

Firestore

Contents

- 関数
 - 関数の定義
 - 引数と戻り値
 - 関数の練習(乱数の生成)
- 自作チャットの作成
 - Firebaseの準備
 - チャット作成の準備
 - チャット処理と画面の作成
- 課題発表→P2Pタイム

rules...

- 授業中は常にエディタを起動！
- 考えたことや感じたことはslackのガヤチャンネルでガンガン発信！
- 質問はslackへ！ 他の人の質問にも目を通そう！（同じ質問があるかも）
- 演習時，できた人はスクショなどslackに貼ってアウトプット！
- まずは打ち間違いを疑おう！
 - `{'";` など
- 書いたら保存しよう！（よく忘れる！）
 - `command + s`
 - `ctrl + s`

本日のゴール

- オンラインでデータを扱う！
- リアルタイムでデータ共有する！
- Firebaseの癖を把握する！

関数 (function)

関数とは？？

```
// 関数 (function)
//   - 関数とは記述した処理をまとめて名前をつけて使い回せるようにしたもの.
//   - 一度処理を定義してしまえば、呼び出すだけで実行可能！

// 例
function test(){
  console.log('関数は便利！');
}
test();
```

// 関数の定義の仕方は決まっている
// 「{ }」内に実行したい処理を記述

// 関数の実行 (console.log(); が実行される)

関数は呼び出さないと実行されない！
定義するだけではNG！

引数と戻り値

- 引数(関数に入力する値)
 - 定義した関数に対して, 処理に必要な値を入力する.
 - 引数の数は一つでも複数でもOK!
- 戻り値(関数から出力されてくる値)
 - 関数の中で計算などを実行した後, 結果を返す処理.
 - 関数内の変数, 配列, オブジェクトなどで返せる.

引数と戻り値

// 関数の定義

```
function add(a, b){  
  const total = a + b;  
  return total;  
}
```

// aとbが引数

// totalが戻り値

// 関数の実行

```
const sum = add(10, 20);  
console.log(sum);
```

// 30が表示される

10と20を入力すると30が返ってくる

プログラミングの関数 = 数学の関数

例: $f(x) = x^2 + 2x + 1$ の関数を定義すると...

$$f(2) = 9,$$

$$f(5) = 36,$$

$$f(10) = 121$$



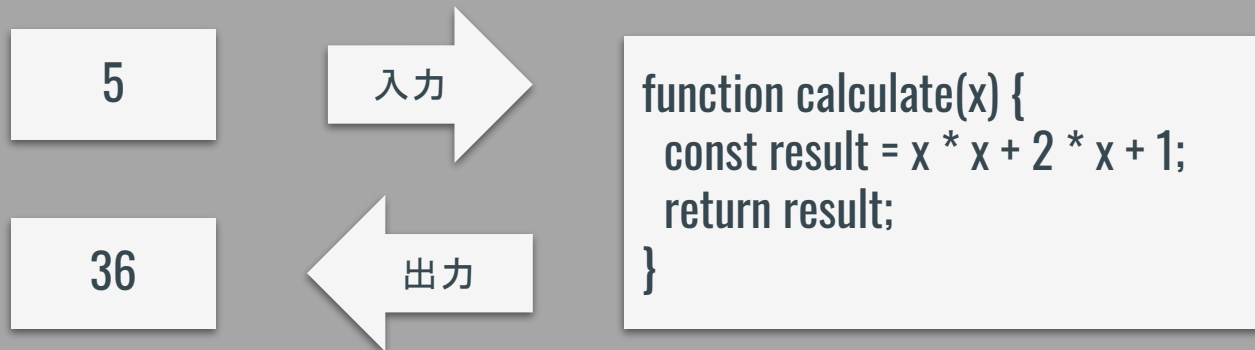
プログラミングの関数 = 数学の関数

例: JavaScriptで書くと...

`calculate(2);` `// 9`

`calculate(5);` `// 36`

`calculate(10);` `// 121`



引数と戻り値がない場合もある(ややこしいポイント)

これまでに登場していた関数

```
// 乱数関連も関数（最初から用意されている関数）  
  
Math.random();           // 0.534714863872  
    // 引数： なし  
    // 戻り値： 0.534714863872  
  
Math.floor(3.1415926535); // 3  
    // 引数： 3.1415926535  
    // 戻り値： 3
```

【おまけ】関数の書き方

```
// 関数の記述方法（関数内の処理は同一）  
function add1(a, b){  
  return a + b;  
}  
  
const add2 = function(a, b){  
  return a + b;  
}  
  
const add3 = (a, b) => {  
  return a + b;  
}
```

全部(大体)同じ！
add(10, 20);で実行！！

関数の利用

関数の使いどころ

- 関数の利点
 - イベントごとに毎回同じ処理を書くのは面倒！
 - 関数を定義しておけば、ボタン押したら実行するだけ！
- 例
 - 押したボタンに応じて、異なる範囲の乱数を発生させたい！

関数の使いどころ

// 関数の定義

```
function generateRandomNumber(min, max){  
  const rand = Math.floor(Math.random() * (max - min + 1) + min);  
  return rand;  
}
```

// 実行するときはこんな感じ

```
const result = generateRandomNumber(1, 9);    // 1から9までの乱数  
console.log(result);
```

関数の使いどころ

```
// ボタンをクリックしたイベントで関数を実行
$('#btn01').on('click', function () {
  const result = generateRandomNumber(1, 10);
  $('#echo').text(result);
});
```

ボタンごとに範囲を設定して実行

```
// ※btn02, btn03も同様
// 【参考】janken.htmlに関数を使用したじゃんけんの例もあります！
```


関数の練習

- 関数の応用
 - 最小値と最大値を入力してランダムな数を返す関数を定義しよう！
 - 各ボタンのクリック時に関数を実行し、結果を#echoに出力しよう！

自作チャットの実装



firebase

Webブラウザでチャット

realtime chat

user:

send

1
2019-06-07T18:24:21
11111111

2
2019-06-07T18:24:10
2222

1
2019-06-07T18:23:48
1111

realtime chat

send

1
2019-06-07T18:24:21
11111111

2
2019-06-07T18:24:10
2222

1
2019-06-07T18:23:48
1111

1. 片方で送信すると...

2. 両方で表示される！

Firestore(cloud firestore)とは？？

Firestoreは、クライアントからアクセス可能なデータベースとしてFirestore Realtime Database(以下 Realtime Database)とCloud Firestoreの2つを用意しています。

Realtime Databaseは、リアルタイムでクライアント全体の状態を同期させる必要があるモバイルアプリ向けの効率的で低レイテンシなものです。

Realtime Databaseはクラウド上でホスティングされるNoSQLのデータベースです。データはすべてのクライアントにわたってリアルタイムに同期され、アプリがオフラインになっても利用可能です。クロスプラットフォームアプリを構築した場合でも、すべてのクライアントが1つのRealtime Databaseを共有して、最新のデータへの更新を自動的に行います。またクライアントからも直接アクセスが可能なため自前のサーバなしで使えるデータベースとしても活用できます。

Cloud Firestoreは、直感的な新しいデータモデルで、Realtime Databaseの性能をさらに向上しており、Realtime Databaseよりも豊かで高速なクエリとスケールを備えています。Cloud Firestoreは2017年のGoogle I/Oで発表されたプロダクトであり、2018年5月現在はベータ版リリースです。

引用: WEB+DB PRESS vol.105 第4章(※2019年2月より正式版として運用されています。)

Firestore (cloud firestore)とは？

- サーバ上にデータを保存できる！
- 保存したデータをリアルタイムに同期できる！
- 異なるデバイスでもデータを共有可能！
 - PCとスマホでリアルタイムにデータを同期できる.
- JavaScriptのみで実装可能！
 - Swift, Go, Pythonなど他の言語でも使用可！

サーバにデータを保存する



データ送信



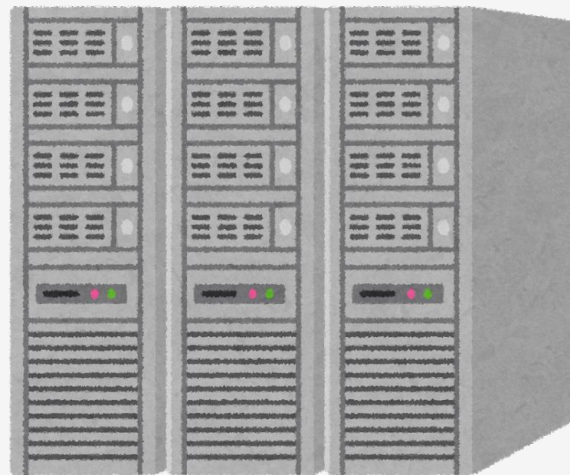
サーバにデータを保存する



①データ追加

②ブラウザに反映

Google



サーバにデータを保存する



①データ追加

②全てに反映

②全てに反映

Google



Firestoreの準備が必要！

準備の流れ

- ①ログイン
- ②プロジェクトの作成
- ③権限の設定
- ④データベースの準備

<https://firebase.google.com/>にアクセス！

The image shows a screenshot of the Firebase website. At the top, there is a navigation bar with the Firebase logo and links for 'プロダクト' (Products), '使用例' (Use Cases), '料金' (Pricing), 'ドキュメント' (Documentation), and 'サポート' (Support). A search bar with the text '検索' (Search) is also present. To the right of the search bar are links for 'コンソールへ移動' (Move to Console) and 'ログイン' (Login). Below the navigation bar, there is a dark blue banner for 'Firebase Crashlytics' with the text 'Prioritize and fix issues with powerful, realtime crash reporting.' Below this banner, there is a large yellow and orange graphic featuring a smartphone with various app icons floating above it. Two white callout boxes with black text are overlaid on the right side of the page: '②コンソール画面へ移動' (Move to Console screen) and '①ログイン' (Login). Below the graphic, the text 'Firebase helps mobile app teams succeed.' is displayed. At the bottom left, there are two buttons: 'スタートガイド' (Get Started) and '動画を見る' (Watch Video).

②コンソール画面へ移動

①ログイン

Firebase helps mobile app teams succeed.

スタートガイド

動画を見る

新規プロジェクトの作成



ドキュメントに移動



Firebase へようこそ

優れたアプリの開発、ユーザーとの交流、モバイル広告からの収益向上に役立つ Google のツールを利用できます。

[🔍 詳細](#) [☰ ドキュメント](#) [🗉 サポート](#)

最近のプロジェクト



プロジェクトを追加



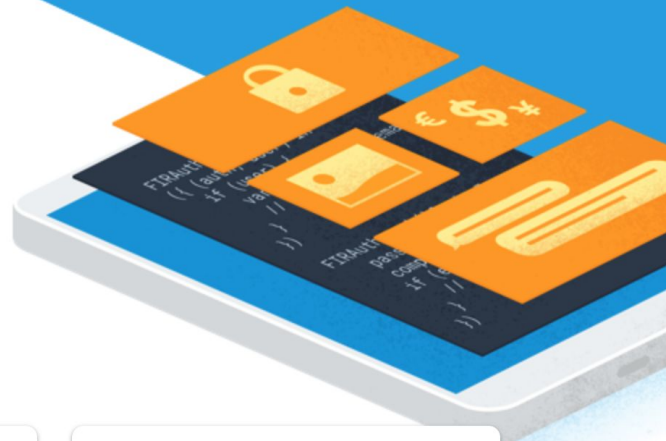
デモ プロジェクトを見る

FireTrip

firetrip-813c6

web-db-test

web-db-test-3fd6d



新規プロジェクトの作成

× プロジェクトの作成 (手順 1/3)

まず

プロジェクトに名前をつけましょう[?]

プロジェクト名

chatapp

✎ chatapp-a0fd6

続行

プロジェクト名 : chatapp+受講番号2桁

続行をクリック

新規プロジェクトの作成

× プロジェクトの作成 (手順 2/2)

Google アナリティクスは無料かつ無制限のアナリティクス ソリューションで、Firebase Crashlytics、Cloud Messaging、アプリ内メッセージング、Remote Config、A/B Testing、Predictions、Cloud Functions で、ターゲティングやレポートなどが可能になります。

Google アナリティクスにより、以下の機能が有効になります。

- × A/Bテスト ⑦
- × Firebase プロダクト全体でのユーザーセグメンテーションとターゲティング ⑦
- × ユーザー行動の予測 ⑦
- × クラッシュに遭遇していないユーザー ⑦
- × イベントベースの Cloud Functions トリガー ⑦
- × 無料で無制限のレポート ⑦

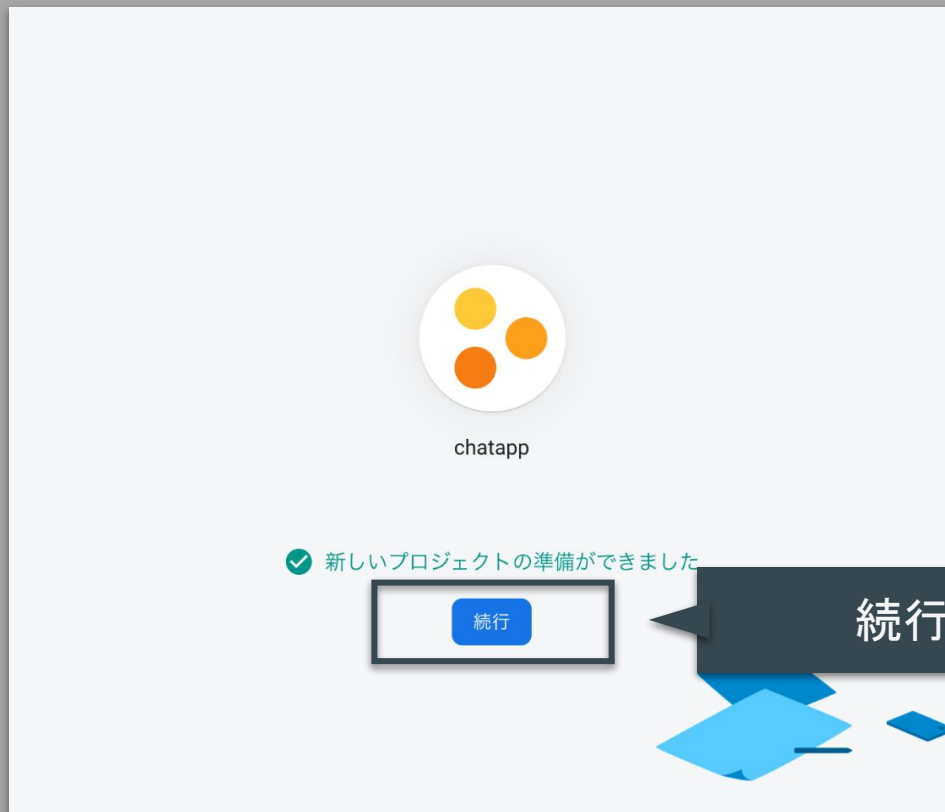
☐ このプロジェクトで Google アナリティクスを有効にする
推奨

無効 !

クリック !!

プロジェクトを作成

新規プロジェクトの作成



Webアプリを追加

Firebase

Project Overview

開発

- Authentication
- Database
- Storage
- Hosting
- Functions
- ML Kit

品質

Crashlytics, Performance, Test Lab

アナリティクス

Dashboard, Events, Conversions, A...

拡大

Predictions, A/B Testing, Cloud M...

chatapp

Spark プラン

ドキュメントに移動

アプリに Firebase を追加して利用を開始しましょう

ほとんどの Firebase 機能と、iOS と Android アプリ用のアナリティクスが含まれた Core SDK をインストールしてください

iOS Android </>

開始するにはアプリを追加してください

適当に設定！！（プロジェクト名と一緒にわかりやすい）

× ウェブアプリに Firebase を追加

1 アプリの登録

アプリのニックネーム [?](#)

chatapp

☐ このアプリの **Firebase Hosting** も設定します。 [詳細](#) [?](#)

Hosting は後で設定することもできます。いつでも無料で始めることができます。

アプリを登録

2 Firebase SDK の追加

必要なコードが表示されるのでコピー

☐ npm を使用する ② ☒ <script> タグを使用する ①

① こちらを選択！！

これらのスクリプトをコピーして <body> タグの下部に貼り付けます前に行ってください。

```
<script type="module">
// Import the functions you need from the SDKs you need
import { initializeApp } from "https://www.gstatic.com/firebasejs/9.0.2/firebase-app.js";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyAfIXSG63mRjrfnWb2pSHH-tCUTPptP-Qw",
  authDomain: "test-466a4.firebaseio.com",
  projectId: "test-466a4",
  storageBucket: "test-466a4.appspot.com",
  messagingSenderId: "1012034593201",
  appId: "1:1012034593201:web:9d9dbbd31cf63275761d90"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
</script>
```

② コピー！！

npm とバンドラ（webpack や Rollup など）を使用している場合は、[モジュール SDK](#) の利用を検討してください。

ウェブ向け Firebase の詳細については、[こちら](#)をご覧ください：[使ってみる](#)、[ウェブ SDK API リファレンス](#)、[サンプル](#)

コンソールに進む

エディタで貼り付け

17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36

```
<script type="module">
  // Import the functions you need from the SDKs you need
  import { initializeApp } from "https://www.gstatic.com/firebasejs/9.0.2/firebase-app.js";
  // TODO: Add SDKs for Firebase products that you want to use
  // https://firebase.google.com/docs/web/setup#available-libraries

  // Your web app's Firebase configuration
  const firebaseConfig = {
    apiKey: "AIzaSyAfiXSG63m
    authDomain: "test-466a4.
    projectId: "test-466a4",
    storageBucket: "test-466
    messagingSenderId: "1012
    appId: "1:1012034593201:
  };

  // Initialize Firebase
  const app = initializeApp(firebaseConfig);
</script>
```



ブラウザに戻ってコンソールに進む

☐ npm を使用する ② ☒ <script> タグを使用する ②

これらのスクリプトをコピーして <body> タグの下部に貼り付けます。この作業は Firebase サービスを使用する前に行ってください。

```
<script type="module">
  // Import the functions you need from the SDKs you need
  import { initializeApp } from "https://www.gstatic.com/firebasejs/9.0.2/firebase-app.js";
  // TODO: Add SDKs for Firebase products that you want to use
  // https://firebase.google.com/docs/web/setup#available-libraries

  // Your web app's Firebase configuration
  const firebaseConfig = {
    apiKey: "AIzaSyAfIXSG63mRjrfnWb2pSHH-tCUTPptP-Qw",
    authDomain: "test-466a4.firebaseio.com",
    projectId: "test-466a4",
    storageBucket: "test-466a4.appspot.com",
    messagingSenderId: "1012034593201",
    appId: "1:1012034593201:web:9d9dbbd31cf63275761d90"
  };

  // Initialize Firebase
  const app = initializeApp(firebaseConfig);
</script>
```

npm とバンドラ（webpack や Rollup など）を使用している場合は、[モジュラー SDK](#) の利用を検討してください。

ウェブ向け Firebase の詳細については、[こちら](#)をご覧ください：[使ってみる](#)、[ウェブ SDK API リファレンス](#)、[サンプル](#)

コンソールに進む

データベースの準備



「Firestore Database」→「データベースの作成」

データベース (CloudFirestore) の準備

「テストモードで開始」→「有効にする」

Realtime Database のセキュリティ ルール

データ構造の定義後に、データを保護するルールを作成する必要があります。
[詳細](#)

☐ ロックモードで開始
読み取りと書き込みをすべて拒否し、
データベースを非公開で作成します

☒ テストモードで開始
読み取りと書き込みをすべて許可し、
設定をすばやく行います

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

データベース参照を所有しているユーザー
でも、データベースの読み取りや書
き込みはできません。

キャンセル 有効にする

データベース (CloudFirestore) の準備

コレクションの開始

- 1 コレクションに ID を付与する
- 2 最初のドキュメントの追加

親パス

「コレクションを追加」→「chat + 受講番号2桁」を作成

コレクション ID 

chat

キャンセル

次へ

データベース(CloudFirestore)の準備

IDは「自動ID」

ドキュメント ID ⓘ

b30EzdzikZnA8bKe2FBi

フィールド		タイプ	値
name	=	string	test
text	=	string	wwwwww
time	=	timestamp	

日付

2019/12/20

時刻

00:00:00

+ フィールドを追加

キャンセル 保存

- name(string)
 - text(string)
 - time(timestamp)
- の3項目を設定！
(値は適当でOK！)

データベース (CloudFirestore) のイメージ

<https://firebase.google.com/docs/firestore/data-model?hl=ja>



チャットの実装

必要な処理一覧

- チャット画面の作成
 - チャットを入力&表示する画面の作成
- データ送信の処理
 - 入力して送信ボタンを押下したらイベント発火.
 - 入力内容を取得する.
 - firebaseにデータを送信. 送信後に入力欄を空にする.
- データ受信処理
 - データ追加時に自動的にデータ取得する.
 - 受信したデータをブラウザ上に表示する.

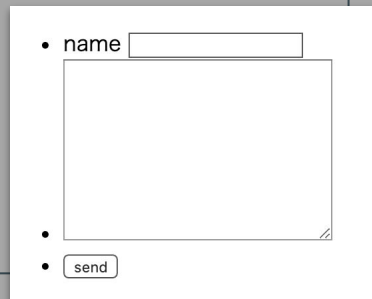
チャット画面の作成

チャット画面の作成

// 入力フォームの作成

```
<ul>
  <li>
    <label for="name">name</label>
    <input type="text" id="name">
  </li>
  <li>
    <textarea name="" id="text" cols="30" rows="10"></textarea>
  </li>
  <li>
    <button id="send">send</button>
  </li>
</ul>
```

こんな感じ！



- name
-
-

データ送信処理の実装

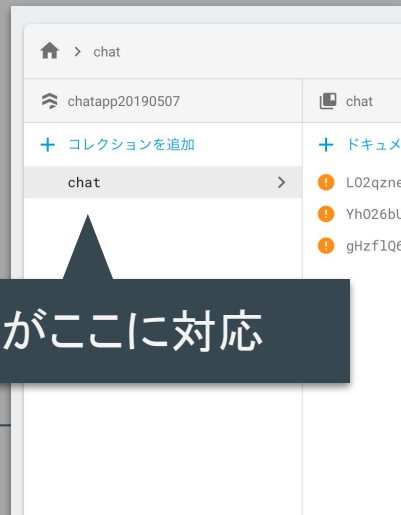
```
import { initializeApp } from "https://...";

// ↓追加
import { getFirestore, collection, addDoc, serverTimestamp, } from
"https://www.gstatic.com/firebasejs/9.0.2/firebase-firestore.js";

const firebaseConfig = {...};

const app = initializeApp(firebaseConfig);

// ↓追加
const db = getFirestore(app);
```



送信ボタンクリック時の処理

送信ボタン	#send
テキストボックス	#name
テキストエリア	#text
メッセージ表示領域	#output

• name

send

```
// 送信ボタンクリックでメッセージ送信
$('#send').on('click', function(){

});
```


データ送信処理の流れ

やること

- 送信ボタンをクリックしたら...
 - (`.on('click', function(){})...`の形)
- 名前と本文を取得.
 - (`.val()`とか)
- `firebase`にデータ(名前, 時間, 本文)を送信.
 - (`addDoc()`の処理を使う!)
- 本文入力用の`textarea`を空にする.
 - (`.val("")`とか)

データ送信処理のコード

```
// 送信処理の記述
// `db`が送信先 送信データはオブジェクトの形
addDoc(collection(db, 'chat'), {
  name: $('#name').val(),      // inputの入力値
  text: $('#text').val(),      // textareaの入力値
  time: serverTimestamp(),     // 登録日時
});

// 送信後にtextareaを空にする処理
$('#text').val('');
```

データ送信処理の動作確認

chatapp20190507	chat	L02qznemG9iUVsf0qv3P
+ コレクションを追加	+ ドキュメントを追加	+ コレクションを追加
chat >	<div>L02qznemG9iUVsf0qv3P ></div> <div>Yh026bUB1aWGUYToo899</div> <div>qHzf106...GGqF8AzJH</div>	+ フィールドを追加
		インターネット上の誰でもこのドキュメントの読み取りと書き込みができます
		user: "1"

送信が成功するとここに表示される！
(ランダムな文字列のidで追加)

データ送信の処理を実装しよう！

- ここまで作ろう！
 - 送信ボタンを押したら入力されたデータを送信！
 - firebaseのコンソール画面で送信されているかどうか確認！
- chatapp.htmlに記述しよう！

データ受信処理の実装

データ受信処理の流れ

やること

- firebaseのデータに変更があったときに...
 - 保存されているデータを新しい順に並び替えて取得.
 - 保存されているデータについて, 1件ずつidとデータを取得.
 - 必要なデータ(idと名前メッセージ時間)のみ入った配列を作成
 - ブラウザに出力するためにデータを適当なタグに入れた配列を作成.
 - 実際にブラウザに表示する.

※ データがわかりにくいので, 都度`console.log()`で確認しよう!

必要な関数を読み込む

```
// ↓一部追加
import { getFirestore, collection, addDoc, serverTimestamp, query,
orderBy, onSnapshot, } from
"https://www.gstatic.com/firebasejs/9.0.2/firebase-firestore.js";

// Firestore関連の処理を実装する場合、使いたい関数を読み込む必要がある。
```

データ受信処理のコード

```
// ↓まずデータ取得の条件を設定
const q = query(collection(db, 'chat'), orderBy('time', 'desc'));
// ↓onSnapshotでcloud firestoreのデータ変更時に実行される！
onSnapshot(q, (querySnapshot) => {
  // querySnapshot.docsにcloud firestoreのデータが配列形式で入る
  const dataArray = [];    // 必要なデータだけが入った新しい配列を作成
  querySnapshot.docs.forEach(function (doc) {
    const data = {
      id: doc.id,
      data: doc.data(),
    }
    dataArray.push(data);
  });
});
```

// 次ページへ続く... (dataArrayをconsoleで確認！)




データ受信処理のコード

```
// ...前ページの続き 「`」で囲んでタグ文字列を表現. 「${ }」で変数を埋め込み
const tagArray = [];      // `dataArray`は前回出てきたオブジェクトの配列
dataArray.forEach(function (data) {
  tagArray.push(`
    <li id=${data.id}>
      <p>${data.data.name}</p>
      <p>${data.data.text}</p>
      <p>${data.data.time.seconds}</p>
    </li>
  `)
})
$('#output').html(tagArray.join(' '));
});
```

↓↓こんな感じで出力↓↓

```
<li id="データのキー名">
  <p>名前</p>
  <p>時間</p>
  <p>本文</p>
</li>
```

データのとり方のイメージ

 chatapp20190507	 doc.id	 L02qznemG9iUVsf0qv3P
+ コレクションを追加	+ ドキュメントを追加	+ コレクションを追加
chat >	<div data-bbox="523 455 1039 514" style="border: 2px solid black; padding: 2px;">! L02qznemG9iUVsf0qv3P ></div> <div data-bbox="556 525 589 558">!</div> Yh026bUB1aWGUYToo899	+ フィールドを追加
	<div data-bbox="556 585 589 618">!</div> gHzf1Q6fXcgGGgE8AzJH	<div data-bbox="1136 563 1168 596">!</div> インターネット上の誰でもこのドキュメントの読み取りと書き込みができます
	<div data-bbox="653 749 1014 858" style="border: 2px solid black; padding: 10px;">doc.data()</div>	<div data-bbox="1054 691 1562 918" style="border: 2px solid black; padding: 10px;">text: "11111111" time: "2019-06-07T18:24:21" user: "1"</div>

やや特殊なデータ構造と扱いのポイント

```
// 配列からのデータ取得方法
// {...}が一つのドキュメントデータ（不要なデータも入っている）
const data = [{...}, {...}, {...}, ...];

// 配列の各要素に対して下記の処理でデータが取れる。
const id = {...}.id;
const data = {...}.data();

// 「必要なデータだけを入れた新しい配列」を作成
const dataArray = [];
dataArray.push({id: id, data: data});      // <- 繰り返し処理で実行
```

データ受信処理の動作確認

realtime chat

• name



•

• test

2020/05/18 23:25:37

www

• test

2020/05/18 23:25:07

wwwwww


• test

2020/05/18 00:00:00

hoge

realtime chat

1. 片方で送信すると...



•

• test

2020/05/18 23:25:37

www

• test

2020/05/18 23:25:07

wwwwww

• test

2020/05/18 00:00:00

hoge

2. 両方で表示される！

データ受信の処理を実装しよう！

- ここまで作ろう！
 - 送信されたデータを画面に表示！
 - 別々のウィンドウで開いてリアルタイムに同期されることを確認！

【おまけ】Enterキーで送信

メッセージ的な操作

```
$('#text').on('keydown', function(e) {  
  console.log(e)  
});
```

keydownイベント

```
m.Event {originalEvent: KeyboardEvent, type: "keydown", isDefaultPrevented: f,  
  timeStamp: 9446.200000005774, jquery11130337889318682532: true, ...}  
  altKey: false  
  bubbles: true  
  cancelable: true  
  char: undefined  
  charCode: 0  
  ctrlKey: false  
  ▶ currentTarget: textarea#text  
    data: undefined  
  ▶ delegateTarget: textarea#text  
    eventPhase: 2  
  ▶ handleObj: {type: "keydown", origType: "keydown", data: undefined, handler: f, gu  
  ▶ isDefaultPrevented: f  
    jquery11130337889318682532: true  
    key: "Enter"  
    keyCode: 13  
    metaKey: false
```

エンターキーのキーコードを確認

(キーコードが13だったら...)

参考情報

- 情報の取得
 - `console.log(e);`を使うとイベントの様々な情報を取得できます.
 - 例えば, `keydown`したキーの番号, クリックした座標, など
- `console.log`を活用していろいろな機能を開発できる!
 - (コナミコマンドとか)
 - 【参考】<https://shgam.hatenadiary.jp/entry/2013/06/27/022956>

課題

Firebaseを使ったアプリケーションを作ろう！

- 最低限ここまで！
 - 「名前」「日時」「メッセージ」を送信&表示
 - 見た目をいい感じに！（LINEやメッセージャーみたいに）
- 追加仕様の例
 - スタンプ送信機能
 - オンラインでじゃんけん
 - MMORPGを開発

※例によってfirebaseを使えば何でもOK！

⚠️⚠️⚠️ 注意点 ⚠️⚠️⚠️

重要！！絶対確認！！

⚠ Githubにpushするときの注意点 ⚠

APIキーの扱い

- FirebaseにはAPIキーが必要になります.
- 誰でも見られるGithubにあげてしまうとあまりよろしくない.

Githubにpushする前に...

- `'git add .'`する前にAPIキー部分は一旦削除しておきましょう.
- 提出フォームのコメント欄にAPIキーを記述してください！

⚠ Githubにpushするときの注意点 ⚠

```
17
18 <script type="module">
19   // Import the functions you need from the SDKs you need
20   import { initializeApp } from "https://www.gstatic.com/firebasejs/9.0.2/firebase-app.js";
21   // TODO: Add SDKs for Firebase products that you want to use
22   // https://firebase.google.com/docs/web/setup#available-libraries
23
24   // Your web app's Firebase configuration
25   const firebaseConfig = {
26     apiKey: "AIzaSyAfIXSG63mRJrfnWb2pSHH-tCUtPptP-Qw",
27     authDomain: "test-466a4.firebaseio.com",
28     projectId: "test-466a4",
29     storageBucket: "test-466a4.appspot.com",
30     messagingSenderId: "1:1012...",
31     appId: "1:1012...",
32   };
33
34   // Initialize Firebase
35   const app = initializeApp(firebaseConfig);
36 </script>
```

⚠ `git add`する前にこの部分を一旦削除
削除したAPIキーは「提出フォームのAPIkey欄」に記述！

締切は厳守！！

P2Pタイム

まずはチーム内で解決を目指す！

訊かれた人は苦し紛れでも応える！！