

# SEGA CHALLENGE!

## ぷよぷよプログラミング

### ◆◆JavaScript & HTML5◆◆

～「ぷよぷよ」をコーディングしよう！～

Enjoy to Code

■必要なもの

- ・パソコン(Windows、Mac、Chromebook、iOSまたはAndroidタブレット)
- ・インターネット接続環境
- ・メールアドレス
- ・ブラウザ(Google Chrome推奨)
- ・気合

■対象

- ・小学校4年生以上

■想定時間

- ・初級コース：約1時間
  - ・中級コース：約4時間
- ※一般大人の場合

■URL

[http://puyo.sega.jp/program\\_2020/](http://puyo.sega.jp/program_2020/)



●Monaca Education

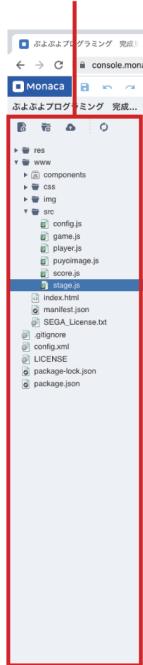
国内の中学校、高等学校、専門学校、大学など800以上の教育機関におけるプログラミング教育で利用されているプログラミング学習環境です。特別なソフトウェアのインストールは不要でブラウザを用いてクラウド環境でプログラミング学習を進めることができます。

※ご利用には、Monaca Educationへの会員登録が必要です。 利用料：無料（機能制限あり） ©Asial Corporation

# Monaca開発画面の操作方法

左カラム(段組み)

ファイル構成



拡大・縮小可能

中央カラム(段組み)

入力画面

```
class Stage {  
    static stageElement;  
    static scoreElement;  
    static zenkishiImage;  
    static board;  
    static puyoCount;  
    static fallingPuyoList = [];  
    static eraseStartFrame;  
    static erasingPuyoInfoList = [];  
  
    static initialize() {  
        // HTML からステージの元となる要素を取得し、大きさを設定する  
        const stageElement = document.getElementById("stage");  
        stageElement.style.width = Config.puyoImgWidth * Config.stageCols + 'px';  
        stageElement.style.height = Config.puyoImgHeight * Config.stageRows + 'px';  
        stageElement.style.backgroundColor = Config.stageBackgroundColor;  
        this.stageElement = stageElement;  
    }  
}
```

右カラム(段組み)

開発中成果物

拡大・縮小可能



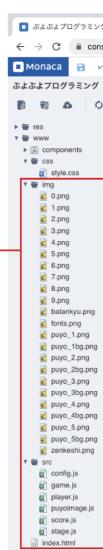
拡大・縮小可能

「レイアウトをリセット」で  
画面レイアウトを元に戻せます。

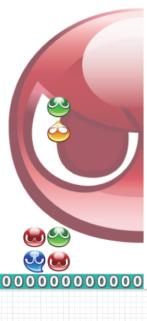
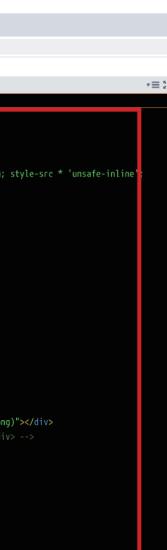
この画面で入力します。  
画面内で右クリックをすると、  
フォントサイズを拡大できます。

開発成果物をスマートフォン  
にアウトプットできます。  
実際の機器の画面サイズで  
プレビューが可能です。

▶でフォルダを開きます。  
赤で囲まれた内容が  
「ふよふよプログラミング」  
のソースコード構成です。  
img : 画像一式  
src : JSコード一式  
Index.html : Html一式  
SEGA\_Lisence : 規約  
※必要なファイルを  
ダブルクリックすると  
中央カラムに展開されます。



https://console.monaca.education/#



# 「ぷよぷよ」が落ちてくる

「ぷよ」が落ちてこなくて困っている。  
君のコーディングで「ぷよ」の雨を降らせてくれ！

## まずはじめに

「ぷよぷよプログラミング」は書き写しをすることでプログラミングに慣れていくだけの教材です。JavaScriptの入力ルールはいくつもありますが、ここでは難しい事は考えないようにしましょう。

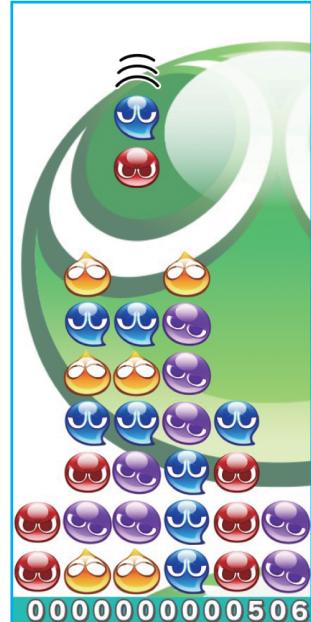
SESSION1から順番に、次頁のサンプルコードをそのまま、正確にコーディング(入力)するだけ。まちがえたら直せば良いので、気持ちを楽に進めましょう。

### ■入力文字(英数字半角、大文字小文字)

ソースコードに日本語(半角、全角)は使わず、英数字半角を使います。また、大文字小文字が区別されますので気をつけましょう。

### ■インデント(字下げ) ※ソースコードを読みやすくする

プログラミングでは、ソースコードを見てわかりやすいようにインデント(字下げ)することが一般的です。このインデントがないと小さなミスにつながり、どこがまちがっているのかわからなくなることもあるので気をつけましょう。



**インデントと空白の入れかた** ※ Tabキーを1回押すごとに、半角4文字分空白移動します。

189                            this.puyoStatus.top+=Config.playerFallingSpeed;  
 英数字半角12文字空け      ピリオド      半角1文字空け      セミコロン

190                            if(isDownPressed){  
 英数字半角12文字空け      左小括弧      右小括弧      左中括弧

192                            this.puyoStatus.top+=Config.playerDownSpeed;  
 英数字半角16文字空け

193                            }  
 英数字半角16文字空け      右中括弧

## キーボード記号の読みかたを覚えよう

Tab	タブ	.	ピリオドまたはドット	+	プラス	=	イコール	;	セミコロン
-----	----	---	------------	---	-----	---	------	---	-------

(	左小括弧	)	右小括弧	{	左中括弧	}	右中括弧
---	------	---	------	---	------	---	------

※「player.js」を選択してください。

### 入力が必要なコード

■初級 全4行 189、190、192、193

■中級 全23行 189、190、192、193、197~205、207~209、211~215、217、218

### 行番号

```

189         this.puyoStatus.top += Config.playerFallingSpeed;
190         if(isDownPressed) {
191             // 下キーが押されているならもっと加速する
192             this.puyoStatus.top += Config.playerDownSpeed;
193         }
194         if(Math.floor(this.puyoStatus.top / Config.puyoImgHeight) != y) {
195             // ブロックの境を超えたので、再チェックする
196             // 下キーが押されていたら、得点を加算する
197             if(isDownPressed) {
198                 Score.addScore(1);
199             }
200             y += 1;
201             this.puyoStatus.y = y;
202             if(y + 1 >= Config.stageRows || Stage.board[y + 1][x] || (y + dy + 1 >= 0 && (y + dy + 1 >= Config.
203             stageRows || Stage.board[y + dy + 1][x + dx]))) {
204                 isBlocked = true;
205             }
206             if(!isBlocked) {
207                 // 境を超えたが特に問題はなかった。次回も自由落下を続ける
208                 this.groundFrame = 0;
209             } else {
210                 // 境を超したらブロックにぶつかった。位置を調節して、接地を開始する
211                 this.puyoStatus.top = y * Config.puyoImgHeight;
212                 this.groundFrame = 1;
213             }
214         }
215     } else {
216         // 自由落下で特に問題がなかった。次回も自由落下を続ける
217         this.groundFrame = 0;
218     }

```



入力が済んだら **CTRL** キー + **s** キーで保存しましょう。

コーディング中にソースコードをまちがってうまくいかなくなったら、作業中のプロジェクトを削除して最初からやり直しましょう。プログラミングはチャレンジの連続です。解決方法は必ず見つかりますよ。

# 「ぷよ」を左右に動かす

今度は「ぷよ」が動かなくなってしまったぞ。  
左右のキーで「ぷよ」が移動できるようにするんだ！

文

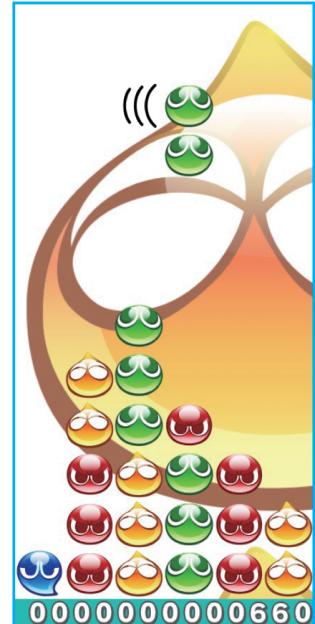
JavaScriptは、文（と式）で構成されています。文の末尾にセミコロン(;)を置くことで文ごとに区切りをつけます。ソースコードとして書かれた文をコンピューターは上から処理していくことで、プログラムが実行されます。

## ■ 空白、改行、括弧、区切り

空白や改行、括弧の位置の有無は、実はJavaScriptの挙動には影響ありません。上記のセミコロンも、入力しなくても動作はしますが、ソースコードを読みやすく、小さなミスを無くすために多くの人が使用しています。

### ■ 1行コメント(メモ書き)

行頭に / (スラッシュ) を 2 つ付けるだけで、その行はソースコードとして無視されます。メモ書きや注意書きとして、後で見たときにわかるようなことを記述します。「ぶよぶよプログラミング」でもたくさん 1 行コメントが書かれていますので確認してみましょう。



## インデントと空白の入れかた

278   this.actionStartFrame=frame;  
英数字半角16文字空け

279 □□□□□□□□□□□□□this.moveSource = x \* Config.puyoImgWidth;  
英数字半角16文字空け アスターイスク

280 □□□□□□□□□□□□□□this.moveDestination=(x+cx)\*Config.puyoImgWidth;

281                         this.puyoStatus.x+=cx;

282 □□□□□□□□□□□□□return□'moving';  
英数字半角16文字空け アポストロフ /

## キーボード記号の読みかたを覚えよう

## シングルクォーテーション

## \* アスタリスク

※「player.js」を選択してください。

## 入力が必要なコード

■初級 全5行 278~282

■中級 全28行 252~261、263~275、278~282

### 行番号

```

252         if(y < 0 || x + cx < 0 || x + cx >= Config.stageCols || Stage.board[y][x + cx]) {
253             if(y >= 0) {
254                 canMove = false;
255             }
256         }
257         if(my < 0 || mx + cx < 0 || mx + cx >= Config.stageCols || Stage.board[my][mx + cx]) {
258             if(my >= 0) {
259                 canMove = false;
260             }
261         }
262         // 接地していない場合は、さらに1個下のブロックの左右も確認する
263         if(this.groundFrame === 0) {
264             if(y + 1 < 0 || x + cx < 0 || x + cx >= Config.stageCols || Stage.board[y + 1][x + cx]) {
265                 if(y + 1 >= 0) {
266                     canMove = false;
267                 }
268             }
269             if(my + 1 < 0 || mx + cx < 0 || mx + cx >= Config.stageCols || Stage.board[my + 1][mx + cx]) {
270                 if(my + 1 >= 0) {
271                     canMove = false;
272                 }
273             }
274         }
275     }
276     if(canMove) {
277         // 動かすことが出来るので、移動先情報をセットして移動状態にする
278         this.actionStartFrame = frame;
279         this.moveSource = x * Config.puyoImgWidth;
280         this.moveDestination = (x + cx) * Config.puyoImgWidth;
281         this.puyoStatus.x += cx;
282         return 'moving';

```



コーディング中にソースコードをまちがってうまくいかなくなったら、作業中のプロジェクトを削除して最初からやり直しましょう。プログラミングはチャレンジの連続です。解決方法は必ず見つかりますよ。

入力が済んだら **CTRL** キー + **S** キーで保存しましょう。

# 「ぷよ」を回してみよう

「ふよ」が回転しないと、ばたんきゅーだ。

「ふよ」が上キー↑で回るようにしてみよう。

## ■ファイル名 「.html」「.js」

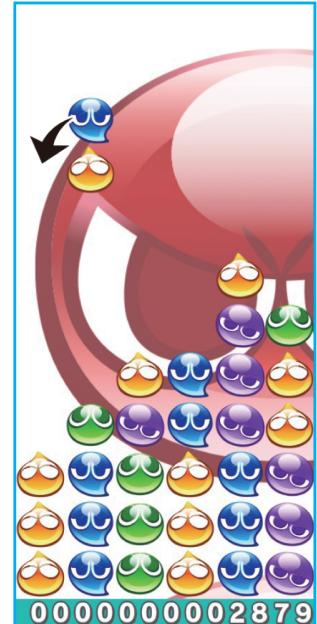
「ぷよぷよプログラミング」は、コンピュータ言語のHTML5とJavaScriptで記述されています。HTMLはWebページを作るための最も基本的なマークアップ言語。JavaScriptは、Webページで複雑な動作を行えるようにするプログラミング言語です。それ故れ、○○.html、○○.jsというファイル名になっています。

「ふよふよプログラミング」は、srcフォルダの中に6つのJavaScriptファイルとindex.html、そして画像(imgフォルダ)で構成されています。

## ■htmlの働き

「ぷよぷよプログラミング」の場合、index.htmlの中で下記のようにコードを書いてJavaScriptファイルを使います。

```
<script src="src/config.js"></script>
<script src="src/stage.js"></script>
<script src="src/score.js"></script>
<script src="src/puyoimage.js"></script>
<script src="src/player.js"></script>
<script src="src/game.js"></script>
```



## インデントと空白の入れかた

コツをつかみましたね？ 360~365行、368~371行の説明は省きます。

366 constdistRotation=(this.puyoStatus.rotation+u90)%360;  
英数字半角16文字空け パーセント

367 □□□□□□□□□□□□const dCombi[] = [[1, 0], [0, -1], [-1, 0], [0, 1]][distRotation]/90];  
英数字半角16文字空け  
スラッシュ

# キーボード記号の読みかたを覚えよう

% パーセント

## スラッシュ

※「player.js」を選択してください。

### 入力が必要なコード

- 初級 全10行 360~363、365~370
- 中級 全18行 351~358、360~363、365~370

行番号

```

351         if(cy === -1) {
352             if(this.groundFrame > 0) {
353                 // 接地しているなら1段引き上げる
354                 this.puyoStatus.y -= 1;
355                 this.groundFrame = 0;
356             }
357             this.puyoStatus.top = this.puyoStatus.y * Config.puyoImgHeight;
358         }
359         // 回すことが出来るので、回転後の情報をセットして回転状態にする
360         this.actionStartFrame = frame;
361         this.rotateBeforeLeft = x * Config.puyoImgHeight;
362         this.rotateAfterLeft = (x + cx) * Config.puyoImgHeight;
363         this.rotateFromRotation = this.puyoStatus.rotation;
364         // 次の状態を先に設定しておく
365         this.puyoStatus.x += cx;
366         const distRotation = (this.puyoStatus.rotation + 90) % 360;
367         const dCombi = [[1, 0], [0, -1], [-1, 0], [0, 1]][distRotation / 90];
368         this.puyoStatus.dx = dCombi[0];
369         this.puyoStatus.dy = dCombi[1];
370         return 'rotating';

```



コーディング中にソースコードをまちがってうまくいかなくなったら、作業中のプロジェクトを削除して最初からやり直しましょう。プログラミングはチャレンジの連続です。解決方法は必ず見つかりますよ。  
入力が済んだら **CTRL** キー + **S** キーで保存しましょう。

# 「ぷよ」を消してみよう

「ぷよ」がたくさんつながったけど消えないぞ。  
同色が4つ以上つながると消えるようにしてみよう！

## ■Windowsショートカット

キーボードのショートカットは便利でカッコいいので、ぜひこの2つは覚えましょう。

- ・**[ctrl]キー+Sキー** 保存するときに使います。
- ・**[ctrl]キー+Zキー** 1つ前のアクションに戻ります。



## インデントと空白の入れかた

ここでも、175~177行、180~184行の説明は省きます。

```
178  _____if(!cell||cell.puyo!=puyo){  
英数字半角16文字空け エクスクラメーションマーク 縦線
```

## キーボード記号の読みかたを覚えよう

<b>!</b>	エクスクラメーションマーク	<b> </b>	縦線
----------	---------------	----------	----



※「stage.js」を選択してください。

**入力が必要なコード****■初級 全9行** 173、175～178、180、181、183、184**■中級 全26行** 154、155、157、158、160～167、169～173、175～178、180、181、183～185

## 行番号

```

154     const orig = this.board[y][x];
155     if(!orig) {
156         // ないなら何もしない
157         return;
158     }
159     // あるなら一旦退避して、メモリ上から消す
160     const puyo = this.board[y][x].puyo;
161     sequencePuyoInfoList.push({
162         x: x,
163         y: y,
164         cell: this.board[y][x]
165     });
166     this.board[y][x] = null;
167
168     // 四方向の周囲ぷよを確認する
169     const direction = [[0, 1], [1, 0], [0, -1], [-1, 0]];
170     for(let i = 0; i < direction.length; i++) {
171         const dx = x + direction[i][0];
172         const dy = y + direction[i][1];
173         if(dx < 0 || dy < 0 || dx >= Config.stageCols || dy >= Config.stageRows) {
174             // ステージの外にはみ出た
175             continue;
176         }
177         const cell = this.board[dy][dx];
178         if(!cell || cell.puyo !== puyo) {
179             // ぷよの色が違う
180             continue;
181         }
182         // そのぷよのまわりのぷよも消せるか確認する
183         checkSequentialPuyo(dx, dy);
184
185     };

```



コーディング中にソースコードをまちがってうまくいかなくなったら、作業中のプロジェクトを削除して最初からやり直しましょう。プログラミングはチャレンジの連続です。解決方法は必ず見つかりますよ。

入力が済んだら **CTRL** キー + **S** キーで保存しましょう。

# 「ぷよ」を変えてみよう

ソースコードを書き換えると、「ぷよぷよ」の動きを確かめてみよう。

## ① 背景の「ぷよ」を入れ変えてみよう

以下の「img/puyo\_1bg.png」の部分を書き換えると、背景が入れ変わります。

### index.html

```
22 <div id="stage" style="position: relative; margin: 0 auto; overflow:  
hidden; background: url(img/puyo_1bg.png)"></div>  
  
[img/puyo_1bg.png] → 緑ぷよ [img/puyo_4bg.png] → 赤ぷよ  
[img/puyo_2bg.png] → 青ぷよ [img/puyo_5bg.png] → 黄ぷよ  
[img/puyo_3bg.png] → 紫ぷよ
```

## ② ステージの大きさを変えてみよう

横6列、縦12段が通常のステージの大きさです。数字を変更すると「ぷよ」の数が変化します。

### config.js

```
9 Config.stageCols = 6; // ステージの横の個数  
10 Config.stageRows = 12; // ステージの縦の個数
```

## ③ 背景の色を変えてみよう

以下の数字、「HTMLカラーコード」を変更すると、背景の色が変わります。

**HTMLカラーコード** で検索して、変えたい色の数値を調べてみましょう。

### config.js

```
17 Config.stageBackgroundColor = '#11213b'; // ステージの背景色  
例 白色 : white [#000000] 黄色 : yellow [#ffff00]
```

**④ 「ぷよ」の色数を変えてみよう**

以下の数字を「4」→「3」に変えると、落ちてくる「ぷよ」が3色のみになります。

※色数は1~5まで設定することができます。

**config.js**

```
24 Config.puyoColors = 4; // 何色のぷよを使うか
```

**⑤ 「ぷよ」の落下スピードを変えてみよう**

以下の数字を「0.9」→「10.0」に変えると、すごいスピードで「ぷよ」が落ちてきます。

**config.js**

```
25 Config.playerFallingSpeed = 0.9; // プレイ中の自然落下のスピード
```

**⑥ 落ちてくる「ぷよ」の色を毎回変える**

「puyoimage.js」の16~17行の間に以下のコードを追加すると、出てくる「ぷよ」の色が毎回変わります。

**puyoimage.js**

```
15         this.puyoImages[i] = image;
16     }
    // 色をシャッフル
    for (let i = this.puyoImages.length - 1; i >= 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [this.puyoImages[i], this.puyoImages[j]] = [this.
            puyoImages[j], this.puyoImages[i]];
17    }
```

# 上級コースに挑戦しよう!

最後はゼロから「ぷよぷよ」を組み立ててみよう。

※入力量：1,015行

上級コースはフォルダだけ用意され、ソースコードが何も書かれていません。時間がかかりますが、すべて自分でコーディングすることで、「ぷよぷよ」を作っていく手順を知ることができます。

## ①まず最初に「index.html」を書く

ここに必要なファイル名や約束的な処理を書いていきます。他のゲームを開発するときでも、毎回同じような内容になっています。

## ②次に「game.js」を書く

ここではゲームの大きな流れや、処理を行う順番などの外枠を書いていきます。(最初に画像ファイルを準備する、新しいぷよを落とし始める、その次はぷよが落ちる、ぷよが消えるなど)

## ③続いて「puyoimage.js」を書く

用意した画像を表示できるようにする処理を行います。

## ④その次に「player.js」を書く

キー入力でぷよぷよを動かす処理を書きます。

## ⑤そして「stage.js」を書く

画像の用意とそれを動かす準備ができたので、キー入力で動かせるようにします。ここまでで基本的な流れを作りました。

## ⑥いよいよ全体的な肉づけを開始

ここから先は、「game.js」でゲームのルール通りにプログラムを肉づけしていきます。

途中で「出現するぷよの色数や落下スピードを変更できるようにしよう」と思ったら、「config.js」に必要なパラメーターを設定しておくなど、その都度、他の.jsファイルにも必要な処理を追加しましょう。

## ⑦デバック

ひと通りプレイできるようになったら、さらにゲームがおもしろくなる要素を追加しても良いでしょう(「ぷよぷよプログラミング」では、スコアが表示される「score.js」を追加しました)。

プログラミングが上手な人は、ソースコード全体をきれいにコーディングできて、最終的にバグが少ない、もしくはバグがあってもすぐに修正できるソースコードを書きあげます。ソースコードがぐちゃぐちゃになり、バグが多くて直しきれないプログラムにならないよう気をつけましょう。

## ⑧おわりに

プログラミングにおける写経＝書き写しには以下のメリットがあります。

- ・パソコンの操作を覚える
- ・コーディング作法を知り、プログラミングに慣れる
- ・小さなミスを自分で直す能力が身につく
- ・完成させたという自信がつく

まずはプログラミングってどういうものだろう？ ということを「ぷよぷよプログラミング」を通じて学んでいただければうれしいです。

