 1)[0];
    tetrominoSequence.push(name);
  }
}

// 順番待ちListから次のテトロミノを取得する
function getNextTetromino() {
  if (tetrominoSequence.length === 0) {
    generateSequence();
  }
  const nextMinoName = tetrominoSequence.pop();
  const matrix = tetrominos[nextMinoName];

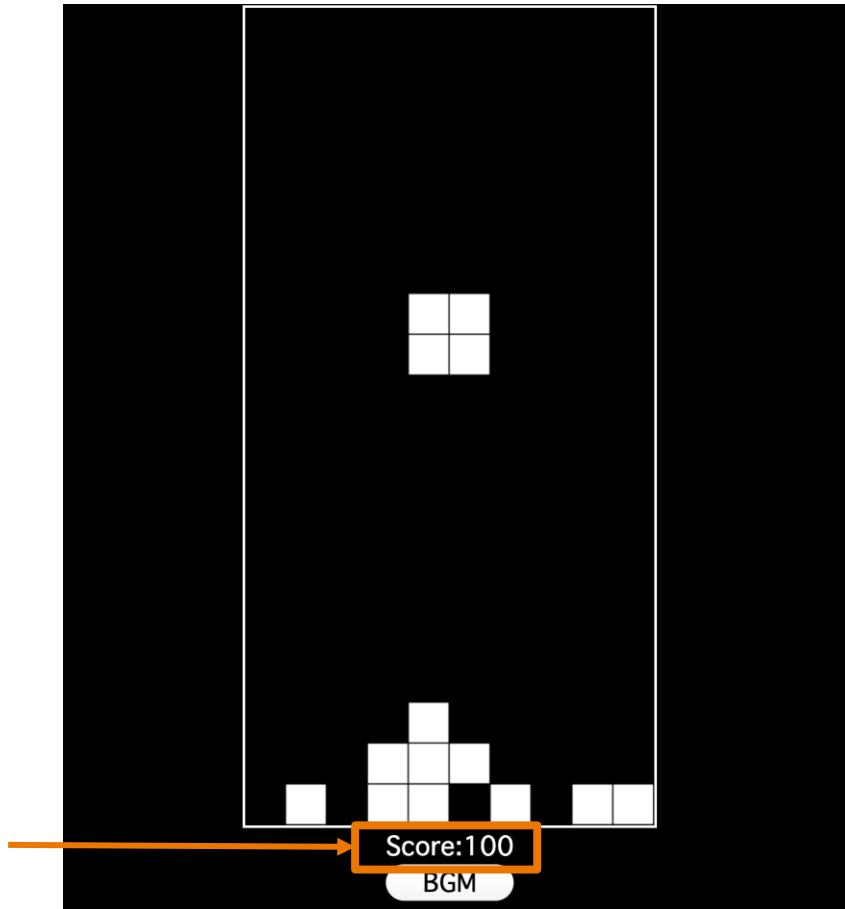
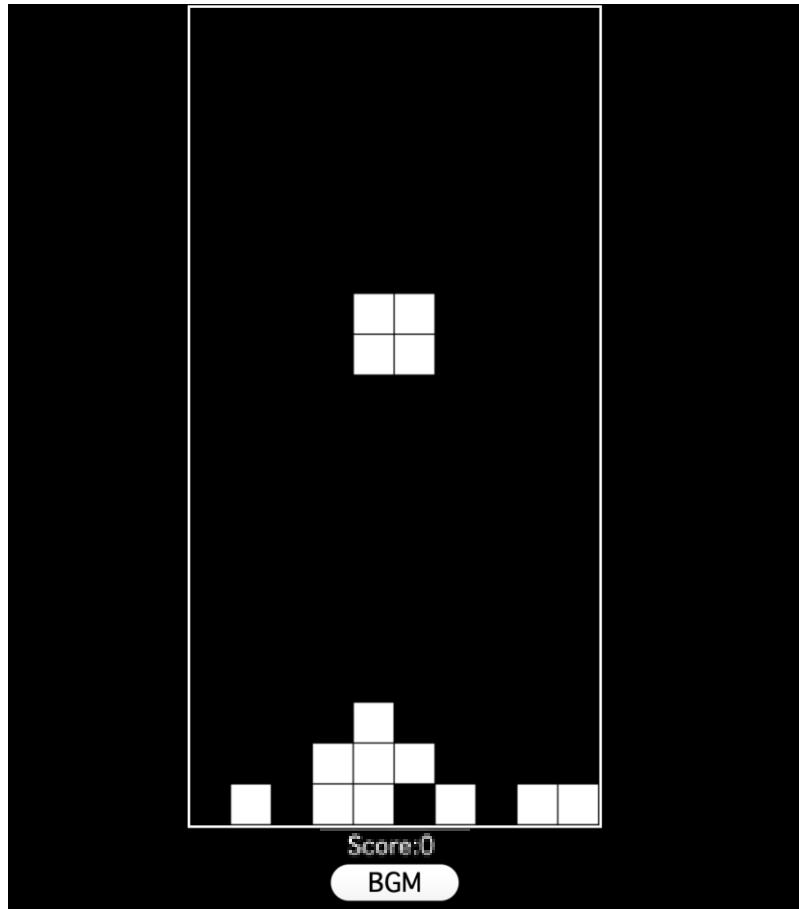
  // スタート列数の設定: I型とO型は中央配置でスタート、それ以外は1マス左寄せ
  const col = playfield[0].length / 2 - Math.ceil(matrix[0].length / 2);
  // スタート行数の設定: I型であれば21行目からスタート、それ以外は22行目からスタート
  const row = nextMinoName === "I" ? -1 : -2;
```



```
// 消されたライン数に応じてスコアを乗算
function calculateScore(c) {
  scoreResult = c * 100;
  // スコア情報を更新
  document.getElementById("score-count").innerHTML = scoreResult;
}
```

か

スコア(Score)を計算して画面に描いてみよう



スコアをつけるには、何を考えよう？

何をしたらスコアが増えるようにする？たとえば・・^ふ

- ・ブロック(ライン)を消したとき
- ・ブロックを早く落としたとき
- ・ブロックをたくさん回転させとき

ここでは、「ブロック (ライン) を消したらスコアが増える」^ふ
を考えてみよう

スコアをつけるには、何を考えよう？

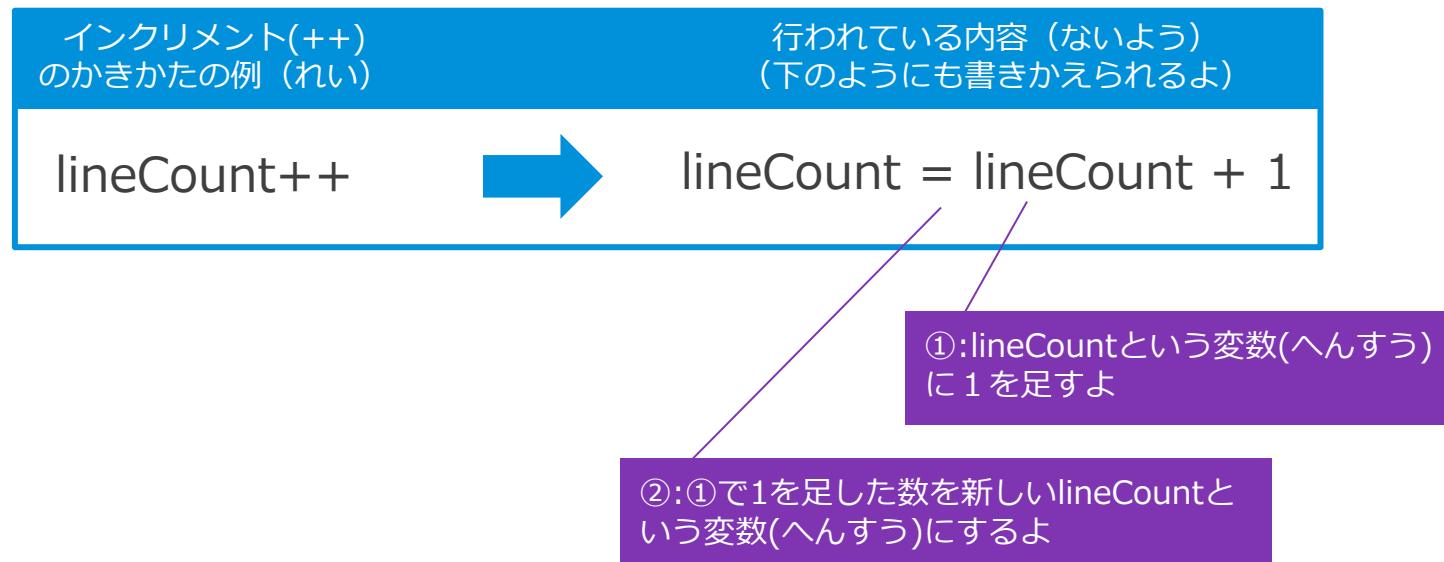
「ブロック(ライン)を消したらスコアが増える」ようにするには、
まず、消したブロック(ライン)の数を数えるとよさそうだね
まず、プログラムの中でブロックを消す部分を見つけてみよう。
「main.js」の中に書かれている//コメント文がヒントになるよ

```
// したから上に向かって揃ったラインを確認する
for (let row = playfield.length - 1; row >= 0; ) {
    if (playfield[row].every((cell) => !!cell)) {
        // 消されたライン数のカウント
        // 消されたラインより上を下にずらす
        for (let r = row; r >= 0; r--) {
            for (let c = 0; c < playfield[r].length; c++) {
                playfield[r][c] = playfield[r - 1][c];
            }
        }
    } else {
        row--;
    }
}
```



ブロック（ライン）を消した数をカウントしてみよう

ブロックを消した時に、「数を数える（カウントする）」にはどうしたらいいかな？ プログラムでは、インクリメント（++）という書き方がよく使われるよ。

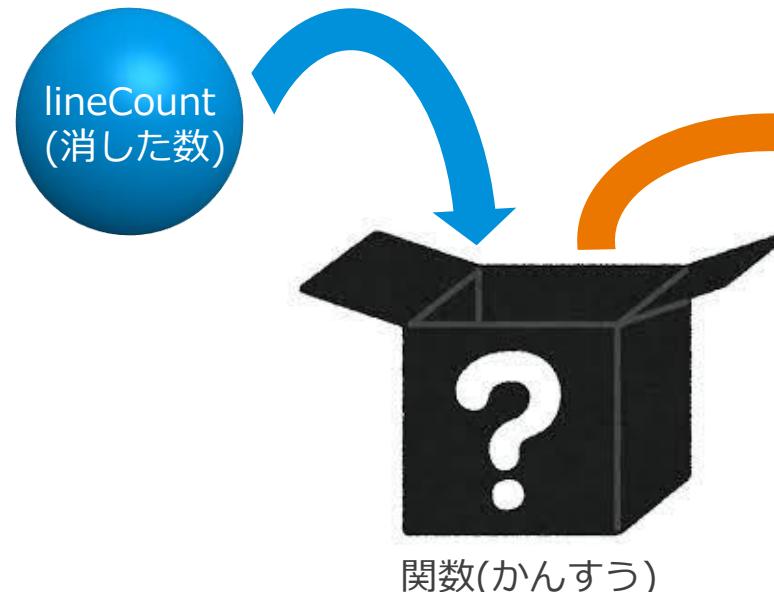


スコアを計算して、画面に描いてみよう

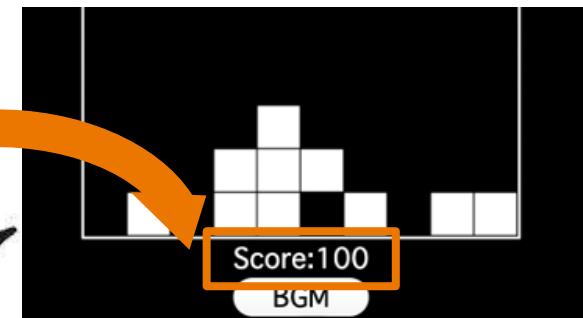
ブロックを消した数が数えられたら、スコアを計算してみよう。

プログラムでは「**関数**」という不思議な箱がスコアを計算して、画面に描いてくれるよ。

「消した数」を入れると・・

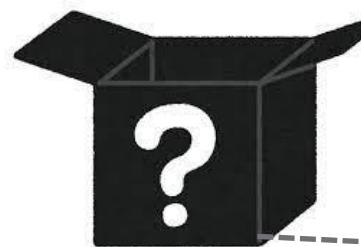


スコアを計算して、画面に描いてくれるよ



かんすう 「関数」の中をのぞいてみよう

かんすう
「関数」にどのような仕事をさせるか、プログラムで書くことができるよ。スコア計算をする「関数」の中をのぞいてみよう。
「stage7.js」に書かれているよ



スコア計算をする「関数(かんすう)
calculateScore()

①: 箱の中に「c」という数（消した数）を入れるよ

```
// 消されたライン数に応じてスコアを乗算
function calculateScore(c) {
    scoreResult = c*0;
    // スコア情報を更新
    document.getElementById("score-count").innerHTML = scoreResult;
}
```

②: 「c」の数を使ってスコアを計算するよ
※この後のページでもんだいがあるよ

③: 計算したスコアを画面に描くよ

【もんだい】消したラインの数を数えるための じゅんびをしよう

たいじょう

対象のファイルは『main.js』の165-170行目あたりです

lineCount というところで、消したラインの数を数えるよ。

一番はじめ（数え始める前）はどうなっていればいいかな？

```
//stage7
① // 消したライン合計数
let scoreResult = 0; // スコアの合計数
```

①一番はじめ（数え始める前）はどうなっていればいいかな？

let lineCount=0;

let lineCount=1;

let lineCount++;

let lineCount+1;

【こたえ】消したラインの数を数えるための じゅんびをしよう

たいじょう

対象のファイルは『main.js』の165-170行目あたりです

lineCount というところで、消したラインの数を数えるよ。

一番はじめ（数え始める前）はどうなっていればいいかな？

```
//stage7
let lineCount = 0; // 消したライン合計数
let scoreResult = 0; // スコアの合計数
```

①一番はじめ（数え始める前）はどうなっていればいいかな？

let lineCount=0;

let lineCount=1;

let lineCount++;

let lineCount+1;

【もんだい】スコアをつけるために ブロック（ライン）を消した数を数えてみよう

たいじょう

対象のファイルは『main.js』です

```
// したから上に向かって揃ったラインを確認する
for (let row = playfield.length - 1; row >= 0; ) {
  if (playfield[row].every((cell) => !!cell)) {
    ① // 消されたライン数のカウント
    // 消されたラインより上を下にずらす
    for (let r = row; r >= 0; r--) {
      for (let c = 0; c < playfield[r].length; c++) {
        playfield[r][c] = playfield[r - 1][c];
      }
    }
  } else {
    row--;
  }
}
```

①ブロックを消した数を数えるには？

lineCount=1;

lineCount+1;

lineCount++;

lineCount=0;

【こたえ】スコアをつけるために ブロック（ライン）を消した数を数えてみよう

たいじょう

対象のファイルは『main.js』です

```
// したから上に向かって揃ったラインを確認する
for (let row = playfield.length - 1; row >= 0; ) {
  if (playfield[row].every((cell) => !!cell)) {
    lineCount++; // 消されたライン数のカウント
    // 消されたラインより上を下にずらす
    for (let r = row; r >= 0; r--) {
      for (let c = 0; c < playfield[r].length; c++) {
        playfield[r][c] = playfield[r - 1][c];
      }
    }
  } else {
    row--;
  }
}
```

①ブロックを消した数を数えるには？

lineCount=1;

lineCount+1;

lineCount++;

lineCount=0;

【もんだい】ブロック（ライン）を消した数の100倍の数をスコアに出す関数を作ってみよう

たいじょう

対象のファイルは『stage7.js』です

かけ算は [*] で表すことができるよ

```
おう          ジョウザン（かけ算のこと）
// 消されたライン数に応じてスコアを乗算

function calculateScore(c) {
    ②
    // ジョウホウ こうしん
    // スコア情報を更新

    document.getElementById("score-count").innerHTML
    = scoreResult;
}
```

ひょうじ

② 消した数の100倍の数をスコアに表示するには？

scoreRusult *= 100;

scoreResult=c + 100;

scoreResult++;

scoreResult=c * 100;

【こたえ】ブロック（ライン）を消した数の100倍 の数をスコアに出す関数を作ってみよう

たいじょう

対象のファイルは『stage7.js』です

かけ算は [*] で表すことができるよ

```
おう          ジョウザン（かけ算のこと）
// 消されたライン数に応じてスコアを乗算

function calculateScore(c) {
    scoreResult = c * 100;
    // スコア情報を更新

    document.getElementById("score-count").innerHTML
    = scoreResult;
}
```

ひょうじ

② 消した数の100倍の数をスコアに表示するには？

scoreRusult *= 100;

scoreResult=c + 100;

scoreResult++;

scoreResult=c * 100;

【もんだい】関数 に「消した数」を入れて、スコアを画面に描いてみよう

たいじょう

対象のファイルは『main.js』の130-135行目あたりです

```
//stage7
①
```

①作ったかんすうに「消した数を数えた数字」を入れるには？？

calculateScore(100);

calculateScore(c);

calculateScore();

calculateScore(lineCount);

【こたえ】関数 に「消した数」を入れて、 スコアを画面に描いてみよう

たいじょう

対象のファイルは『main.js』の130-135行目あたりです

```
//stage7  
calculateScore(lineCount);
```

①作ったかんすうに「消した数を数えた数字」を入れるには？？

calculateScore(100);

calculateScore(c);

calculateScore();

calculateScore(lineCount);



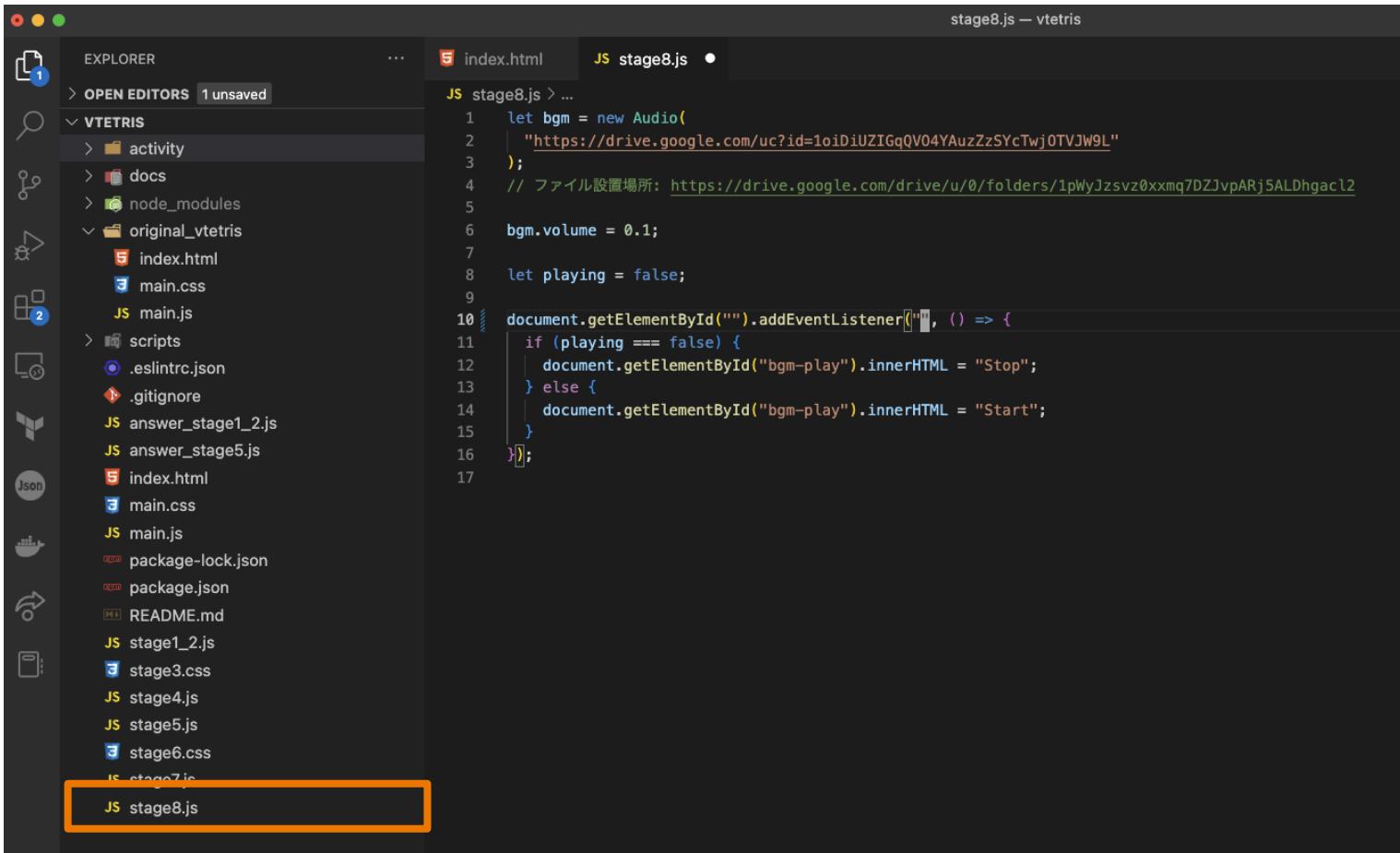
Stage 8

BGMをつけよう

stage 8 – BGMを流してみよう

Stage 8でつかうソースコード

このStageでは『stage8.js』をつかうよ！



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists several folders and files under a 'VTETRIS' root directory. The 'stage8.js' file is highlighted with an orange rectangle at the bottom of the list. The main editor area displays the contents of the 'stage8.js' file:

```
JS stage8.js > ...
1 let bgm = new Audio(
2   "https://drive.google.com/uc?id=1oiDiUZIGqQV04YAuzZzSYcTwj0TVJW9L"
3 );
4 // ファイル設置場所: https://drive.google.com/drive/u/0/folders/1pWyJzsvz0xxmq7DZJvpARj5ALDhgacL2
5
6 bgm.volume = 0.1;
7
8 let playing = false;
9
10 document.getElementById("").addEventListener(" ", () => {
11   if (playing === false) {
12     document.getElementById("bgm-play").innerHTML = "Stop";
13   } else {
14     document.getElementById("bgm-play").innerHTML = "Start";
15   }
16 });
17
```

コードを見てみよう 1

```
JS stage8.js > ...
1  let bgm = new Audio(
2  "https://drive.google.com/uc?id=1oiDiUZIGqQV04YAuzZzSYcTwj0TVJW9L"
3  );
4  // ファイル設置場所: https://drive.google.com/drive/u/0/folders/1pWyJzsvz0xxmq7DZJvpARj5ALDhgacl2
5
6  bgm.volume = 0.1;
7
8  let playing = false;
9
10 document.getElementById( Q 1 ).addEventListener( Q 2 () => {
11   if (playing === false) {
12     // playing フラグを書きかえるよ
13     Q 3
14     // 音楽を流すよ
15     Q 4
16     document.getElementById("bgm-play").innerHTML = "Stop";
17   } else {
18     // playing フラグを書きかえるよ
19     Q 3
20     // 音楽を止めるよ
21     Q 4
22     document.getElementById("bgm-play").innerHTML = "Start";
23   }
24 });
25 |
```

どうなって
いるんだろう？？



コードを見てみよう 2

```
1 let bgm = new Audio(  
2   "https://drive.google.com/uc?id=1oiDiUZIGqQV04YAuzZzSYcTwj0TVJW9L"  
3 );  
4 // ファイル設置場所: https://drive.google.com/drive/u/0/folders/1pWyJzsvz0xxmq7LZJvpA...GALDhpgr12  
5  
6 bgm.volume = 0.1;  
7
```

①ここで音楽を指定しているよ

②音のボリュームを決めているね

チャレンジ①



音楽ファイルを指定して “audio オブジェクト”を作っているよ

bgm 変数が “audio オブジェクト”ということだね

ここは後で詳しく見ていくよ

背景画像を選んだ時(Stage3)に使ったGoogle Drive から

好きな音楽を選んでみよう！

※Google DriveへはDesktopのアイコンからアクセスできるよ

チャレンジ②

bgm.volume ってなんだろう…？

bgm 変数は “audio オブジェクト”なので、 “audio オブジェクト”が
持つ色々な操作を呼び出せるよ

volume	音量の大小を設定できるよ
play	音楽を再生できるよ
pause	音楽を停止できるよ

コードを見てみよう 3



「playing」へんすうという変数しょくちを使って“フラグ”のようしょりに使うよ
初期値しょきちを **false**にして、後の処理で使えるようにしているよ

```
8 let playing = false;
9
10 document.getElementById("bgm-play").addEventListener("click", () => {
11   if (playing === false) {
12     // playing フラグを書きかえるよ
13     Q 3
14     // 音楽を流すよ
15     Q 4
16     document.getElementById("bgm-play").innerHTML = "Stop";
17   } else {
18     // playing フラグを書きかえるよ
19     Q 3
20     // 音楽を止めるよ
21     Q 4
22     document.getElementById("bgm-play").innerHTML = "Start";
23   }
24 });
25 }
```

「playing」へんすうのフラグ変数しょくちをif文で分岐ぶんきさせたり、
書きかえることで、処理を変えているよ

コードの内容を詳しく見てみよう

```
10 |   document.getElementById("Q1").addEventListener("Q2", () => {  
11 |     // ここに動作を記述  
12 |   })
```

`getElementById(" ")`

`index.html` の中からキーワードに指定したIdの"HTMLタグ"をとってくるように
指示しているよ！

キーワードは「" "」の中に書いて決めているよ。

`addEventListener(" ")`

" "の中に書いてあるイベントが起きたときに、何らかの動作を
するよう紐付けているよ！

問題を試してみよう -Q1- ①

```
10 |   document.getElementById("Q1").addEventListener("Q2", () => {  
11 |     // ここに JavaScript を書く  
12 |   })
```

Q1

getElementById(" ") の " " に入る キーワード を
「index.html」から探してみよう！



ため 問題を試してみよう -Q1- ②

```
5 index.html > ...
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <title>vTetris</title>
5       <meta charset="UTF-8" />
6       <link rel="stylesheet" href="main.css" />
7       <link rel="stylesheet" href="stage3.css" />
8       <link rel="stylesheet" href="stage6.css" />
9     </head>
10    <body>
11      <script src="stage1_2.js"></script>
12      <div class="board">
13        <canvas width="320" height="640" id="game"></canvas>
14      </div>
15      <div class="scoreResult">
16        <div class="score">Score:</div>
17        <div class="score" id="score-count">0</div>
18      </div>
19      <script src="stage4.js"></script>
20      <script src="stage5.js"></script>
21      <script src="stage7.js"></script>
22      <div class="bgmButton">
23        <bgmButton type="button" id="bgm-play">BGM</bgmButton>
24      </div>
25      <script src="stage8.js"></script>
26      <script src="main.js"></script>
27    </body>
28  </html>
```

このあたりがStage 8に
かんけい
関係ありそう。..
くわ
詳しく見てみよう！！

問題を試してみよう -Q1- ③

この“HTMLタグ”(bgmButton)を取ってくるにはどうすればいいだろう？

```
22   <div class="bgmButton">
23     <bgmButton type="button" id="bgm-play">BGM</bgmButton>
24   </div>
25   <script src="stage8.js"></script>
26   <script src="main.js"></script>
--
```

“script タグ”を使って、
外部のJavaScriptファイル(stage8.js)を読みこんでいるよ

答え -Q1-

答えは "bgm-play"

```
22  <div class="bgmButton">
23    <bgmButton type="button" id="bgm-play">BGM</bgmButton>
24  </div>
25  <script src="stage8.js"></script>
26  <script src="main.js"></script>
27  </body>
```

答え -Q1-

”bgm-play”を入れるとこうなるよ！

```
10  document.getElementById("bgm-play").addEventListener( Q 2 () => {
11    if (playing === false) {
12      // playing フラグを書きかえるよ
13      Q 3
14      // 音楽を流すよ
15      Q 4
16      document.getElementById("bgm-play").innerHTML = "Stop";
17    } else {
18      // playing フラグを書きかえるよ
19      Q 3
20      // 音楽を止めるよ
21      Q 4
22      document.getElementById("bgm-play").innerHTML = "Start";
23    }
24  });
25 }
```

ため 問題を試してみよう -Q2- ①

```
10 |   document.getElementById("bgm-play").addEventListener(Q 2 () => {  
11 |     // ここにイベント動作を記述  
12 |   })
```

Q2

addListener("xxx", ...) xxxに入るイベント動作を

いちらん

えら

そうさ

次のページの一覧から選んで操作してみよう！



問題を試してみよう -Q2- ②

一覧からイベント動作を選んで “” に入力し、試してみよう！

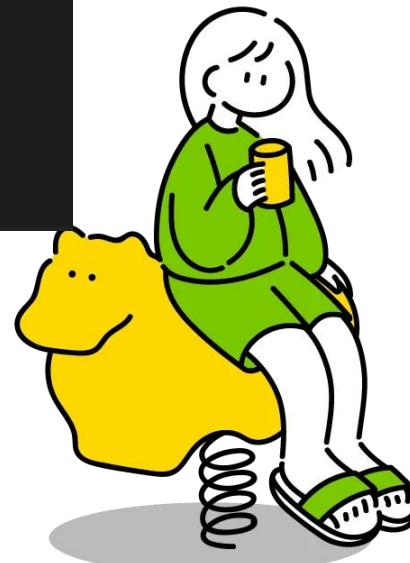
イベント名	発生するタイミング
click	左クリックしたとき
dblclick	左ダブルクリックしたとき
contextmenu	右クリックしてメニューが開くとき
mousedown	マウスのいずれかのボタンを押した瞬間
mouseup	マウスのいずれかのボタンを離した瞬間
wheel	マウスホイールが転がったとき
mouseover/mouseenter	マウスカーソルがHTML要素に乗ったとき
mouseout/mouseleave	マウスカーソルがHTML要素から離れたとき
mousemove	マウスカーソルが動いたとき

ため 問題を試してみよう -Q3- ①

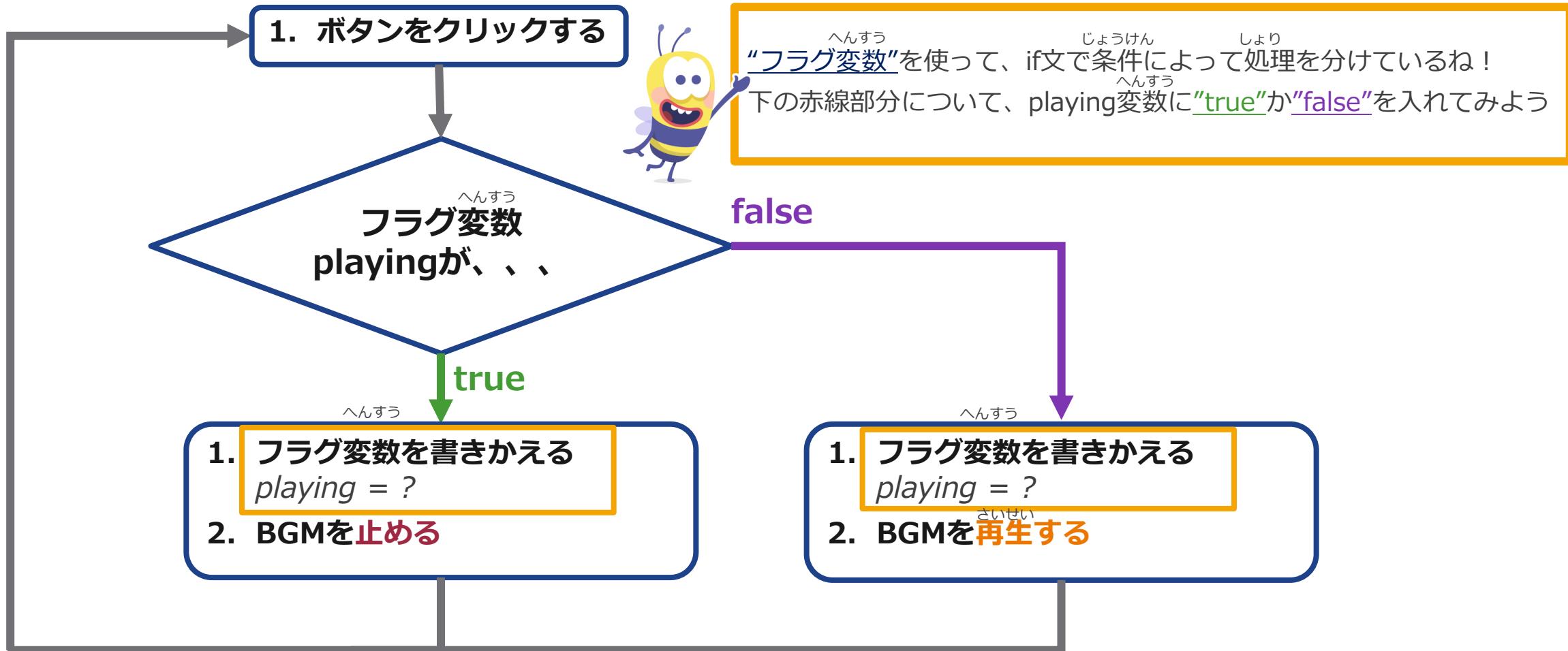
```
10  document.getElementById("bgm-play").addEventListener("click", () => {  
11    if (playing === false) {  
12      // playing フラグを書きかえるよ  
13      Q 3  
14      // 音楽を流すよ  
15      Q 4  
16      document.getElementById("bgm-play").innerHTML = "Stop";  
17    } else {  
18      // playing フラグを書きかえるよ  
19      Q 3  
20      // 音楽を止めるよ  
21      Q 4  
22      document.getElementById("bgm-play").innerHTML = "Start";  
23    }  
24  });
```

Q3

if文中のフラグ変数を書きかえてみよう
へんすう



ため 問題を試してみよう -Q3- ②

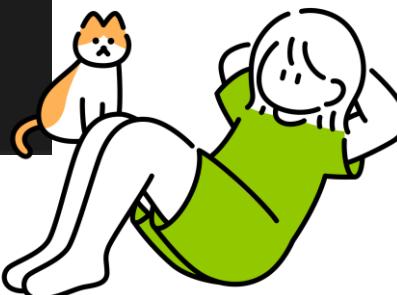


ため

問題を試してみよう -Q4- ①

```
10  document.getElementById("bgm-play").addEventListener("click", () => {
11    if (playing === false) {
12      // playing フラグを書きかえるよ
13      playing = true;
14      // 音楽を流すよ
15      Q 4
16      document.getElementById("bgm-play").innerHTML = "Stop";
17    } else {
18      // playing フラグを書きかえるよ
19      playing = false;
20      // 音楽を止めるよ
21      Q 4
22      document.getElementById("bgm-play").innerHTML = "Start";
23    }
24  });

```



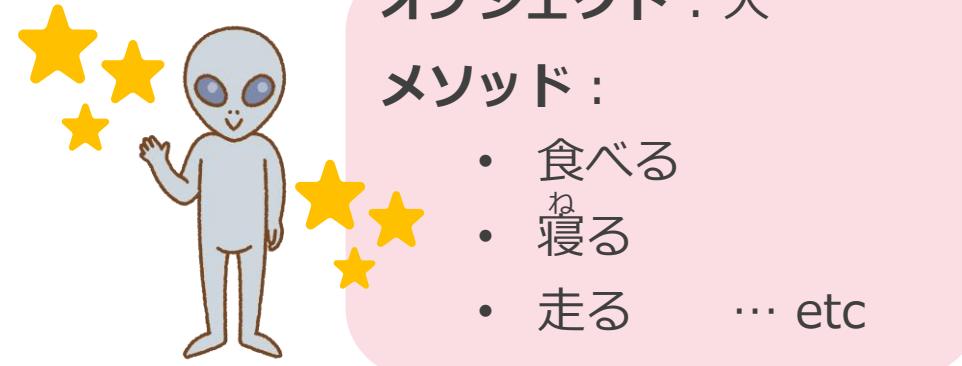
Q4

さいせい ていし
if文 中に 入る BGM の "再生/停止" の やり方 は ?

ため 問題を試してみよう -Q4- ②

```
1 let bgm = new Audio(  
2   "https://drive.google.com/uc?id=1oiDiUZIGqQV04YAuzZzSYcTwj0TVJW9L"  
3 );  
4 // ファイル設置場所: https://drive.google.com/drive/u/0/folders/1pWyJzsvz0xxmq7DZJvpARj5ALDhgac12  
5  
6 bgm.volume = 0.1;  
7
```

1行目でbgmという変数は “audio オブジェクト” になっているね
オブジェクト = 物 = データと処理(メソッド)の集まり
のことを指すよ



問題を試してみよう -Q4- ③

Audio オブジェクトにはいくつかのメソッドがあるよ

<オブジェクト変数>.<メソッド> で動作をつけることができるよ

メソッド	書き方例	動作
volume	bgm.volume = 0.1	音量の大小を設定できるよ(例はvolume 0.1の大きさ)
play	bgm.play()	音楽を再生できるよ
pause	bgm.pause()	音楽を停止できるよ
currentTime	bgm.currentTime = 0	音楽の再生位置を設定できるよ(例は初期位置に戻す)

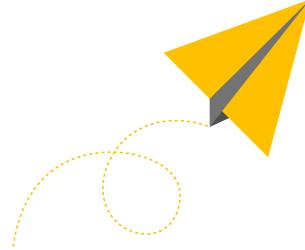
さあ、コードに書いてみよう！



Stage 9

オリジナルを作ろう

きのう
stage 9 – すきな機能を考えてみよう



アイディアを出してみよう！

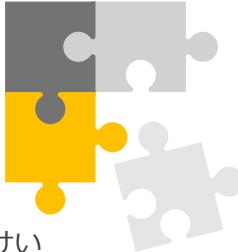
思いついたら書き出してみよう！



たとえば...

- ・最高スコアを表示しておきたい ...など





ひつよう そのためには何が必要かな？

かんけい
関係のありそうなステージはあったかな？



たとえば...

- “スコア”って Stage7 で出てきたな ...など





調べながら
じっさい
実際に作ってみよう！

Thank You