

firebase



G's ACADEMY
FUKUOKA



アジェンダ

- 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
 - 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

授業のルール

- 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zoomチャットでガンガン発信！
- 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

今日のゴール

- 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(){           // 関数の定義の仕方は決まっている
    console.log('関数は便利！'); // 「{}」内に実行したい処理を記述
}
test(); // 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

- 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

10と20を入力すると
30が返ってくる

// 関数の実行

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

// 30が表示される

プログラミングの関数も数学の関数と同じ

例: $f(x) = x^2 + 2x + 1$ の関数を定義すると...

$$f(2) = 9,$$

$$f(5) = 36,$$

$$f(10) = 121$$



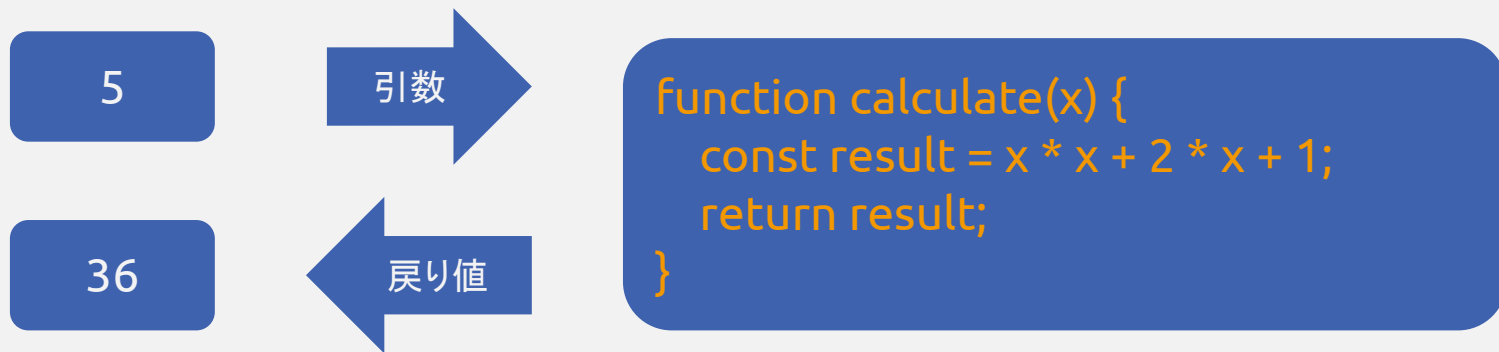
プログラミングの関数も数学の関数と同じ

例: JavaScriptで書くと...

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

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

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



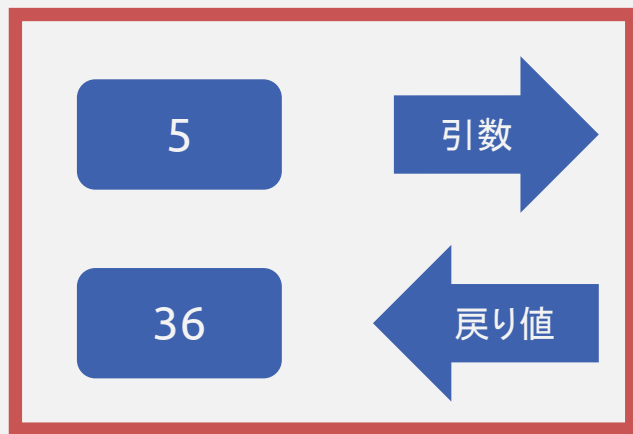
プログラミングの関数も数学の関数と同じ

例: JavaScriptで書くと...

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

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

```
calculate(10);   // 121    ※ 引数と戻り値を設定しない場合もある
```



```
function calculate(x) {  
  const result = x * x + 2 * x + 1;  
  return result;  
}
```

実はこれまでも関数は登場していたっ... !

// 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

関数はいつ使えばいいのか．．??

- 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！

関数はいつ使えばいいのか...??

// 関数の定義

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

最小値と最大値を設定して
乱数を生成

// 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);
```


関数はいつ使えばいいのか...??

```
// 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

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

1. 片方で送信すると...

send

1
2019-06-07T18:24:21

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

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

2. 両方で表示される！

firebase(cloud firestore)とは？？

Firebase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 Firebase 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。)

firebase(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！
- JavaScriptのみで実装可能！

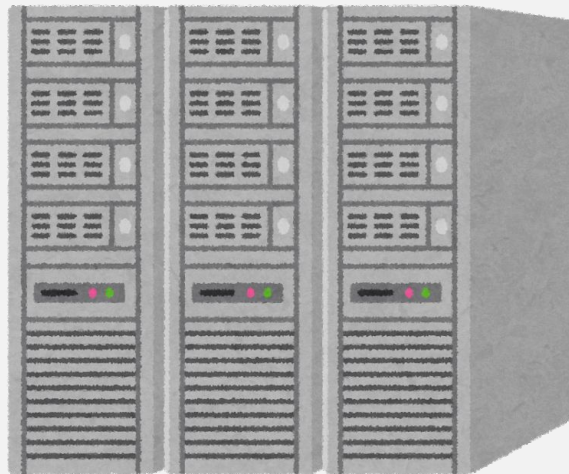
サーバにデータを保存とは??

ブラウザで入力したデータがgoogleのサーバ上に保管される！



データ送信

Google



データをリアルタイムに同期とは？？

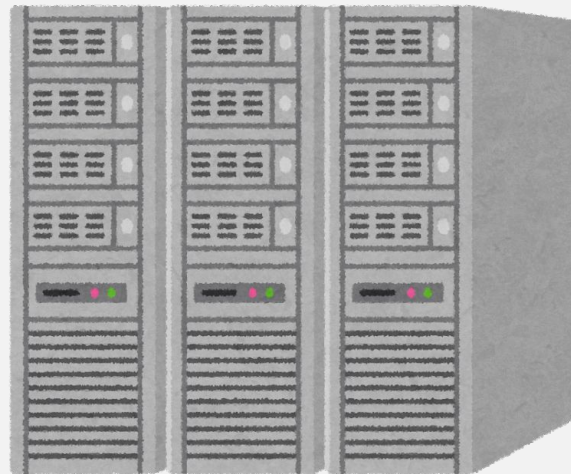
サーバ上のデータが変更されるとブラウザにも反映される！



①データ追加

②ブラウザに反映

Google



異なるデバイスで同期とは??

別のPCで開いているブラウザにも反映されるので同じデータを見られる！



①データ追加



②全てに反映



②全てに反映

Google



準備が必要なので進めよう！

準備の流れ(コードを書く前の準備)

- 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

The screenshot shows the Firebase website's header and main content area. The header includes the Firebase logo and navigation links in Japanese: プロダクト (Products), 使用例 (Use Cases), 料金 (Pricing), ドキュメント (Documentation), サポート (Support), a search bar (検索), and links to the console (コンソールへ移動) and login (ログイン). Below the header, a dark banner for Firebase Crashlytics is visible. The main content area features a large illustration of a smartphone with various app icons. Two blue callout boxes with white text are overlaid on the right side of the image: ②コンソール画面へ移動 (Move to console screen) and ①ログイン (Login). Below the illustration, 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 '動画をみる' (Watch video).

Firebase

プロダクト 使用例 料金 ドキュメント サポート

検索

コンソールへ移動 ログイン

Firebase Crashlytics Prioritize and fix issues with powerful, realtime crash reporting.

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

動画をみる

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](#)

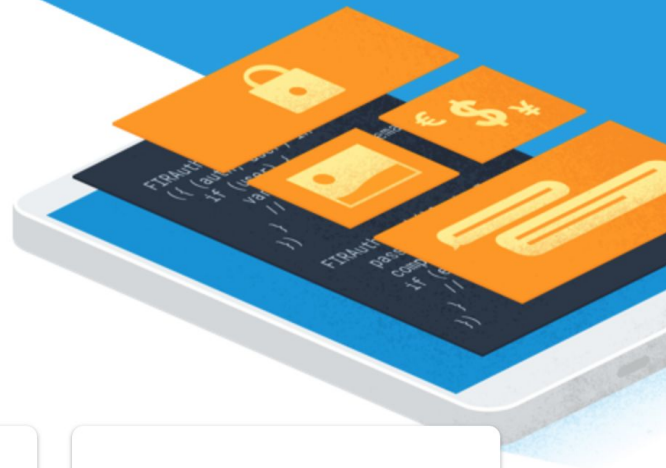
最近のプロジェクト



プロジェクトを追加



デモ プロジェクトを見る



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

プロジェクト名 : chatapp

✎ chatapp-a0fd6

続行

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 ②
- × タラッシュに遭遇していないユーザー ②
- × 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
- × イベントベースの Cloud Functions トリガ ②
- × ユーザー行動の予測 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

どちらでも

クリック！！

プロジェクトを作成

新規プロジェクトの作成



chatapp

✔ 新しいプロジェクトの準備ができました

続行

続行！！！！

webアプリにfirebaseを追加

The screenshot shows the Firebase console interface. On the left is a dark sidebar with the 'Firebase' logo at the top. Below it is a 'Project Overview' section with a home icon and a settings gear icon. Further down are categories: '開発' (Development) with icons for Authentication, Database, Storage, Hosting, Functions, and ML Kit; '品質' (Quality) with 'Crashlytics, Performance, Test Lab'; 'アナリティクス' (Analytics) with 'Dashboard, Events, Conversions, A...'; and '拡大' (Scale) with 'Predictions, A/B Testing, Cloud M...'. The main content area has a blue background with the project name 'chatapp' and a 'Spark プラン' (Spark Plan) button. The central text reads 'アプリに Firebase を追加して利用を開始しましょう' (Add Firebase to your app and get started). Below this, a large blue arrow points to three circular icons: 'iOS', 'Android', and a code icon '</>'. The code icon is highlighted with a red square. Below the icons, the text says '開始するにはアプリを追加してください' (To get started, add an app). At the bottom, there is an illustration of a woman in a yellow shirt and a man in a blue jacket holding a smartphone, both looking at the code icon.

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

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！！（ニックネームは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

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。

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/6.0.0/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#config-web-app -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyB_qDNj1Ae0naKGxr5vBzbcK6kL1tUVY30",
    authDomain: "testapp-279d8.firebaseio.com",
    databaseURL: "https://testapp-279d8.firebaseio.com",
    projectId: "testapp-279d8",
    storageBucket: "testapp-279d8.appspot.com",
    messagingSenderId: "760501105214",
    appId: "1:760501105214:web:af0034bfc6598206"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

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

```

22 <!-- The core Firebase JS SDK is always required and must be listed first -->
23 <script src="https://www.gstatic.com/firebasejs/6.0.0/firebase-app.js"></script>
24
25 <!-- TODO: Add SDKs for Firebase products that you want to use
26 .... https://firebase.google.com/docs/web/setup#config-web-app -->
27
28 <script>
29 .... // Your web app's Firebase configuration
30 .... var firebaseConfig = {
31 ....   apiKey: "AIzaSyB_c",
32 ....   authDomain: "testa",
33 ....   databaseURL: "http",
34 ....   projectId: "testa",
35 ....   storageBucket: "te",
36 ....   messagingSenderId: "1:76050110",
37 ....   appId: "1:76050110",
38 .... };
39 .... // Initialize Firebase
40 .... firebase.initializeApp(firebaseConfig);
41 </script>

```



```
22 <!-- The core Firebase JS SDK is always required and must be listed first -->
23 <script src="https://www.gstatic.com/firebasejs/6.0.0/firebase-app.js"></script>
```

「-app」を削除！！！！

```
24
25 <!-- TODO: Add SDKs for Firebase products that you want to use
26 https://firebase.google.com/docs/web/setup#config-web
```

```
27
28 <script>
29 // Your web app's Firebase configuration
30 var firebaseConfig = {
31   apiKey: "AIzaSyB...",
32   authDomain: "test...",
33   databaseURL: "http...
```

コレやらないと一切動かない！！！！

```
34
35
36
37
38 };
39 // Initialize Firebase
40 firebase.initializeApp(firebaseConfig);
41 </script>
```

ブラウザに戻ってコンソールに進む

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。


```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/6.0.0/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#config-web-app -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyB_qDNj1Ae0naKGxr5vBzbcK6kL1tUVY30",
    authDomain: "testapp-279d8.firebaseio.com",
    databaseURL: "https://testapp-279d8.firebaseio.com",
    projectId: "testapp-279d8",
    storageBucket: "testapp-279d8.appspot.com",
    messagingSenderId: "760501105214",
    appId: "1:760501105214:web:af0034bfc6598206"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

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



The screenshot shows the Firebase console interface. On the left sidebar, the 'Database' option is highlighted with a red box. A purple arrow points from this box to the 'Cloud Firestore' page. The main content area has an orange background and features the text 'Cloud Firestore' and 'クエリと自動スケーリング機能を備えた次世代の Realtime Database'. A button labeled 'データベースの作成' (Create Database) is highlighted with a red box. A magnifying glass is positioned over the right side of the page, focusing on the 'データベースの作成' button.

「Database」→「データベースの作成」

データベースの準備 (cloud firestore)

「テストモードで開始」→「有効にする」

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...

cloud firestore 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
き込みが許可されます。

キャンセル 有効にする

データベースの準備 (cloud firestore)

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」を作成

コレクション ID ?

chat



キャンセル

次へ

データベースの準備 (cloud firestore)

IDは「自動ID」

ドキュメント ID ②

b30EzdziKZnA8bKe2FBi

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

日付

2019/12/20

時刻

00:00:00

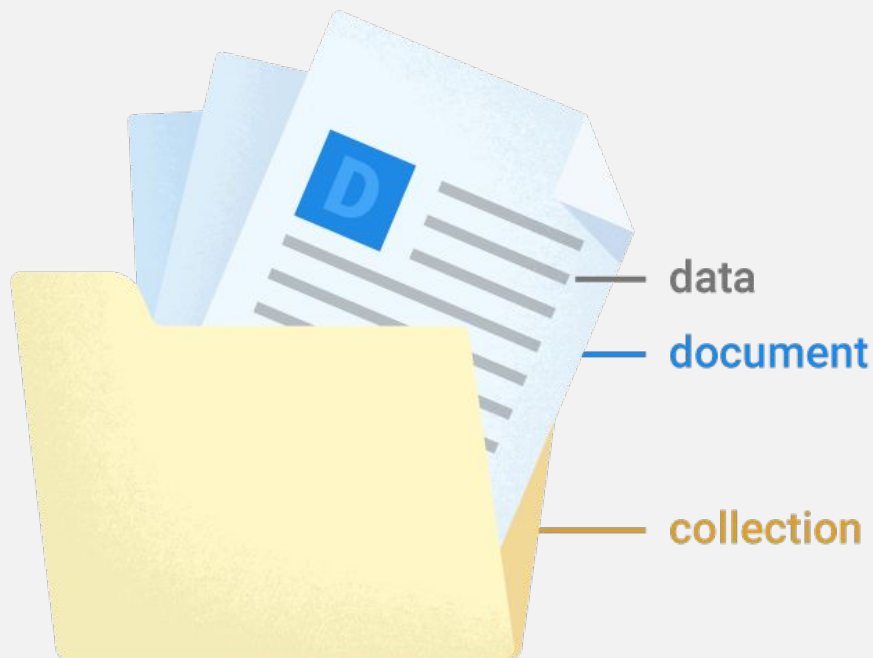
+ フィールドを追加

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！)

データベースのイメージ(**cloud firestore**)

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



チャットの実装

必要な処理

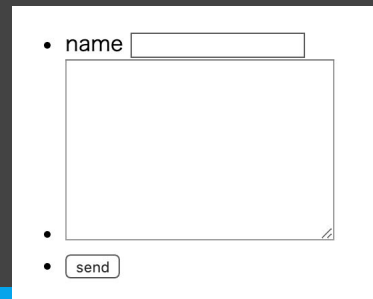
- チャット画面
 - チャットを入力&表示する画面の作成
- データ送信の処理
 - 入力して送信ボタンを押下したらイベント発火.
 - 入力内容を取得.
 - firebaseにデータを送信. 送信後に入力欄を空にする.
- データ受信処理
 - データ追加時に自動的にデータ取得.
 - 受信したデータをブラウザ上に表示.

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！

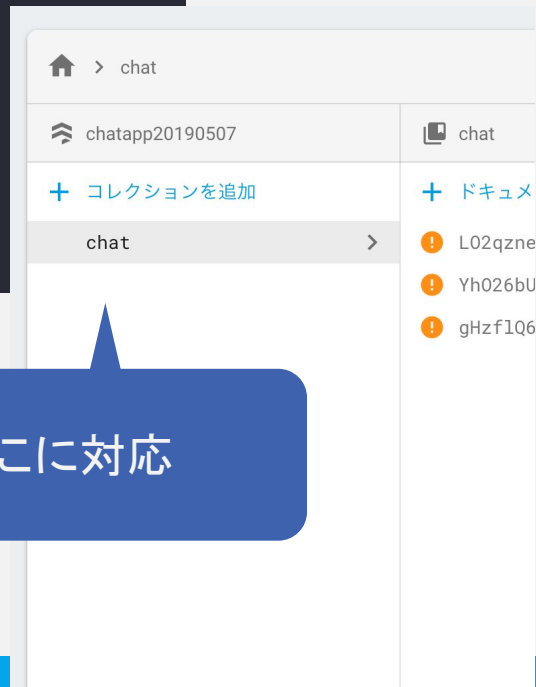


- name
-
-

データ送信処理の作成


```
messagingSenderId: "445936384974",  
appId: "1:445936384974:web:44b5b67ccfd0b39d"  
};  
firebase.initializeApp(firebaseConfig);
```

```
// cloudfirestoreの場所を定義する処理  
var db = firebase.firestore().collection('chat');
```



「db」がここに対応

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(){
    ...
});
```

- 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.

(`db.add()`の処理を使う!)

- 本文入力用のtextareaを空にする.

(`.val('')`とか)

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

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

firebaseのコンソール画面で確認

The screenshot displays the Firebase console interface. On the left, the project 'chatapp20190507' is selected. The main area shows the 'chat' collection. A new document with the ID 'L02qznemG9iUVsf0qv3P' is being added, highlighted by a red box. Below this, other documents are listed with IDs like 'Yh026bUB1aWGUYToo899' and 'qHzf106...'. On the right, the document details for 'L02qznemG9iUVsf0qv3P' are shown, including a warning about public access and a JSON snippet with 'user: "1"'. A blue callout box points to the new document ID.

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

+ フィールドを追加

chat >

! L02qznemG9iUVsf0qv3P >

! Yh026bUB1aWGUYToo899

! qHzf106...GGqF8AzJH

! 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)

- 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！

データ受信処理の作成(ちょっとむずい)

- 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.
 - ブラウザに出力するためにデータを適当なタグに入れる.
 - 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!


```
// 受信処理の記述
db.orderBy('time', 'desc').onSnapshot(function (querySnapshot) {
  // onSnapshotでデータ変更時に実行される！
  // querySnapshot.docsにデータが配列形式で入る
  let str = '';
  querySnapshot.docs.forEach(function (doc) {
    // doc.idでidを, doc.data()でデータを取得できる
    const id = doc.id;
    const data = doc.data();
    const datetime =
convertFromFirestoreTimestampToDatetime(data.time.seconds);
  // 次ページへ続く...
```

```
// ...前ページの続き
str += '<li id="' + id + '>';
str += '<p>' + data.name + '</p>';
str += '<p>' + datetime + '</p>';
str += '<p>' + data.text + '</p>';
str += '</li>';
});
$('#output').html(str);
});
```

//idにkey名を追加

↓↓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

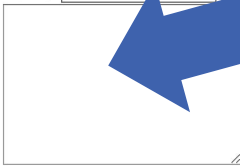
The screenshot displays a web application interface for document management. It features a sidebar on the left with a collection named 'chatapp20190507' and a main content area. The main area shows a list of documents under the heading 'ドキュメントを追加'. The first document in the list is 'L02qznemG9iUVsf0qv3P', which is highlighted with a red box and a blue callout bubble labeled 'doc.id'. Below this list, there is a section titled 'doc.data()' containing a JSON object with the following data:

```
{  "text": "11111111",  "time": "2019-06-07T18:24:21",  "user": "1"}
```

The interface also includes buttons for '+ コレクションを追加' and '+ フィールドを追加', and a warning message indicating that the document is publicly accessible on the internet.

realtime chat

• name



•

• test

2020/05/18 23:25:37

www

• test

2020/05/18 23:25:07

wwwww

• 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

wwwww

• test

2020/05/18 00:00:00

hoge

2. 両方で表示される！

- 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！

Enterキーで送信してみよう！

メッセージャー的な操作！

```
... $('#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...)

- 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
- `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

【課題】firebaseを使ったアプリケーション

- 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！

提出は次回授業前まで！！

やばいいいいい . . .
(` ; ω ; `)

詰んだ... どうしようもない... という方は

写☆経

※写経とは

誰かが書いた動作するコードをひたすら書き写すこと

```
1  <!DOCTYPE html>
2  <html lang="ja">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>chatapp</title>
8      <style>
9          #output li {
10              background: #ccc;
11          }
12      </style>
13  </head>
14
```

```
15 <body>
16   <h1>realtime chat</h1>
17   <ul>
18     <li>
19       <label for="name">name</label>
20       <input type="text" id="name">
21     </li>
22     <li>
23       <textarea name="" id="text" cols="30" rows="10"></textarea>
24     </li>
25     <li>
26       <button id="send">send</button>
27     </li>
28   </ul>
29   <ul id="output"></ul>
30
```


A bottle of Laphroaig 10 Year Old Highland Malt Scotch Whisky stands next to a glass filled with whisky and ice cubes on a bar counter. The bottle is white with black text, and the glass is clear with a dark base. The background is dark and out of focus, showing a bar setting.

APIキーは自分のものを！

```
46 <script>
47   function convertFromFirestoreTimestampToDatetime(timestamp) {
48     const _d = timestamp ? new Date(timestamp * 1000) : new Date();
49     const Y = _d.getFullYear();
50     const m = (_d.getMonth() + 1).toString().padStart(2, '0');
51     const d = _d.getDate().toString().padStart(2, '0');
52     const H = _d.getHours().toString().padStart(2, '0');
53     const i = _d.getMinutes().toString().padStart(2, '0');
54     const s = _d.getSeconds().toString().padStart(2, '0');
55     return `${Y}/${m}/${d} ${H}:${i}:${s}`;
56   }
57
```

```
58     .. .. const db = firebase.firestore().collection('chat');
59     .. .. $('#send').on('click', function () {
60     .. .. .. db.add({
61     .. .. .. .. name: $('#name').val(),
62     .. .. .. .. text: $('#text').val(),
63     .. .. .. .. time: firebase.firestore.FieldValue.serverTimestamp(),
64     .. .. .. });
65     .. .. $('#text').val('');
66     .. .. });
67
```

```
68 db.orderBy('time', 'desc').onSnapshot(function(querySnapshot){
69   let str = '';
70   querySnapshot.docs.forEach(function(doc){
71     const id = doc.id;
72     const data = doc.data();
73     const datetime = convertFromFirestoreTimestampToDatetime(data.time.seconds);
74     str += '<li id="' + id + '>'; → //idにkey名を追加
75     str += '<p>' + data.name + '</p>';
76     str += '<p>' + datetime + '</p>';
77     str += '<p>' + data.text + '</p>';
78     str += '</li>';
79   });
80   $('#output').html(str);
81 });
82
83 </script>
84 </body>
85
86 </html>
```

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！！