



# Girls Who Code At Home

流星キャッチャーゲーム **Part1**  
プランニング + p5.js 入門

## 学習概要

このプロジェクトでは、アーティストやデザイナーのために作られた JavaScript ライブラリである **p5.js** を使って、収集ゲームをプログラミングすることを学びます。このプロジェクトの **Part1** では、ゲームの異なるパーツがシステムとしてどのように連携しているかについて学び、p5.js の基本を探ります。関数を使ってスケッチを作成し、プログラムフロー（プログラム内でコードがどのように実行されるか）について詳しく学びます。



## 学習目標

このアクティビティが終わるころには、次のことができるようになるでしょう。

- ❑ ゲームの部品を特定し、それらがプログラム内のシステムとしてどのように機能するかを説明する
- ❑ p5.js の背後にあるインスピレーションを説明し、環境をナビゲートする。
- ❑ 条件式や制御フローなど、関連する用語を用いてプログラムの流れを記述する。

## マテリアル

- [p5.js オンラインエディタ](#)
- [流星キャッチャーゲーム サンプルプロジェクト](#)
- [流星キャッチャーゲーム その1 リファレンスガイド](#)

## 予備知識

このプロジェクトに着手する前に、以下のことができることをお勧めします。

- ❑ [変数](#) とは何かを自分の言葉で説明し、プログラム中でどのように使われるかを説明できる。
- ❑ [条件](#) とは何かを自分の言葉で説明し、プログラム中でどのように使われるかを説明できる。

## Women in Tech スポットライト: キャシー・タラカジーン

Cassie Tarakajian は、ソフトウェア開発者、ハードウェアエンジニア、クリエイティブテクノロジー、ミュージシャン、そして教育者です。また、キャシーは [ノンバイナリー](#) であり、代名詞として they/them を使用しています。キャシーが初めてコードを学んだのは大学で [Java](#) 入門の授業を受けたと



画像出典: [NYU Tisch](#)

きです。非常にシンプルなテキストエディタを使って、非常にシンプルなテキストベースのタスクをこなすことでしか、学習しなかったと記憶しています。その後、キャシーは、Processing プラットフォームをベースにしたインタラクティブなアートやサウンドを作るための JavaScript ライブラリ、[p5.js](#) というオープンソースプロジェクトへの貢献を依頼されました。ウェブエディタ p5.js の開発者兼リードメンテナとして、ブラウザ上でコードを書き、実行できる環境を開発し、初心者でも簡単に使えるようにしました。ブラウザ上でコードを書き、実行できる環境を開発し、初心者の方も簡単に利用できるようにしました。このプラットフォームは、スクリーンリーダーやハイコントラスト表示にも対応しており、視覚に障がいのある方にも安心してご利用いただけます。

キャシーは p5.js の仕事に加え、[Cycling '74](#) のエンジニア、[Girlfriends Lab](#) の共同設立者、そして [Tisch School of the Arts](#) の非常勤教授でもあるのだそうです。キャシーは、プログラマー像が一つの物語だけに限定されないことを証明するために、これらすべての役割を日々体現し続けています。

この[ビデオ](#) (6:00 まで) では、キャシーと p5.js を使った彼らの仕事について詳しく説明しています。Cassie とキャシーの仕事についてもっと知りたいですか？彼らの[個人的なウェブサイト](#)をご覧ください。p5.js に関する彼らの仕事と、彼らがどのように始めたかについての[記事](#)を読んでみてください。

## 考えてみましょう

コンピュータサイエンティストであることは、単にコーディングが得意というだけではありません。キャシーと彼らの仕事が、優れたコンピュータ科学者が重視している勇氣、レジリエンス、創造力、そして目的という強みにどのように関係しているか、時間をかけて考えてみてください。



URPOSE

p5.js を作る際、キャシーが重要視したのは、コードを簡単に学べる環境を作ることと、さまざまなニーズに対応できるようにすることでした。p5.js におけるアクセシビリティの大きな要素は、スクリーンリーダーとの互換性とカスタマイズ可能な設定の提供です。「すべての人のための」ウェブエディタを作ることが、なぜキャシーにとって重要だったのでしょうか？

あなたの回答を家族や友人と共有しましょう。他の人にもキャシーについてもっと読むように勧めて、議論に参加してもらいましょう！

## ステップ 1：流星キャッチャーゲームの探索 (10～15 分)

### ゲームのパーツを紹介する (2 分)

すべての (あるいは多くの) ゲームには、ゴール、チャレンジ、コアメカニクス、コンポーネント、ルール、スペースの 6 つのパーツがあります。これらの部品は、関係する人々の間で遊びを生み出すシステムとして一緒に機能します。ゲームデザイナーとして、ゲームをプログラムするためには、すべてのパーツがシステムとしてどのように連携しているかを理解することが不可欠です。



- **ゴール**：ゲームに勝つために、選手やチームは何をしなければならないか？
- **チャレンジ**：プレイヤーがゴールに到達するまでに、どのような障害があるのか？
- **コアメカニクス**：ゲームのプレイを推進するために、プレイヤーはどのようなアクションや動作を行うのか？
- **コンポーネント(構成要素)**：遊びの材料を構成する要素は何か？
- **ルール**：ゲーム内でプレイヤーができること、できないことは？
- **場所**：ゲームの舞台となるのはどこか？

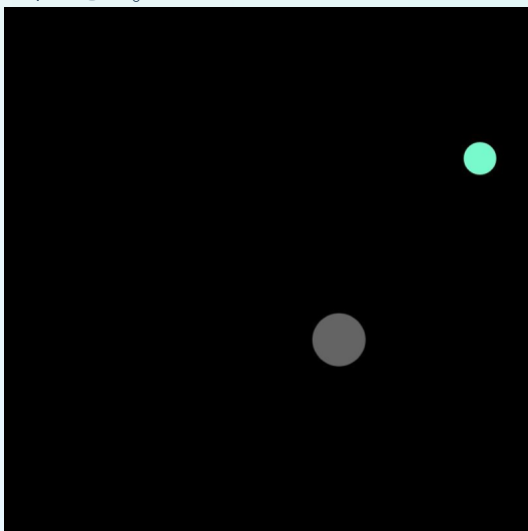
例

### OX ゲーム

- **ゴール**：3 連勝すること
- **チャレンジ**：相手がどこにシンボルを置くかわからない
- **コアメカニクス**：ブロックを置くこと、書くこと
- **構成要素**：筆記用具、紙、プレイヤー、X と O。
- **ルール**：プレイヤーは 2 人。各プレイヤーは交互に記号を書き、マスが一杯になるか、片方のプレイヤーが縦、横、斜めに 3 つ並べるまで書き続けます。
- **場所**：紙上に 3×3 のマス目

### ゲームをする (1 分)

ゲームデザイナーが腕を磨くには、ゲームをするのが一番!これから作るゲームを試してみましょう。この[リンク](#)をクリックすると、約 30 秒間ゲームをプレイしてみましょう。遊びながら、このゲームの部品を確認してみてください。

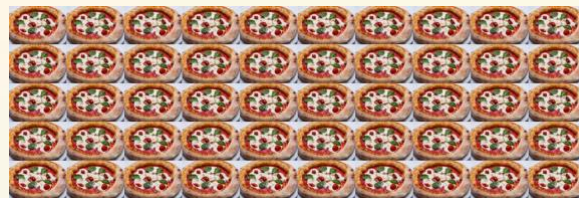


新しいプログラミング技術を学ぶために流星をつかまえるゲームを企画・制作しますが、「拡張機能」のセクションでは、それをカスタマイズすることもできます。

## ゲームのパーツを特定する (5-10 分)

このステップの目標は、大きな問題を小さなパーツに分解することで、「流星キャッチャー」のシステムがどのように機能しているかを理解することです。このプロセスを**分解**と呼びます。

例えば、あなたのタスクが子供たちのために 100 枚のピザを作ることだとしましょう。大変な仕事ですが、これを小さなステップやサブ問題に分解すれば、もっと管理しやすくなります。例えば、すべての生地を最初に作り、すべての材料を調理し、一度に 5 枚ずつ組み立て、そしてまとめて調理します。



複雑な問題をより単純なパーツに分解する方法はたくさんあります。私たちはすでにゲームを持っているので、**リバースエンジニアリング**のアプローチを取ります。つまり、ゲームをゼロから作るのではなく、完成した製品を分解して、どのように作られたかを理解するのです。まず、「流星キャッチャー」のすべてのパーツを定義し、それを使って擬似コードを書きます。下の空欄に、流星キャッチャーゲームのパーツを分解してみましょう。その際、**リファレンスガイド**で自分の考えを確認することを忘れないでください。

### 流星キャッチャーゲームのパーツ

→ 先ほどのステップで遊んだばかりのゲーム、流星キャッチャーゲームの**ゴール**を説明してください。ゲームに勝つためには、プレイヤーやチームは何をしなければならないのでしょうか？

→ 流星キャッチャーの**構成要素**には、流星、キャッチャー、壁、プレイヤーがあります。各要素には特有の性質（大きさ、色、形など）と動作（その構成要素から連想される動詞）があり、ゲームのシステムに寄与しています。例えば、流星の性質は丸で、動作は画面の上から下へ落ちるというものです。

下の表の各構成要素の性質と動作について、**2～3 分**考えてみてください。

構成要素	性質	動作
ゲームのプレイに欠かせないものは？	構成要素の属性や特徴は？(例：サイズ、色、形など)	それは何をするもの？ どんな動詞を連想する？

--	--	--

→ ゲームの**舞台**について説明してください。どこで行われますか？(空間が複数のものになることもあることに注意してください。例えば、チェスはチェス盤の上で行われますが、リビングルーム、公園、カフェテリアなどでも行われます)

→ **チャレンジ**を設定してください。プレイヤーがゴールにたどり着くまでに、どんな難関がありますか？

落ちてくる流星をキャッチすることがチャレンジです。流星はそれぞれ異なる位置から、異なる速度で落下し、新しい流星は画面に現れるたびに異なるサイズになります。

→ ゲームの**コアメカニク**を説明してください。ゲームをプレイするために、プレイヤーはどのような中心となる行動や動きをする必要がありますか？ゲームをプレイするために、プレイヤーはどのような中心となる行動や動きをしますか？

→ ゲームの**ルール**のリストを書きます。ルールは、ゲームで何ができて、何ができないかを決めるものです。プレイヤー、構成要素、舞台などに適用することができます。



リファレンスガイドの3ページで自分のアイデアを確認することも忘れないでください

## ステップ2：ゲームの疑似コードを書く（5～10分）



このステップでは、プログラムの指示を紙やコンピュータに高レベルで書きだしてみます。これは**疑似コード**と呼ばれます。疑似コードは、あなたのコードが何を行うかを平易に記述したものです。これは、プログラムのフローとロジックを考え、プログラムを書くために必要なステップを決定するのに役立ちます。疑似コードは、特定のコード構文を使用しませんが、一見コードのように見えます。例えば、すべてのプログラミング言語に適用される if や他のコアキーワードを使用することができます。常に疑似コードから始めましょう。

すでに、いくつかの事項を記入しましたので、参考にしてください。また、上述のゲームのパーツを使って、何を入れるべきかを決めるとよいでしょう。もし行き詰まったら、ゲームについて自分自身に問いかけてみてください。例えば、

→ ゲーム開始時に必要なことは？

→ ゲーム中に必要なことは？

具体的に書くように心がけましょう。しかし、すべてを把握できなくても、どうすればいいかわからないことを書いても、気にしないでください。疑似コードの書き方は人それぞれなのです。この疑似コードはプロジェクト中に再び登場しますので、必ず保存しておいてください。

```
// Starter 疑似コード  
任意の変数を宣言します。  
  
一度だけ実行する  
  キャンバスの大きさを 400 ピクセル×400 ピクセルに設定する  
  
これをループ毎に行う  
  背景色を設定する  
  
// なぜこれが draw() vs setup() になるかは後ほど説明します!  
  
流星を描く
```

流星が描かれた後、ゲームではどうなるのでしょうか？ 思い出せない場合は、ゲームを再プレイしてみてください。

**ヒント：**構成要素とルールはとても役に立ちます！



リファレンスガイドの 4 ページで自分のアイデアを確認することも忘れないでください。

## ステップ 2：p5.js と出会う！（10-15 分）

P5.js（または単に p5）は、ウェブブラウザ用のインタラクティブアートを作成することができます。これは、クリエイティブなコーディング、つまり、機能性だけでなく、表現力のためにコードを使用するプロジェクトのためのツールなのです。P5.js は、Web 上にインタラクティブ性を付加することができるプログラミング



言語である JavaScript のライブラリです。ライブラリであるということは、p5 は JavaScript であるということですが、クリエイターが特化した関数/メソッドのコレクション（ライブラリ）を作ったので、すべてをゼロから行う必要はありません。Web ベースなので、作ったものを簡単に共有することができますよ。[p5 のホームページ](#)では、その成り立ちやコミュニティについて詳しく紹介しています。[ショーケースのページ](#)を訪れて p5 を使用して作成されたプロジェクトの例をご覧ください。



p5.js の p は Processing の略です。Processing は、アーティストやデザイナーが自分のプロジェクトにコードを統合するために作られたプログラミング言語です。Processing は、アニメーションからデータビジュアライゼーション、楽器、ゲーム、大規模なインスタレーションまで、様々なインタラクティブメディアを初心者でも簡単に作成できるように設計されています。[Processing Foundation のホームページ](#)で、より詳しい情報をご覧ください。

### アカウントの作成（3～5 分）



p5.js を使うには、オンラインのウェブエディタかテキストエディタ、ローカルコンピュータにダウンロードした p5.js のコピーを使用する 2 つの方法があります。p5 を使い始める最も簡単な方法は、オンライン・エディタです。これは、Web ブラウザでコードを書き、プログラムを実行することができます。このチュートリアルでは、ステップを参照し、例を説明するためにのみ、ウェブエディタを使用する予定です。

ウェブエディタを使い始めるには、アカウントを作成する必要があります。

□ <https://editor.p5js.org/signup> にアクセスしてください。

□ **サインアップしてください。**すべてのフィールド（ユーザーネーム、メールアドレス、パスワード、パスワードの確認）を埋めてから "Sign up" をクリックするか、Google または GitHub でサインインすることを選択します。

□ **電子メールを認証してください。**アカウントの確認と認証のためのメールが届きます（3-5 分以内に表示されない場合は迷惑メールを確認してください）。リンクをクリックし、新しい認証情報（ユーザー名とパスワード）を使ってサインインしてください。

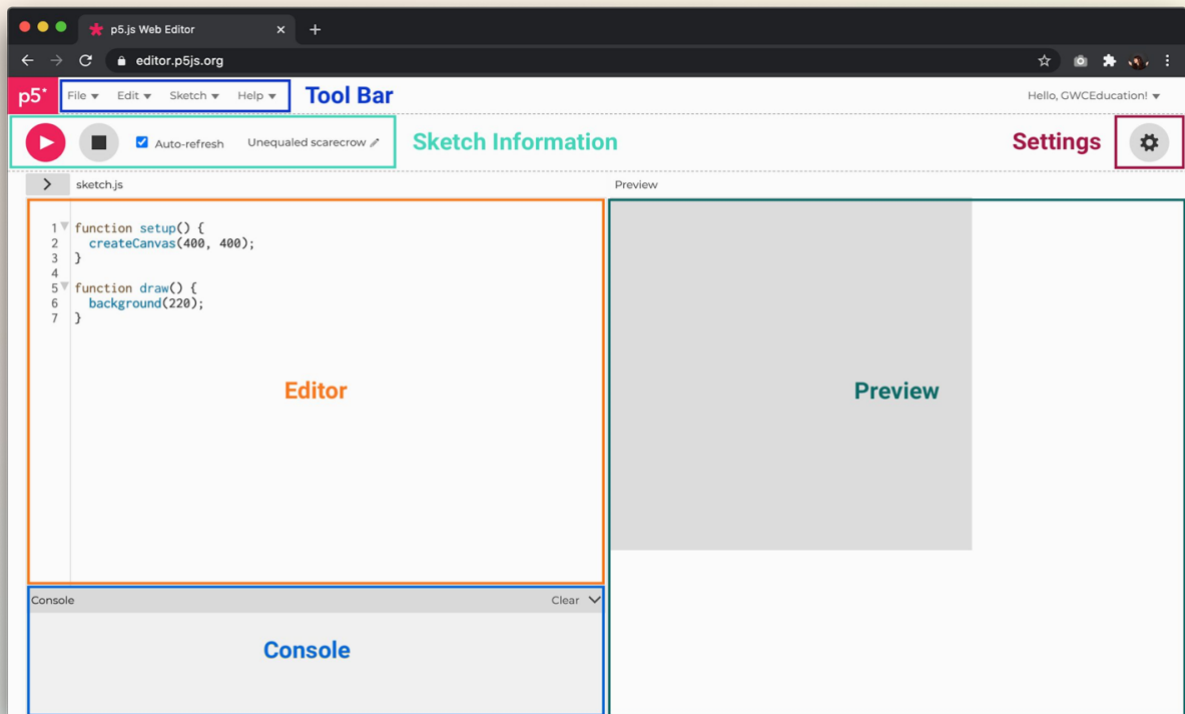
□ **再度ログインできるように、認証情報を安全な場所に保存してください。**パスワードを忘れた場合は、[ログイン](#) ページの一番下にある「パスワードのリセット」をクリックしてください。

インターネット接続が途切れがちであったり、ローカルにエディタで作業したい場合は、2 番目のオプションを検討することができます。必要な材料とライブラリのダウンロード方法については、この [Getting Started](#) をご覧ください。さらにサポートが必要な場合は、遠慮なく Google で検索してください。もし、オフラインで作業すると例は違って見えるかもしれませんが、結果は同じになります。

### 環境探検（5～8 分）



アカウントを取得したところで、p5 オンラインエディタのインターフェースを検証してみましょう。これは、プログラムを書いて実行するための IDE（統合開発環境）を一元化したものです。p5.js で書かれたプログラムは、スケッチと呼ばれます。この環境は、すでに道具が揃っている スケッチブックのようなものだと思います。



→ ツールバー：ページの上部にあるのはツールバーです。

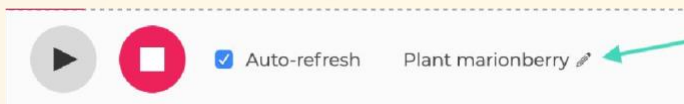
- ◆ **File** メニューでは、スケッチの作成、スケッチの保存、スケッチの複製、複数のフォーマットでのスケッチの共有、スケッチ ファイルのダウンロード、スケッチを開く、および サンプルを開くことができます。注:これらのオプションの中には、スケッチを保存するまで表示されないものもあります。
- ◆ **Edit** のドロップダウンメニューでは、コードを整えたり、スケッチの中の文字や単語を探し(find) たり、それらをナビゲートしたりすることができます。
- ◆ **Sketch** メニューでは、コードにファイルやフォルダを追加したり、スケッチを実行・停止したりすることができます。
- ◆ **Help** メニューには、いつでも役に立つキーボードショートカット、p5 リファレンスページへのリンク、p5 についての詳細が掲載されています。

p5.js のキーボードショートカットの一部は[こちら](#)でご確認ください。

→ **スケッチインフォメーション**：ツールバーの下には、再生ボタンと停止ボタンがあります。再生ボタンは、プログラムの実行を開始します。停止ボタンはプログラムを停止します。もし、変更した後 に再度再生ボタン

をクリックすることなく、プログラムを実行し続けたい場合は、「Auto refresh」ボックスをチェックしてみてください。

右側には、スケッチに予め設定されたタイトルが表示されます。スケッチの名前を変更するには、鉛筆のアイコンをクリックし、新しいタイトルを入力します。



→ **Settings** : スケッチインフォメーションの左側にある歯車のアイコンをクリックすると、設定にアクセスすることができます。ここでは、テーマ、テキストサイズ、アクセシビリティの設定を変更することができます（アクセシビリティについては、もう少し詳しく説明します）。一般的な設定で、自動保存をオンにすることを強くお勧めします。

→ **エディタ** : コードを書く場所です。各行には番号が振られているので、簡単に参照することができます。番号の横にある小さな矢印は、それをクリックするとテキストが折りたたまれることを意味しています。例えば、複数行のコメントを表示する必要がない場合、それらを折りたたむことができます。

→ **プレビュー** : このウィンドウには、プログラムを実行したときのコードの結果が表示されます。

→ **コンソール** : エディタの下にはコンソールがあります。このウィンドウは、エラーメッセージや、変数の値のような、プログラムでアクセスしたいデータなど、プログラムに関する情報を表示します。

## p5.js のアクセシビリティ (2 分)



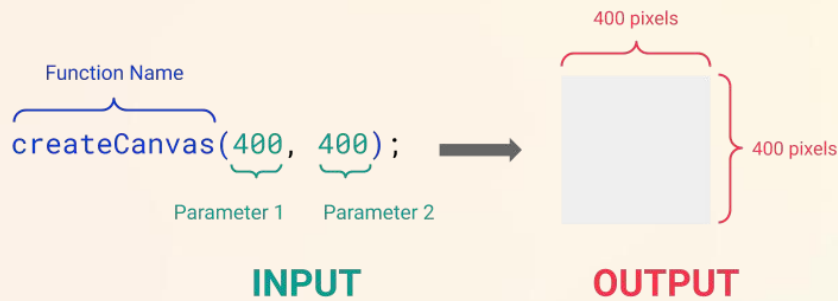
p5 の開発者は、視覚障害者がエディタにアクセスできるようにすることに高い優先順位を置いています。これらのツールは活発に開発されており、NYU でホスト されている進行中の大きな研究プロジェクトの一部です。オンラインエディタの ウェブサイトとエディタ自体は、スクリーンリーダーで読むことができます。オンライン・エディタのウェブサイトとエディタ自体はスクリーンリーダーで読むことができます。アクセシビリティの開発の多くは、プレビューウィンドウの 視覚的出力をスクリーンリーダーで読めるようにするために行われました。この機能の使用に関する詳細は、[p5 ウェブサイト](#)を参照してください。

さまざまな言語やプラットフォームでプログラミングを学ぶ際には、アクセシビリティとインクルーシビリティを常に最前線に置いておく必要があります。歴史的に、デザイナー、エンジニア、プログラマーは、ソフトウェアやハードウェアを作る際に、障がい者のことを優先していませんでした。顔認識などのソフトウェアの台頭により、プログラマーの暗黙の偏見がコードに反映されてしまうため、これは有色人種、女性、その他の社会から疎外された人々にも当てはまります。このような状況は、意識の高まりとともに変わり始めていますが、まだやるべきことはたくさんあります。あなたが作ったものを誰もが使えるようにするために、時間をかけてください。

## ステップ 4 : p5.js の関数を調べる (5~10 分)

**プログラム**とは、コンピュータに実行させるために作成した命令の集合体です。同じ命令を何度も書くのではなく、命令をチャンクにまとめることで、後で再利用することができます。このようなかたまりを**関数**と呼びます。関数は、一連の動作を実行するコード行です。動詞のようなもので、何かをするものと考えてください。P5 では、関数という形でプログラムに指示を与えます。使用する関数のほとんどは、p5.js ライブラリで定義されてい

まず（自分で関数を作成することもできますが、このチュートリアルではその方法を説明しません）。



関数を呼び出したり使ったりすると、プログラムはその中のコードを実行します。例えば、最も重要な関数の1つは `createCanvas()` 関数です。この関数は、グラフィックスを描画し、スケッチを表示する canvas 要素を作成します。つまり、画面サイズを決定するのです。しかし、どのように関数に欲しい画面のサイズを伝えるのでしょうか？そのために、関数に **パラメータ** を渡して、欲しい出力を得るのです。パラメータとは、関数を実行するために使用する入力値のことです。それでは、`createCanvas()` 関数の構文を見てみましょう。

JAVASCRIPT	DESCRIPTION
<code>createCanvas (width, height);</code>	<ul style="list-style-type: none"><li>→ <code>createCanvas</code>: 関数名</li><li>→ <code>()</code>: 括弧を使用して、プログラムに関数を呼び出す必要があることを伝えます。時には、関数のパラメータや入力を括弧の中に入れることもあります。</li><li>→ <code>width</code>: キャンバスの幅をピクセル単位で指定する最初のパラメータです。</li><li>→ <code>height</code>: キャンバスの高さをピクセル単位で指定する第2パラメータです。</li><li>→ <code>;</code>: JavaScript のコード行はすべてセミコロンで終わらなければなりません。</li></ul>

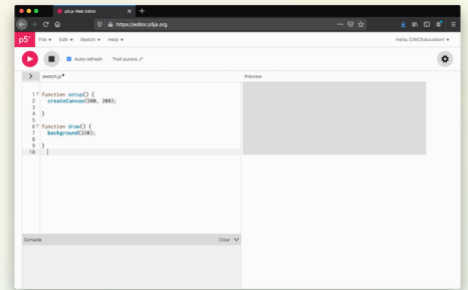
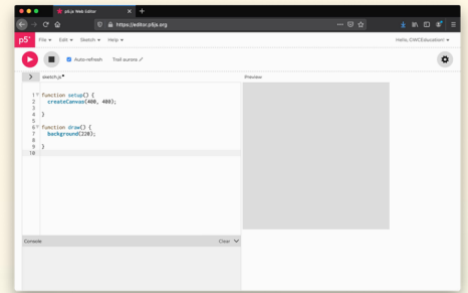
パラメータは、キャンバスの寸法をピクセル単位で設定します。ピクセルは、デジタル画面のグラフィック構成要素です。各ピクセルは画面上の 1 点を表し、1 つの色を持っています。この関数は、すべての p5 スケッチに含める必要があります。

パラメータ値を変更して、キャンバスのサイズを変更してみてください。

❑ **p5 web editor を開き、ログインしてください。** スケッチには、私たちの友人である **createCanvas()** を含むスターターコードがあらかじめ入力されていることにお気づきかもしれません。キャンバスのデフォルトのサイズは、幅 400 ピクセル、高さ 400 ピクセルです。

❑ **再生ボタンをクリックすると、コードが実行されます。** キャンバスの大きさに注目してください。

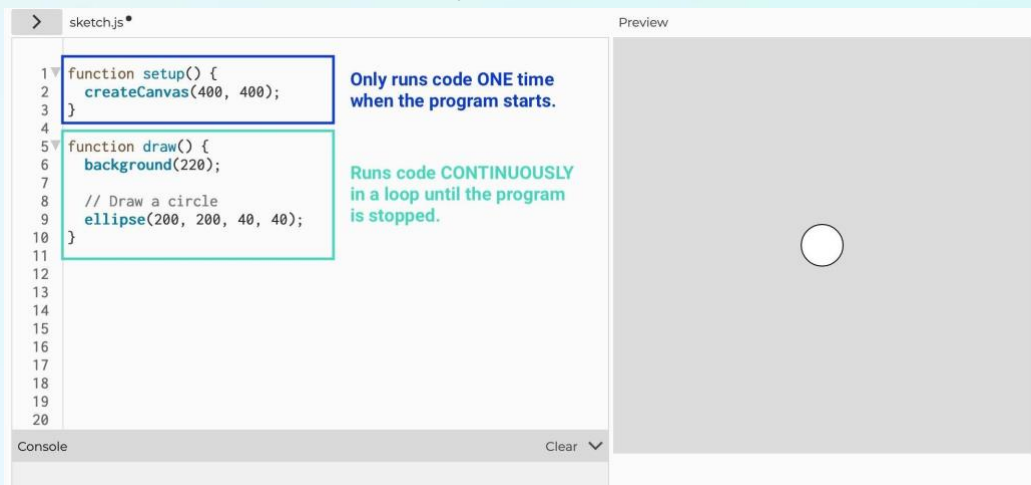
❑ **1 つまたは両方の値を変更してから、再生ボタンをクリックしてコードを再実行してください。** オートリフレッシュボックスをチェックすると、変更するたびに再生ボタンを押す必要がありません。



プレビューウィンドウに、パラメータと同じ大きさのグレーのボックスが表示されるはずですが。

## ステップ 5：プログラムの流れについて学ぶ（5～10 分）

プログラムへの指示の出し方はわかりましたが、その指示をどこに出すのでしょうか？ いつ実行するのでしょうか？ 命令の順番は重要なのでしょうか？ 関数は他の関数の中に入れることができるのでしょうか？ これらの疑問はすべて**プログラムの流れ**に関係しています。これは、プログラムがコード行を実行する順序のことである。p5.js では、プログラムは各コード行を順番に実行します。つまり、1 行目のコード、2 行目、3 行目.....と実行されていくのです。プログラムの流れを制御する方法については後述しますが、まずは p5.js の 2 つのコア関数である **setup()** と **draw()** について知っておく必要があります。



	定義 これって何？	内容 中に何を含めばいい？
setup()	<b>setup()</b> 関数は、プログラム開始時に1回だけ実行されます。スケッチごとに1回だけ実行され、1回目の実行後は再度呼び出すことはできません。	<b>createCanvas()</b> で画面サイズを指定したり、背景色を指定したり（たまたに）、画像やフォントなどのメディアをプログラム起動時に読み込んだりと、プログラム起動時にすぐに実行させたい関数がある。ここで変数を作成すると、 <b>draw()</b> や他の関数でアクセスできなくなります。
draw()	<b>draw()</b> 関数は、そのブロックに含まれるコード行を、プログラムが停止するまで <b>継続的に</b> 実行します。これはメインループであり、アクションが起こる場所です。スケッチごとに1つだけあり、 <b>setup()</b> 関数の後に呼び出されます。	繰り返し起こってほしいことは何でも大丈夫です。

両者の違いをよりよく理解するために、**background()**関数をどこに置くかによってスケッチがどのように変化するかを検証してみましょう。これは、p5.jsのcanvasの背景に使用される色を設定します。RGBや16進数値など、さまざまな色値パラメータを取ることができます。色については、次のセクションで詳しく説明します。ここでは、0（黒）から255（白）までの1つの値を渡すだけで、グレースケールの色を設定します。

JAVASCRIPT	DESCRIPTION
<b>background(redValue, greenValue, blueValue);</b>	<ul style="list-style-type: none"> <li>➔ 背景：関数名。</li> <li>➔ <b>()</b>：括弧を使用して、プログラムに関数を呼び出す必要があることを知らせます。時には、関数のパラメータや入力を括弧の中に入れることもあります。</li> <li>➔ <b>redValue</b>：0～255の赤の値。</li> <li>➔ <b>blueValue</b>：0～255の青色を指定します。</li> <li>➔ <b>greenValue</b>：緑色の値を0～255で指定します。</li> <li>➔ <b>;</b>：JavaScriptのコードはすべてセミコロンで終わらなければなりません。</li> </ul>

スケッチの**setup()**または**draw()**にbackground関数を配置すると異なる結果が得られます。

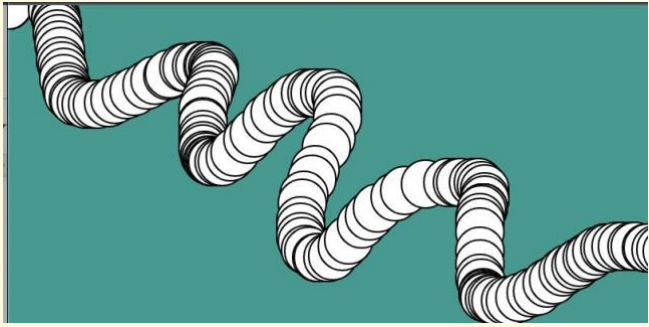
このコードを考えてみましょう。

```
function setup() {
  createCanvas(400, 400);
}
function draw() { ellipse(mouseX,
  mouseY,50,50);
}
```

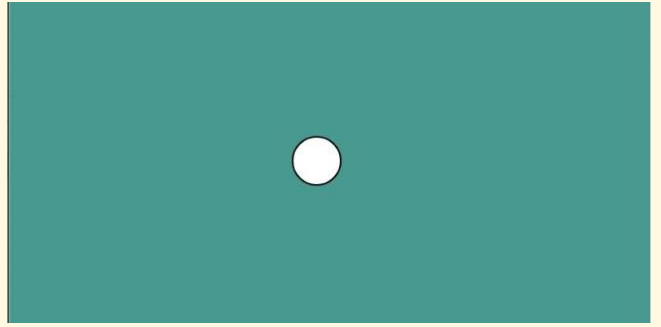
今、このスケッチには背景がありません。キャンバスを作成し、マウスの位置に円または楕円を画面に描きます。**ellipse()**関数は**draw()**の中にあるので、このプログラムはループするたびに、**1秒間に約60回**、プログラムを停止するよう指示するまで新しい円を画面に描き続けます。

2つのスケッチ例で、`draw()` と `setup()` 関数を使用します。一方は `setup()` の中で `background()` を持ち、もう一方は `draw()` の中で `background()` を持ちます。

### SKETCH 1



### SKETCH 2



- [スケッチ 1](#)を開き、キャンバス上でマウスを動かします。円はマウスを追いかけて、その軌道上に他の円の軌跡を残します。
- [スケッチ 2](#)を開き、マウスをキャンバス上に移動させます。これで、円はマウスに追従し、跡を残しません。
- 考えてみてください。どのスケッチが `setup()` の中で `background()` 関数を持っていますか？どのスケッチの `draw()` の中に `background()` 関数があるのでしょうか？それはなぜでしょうか？



リファレンスガイドの 5 ページで自分のアイデアを確認することも忘れないでください。

## ステップ 6：理解度の確認（2 分）

p5.js の関数、`setup()` と `draw()` に関連するプログラムの流れについて、理解度を確認してください。



あなたは、夕食に友人の好物である餃子を一食分作って驚かせようと決めました。手順は覚えています、正しい順番で作っているかどうか確認したいとしましょう。あなたは餃子のために書いた過去のプログラムを見つけたが、不完全なものでした！

プログラムの流れについて学んだことを踏まえて、次のような動作、つまり「関数」を「プログラム」のどこに配置すれば、正しく動作するでしょうか？

追加する必要がある関数：

- 餃子の口を閉じる
- 餃子の種用の材料を計量する
- フライパンから餃子を取り出す

現在のプログラム：

```
setup() {  
  具の材料を混ぜる  
  餃子の皮をすべて集める  
}  
  
draw() {  
  具を包み込む  
  餃子をフライパンに置く  
  餃子を調理する  
  餃子を食べる  
}
```



リファレンスガイドの 6 ページで自分のアイデアを確認することも忘れないでください。

## ステップ 7 : Girls Who Code at Home プロジェクトを共有しよう (5 分)

私たちはみなさんの作品を見るのを楽しみにしていますし、他の人も同じでしょう。あなたの疑似コードを私たちとシェアしてください。@girlsswhocode #codefromhome のタグをお忘れなく！私たちのアカウントであなたを紹介するかもしれませんよ。

**今後の Girls Who Code at Home プロジェクトをお楽しみに！**