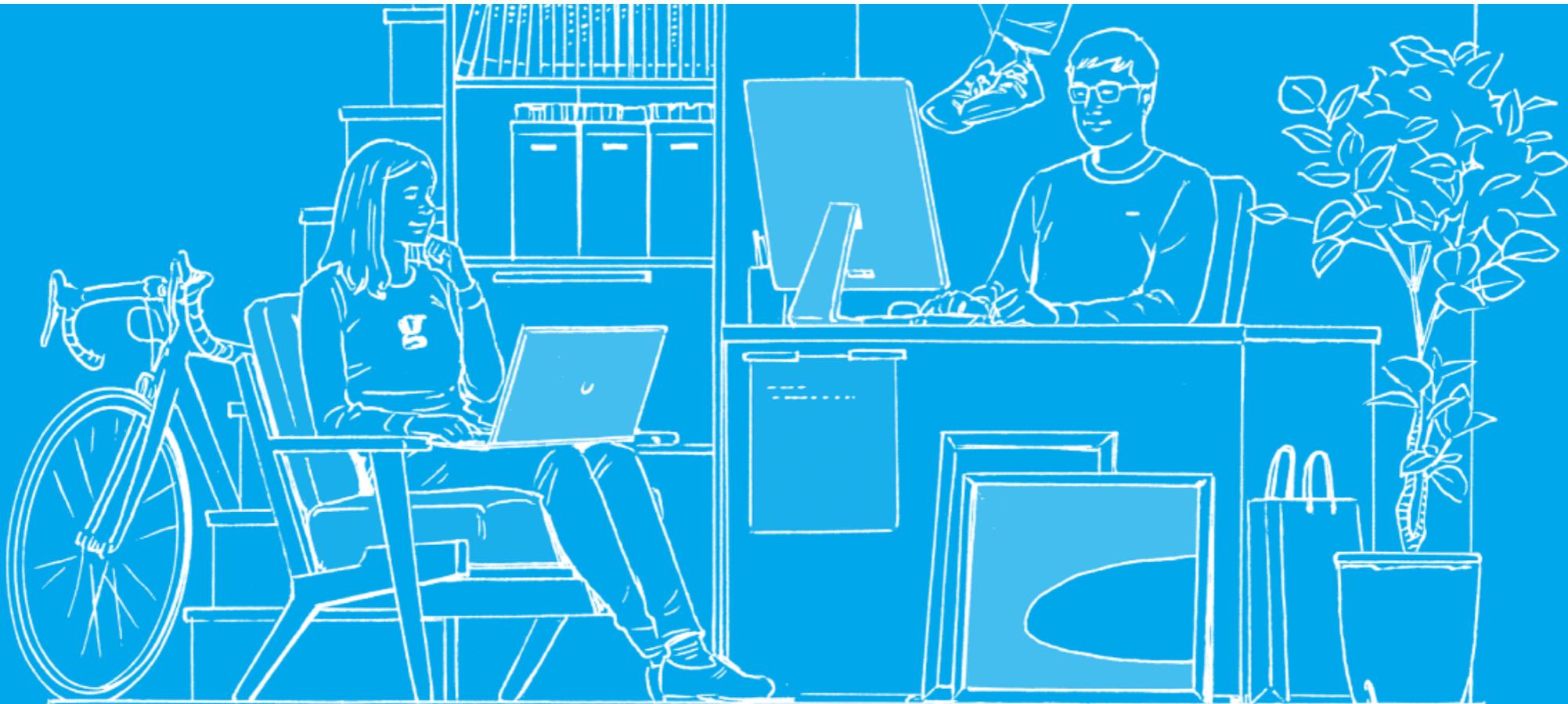




G's ACADEMY
TOKYO

Firebase Realtime Database

Googleアカウントが必要です



アジェンダ

❖ 本日のアジェンダ

- ・ オブジェクト変数
- ・ Firebase RealtimeDB
- ・ 課題実習タイム (Chatアプリ or リアルタイム通信)

オブジェクト

- Object -

★オブジェクトとは？？

超絶簡単にまとめると「データの塊」だと思ってください😊

※ここではなるべく専門用語を省いています

```
const gsStaff = {  
    kodama: 'トップオブトップ',  
    ranko: 'No2',  
    oohori: '長野で研究員',  
    fujii: 'ぎゃる'  
}
```

このように、データを一つにまとめてあるもの or まとまっている塊と思ってください😊

★オブジェクトとは？？

まずは塊の中のポイントにすべき箇所を押さえましょう！

```
const gsStaff = {  
    kodama: 'トップオブトップ',  
    ranko: 'No2',  
    oohori: '長野で研究員',  
    fujii: 'ぎやる'  
}
```

塊の中にあるオレンジの箇所を「鍵」

塊の中にある緑の箇所を「値」

※この**鍵がポイント**です！そして、必ず2つで1つの組み合
わせになっています😊

★オブジェクトの操作は？？

データの塊の名前にドットを使用して鍵にアクセスすることで
その中身にアクセスできます↓

```
const gsStaff = {  
    kodama: 'トップオブトップ',  
    ranko: 'No2',  
    oohori: '長野で研究員',  
    fujii: 'ぎやる'  
}
```

```
console.log(gsStaff.fujii) //ぎやるという文字が取得できる
```

★オブジェクトを学び、学習で触れるの？？

今回触るfirebaseもそうですが、webアプリケーションを開発する際にバックエンドで作られるデータをAPIと呼んでいます😊

そしてそのAPIのデータをjsで操作する際は必ず「**データの塊=オブジェクト**」で利用されます。つまり以下のような形↓

よって、今回触る「お作法」を理解することで、データの操作の仕方を学ぶことができるようになります😊

※APIは触ります😊

ではここから授業本編のFirebaseを学習していきます😊

チャット作成方法

JavaScript初級編



Firebase Realtime Database

サンプルレコード

[https://github.com/yamazakidaisuke/
youtube_chatRealDB](https://github.com/yamazakidaisuke/youtube_chatRealDB)

Key取得

Chromeブラウザでアクセス

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

A screenshot of the Firebase homepage. At the top, there's a navigation bar with links for 'Firebase', 'プロダクト', 'ソリューション', '料金', 'ドキュメント', 'もっと見る', '検索', 'Language', 'コンソールへ移動', and 'ログイン'. A red arrow points from the bottom right towards the 'ログイン' button. Below the navigation, a banner says 'Catch up on highlights from Firebase at Google I/O 2023. [Learn more](#)'. The main section features a large blue background with abstract shapes and icons related to development and deployment. On the left, there's a heading 'アプリを最適な状態へ' and a paragraph about Firebase being a platform for app and game development. At the bottom left, there are three buttons: '使ってみる' (Try it), 'デモを試す' (Try the demo), and '動画を見る' (Watch the video). A large white rectangular area is visible at the bottom.

1.00

1.00

アプリを 最適な状態へ

Firebase は、ユーザーに愛されるアプリやゲームの構築と拡大を支援するアプリ開発プラットフォームです。Google のインフラが支える、世界中の多くの企業から高い信頼を得ているサービスです。



使ってみる

デモを試す

動画を見る

アプリのライフサイクル全体を通して信頼できるプロダ

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

dev245

dev245-5dbef

react-future

react-future-d8bb5

</>

gs23

gs23-7ad5d

</>

reactdev22**GSdemo**

× 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付
けましょう^①

プロジェクト名を入力します

● プロジェクト名は必須です

gsdev-c3bde

続行

gsdev0603
と入力し続行

Google アナリティクス (Firebase プロジェクト向け)

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す。これにより、Firebase Crashlytics、Cloud Messaging、アプリ内メッセージング、Remote Config、A/B Testing、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って以下の機能が有効になります。

×

A/B テスト 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 ②

×

クラッシュに遭遇していないユーザー 数 ②

×

イベントベースの Cloud Functions トリガー ②

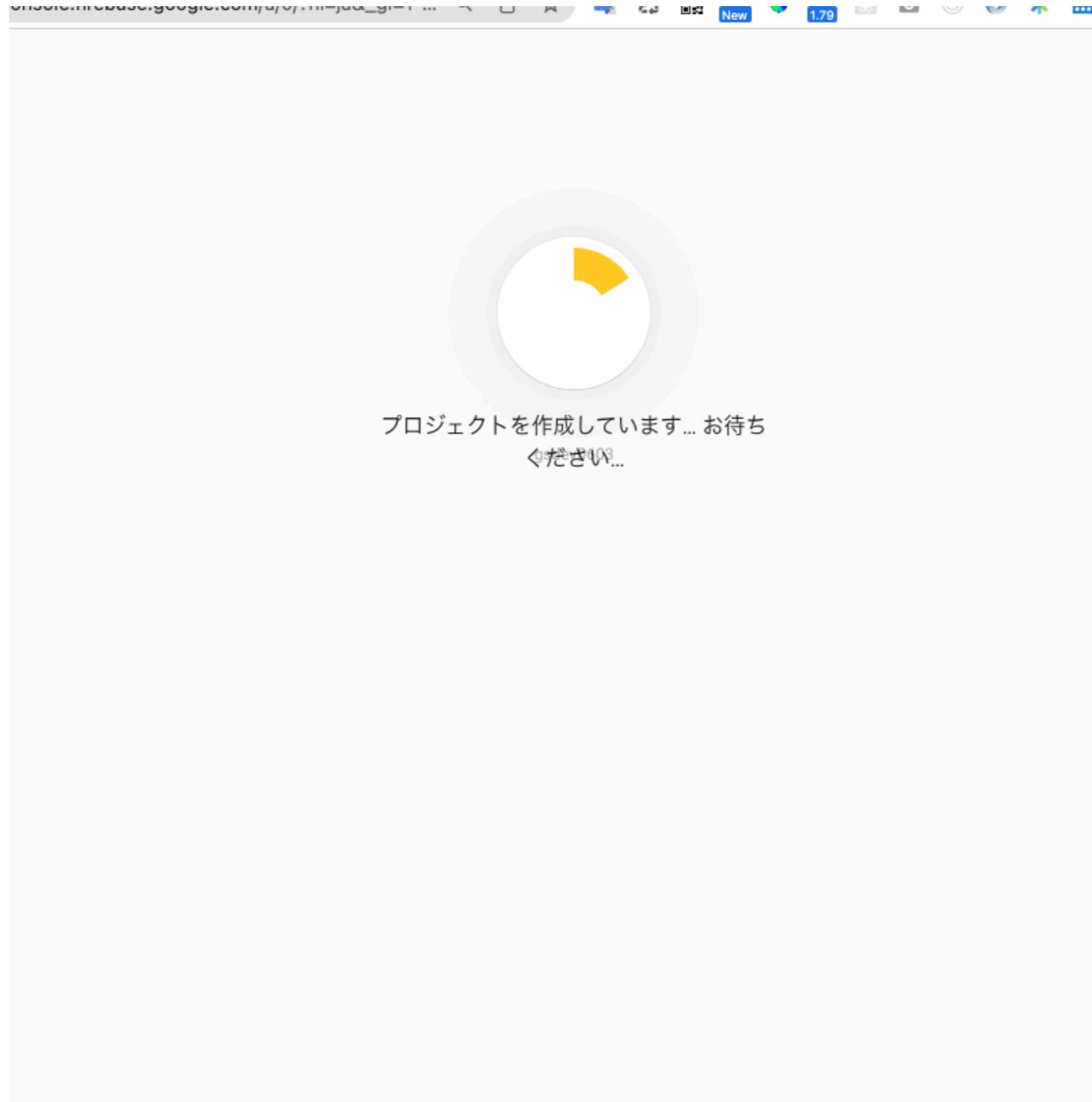
×

無料で無制限のレポート ②

このプロジェクトで Google アナリティクスを有効にする
おすすめ

前へ

プロジェクトを作成





gsdev0603

✓ 新しいプロジェクトの準備ができました

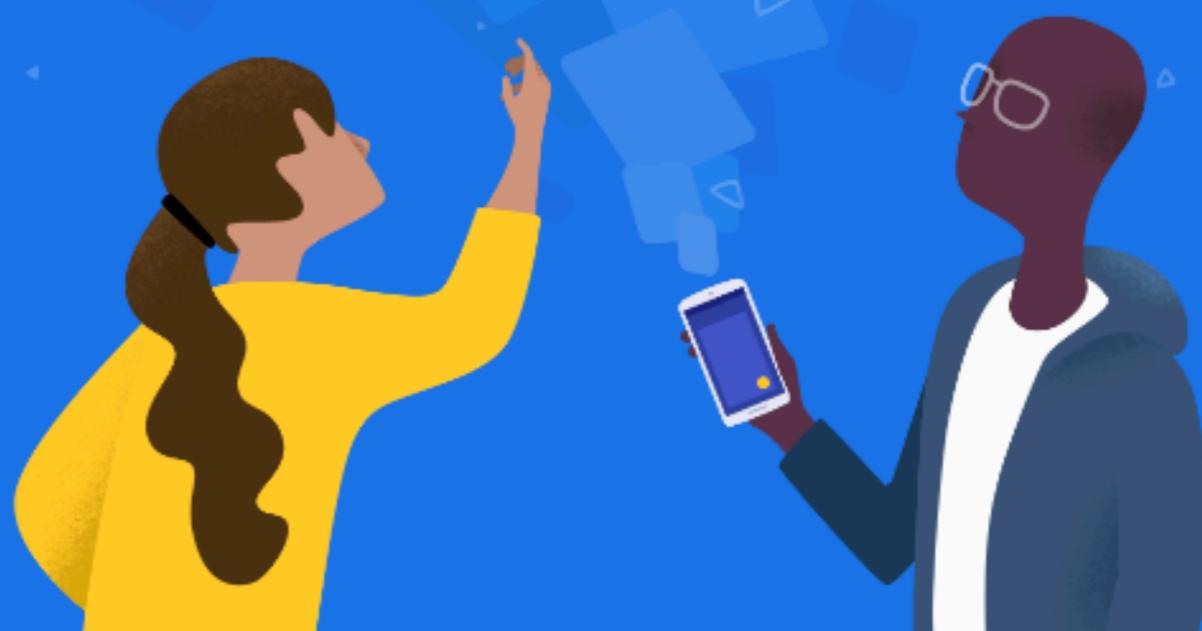
続行

アプリに Firebase を追加
して利用を開始しましょ
う

ウェブ



開始するにはアプリを追加してください



アプリデータを瞬時に保存して同期



× 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 ⓘ

マイ ウェブアプリ

● アプリのニックネームは必須です。

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。いつでも追加料金なしで開始できます。

アプリを登録

2 Firebase SDK の追加

gsdev0603

と入力し続行（名前は同じにしてください）

✓ アプリの登録

2 Firebase SDK の追加

- npm を使用する <script> タグを使用する

npm とモジュールバンドラ（webpack や Rollup など）をすでに使用している場合は、次のコマンドを実行して最新の SDK をインストールできます。 (詳細)

```
$ npm install firebase
```



次に Firebase を初期化し、使用するプロダクトの SDK の利用を開始します。

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// 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: "AIzaSyAP8gk7HZwp9NeVz-5H9n49ZleZN-dz3cc",
  authDomain: "gsdev0603.firebaseio.com",
  projectId: "gsdev0603",
  storageBucket: "gsdev0603.appspot.com",
  messagingSenderId: "827789072831",
  appId: "1:827789072831:web:d3635a155d7020a84fe90a"
};

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



注: このオプションではモジュラー JavaScript SDK を使用します。これにより SDK のサイズが小さくなります。

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





Firebase

gsdev0603 ▾



プロジェクトの概要



プロジェクトの設定

最新情報



Extensions 新機能



Functions 新機能

プロダクトのカテゴリ

構築



リリースとモニタリング



分析



エンゲージメント



すべてのプロダクト

ユーザーと権限

使用量と請求額

gsdev0603

Spark プラン

1 個のアプリ

+ アプリを追加

アプリに追加するプロダクトを選択します

アプリデータを瞬時に保存して



概要



ウェブアプリ

 gsdev0603
Web App

アプリのニックネーム

gsdev0603

アプリ ID

1:827789072831:web:d3635a155d7020a84fe90a

[Firebase Hosting サイトへリンク](#)

SDK の設定と構成



npm



CDN



Config

アプリの Firebase 構成オブジェクトのスニペットを入手します。 [詳細](#)

アプリのキーと ID が含まれている Firebase 構成オブジェクト:

```
const firebaseConfig = {
  apiKey: "AIzaSyAP8gk7HZwp9NeVz-5H9n49ZleZN-dz3cc",
  authDomain: "gsdev0603.firebaseio.com",
  projectId: "gsdev0603",
  storageBucket: "gsdev0603.appspot.com",
  messagingSenderId: "827789072831",
  appId: "1:827789072831:web:d3635a155d7020a84fe90a"
};
```

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](#) [このアプリを削除](#)



Firebase

gsdev0603 ▾

プロジェクトの設定



プロジェクトの概要



最新情報

Extensions (新機能)Functions (新機能)

プロダクトのカテゴリ

構築



Authentication

App Check

Firestore Database

Realtime Database

Extensions (新機能)

Storage

Hosting

Functions (新機能)

Machine Learning

Remote Config

リリースとモニタリング



分析



エンゲージメント



ウェブアプリ

 gsdev0603
Web App

アプリケーション

gsdev0603

アプリケーション

1:8277

Fire

SDK の



アプリケーション

アプリケーション

con

a

a

p

s

m

a

};

npm と
検討しウェブ
ウェブ



Firebase

gsdev0603 ▾

[プロジェクトの概要](#)[プロジェクト ショートカット](#)[Realtime Database](#)[最新情報](#)[Extensions \(新機能\)](#)[Functions \(新機能\)](#)[プロダクトのカテゴリ](#)[構築](#)[リリースとモニタリング](#)[分析](#)[エンゲージメント](#)[すべてのプロダクト](#)

Realtime Database

Realtime Database

データをリアルタイムで保存して同期

[データベースを作成](#)目的の用途に Realtime Database は適しているでしょうか。 [データベースを比較](#) □[詳細](#)[開始方法](#)[ドキュメントを表示](#)

Introducing Firebase Realtime Database



後で見る



共有



データベースの設定

×

- ① データベースのオプション —— ② セキュリティルール

ロケーションの設定は、Realtime Database データが格納される場所を定めます。

Realtime Database のロケーション

米国 (us-central1)



キャンセル

次へ

Realtime Database

データをリアルタイムで保存して同期

データベースの設定



データベースのオプション

2

セキュリティ ルール

X

データ構造の定義後に、データのセキュリティ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

データはデフォルトで限定公開になります。セキュリティ ルールで指定されているとおりに、クライアントの読み取り / 書き込み権限のみ付与されます。

```
{  
  "rules": {  
    ".read": "now < 1688310000000", // 2023-7-3  
    ".write": "now < 1688310000000", // 2023-7-3  
  }  
}
```

テストモードで開始する

データはデフォルトでオープン状態となり、迅速なセットアップが可能になります。ただし、クライアントの読み取り / 書き込み権限を長期に渡って有効にするには、30日以内にセキュリティ ルールを更新する必要があります。

! テストモードのデフォルトのセキュリティ ルールにより、今後30日間、データベース参照を所有しているユーザーなら誰でもデータベースのすべてのデータの表示、編集、削除を行うことができます

キャンセル

有効にする



Realtime Database の料金

[料金を見る](#)

Realtime Database

[データ](#) [ルール](#) [バックアップ](#) [使用状況](#) [Extensions](#) [新規](#)[ルールを編集](#)[ルールをモニタリング](#)[公開されていない変更](#)[公開](#)[破棄](#)[ルール](#)

```
1  {
2    "rules": {
3      ".read": true, // 2023-7-3
4      ".write": true, // 2023-7-3
5    }
6 }
```

初期設定

EXPLORER

...

simple.html ×

realtime_complete.html

index.html

JS_FIREBASE

index.html

realtime_complet...

simple.html

simple.html > {} "simple.html" > html > body > script

```
24  
25  
26     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>  
27     <script type="module">  
28         // Import the functions you need from the SDKs you need  
29         import { initializeApp } from "https://www.gstatic.com/firebasejs/9.1.0.firebaseio-app.js";  
30         import { getDatabase, ref, push, set, onChildAdded, remove, onChildRemoved }  
31             from "https://www.gstatic.com/firebasejs/9.1.0.firebaseio-database.js";  
32         // Your web app's Firebase configuration  
33         const firebaseConfig = {  
34             apiKey: "",  
35             authDomain: "",  
36             projectId: "",  
37             storageBucket: "",  
38             messagingSenderId: "",  
39             appId: ""  
40         };  
41         const app = initializeApp(firebaseConfig);  
42         const db = getDatabase(app); //RealtimeDBに接続  
43         const dbRef = ref(db, "chat"); //RealtimeDB内の"chat"を使う  
44  
45         //データ登録(Click)  
46  
47         //データ登録(Enter)  
48  
49             //最初にデータ取得&onSnapshotでリアルタイムにデータを取得  
50         </script>  
51     </body>  
52  
53 </html>
```

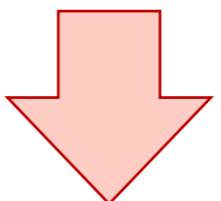
ここに自分の設定を行う

★ここ重要POINT！

firebaseのverが更新される度に書き方が変わります

```
26 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
27 <script type="module">
28 // Import the functions you need from the SDKs you need
29 import { initializeApp } from "https://www.gstatic.com/firebasejs/9.1.0.firebaseio.js";
30 import { getDatabase, ref, push, set, onChildAdded, remove, onChildRemoved } from "https://www.gstatic.com/firebasejs/9.1.0.firebaseio-database.js";
31 // Your web app's Firebase configuration
32 const firebaseConfig = {
```

別のサンプルファイル
からコピーできます！！



<<解説>>

デフォルト 「**firebase-app.js**」 では最小限のコアライブラリのみ読み込んでします。
新たに 「**firebase-database.js**」 を読み込み**RealtimeDatabase**を使えるようにし、**import**で必要な機能を利用できるようにします！

<https://firebase.google.com/docs/database/web/start?hl=ja>

Chat設定

[プロジェクトの概要](#)[プロジェクト ショートカット](#)[Realtime Database](#)[Authentication](#)[最新情報](#)[Extensions \(新機能\)](#)[Functions \(新機能\)](#)[プロダクトのカテゴリ](#)[構築](#)[Authentication](#)[App Check](#)[Firestore Database](#)[Realtime Database](#)[Extensions \(新機能\)](#)[Storage](#)[Hosting](#)[Functions \(新機能\)](#)[Machine Learning](#)[Remote Config](#)[リリースとモニタリング](#)[分析](#)[エンゲージメント](#)

Authentication

サーバーサイドのコードを使わずに、さまざまなプロバイダのユーザーを認証し管理します

[始める](#)[詳細](#)

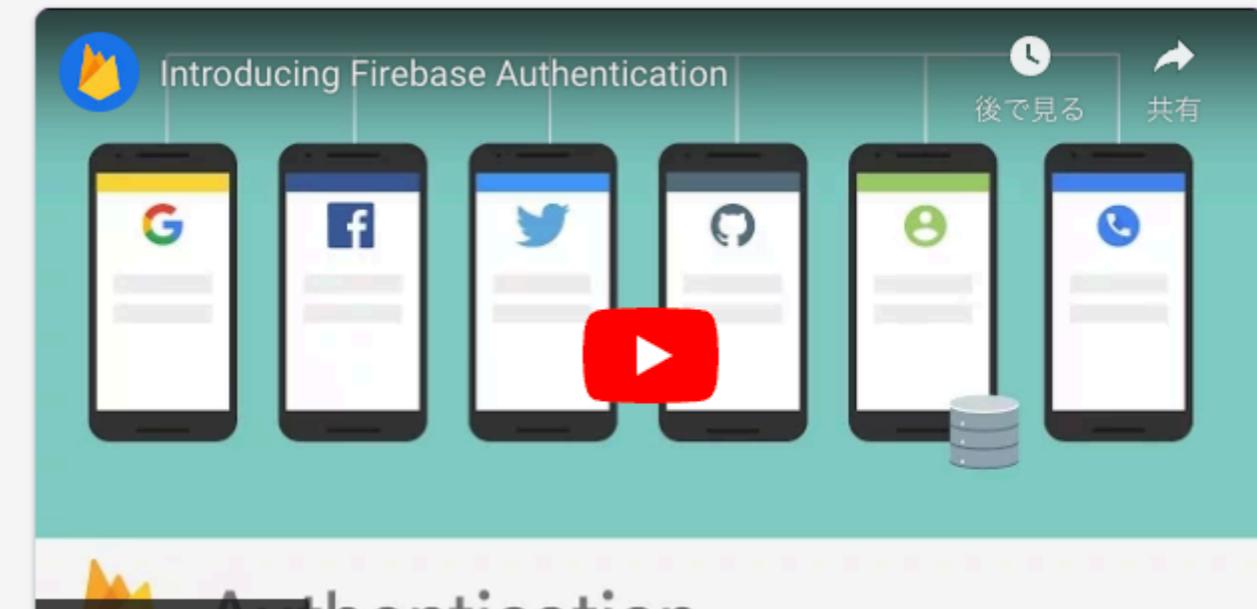
開始方法

[ドキュメントを表示](#)

Authentication の仕組み

[ドキュメントを表示](#)

Authentication で実現できること





Authentication

[Users](#)[Sign-in method](#)[Templates](#)[Usage](#)[Settings](#)[Extensions 新規](#)

ログイン プロバイダ

ログイン方法を追加して Firebase Auth の利用を開始しましょう

ネイティブのプロバイダ

メール / パスワード

電話番号

匿名

追加のプロバイダ

Google

Facebook

Play Games

Game Center

Apple

GitHub

Microsoft

Twitter

Yahoo!

詳細

Authentication

Users Sign-in method Templates Usage Settings Extensions 新規

ログイン プロバイダ

👤 匿名

有効にする

アプリケーションで匿名ゲスト アカウントを有効にします。これにより、認証情報の入力を要求することなく、ユーザー固有のセキュリティ ルールおよび Firebase ルールを強制できます。 [詳細](#)

キャンセル

保存

[詳細](#)

Authentication

Users Sign-in method Templates Usage Settings Extensions 新規

ログイン プロバイダ

新しいプロバイダを追加

プロバイダ

ステータス

匿名

✓ 有効

詳細



SMS 多要素認証

アカウントのセキュリティ レベルを高めることができます。有効にし、統合と構成を行うと、ユーザーは SMS を使用して 2 ステップで自分のアカウントにログインできます。[詳細](#)



MFA とその他の高度な機能は、Identity Platform で使用できます。Identity Platform は、Google Cloud の包括的な顧客 ID ソリュ

画面作成

エディターを開きHTMLを追記します。

☆Emmet: [div>div*3] を使いdivのブロックだけ作りましょう！

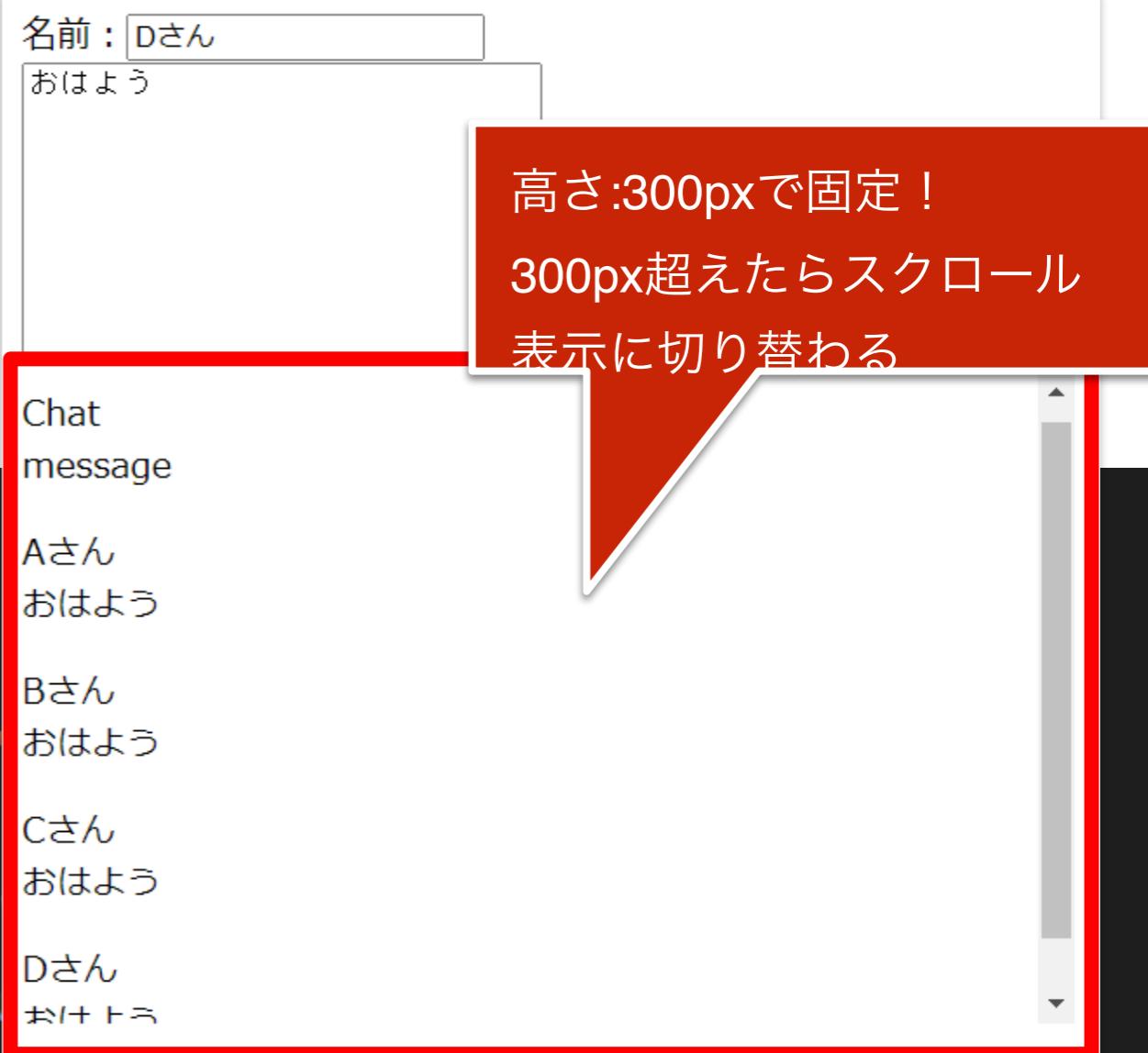
```
9   <!-- コンテンツ表示画面 -->
10
11  <div>
12    <div>名前:<input type="text" id="uname"> </div>
13    <div>
14      <textarea id="text" cols="30" rows="10"></textarea>
15      <button id="send">送信</button>
16    </div>
17    <div id="output"></div>
18  </div>
19
```



HTMLを追記後、HTMLをブラウザで表示します。

styleを追加！

```
9   <!-- コンテンツ表示画面 -->
10
11 <div>
12   <div> 名前 : <input type="text">
13   <div>
14     <textarea id="text" cols="50" rows="10" style="width: 100%; height: 100%; border: none; font-size: 1em; margin-bottom: 10px;"></textarea>
15     <button id="send">送信 </button>
16   </div>
17   <div id="output" style="overflow: auto; height: 300px;"></div>
18 </div>
```



【インラインで追加】

style="overflow: auto; height: 300px;"

Chat処理記述

送信ボタンのクリックイベントを作成

△使うElement(要素)

- ・ ボタン #send
- ・ テキストボックス #uname
- ・ テキストエリア #text
- ・ div (メッセージ表示領域) #output



△ボタンclickイベントを作成

```
57 //送信
58 $("#send").on("click",function(){
59     const uname = $("#uname").val();
60     const text = $("#text").val();
61     alert(uname+text); //取得確認
62 }
63 );
```

データ送信：処理を記述します。

```
46 //データ登録
47 $("#send").on("click",function() {
48     const msg = {
49         uname: $("#uname").val(),
50         text: $("#text").val()
51     }
52     const newPostRef = push(dbRef); //Pushできる状況を作って
53     set(newPostRef, msg);          //DBに値をセットする
54 });


```

以下処理の流れ

47: id="send"をクリックしたら

49: id="uname" から入力データを取得

50: id="text"から入力データを取得

52: "chat"データベースにデータを追加する準備

53: Realtime Database"chat"にオブジェクトデータを追加！！

データ受信：処理を記述します。

`onChildAdded()` を使い送信してくるデータを監視！

例) 受信完成コード

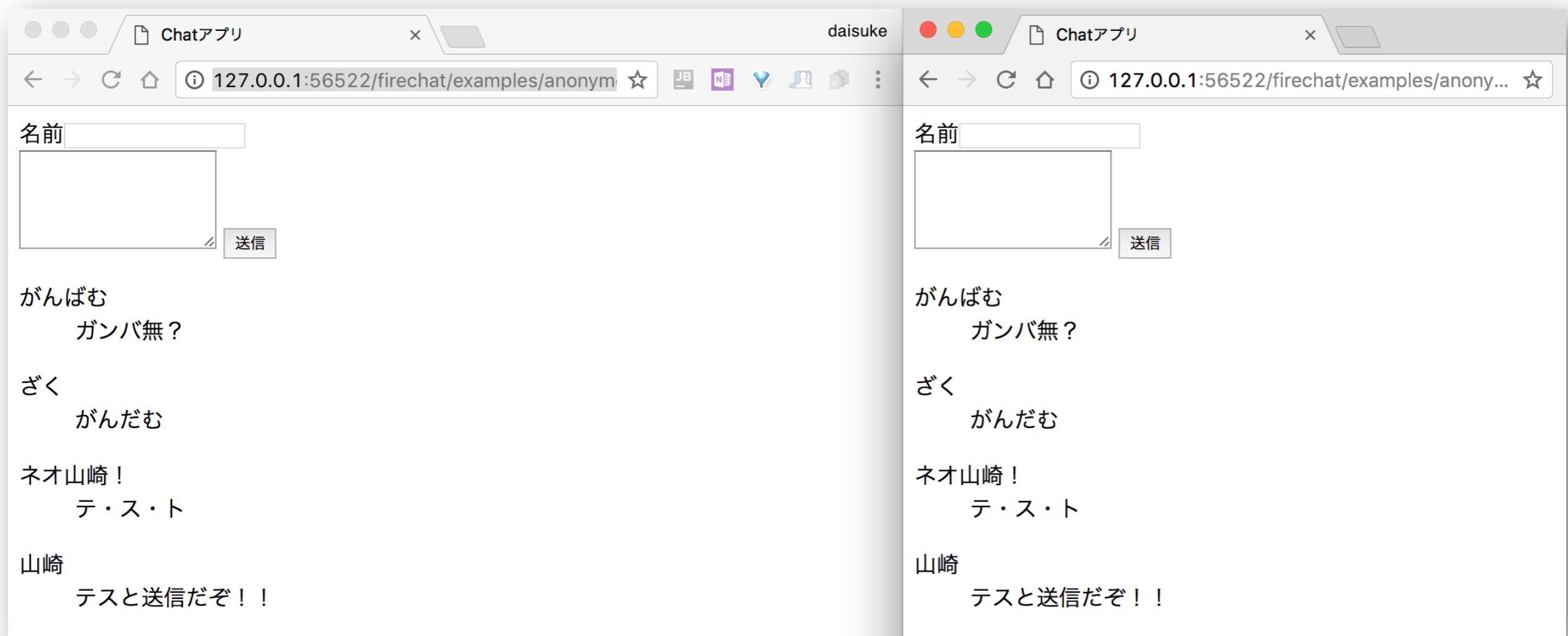
```
56 //最初にデータ取得 & onSnapshotでリアルタイムにデータを取得
57 onChildAdded(dbRef, function(data){
58     const msg = data.val();
59     const key = data.key;
60     let h = '<p>';
61     h += msg.uname;
62     h += '<br>';
63     h += msg.text;
64     h += '</p>';
65     $("#output").append(h); //#outputの最後に追加
66 })
```

受信したデータを変数に代入

この赤枠内で
表示するHTMLを作成して
表示させています！！

チャット動作確認

ブラウザ2つで開き、チャット送信ができているか確認しましょう。



EnterKeyで送信

Enter Keyで送信する方法

keydownイベント：キー入力を取得

```
48 ... $("#text").on("keydown", function(e){  
49     ...     console.log(e);  
50     ...});
```

console画面で確認

```
▼ m.Event {originalEvent: KeyboardEvent, type: "keydown", timeStamp: 6131.685000000001,  
  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, guid: 4, handler:  
  isDefaultPrev  
  jQuery1113088  
  key: "Enter"  
  keyCode: 229  
  metaKey: false  
  originalEvent: KeyboardEvent {isTrusted: true, key: "Enter", code: "F13", location: "Left", repeat: false}  
  relatedTarget: undefined  
  shiftKey: false  
  target: textarea#text}
```

keyCode:13がEnter

eがキモ！

Enterを押したら
送信を作って見よう！

ポイント

console.logを使うことが大切です！！！

EnterKey の番号を取得したり、
何処をクリックしたのか？など、X,Y座標も取得したり幅広
く使います。

console.logを活用してデータを可視化することで開発スピ
ードが上がります。

課題発表

Line風アプリ制作

●課題

◇ Line風アプリの課題最低限機能

1. 「メッセージ表示領域を超えた処理」

※表示エリア : height:300px;overflow:auto; などでスクロール表示

※授業後に時間があればグループでこの問題を解決しましょう！！

2. 削除機能

※授業後に時間があればグループでこの問題を解決しましょう！！

3. 見た目の装飾とか

さらにあれば良いと思わる機能

「アイコン」「メッセージ翻訳」「対戦ジャンケン」「メモ帳をCloudに保存」とか？なんでもあり！