 Monaca で学ぶ

チャットアプリ開発



チャットアプリの仕様と開発計画(仮題)

アプリの種類と難易度

- 基本版
 - └ Lv1:一言チャット
 - └ Lv2:一言チャット (order&limit)
- カスタム版 (easy)
 - └ Lv3:名前付きチャット
 - └ Lv4:吹き出しチャット
- もっとカスタム版 (hard)
 - └ Lv5:アイコンチャット
 - └ Lv6:真アイコンチャット
- 最終カスタム版 (hard)
 - └ Lv7:起動画面付きチャット

基本版の機能

- メッセージを投稿できる
- メッセージを一覧表示できる

チャットアプリ

新しいスマホが欲しい

ご予算は？

3万円前後でお願いします

機能か性能のどちらかを妥協する
必要がありますね

ゲームとかしないから性能は妥協する、防水機能は欲しい

ひとこと

送信

カスタム版の機能

- 名前とメッセージを投稿できる
- 投稿者の名前と投稿日時が表示される
- 自分の投稿は吹き出しの向きなどが変わる

チャットアプリ

あんこ:新しいスマホが欲しい
(11/2 10:36:28)

なる:ご予算は?(11/2
10:36:40)

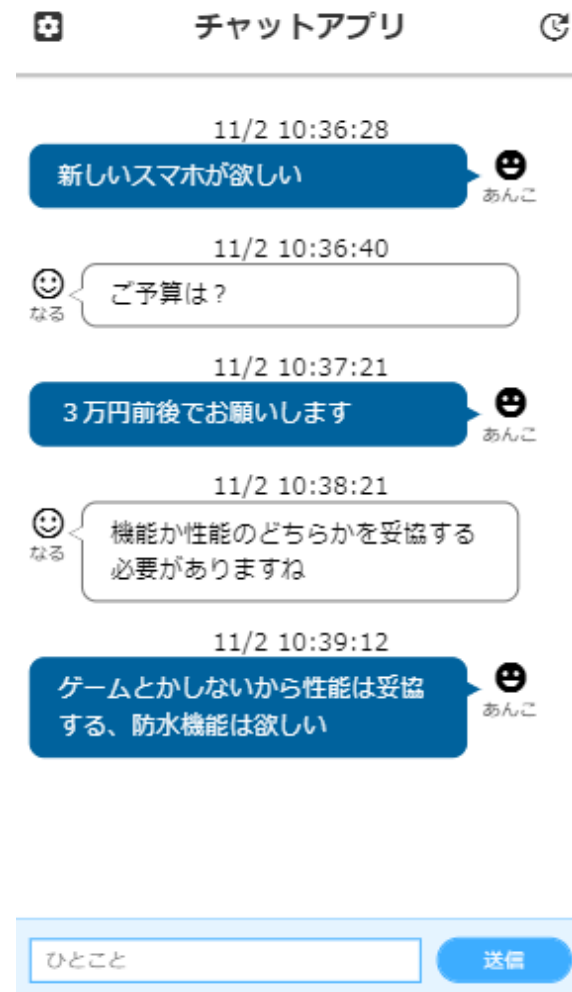
あんこ:3万円前後でお願いします
(11/2 10:37:21)

なる:機能が性能のどちらかを妥協する必要がありますね(11/2
10:38:21)

あんこ:ゲームとかしないから性能は妥協する、防水機能は欲しい(11/2 10:39:12)

もっとカスタム版の機能

- 最新の投稿一覧を取得できる
(リロード)
- 投稿時にアイコンも
登録できる
- 投稿一覧でアイコンが
表示される
- 名前と投稿日時が
メッセージの外に表示される



最終カスタム版

- 開始画面が新設される
- アイコンを一覧から選べる
- チャットルームが選べる

チャットアプリ - HOME

チャットアプリです。マナーや法令を守って利用しましょう。



開始

チャットアプリのデータ設計

■ chat

- └ name : 投稿者
- └ message : 投稿内容
- └ icon : アイコンの名前
- └ createDate : 投稿日時 ※DBが自動的に記録します

実データの例



The screenshot shows a web interface for a chat application. At the top, there is a header bar with a back arrow, the text 'chat', and buttons for '+新しいレコード', '+新しいフィールド', and '削除'. Below the header, there is a search filter '絞り込み' and a pagination control showing '62件の内 20 件を表示' with page numbers 1, 2, 3, 4 and navigation arrows. The main content is a table with 6 columns: a checkbox, 'objectId', 'icon', 'message', 'name', and 'createDate'.

<input type="checkbox"/>	objectId	icon	message	name	createDate
<input type="checkbox"/>	X6UWjaKSkj9dgpEb	emoji_emotions	ゲームとかしないから性能は妥協する、防水機能…	あんこ	2019-11-05T10:39:12.612+09:00
<input type="checkbox"/>	Ppq232SpxusQQBxP	sentiment_satisfied_alt	機能が性能のどちらかを妥協する必要がありますね	なる	2019-11-05T10:38:21.208+09:00
<input type="checkbox"/>	Dx5P6FuB98g3QSOS	emoji_emotions	3万円前後でお願いします	あんこ	2019-11-05T10:37:21.220+09:00
<input type="checkbox"/>	TKEmPh6ad6OkPFKT	sentiment_satisfied_alt	ご予算は？	なる	2019-11-05T10:36:40.493+09:00
<input type="checkbox"/>	HFamSta6CY93H7Hj	emoji_emotions	新しいスマホが欲しい	あんこ	2019-11-05T10:36:28.418+09:00

主要な独自関数

■ read()

└ DBの投稿内容を取得して一覧表示する。

Lvが上がるにつれて表示が複雑になるため、read()の処理もどんどん増えていく。

■ send()

└ DBにメッセージを登録する。

Lvが上がると名前やアイコンも登録するが、処理はあまり増えない。

■ init()

└ アプリの初期化処理を行う。

基本版ではread()関数を実行するだけだが、

カスタム版では名前などの初期設定に活用されている。

■ start()

└ 最終カスタム版のみ使用。

スタート画面のフォームに記述された設定を読み込み、アプリをスタートさせる。

主要な標準関数

■ new Date()

- └ 日付オブジェクトを作成する。
- └ 引数無しの場合は現在時刻のオブジェクトになる。
- └ 引数で日付を指定した場合、指定した日付のオブジェクトになる。

■ Dateオブジェクト

- └ .getMonth() メソッド：月を返す。実際には+1する必要がある。
- └ .getDay()メソッド：日を返す。
- └ .getHours()メソッド：時を返す。
- └ .getMinutes()メソッド：分を返す。
- └ .getSeconds()メソッド：秒を返す。

主要な標準関数

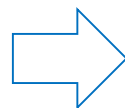
- `document.getElementById()`
 - └ DOMオブジェクトをID名で探して取得する。
 - └ 投稿一覧を書きかえたり、メッセージフォームの値を取得したりする時に使います。
- `document.createElement()`
 - └ DOMオブジェクトを新規作成する。
 - └ 投稿一覧に追記するDOMを作成するときに使います。
 - └ LV5以上は投稿内容が複雑になるため、この関数を多用します。

DOMオブジェクトのプロパティ確認

- `.value`プロパティ
 - └ フォームの値を取得する。
- `.innerHTML`プロパティ
 - └ 要素の内容をHTMLレベルで取得ないし変更する。
- `.textContent`プロパティ
 - └ 要素の内容をTEXTレベルで取得ないし変更する。
- `.appendChild()`メソッド
 - └ DOM要素を子要素として末尾に追加する。

リストタグとチャット表示の対応（基本版）

```
<ul id="chat">
  <li class="box">新しいスマホが欲しい
</li>
  <li class="box">ご予算は？</li>
  <li class="box">3万円前後でお願いします</li>
  <li class="box">機能か性能のどちらかを妥協する必要がありますね</li>
  <li class="box">ゲームとかしらないから性能は妥協する、防水機能は欲しい</li>
</ul>
```



チャットアプリ

新しいスマホが欲しい

ご予算は？

3万円前後でお願いします

機能か性能のどちらかを妥協する必要がありますね

ゲームとかしらないから性能は妥協する、防水機能は欲しい



チャットアプリ開発



チャットアプリ開発

Lv1

プロジェクトのインポート



アプリ・インポート

チャット基本版
20191104_LV0

このアプリをプロジェクトとしてインポート
します。クラウドIDEやMonacaデバッガーを
利用してデバイス上で起動できます。

インポート

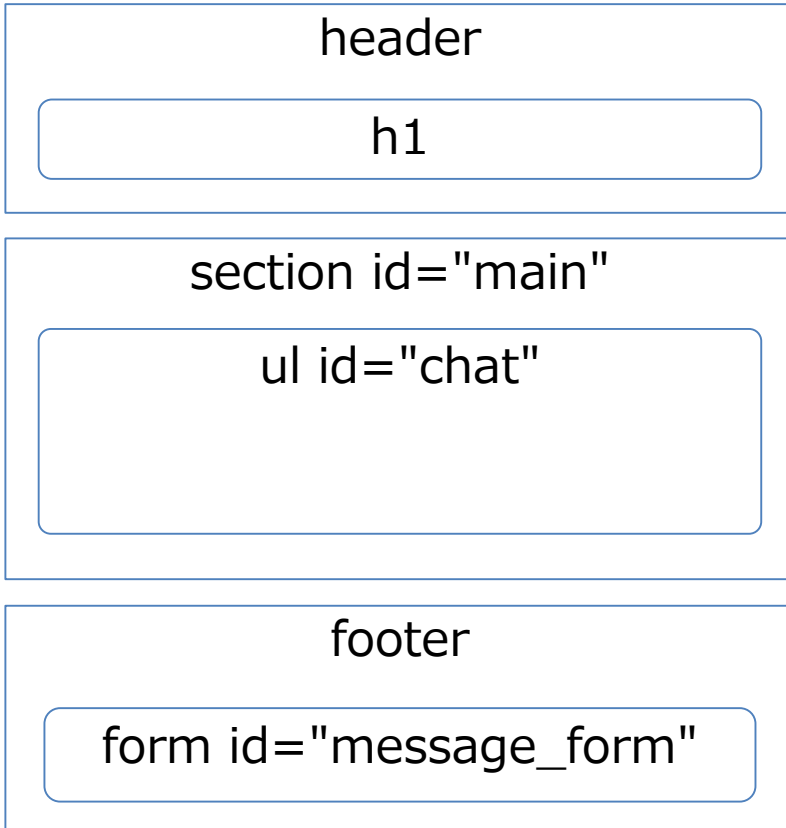
[ダッシュボードへ](#)

初期状態 (HTML)

Lv1ではHTMLの編集はありません

```
<body onload="init()">
  <header>
    <h1>チャットアプリ</h1>
  </header>
  <section id="main">
    <ul id="chat">
    </ul>
  </section>
  <footer>
    <form id="message_form">
      <input type="text" id="message" value="" placeholder="ひとこと">
      <input type="button" value="送信" id="sendbtn" onclick="send()">
    </form>
  </footer>
</body>
```

初期状態 (HTML)



初期状態 (JavaScript)

Lv1で実装すること

- 値の設定
 - └ DB接続情報
- 関数
 - └ read()
 - └ send()

```
// アプリの設定
var room = "chat";

// ncmbのデータベース(DB)に接続
var APIKEY = "*****";
var CLIENTKEY = "*****";
var ncmb = new NCMB(APIKEY, CLIENTKEY);

// チャットテーブルのオブジェクトを作成
var chatTable = ncmb.DataStore(room);

// アプリの初期化处理
function init(){
    read();
}

// DBからチャットデータを取得して描画する
function read() {
}

// 新規投稿をDBに登録する
function send() {
}
```

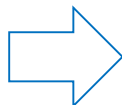
ncmbの接続情報取得 (アプリ新規作成の場合)

アプリの新規作成

アプリ名

半角英数字もしくはアンダースコア

[戻る](#) [新規作成](#)



✓ 新しいアプリが作成されました

chatアプリが作成されました。

SDKを利用して、mobile backendと連携したアプリを開発してみましょう！
詳しくはクイックスタートをご覧ください ([iOS](#) / [Android](#) / [Javascript](#) / [Unity](#))

APIキー

アプリケーションキー ? [コピー](#)

クライアントキー ? [コピー](#)

APIキーは[アプリ設定](#)からいつでも参照できます。

[OK](#)

ncmbの接続情報取得（既にアプリがある場合）

The screenshot shows the niftycloud mobile backend console interface. The browser address bar displays `console.mbaas.nifcloud.com/#/account/applications`. The navigation bar includes links for 'アプリ一覧', 'ダッシュボード', 'ドキュメント', '開発TIPS', 'コミュニティ', and '連携サービス'. A green button labeled '+新しいアプリ' and a blue button labeled 'アプリ一覧' are visible. The main content area shows the details for an application named 'chat', dated November 2019. It features four performance metrics: APIリクエスト (0回, 残り998,885回), プッシュ通知 (0回, 残り1,000,000回), ストレージ (0MB, 空き5,119.07MB), and スクリプト (0回, 0.0秒). At the bottom, the application key and client key are displayed with copy buttons.

APIリクエスト	プッシュ通知	ストレージ	スクリプト
0回	0回	0MB	0回
残り998,885回	残り1,000,000回	空き5,119.07MB	0.0秒

アプリケーションキー ? `676db3b7afc685cb16f63a3f166ed5bde0630f141bb13f9c5` コピー

クライアントキー ? `32ebb428b80a41b55ddab7082ced935c00bd882c55579b:` コピー

DB接続情報設定 (JavaScript)

- キーの情報をncmbからコピーして入力してください

```
// アプリの設定
var room = "chat";

// ncmbのデータベース(DB)に接続
var APIKEY = "*****";
var CLIENTKEY = "*****";
```



✓ 新しいアプリが作成されました

chatアプリが作成されました。

SDKを利用して、mobile backendと連携したアプリを開発してみましょう！
詳しくはクイックスタートをご覧ください ([iOS](#) / [Android](#) / [Javascript](#) / [Unity](#))

APIキー

アプリケーションキー  676db3b7afc685cb16f63a3f166ed5bde0630f141bb13f9c5 

クライアントキー  32ebb428b80a41b555ddab7082ced935c00bd882c55579b 

APIキーは[アプリ設定](#)からいつでも参照できます。

OK

read()関数の実装 (JavaScript)

```
function read() {  
  // 画面の初期化  
  document.getElementById("chat").innerHTML = null;  
  // チャットデータの取得と反映  
  chatTable.fetchAll().then(function(records){  
    for (var i = 0; i < records.length; i++) {  
      // 投稿を表示するためのリストタグを生成  
      var dom = document.createElement("li");  
      // リストタグにClass属性を付与  
      dom.classList.add('box');  
      // 投稿内容を作成  
      dom.textContent = records[i].get("message");  
      // 投稿内容をチャットエリアに追記  
      document.getElementById("chat").appendChild(dom);  
    }  
  })  
  .catch(function(error){  
    // もしfetchAll()が失敗したらエラーをログに書き出す  
    console.log(error);  
  });  
}
```

チャット初期化

DB取得して
件数分繰り返し処理

データを元にliのDOM生成

チャットに追記

send()関数の実装 (JavaScript)

```
function send() {  
  // フォームの値を取得  
  var message = document.getElementById("message").value;  
  // チャットレコードの作成と反映 (変数に入れることを省いている)  
  new chatTable()  
    .set("message",message)  
    .save()  
    .then(function() {  
      read();  
    })  
    .catch(function(error){  
      // もしsave()が失敗したらエラーをログに書き出す  
      console.log(error);  
    });  
}
```

フォームから
メッセージを取得

DBにメッセージを保存

read()関数呼んで
チャット表示を更新

チャットアプリ開発

Lv2

Lv2で実装すること

- 取得件数の制限
 - └ limitを掛ける
- 取得順の変更
 - └ 新着順
- 表示順の変更
 - └ 古い順

チャットアプリ

新しいスマホが欲しい

ご予算は？

3万円前後でお願いします

機能か性能のどちらかを妥協する
必要がありますね

ゲームとかしないから性能は妥協する、防水機能は欲しい

ひとこと

送信

limitの宣言

```
// アプリの設定  
var limit = 5;  
var room = "chat";
```

limitを宣言、今回は5件

取得プログラムの変更

read()関数内

```
// チャットデータの取得と反映
chatTable
.order("createDate",true)
.limit(limit)
.fetchAll()
.then(function(records){
  records.reverse();
  for (var i = 0; i < records.length; i++) {
```

DB取得時に作成日でorderを掛ける

DB取得時にlimitを設定

取得結果を逆順ソート



カスタム版(easy)



チャットアプリ開発

Lv3

Lv3で実装すること

- 名前の投稿
 - └ フォームを増やす
 - └ send()関数を改造
- 名前の表示
 - └ read()関数を改造
- 日付の表示
 - └ read()関数を改造

チャットアプリ

あんこ:新しいスマホが欲しい
(11/2 10:36:28)

なる:ご予算は?(11/2
10:36:40)

あんこ: 3万円前後でお願いします(11/2 10:37:21)

なる:機能が性能のどちらかを妥協する必要がありますね(11/2 10:38:21)

あんこ:ゲームとかしないから性能は妥協する、防水機能は欲しい(11/2 10:39:12)

HTMLの変更


```
<form id="message_form">
  <input type="text" id="myName" value="" placeholder="なまえ">
  <input type="text" id="message" value="" placeholder="ひとこと">
  <input type="button" value="送信" id="sendbtn" onclick="send()">
</form>
```

名前の欄を増やす

send()プログラムの変更

```
// フォームの値を取得  
var message = document.getElementById("message").value;  
var myName = document.getElementById("myName").value;
```

名前の取得を増やす



send()プログラムの変更

```
function send() {
  // フォームの値を取得
  var message = document.getElementById("message").value;
  document.getElementById("message").value = "";
  // チャットレコードの作成と反映
  new chatTable()
    .set("message",message)
    .set("name",myName)
    .save()
    .then(function() {
      read();
    })
    .catch(function(error){
      // もしsave()が失敗したらエラーをログに書き出す
      console.log(error);
    });
}
```

名前の取得を増やす

read()プログラムの変更

```
// 投稿日の日付オブジェクトを生成
var date = new Date(records[i].get("createDate"));

var time = date.getMonth() + 1 + "/"
           + date.getDay() + " "
           + date.getHours() + ":"
           + date.getMinutes() + ":"
           + date.getSeconds();
```

DBの値をJSの日付オブジェクトにする

日付オブジェクトから
表示したい文字列を作る

read()プログラムの変更

```
// 投稿内容を作成
dom.textContent = records[i].get("name")
                  + ":" + records[i].get("message")
                  + "(" + time + ")";
```

名前とメッセージと日付を
DOMに反映

チャットアプリ開発

Lv4

Lv4で実装すること

- 名前を先に設定
 - └ アプリ起動時に名前を聞く
- 条件分岐
 - └ 自分の投稿と他人の投稿でデザインを分ける

チャットアプリ

あんこ:新しいスマホが欲しい
(11/2 10:36:28)

なる:ご予算は?(11/2
10:36:40)

あんこ:3万円前後でお願いします(11/2 10:37:21)

なる:機能か性能のどちらかを妥協する必要がありますね(11/2 10:38:21)

あんこ:ゲームとかしないから性能は妥協する、防水機能は欲しい(11/2 10:39:12)

HTMLの変更

```
<form id="message_form">
  <input type="text" id="message" value="" placeholder="ひとこと">
  <input type="button" value="送信" id="sendbtn" onclick="send()">
</form>
```


名前の欄を消す



send()プログラムの変更

```
// フォームの値を取得  
var message = document.getElementById("message").value;
```

名前の取得を消す



アプリの初期化処理の変更

```
// アプリの初期化処理
```


```
function init(){
```

```
  myName = prompt("なまえ");
```

```
  read();
```

```
}
```

prompt命令で名前をユーザーに入力させる



アプリの初期化処理の変更

```
// 投稿を表示するためのリストタグを生成
var dom = document.createElement("li");
// リストタグにClass属性を付与
dom.classList.add('box');
if (records[i].get("name") == myName) {
    dom.classList.add('box-right');
} else {
    dom.classList.add('box-left');
}
```

表示する投稿の名前が自分の名前なら
class「box-right」を付与

表示する投稿の名前が自分でなければ
Class「box-left」を付与



もっとカスタム版(hard)



チャットアプリ開発

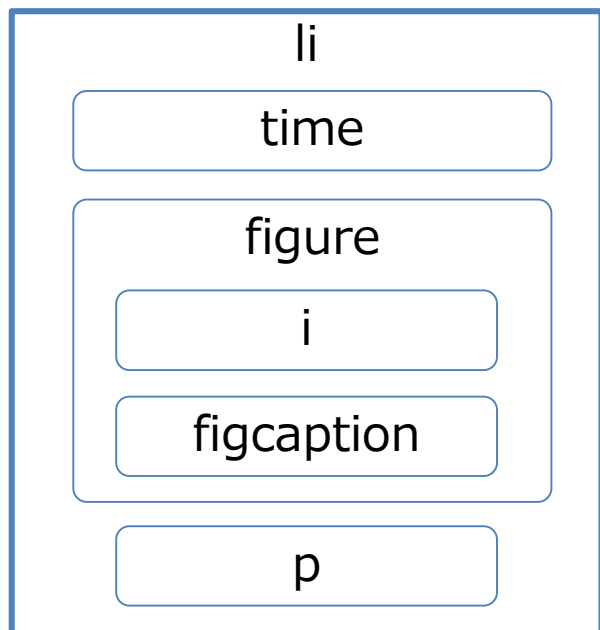
Lv5

Lv5で実装すること

- 投稿内容の分割
 - └ 複数のタグで構成
- デザイン変更
 - └ アイコンが付く
 - └ 吹き出しになる
 - └ 名前はアイコンの下
 - └ 日時は吹き出しの上

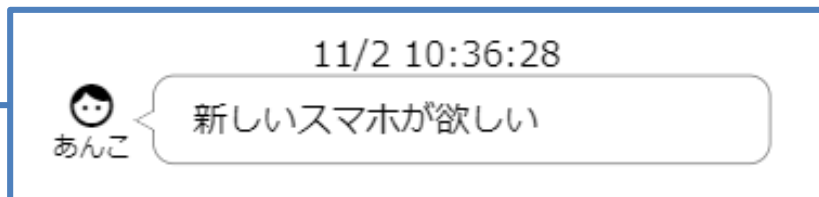


投稿内容を複数タグで構成する



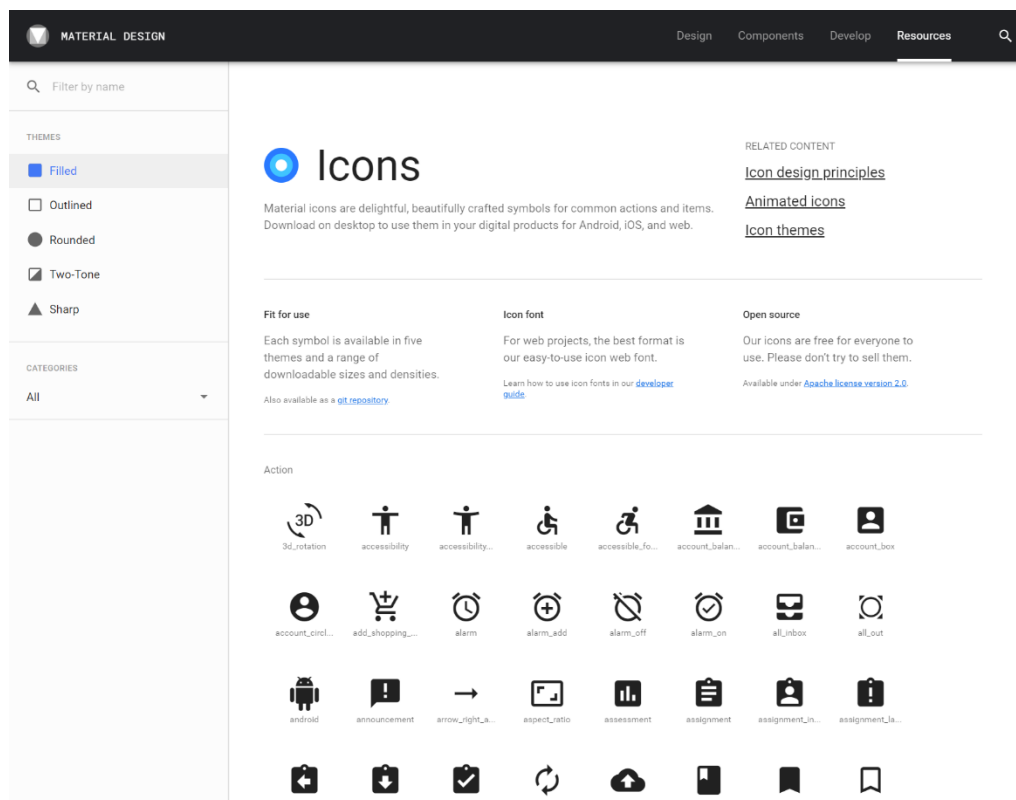
```
<li>
  <time>11/2 10:36:28</time>
  <figure>
    <i class="material-icons">face</i>
    <figcaption>あんこ</figcaption>
  </figure>
  <p class="box box-left">新しいスマホが欲しい</p>
</li>
```

チャットアプリ



アイコンの表示について

- 「Webアイコンフォント」を利用します
 - 具体的にはGoogleの「Material Icons」



Material Iconsの使い方

class名に「material-icons」と記述

```
<i class="material-  
icons">face</i>
```

タグは何でも良い

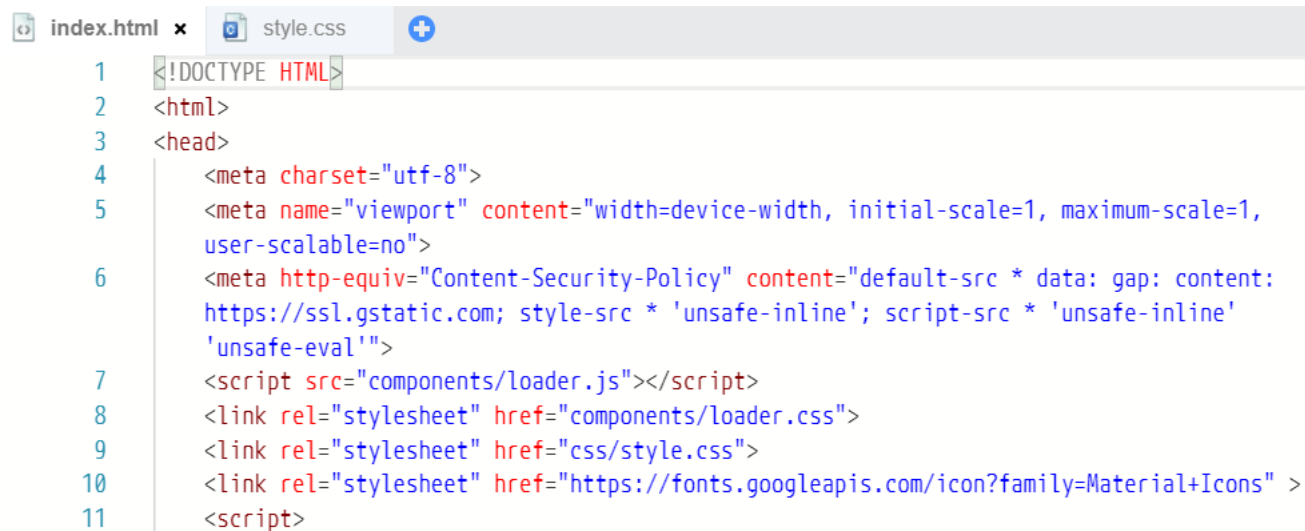
要素の値にアイコン名を記述

Material Iconsの導入

■ HTML側のheadに一行追加すればOK

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons"
```

■ 設定イメージ



```
index.html x style.css +
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1,
6     user-scalable=no">
7   <meta http-equiv="Content-Security-Policy" content="default-src * data: gap: content:
8     https://ssl.gstatic.com; style-src * 'unsafe-inline'; script-src * 'unsafe-inline'
9     'unsafe-eval'">
10  <script src="components/loader.js"></script>
11  <link rel="stylesheet" href="components/loader.css">
12  <link rel="stylesheet" href="css/style.css">
13  <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons" >
14  <script>
```

read()関数のfor文ループ内大幅変更 (1/2)

```
for (var i = 0; i < records.length; i++) {
  // 投稿日の日付オブジェクトを生成
  var date = new Date(records[i].get("createDate"));

  var figure = document.createElement("figure");
  var icon = document.createElement("i");
  icon.textContent = "face";
  icon.classList.add('material-icons');
  var figcaption =
document.createElement("figcaption");
  figcaption.textContent = records[i].get("name");
  figure.appendChild(icon);
  figure.appendChild(figcaption);

  var time = document.createElement("time");
  time.textContent = date.getMonth() + 1 + "/"
                    + date.getDay() + " "
                    + date.getHours() + ":"
                    + date.getMinutes() + ":"
```

read()関数のfor文ループ内大幅変更 (2/2)

```
// 投稿を表示するためのリストタグを生成
var li = document.createElement("li");
// 投稿内容を作成
var p = document.createElement("p");
p.classList.add('box');
p.textContent = records[i].get("message");

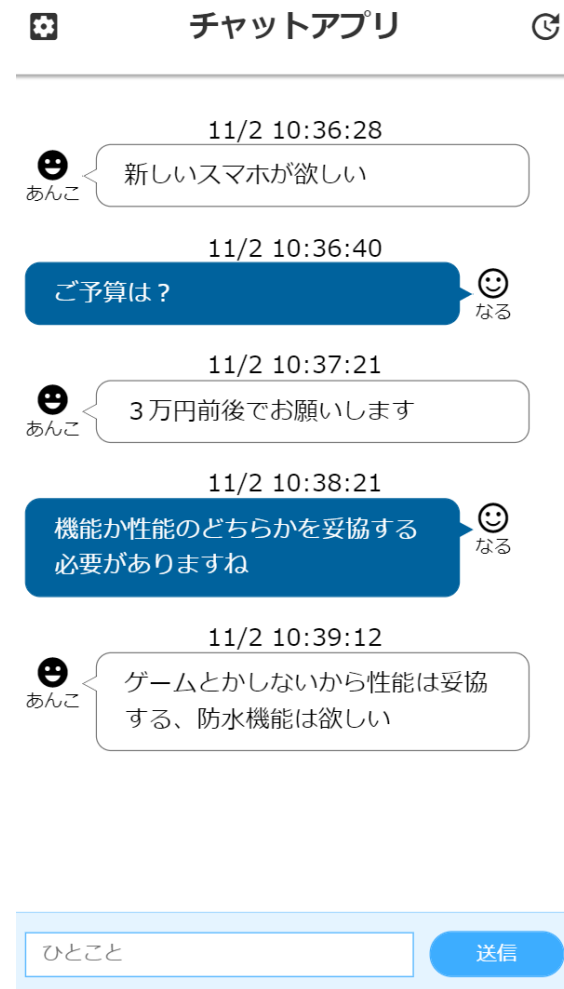
// 投稿内容をチャットエリアに追記
li.appendChild(time);
if (records[i].get("name") == myName) {
    p.classList.add('box-right');
    li.appendChild(p);
    li.appendChild(figure);
} else {
    p.classList.add('box-left');
    li.appendChild(figure);
    li.appendChild(p);
}
document.getElementById("chat").appendChild(li);
}
```

チャットアプリ開発

Lv6

Lv6で実装すること


- アイコンが選べる
 - └ 起動時にアイコン名を聞く
- 名前とアイコンを変更できる
 - └ init()関数を再度呼ぶだけ
- メッセージを送らなくても表示を更新できる
 - └ read()関数を呼ぶだけ



アプリの設定の変更

```
// アプリの設定  
var limit = 5;  
var room = "chat";  
var myName = null;  
var myIcon = "face";
```

アイコン名を設定できるようにする



init()関数の変更

```
// アプリの初期化処理
```

```
function init(){
```

```
  myName = prompt("なまえ");
```

```
  myIcon = prompt("アイコン名");
```

```
  read();
```

```
}
```

アイコン名を聞く



send()プログラムの変更

```
function send() {  
  // フォームの値を取得  
  var message = document.getElementById("message").value;  
  document.getElementById("message").value = "";  
  // チャットレコードの作成と反映  
  new chatTable()  
    .set("message", message)  
    .set("name", myName)  
    .set("icon", myIcon)  
    .save()  
    .then(function() {  
      read();  
    })  
    .catch(function(error){  
      // もしsave()が失敗したらエラーをログに書き出す  
      console.log(error);  
    });  
}
```

iconを保存する

read()関数のアイコン部分の小規模な変更

```
for (var i = 0; i < records.length; i++) {  
  // 投稿日の日付オブジェクトを生成  
  var date = new Date(records[i].get("createDate"));  
  var figure = document.createElement("figure");  
  var icon = document.createElement("i");  
  
  if (records[i].get("icon")) {  
    icon.textContent = records[i].get("icon");  
  } else {  
    icon.textContent = "face";  
  }  
  
  icon.classList.add('material-icons');  
  var figcaption =  
document.createElement("figcaption");  
  figcaption.textContent = records[i].get("name");  
  figure.appendChild(icon);  
  figure.appendChild(figcaption);  
}
```

DBにアイコンがあれば適応
なければ"face"を適応



最終カスタム版 (hard)

チャットアプリ開発

Lv7

Lv7で実装すること

- 複数画面の実装
 - └ 今回はCSSで切り替えます
- 開始画面を付ける
 - └ 脱prompt()
- 複雑なフォーム対応
 - └ アイコンを一覧から選べる
 - └ チャットルームを一覧から選べる

チャットアプリ - HOME

チャットアプリです。マナーや法令を守って利用しましょう。



開始

Lv7のHTML文 1/2 ※ アイコン一覧は長いので一部省略

```
<body>
  <section id="home">
    <header>
      <h1>チャットアプリ - HOME</h1>
    </header>
    <p class="descriptive_text">チャットアプリです。マナーや法令を守って利用しましょう。</p>
    <section class="name_section">
      <input type="text" id="myName" value="" placeholder="なまえ">
    </section>
    <form id="home_form">
      <section class="icon_section">
        <ul class="icon_select">
          <li><input name="icon" type="radio" id="icon0" value="assignment_ind"><label for="icon0"
class="material-icons">assignment_ind</label></li>
          <li><input name="icon" type="radio" id="icon1" value="emoji_emotions"><label for="icon1"
class="material-icons">emoji_emotions</label></li>
        </ul>
      </section>
      <section class="room_section">
        <select name="room">
          <option value="chat">チャットルーム(一般)</option>
          <option value="chata">チャットルーム(A)</option>
          <option value="chatb">チャットルーム(B)</option>
          <option value="chatc">チャットルーム(C)</option>
        </select>
      </section>
      <input type="button" class="startbtn" value="開始" onclick="start()">
    </form>
  </section>
```

Lv7のHTML文 2/2

※こちらはほぼ今まで通り、mainセクション内にheaderやfooterを内包

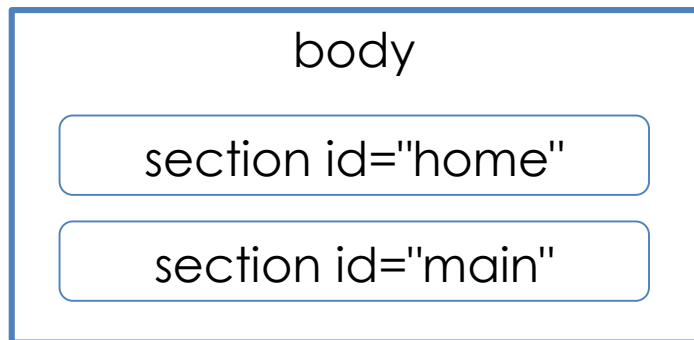
```
<body>
  <section id="main">
    <header>
      <i class="material-icons left_btn" onclick="location.reload()">home</i>
      <h1>チャットアプリ</h1>
      <i class="material-icons right_btn" onclick="read()">update</i>
    </header>
    <ul id="chat">
    </ul>
    <footer>
      <form id="message_form">
        <input type="text" id="message" value="" placeholder="ひとこと">
        <input type="button" value="送信" id="sendbtn" onclick="send()">
      </form>
    </footer>
  </section>
</body>
```

Lv7のCSS ※mainのセクションを非表示にする

```
#main {  
  display: none;  
}
```

body内に2つのページが存在する

- 必要に応じて表示・非表示を切り替える
 - 起動時は「home」を表示
 - 開始時は「main」を表示



新関数 start()の実装

- 役割的にはLv6までのinit()関数と似たような処理
- 主な相違点はフォームの値を変数に取り込む部分

```
// home画面のスタート処理
```

```
function start(){
```

```
    myName = document.getElementById("myName").value;
```

```
    myIcon = document.getElementById("home_form").icon.value;
```

```
    room = document.getElementById("home_form").room.value;
```

```
    chatTable = ncbm.DataStore(room);
```

```
    document.getElementById("home").style.display = "none";
```

```
    document.getElementById("main").style.display = "block";
```

```
    read();
```

```
}
```

フォームの値を取得

指定されたチャットルームで
DBテーブル選択

表示・非表示切り替え

init()関数の修正

■ 今回はやること無し、消しても良い

- 将来的に、起動画面で何か初期化処理を行いたい場合には引き続きinit()に記述する。

```
// アプリの初期化処理
function init(){
// 今回はやることなし
}
```