

プログラミングで○×
ゲームを作ってみよう

目次

- 本講義の目的(5min)
- システムの構成について(10min)
- JavaScriptを学んでみよう(30min)
- ○×ゲームを作成しよう(45min)

目的

1. SEになる選択肢を広げてもらう-> 実際のWeb制作現場でも同じような工程が発生し、頭を使って実装を進め思い通りに動いた時の楽しさを味わってもらいたい.
 2. 就活で実際に活用して欲しい->本講座でプログラミング言語を学んで他の学生には無い力をつける事が出来る.
- 大学生でJavaScriptを使ってアプリを作った経験があるだけでどの業界でも加点になる

システムの構成について

- 実際のWeb制作現場では一人で作るわけではなく、チームで作る事が多い.

例

ディレクター 1名 (ビジネス戦略を考える)

プロダクトマネージャー 1名 (制作チームをまとめる)

サーバーサイドエンジニア 3名

フロントサイドエンジニア 3名

システムの構成について

- 今回皆さんに体験していただくのは、サーバーサイドエンジニアの分野です.

フロントサイドと何が違うのか？

サーバーサイドはユーザーから見えない部分を作成

フロントサイドはユーザーから見える部分を作成

※給料や技術難易度はほとんど変わりません

具体的に見えない部分って？

システムが動いているロジックを頭を使って作成していく。
サーバーサイドのプログラミング言語の例

Java

C言語

Node.js(JavaScriptのサーバーサイド)

Python

Go

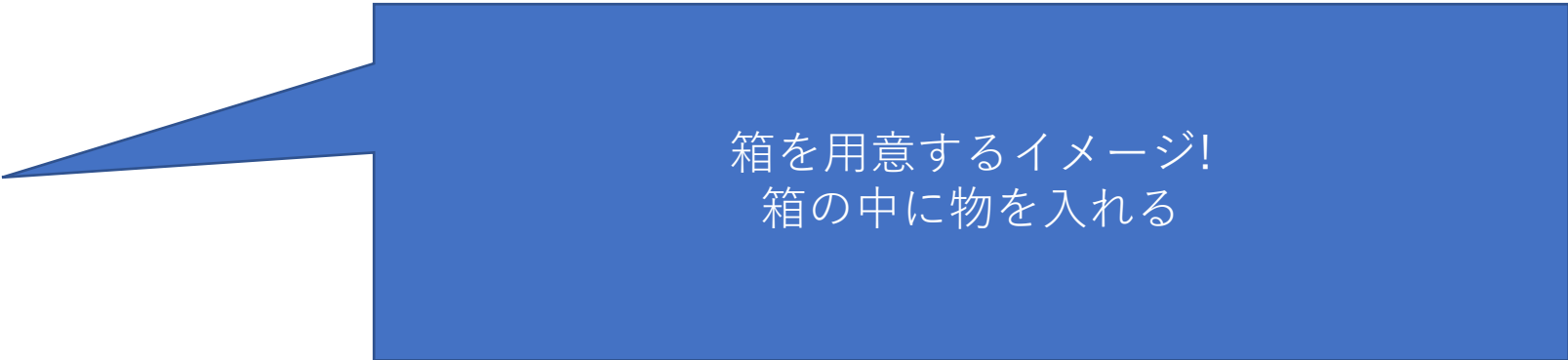
などを使う…

JavaScriptを学んでみよう

```
1 // Promise from setTimeout
2 const afterSomeTime = (time) => new Promise(resolve => {
3   setTimeout(() => {
4     resolve(true);
5   }, time);
6 });
7 const callAfterSomeTime = (callback, time) => afterSomeTime(time).then(callback);
8
9 callAfterSomeTime(() => console.log('Hello after 1500ms'), 1500);
10
11 const getData = async (url) => fetch(url);
12
13 document
14   .querySelector('#submit')
15   .addEventListener('click', function() {
16     const name = document.querySelector('input[name]').value;
17
18     // send to backend
19     const user = await fetch('/users?name=${name}');
20     const posts = await fetch('/posts?userId=${user.id}');
21     const comments = await fetch('/comments?post=${posts[0].id}');
22
23     //display comments on DOM
```

- 変数宣言

```
let number  
number = 5
```



箱を用意するイメージ!
箱の中に物を入れる

let 任意の変数名

変数名の中に、文字や数字を格納できるぞ!

足し算や引き算，掛け算も出来る!

```
let number;  
number = 5  
number = number * 3  
// numberの中身は 15になる
```

3行目でnumberの中を上書きした

- 条件分岐

```
let number = 5;  
if (number == 5) {  
  number = number + 10;  
}
```

先程の変数宣言では = が一つだった。
プログラミングでは, = は代入の意味

==で同等という意味になる!

if(条件){
 条件が成立した時だけ実行
}

```
let number = 5;  
let number2 = 2  
if (number > 3 && number2 > 1) {  
  number = number + 10;  
}
```

2つ以上の条件を入れたいときは
&を二つつけて
「かつ」という使い方も出来る

- 繰り返し分

```
let number = 0;  
for (let i = 0; i < 3; i++) {  
  number = number + i;  
}
```

iが0の時
iが1の時
iが2の時
の処理を繰り返してくれる!

forの括弧の中を説明すると
iという変数を宣言して,0を代入
iが任意の条件になるまで
iをプラス1していく

for文では処理を繰り返す.

括弧の中身がわかりにくい人は,

for(let i=0 ; i< 繰り返したい回数を入れる ;i++)

で覚えてしまおう!

「;」を忘れずに!

- 配列

```
let state = new Array(3);  
state[0] = 1;  
state[1] = 2;  
state[2] = 100;
```

1行目の new Array(3)は3つの箱を準備するという宣言.
今回は宣言の部分は使わないので, 格納方法だけ覚えておこう

配列は複数の値を格納したい時に使う
変数宣言では格納できる値が1つだけだった.
変数と少し違うのは格納方法.
何番目に格納するかを指定する!

- 二次元配列

```
let state = new Array(3) ;  
for (let i = 0; i < 3; i++) {  
  state[i] = new Array(3) ;  
}  
  
state[0][0] = 2 ;  
state[0][2] = 5 ;  
state[2][2] = 3 ;
```

赤枠の部分では二次元配列を作っている
3個の箱を作って、
その後、一つ一つの箱に更に3個の箱を
作っている。

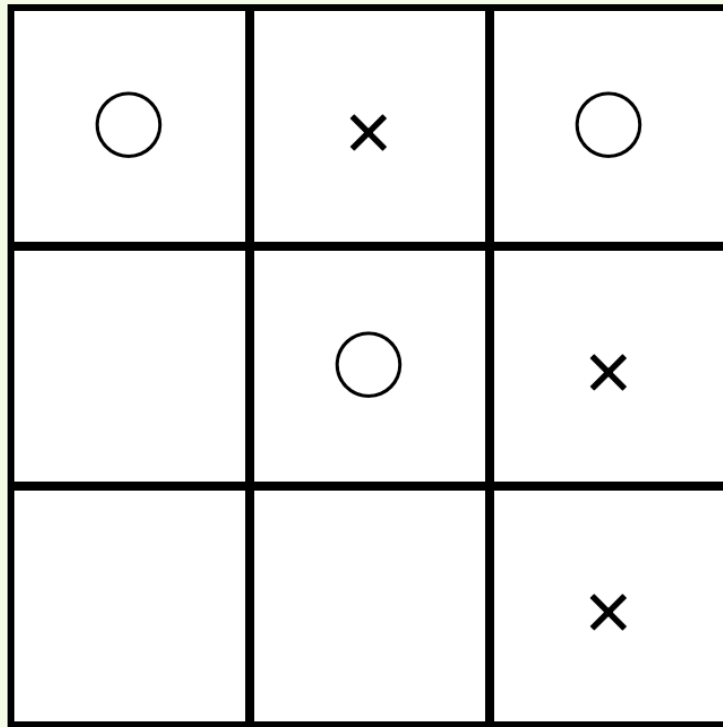
二次元配列は、先程の配列とほぼ同じ
格納方法は、変数名[y][x] = 値

※今回は二次元配列の作り方(赤枠)は覚えなくても良いです。

代入方法は覚えましょう。

2	[0,0]	[0,1]	[0,2]	5
	[1,0]	[1,1]	[1,2]	
	[2,0]	[2,1]	[2,2]	3

○×ゲームを作成しよう



○	×	○
	○	×
		×

GitHubから雛形をダウンロードしよう

Paiza cloudを開いて
ターミナルを選択

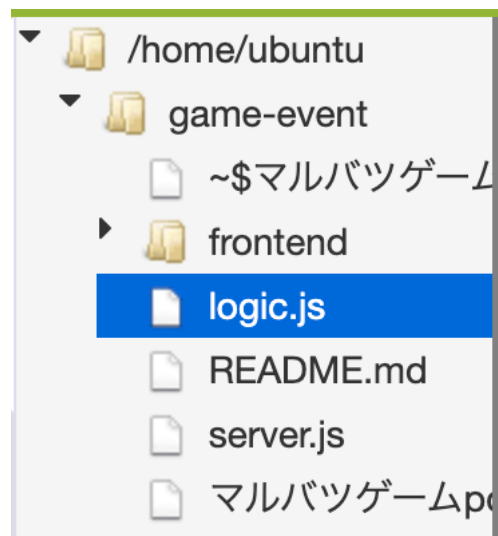


ターミナルで

git clone <https://github.com/moyuta/games.git>

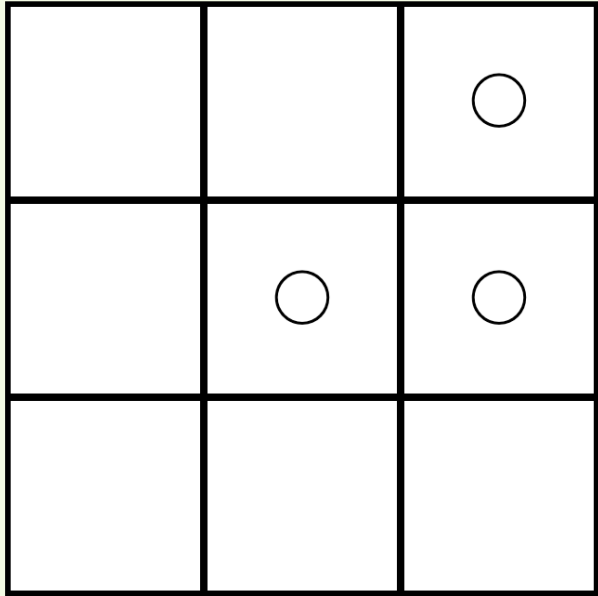
を入力し, enter その後, cd game-event

node server を入力



皆さんが変更するファイルは、logic.jsのみです。
game-eventをクリックした後、logic.jsが出現するので、
ダブルクリック!

問題1 ○と×が交互に置けるようにしよう



現状、丸のみ置かれる。

turnを1と-1が交互で置かれるように工夫しよう!

ヒント: $\text{turn} = \text{turn} * ???$

を挿入

問題2. 引き分けのパターンを考えよう

○×ゲームにおいてどうなったら引き分けになるのか？

ヒント: countの数が?個になったら引き分け

×	○	×
×	○	○
○	×	○

問題3. × と○が勝つパターンを考えよう

- × も○も勝つパターンは一緒です.

ヒント: それぞれ8パターンずつあります

For文を使えたらfor文を使って簡略化して書こう!