



**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 -

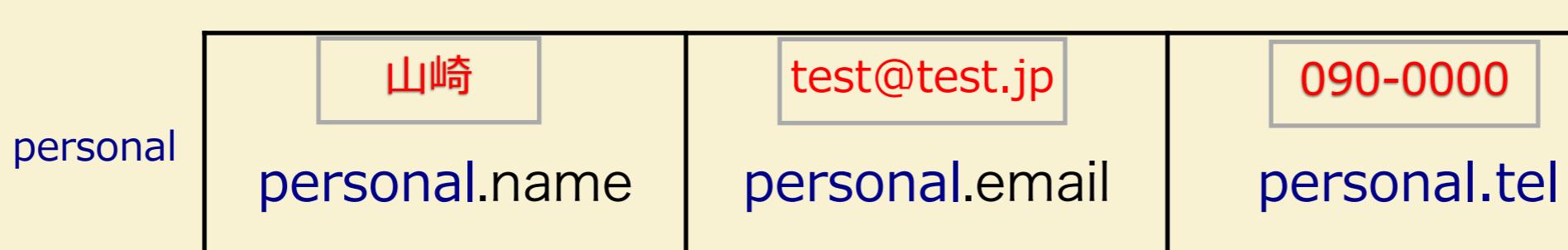
## 【オブジェクト (Object)】

オブジェクトは配列と違い「インデックス」ではなく「プロパティ」で値を管理することができます。プロパティは「名前:値」のペアになっており最近ではよく使用されるデータ保持の方法の1つです。

```
<script>
const personal = { name:"山崎", email:"test@test.jp", tel:"090-0000" };
const personals = {
  per1: { name:"山崎", email:"test1@test.jp", tel:"090-1111" },
  per2: { name:"鈴木", email:"test2@test.jp", tel:"090-2222" }
};
</script>
```

## 【オブジェクトの参照イメージ】

personalオブジェクト内のプロパティに値が格納されます。personalの中の値を取得する方法は、「`personal.プロパティ名`」で取得可能



# チャット作成方法

## JavaScript初級編



Firebase Realtime Database

# サンプルコード

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

# Key取得

Chromeブラウザでアクセス

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

1

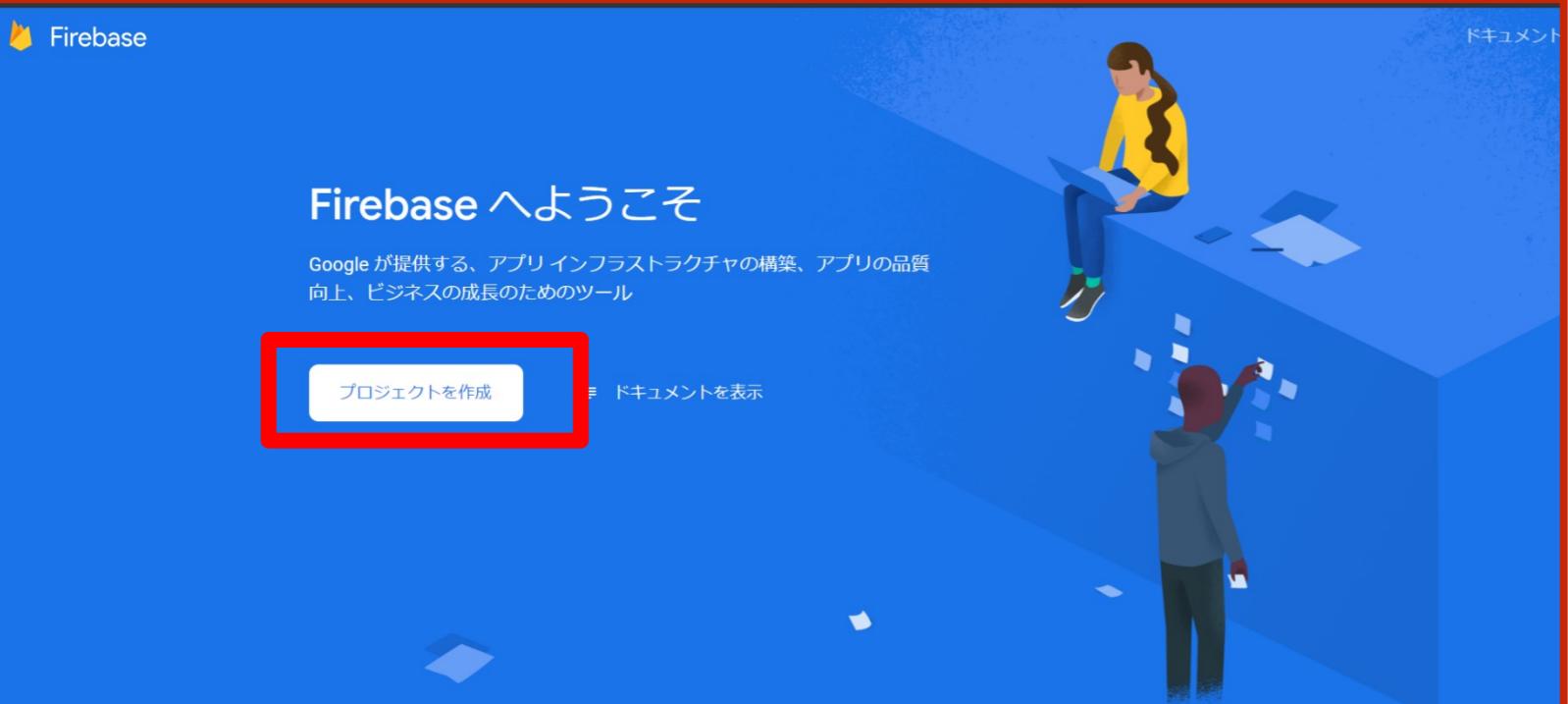
The screenshot shows the official Firebase website. At the top, there's a navigation bar with links for 'Firebase', 'プロダクト', '使用例', '料金', 'ドキュメント', 'サポート', a search bar, 'Language' dropdown, and a 'Console' button which is highlighted with a red box. Below the navigation is a large blue banner with white text that reads 'Firebase で モバイルとウェブ アプリを成功させる'. On the right side of the banner, there's an illustration of two people: one standing and pointing upwards, and another sitting cross-legged on the floor. The background of the banner has a subtle geometric pattern. A large white rectangular area is visible at the bottom of the page.

Firebase で  
モバイルとウェブ  
アプリを成功させる

使ってみる

動画を見る

# 1.新規プロジェクト作成



Firebaseへようこそ

Googleが提供する、アプリインフラストラクチャの構築、アプリの品質向上、ビジネスの成長のためのツール

プロジェクトを作成 ドキュメントを表示

初回画面

デモプロジェクトを見る iOS

Firebase プロジェクトのコンテナ

プロジェクト内の Database やアナリティクスを有しています。

Learn more

ドキュメントに移動

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

すべての Firebase プロジェクト

2回目以降

The image shows the initial Firebase console interface. A red box highlights the 'プロジェクトを作成' (Create Project) button in the top left corner. Below it, a large central area features a 3D illustration of two people working on a computer. To the left, there's a sidebar with a 'デモプロジェクトを見る' (View Demo Project) button for iOS. The main content area has a section titled 'Firebase プロジェクトのコンテナ' with a note about database and analytics. On the right, there's a 'recent projects' section with a '+ Add project' button and a 'View demo project' button. The bottom of the page says 'すべての Firebase プロジェクト' (All Firebase Projects). Red boxes also highlight the '+ Add project' button in the second screenshot and the 'View demo project' button in the third screenshot.

× 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るましょう<sup>?</sup>

1

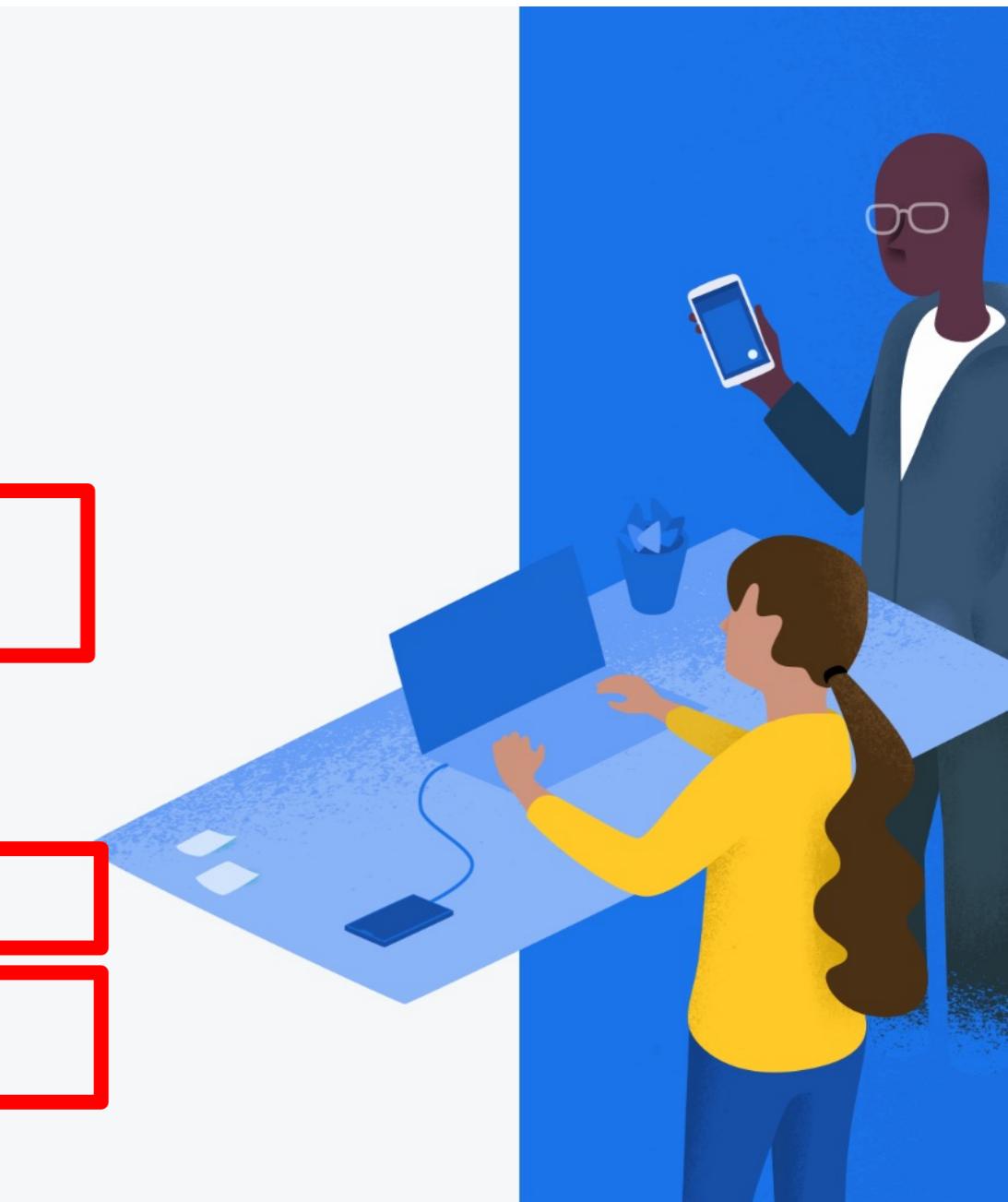
プロジェクト名

GSdemo

 gsdemo-29f17

2  [Firebase の規約](#)に同意します

3 [続行](#)



× プロジェクトの作成（手順 2/2）

## 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、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
- × Firebase プロダクト全体でのユーザー登録とターゲティング ②
- × ユーザー行動の予測 ②
- × クラッシュに遭遇していないユーザー 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

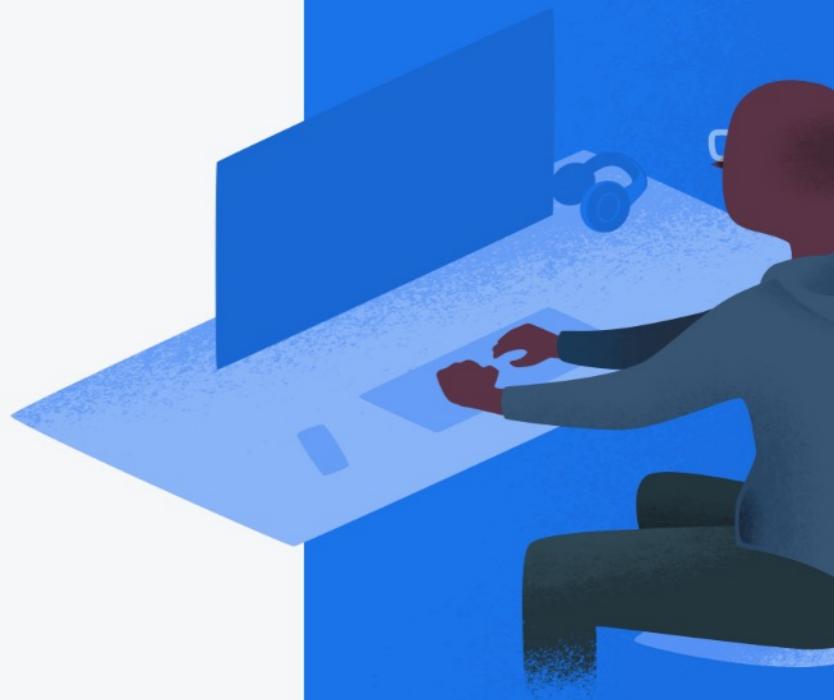
1

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

前へ

2

プロジェクトを作成





gsdemo

✓ 新しいプロジェクトの準備ができました

続行



# 5. 「</>」を選択（Webアプリ）

The screenshot shows the Firebase Project Overview page for a project named 'dev12'. The left sidebar contains navigation links for 'Project Overview', 'Authentication', 'Database', 'Storage', 'Hosting', 'Functions', 'ML Kit', 'Crashlytics, Performance, Test L...', 'Analytics', 'Dashboard, Events, Conversions,...', 'Predictions', and 'AMP Tools'. The main content area displays the text 'dev12 Spark プラン' and 'アプリに Firebase を追加して利用を開始しましょう'. Below this, it says 'ほとんどの Firebase 機能と、iOS や 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'. At the bottom, there are three circular icons for 'iOS', 'Android', and 'Web' (represented by '</>'), with the '</>' icon highlighted by a red box. The text '開始するにはアプリを追加してください' is located below these icons. In the background, there is an illustration of two people, one holding a smartphone.

# 6. アプリ名「chat」と登録。

× 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 ①

chat

今日はChatにしておく

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



7. コピーボタンで[CODE]をコピーしておく。  
Firebaseに接続する大事にコードになります。

1. **<script>を選択**

アプリの登録

2. Firebase SDK の追加

npm を使用する ②  <script> タグを使用する ③

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/fire
  // 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: "AIzaSyCB-sBzBVC3lzx6PSSaeIl27mH-_pDbswI",
    authDomain: "test-3926b.firebaseio.com",
    projectId: "test-3926b",
    storageBucket: "test-3926b.appspot.com",
    messagingSenderId: "994269086377",
    appId: "1:994269086377:web:eb6cc57dc62d512f5c9fb8"
  };

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

2. コードをコピー

3. **コンソールに進む**

## 8. エディターを開きスクリプトを記述する準備をしましょう。

```
1  <!DOCTYPE html>
2 ▼ <html lang="ja">
3 ▼ <head>
4   .... <meta charset="UTF-8">
5   .... <title>Document</title>
6 </head>
7 ▼ <body>
8   ....
9   ....
10  ....
11 ...
12 ...
13 </body>
14 </html>
```

最低限のHTMLは記述しておきました。

# 9. コピーしたスクリプトを「貼り付け」します。 今回はサンプルファイル「29行目」に貼り付 け

simple.html

```
23  <!-- JQuery -->
24  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
25  <!-- JQuery -->
26
27
28  <!--** 以下Firebase **-->
29
30
31
32
33  Firebaseに接続する大事なKEYをここに貼り付けます
34
35
36
37
38
39
```

# サンプルファイルの 29行目に貼り付けた例です

```
28 <!--* 以下Firebase *-->
29 <script type="module">
30 // Import the functions you need from the SDKs you need
31 import { initializeApp } from "https://www.gstatic.com/firebasejs/9.0.2/firebase-app.js";
32 // TODO: Add SDKs for Firebase products that you want to use
33 // https://firebase.google.com/docs/web/setup#available-libraries
34
35 // Your web app's Firebase configuration
36 const firebaseConfig = {
37   apiKey: "*****",
38   authDomain: "*****",
39   projectId: "*****",
40   storageBucket: "*****",
41   messagingSenderId: "*****",
42   appId: "*****:*****"
43 };
44
45 // Initialize Firebase
46 const app = initializeApp(firebaseConfig);
47 </script>
```

ちゃんと29行目？

次のページへ

# ★ここ重要POINT！

## コピーしたコードに追加が必要！！

```
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 = {
```

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>

# 完成ファイル realtime\_complete.html のコードをコピー！！

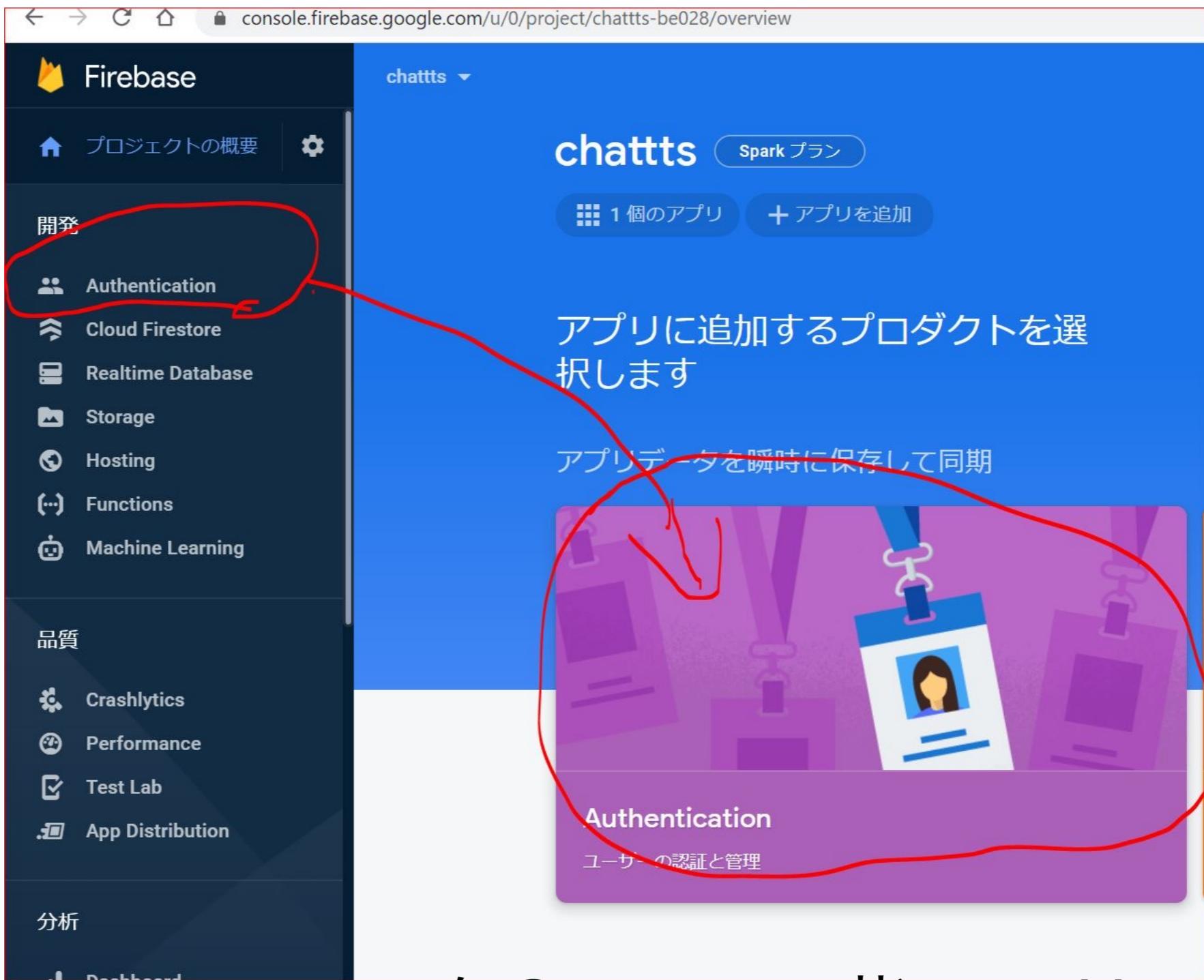
```
import { getDatabase, ref, push, set, onChildAdded,  
remove, onChildRemoved }  
from "https://www.gstatic.com/firebasejs/9.1.0.firebaseio.js"
```

同じ行くらいにあるので追加してください！

# Chat設定

## Authentication

# 11. コンソール画面



AuthenticationをClick!!→ 紫のエリアをクリック！



Firebase

chattts ▾

ホーム プロジェクトの概要



開発

Authentication

Cloud Firestore

Realtime Database

Storage

Hosting

Functions

Machine Learning

品質

Crashlytics

Performance

# Authentication

サーバー側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

始めるをクリック！！

# 12. 「Authentication」→「ログイン方法」→「匿名」の順番に選択

The screenshot shows the Firebase console interface. On the left, a sidebar lists various services: Project Overview, Authentication (which is selected and highlighted with a red box), Firestore Database, Realtime Database, Storage, Hosting, Functions, Machine Learning, Releases & Monitoring, and Analytics. The main content area is titled 'Authentication'. At the top of the main area, there are tabs: 'Users', 'Sign-in method' (which is also highlighted with a red box), 'Templates', and 'Usage'. Below these tabs, there's a heading 'ログインプロバイダ' (Login Providers) with the sub-instruction 'ログイン方法を追加して Firebase Auth の利用を開始しましょう' (Add a login method to start using Firebase Auth). The page is divided into two sections: 'ネイティブのプロバイダ' (Native providers) and '追加のプロバイダ' (Additional providers). Under 'ネイティブのプロバイダ', there are buttons for 'メール / パス...' (Email / Password) and '電話番号' (Phone number). Under '追加のプロバイダ', there are buttons for Google, Facebook, Play ゲーム, Game Center, Apple, GitHub, Microsoft, Twitter, and Yahoo!. A red arrow points from the 'Authentication' sidebar item to the 'Sign-in method' tab, and another red arrow points from the 'Sign-in method' tab to the 'Anonymous' provider button.

# 13. 「有効にする」 → 「保存」 の順番でクリック

The screenshot shows the Firebase console's Authentication screen for a project named "mychat". On the left sidebar, under the "DEVELOP" section, the "Authentication" option is selected. The main content area displays the "Authentication" tab with a list of providers: Google, Facebook, Twitter, GitHub, and Anonymous. The "Anonymous" provider is currently disabled (indicated by the word "無効" next to its icon). Below the provider list, there is a section for "Anonymous" with a toggle switch labeled "有効にする" (Enable) which is turned on (blue), and a "保存" (Save) button. Both the "有効にする" switch and the "保存" button are highlighted with red boxes.

Firebase

mychat ▾ Authentication ドキュメントに移動

Project Overview |

DEVELOP

Authentication Database Storage Hosting Functions

STABILITY Crash Reporting, Performance, Test ...

ANALYTICS Dashboard, Events, Audiences, Attrib...

Authentication

Google 無効

Facebook 無効

Twitter 無効

GitHub 無効

匿名

有効にする  保存

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

# Chat設定

## Realtime Database

# 14. 「Database」 → 「データベースの作成」

The screenshot shows the Firebase Realtime Database creation interface. On the left sidebar, under the '開発' (Development) section, the 'Realtime Database' option is highlighted with a red box and a red arrow pointing to it from the top-left corner. A larger red circle highlights the 'データベースを作成' (Create Database) button in the center of the main area. The main title 'Realtime Database' is displayed prominently, along with the Japanese text 'データをリアルタイムで保存して同期' (Save data in real-time and sync). Below the main title, there is a comparison section asking '目的の用途に Realtime Database は適しているでしょうか。' (Is Realtime Database suitable for your intended purpose?) with a 'データベースを比較' (Compare databases) link. At the bottom, there is a '詳細' (Details) section with a '開始方法' (Getting Started) button and a 'Introducing Firebase Realtime Database' video thumbnail.

# 15. 「ルール」 → 「テストモード」に設定

The screenshot shows the Firebase Realtime Database security rules configuration dialog. It includes the following elements:

- Realtime Database のセキュリティ ルール**: A heading with a close button (X).
- 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。**: A note.
- 詳細**: A link to detailed information.
- ロックモードで開始**: An option with a description: "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**: An option with a description: "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". This option is highlighted with a red box.
- Code Preview**: A code block showing the JSON rules:

```
{  "rules": {    ".read": true,    ".write": true  }}
```
- 注意**: A yellow callout with text: "データベース参照を所有しているユーザーなら誰でも、データベースの読み取りや書き込みを行えるようになります".
- キャンセル**: A button to cancel.
- 有効にする**: A blue button to enable the rules, which is also highlighted with a red box and has a red arrow pointing to it from the explanatory text below.

これで「匿名」の誰でもチャット参加することが可能になりました。  
次にチャットの最低限のコードを実装していきましょう。

# 16. DB作成完了

The screenshot shows the Firebase Realtime Database console for a project named "gsdemo". The "Database" tab is selected, and the "Realtime Database" sub-tab is active. The "Data" tab is currently selected. A prominent red warning message is displayed: "⚠ セキュリティ ルールが公開として定義されているため、誰でもデータベース内のデータを窃取、変更、削除できます。" (A security rule is defined publicly, so anyone can read, change, or delete data from the database). Below the message is a URL: "https://gsdemo-15ec4.firebaseio.com/". At the bottom of the modal, there are buttons for "詳細" (Details) and "閉じる" (Close). The left sidebar shows other Firebase services: Authentication, Database, Storage, Hosting, Functions, and ML Kit. The top right corner has a link to "ドキュメントに移動" (Move to documentation).

# 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をブラウザーで表示します。

# 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">
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

# jqueryを使用できるようしています。 まずは簡単に動作させるため！

```
22  <!-- JQuery -->
23  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
24  <!-- JQuery -->
```

今回はCDN（Webサーバー上に用意されてるライブラリ）から読み込んでいます。

※各自でライブラリをダウンロードして使ってもOKです。

# Realtime Databaseを使う2行を追加！

```
33 const firebaseConfig = {  
34   apiKey: "*****",  
35   authDomain: "*****",  
36   projectId: "*****",  
37   storageBucket: "*****",  
38   messagingSenderId: "*****",  
39   appId: "*****"  
40 };  
41 // 9941  
42 const app = initializeApp(firebaseConfig);  
43 const db = getDatabase(app); //RealtimeDBに接続  
44 const dbRef = ref(db, "chat"); //RealtimeDB内の"chat"を使う  
45
```

## 【2行追記】

```
const db      = getDatabase(app); //
```

RealtimeDatabase使うよ

```
const dbRef = ref(db, "chat"); //
```

RealtimeDatabase"chat"を使うよ

# 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 });

64
```

# 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

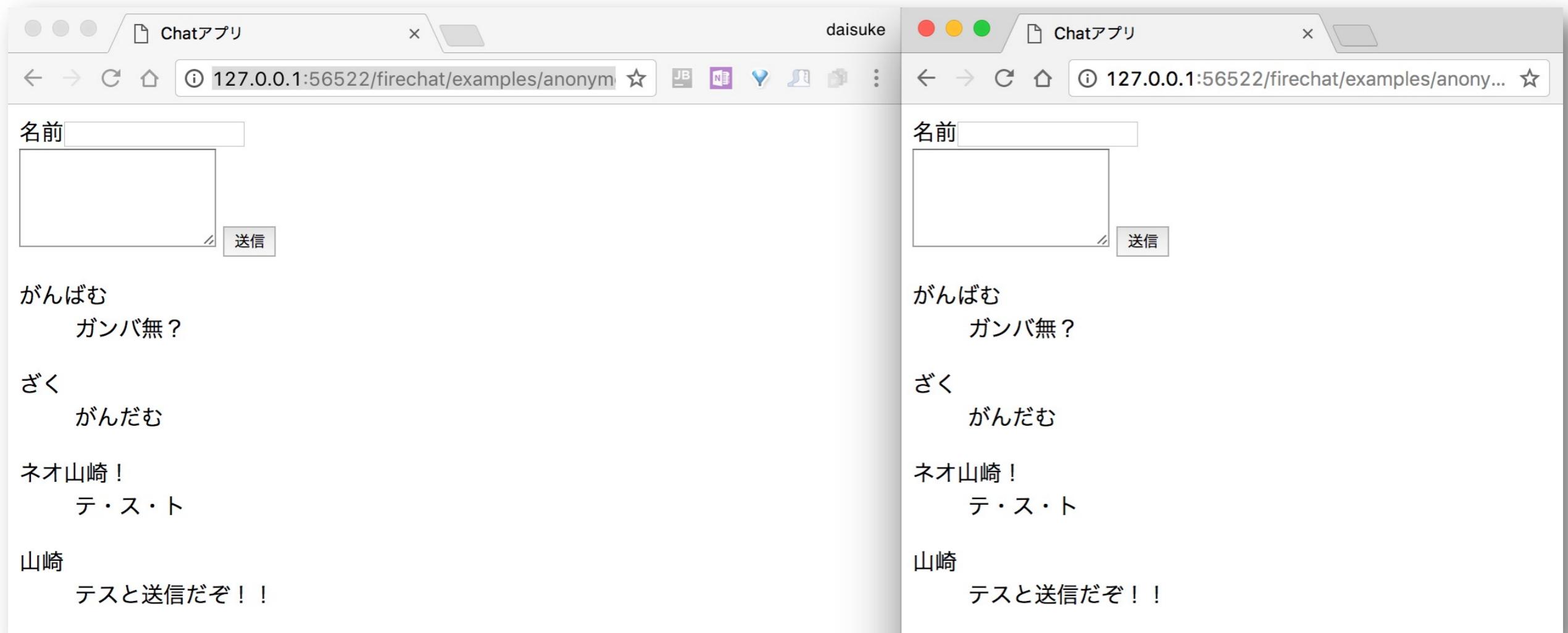
```
56 //最初にデータ取得 & onSnapshotでリアルタイムにデータを取得
57 onChildAdded(dbRef, function(data){
58     const msg = data.val();
59     const key = data.key;
60     let h = '<p>';
61     h += msg.uname;
62     h += '<br>';
63     h += msg.text;
64     h += '</p>';
65     $("#output").append(h); //#outputの最後に追加
66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

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。



# 管理画面→「Database」を表示

データ構造を見ることが出来ます。

The screenshot shows the Firebase Realtime Database interface. On the left, a sidebar lists various services: Overview, Analytics, Authentication, **Database**, Storage, Hosting, Functions, Test Lab, Crash Reporting, and Performance. The 'Database' item is highlighted with a red box and has a red circle with the number 1 next to it. A large red arrow points from this circle to the database root node 'chatapp-1aa3a' on the right. The root node is also highlighted with a red box and has a red circle with the number 2 next to it. Below the root node, there are several child nodes, each represented by a plus sign icon. The fourth child node, '-KkNr9YkYPifW\_gIVeV1', is highlighted with a red box and has a red circle with the number 3 next to it. This node has two children: 'text' and 'username'. The 'text' child contains the value 'ガンバ無?\n' and the 'username' child contains the value 'がんばむ'.

```
chatapp-1aa3a
├── -KitDXv-QN26PLZITReo
├── -KitDn9IEVX3TNDmYqG1
├── -KkNqu-RS8cEuP6YSR7F
└── -KkNr9YkYPifW_gIVeV1
    ├── text: "ガンバ無?\n"
    └── username: "がんばむ"
```

# 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: "Enter", char: "\u000d", keyCode: 13, which: 13, location: 1, repeat: false, shiftKey: false, altKey: false, ctrlKey: false, metaKey: false, bubbles: true, cancelable: true, target: this, relatedTarget: null, eventPhase: 2, isDefaultPrevented: false, isPropagationStopped: false, isTrusted: true},  
  relatedTarget: undefined  
  shiftKey: false  
  ▶ target: textarea#text}
```

eがキモ！

keyCode:13がEnter  
Enterを押したら  
送信を作つて見よう！

# そしてconsole.logがキモ

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

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？