Cloud Functions
+
Cloud Firestore で
ミニゲームのバックエンド作ったよ

Yuya Miyazaki
2019.05.24@#1 Fukuoka.Firebase

1. **イントロダクション**
2. 前回までのあらすじ
3. 作ったものの紹介（デモ）
4. Firebase の超基礎的知識
    1. Firebase Cloud Functions とは
    2. Firebase Cloud Firestore とは
    3. Firebase Hosting とは
5. ゲームの設計とか

5/14 Vue Night in Fukuoka でフロントエンドの話をしました

時間	詳細	発表時間	発表者
19:00	**開場**		
19:15	**オープニング**		
19:20	**発表開始**		
	久しぶりにVue.jsを使って簡単なゲームを作ったょ	10	Yuya Miyazaki
	VueでAmplify Frameworkに挑戦してみた	8	夏目祐樹
	プロジェクトでCSSモジュールを使った感想	5	zero
	VueとGraphQLのアプリケーション開発について	15	daiki7nohe
	Nuxt.js のbuid オプションのいくつかについて	10	sunecosuri
	VeturのAuto completionにGridsomeを対応させた話	10	チャンカツ
	「Vue.js 分からーん！」と思ってたのは、自分が『オブジェクト指向の奴隷』だったんだと気がついた話	5	垣花　暁
	個人的 Vue.js Tips 集 - 2019年春	15	mya-ake
未定	発表終わり次第交流		
21:30	撤収		

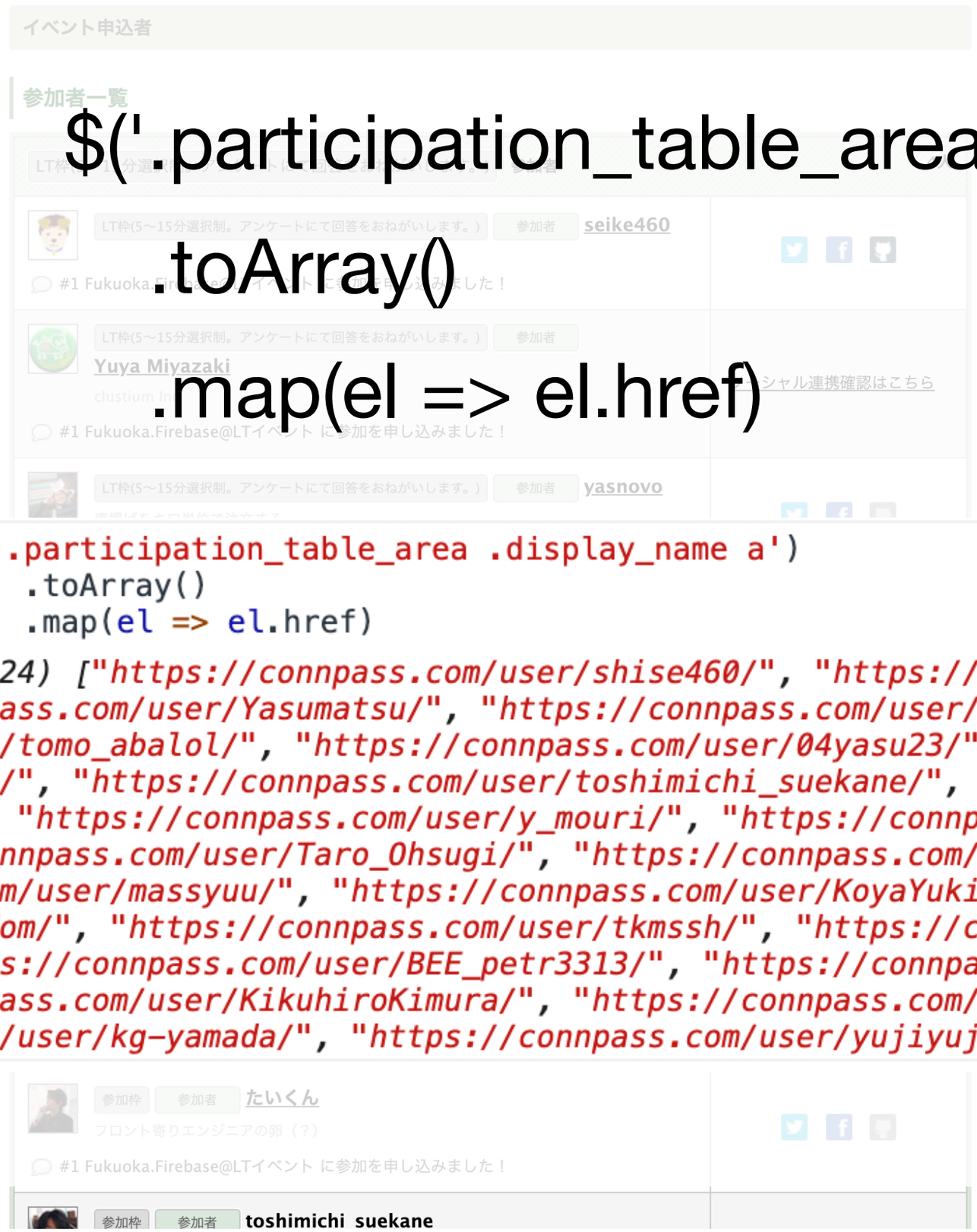
仕様の話とかちょっと被るので...

Vue Night in Fukuoka 参加者

∩

今日の参加者

Connpass 参加者一覧ページ



```
$('.participation_table_area .display_name a')
    .toArray()
    .map(el => el.href)
```

※ユーザー名はユニーク制約ついてないからNG

```
> $('.participation_table_area .display_name a')
    .toArray()
    .map(el => el.href)

<· (24) ["https://connpass.com/user/shise460/", "https://connpass.com/user/38kun/", "https://connpass.com/user/Yasumatsu/", "https://connpass.com/user/lune100106/", "https://connpass.com/user/tomo_abalol/", "https://connpass.com/user/04yasu23/", "https://connpass.com/user/TaikiKihara/", "https://connpass.com/user/toshimichi_suekane/", "https://connpass.com/user/near_future/", "https://connpass.com/user/y_mouri/", "https://connpass.com/user/KazukiNishida/", "https://connpass.com/user/Taro_Ohsugi/", "https://connpass.com/user/karyon_noyer/", "https://connpass.com/user/massyuu/", "https://connpass.com/user/KoyaYuki/", "https://connpass.com/user/earthfreedom/", "https://connpass.com/user/tkmssh/", "https://connpass.com/user/RyuichiSuetsugu/", "https://connpass.com/user/BEE_petr3313/", "https://connpass.com/user/sla10132000/", "https://connpass.com/user/KikuhiroKimura/", "https://connpass.com/user/HayatoKoba/", "https://connpass.com/user/kg-yamada/", "https://connpass.com/user/yujiyujiyujiyuji/"]
```

結果の配列を

const firebaseParticipants

const vueParticipants

に代入。

値が同じ要素の数（つまり両方に参加している人の数）を求める。

```
firebaseParticipants.reduce(
  (count, participant) =>
      vueParticipants.includes(participant)
      ? count + 1 : 0      ←訂正: 最後 0 ではなく count を返すのが正解でした
, 0);
```

結果

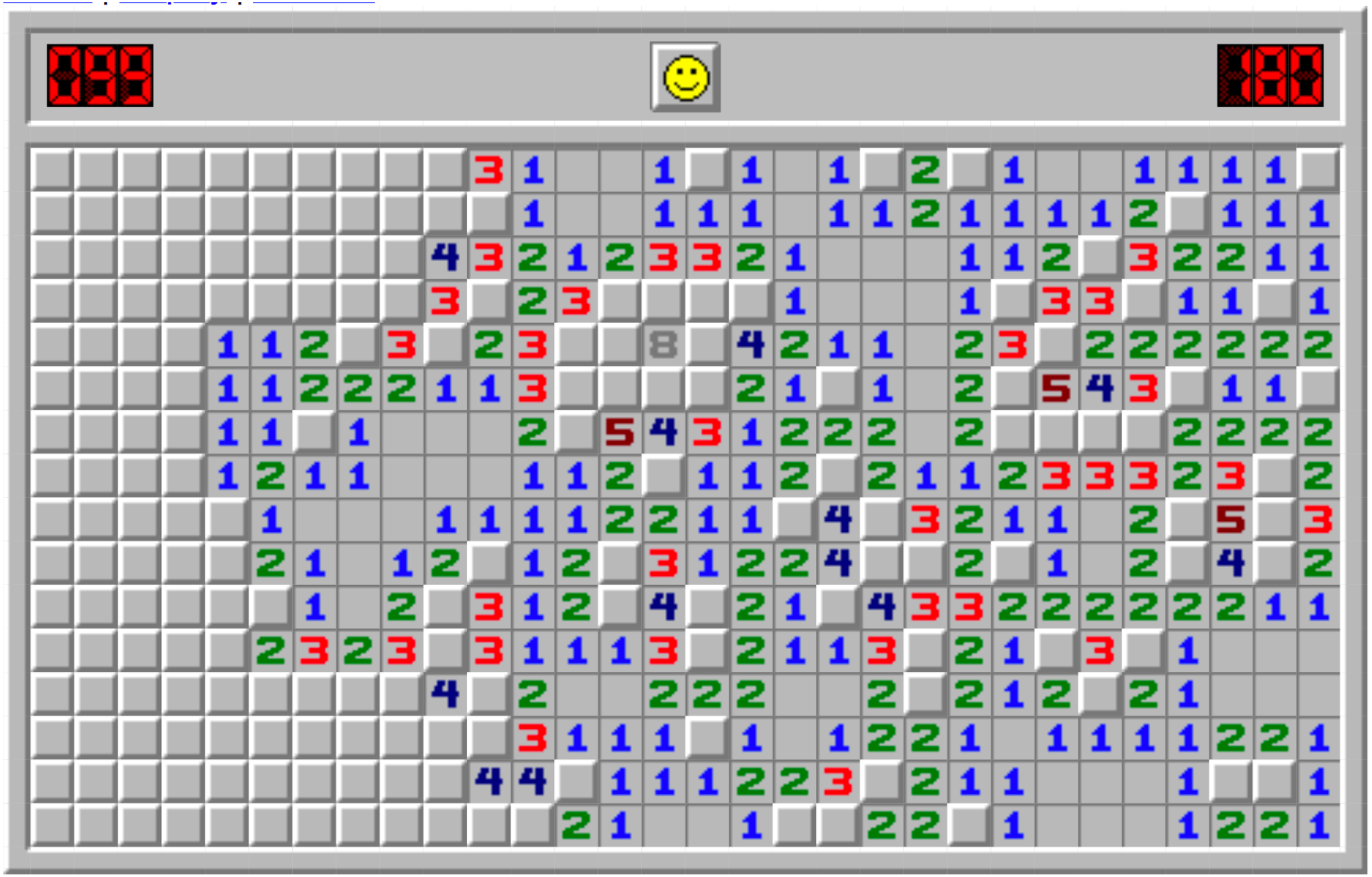
0

訂正: 正しく計算すると11で約半分でしたすみません
（以降のスライドの内容には影響しません）

1. **前回までのあらすじ**
2. 作ったものの紹介（デモ）
3. 基本となるFirebaseの知識
    1. Cloud Functions for Firebase とは
    2. Firebase Cloud Firestore とは
    3. Firebase Hosting とは
4. ゲームの設計

前回の Vue Night では....

1. 前回までのあらすじ
2. **作ったものの紹介**
3. 基本となるFirebaseの知識
    1. Cloud Functions for Firebase とは
    2. Firebase Cloud Firestore とは
    3. Firebase Hosting とは
4. ゲームの設計



※画像はイメージです

要件

- 自分の会社のサイトにミニゲームとしてマインスイーパーを設置する。
- マインスイーパーをクリアするとサイトのコンテンツが閲覧できる
- チートできないようにしたい
- 特定のフレームワークにゲームのコア部分の処理を依存させたくない

Demo Play

https://www.clustium.com/

いただいたフィードバック

クリアしたのに大したコンテンツなくて悲しかった

→すみませんプログラム組むだけで時間なくなりました。

普通にコンテンツを掲載していても読まないので、あえてゲームにしてハードルを設けることで読ませようという目的があるんですよね。よく考えましたね〜

→すみませんそんな考えてないです。

Network Errror が出る

→すみません Cloud Function のプラン上限超えてました。

1. 前回までのあらすじ
2. 作ったものの紹介（デモ）
3. **基本となるFirebaseの知識**
    1. Cloud Functions for Firebase とは
    2. Firebase Cloud Firestore とは
    3. Firebase Hosting とは
4. ゲームの設計

Cloud Functions for Firebase

Cloud Functions for Firebase を使用すると、Firebase 機能や HTTPS リクエストによってトリガーされたイベントに応じて、バックエンド コードを自動的に実行できます。コードは Google のクラウドに保存され、マネージド環境で実行されます。独自のサーバーを管理およびスケーリングする必要はありません。

ー公式ドキュメントより

Cloud Functions for Firebase

もともと別にあった Cloud Functions の Firebase 関連の連携強化版。HTTP アクセスや Firestore上の Storage の更新等をトリガーとして任意の関数を実行できる。

人によっては AWS Lambda と同じようなもの、というとわかりやすい？

Node.js v6 か v8 で実行される（v10 はまだベータ段階）。しばらく放置しているとスリープモードになる。トリガーで再起動するが3秒くらいかかる。

Firebase Cloud Firestore

Cloud Firestore は、Firebase と Google Cloud Platform からのモバイル、ウェブ、サーバー開発に対応した、柔軟でスケーラブルなデータベースです。Firebase Realtime Database と同様に、リアルタイム リスナーを介してクライアント アプリ間でデータを同期し、モバイルとウェブのオフライン サポートを提供します。これにより、ネットワークの遅延やインターネット接続に関係なく機能するレスポンシブ アプリを構築できます。Cloud Firestore は、その他の Firebase および Google Cloud Platform プロダクト（Cloud Functions など）とのシームレスな統合も実現します。

ー公式ドキュメントより

Firebase Cloud Firestore

リアルタイムにサーバー/クライアント間のデータを同期できるスキーマレスなデータベース。

従来の Realtime Database の改良版で、今年になってやっとベータではなく正式版となったので、今後はこちらを使うと良い。

※なぜかドキュメントにはいまだにベータ版と書いてあるが、1月の公式ブログには **is officially out of beta and in General Availability!** と書かれている。

Firebase Hosting

Firebase Hosting は、ウェブアプリ、静的コンテンツ、動的コンテンツ向けの高速で安全性の高いホスティングを提供します。

Firebase Hosting はデベロッパー向けの、本番環境レベルのウェブ コンテンツ ホスティングです。1 つのコマンドですばやく簡単にウェブアプリをデプロイすることができ、静的コンテンツと動的コンテンツの両方をグローバル コンテンツ配信ネットワーク（CDN）に配信できます。

ー公式ドキュメントより

Firebase Hosting

名前の通り、ホスティングサービス。HTMLや画像等の静的リソースを配信してくれる。コマンドひとつでデプロイでき、自動的にCDNに載せてくれるのでページ表示も速い。
Cloud Functions と連携することで動的ページにも対応。

類似のサービスとして Netlify とか GitHub Pages とかがあるが、 Firebaseの他の機能と連携することで動的ページもホスティングできるのが特長（だと思う）。

1. 前回までのあらすじ
2. 作ったものの紹介（デモ）
3. 基本となるFirebaseの知識
    1. Cloud Functions for Firebase とは
    2. Firebase Cloud Firestore とは
    3. Firebase Hosting とは
4. **ゲームの設計**

設計

Nuxt.js
(Vue.js)

Cloud Functions

Cloud Firestore

Hosting

ユーザーの操作のみを起点として処理が動く

昔ながらのAPI設計のイメージ

ちなみに...

別の設計案

Firebase Cloud Functions は Cloud Firestore のデータ更新をトリガーとして 実行することも可能なので、接続先を Cloud Firestore のほうのみに限定し、データ更新されたら裏で Functions で処理して Firestore に保存し、リアルタイム反映するという設計もあり得る

そのほうがレイテンシーが向上するかもしれない（未検証）。

マルチプレイなどでリモートとのステートのリアルタイム同期が必要となったら Firestore との直接接続をすべき。その場合は Functions とも接続するのは複雑になるので、ブラウザから Functions へのHTTPアクセスはしない想定が良いと思う。

※このページの画像は都合により発表時から差し替えられました。

別の設計案のイメージ

DBの状態が（ほぼ）リアルタイムで同期される

まあとにかく今回はブラウザの接続先は

Cloud Functions による API にした

API エンドポイント

/game/init
ユーザーデータ（解放済みのコンテンツ情報など）を作成

/game/restore/:playerId
ユーザーデータを取得（再訪問時用）

/game/start
新しいステージを開始

/game/open
マスを開く

/article/:id
コンテンツを取得

主なDBスキーマ

articles/:article_id
title: string
body: string

players/:player_id
availableArticles: map
maskedMap: string　　←　ユーザーに見えているゲームの状態
rawMap: string　←　すべての地雷位置を含むゲームの状態
seed: number

/stages/:stage_id
articleId: string　　マップサイズと地雷数が変わる。
level: number　←　いまのところ 1 (EASY) のみ

順番にゲームロジックを見てみる

1. 初期化

- サイトへの初回アクセス時にプレイヤーの初期データを生成。
- localStorage にプレイヤーIDを保存し、以後そのIDを使ってコンテンツの解放情報を取得する。

順番にゲームロジックを見てみる

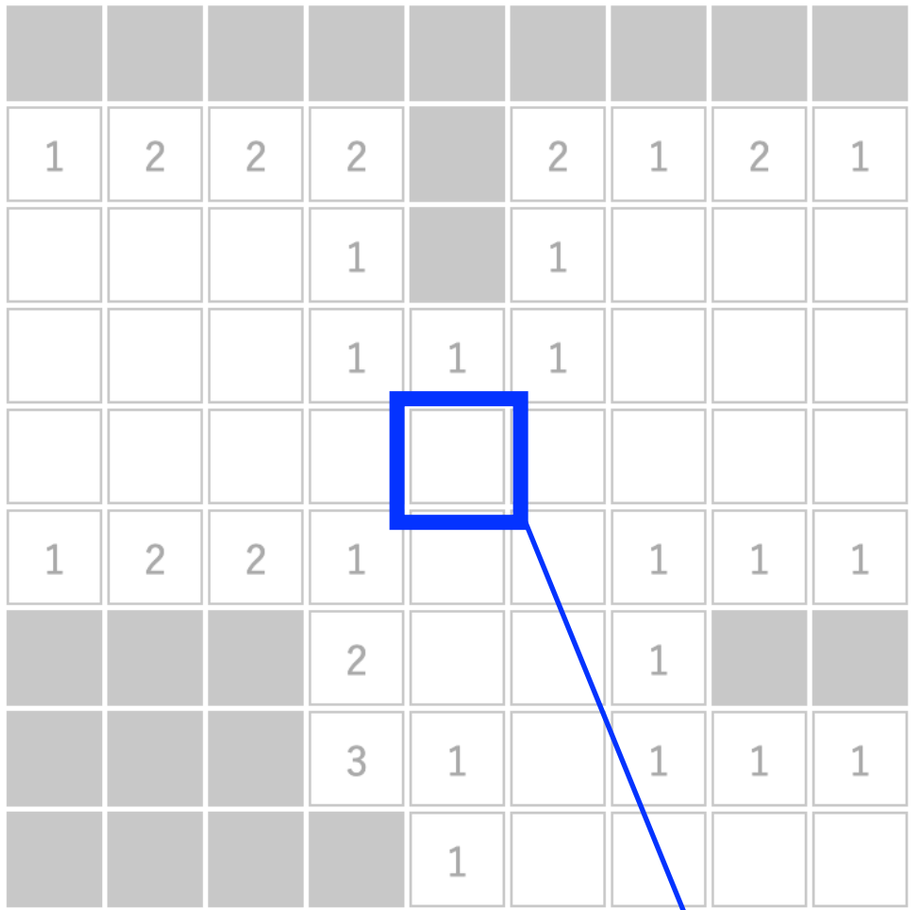
2. ステージ開始

https://www.clustium.com/game/about

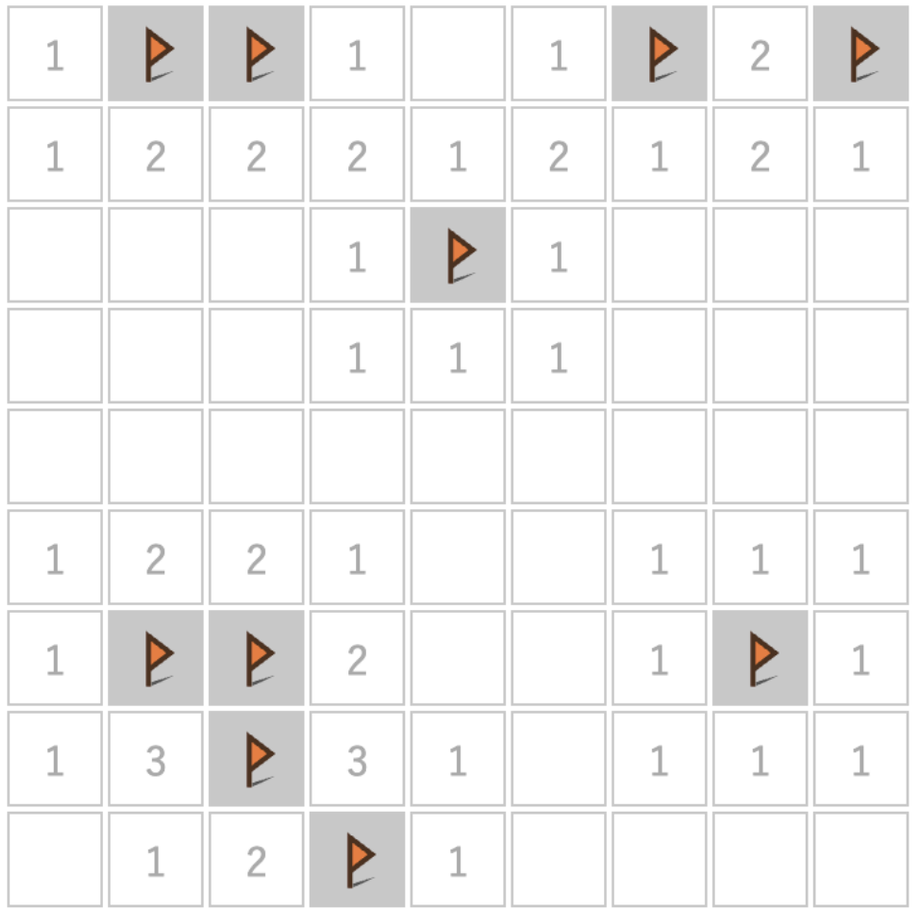
- URLからステージID（**about**）が決まる。最初にマスをクリックすると、ステージIDとクリックされたマスの座標を含めてゲーム開始API（game/start）を呼び出す。
- APIは Firestore から該当ステージの難易度を取得し、マップサイズを決定。その上で乱数を使って地雷位置を確定する。このとき最初にクリックされたマスとその周囲には地雷は配置されない。
- クリックされたマスの周囲には地雷がないので、マスの表示は「0」となるが、「0」のマスが展開されるときは（初回に限らず）周囲8マスも同時に展開されるので、その処理を行う。

ゲーム開始直後の状態

maskedMap



rawMap



最初にクリックされたマス

順番にゲームロジックを見てみる

3. ゲーム進行

- ゲーム開始APIが maskedMap を返すので、その状態を Vue.js のステートとして保持し、コンポーネントに反映。
- 新しいマスを開くときはマス展開API（game/open）に座標を送る。
- 展開時に地雷があるマスを開いたらゲームオーバー。
- 展開の結果、すべての地雷のないマスが展開されたらゲームクリア。
- ちなみに右クリック（長押し）で立てる旗についてはサーバーとの通信はしていない。Vue.js のステートとして持っているだけ。

players > 2Q8VOtMciIWX...

clustium-web-ver3

+ Add collection

articles

players >

stages

players

+ Add document

0tMMe[redacted]

2Q8VO

3KyI4

3a6UU

3aVRX

4q9r3

56GFo

58oxn

5FMVt

6oDAA

7DkaN

87Avb

8WA2R

9GCyC

9cyxY

9dtY0

9gbv5

9jnLf

AaYwz

DjUlM

FEyBE

G784o

2Q8VOtMciIWXkSGXToFs

+ Add collection

+ Add field

availableArticles

about: [redacted]

createdAt: 1557833539882

currentStageId: "about"

firstX: 6

firstY: 3

gameStartedAt: 1557833946756

lastLoginAt: *null*

maskedMap: "[[1,1,1,0,1,2,2,1,0],[1,-3,1,0,1,-3,-3,1,0],
[2,3,3,1,1,2,2,1,0],[2,-3,-3,2,0,0,0,1,1],
[-3,4,-3,2,0,0,0,1,-3],[1,2,1,1,0,0,1,2,2],
[1,1,1,0,0,0,1,-3,1],[1,-3,1,0,0,0,1,1,1],
[1,1,1,0,0,0,0,0,0]]"

rawMap: "[[1,1,1,0,1,2,2,1,0],[1,-2,1,0,1,-2,-2,1,0],[2,3,3,1,1,2,2,1,0],
[2,-2,-2,2,0,0,0,1,1],[-2,4,-2,2,0,0,0,1,-2],[1,2,1,1,0,0,1,2,2],
[1,1,1,0,0,0,1,-2,1],[1,-2,1,0,0,0,1,1,1],[1,1,1,0,0,0,0,0,0]]"

seed: 0.8359016808135757

updatedAt: 1557834107753

順番にゲームロジックを見てみる

4. ゲームクリア

- 展開の結果、すべての地雷のないマスが展開されたらゲームクリア。
- クリアしたステージに対応するコンテンツID（articleId）をプレイヤーデータとして保存する。
- https://www.clustium.com/article/:articleId でコンテンツにアクセスできる。articleId がわかれば誰でも該当コンテンツにアクセスできるので、URLをコピペすればゲームクリアしていない人でもアクセス可能。

主なDBスキーマ

articles/:article_id
title: string
body: string

players/:player_id
availableArticles: map
maskedMap: string
rawMap: string
seed: number

/stages/:stage_id
articleId: string
level: number

（参考）Firebase Functions を North-East （東京）リージョンで試したときのレスポンス取得にかかる時間

- Firestore からデータ取得するのみのAPI(article)
  → 100ms 前後
- Firestore への書き込み等のあるAPI(start, open)
  → 200ms 前後
- 上記に加えて Preflight リクエストで 30ms 〜 100ms 程度
- インスタンス起動が必要となる場合はプラス数秒

終

すみませんコンテンツは作成中です。