

# Javascript 03

FINAL



G's ACADEMY  
FUKUOKA



# 本日の内容

講義 + 作業 : 2.5 h程度  
演習 : 1.5 h程度

# 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

# 関数 function

# 関数とは??

function01.html

## ■関数

- ・ 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(定義)

```
function test(){  
    console.log('関数は便利！');  
}
```

## ■例(実行)

```
test(); //関数は便利！
```

呼び出さないと実行されない点に注意！  
(定義しただけでは実行されない)

## ■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．

# 引数と戻り値

function02.html

## ■例(定義)

```
function add(a, b){  
  const total = a + b;  
  return total;  
}
```

aとbを加算して合計の値を返す

## ■例(実行)

```
const sum = add(10, 20);  
console.log(sum);      //30
```

10と20を入力すると,  
30が返ってくる



# 【おまけ】

## ■関数の記述方法

```
function add(a, b){  
  var sum = a + b;  
  return sum;  
}
```

```
const add = (a, b) => {  
  var sum = a + b;  
  return sum;  
}
```

```
const add = function(a, b){  
  var sum = a + b;  
  return sum;  
}
```

**全部同じ！  
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！

# 関数の応用例

# 関数はいつ使うのか??

function03.html

## ■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よう！

# 関数はいつ使うのか??

function03.html

## ■関数の定義

```
function rand(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

# 関数はいつ使うのか??

function03.html

## ■ ボタン押したイベントで実行

```
$('#btn01').on('click', function () {  
    var result = rand(1, 10);  
    $('#echo').text(result);  
});
```

※btn02, btn03も同様

ボタンごとに範囲を指定して実行

【参考】 janken.htmlに関数を使用したじゃんけんの例もあります！

# 関数を使おう！

## ■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！

# 自作チャットの作成



Firebase



# firebase(realtime database)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。

Firebase Realtime Database Security Rulesによって不適切なユーザーからの書き込みや読み取りの防止、不正な値が入らないなどのバリデーションを実現しています。

引用：WEB+DB PRESS vol.105 第4章

# firebase(realtime database)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！

**早速準備を進めましょう！**

# 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/>



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.

① ログイン

② コンソール画面へ移動

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](#)

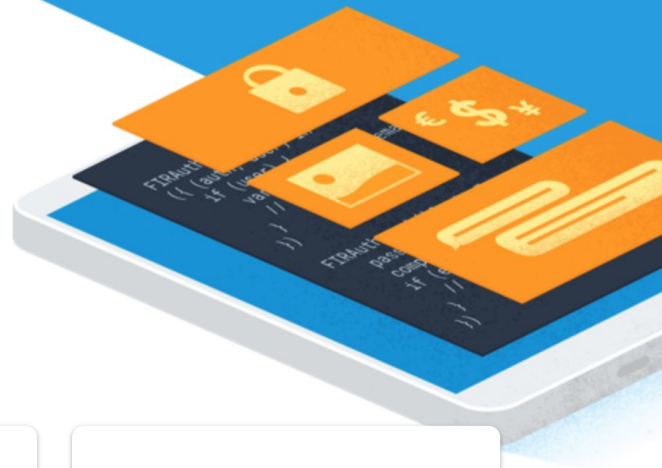
最近のプロジェクト



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

ドキュメントに移動

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 Google と共有します
- ✓ 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 Google と共有します
- ✓ 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 Google と共有します
- ✓ 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

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. Further down are categories: '開発' (Development) with icons for Authentication, Database, Storage, Hosting, Functions, and ML Kit; '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 text 'アプリに Firebase を追加して利用を开始しましょう' (Add Firebase to your app and start using it) is prominently displayed. Below this, a paragraph explains that most Firebase features and Core SDKs for iOS and Android are included. At the bottom, three circular icons are shown: 'iOS', 'Android', and a code icon '</>'. The code icon is highlighted with a red square, and a large purple arrow points from the text above towards it. An illustration of a woman and a man looking at a smartphone is also present.

chatapp ▼ ドキュメントに移動 🔔 ☕

chatapp Spark プラン

## アプリに Firebase を追加して利用を开始しましょう

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

iOS Android </>

开始するにはアプリを追加してください



# 必要なコードが表示されるのでコピー！！

## ウェブアプリに Firebase を追加



HTML の一番下、他のスクリプトタグの前に、以下のスニペットをコピーして貼り付けてください。

```
<script src="https://www.gstatic.com/firebasejs/5.3.1/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyAGB4H2MqR8TeoPb3L6IMR1_GFN_m0AB1A",
    authDomain: "chatapp-eb52d.firebaseio.com",
    databaseURL: "https://chatapp-eb52d.firebaseio.com",
    projectId: "chatapp-eb52d",
    storageBucket: "chatapp-eb52d.appspot.com",
    messagingSenderId: "138570310714"
  };
  firebase.initializeApp(config);
</script>
```

コピー

これらのリソースを参照し、ウェブアプリ用の Firebase の詳細について確認してください。

[Get Started with Firebase for Web Apps](#)

[Firebase Web SDK API Reference](#)

[Firebase Web Samples](#)

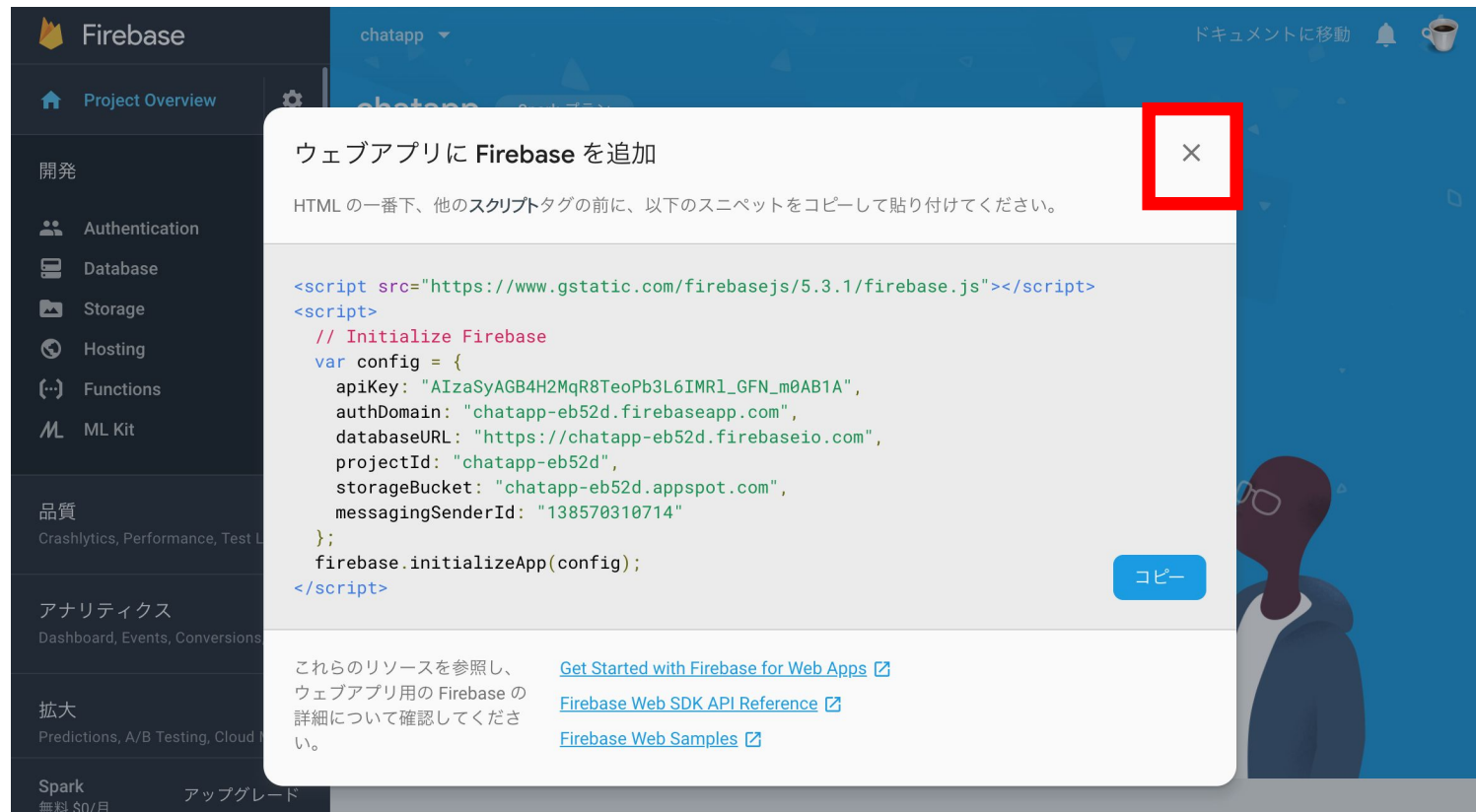
# エディタに貼り付け！

chatapp.html

```
14 ...<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
15
16
17 ...<script src="https://www.gstatic.com/firebasejs/5.7.1/firebase.js"></script>
18
19 <script>
20   // Initialize Firebase
21   var config = {
22     apiKey: "AIzaSyC9ihYAAJJcVD7",
23     authDomain: "chatapp-7fad5.firebaseio.com",
24     databaseURL: "https://chatapp-7fad5.firebaseio.com",
25     projectId: "chatapp-7fad5",
26     storageBucket: "",
27     messagingSenderId: "16451294",
28   };
29   firebase.initializeApp(config);
30 </script>
31 </body>
```



# ブラウザに戻ってダイアログを閉じる。



# ログインしなくてもチャットできるようにする設定

The screenshot shows the Firebase Authentication console for a project named 'chatapp'. The left sidebar contains the 'Authentication' menu item, which is highlighted with a red box. The main content area shows the 'Login Methods' tab, also highlighted with a red box. Below this, a table lists various login providers and their status. The 'Anonymous' provider is highlighted with a red box at the bottom of the table. A text box with arrows indicates the navigation path: 'Authentication' -> 'Login Methods' -> 'Anonymous'.

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
Facebook	無効
Twitter	無効
GitHub	無効
匿名	無効

「Authentication」 → 「ログイン方法」 → 「匿名」の順にクリック

# ログインしなくてもチャットできるようにする設定

Firebase

chatapp Authentication

ドキュメントに移動

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...

Play ゲーム	無効
Facebook	無効
Twitter	無効
GitHub	無効
匿名	有効にする <input checked="" type="checkbox"/>

「有効にする」 → 「保存」

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 ルールを強化します。 [詳細](#)

キャンセル 保存

承認済みドメイン

# データベースの準備(realtime databaseを選択)

Firebase

chatapp Database

ドキュメントに移動

Project Overview

開発

- Authentication
- Database**
- Storage
- Hosting
- Functions
- ML Kit

「Database」 → 「Realtime Database」 → 「データベースを作成」

Realtime Database

データベースを作成

# データベースの準備(realtime databaseを選択)

「テストモードで開始」 → 「有効にする」  
または Realtime Database を選択

### Realtime Database 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  
も、データベースの読み取りや書  
き込みを行います

キャンセル **有効にする**

# チャット画面の作成



# htmlの準備(入力欄と表示欄の作成)

chatapp.html

```
<div>
```

```
  <div>name<input type="text" id="name">
```

```
</div>
```

```
<div>
```

```
  <textarea name="" id="text" cols="30" rows="10"></textarea>
```

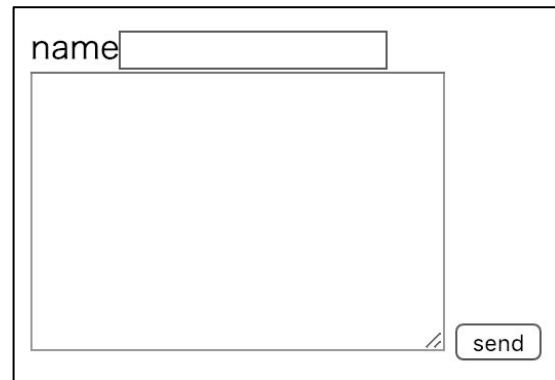
```
  <button id="send">send</button>
```

```
</div>
```

```
<div id="output"></div>
```

```
</div>
```

こんな感じ！



# データ送信処理の作成

# リアルタイム通信の準備！

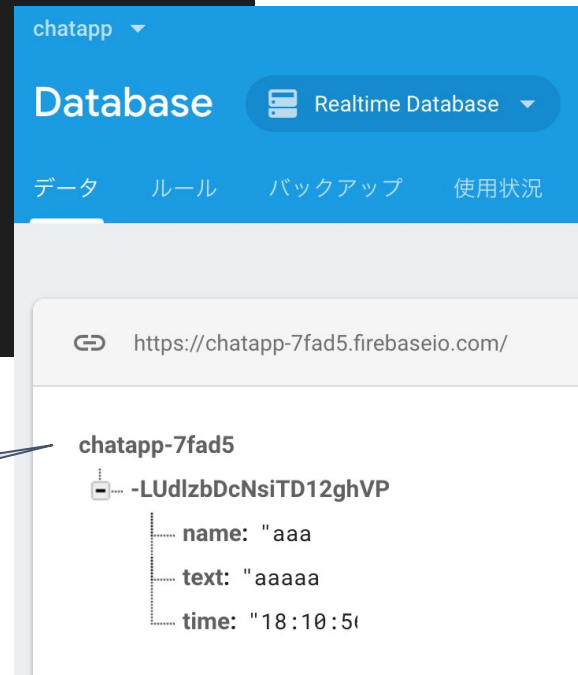
chatapp.html

```
projectId: "chatapp-eb52d",
storageBucket: "chatapp-eb52d.appspot.com",
messagingSenderId: "138570310714"
};
firebase.initializeApp(config);
```

// メッセージ送受信準備

```
var newPostRef = firebase.database().ref();
```

「newPostRef」がここに対応



# リアルタイム通信の準備！

chatapp.html

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() {
```

```
});
```

# データ送信の処理を記述！

chatapp.html

## ■送信処理の書き方

```
newPostRef.push({          //newPostRefが送信先
    name: $('#name').val(), //オブジェクトの形でデータを送る
    time: ymd(),
    text: $('#text').val()
});
```

## ■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 shows the Firebase Realtime Database console interface. At the top, the 'Database' header is followed by a 'Realtime Database' dropdown menu and a help icon. Below this is a navigation bar with tabs for 'データ' (Data), 'ルール' (Rules), 'バックアップ' (Backups), and '使用状況' (Usage). The 'データ' tab is selected, displaying a tree view of the database structure. The root node is 'chatapp-7fad5'. Under it, a node with the key '-LUdlzbDcNsiTD12ghVP' is highlighted with a red rectangle. This node contains a JSON object with the following fields: 'name' with value 'aaa', 'text' with value 'aaaaa', and 'time' with value '18:10:50'. A speech bubble points to this node with the text: '送信が成功する则表示される！ (ランダムな文字列がkey名で追加)' (Displayed upon successful transmission! (Random string added as key name)).

Database Realtime Database ?

データ ルール バックアップ 使用状況

https://chatapp-7fad5.firebaseio.com/ + - ⋮

chatapp-7fad5

-LUdlzbDcNsiTD12ghVP

- name: "aaa"
- text: "aaaaa"
- time: "18:10:50"

送信が成功する则表示される！  
(ランダムな文字列がkey名で追加)

# データ送信の処理を記述！

chatapp.html

## ■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



# データが追加されたら自動的に表示！

chatapp.html

## ■受信処理の書き方

```
newPostRef.on('child_added', function(data){    //データ追加時  
    var k = data.key;  
    var v = data.val();  
    //次ページへ続く...  
});
```

child\_addedでデータ追加時に実行される！

dataに全部のデータが入る！  
keyでキー名を取得！  
val()で保存したデータを取得！

# データが追加されたら自動的に表示！

chatapp.html

## ■ブラウザに表示

//...前ページの続き

```
var str = '';
```

```
str += '<div id="' + k + '">';
```

//idにkey名を追加

```
str += '<p>' + v.name + '</p>';
```

```
str += '<p>' + v.time + '</p>';
```

```
str += '<p>' + v.text + '</p>';
```

```
str += '</div>';
```

```
$('#output').prepend(str);
```

↓↓こんな感じで出力↓↓

```
<div id="key名">  
  <p>名前</p>  
  <p>時間</p>  
  <p>本文</p>  
</div>
```

# 各データのとり方のイメージ

Database



Realtime Database



データ

ルール

バックアップ

使用状況

<https://chatapp-7fad5.firebaseio.com/>



chatapp-7fad5

-LUdlzbDcNsiTD12ghVP

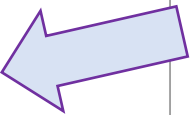
```
{
  name: "aaa
  text: "aaaaa
  time: "18:10:50"
```

data.key;

data.val();  
data.val().name;

# リアルタイムチャットの動作確認(ここまで作ろう！)

name



aaa

19:11:39

wwwww

aaa

18:10:56

aaaaa

①片方で送信して.

name

...

aaa

19:11:39

②両方で表示されればOK

aaa

18:10:56

aaaaa





# データ送信の処理を記述！

chatapp.html

## ■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で送信

# メッセージャー的な操作も作れる！

```
$( '#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: false  
    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. . .)

## 参考情報

### ■ここまで作ろう！

`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ます！

(コナミコマンドとか)

【参考】 <https://shgam.hatenadiary.jp/entry/2013/06/27/022956>



# 課題

# 【課題】チャットアプリ作成

## ■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！

**提出は次週木曜日「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>Chatアプリ</title>
9 </head>
10
11 <body>
12
13     <div>
14         <div>name<input type="text" id="name"></div>
15         <div>
16             <textarea name="" id="text" cols="30" rows="10"></textarea>
17             <button id="send">send</button>
18         </div>
19         <div id="output"></div>
20     </div>
```

```
21
22 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
23 <script src="https://www.gstatic.com/firebasejs/5.7.1/firebase.js"></script>
24 <script>
25     // Initialize Firebase
26     var config = {
27         apiKey: "AIzaSyC9ihYAAJJcVD7",
28         authDomain: "chatapp-7fad5.firebaseio.com",
29         databaseURL: "https://chatapp-7fad5.firebaseio.com",
30         projectId: "chatapp-7fad5",
31         storageBucket: "",
32         messagingSenderId: "16451294",
33     };
34     firebase.initializeApp(config);
35
36     // メッセージ送受信準備
37     var newPostRef = firebase.database().ref();
38
```





```
39  .....// 日時を取得する関数
40  .....function ymd(){
41  .....    var date = new Date();
42  .....    return date.getHours() + ":" + date.getMinutes() + ":" + date.getSeconds()
43  .....}
44
45  .....// 送信ボタンクリックでメッセージ送信
46  .....$('#send').on('click', function(){
47  .....    newPostRef.push({
48  .....      name: $('#name').val(),
49  .....      time: ymd(),
50  .....      text: $('#text').val()
51  .....    });
52  .....    $('#text').val('');
53  .....  });
54
```

```
55  .... //メッセージが追加されたら自動的に表示
56  .... newPostRef.on('child_added', function(data){
57  ....     var k = data.key; //ユニークkey取得
58  ....     var v = data.val(); //データ取得
59  ....     var str = '';
60  ....     str += '<div id="' + k + '>';
61  ....     str += '<p>' + v.name + '</p>';
62  ....     str += '<p>' + v.time + '</p>';
63  ....     str += '<p>' + v.text + '</p>';
64  ....     str += '</div>';
65  ....     $('#output').prepend(str);
66  .... });
67
68  </script>
69
70  </body>
71
72  </html>
```

**「写経」これでいける！！**  
**提出は毎週木曜日「23:59:59」まで！！**

# チュータリングタイム

17:00までは一人でもくもく  
後半は近くのメンバーで教え合おう！