

JavaScript - Firebase を使う

文法 - オブジェクト

オブジェクト (Object)

▼ オブジェクトとは

【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



▼ 分割代入

```

const post = {
  title: "投稿タイトル",
  text: "本文"
};

// 分割代入
const {title, text} = post;
// title に post.title, text に post.text の中身がそれぞれ代入される
console.log(title); // "投稿タイトル"
console.log(text); // "本文"

```

分割代入 - JavaScript | MDN

分割代入 (Destructuring assignment) 構文は、配列から値を取り出して、あるいはオブジェクトからプロパティを取り出して別個の変数に代入することを可能にする JavaScript の式です。 オブジェクトリテ

 https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment



Realtime Database (Firebase)

概要・初期設定

※ 2021.12.05 時点のデータを基にしています

▼ Firebaseとは

- Googleが提供している BaaS (Backend as a Service)
- 開発者がWebサーバやデータベースを用意せずともそれに準ずる機能を使うことができる
- 主なFirebaseのサービス
 - Realtime Database (JSON形式でのデータ保存・同期)
 - 今回使うのはこれ！
 - Firebase Authentication (認証・ログイン機能)
 - Cloud Storage for Firebase (画像や動画ファイルの保存)
 - Cloud Firestore (サーバーレスアプリデータ保存)
 - Firebase Hosting (Webサーバー)
- ... などなど

▼ プロジェクトを作成

<https://firebase.google.com> にアクセスし、「使ってみる」を押下



プロジェクトの作成を押下



プロジェクト名を入力し、「続行」を押下



Google Analytics を オフに設定し、「プロジェクトを作成」を押下



プロジェクトが作成されるまで待機



作成されたら「続行」を押下



作成したプロジェクトが表示されたらOK！



▼ アプリを作成

</> ボタンを押下



▼ [Q&A]↑のページが行方不明になりました

<https://console.firebaseio.google.com> から作成したプロジェクトを開き直すと表示されます



アプリのニックネームを "chat" で登録し、「アプリを登録」を押下

× ウェブアプリへの Firebase の追加

1 アプリの登録

アプリのニックネーム ②
chat

このアプリの Firebase Hosting も設定します。 詳細 ▾
Hosting 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

<script> タグを使用する を選択肢、コードをコピーする

× ウェブアプリへの Firebase の追加

1 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 [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/9.6.0/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: "AIzaSyDmXmb44433wC8bouBVTs7SiTDJ9kSigMA",
    authDomain: "fir-demo-962b2.firebaseioapp.com",
    projectId: "fir-demo-962b2",
    storageBucket: "fir-demo-962b2.appspot.com",
    messagingSenderId: "767762019644",
    appId: "1:767762019644:web:8774d86f7a4062515a9a39"
  };

  // 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](#)

▼ [Q&A] コードのコピー忘れてしまいました... !

設定 > プロジェクトの設定



プロジェクトの概要 [⚙️](#)

stress-check

ドキュメントに移動 [×](#)

ログイン [X](#)

構築

Authentication

プロジェクトの設定 [機能、リサーチ、イベントに関する最新情報のメールを受け取る](#)

ユーザーと権限

使用量と請求額

stress-check [Spark プラン](#)

マイアプリ > ウェブアプリ から作成したアプリを選択後 SDK の設定と構成で [CDN](#) を選択

マイアプリ

ウェブアプリ

chat Web App

アプリのニックネーム
chat

アプリ ID
1:767762019644:web:8774d86f7a4062515a9a39

Firebase Hosting サイトへリンク

SDK の設定と構成

npm CDN 構成

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やってください。

```
<script type="module">
  // Import the functions you need from the SDKs you need
  import { initializeApp } from "https://www.gstatic.com/firebasejs/9.6.0/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: "AIzaSyDmX64433wC8bouVTs7S1TDj9kS1gMA",
  }
```

▼ sample.html にスクリプトを準備

アプリ作成時にコピーしたプログラムを

<!--*--> 以下Firebase *--> 以下に貼り付ける

```
28 <!--*-->
29 <script type="module">
30   // Import the functions you need from the SDKs you need
31   import { initializeApp } from "https://www.gstatic.com/firebasejs/9.6.0/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 = {
```

使いたいサービスのSDK(便利ライブラリ)を読み込む

▼ [補足]SDKとは？

- Software Development Kit の略
- ソフトウェアやサービスを開発するときに便利なプログラムやAPI、サンプルコードをひとまとめにしたもの
- 要するに開発のお助けキット

今回はRealtime DB 関連ライブラリを読み込む

```
import { getDatabase, ref, push, set, onChildAdded, remove, onChildRemoved } from "https://cdnjs.cloudflare.com/ajax/libs.firebaseio/9.6.0.firebaseio-database.min.js";
```

JavaScript でのインストールと設定 | Firebase Documentation

Firebase Realtime Database はクラウドホスト型データベースです。データは JSON として保存され、接続されたすべてのクライアントとリアルタイムに同期されます。Android、Apple プラットフォーム、JavaScript SDK を使用してクロスプラットフォーム アプリを構築した場合、すべてのクライアントが、1 つの Realtime Database

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

Realtime Database の構築

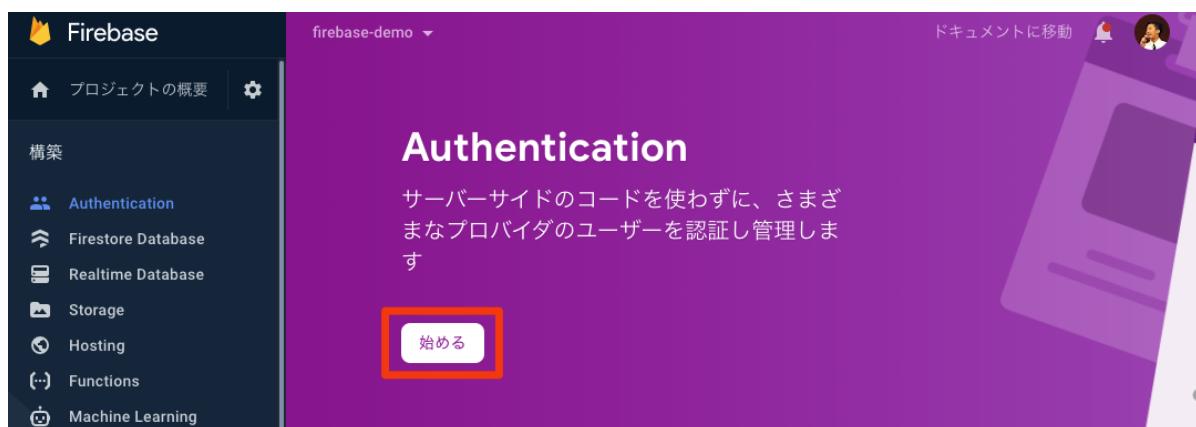
▼ 認証情報を設定

今回は誰でも書き込みが可能な設定にする

Authentication を選択



「始める」を押下



Sign-in method タブを選択後 「匿名」を押下

Firebase

プロジェクトの概要

構築

- Authentication
- Firestore Database
- Realtime Database
- Storage
- Hosting
- Functions
- Machine Learning

リリースとモニタリング

- Crashlytics
- Performance
- Test Lab
- App Distribution

Authentication

Sign-in method

Templates Usage

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
- 名前

追加のプロバイダ

- Google
- Facebook
- Play ゲーム
- Game Center
- Apple
- Github
- Microsoft
- Twitter
- Yahoo!

「有効にする」を選択後「保存」を押下

プロジェクトの概要

構築

- Authentication
- Firestore Database
- Realtime Database
- Storage
- Hosting
- Functions
- Machine Learning

リリースとモニタリング

- Crashlytics
- Performance
- Test Lab

Authentication

Sign-in method

Templates Usage

ログイン プロバイダ

名前

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](#)

有効にする

保存

匿名プロバイダが有効になっていることを確認

プロジェクトの概要

構築

- Authentication
- Firestore Database
- Realtime Database
- Storage
- Hosting
- Functions
- Machine Learning

Authentication

Sign-in method

Templates Usage

ログイン プロバイダ

新しいプロバイダを追加

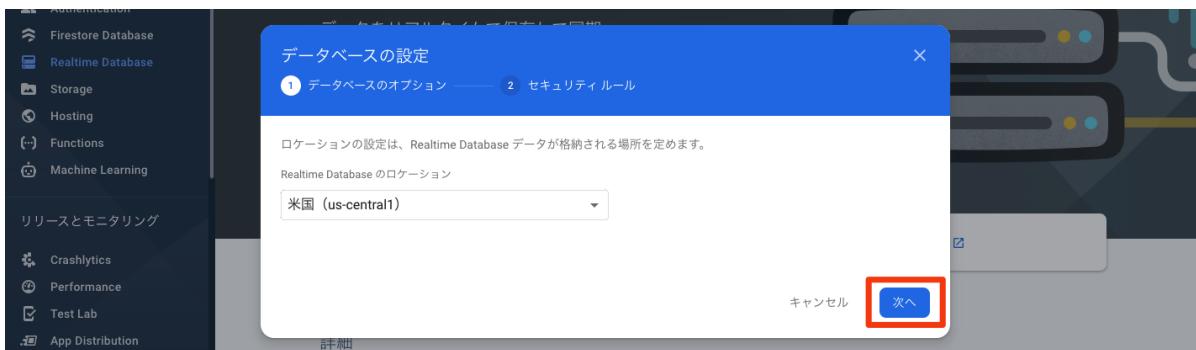
| プロバイダ | ステータス |
|-------|-------|
| 名前 | 有効 |

▼ Realtime Databaseの構築

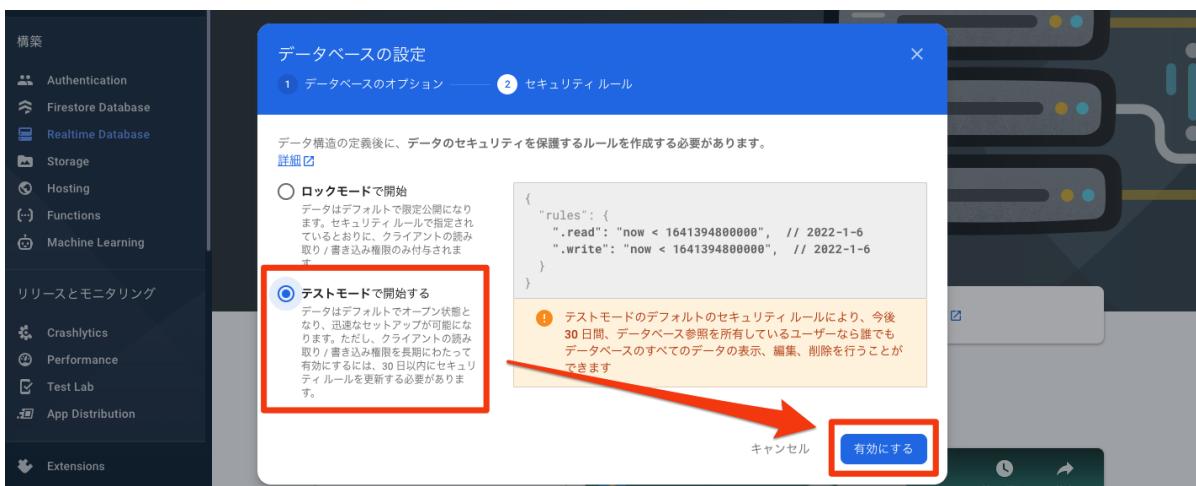
Realtime Database を選択後 「データベースを作成」 を押下



ロケーションは「米国」のまま「次へ」を押下



セキュリティルールを「テストモード」に設定し、「有効にする」を押下



▼ [Q&A] セキュリティルールとは

- 作成したデータベースに誰がアクセスしていいか、という設定ができる機能
- セキュリティ設定の例
 - 特定のIP・ドメインからのアクセスのみ許可
 - 認証済みユーザのみアクセス許可
- テストモードにすると全てのアクセスを許可する状態にする
 - Realtime Databaseの機能を試してみるにはとても便利な機能である一方、セキュリティ的には脆弱
 - 従ってテストモードで運用する際にはAPIキーは全地に公開してはいけない

Reatime Databaseの作成が完了

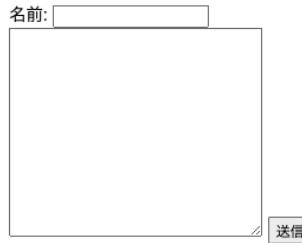
The screenshot shows the Firebase Realtime Database console for a project named "firebase-demo". The left sidebar lists various services: Authentication, Firestore Database, Realtime Database (which is selected), Storage, Hosting, Functions, and Machine Learning. The main area is titled "Realtime Database" and contains tabs for "データ" (Data), "ルール" (Rules), "バックアップ" (Backup), and "使用状況" (Usage). A banner at the top right says "請求詐欺やフィッシングなどの不正行為から Realtime Database のリソースを保護します" and "App Check を構成する". Below the tabs, there's a URL field with "https://fir-demo-962b2-default-rtdb.firebaseio.com/" and a preview pane showing "fir-demo-962b2-default-rtdb: null".

チャットアプリの作成

▼ 送信画面UIの作成

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

完成イメージ



▼ Realtime Databaseの初期設定

```
const db = getDatabase(app); // RealtimeDBに接続
const dbRef = ref(db, "chat"); // RealtimeDB内の"chat"を使う
```

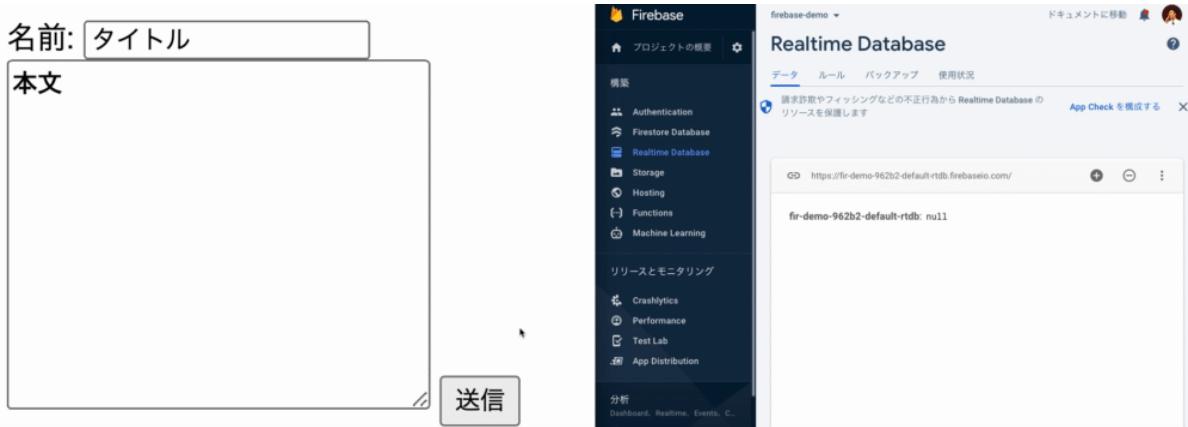
▼ データ送信機能の実装 (push , set)

送信ボタンをクリックしたら `#uname` , `#send` 内のテキスト情報を取得する

```
$("#send").on("click", function () {
  const uname = $("#uname").val();
  const text = $("#text").val();
  console.log(uname, text); // 取得確認
});
```

データ送信処理を記述する

```
$("#send").on("click", function () {
  const msg = {
    uname: $("#uname").val(),
    text: $("#text").val()
  }
  const newPostRef = push(dbRef); // pushできる状態にする
  set(newPostRef, msg); // DBに値を送信
});
```



▼ データ受信機能の実装 (`onChildAdded`)

`<p>タイトル
本文</p>` 形式のHTMLを作成し、`#output` に追加していく

```
onChildAdded(dbRef, function (data) {
  const msg = data.val();
  const key = data.key;
  let h = "<p>";
  h += msg.uname;
  h += "<br>";
  h += msg.text;
  h += "</p>";
  $("#output").append(h); // #output の最後に追加
});
```

- `onChildAdded()`
 - 既存アイテムのリストを取得した後、アイテムリストへの追加がある度に発火する
- `data` は `DataSnapshot`型のオブジェクトが取れている

[DataSnapshot | JavaScript SDK | Firebase](#)

Reference for DataSnapshot

🔥 <https://firebase.google.com/docs/reference/node/firebase.database.DataSnapshot#val>

Fireba

▼ EnterKeyで送信(自走課題)

Enter Keyで送信する方法（ヒント！）

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  
jQuery113088  
key: "Enter"  
keyCode: 229  
metaKey: false  
originalEvent: KeyboardEvent {isTrusted: true, key: "Enter", code: "Enter",  
relatedTarget: undefined  
shiftKey: false  
target: textarea#text}}
```

e がキモ！

keyCode: 13 が Enter

Enterを押したら
送信を作って見よう！

ヒント！！
送信処理はクリック送
信と一緒に！！



課題: LINE風アプリの作成

最低限機能

- メッセージ表示領域を超えた処理

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

- 削除機能

- 見た目の装飾

更にあると良いと思われる機能

アイコン、メッセージ翻訳、対戦ジャンケン、
メモ帳をクラウド保存 … などなど

⚠️課題提出時の注意



APIキーはgithubにアップロードしないでください！！