



# PAAK勉強会 JS・掲示板編

高橋一平  
2014/12/05

# 自己紹介

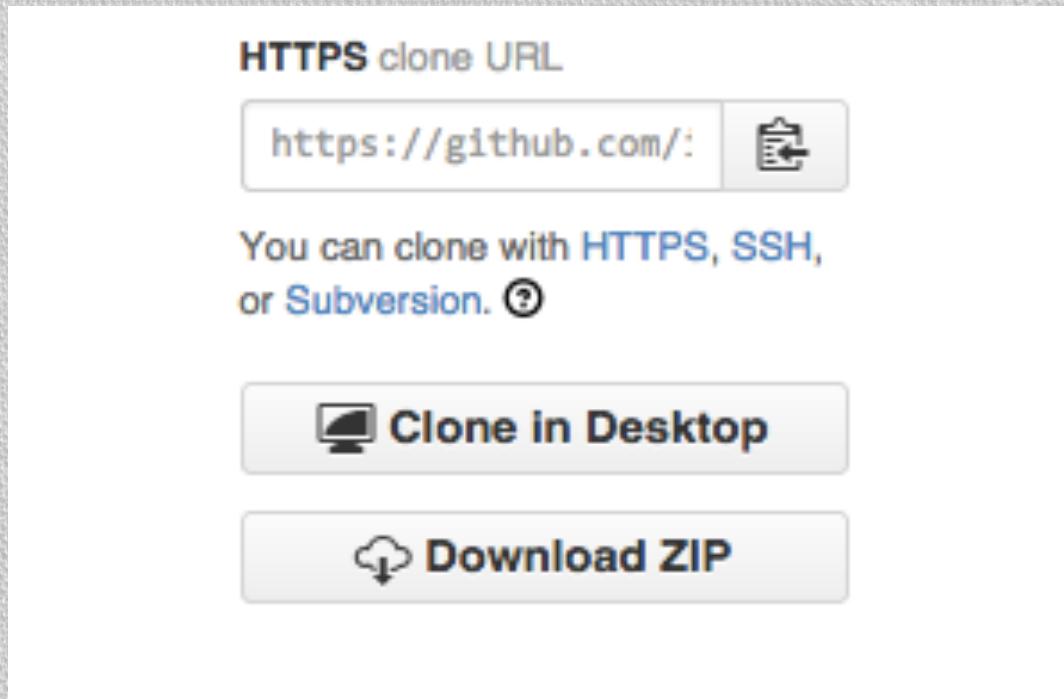
- ❖ 名前：高橋一平
- ❖ 年齢：24
- ❖ 所属：東京工業大学修士2年, リクルート内定者
- ❖ 趣味：ボルダリング, 音ゲー
- ❖ 好きなプログラミング言語：Haskell

# 今日の内容

- ❖ JavaScriptについて
  - ❖ 基本的な文法や、考え方など
- ❖ 掲示板の作成を通して、Node.js, MongoDBの基礎を学ぶ
- ❖ アンケート

# 必要なファイルをダウンロード

1. [https://github.com/ippei-takahashi/paak\\_2014\\_12\\_05](https://github.com/ippei-takahashi/paak_2014_12_05)にアクセス
2. 右下のほうにあるDownload ZIPをクリックすればダウンロードできる

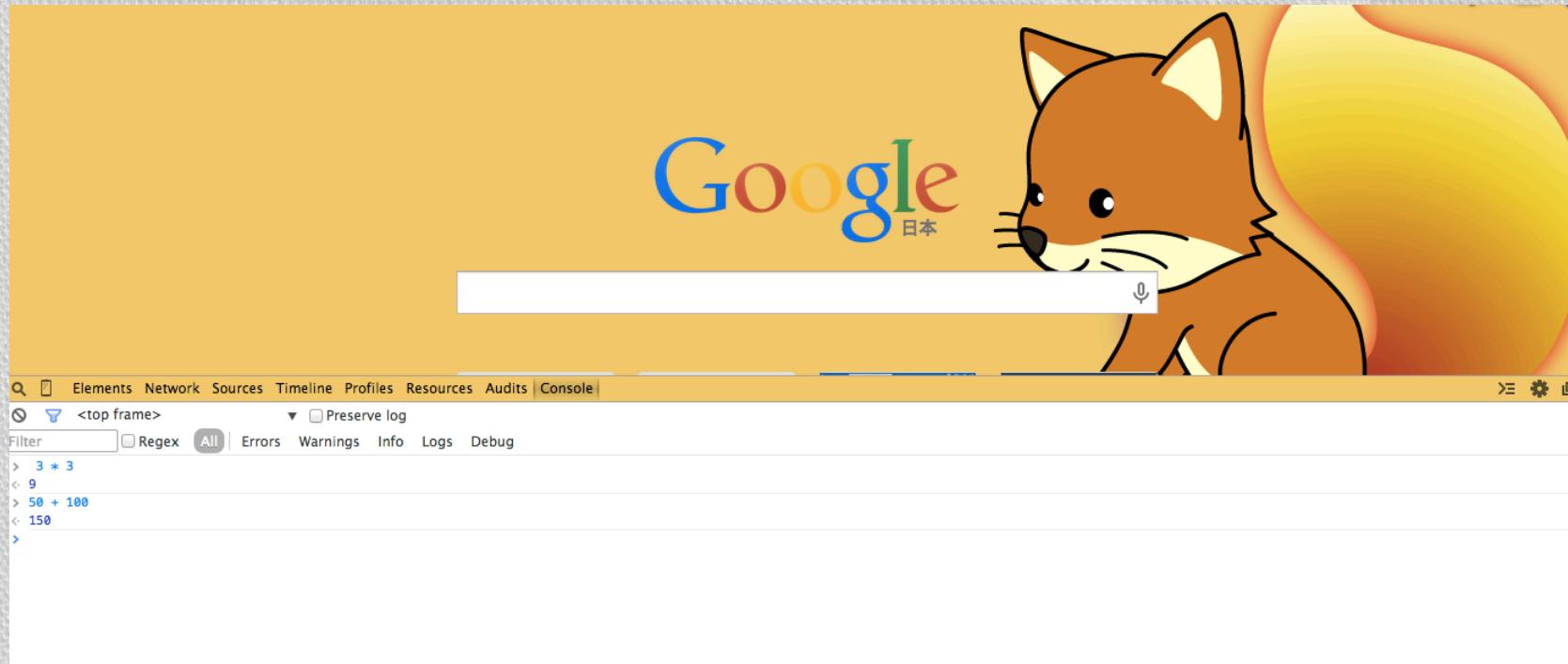


# JavaScriptの特徴

- ❖ 様々な環境で動く
  - ❖ ブラウザ, サーバサイド, iOS/Androidアプリetc...
- ❖ オブジェクト指向プログラミングをサポート
- ❖ 関数型プログラミングをサポート

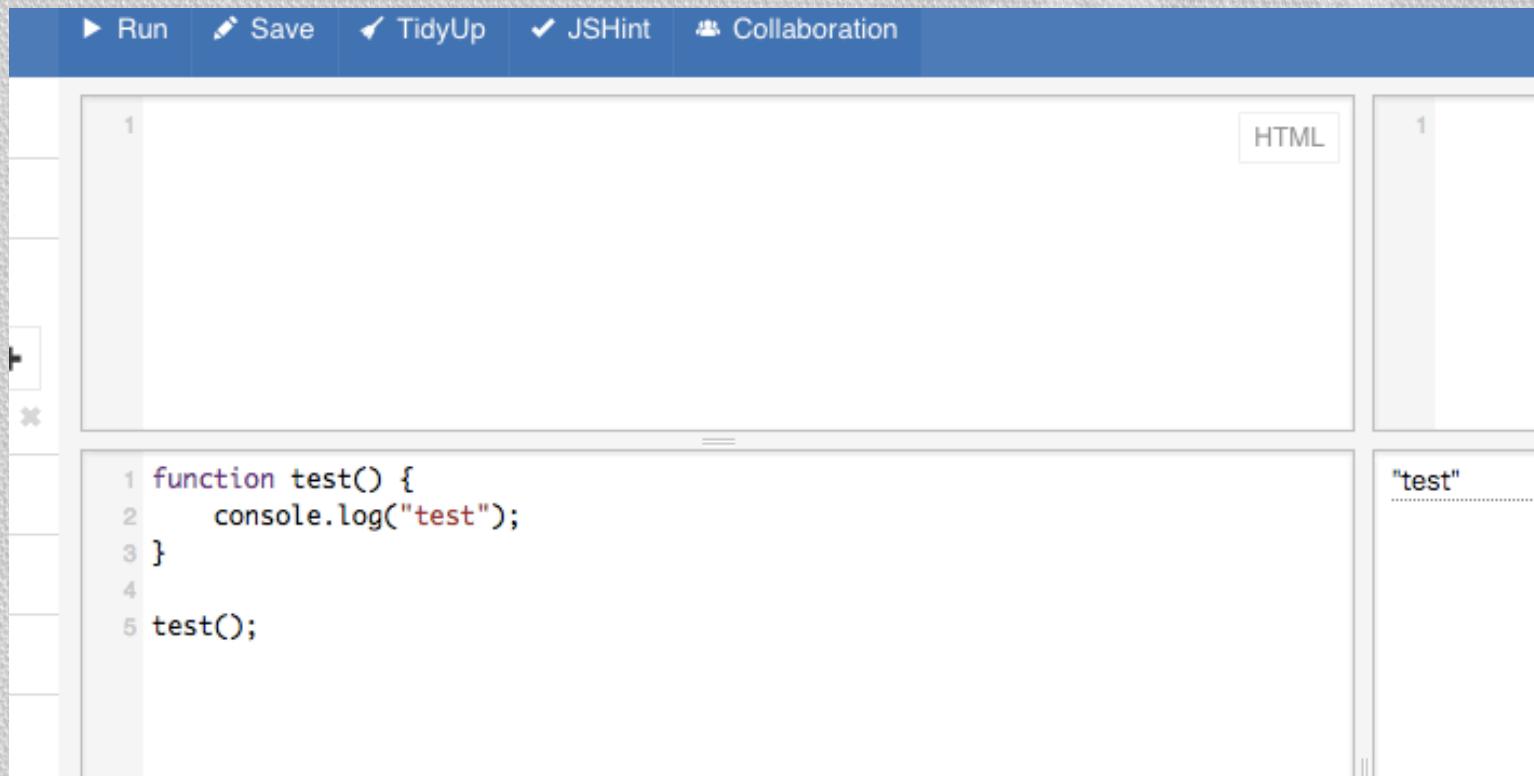
# ブラウザで動かす

- ❖ 例えばGoogle Chromeなら, Command + Option+ Jあるいは右クリックして要素の検証をクリックし, Consoleタブを選択することで, JavaScriptコンソールが開ける



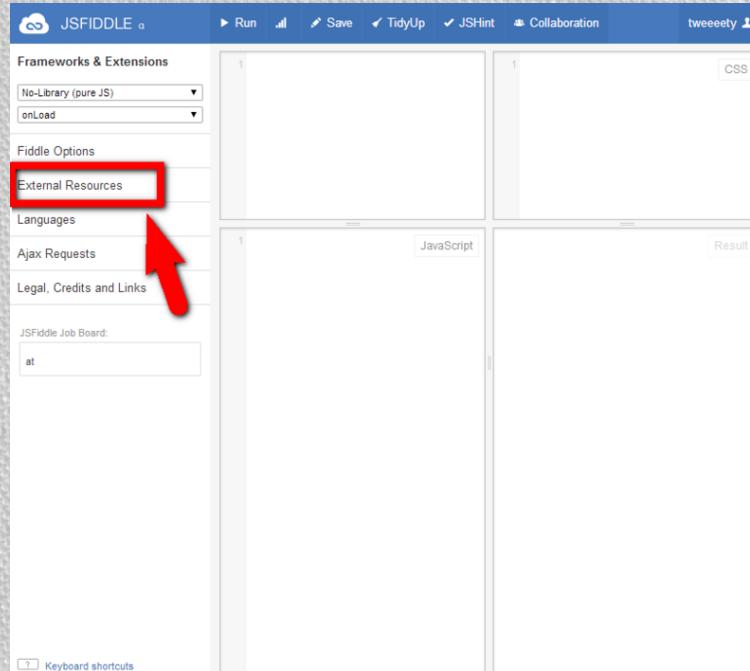
# もっと便利な方法

- ❖ JsFiddleを使う (<http://jsfiddle.net/>)
  - ❖ 左下のボックスにコードを書いて、左上のRunボタンで実行できる
  - ❖ htmlやCSSを書くことも可能



# 下準備

- ❖ 下記の位置にあるExternal Resourceをクリックし,  
<https://rawgithub.com/tweeeety/logFiddle/master/execLogFiddle.js>  
を追記して「+」ボタンをクリック



# JavaScriptの文法

❖ JSを動かす準備ができたので、早速文法を学んでみよう！

# 変数

```
1 var a = 3; // varをつけて変数を宣言
2
3 console.log(a);
4
5 var b = a;
6 a = "Hello"; // 数字だけでなく、文字列なども入れられる
7
8 console.log(a);
9 console.log(b); // bにaを代入した時点の、3となる
10
11 var c = 4, d = 5; // カンマ区切りでまとめて宣言もできる
12
13 console.log(c);
14 console.log(d);
```

```
3
"Hello"
3
4
5
```

# if文

```
1 if (true) {  
2     console.log("true");  
3 }  
// if文では、()の中身がtrue（正しい）なら、{}内を実行する  
5  
6 if (false) {  
7     console.log("false");  
8 }  
// ()の中身がfalse（間違い）のときは、{}内を実行しない  
10  
11 var a = 3;  
12 if (a === 3) {  
13     console.log("aは3と等しい");  
14 }  
// ===で、2つの値が等しいか比較する  
16  
17 if (a < 3) {  
18     console.log("if");  
19 } else {  
20     console.log("else");  
21 }  
// 条件が間違っているときに実行させるにはelse文を使う
```

"true"  
"aは3と等しい"  
"else"

# for文

```
1 var i, j;
2 for (i = 0; i < 10; i++) {
3     console.log(i);
4 }
5 // for (初期化; ループの継続条件; カウンタ変数の更新)で、命令を複数回繰り返すことができる
6 // i++は、iに1を足すという意味 (i = i + 1と同じ)
7
8 for (i = 0; i < 4; i++) {
9     for (j = 0; j < i; j++) {
10         console.log("i = " + i + ", j = " + j);
11     }
12 }
13 // forを二重に重ねることもできる
```

```
0
1
2
3
4
5
6
7
8
9
"i = 1, j = 0"
"i = 2, j = 0"
"i = 2, j = 1"
"i = 3, j = 0"
"i = 3, j = 1"
"i = 3, j = 2"
```

# 演習

- ❖ 結果画面に以下を出力するプログラムを書いてみよう
- ❖ ヒント :  $i--$ で $i$ から $i$ を引く,  $i \geq 0$ で $i$ が0以上という意味になる

```
ipt
10
9
8
7
6
5
4
3
2
1
0
```

## 解答例

```
1 var i;  
2 for (i = 10; i >= 0; i--) {  
3     console.log(i);  
4 }
```

# 演習（応用）

- ❖ 結果画面に以下を出力するプログラムを書いてみよう
- ❖ ヒント：for文を二重に使う、文字列の連結には+を使う

("a" + "bc" => "abc")



```
***  
***  
*** *  
*** **  
*** ***  
*** ****  
*** *****  
*** ***** *  
*** ***** **  
*** ***** ***  
*** ***** ** *
```

## 解答例

```
1 var i, j, str;
2 for (i = 0; i < 10; i++) {
3     str = "";
4     for (j = 0; j < i; j++) {
5         str += "*";
6     }
7     console.log(str);
8 }
```

# JavaScriptの関数

- ❖ 関数とは？
    - ❖ 引数を受け取って何らかの処理をして返り値を返すもの
      - ❖  $y = f(x)$  の  $f$
  - ❖ JavaScriptは関数型プログラミングをサポート
    - ❖ 関数が第一級オブジェクト
      - ❖ 関数は変数に代入できる
      - ❖ 関数は関数の引数に渡せる
      - ❖ 関数は関数の返り値にできる
- 関数は数値や文字列と同じように扱える！

# 関数

```
1 function plus(x, y) {  
2     return x + y; // returnで返り値（計算結果）を返す  
3 }  
4 // plusという名前の関数を宣言  
5 // x, yという2つの引数を受け取り、その和を返す  
6  
7 var result = plus(2, 3);  
8 console.log(result);  
9  
10 var x = plus;  
11 // 変数に代入することもできる  
12  
13 console.log(x(5, 4));  
14  
15 var minus = function(x, y) {  
16     return x - y;  
17 }  
18 // function() {}と、名前を付けずに関数を作ることもできる  
19  
20 console.log(minus(9, 2));  
21
```

```
5  
9  
7
```

# JavaScriptの関数を更に学ぶには

- ❖ 以下のことを調べてみよう
  - ❖ 高階関数
  - ❖ スコープ
  - ❖ クロージャ

# JavaScriptのオブジェクト

- ❖ オブジェクトとは
    - ❖ キーと値によってデータをまとめた構造
      - ❖ 例) 名前（キー）が太郎（値）で、年齢（キー）が20歳（値）のオブジェクト
    - ❖ JavaScriptの配列はオブジェクトの1つ
      - ❖ キーが数値のオブジェクト
  - ❖ JavaScriptはオブジェクト指向プログラミングをサポート
    - ❖ クロージャによるカプセル化
    - ❖ プロトタイプによる継承
- ひとまず、オブジェクトが便利だということを知っていればOK!

# オブジェクト

```
1 var taro = {  
2     name: "Taro",  
3     age: "20"  
4 };  
5 // キー:値という形式で、オブジェクトを定義  
6 // nameやageのことを、オブジェクトのプロパティと呼ぶ  
7  
8 console.log(taro.name);  
9 console.log(taro.age);  
10 // オブジェクト.キーで、プロパティにアクセス可能  
11  
12 taro.sex = "male";  
13 // あとからプロパティを追加することもできる  
14 console.log(taro.sex);  
15  
16 taro.sayHello = function() {  
17     return "Hello, I'm " + this.name;  
18     // プロパティ内でthisを使うと、オブジェクト自身を取得（この場合はtaro）  
19 };  
20 // もちろん関数もプロパティにできる  
21  
22 console.log(taro.sayHello());  
23 taro.name = "Jiro";  
24 console.log(taro.sayHello());  
25
```

```
"Taro"  
"20"  
"male"  
"Hello, I'm Taro"  
"Hello, I'm Jiro"
```

# 配列

```
1 var array = [1, 2, 3];
2 // フラグとカンマ区切りで配列を作れる
3
4 console.log(array[0]);
5 console.log(array[1]);
6 console.log(array[2]);
7 // arrayのプロパティ（配列のキーは0からはじまる）を取得
8
9 var obj = {
10   0: 1,
11   1: 2,
12   2: 3
13 }
14
15 console.log(obj[0]);
16 console.log(obj[1]);
17 console.log(obj[2]);
18 // objとarrayはほぼ等しい
19
20 console.log(array.length);
21 // 配列はlengthというプロパティを持っている
22 console.log(obj.length);
23 // objはlengthというプロパティを持っていない
```

```
1
2
3
1
2
3
3
undefined
```

# JavaScriptのオブジェクトを更に学ぶには

- ❖ 以下のことを調べてみよう
  - ❖ コンストラクタ関数
  - ❖ プロトタイプ
  - ❖ 繙承

# サーバサイドのJavaScript

- ❖ Node.jsの登場により、クライアントーサーバサイド両方をJavaScriptで記述することが一般的に
- ❖ Node.js: サーバサイドで動作するJavaScript環境
- ❖ JavaScriptさえ覚えればWebアプリケーションを一人で作れる！
- ❖ 同じ言語なので、もろもろの変換処理などが必要ない
- ❖ MongoDBを使うと、データは全て「JSON」として処理できる

# Node.jsの特徴

- ❖ 別個でApacheなどのWebサーバーを用意しなくてもよい
- ❖ パッケージ管理ツールなど、開発に必要なものは一通りそろっている
- ❖ たくさんのリクエストを処理するのが得意
  - ❖ チャットやメッセンジャーのようなサービスには適任
- ❖ 重い計算処理は苦手
- ❖ 基本的に、結構速い
  - ❖ <http://www.techempower.com/benchmarks/>

# さっそく、掲示板作りをスタート

1. ターミナルを開く
  - ❖ Launchpad -> その他に入っている
2. cd <zip解凍先フォルダ>/msgboard で掲示板アプリの土台があるフォルダに移動



# 動かしてみる

- ❖ npm install で必要モジュールをインストール
- ❖ node app.js で起動
- ❖ ブラウザで http://localhost:3000 にアクセスし, Hello World! と表示されれば成功

# ソースコードの解説

- ❖ 今回、**Express** というフレームワークを利用
- ❖ テキストエディタ(Sublime Text2がオススメ)で、routes.jsを開く（サーバ側で動くコード）

```
app.get("/", function(request, response) {  
    response.redirect("/index.html");  
});
```

この部分で、localhost:3000 にアクセスしたとき、  
public/index.html の内容を表示するよう設定している

“/” という部分がパスで、たとえばこれが “/user” なら、  
localhost:3000/user にアクセスしたときの設定になる

# 演習

1. localhost:3000 にアクセスしたときに、違う文字を表示する
2. localhost:3000/index にアクセスしたときも、index.htmlを表示する

# 掲示板画面を開く

- ❖ 以下のように、/msgboardにアクセスしたときに msgboard.htmlを開くようにする

```
app.get("/msgboard", function(request, response) {  
    response.redirect("/msgboard.html");  
});
```

- ❖ サーバ書き換え時にはアプリの再起動が必要
  - ❖ node app.js したターミナルでCtrl + C
  - ❖ node app.s でアプリを再起動)
  - ❖ ブラウザで http://localhost:3000 にアクセスし、掲示板っぽい画面が表示されれば成功

# html（抜粋）

```
<!-- フォームの中身をサーバに送信 -->
<form name="form" action="/sendmsg" method="POST">
  <span>名前 : </span>
  <div class="form-group">
    <!-- nameを付けた要素の値が送られる -->
    <input class="form-control" type="text" name="name" style="width: 200px;" />
  </div>
  <span>内容 : </span>
  <div class="form-group">
    <!-- nameを付けた要素の値が送られる -->
    <textarea class="form-control" name="body"></textarea>
  </div>
  <!-- formの中身を送信するボタン -->
  <div class="btn btn-primary" onclick="document.form.submit();">送信</div>
</form>
```

# サーバで受け取ったデータを処理

- ❖ routes.jsに以下を追記

```
app.post("/sendmsg", function(request, response) { // sendmsgにPOSTされたときの処理
  dbHandle.collection(
    "msgboard", // msgboardテーブルにデータを追加する
    function(outerError, collection) {
      var
        objMap = request.body,
        // ここに、 formの内容がオブジェクトとして格納される
        optionsMap = {};
      collection.insert(
        objMap,
        optionsMap,
        function(innerError, mapList) {
          response.redirect("/msgboard.html");
        });
    });
});
```

# データをDBに追加する

1. サーバを再起動する
2. localhost:3000/msgboardにアクセスし、下部にある入力フォームに名前と内容を適当に入力し、送信ボタンを押す
3. ターミナルで以下の操作を行う

mongo でmongoシェルを開く

```
> use msgboard
```

```
> db.msgboard.find()
```

- ❖ ここで、何らかの値が入っていれば成功
- ❖ このように、MongoDBでは事前にテーブル作成等をすることなく、データを追加したり取得することができる

# DBの中身を画面に表示する

- ❖ サーバからデータを取得するのに, **Ajax**を用いる
  - ❖ Ajax: 画面遷移をせずにデータをやり取りする仕組み
- ❖ サーバは**JSON**としてデータを返し, クライアント側で処理
  - ❖ JSON: JavaScriptのオブジェクトとほぼ同等
  - ❖ 例) {"name": "taro", "age": 20}

# サーバサイド

- ❖ routes.jsに以下を追記

```
app.get("/fetchmsg", function(request, response) { // sendmsgにGETされたときの処理
  dbHandle.collection(
    "msgboard", // msgboardテーブルからデータを取得
    function(outerError, collection) {
      var
        findMap= {},
        optionsMap = {};

      collection.find(
        findMap,
        optionsMap
      ).toArray(
        function(innerError, mapList) {
          response.send(mapList); // DBから取得したデータを全て返す
        });
    });
});
```

# JSONを見てみる

- ❖ アプリを再起動し、ブラウザからlocalhost:3000/fetchmsgにアクセスし、JSONが返ってくることを確認

# クライアントサイド

- ❖ public/js/main.jsに以下を記述

```
$(document).ready(function() {
  $.ajax({ // 非同期通信でデータを取得
    url: "/fetchmsg", // fetchmsgのURLからデータを取得
    success: function(json) {
      var i, data;
      for (i = 0; i < json.length; i++) {
        data = json[i];
        $("#messages")
          .append($("<li />").addClass("list-group-item"))
          .append($("<p />").text(data.name))
          .append($("<p />").text(data.body))
      };
    }
  });
})
```

# 結果を確認

- ❖ ブラウザからlocalhost:3000/msgboardにアクセスし、先ほど送信した内容が表示されるか確認
- ❖ 下部のフォームから新たなデータを追加すると、上部に追加されていくことを確認
- ❖ これで、掲示板が完成！

# Webアプリの別の作り方

- ❖ GrowUI
  - ❖ Webアプリをドラッグ&ドロップで作るためのツール
  - ❖ <http://growui.com>
  - ❖ Google Chrome推奨

# アンケート

- ❖ [https://docs.google.com/forms/d/1v\\_BdsLpcXb4qvmmVDmhR6U1Z5RZfkeTHSfIUJgR9aIQ/viewform](https://docs.google.com/forms/d/1v_BdsLpcXb4qvmmVDmhR6U1Z5RZfkeTHSfIUJgR9aIQ/viewform)

ありがとうございました！

# 時間が余ったら・・・

- ❖ 揭示板に投稿日時を載せるようにする
  - ❖ ヒント：POST時にサーバ側で付ける等で可能
- ❖ スレッドを作れるようにする
  - ❖ ヒント：スレッドのための新たなテーブルを作る