Canvas 授業

Canvas とは？？

HTML5 より導入された API。JavaScript を駆使して、Web 上に図形や絵の描画、画像映像の処理など、様々なことが出来る。いわば「グラフィック表現」が Web で可能になる。

・IE9 から対応、他のモダンブラウザは大抵対応（スマホも）
（https://caniuse.com/#search=canvas）

・ピクセルベースで描画表現を行う。
・Canvas 上でアニメーションさせる場合、は自力でスクリプトを駆使する必要がある。

先生呼び出しサービス

http://www.ca-ll.in/room/gsdev09th

Canvas の使用例

http://festival.lattexplus.com/

http://experience.mausoleodiaugusto.it/en/intro

https://tkmh.me/

Canvas はロジックの組み立てが大事！(復習)

CSS

```
①
.area{          ②          ③
    background-color:red;
}
```

① class="area" がついている HTML タグの（どこの）
② background-color（背景色）を（何を）
③ red（赤色）にしたい（どうしたい）

JQ

```
$(".btn").on("click",function(){   ①
                イベント名

    $(".text").fadeOut(3000)
        メソッドとセレクタは「.」で繋げる

});.btn がクリックされたら function(){} の中身を順番に実行！！
```

① class="btn がついている HTML タグがクリックされたら
(いつ・どのタイミングで)
② class="text" がついている HTML タグを（何を）
③ 3 秒かけてだんだん消したい（どうしたい）

■ CSS

命令したい場所を決めて（どこの）
命令したい内容（何をどうしたい）を書く！

■ JavaScript（jQuery）

命令をするタイミングを指定し（いつ・イベント）
命令をする場所を決めて（どこを・セレクタ）
実際に命令する内容を書く（どうしたい・メソッド）

5 歳の息子に初めてのお使いを頼むのと同じ！

例)
お昼になったら（イベント）
近所のスーパーに行って（セレクタ）
大根を買ってきて！（メソッド）

Canvas を扱ってみよう！

1、HTML 上に Canvas 要素を書く。
2、JavaScript で描画のための準備をする。
3、描画処理を書く。

1-1、Canvas 描画の準備をする

canvas 上に描画を色々としたい場合、「canvas タグ（場所）」に「描画（命令）」をする、となりますので、Canvas タグを何度も呼び出すことになります。
何度呼び出されてもいいように、変数に canvas タグを入れておきましょう！


```HTML
<canvas id="×××" width="1024" height="760"></canvas>

//id      →Canvas 要素に命名するユニーク ID
//width   →canvas 要素の幅
//height →canvas 要素の高さ

※ CSS で幅と高さを指定すると、おかしなことになるので注意！！
(※ http://jsdo.it/castero/vo2w)
```

1-1、Canvas 描画の準備をする

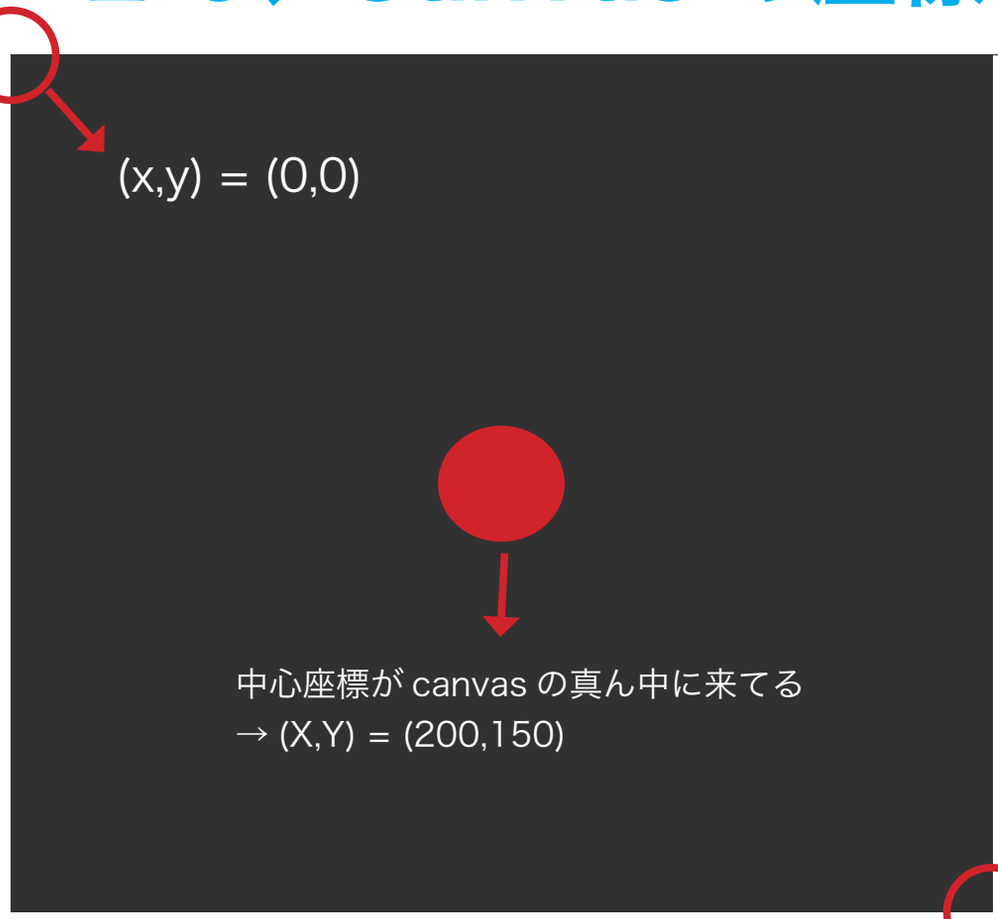
・canvas タグを書いただけでは、描画は実行されない！
・getContext メソッドを実行して初めて描画が出来る！
（変数 ctx に canvas タグを自由に操作できる機能がたくさん入ったことになります）


```
//JavaScript で書くと
const can = document.getElementById("Canvas タグの ID");
const ctx = can.getContext("2d");

//jQuery で書くと
const can = $("Canvas タグの ID")[0];
const ctx = can.getContext("2d");
```

jQuery

2-0、Canvas の座標の考え方

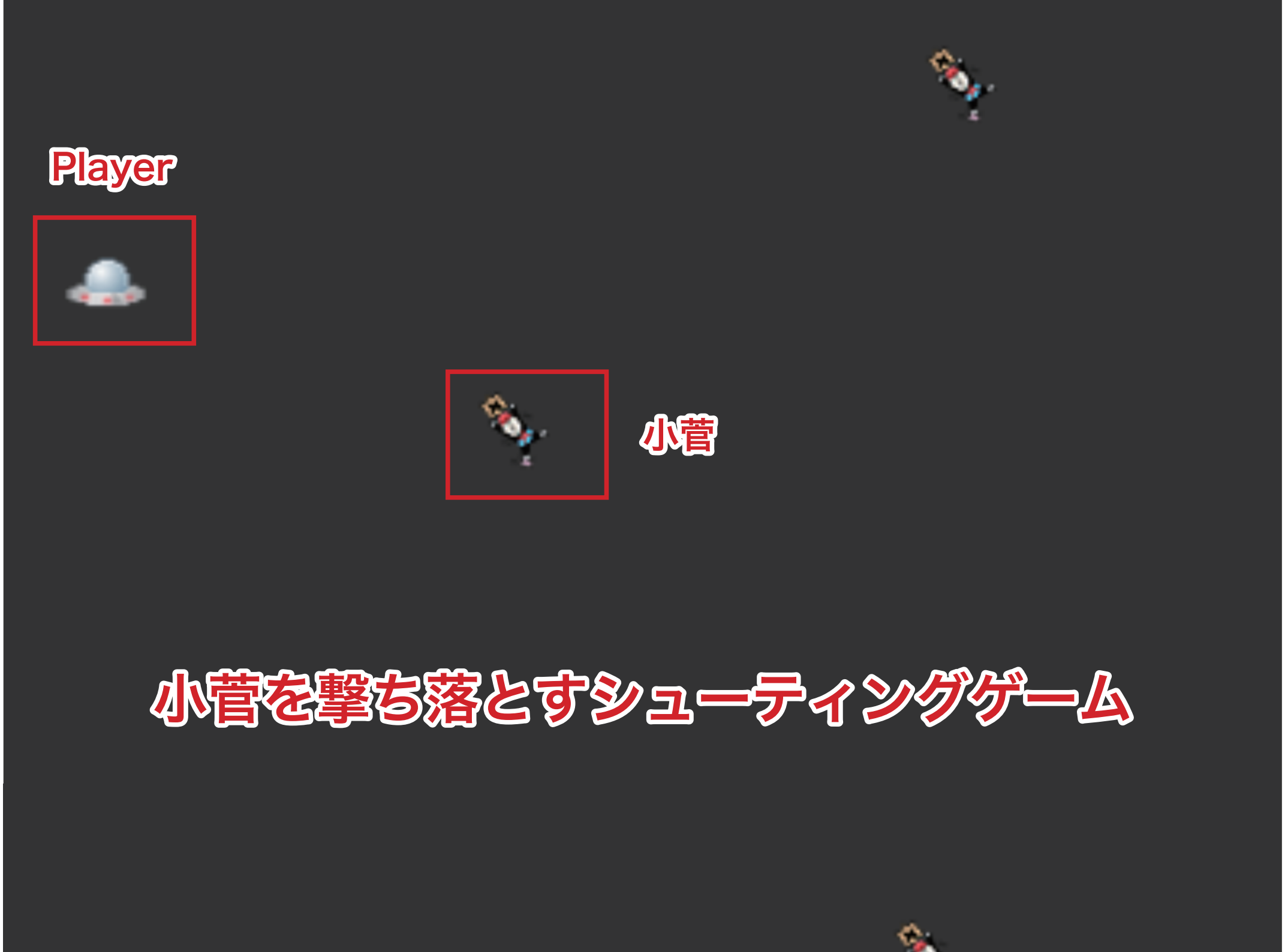


※ canvas 要素の width=400,height=300 とした場合

Canvas 要素は左上を
x 座標 :y 座標 = 0 : 0
として考えます。
描画した図形・線の中心点が x 座標 y 座標
として定義されます。

(x,y) = (400,300)

演習・描画アプリを作る


Player
小菅
小菅を撃ち落とすシューティングゲーム

制作工程 - プレイヤー編 -

drawImage()でcanvas上にPlayer画像配置

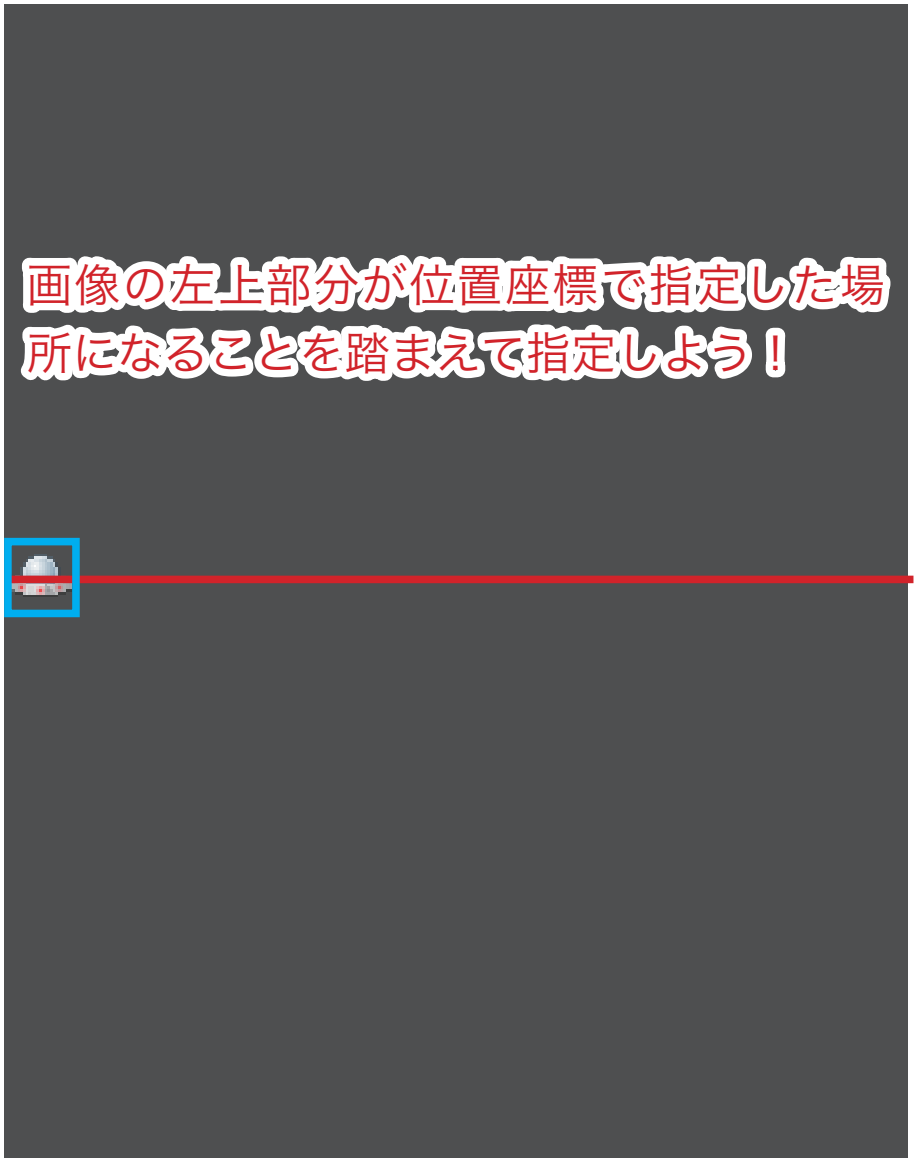
ctx.drawImage(配置画像 ,X 軸位置 ,Y 軸位置 , 横幅 , 高さ);

※ http://www.html5.jp/canvas/ref/method/drawImage.html
※動画も配置可能

まずは Player 機を Canvas 上に描いてみましょう。
drawImage() が実行される時点で、画像データがきちんと読み込めている状態であるかどうかが注意点になります。

※まずは canvas エリアの一番左、上下中央に配置します！

drawImage()でcanvas上にPlayer画像配置




```
const img = new Image();
img.src = "images/ufo.gif";
img.onload =function(){
      ctx.drawImage(img,0,can.height/2-
img.height/2);
    player.image=img;
}
```

※ X 座標・Y 座標の値を変更して、player 機の位置が変わることを確認してみてください！

十字キー操作で player 機を動かす

window.addEventListener("mousedown", 関数名 ,true);

※ jQuery
$(window).on("mousedown", 関数名);

次に canvas 上でマウスを動かして player 機を動かす機能を入れてみましょう。マウスの動きは、mousedown イベントを使って定義可能です。

「マウスが canvas 上で mousemove されたときだけ」player 機を動かしたい！

イベントオブジェクト

```
$(window).on("mousemove",function(e){
    // 実行内容
});
```

function() のところによく書かれているのを見る「e」。
これは**イベントオブジェクト**といって、そのイベントが発生した場所における各種データのまとまりです。

上記例であれば、mouse イベントが発生した場所（地点）における様々なデータが、e という変数の中に入っているのです！

※例
座標位置・・・・e.offsetX（jQuery 式だと e.offset().left）

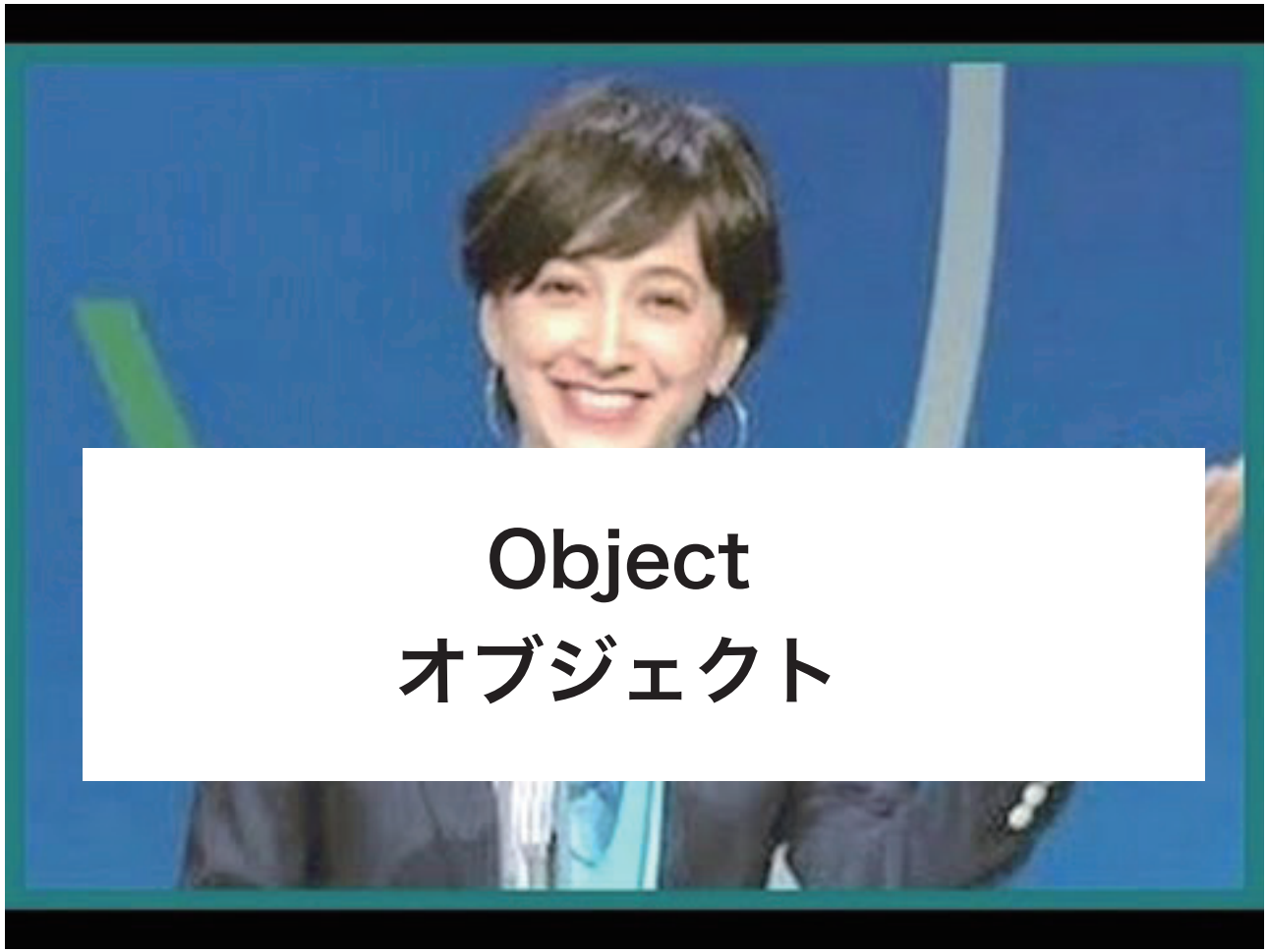

▼KeyboardEvent {isTrusted: true, key: "ArrowLeft", code: "ArrowLeft", location: 0, ctrlKey: false, …}
altKey: false
bubbles: true
cancelBubble: false
cancelable: true
charCode: 0
code: "ArrowLeft"
composed: true
ctrlKey: false
currentTarget: null
defaultPrevented: fals
detail: 0
eventPhase: 0
isComposing: false
isTrusted: true
key: "ArrowLeft"
keyCode: 37
location: 0
metaKey: false
▶path: (4) [body, html, document, Window]
repeat: false
returnValue: true
shiftKey: false
▶sourceCapabilities: InputDeviceCapabilities {firesTouchEvents: false}
▶srcElement: body
▶target: body
timeStamp: 17453.19999998901
type: "keydown"
▶view: Window {postMessage: ƒ, blur: ƒ, focus: ƒ, close: ƒ, frames: Window, …}
which: 37
▶__proto__: KeyboardEvent
例：keyCode →それぞれのキーに割り振られている番号のこと！
(例えば、enter キーなら 13 番 !)

マウスを動かす度にPlayer機位置が変わる
→位置座標の管理方法は？？

x 座標用の変数・Y 座標用の変数を
それぞれ作ってもいいけど・・・面倒！


Omotenasi
オモテナシ


Omo
オモテナ


Object
オブジェクト

オブジェクト

```
let player = {
        posX:0,//X 座標
        posY:can.height/2,//Y 座標
        image:""// 画像データ管理
        w:null,
        h:null


}
```

オブジェクトは、特定のものに関するデータの集まり・固まりのことです。
{ } を使って、複数のデータを key - value 型でカンマ区切りで管理します。

実はみなさんも今まで触れていたオブジェクト

Math オブジェクト	数値演算に関するデータとデータ操作方法がまとまっている。（じゃんけんアプリでやった！！）
Date オブジェクト	時間に関するデータと、データ操作方法がまとまっている。(new Date() して、getMonth() 的な)
DOM オブジェクト	HTML タグに関するデータと、データ操作方法がまとまっている（$("body").CSS("color","red))

JavaScript →様々なデータを操作

オブジェクト
→データそのものの集まり + データ操作方法・関数の集まり

※例
alert()・・・window オブジェクトに紐づいた関数

オブジェクトは関数も管理できる

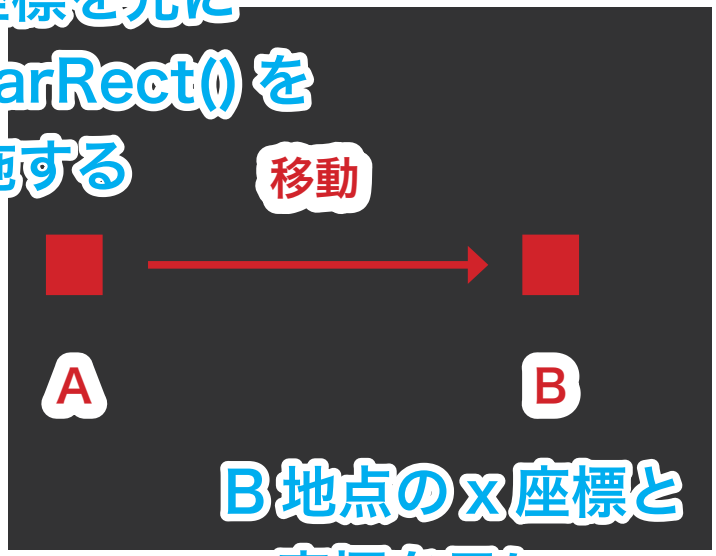
```
let player = {// 途中省略
        image:""// 画像データ管理
        draw:function(){ // 画像ロードのための関数
            const img = new Image();
            img.src = "images/ufo.gif";
            img.onload =function(){
                ctx.drawImage(img,player.posX,player.posY);
                player.w = img.width;
                player.h = img.height;
                player.image = img;
            }
        }
}
```

player.image() 関数を実行すれば画像が canvas 上に表示されるようになる！

clearRect（canvas 上の描画物消去）

ctx.clearRect(基準Ｘ座標 , 基準Ｙ 座標 , 消去エリアの幅 , 消去エリアの高さ)

A 地点の x 座標と y 座標を元に clearRect() を実施する



B 地点の x 座標と y 座標を元に drawImage() を実施する

canvas 上の物は DOM オブジェクトのように扱えないため、animate などが効かない！

→ A から B への移動を表現する場合、**A に描かれている内容を一度消去して、そのあと B に新しく描き直す**ことで、「移動」を表現します。
（clearRect → drawImage を繰り返す）

ここまでの演習

十字キーを動かしたら、Player 機が上下左右に動くようにする !!

■注意点

・canvas エリアからはみ出して Player 機が表示されないようにする。

・Player 機が重複して表示されないようにする。

Player 機から弾を発射させる！

制作工程 - 弾の発射から弾を動かすまで -

弾の座標情報は Player の座標に依存する

Math.random を活用し出現確率を操作

敵を動かし画面外になった敵は削除

fillRect() でcanvas上に四角形の描画

ctx.fillRect(X 軸位置 ,Y 軸位置 , 横幅 , 高さ);

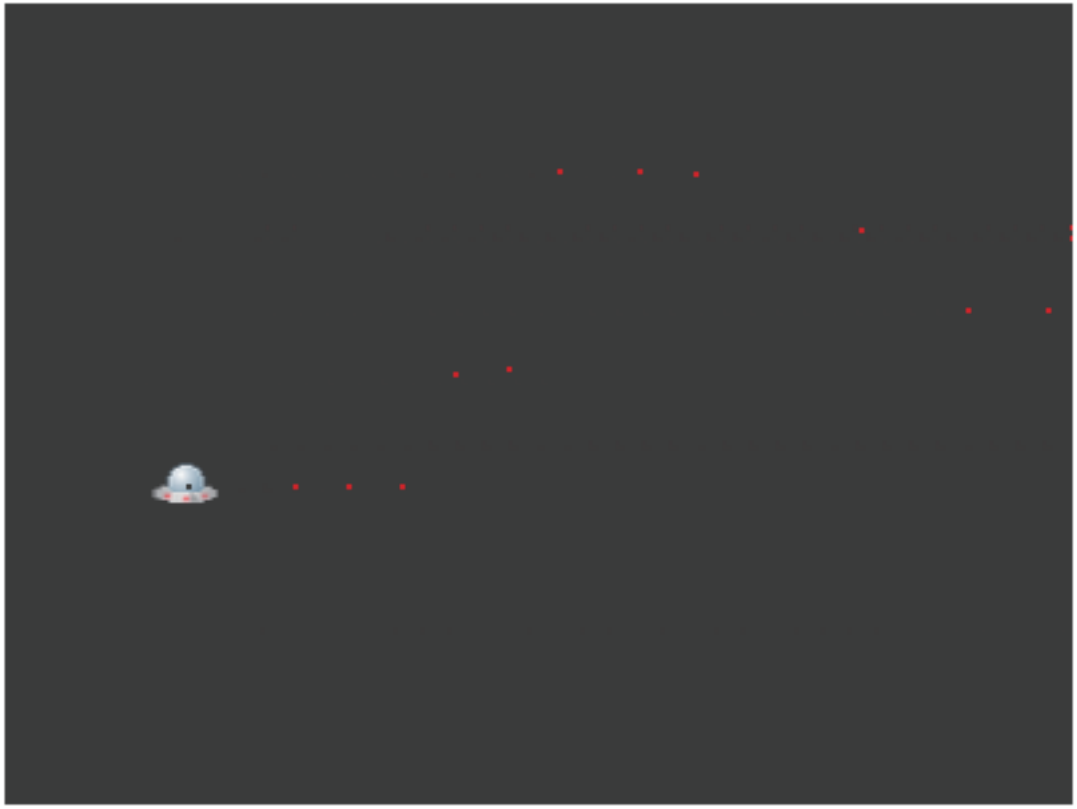
※ http://www.html5.jp/canvas/ref/method/fillRect.html

次に、弾を描くための練習として、canvas 上に四角形を描いてみましょう。
X 軸、Y 軸位置で指定した座標は、四角形の左上部分に該当します。

※色などの変更も可能（ fillRect() の前に必ず記述）
ctx.fillStyle ="**#f00**"; // 赤への変更

弾オブジェクトの作成（１発１発の）

```
let ball = {
        speed:10, // 弾の発射スピード
        w:8, // 弾の幅
        h:8, // 弾の高さ
        posX:0, // 弾の X 位置情報
        posY:0, // 弾の Y 位置情報
        color:"#d00", // 弾の色
}
```

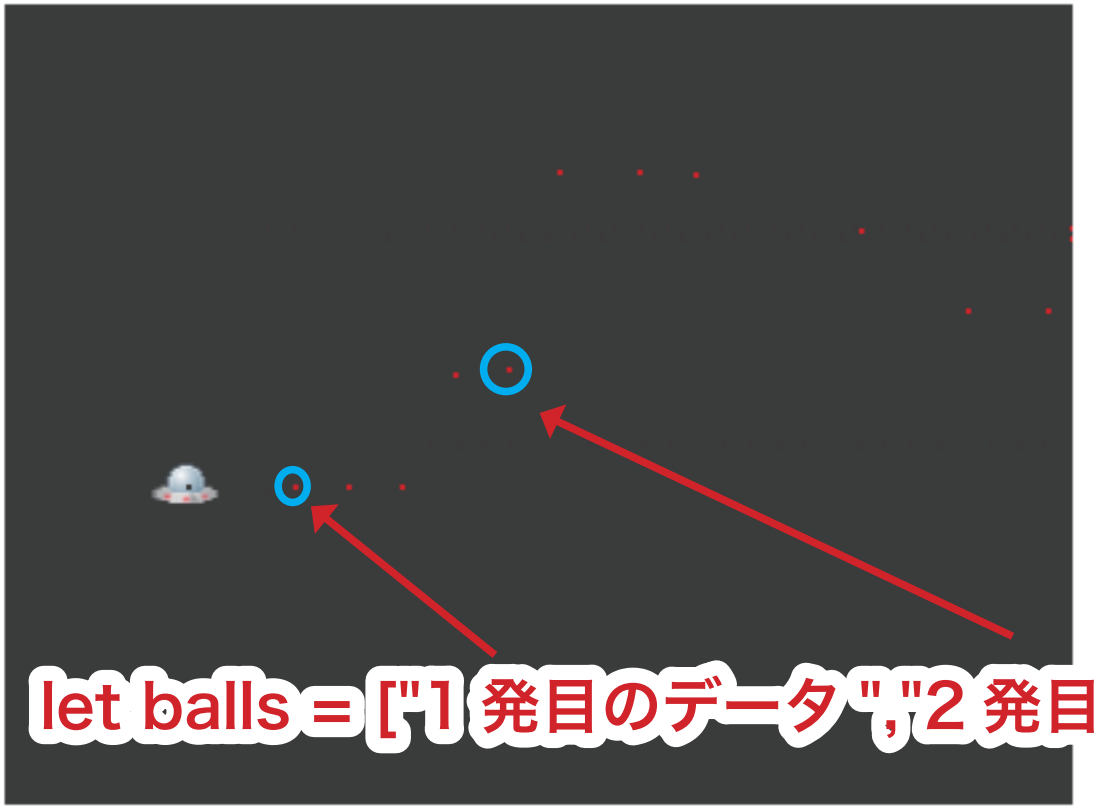


弾は複数打ちたい！

複数の弾の位置座標管理はどうすればいい？
オブジェクト１つだと上書きされちゃう・・・

そうだ！配列だ！

配列を使って弾のデータを個別で管理



マウスクリックの度に配列に ball オブジェクトのデータを複製し、
新しい弾のデータを push!!!

Object.assign()

JavaScript では、オブジェクトを他の変数にコピー複製すると、元オブジェクトへの参照がコピーされるため、元あるいはコピーオブジェクトのプロパティ変更をすると、双方のプロパティが変更されてしまいます。

※例

```
let ball = {
    speed:10,
    w:8,
    h:8,
    color:"#d00"
}
ball2 = ball;
ball2.color="#00f";
console.log(ball);//color → #f00 になっている！
```

これだと状況に応じて弾の見た目を変更したりすることができない！

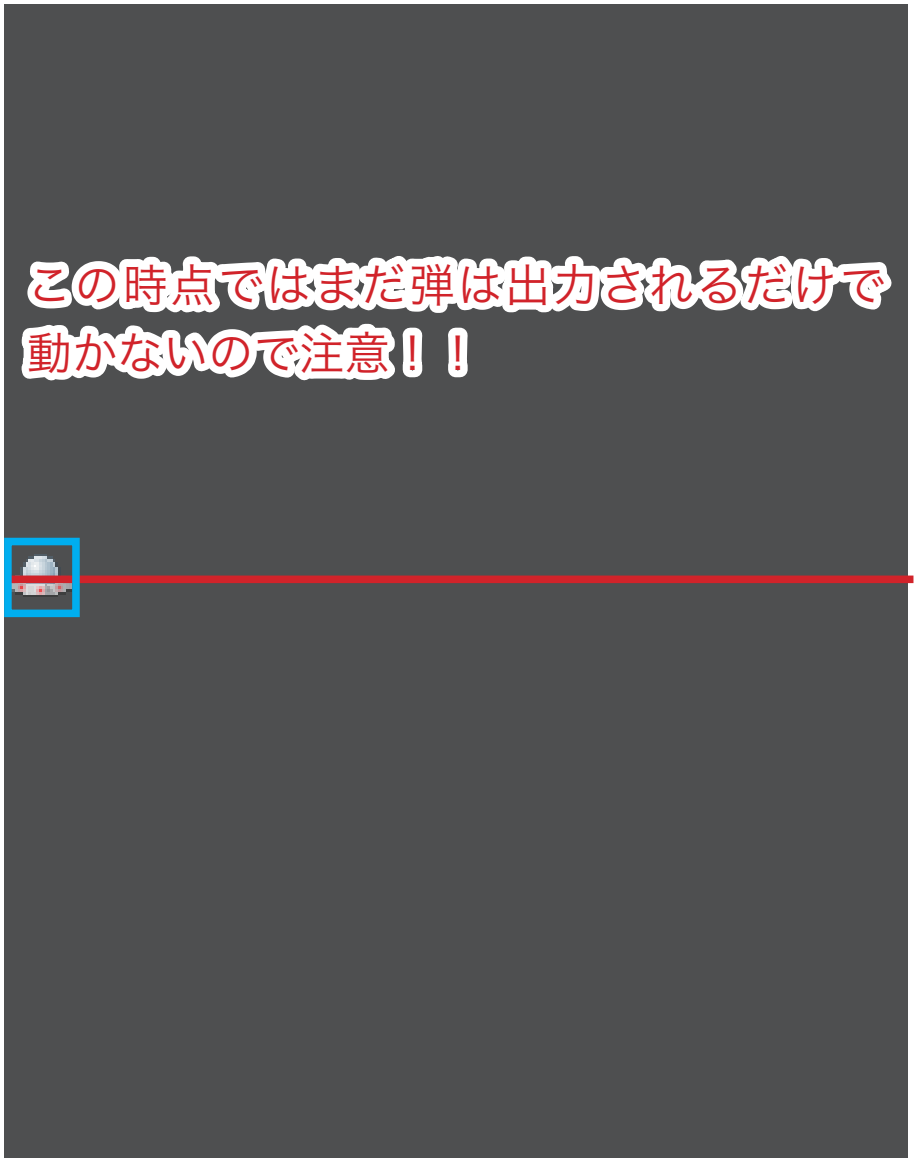
Object.assign()

let copyObject = Object.assign({},baseObject);

Object.assign() を使用すると、元のオブジェクトからデータのみをコピーし、複製オブジェクトを作成することができます。
ちなみに、複数のオブジェクトをがっちゃんこして、新しいオブジェクトを作成することも可能です。

```
※例
let baseKosuge = {age:35,gender:"men"};
let newKosuge = Object.assign({prefecture:"Saitama"},baseK
osuge);
console.log(newKosuge);
→{age:35,gender:"men",prefecture:"Saitama"};
```

弾を出力する（Object.assign() を活用）




```
let ball = {
        speed:10,
        w:8,
        h:8,
        color:"#d00",
}

let balls = []; // 弾を格納する配列を作成

let ballInit = function(e){
        let newball = Object.
assign({posX:player.posX+player.image.
width,posY:player.posY+player.image.
height/2},ball);
        ctx.fillStyle =newball.color;
        balls.push(newball);
}
```

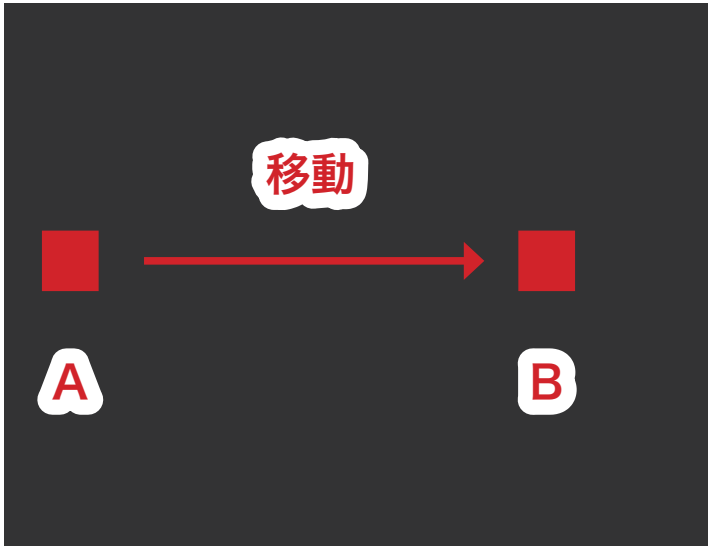
弾を動かす

弾を動かす！！

基本的には Player 機を動かすときの考え方と同じになります。

弾の場合は画面外に消えるまで弾を動かし続けなければならないため、

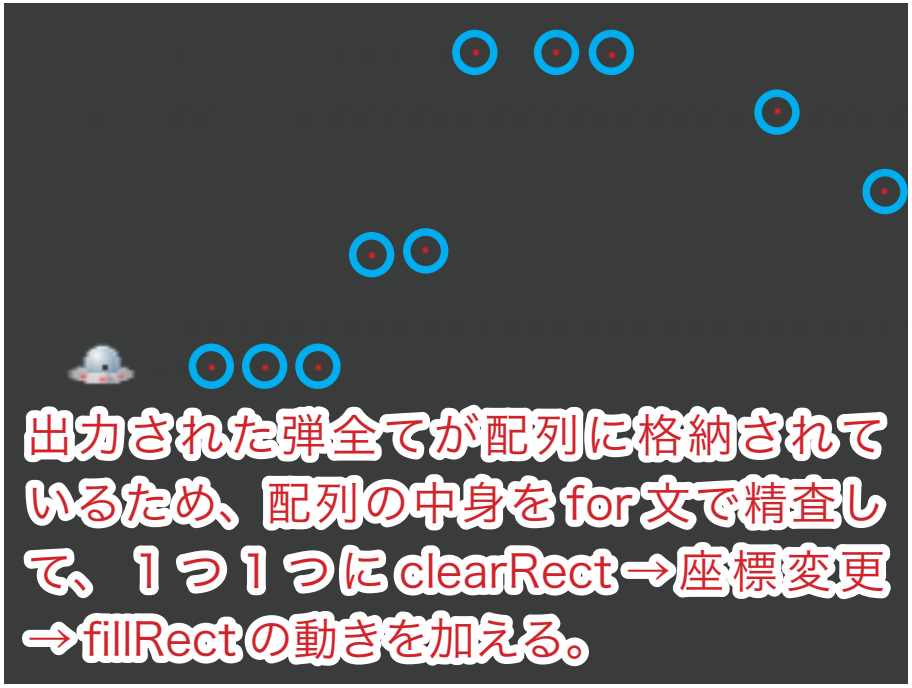
動かす関数を作ってループで動かす仕組みが必要です。



canvas 上の物は DOM オブジェクトのように扱えないため、animate などが効かない！

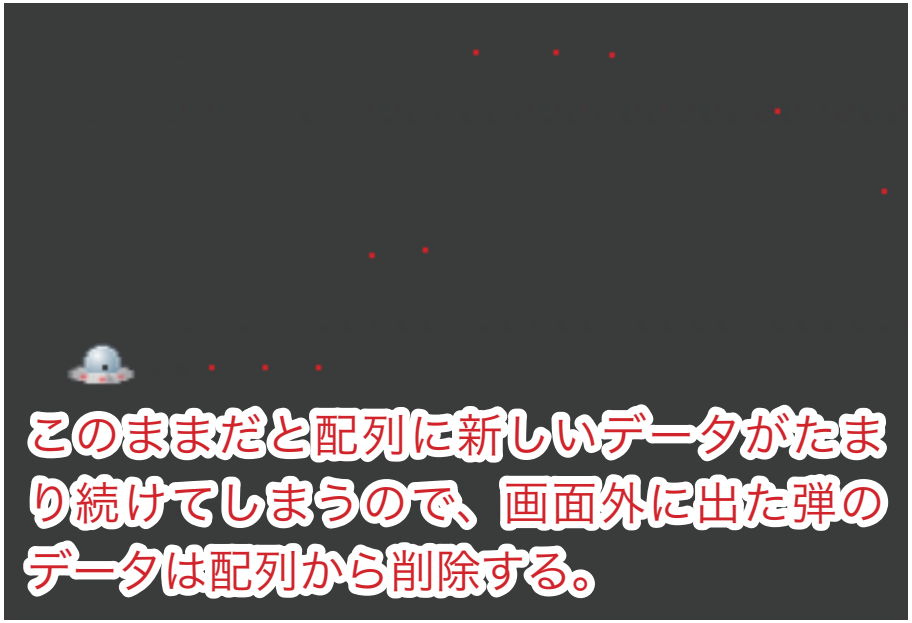
→ A から B への移動を表現する場合、**A に描かれている内容を一度消去して、そのあと B に新しく描き直す**ことで、「移動」を表現します。
（clearRect → drawImage を繰り返す）

弾を動かす



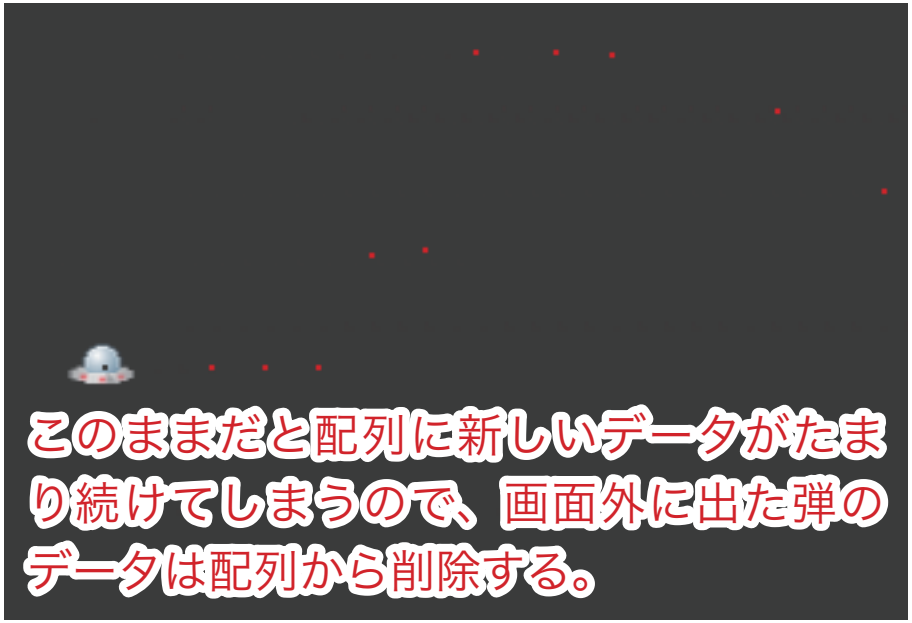

```
let ballMove=function(){
      for(let i=0;i<balls.length;i++){
            ctx.clearRect(balls[i].posX,balls[i].posY,balls[i].w,balls[i].h);
            balls[i].posX +=balls[i].speed;
            ctx.fillRect(balls[i].posX,balls[i].posY,balls[i].w,balls[i].h);
      }
};
```

画面外に出た弾のデータは削除




```
let ballDelete=function(){
      new_balls = []; // 精査用に新しい配列作成
      for(let i = 0; i < balls.length; i++) {
            if (balls[i].posX>=can.width) {
                  delete balls[i]; // 配列から削除
            }else {
                  new_balls.push(balls[i]);
            }
      }
      balls = new_balls;// 精査後、元の配列へ
};
```

setInteval 関数でループで関数実行




```
setInterval(function(){
        ballMove();
        ballDelete();
},100);
```

ここまでの演習

十字キーを動かしたら、Player 機が上下左右に動き、かつ弾を自由自在に発射させる！！！

■注意点

・配列に弾のデータがたまりすぎないように削除処理をきちんと行う！

・位置の管理の仕方間違えると canvas 上に描画が残るので注意。

敵を出現

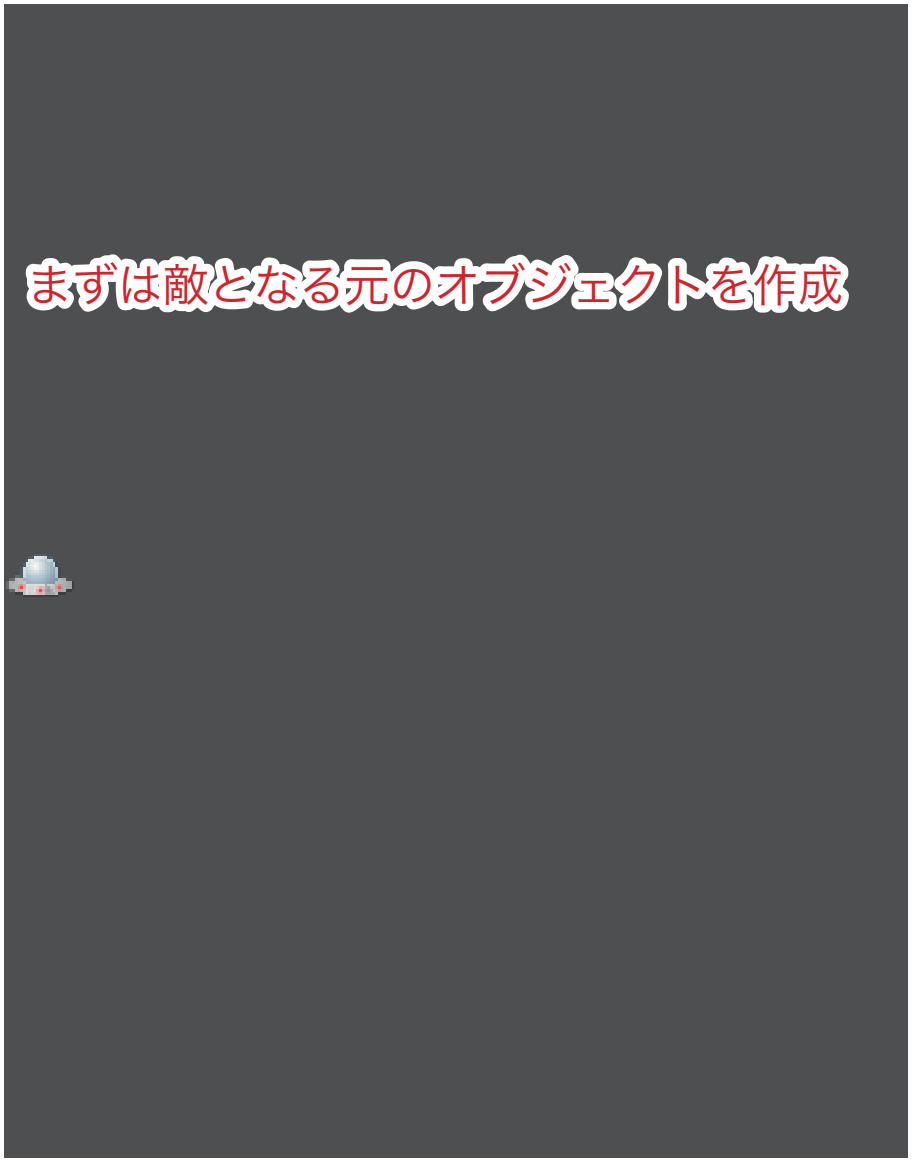
制作工程 - 敵の出現編 -

敵画像を読み込む関数作成しオブジェクトで他データと共に管理

Math.random を活用し出現の確率を操作

敵を動かし画面外になった敵は削除する処理を行う

drawImage()でcanvas上に敵画像配置



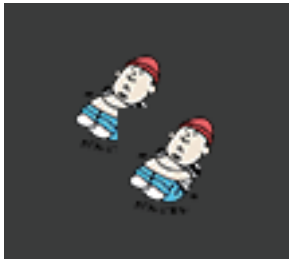

```
let enemy = {
        speed:5,
        w:null,
        h:null,
        posX:can.width-8,
        image:"",
        draw:function(){
                const img = new Image();
                img.src = "images/stamp5.png";
                img.onload = function(){
                        enemy.image = img;
                }
        }
}
```

演習

敵を動かして出現させる！

■注意点

・敵の表示位置（Y 軸）がランダムになるように！
・if 文を使って出現確率を制御できたら Good！
・直前に出た敵と被らないようにできたら最高！
（時間的に授業内で扱えないので任意）



当たり判定

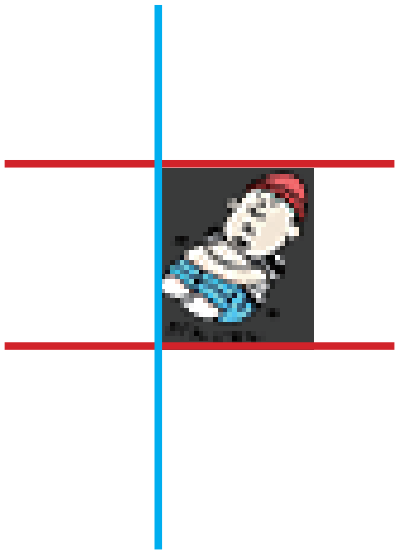
弾の右端の座標が敵の左端の座標以上になり

弾の上端が敵下端以下にあり

弾の下端が的上端以上にある

↓

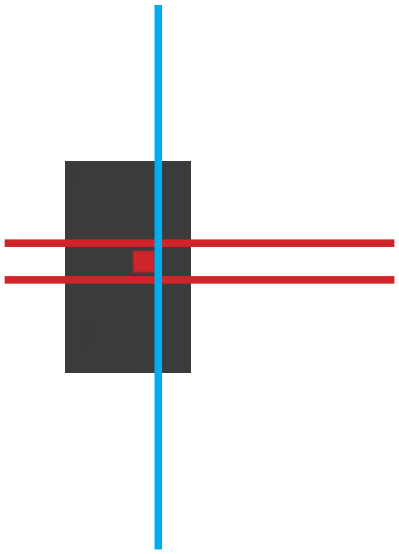
当たり！と判定できる。



Y 座標：ememies[i].posY

Y 座標：ememies[i].posY+ememies[i].h

X 座標：ememies[i].posX



Y 座標：ball[i].posY

Y 座標：ball[i].posY+ball[i].h

X 座標：ball[i].posX+ball[i]+w

演習

当たり判定を実施して、当たった弾と敵は消去しよう！

■注意点

・配列から該当要素を削除し、インデックスを詰める場合は、splice() を使用するとよいです。

https://msdn.microsoft.com/ja-jp/library/wctc5k7s(v=vs.94).aspx

課題

課題内容

弾・Player・敵以外に新しいオブジェクトを作成し、ゲームの機能拡張をする。

■例

・**ボスオブジェクト**を作成し、ボスには３発当てないと倒せないようにする。
・敵も弾を発射するようにし、**敵の弾オブジェクト**を作成する。
・**アイテムオブジェクト**を作成し、アイテムをゲットしたら弾が大きくなる。

■プラスアルファ要素例

・操作をスムーズにして UI/UX を高める。
・スコア機能・時間制限を追加する。

Developer Tools のブレイクポイント機能

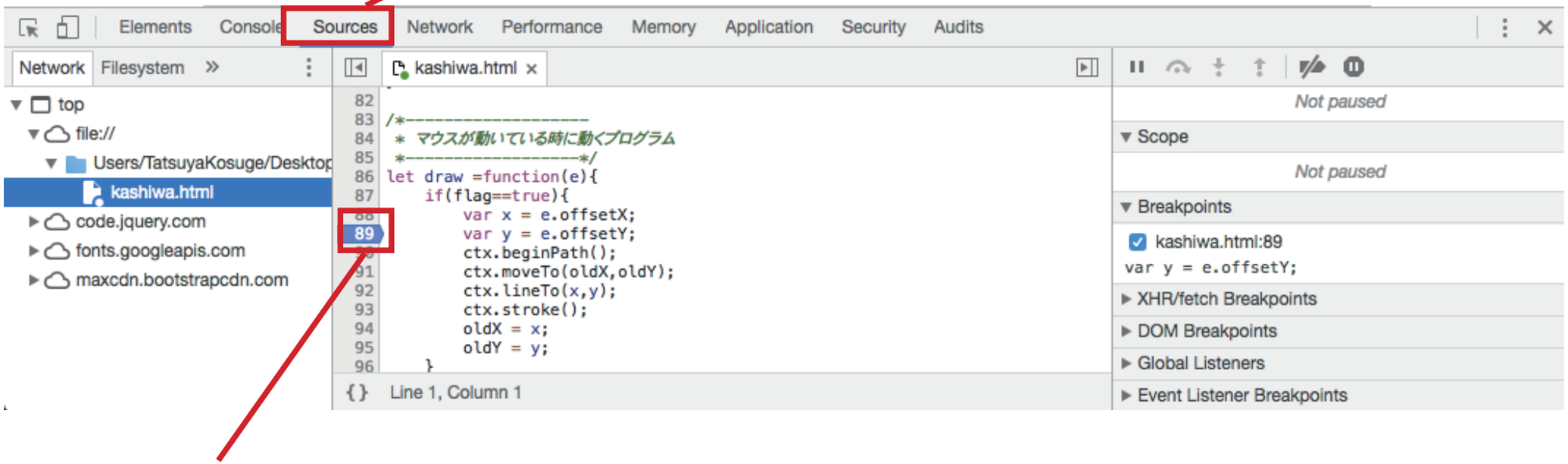
Developer Tools のブレイクポイント機能を使ってデバッグ上手に！

指定した関数が動かない！？そんなときはありませんか？？
そんなときは、Developer Tools のブレイクポイントを使うと、かなり便利に効率的にデバッグを行うことができます。

今回のうちに、ブレイクポイント機能の使い方をマスターしましょう！

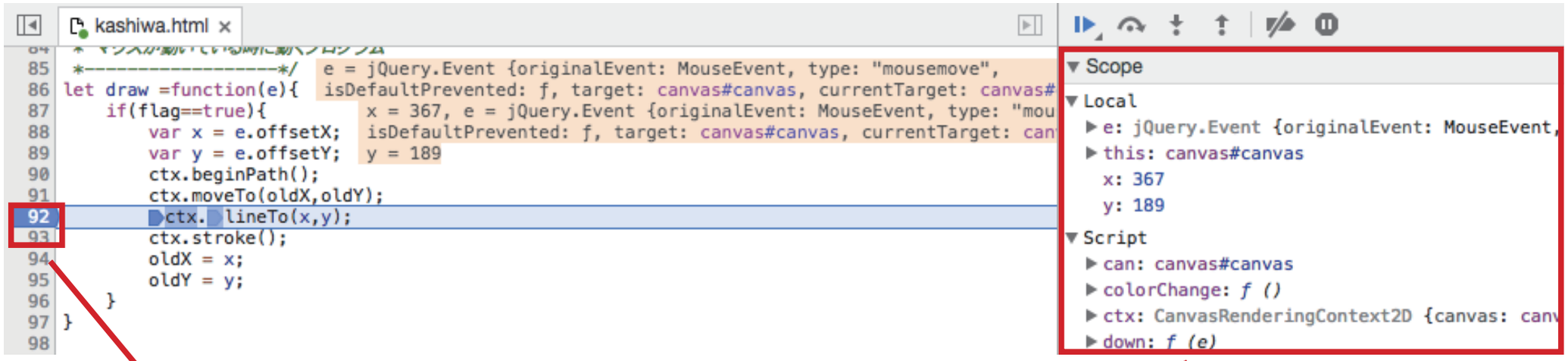
Developer Tools のブレイクポイント機能

Sources を選択



プログラムの動きを止めたい行にマーカーを打ちます。その行が読み込まれた時点で、一度動きを止めることができます。

Developer Tools のブレイクポイント機能



プログラムの動きを止めたい行にマーカーを打ちます。その行が読み込まれた時点で、一度動きを止めることができます。

動きを止めた時点で、各変数のスコープ範囲や、具体的な変数の中身などを確認することができます。
（scope 欄）