

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→チュータリング(演習)タイム

授業のルール

- 授業中は常にエディタを起動！
- 隣の人と相談するときは周りの迷惑にならない大きさで.
- 周りで困ってそうな人がいたらおしえてあげましょう！
- まずは**打ち間違い**を疑おう！

{ } ' " ; など

- 書いたら保存しよう！

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 (function)
//   - 関数とは記述した処理をまとめ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

- 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.

// 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...

$$f(2) = 9,$$

$$f(5) = 36,$$

$$f(10) = 121$$



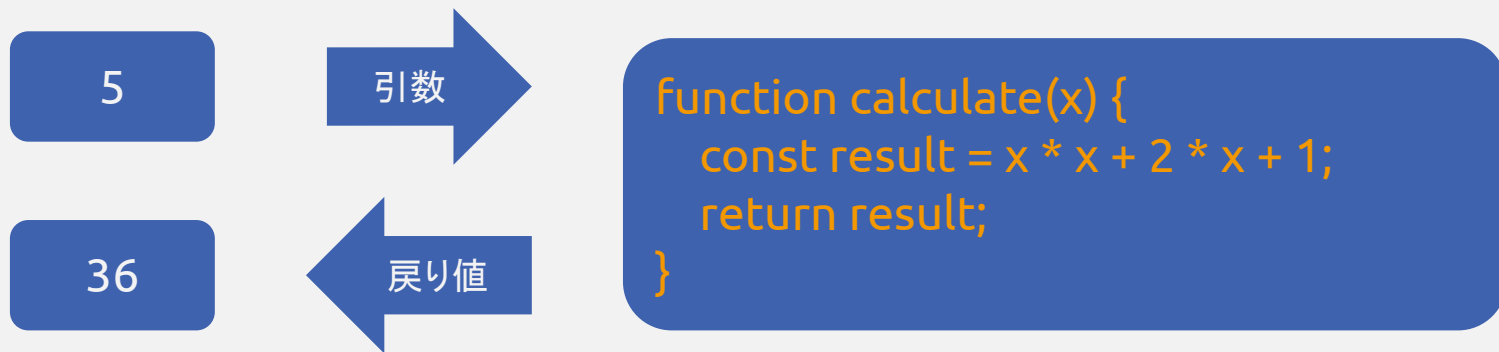
プログラミングの関数も数学の関数と同じ

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！！

関数の練習

- 関数の定義と実行(function01.html)
 - 関数を定義しよう！
 - 定義した関数を実行しよう！
- 引数と戻り値の練習(function02.html)
 - 引数と戻り値を持った関数を定義しよう！
 - 引数を渡して実行し、結果を表示しよう！

関数の利用

- 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

```
// ボタンをクリックしたイベントで関数を実行
$('#btn01').on('click', function () {
    var 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！
```

関数を使ってみよう！！

- 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

1. 片方で送信すると...

realtime chat

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！

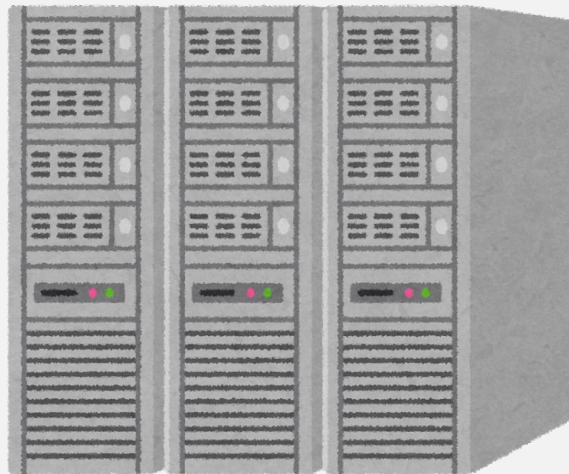
サーバにデータを保存とは??

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！



データ追加

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!

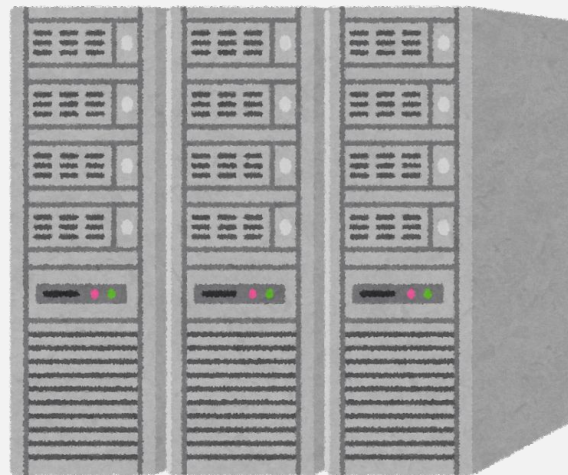


データ追加

全てに反映

全てに反映

Google

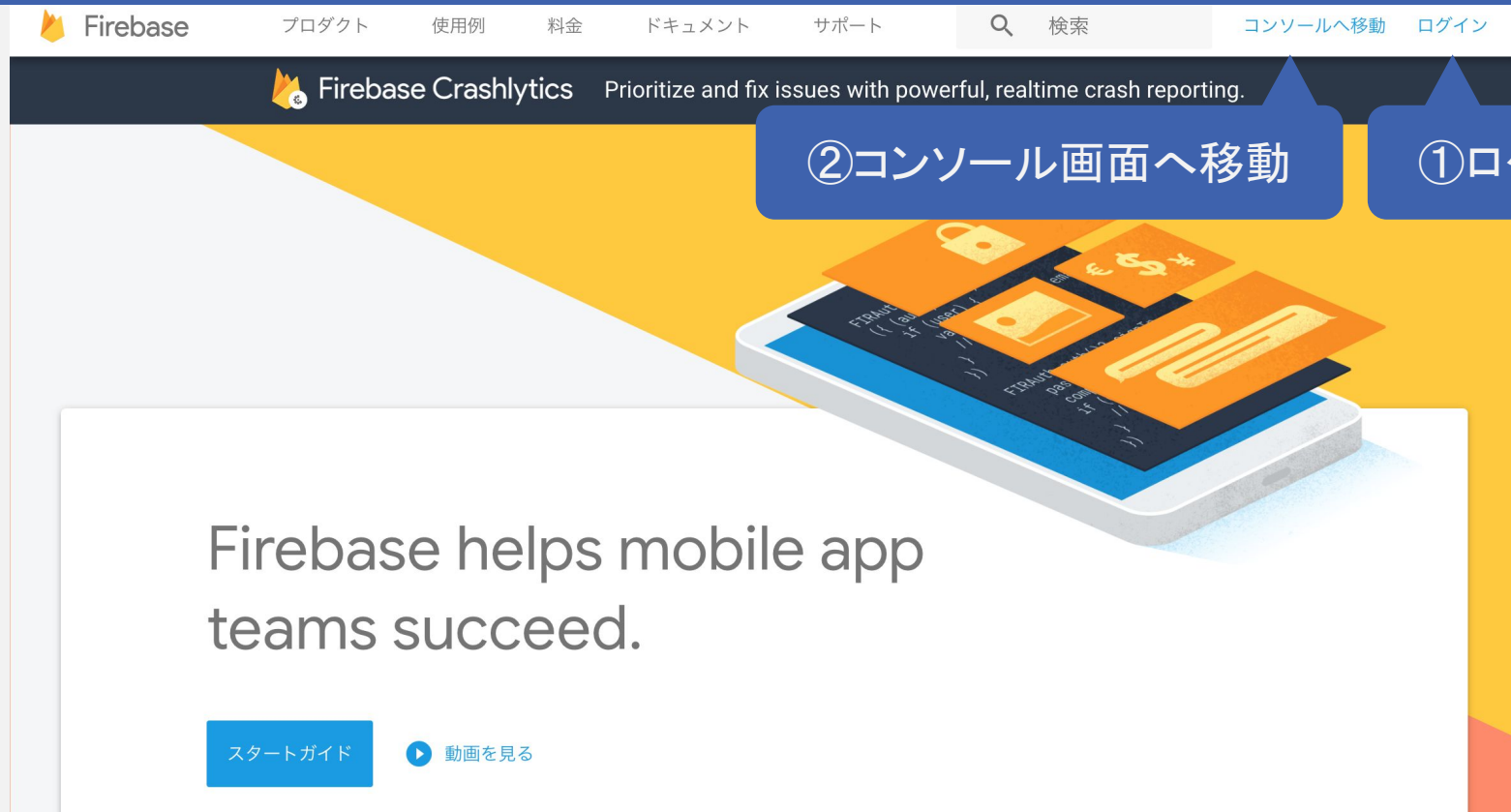


準備が必要なので進めよう！

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

<https://firebase.google.com/>



The screenshot shows the Firebase website in Japanese. The top navigation bar includes links for 'Firebase', '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, and links to 'コンソールへ移動' (Move to Console) and 'ログイン' (Login). Below the navigation bar, 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 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 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](#)

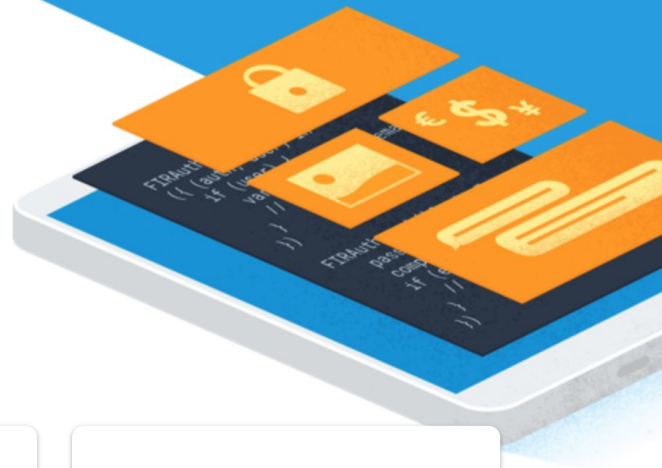
最近のプロジェクト



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



Firebase

Firebase へよう

優れたアプリの開発、ユーザー
収益向上に役立つ Google の

🔍 詳細 📄 ドキュメント 📁

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

FireBooks

firebooks-c83e4

プロジェクトの追加

プロジェクト名

chatapp

プロジェクト ID ⓘ

chatapp-f5d3b ✎

アナリティクスの地域 ⓘ

日本

Cloud Firestore のロケーション ⓘ

us-central

☐ Firebase 向け Google アナリティクスのデータ共有にデフォルトの設定を使用する

- ✓ Google サービスの改善のために、アナリティクス データを
- ✓ テクニカル サポートを有効にするために、アナリティクス データを共有します
- ✓ ベンチマークを有効にするために、アナリティクス データを共有します
- ✓ アナリティクス データを Google アカウント スペシャリストと共有します

☐ 測定管理者間のデータ保護条項に同意します。Google サービスの改善のためにアナリティクス データを共有する場合は同意する必要があります。詳細

キャンセル

次へ

プロジェクト名 : chatapp

アナリティクスの地域 : 日本
ロケーション : asia-northeast1

次へ → プロジェクト作成

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. A list of development tools follows: Authentication, Database, Storage, Hosting, Functions, and ML Kit. Below these are sections for '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 start using it). Below this text is a large blue arrow pointing towards the right. At the bottom of the main area are three circular icons: 'iOS', an Android robot, and a code symbol '</>'. The code symbol icon is highlighted with a red square. Below these icons, the text '開始するにはアプリを追加してください' (To get started, add an app) is visible. In the top right corner of the main area, there are links for 'ドキュメントに移動' (Move to documentation), a bell icon for notifications, and a coffee cup 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

ドキュメントに移動

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

データベースの準備



「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」

The screenshot shows the Google Cloud console interface. On the left is a sidebar with navigation links: 開発 (Development) with sub-links for Authentication, Database, Storage, Hosting, Functions, and ML Kit; 品質 (Quality) with links for Crashlytics, Performance, and Test Lab; アナリティクス (Analytics) with links for Dashboard, Events, Conversions, and A/B Testing; and 拡大 (Scale) with links for Predictions, A/B Testing, and Cloud ML. The main content area displays the 'cloud firestore のセキュリティ ルール' (Firestore Security Rules) dialog. The dialog has a title bar with a close button (X). Below the title, it states: '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。' (After defining the data structure, you need to create rules to protect the data.) followed by a '詳細' (Details) link. There are two radio button options: 'ロックモードで開始' (Start in Lock Mode) and 'テストモードで開始' (Start in Test Mode). The 'テストモードで開始' option is selected and highlighted with a red rectangle. Below the options is a code block showing the default security rules:

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

 A yellow warning box below the code states: '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拒否されます。' (Even users who own database references will be denied read or write access to the database.) A large purple arrow points from the '有効にする' (Enable) button to the warning box. The '有効にする' button is highlighted with a red rectangle. At the bottom of the dialog are 'キャンセル' (Cancel) and '有効にする' (Enable) buttons.

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拒否されます。

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

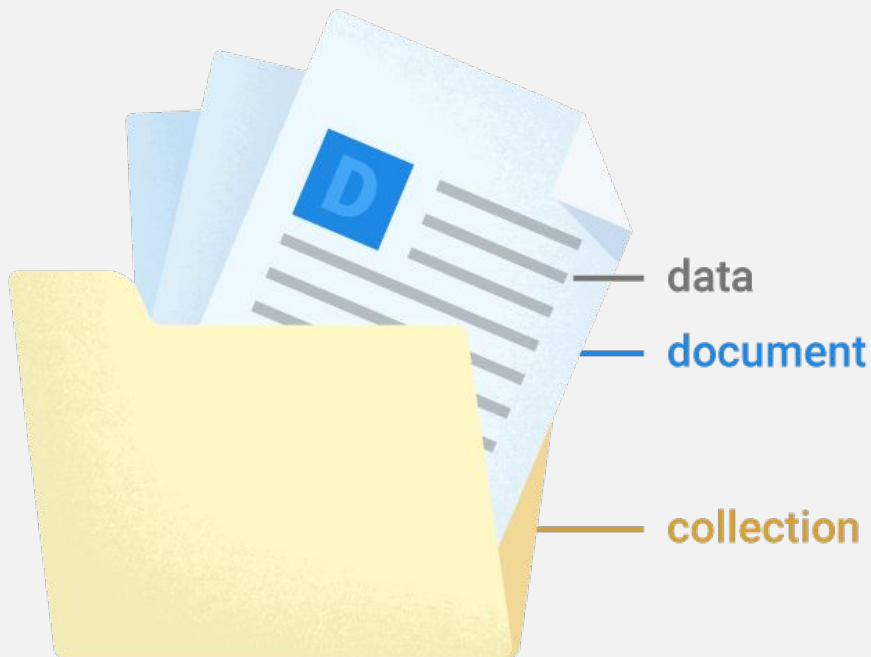
ほかは適当に！

キャンセル

保存

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

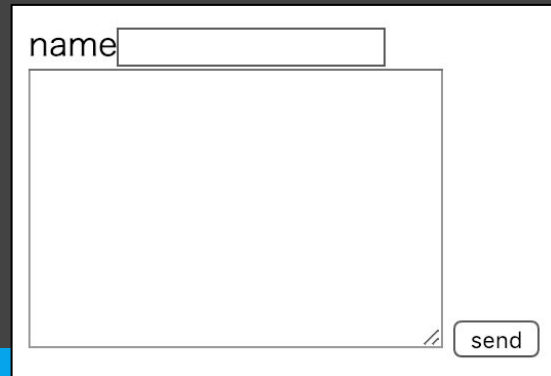
- データ追加時に自動的にデータ取得.
- 受信したデータをブラウザ上に表示.

チャット画面の作成

// 入力フォームの作成

```
<div>
  <div>name<input type="text" id="name">
</div>
<div>
  <textarea name="" id="text" cols="30" rows="10"></textarea>
  <button id="send">send</button>
</div>
<div id="output"></div>
</div>
```

こんな感じ！



name

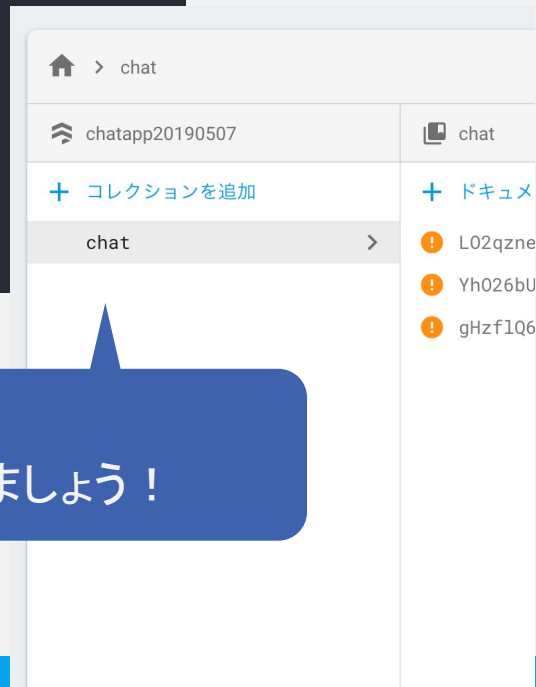
send

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
適当にデータを入れておきましょう！



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...
 - 名前と本文を取得.
 - firebaseにデータ(名前, 時間, 本文)を送信.
 - 本文入力用のtextareaを空にする.

```
// 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
  time: firebase.firestore.FieldValue.serverTimestamp(), // 登録日時
  text: $('#text').val(),              // textareaの入力値
});

// 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 a collection named 'chat'. A document with the ID 'L02qznemG9iUVsf0qv3P' is highlighted with a red box. A blue callout box points to this document ID with the text: '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)'. Below the highlighted document, other documents are listed with IDs like 'Yh026bUB1aWGUYToo899' and 'qHzf106...'. On the right, the document details for 'L02qznemG9iUVsf0qv3P' are shown, including a warning message: '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'.

- 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

- 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();
  });
  // 次ページへ続く...
```


// ...前ページの続き

```
    str += '<div id="' + id + '>';  
    str += '<p>' + data.name + '</p>';  
    str += '<p>' + data.time + '</p>';  
    str += '<p>' + data.text + '</p>';  
    str += '</div>';  
  });  
  $('#output').html(str);  
});
```

//idにkey名を追加

↓↓こんな感じで出力↓↓

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

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.

動作確認

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
11111111

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

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

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】チャットアプリ実装？

- 最低限ここまで！
 - 「名前」「日時」「メッセージ」を送信&表示
 - 表示領域を超えたときの処理を実装(overflow:auto;など)
 - 見た目をいい感じに！
- 追加仕様の例
 - 自分とそれ以外の投稿を分ける(メッセージャーみたいに)
 - 画像を表示
 - オンラインでじゃんけん
- ※例によってfirebaseを使えば何でもOK！

提出は次回授業前木曜「23:59:59」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   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   <title>chatApp</title>
9   <style>
10     #output div {
11       background: #ccc;
12     }
13   </style>
14 </head>
15
```

```
16 <body>
17   <h1>realtime chat</h1>
18   <!-- 入力&出力場所を作成しよう -->
19   <div>
20     <label for="name">name:</label>
21     <input type="text" id="name">
22   </div>
23   <textarea name="" id="text" cols="30" rows="10"></textarea>
24   <button id="send" type="button">send</button>
25
26   <div id="output"></div>
27
28   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
29   <!-- 以下にfirebaseのコードを貼り付けよう -->
30
31   <script src="https://www.gstatic.com/firebasejs/6.1.0/firebase.js"></script>
```

```
32 <script>
33   var firebaseConfig =
34     {
35       apiKey: "AIzaSyDbr
36       authDomain: "chat-
37       databaseURL: "http
38       projectId: "chat-1
39       storageBucket: "ch
40       messagingSenderId:
41       appId: "1:10106923
42     };
43   firebase.initializeApp(firebaseConfig);
44   // cloudfirestoreの場所を定義する処理
45   var db = firebase.firestore().collection('chat');
46 </script>
47
```



APIキーは自分のものを！


```

48 <script>
49 // 日時を取得する関数
50 function convertTimestampToDatetime(timestamp) {
51     const date = new Date(timestamp);
52     const year = date.getFullYear();
53     const month = ('0' + (date.getMonth() + 1)).slice(-2);
54     const day = ('0' + date.getDate()).slice(-2);
55     const hour = ('0' + date.getHours()).slice(-2);
56     const min = ('0' + date.getMinutes()).slice(-2);
57     const sec = ('0' + date.getSeconds()).slice(-2);
58     return `${year}-${month}-${day} ${hour}:${min}:${sec}`
59 }
60
61 // 送信ボタンクリック時にデータを送信する処理
62 $('#send').on('click', function() {
63     db.add({
64         name: $('#name').val(),
65         time: getNowDatetime(),
66         text: $('#text').val()
67     });
68     $('#text').val('');
69 });
70

```

```

71 // データをリアルタイムに取得する処理
72 db.orderBy('time', 'desc').onSnapshot(function(querySnapshot){
73     // console.log(querySnapshot.docs);
74     let str = '';
75     querySnapshot.docs.forEach(function(doc){
76         console.log(doc.data());
77         const id = doc.id;
78         const data = doc.data();
79         str += '<div id="' + id + '>';
80         str += '<p>' + data.name + '</p>';
81         str += '<p>' + convertTimestampToDatetime(data.time.seconds) + '</p>';
82         str += '<p>' + data.text + '</p>';
83         str += '</div>'
84     });
85     $('#output').html(str);
86 });
87
88 </script>
89 </body>
90
91 </html>

```

「写経」これでいける！！
提出は次回授業前木曜「23:59:59」まで！！

チュータリングタイム

わからなければ隣の人に訊く！！
訊かれた人は苦し紛れでも応える！！