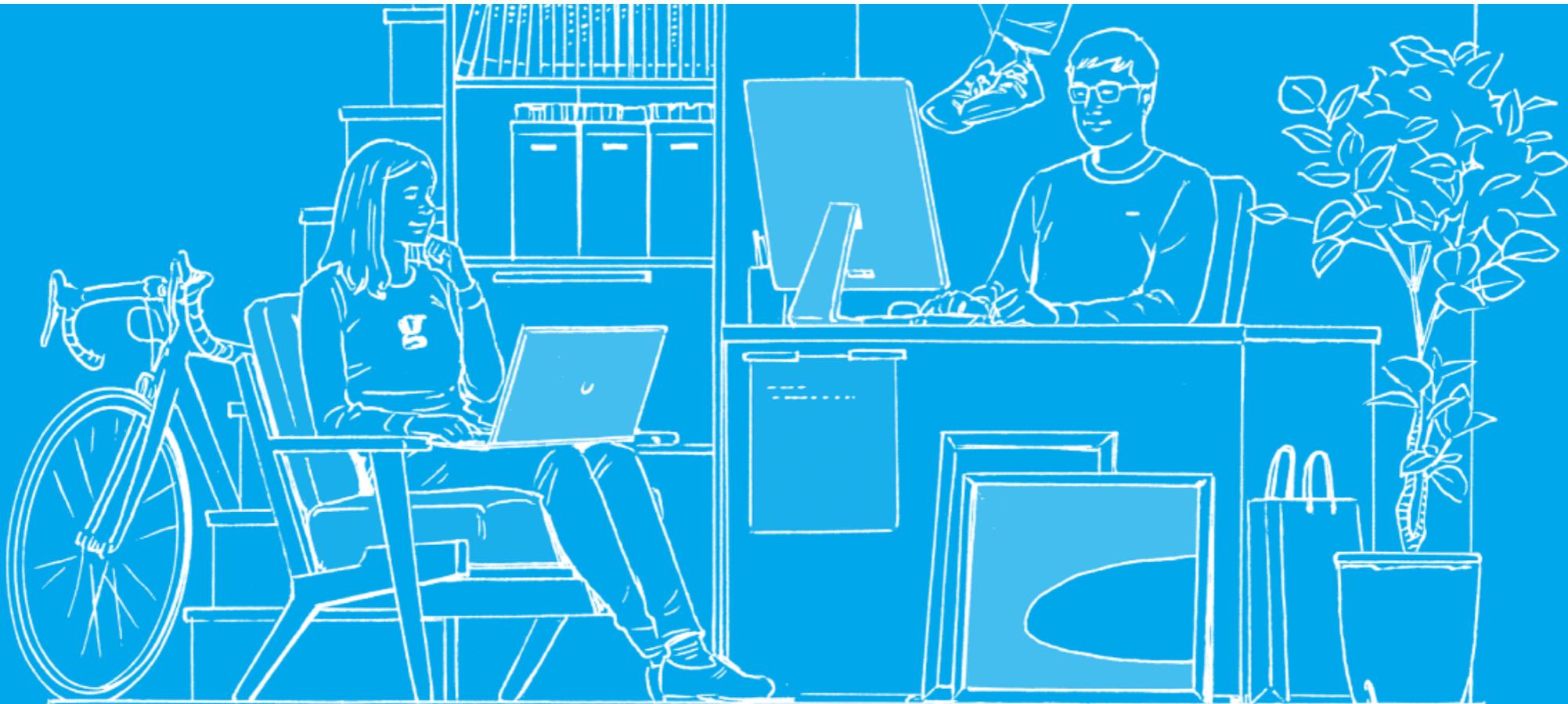




**G's ACADEMY**  
**TOKYO**

# Firebase

Googleアカウントが必要です



# アジェンダ

---

- ❖ オブジェクト

- FireBase

- 課題実習タイム (Chatアプリ or リアルタイム通信)

- ❖ チュータリングタイム

# オブジェクト

- Object -

# JavaScriptのオブジェクト ( js03/ob1.html )

## 【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ています。

```
<script>
const object1 = { a:"山崎", b:"鈴木", c:"中村" };

const object2 = {
    obj1: { c1:"A", c2:"B", c3: "C" },
    obj2: { c1:"A", c2:"B", c3: "C" }
};
</script>
```

## 【オブジェクトの参照イメージ】

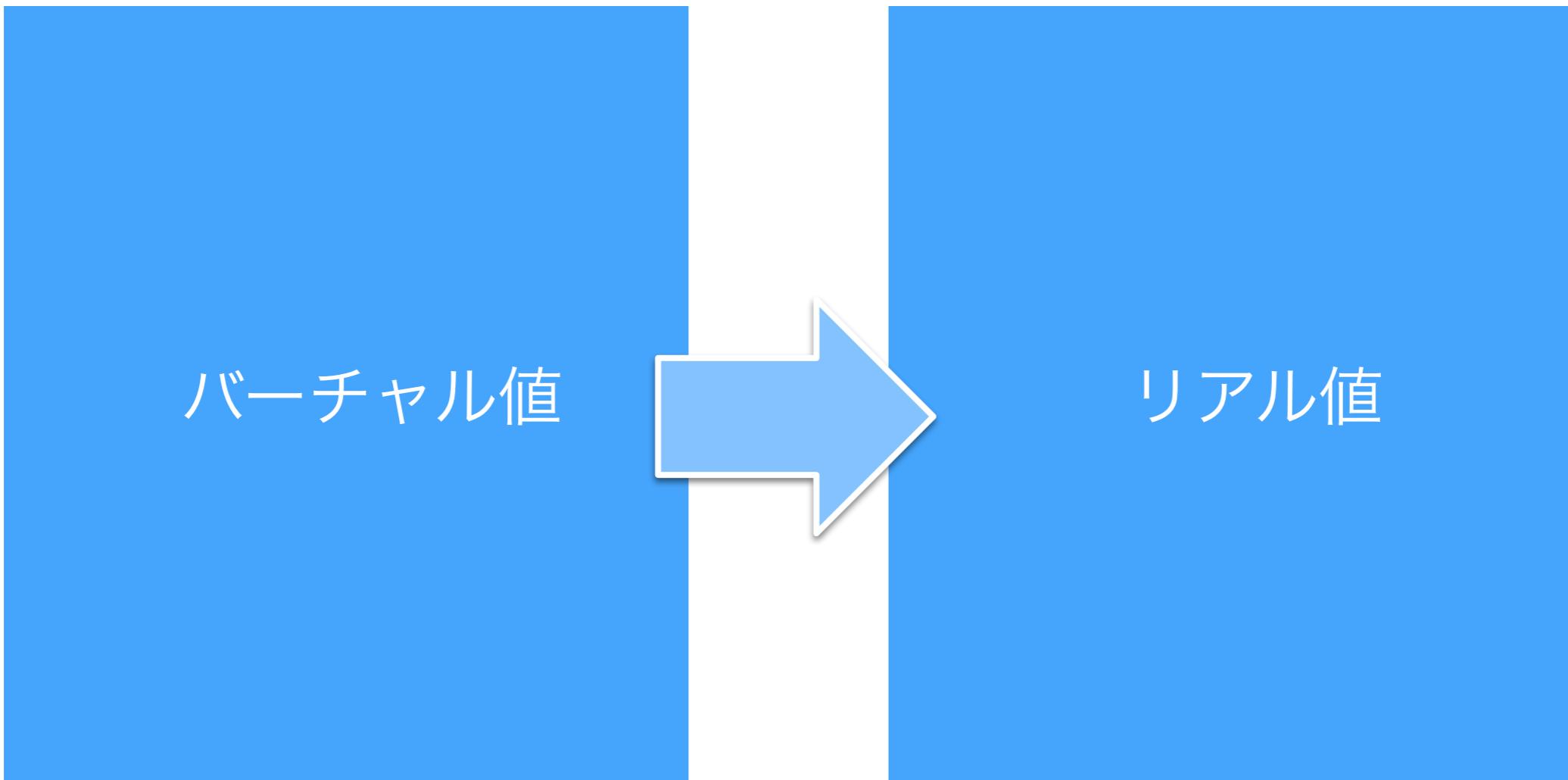
object1オブジェクト内のプロパティに値が格納されます。object1の中の値を取得する方法は、「object1.プロパティ名」で取得可能



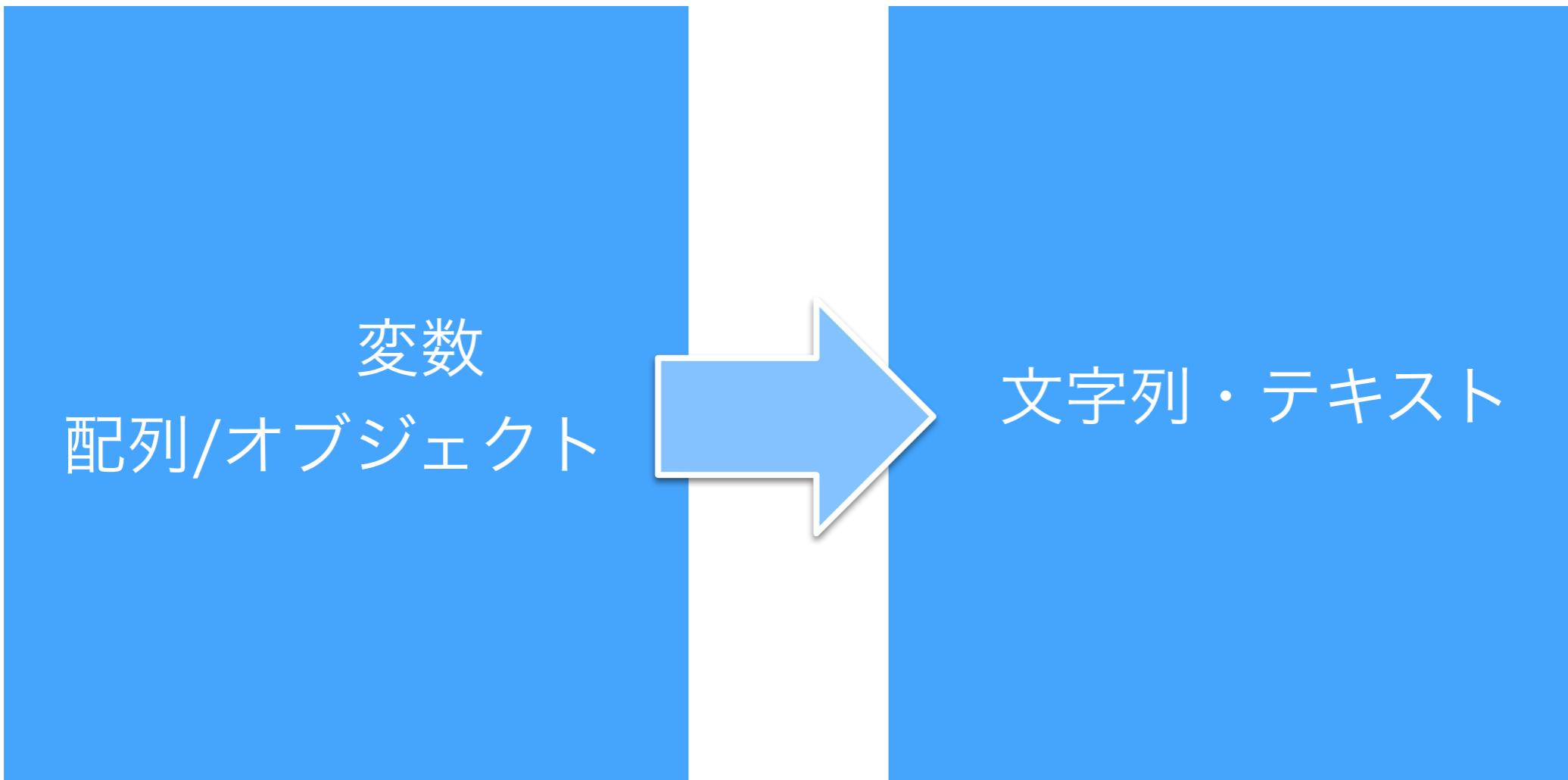
# JSON

配列・オブジェクトを保存したい

# JSONとは



# JSONとは



オブジェクトをまんま文字列に変換したもの

# JSON とは何か？どう使うのか？

～配列/オブジェクトは生データ→文字変換～

パソコン:ブラウザ上

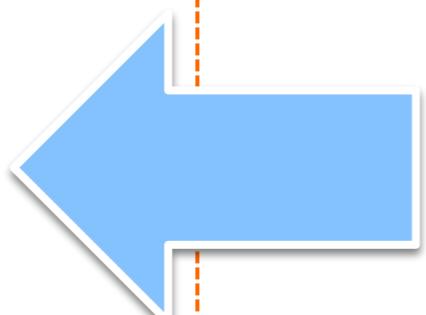
①変数  
`let data =`

```
[{"fname": "Tarou", "lname": "Yamazaki"},  
 {"fname": "Anna", "lname": "Smith"},  
 {"fname": "Peter", "lname": "Jones"}]
```

②JSON形式の文字列に変換

```
let json_text = JSON.stringify(data);
```

④データ保存



③json\_text (JSON形式の文字列)

```
[  
 {"fname": "Tarou", "lname": "Yamazaki"},  
 {"fname": "Anna", "lname": "Smith"},  
 {"fname": "Peter", "lname": "Jones"}]  
]
```

※変数では他言語にデータを渡せない→JSON形式の”文字列”に変換することで可能になる！

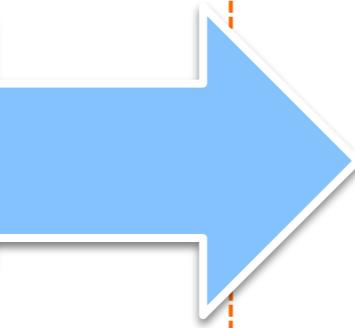
# javascriptのJSON

## 【 JSONデータ (文字列からオブジェクトに変換)】

### ⑤データ取得



Database,  
LocalStorage



```
let json_text = [  
    {"fname":"Tarou","lname":"Yamazaki"},  
    {"fname":"Anna","lname":"Smith"},  
    {"fname":"Peter","lname":"Jones"}]
```

JSON形式の文字列

## 【 JSON文字列 → 配列/オブジェクト 変数に変換】

```
let data = JSON.parse(json_text); //JSONからオブジェクトに変換
```

```
console.log(data);
```



## 【 オブジェクトへのデータ参照方法 】

data

data[0].fname

Yamazaki

data[1].fname

Anna

data[2].fname

Peter

# コード例：JSON文字列変換 (js03/json.html)

```
13 <script>
14     // 1. a配列を作成
15     const a = [];
16
17     // 2. オブジェクト変数を3個作成
18     const ob1 = { name: "yama", tel: "090-0000-0000" };
19     const ob2 = { name: "sasa", tel: "090-0000-0000" };
20     const ob3 = { name: "suzu", tel: "090-0000-0000" };
21
22     // 3. 配列にpush(追加)する
23     a.push(ob1);
24     a.push(ob2);
25     a.push(ob3);
26     console.log("オブジェクト変数：" , a);
27
28     // 4. JSON.stringifyを使いJSON文字列へ変換
29     const b = JSON.stringify(a);
30     console.log("JSON文字列：" + b);
31
32     // 5. JSON.parseを使いオブジェクト変数へ戻す
33     const c = JSON.parse(b);
34     console.log("オブジェクト変数：" , c);
35 </script>
```

# 自作チャット作成方法

## 初級編



# Key取得

Chromeブラウザでアクセス

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

The screenshot shows the official Firebase website homepage. At the top, there is a navigation bar with links for 'Firebase' (with a yellow flame icon), 'プロダクト', '使用例', '料金', 'ドキュメント', and 'サポート'. To the right of the navigation bar are a search bar ('検索'), a 'Language' dropdown, a 'コンソールへ移動' link, and a 'ログイン' link, which is underlined and has a red arrow pointing to it. The main content area features a large blue background image of two people interacting with a wall covered in floating blue diamond shapes. Overlaid on this image is Japanese text: 'Firebase で モバイルとウェブ アプリを成功させる'. Below this text are two buttons: '使ってみる' and '動画を見る'. A white callout box at the bottom left contains the text: 'アプリをすばやく作成、インフラストラクチャの管理は不要' and 'Firebase は、機敏に行動してユーザーに対応す'. Another callout box at the bottom center contains the text: 'Google のインフラが支える、多数の人気アプリが信頼するサービス'. A third callout box at the bottom right contains the text: '連動する全プロダクトを 1 つのプラットフォームで管理' and 'Firebase のプロダクトはそれぞれ単独でも十分'.

 Google

## ログイン

Google Developers に移動する

[メールアドレスを忘れた場合](#)

[グーグルアカウントでログインする](#)

ご自分のパソコンでない場合は、ゲストモードを使用して非公開でログインしてください。 [詳細](#)

[アカウントを作成](#)

[次へ](#)

日本語 ▾

ヘルプ

プライバシー

規約

# 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](#)

ログインできると  
コンソールに移動をクリック



アプリをすばやく作成、インフ  
ラストラクチャの管理は不要

Firebase は、機敏に行動してユーザーに対応す  
るための、アナリティクス、データベース、メ

Google のインフラが支える、多  
数の人気アプリが信頼するサー  
ビス

Firebase は Google のインフラストラクチャ上

連動する全プロダクトを 1 つの  
プラットフォームで管理

Firebase のプロダクトはそれぞれ単独でも十分  
に役立ちますが、データや分析情報は共有され

# プロジェクトを追加をクリック

Firebase プロジェクト

+  
プロジェクトを追加

chatapp  
chatapp-f5ea2

デモ プロジェクトを見る



Firebase プロジェクトがアプ  
リのコンテナとなります

プロジェクト内のアプリは、Realtime  
Database やアナリティクスなどの機能を共  
有しています。

[Learn more](#)



[サポート](#) · [利用規約](#) · [プライバシー ポリシー](#)

まずプロジェクトに名前を付  
けましょう

プロジェクト名



[Firebase の規約に同意します](#)

**i** あと 3 個のプロジェクトでプロジェクト数の上限に達します。既存のプロジェクトに Firebase を追加するか、上限の引き上げをリクエストすることを検討してください。

[上限の引き上げをリクエスト](#)

続行

# Google アナリティクス (Firebase プロジェクト向け)

Google アナリティクスでは、iOS と Android アプリ向けの無料かつ無制限の分析を利用できます。

次のサービスで、セグメンテーション、ターゲティング、および[その他の機能](#)が有効になります:

-  Crashlytics [?](#)
-  Cloud Messaging [?](#)
-  アプリ内メッセージング [?](#)
-  Remote Config [?](#)
-  Cloud Functions [?](#)

プロジェクト用に Google アナリティクスを設定する  
次のステップで構成します

今は設定しない  
後で変更できます

以下のサービスにはアナリティクスが必要です:

-  A/B Testing [?](#)
-  Predictions [?](#)

[前へ](#)

[プロジェクトを作成](#)

しばらく待機するとプロジェクトができます



続行をクリック



# 「</>」を選択 (Webアプリ)



# 6. アプリ名「chat」と登録。

× 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 **今日はChatにしておく**

chat

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

Firebaseに接続する大事にKEYになります。



アプリの登録



Firebase SDK の追加

## 赤枠全部コピー

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/6.6.1.firebaseio.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: "AIzaSyDQLnYrg96JQ_LHTkYzM02WBUsWCM0if9c",
    authDomain: "camp-dhu.firebaseio.com",
    databaseURL: "https://camp-dhu.firebaseio.com",
    projectId: "camp-dhu",
    storageBucket: "",
    messagingSenderId: "28465852467",
    appId: "1:28465852467:web:2f212d9c12647db97a7346"
  };
  // 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

# chatフォルダのsimple.htmlを開く

```
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>
```

ここからエディターにコードを記述します

コピーしたスクリプトを「貼り付け」ます。  
場所は「</body>」の上にします。

## simple.html

```
22
23  <!-- JQuery -->
24  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/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
```

# simple.htmlに記述

```
<!-- JQuery -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<!-- JQuery -->

<!--** 以下Firebase **-->

<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/6.6.1.firebaseio.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: "AIzaSyDQLnYrg96JQ_LHTkYzM02WBUaWCM0if9c",
    authDomain: "camp-dhu.firebaseio.com",
    databaseURL: "https://camp-dhu.firebaseio.com",
    projectId: "camp-dhu",
    storageBucket: "",
    messagingSenderId: "28465852467",
    appId: "1:28465852467:web:2f212d9c12647db97a7346"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

**firebaseのコード  
をコピペで貼り付け**

# 重要POINT！

## コピーしたコードの一部を削除

```
<script src="https://www.gstatic.com/firebasejs/5.9.1.firebaseio-app.js"></script>
```

「-app」の文字を削除

```
<script src="https://www.gstatic.com/firebasejs/5.9.1.firebaseio.js"></script>
```

解説

デフォルト 「firebase-app.js」 では以下のようにコアライブラリのみ読み込んでいてdatabase.jsなどたの処理は入っていない。そのため授業では、全部入り 「firebase.js」 を読み込んで使う。

```
<script src="https://www.gstatic.com/firebasejs/5.9.1.firebaseio.js"></script>
```

!-- Add additional services that you want to use -->

```
<script src="https://www.gstatic.com/firebasejs/5.9.1.firebaseio-auth.js"></script>
<script src="https://www.gstatic.com/firebasejs/5.9.1.firebaseio-database.js"></script>
<script src="https://www.gstatic.com/firebasejs/5.9.1.firebaseio-firebasestore.js"></script>
<script src="https://www.gstatic.com/firebasejs/5.9.1.firebaseio-messaging.js"></script>
<script src="https://www.gstatic.com/firebasejs/5.9.1.firebaseio-functions.js"></script>
```

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

# Chat設定



アプリの登録



Firebase SDK の追加

## 赤枠全部コピー

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/6.6.1.firebaseio.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: "AIzaSyDQLnYrg96JQ_LHTkYzM02WBUaWCM0if9c",
    authDomain: "camp-dhu.firebaseio.com",
    databaseURL: "https://camp-dhu.firebaseio.com",
    projectId: "camp-dhu",
    storageBucket: "",
    messagingSenderId: "28465852467",
    appId: "1:28465852467:web:2f212d9c12647db97a7346"
  };
  // 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

コンソールに進むをクリック

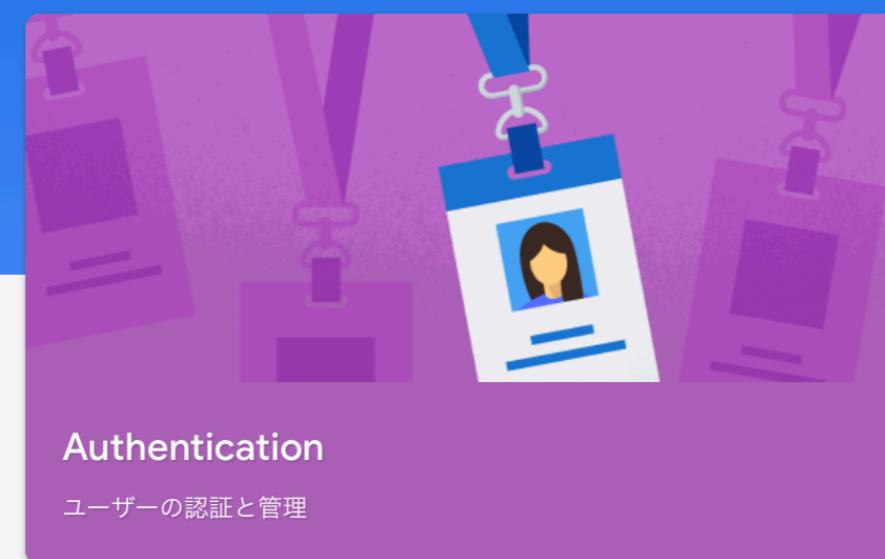
# コンソール画面

作成した  
名前になっているか確認！

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同期

×



すべての 開発 の機能を表示

アプリの品質をチェックする

×



「Authentication」→「ログイン方法」→「匿名」の順番に選択

The screenshot shows the Firebase console interface for a project named "camp-dhu". The left sidebar includes links for Project Overview, Authentication (which is selected and highlighted in red), Database, Storage, Hosting, Functions, ML Kit, Crashlytics, Performance, Test Lab, Analytics, and Predictions. The main content area is titled "Authentication" and has tabs for "User", "Sign-in methods", "Template", and "Usage status". The "Sign-in methods" tab is active, indicated by a red underline. On the right, a table lists various sign-in providers: Mail / Password (inactive), Phone number (inactive), Google (inactive), Play Games (inactive), Game Center (Beta), Facebook (inactive), Twitter (inactive), GitHub (inactive), Yahoo! (inactive), Microsoft (inactive), and Anonymous (inactive). The "Anonymous" provider is highlighted with a red underline.

プロバイダ	ステータス
メール / パスワード	無効
電話番号	無効
Google	無効
Play ゲーム	無効
Game Center (Beta)	無効
Facebook	無効
Twitter	無効
GitHub	無効
Yahoo!	無効
Microsoft	無効
匿名	無効

# 「有効にする」 → 「保存」 の順番でクリック



# 「Database」 → 「データベースの作成」

The screenshot shows the Firebase Project Overview interface for a project named "camp-dhu". The left sidebar lists various services: Authentication, Database (which is selected and highlighted in blue), Storage, Hosting, Functions, and ML Kit. The main content area is titled "Cloud Firestore" and describes it as "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 large red arrow points from the sidebar's "Database" link to a white button labeled "データベースの作成" (Create Database). Below this, a message says "Cloud Firestore が準備できました" (Cloud Firestore is ready) next to a blue circular icon.

Firebase

camp-dhu ▾

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, Au...

拡大

Predictions, A/B Testing, Cloud Mes...

Cloud Firestore

クエリと自動スケーリング機能を備えた次  
代の Realtime Database

データベースの作成

Cloud Firestore が準備できました

# 「テストモード」に設定

データベースの作成

1 Cloud Firestore のセキュリティ 保護ルール 2 Cloud Firestore のロケーション を設定します

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  
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

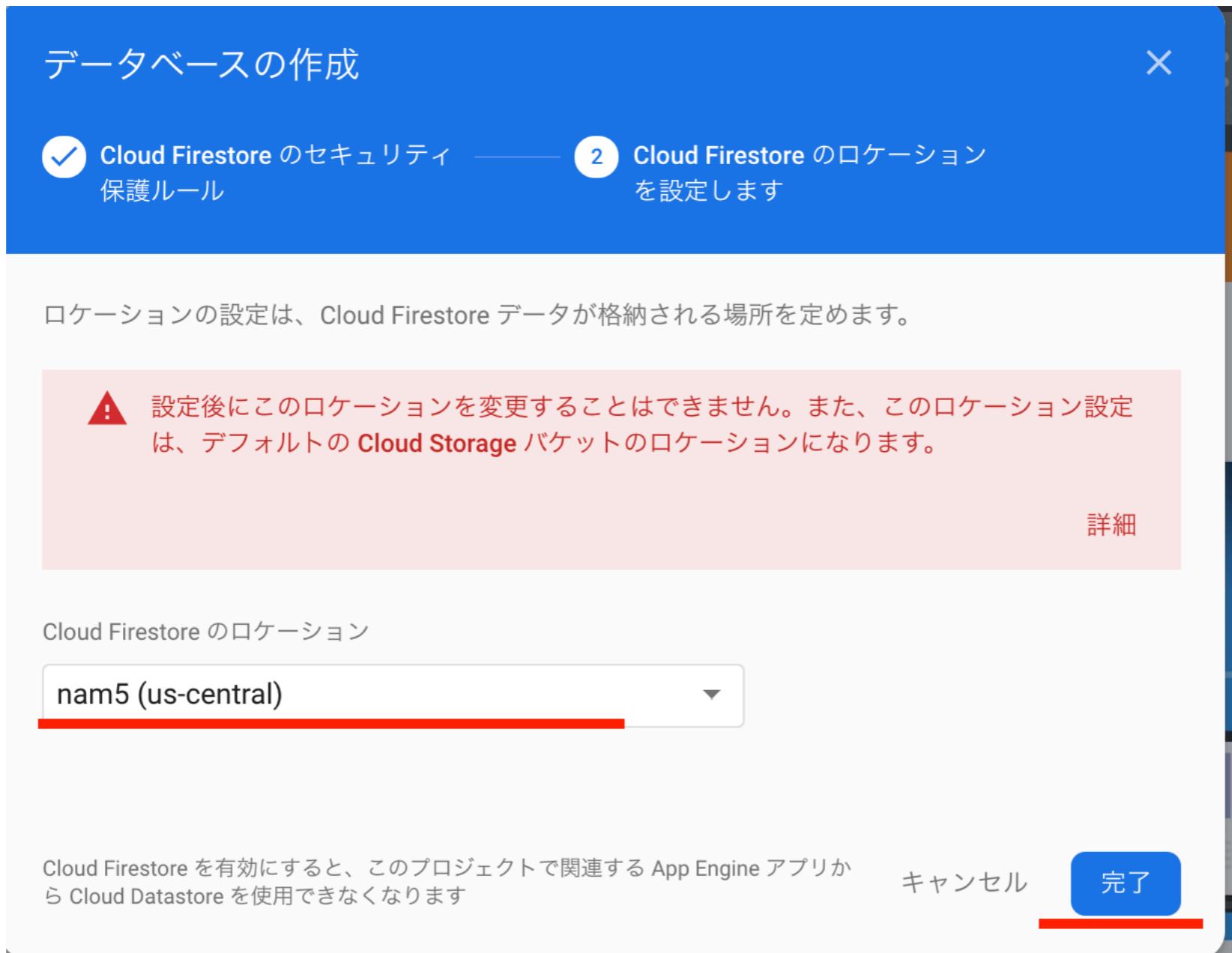
```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write;
    }
  }
}
```

! 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

Cloud Firestore を有効にすると、このプロジェクトで関連する App Engine アプリから Cloud Datastore を使用できなくなります

キャンセル 次へ

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。



# Database

Realtime Database

データ

ルール

バックアップ

使用状況

ここをクリック

<https://camp-dhu.firebaseio.com/>

camp-dhu: nul



Realtime Database

接続されているすべてのクライアントでデータの保存と同期をリアルタイムで実行します



Cloud Firestore

クエリと自動スケーリング機能を備えた次世代の Realtime Database



これを選んでクリック



公開されていない変更 | 公開 破棄

```
1 {  
2   /* Visit https://firebase.google.com/docs/database  
3   "rules": {  
4     ".read": true,  
5     ".write": true  
6   }  
7 }
```

falseからtrueに

公開をクリック

# 画面作成

```
..<div>
..  ..<div>
..    ..<!-- 名前 -->
..    ..名前
..    ..<input type="text" id="username">
..  </div>

..  ..<div>
..    ..<!-- テキストエリア -->
..    ..テキスト
..    ..<textarea name="" id="text" cols="30" rows="10"></textarea>
..  </div>

..  ..<div>
..    ..<!-- 送信ボタン -->
..    ..<button id="send">送信</button>
..  </div>

..  ..<div>
..    ..<!-- 保存されたデータが表示される箇所 -->
..    ..<div id="output"></div>
..  </div>

..</div>
```

# Chat処理記述

# リアルタイム通信の記述をしましょう。

```
<> simple.html •  
  
<> simple.html > ⚒ html > ⚒ body > ⚒ script  
48   · · · <!-- TODO: Add SDKs for Firebase products that you want  
49   · · ·     https://firebase.google.com/docs/web/setup#config-wel  
50  
51   · · <script>  
52   · · · // Your web app's Firebase configuration  
53   · · · var firebaseConfig = {  
54   · · ·     apiKey: "AIzaSyDQLnYrg96JQ_LHTkYzM02WBUaWCM0if9c",  
55   · · ·     authDomain: "camp-dhu.firebaseio.com",  
56   · · ·     databaseURL: "https://camp-dhu.firebaseio.com",  
57   · · ·     projectId: "camp-dhu",  
58   · · ·     storageBucket: "",  
59   · · ·     messagingSenderId: "28465852467",  
60   · · ·     appId: "1:28465852467:web:2f212d9c12647db97a7346"  
61   · · };  
62   · · // Initialize Firebase  
63   · · firebase.initializeApp(firebaseConfig);  
64  
65   · · const newPostRef = firebase.database().ref();  
66
```

const newPostRef = firebase.database().ref();

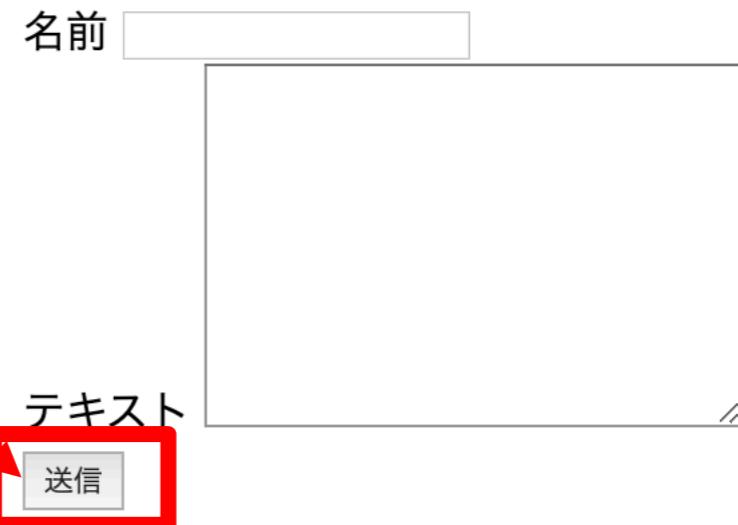
のref();を追加することでリアルタイムに通信するようになります。

# 送信ボタンのクリックイベントを作成

今回jQueryで操作する対象のID要素（操作する対象）

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

#send  
#username  
#text  
#output



送信を押したら[click]イベントを発動させる

```
const newPostRef = firebase.database().ref();  
送信を押したらクリックイベント  
$("#send").on("click", function () {  
    // ここにfirebaseに保存させる処理を記述  
})
```

# データをfirebaseに保存する

記述式)

```
送信オブジェクト.push({  
    送信プロパティ名: 値,  
    送信プロパティ名: 値  
});
```

```
.... $("#send").on("click", function(){  
....     newPostRef.push(  
....         username:$("#username").val(),  
....         text: $("#text").val()  
....     })  
....     $("#text").val("");  
.... })
```

# firebaseに保存されているデータを確認

## Database Realtime Database

データ ルール バックアップ 使用状況

https://camp-dhu.firebaseio.com/

【ここが作成した名前になる】

- Lorj4HHPBUNgNEIVXGt
  - text: "テキスト保存チェック中"
  - username: "てすと"
- LorkHcshagHDyr2Pjbz
  - text: "ててててて"
  - username: "てすと"

データが送られるとここに保存されていきます！ 🙌

# 21.データ受信：処理を記述します。

onメソッドの 'child\_added' イベントを取得し、受信処理を行います。

※ child\_added はFirebaseが用意してるものなので、それを用いて受信をしていきます  
(最初はイメージわからないかもしれません、こういうものだと思ってください)

```
newPostRef.on("child_added", function (data) {  
    ...  
})
```

## 22. データ受信：処理を記述します。

データ取得方法)

以下のように「`data.val()`」で受信し、`v`変数に代入する。

---

```
var v = data.val(); //データ取得  
var k = data.key; //ユニーク KEY 取得
```

---



`v.username`

`v.text`

---

「`v.送信プロパティ名`」で取得します。

※ユニーク KEY は Firebase 側で自動で割り振られる KEY です。

送信データには全て KEY が割り振られます。

# データ受信：処理を記述します。

例) 受信完成コード

```
newPostRef.on("child_added", function (data) {
  let v = data.val(); //ここに保存されたデータが全て入ってくる
  let k = data.key; //今回は使いません

  let str = `<p>${v.username}<br>${v.text}</p>`;

  //ここでデータをhtmlに埋め込む
  $("#output").prepend(str);
})
```

# 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るか確認しましょう。

名前

テキスト

rrrr

ddd

ssss

sss

てすと

ててててて

てすと

テキスト保存チェック中

← → ⌛ ⌂ ⓘ ファイル | /Users/mitukun/Downloads

名前

テキスト

rrrr

ddd

ssss

sss

てすと

ててててて

てすと

テキスト保存チェック中

# 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

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

```
charCode: (...)  
clientX: (...)  
clientY: (...)  
code: (...)  
ctrlKey: (...)  
► currentTarget: textarea#text  
data: undefined  
► delegateTarget: textarea#text  
detail: (...)  
eventPhase: (...)  
► handleObj: {type: "keydown", origType: "ke...  
► isDefaultPrevented: f Se()  
jQuery34109597488976965831: true  
key: "Enter"  
keyCode: 13  
metaKey: false  
offsetX: undefined  
offsetY: undefined  
► originalEvent: KeyboardEvent {isTrusted: t  
pageX: (...)  
pageY: (...)
```

# console.logがポイント



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。

# Enter Keyで送信する方法

```
$("#text").on("keydown", function(e) {  
    // eの中の[keyCode]の数字が13の時に送信イベントを発動させる  
    if (e.keyCode === 13) {  
        newPostRef.push({  
            username: $("#username").val(),  
            text: $("#text").val()  
        })  
        $("#text").val("");  
        $("#username").val("");  
    }  
})
```

# ChatにIcon

初~中級向け

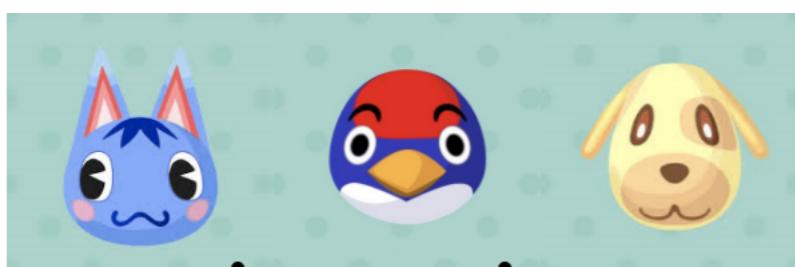
理解できていて次に進める人はどうぞ

# iconを使う 1 (HTML)

## 1. HTMLに追加で記述

```
<!-- 画像 -->  
<div>  
    
    
    
</div>
```

## 2. img/フォルダ内に以下画像があります



# iconを使う2 (JavaScript)

```
let d = 0;
const img = ["buke.png", "toku.jpg", "carama.png"]
$(".icon").on("click", function () {
  console.log(this);
  d = $(this).attr("data-img")
})]

$( "#send" ).on("click", function () {
  newPostRef.push({
    username: $("#username").val(),
    text: $("#text").val(),
    icon: d
  })
  $("#text").val("");
  $("#username").val("");
})
```

```
newPostRef.on("child_added", function (data) {
  let v = data.val() //ここに保存されたデータが全て入ってくる
  let k = data.key; //今回は使いません

  console.log(d);
  let str = `<p>${v.username}<br>${v.text}<p><img src='img/${img[v.icon]}'>`;
  //ここでデータをhtmlに埋め込む
  $("#output").prepend(str);
})
```

# Google Cloud Platform

## 『Google Cloud プラットフォーム』検索



Google Cloud プラットフォーム

すべて ニュース 画像 ショッピング 動画 もっと見る 設定 ツール

約 5,870,000 件 (0.43 秒)

**Google Cloud プラットフォーム**

<https://console.cloud.google.com/?hl=ja> ▾

Google Cloud Platform では、Google と同じインフラストラクチャでアプリケーション、ウェブサイト、サービスを構築、導入、拡大することができます。

他の人はこちらも検索

google developer console 料金 google cloud platform 料金  
google api 使い方 google api key  
google cloud platform 使い方 gcp 無料

# 参考URL

サインアウト参考：

<https://firebase.google.com/docs/auth/web/password-auth?hl=ja>

ルール参考：

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

認証サンプルファイル参考：

[https://webbibouroku.com/Blog/Article.firebaseio-auth?fbclid=IwAR0WV7Z5nFz02ljK\\_205jmPSCidZk8IEjojW1tvnEufOe63M0ljuPWTCDX4](https://webbibouroku.com/Blog/Article.firebaseio-auth?fbclid=IwAR0WV7Z5nFz02ljK_205jmPSCidZk8IEjojW1tvnEufOe63M0ljuPWTCDX4)