「情報Ⅱ オンライン学習会」
～情報システムを作ってみよう～

明星大学　長慎也

https://bitarrow.eplang.jp/?joho2

「情報システム」とは？

- 複数のシステム
  - クライアント・サーバシステムとも
- 複数人で開発する
- テストする
  - テストケースの作成
  - 単体テスト・統合テスト

情報システムを作るには

- 複数のシステム（≒複数のコンピュータ）を用意する必要がある
- 複数のシステムが「情報通信ネットワーク」で接続されている必要がある
  - ほとんどのシステムでは，情報通信ネットワーク＝インターネット(TCP/IP)

高校で演習する上での障壁

- 必要な開発環境を教室のPCにインストール
  - 管理者権限が必要
- ネットワーク通信をする
  - セキュリティの関係上，制限されている

→難しい！

高校での演習に適した環境

- BitArrow
  - Webブラウザがあれば，開発から実行までが演習できる
  - 多数のプログラミング言語に対応
    - JavaScript/ Python/ドリトル/簡易C/DNCL/Tonyu…
  - 他のシステムやデータと連携が容易
    - ファイル共有・簡易DB
    - 組み込み機器
    - ユーザが作成したプログラム間の通信←情報システム！
  - 教員による管理機能
    - クラス登録
    - ユーザ登録
    - ログ閲覧

情報システムを一から作ってみよう
～グループチャットを例に～

- サーバ側
  - 書き込みの追加
  - 書き込みの読み出し

- クライアント（ブラウザ）側
  - 書き込みフォームの表示
    - 入力欄
    - 書き込みボタン
    - 書き込み表示欄
  - 書き込みの表示
  - 書き込み追加の実行

Pythonで作成

HTML+JavaScriptで作成

Bit Arrowの準備をしよう

- https://bitarrow.eplang.jp/
- 教員の方は，「授業利用に向けた準備」から，教員ユーザ登録ができます．
- 教員ユーザでできること
  - クラスの作成
  - 生徒用のアカウントの作成
  - ログ（生徒の活動）の閲覧
- 今回の講演では，皆様用のクラスを用意しています

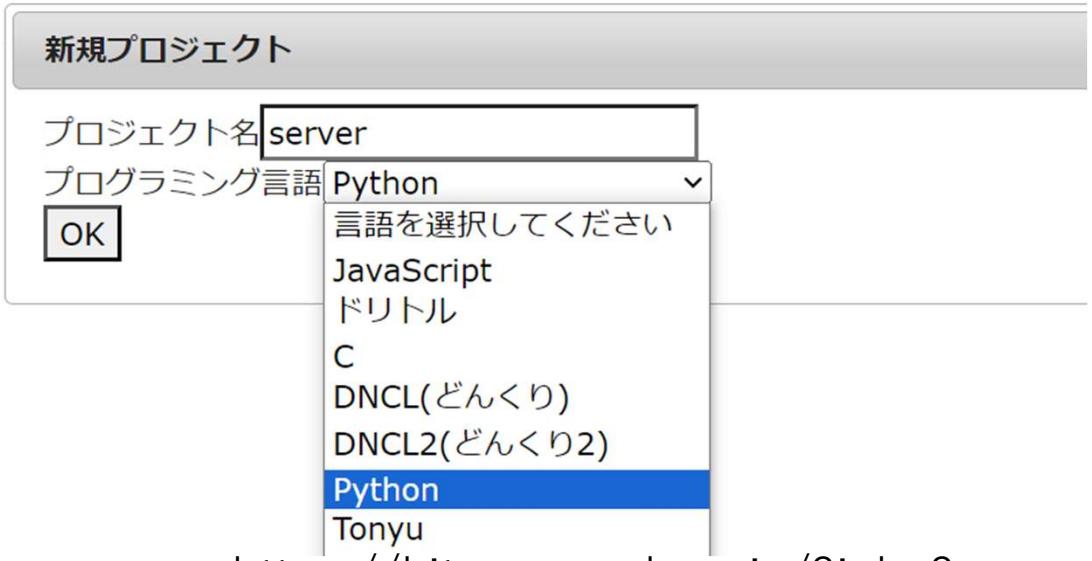
今回の講演で利用できるクラス

- https://bitarrow3.eplang.jp/bitarrow/
  - クラス名 `joho2_2402`
  - ユーザ名 *<<任意の半角英数>>*
    - 他の方とかぶりにくいものをお選びください
  - パスワード　（空欄）

※講演終了後，こちらのpptxをご覧になりながらでも作業可能です．

プロジェクトの作成

- サーバ側のプロジェクト
  - プロジェクト名： server（←例．何でもよいです）
  - 言語：　Python



サーバ側：書き込みの追加
ファイルの作成

- ファイル→新規

add_chatの中身

add_chat

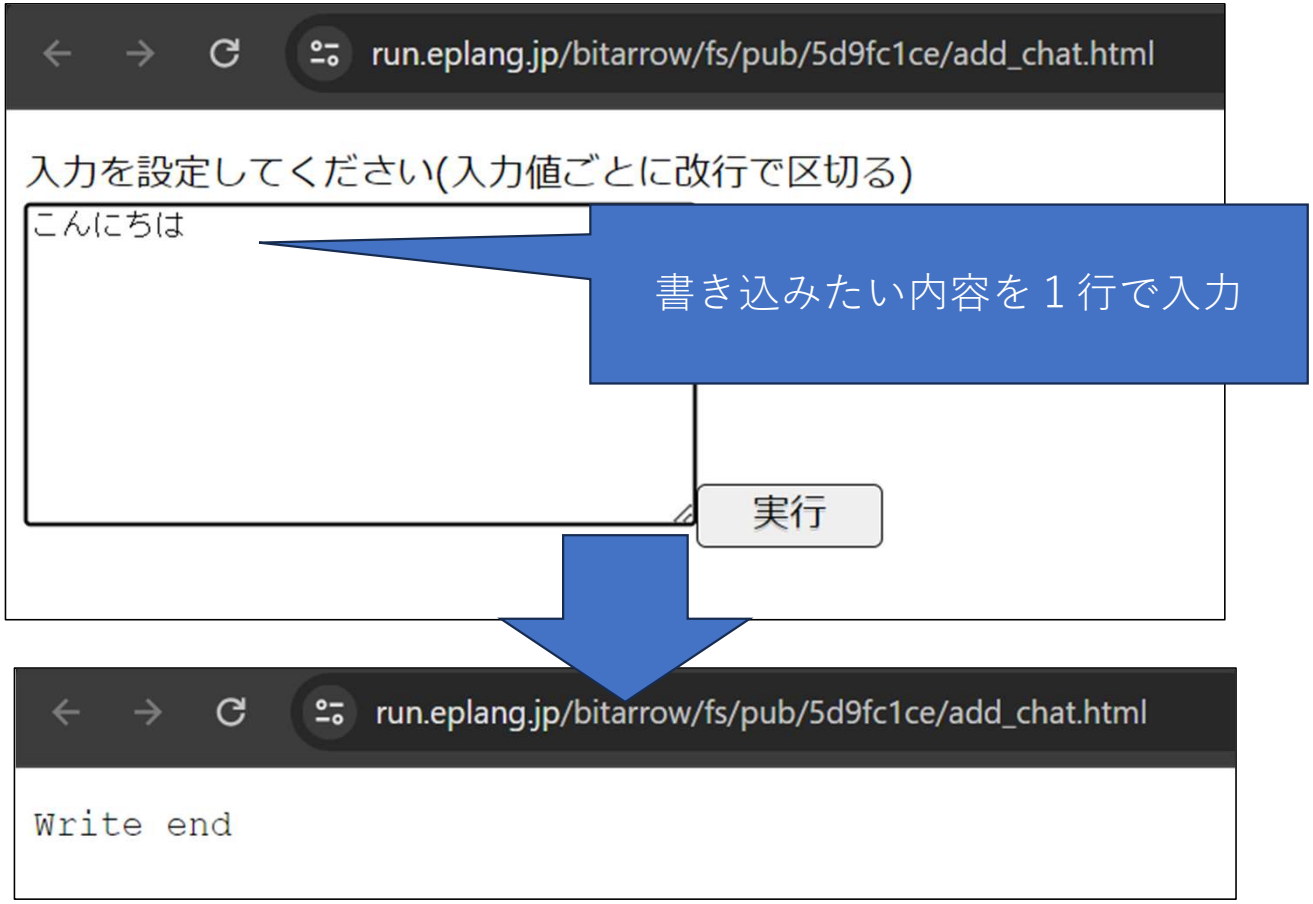
```
line=input()
f=open("user/chat.txt","a")
f.write(line+"¥n")
f.close()
print("Write end")
```

- 書き込む内容を入力
- ファイルを追記モードで開く
- 書き込む内容を追記
- ファイルを閉じる
- メッセージを表示

実行

- 同じ画面でも実行できますが，今回は「別ページで表示」を使います．

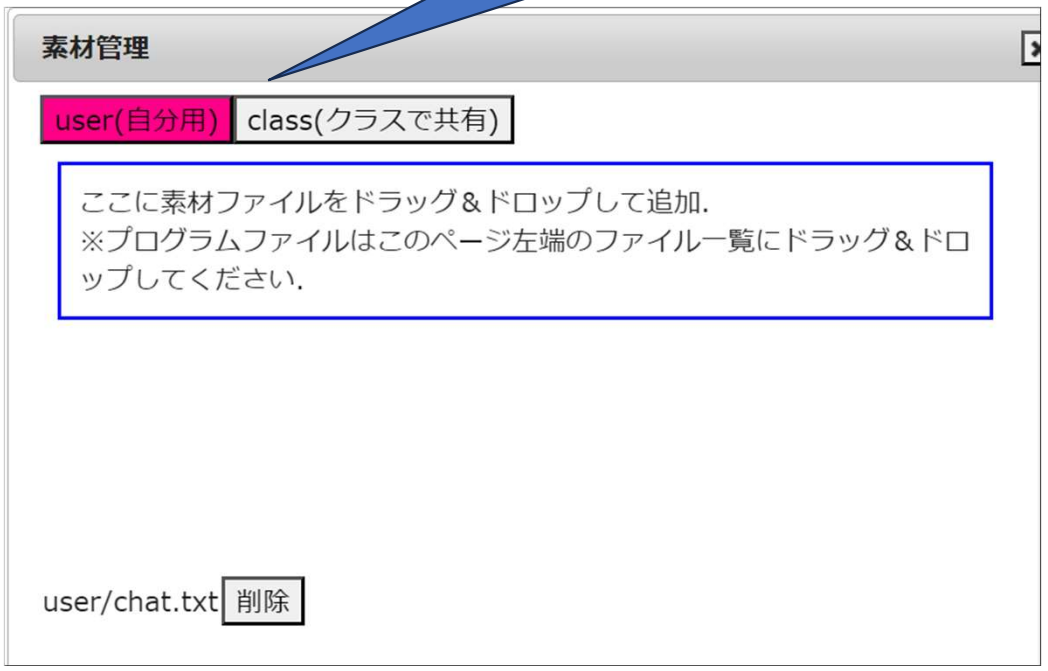
入力を行う


run.eplang.jp/bitarrow/fs/pub/5d9fc1ce/add_chat.html
入力を設定してください(入力値ごとに改行で区切る)
こんにちは
書き込みたい内容を１行で入力
実行
run.eplang.jp/bitarrow/fs/pub/5d9fc1ce/add_chat.html
Write end

ファイルが書き込まれたｋの確認

- 「ファイル」→「素材管理」
  - user/chat.txtをクリックして中身を確認

出ない場合は，userとclassを
交互に押して最新情報に更新



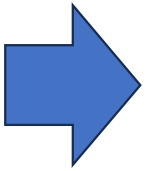
サーバ側：書き込みの読み出し

read_chat

```
f=open("user/chat.txt","r")
for i in f:
    print(i,end="")
f.close()
```

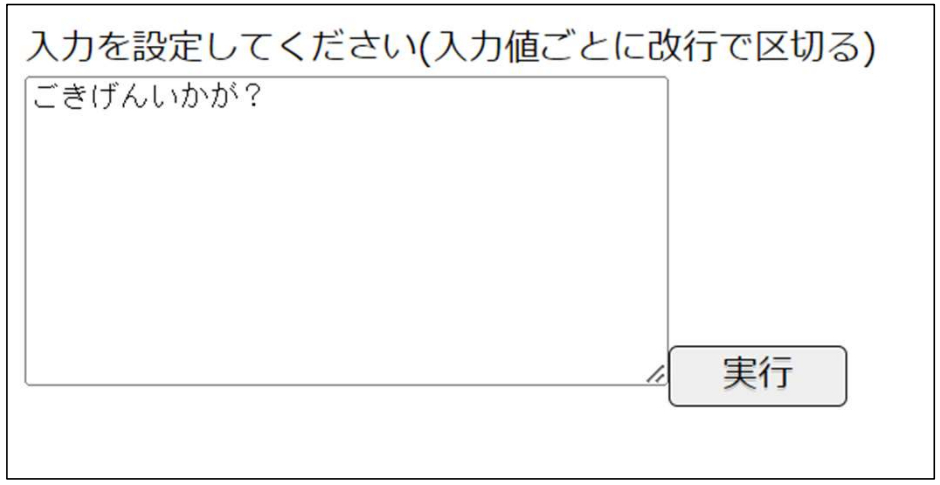
- ファイルを読み込みモードで開く
- 書き込まれた行ごとに
  - 内容を表示
- ファイルを閉じる

- add_chatと同様に作成して，実行してみると.



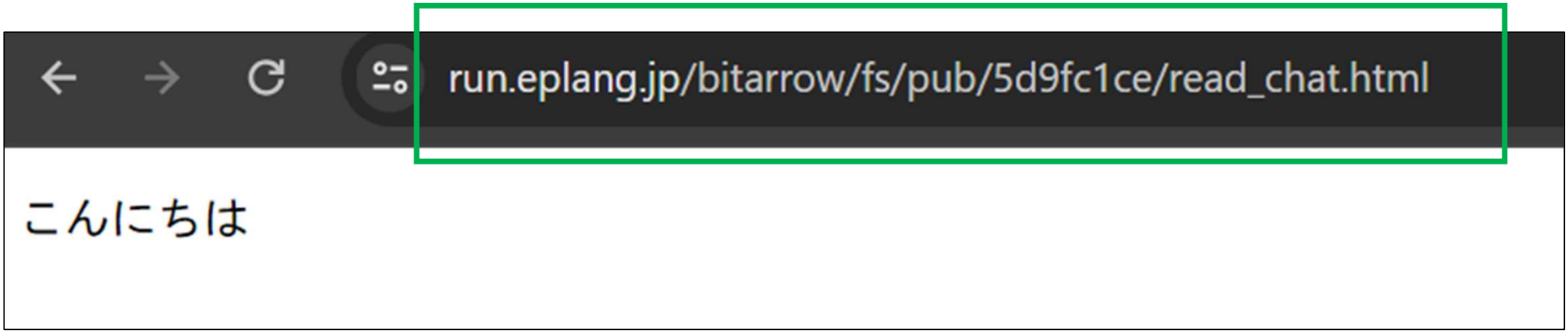
何件か書き込んでみよう（単体テスト）

- add_chatを実行
- read_chatを実行して，追記されているかを確認



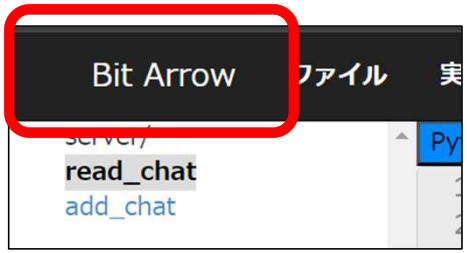
URLをメモしておきましょう

- 実行したときにアドレスバーに出ているURL↓（ユーザによって変わります）をメモしておきましょう
- **クライアント側で使います！**



クライアント側のプログラム

- 最初の画面に戻る
- プロジェクトをもう１個作成
  - プロジェクト名： `client`（←例. 何でもよいです）
  - 言語： JavaScript



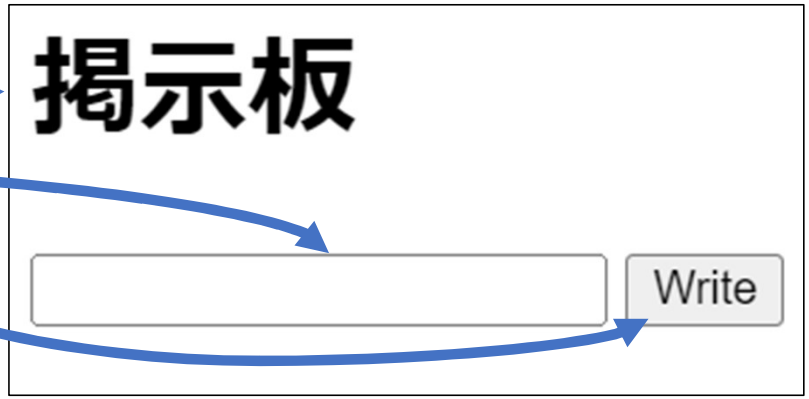
画面を作る

- ファイル→新規
  - 適当な名前（Mainなど）
- HTML　を選択

画面（HTML）の内容

Main （HTML）

```
<html>
<h1>掲示板</h1>
<input name="text"/>
<button name="write">Write</button>
<br/>
<div name="result"></div>
<pre name="hist"></pre>
</html>
```

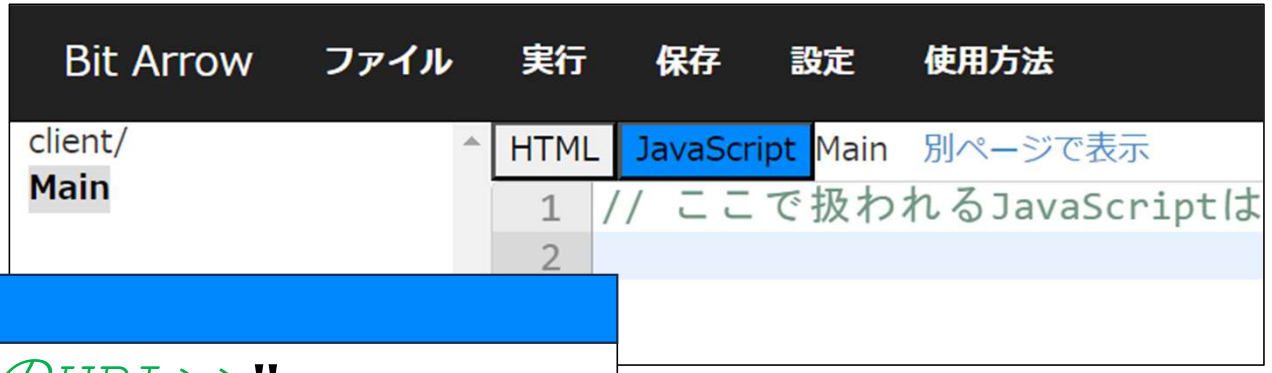


実行結果の表示欄（見えていない）

書き込み履歴の表示（見えていない）

クライアントの動作（JavaScript）の作成
書き込みの表示

- JavaScript　を選択



Main　(JavaScript)

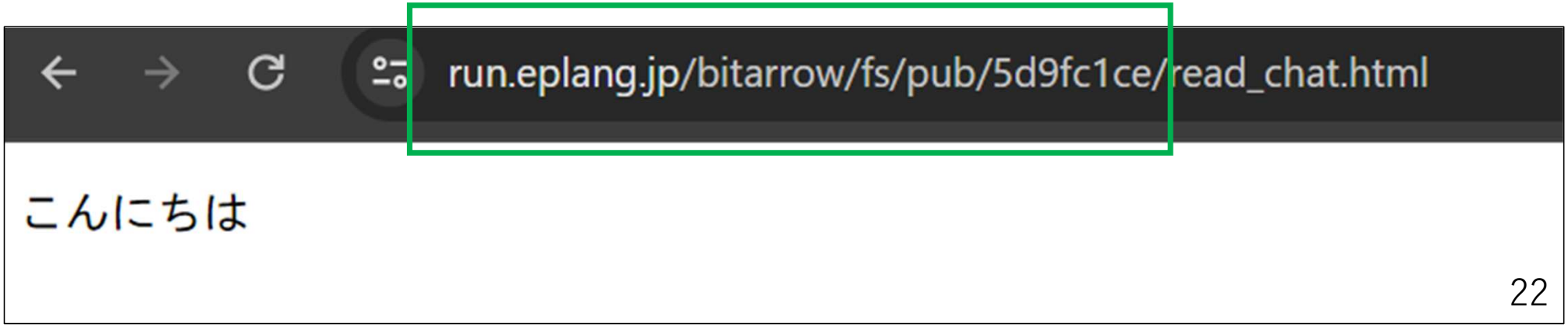
```
serverTop="<<サーバ側のURL>>";
readUrl=serverTop+"read_chat.html";
read();
function read() {
    r=callServer(readUrl);
    setText("hist",r);
}
```

- サーバ側のURLを指定
- read関数を呼び出す
- read関数の定義
  - サーバ側のプログラムを呼び出す
  - 結果を書き込み履歴に表示

サーバと連携するときの設定

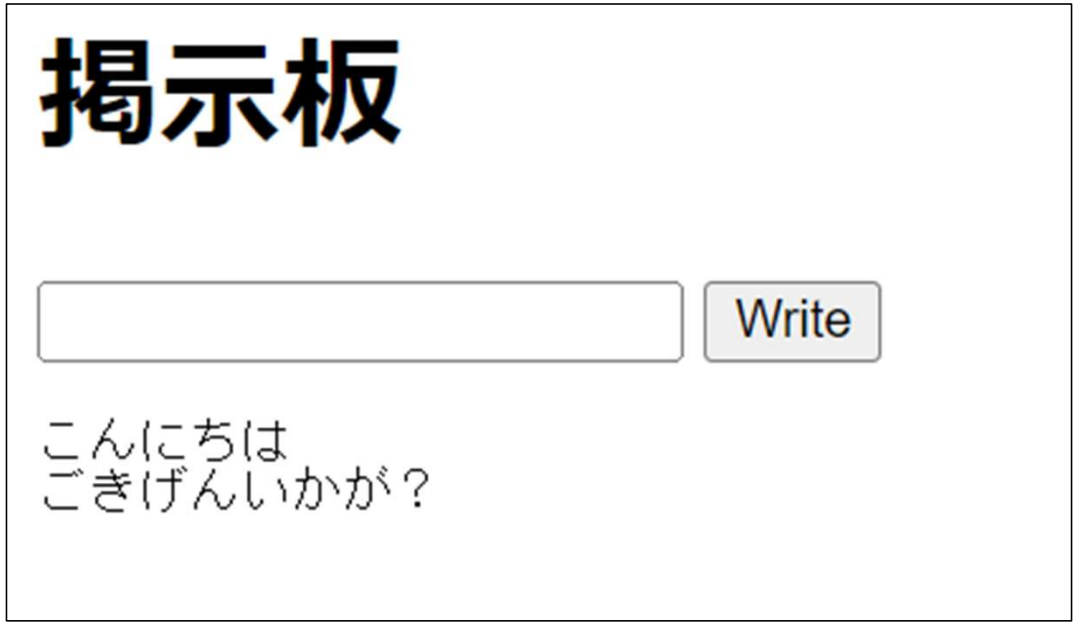
```
serverTop="https://run.eplang.jp/bitarrow/fs/pub/5d9fc1ce/";
readUrl=serverTop+"read_chat.html";
```

- serverTop には，先ほどメモしておいたサーバプログラムのURL（read_chat.htmlより前まで）を指定してください



テストしてみよう（書き込みの表示）

- 「何件か書き込んでみよう」のスライドで，サーバ側で実行したときの書き込みが表示されるか？



クライアントの動作（JavaScript）の作成
書き込み追加の実行(太字部分を追加)

Main　(JavaScript)

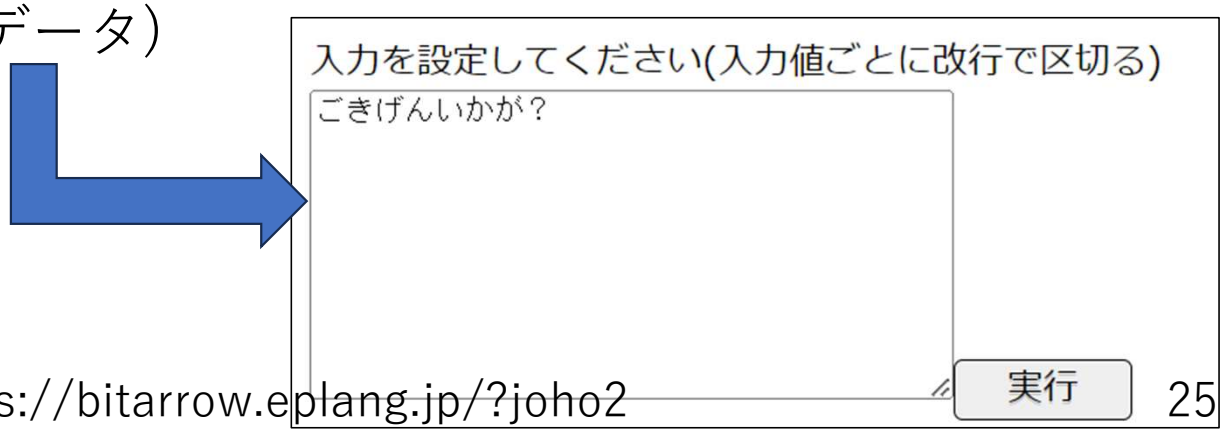
```
serverTop="https://run.eplang.jp/bitarrow/fs/pub/5d9fc1ce/";
readUrl=serverTop+"read_chat.html";
addUrl=serverTop+"add_chat.html";
onClick("write",write);
read();
function write() {
    setText("result","Running at Server...");
    result=callServer(addUrl,getText("text"));
    setText("result",result);
    read();
}
function read() {
    r=callServer(readUrl);
    setText("hist",r);
}
```

- サーバ側のURLを指定
- ボタンがクリックされたら，write関数を呼び出す
- write関数の定義
  - サーバ側のプログラムを呼び出す
  - 実行結果の表示欄に結果を表示

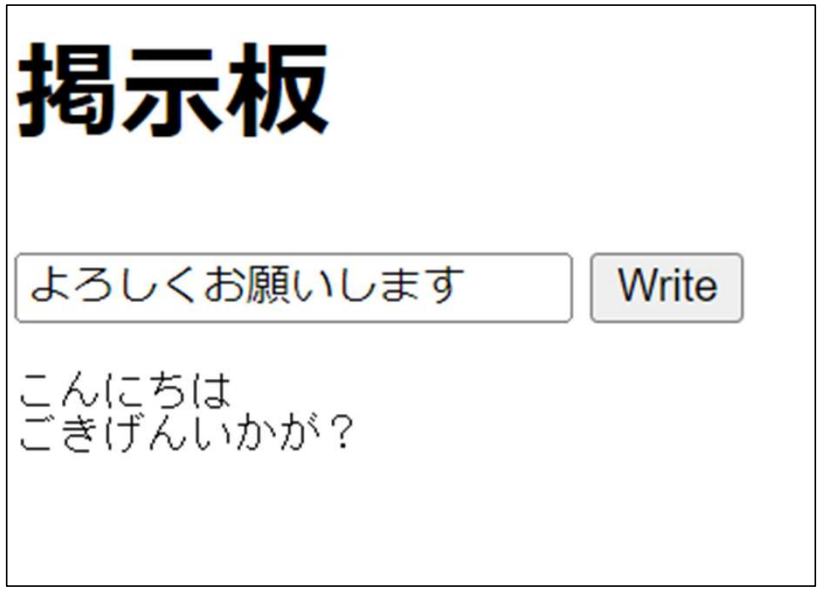
……ちょっと長いですね

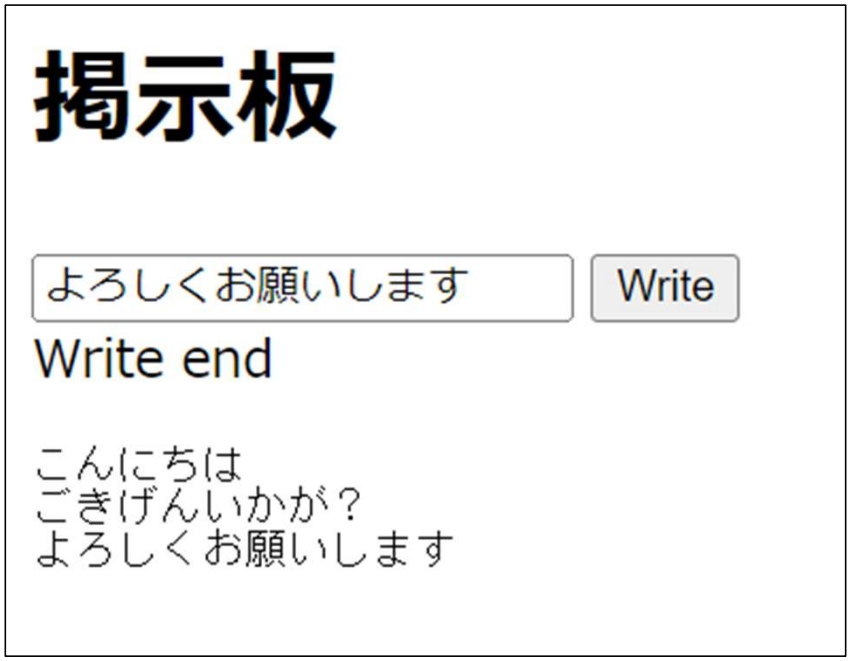
JavaScriptについての参考情報

- https://bitarrow.eplang.jp/　の「JavaScriptを使う」→「ゲームを作る」　あたりに，主な命令の使い方の解説があります
  - onClick
  - getText
  - setText
- サーバと通信するための命令
  - callServer(URL, 入力データ)



テストしてみよう（書き込み追加の実行）





クライアント側で単体テストをするには

```
log="";
serverTop="https://run.eplang.jp/bitarrow/fs/pub/5d9fc1ce/";
readUrl=serverTop+"read_chat.html";
addUrl=serverTop+"add_chat.html";
read();
onClick("write",write);
function read() {
    r=callServer(readUrl);
    setText("hist",r);
}
function write() {
    setText("result","Running at Server...");
    result=callServer(addUrl,getText("text"));
    setText("result",result);
    read();
}
function callServer(url, param) {
    if (url===readUrl) {
        return log;
    } else if (url===addUrl) {
        log+=param+"¥n";
        return "Write End";
    }
}
```

- ログデータを文字列で保存

- callServerを書き換えて，サーバと通信しないようにして実行

※「ファイル」→「コピー」で別のファイルに作成することをおすすめします

授業の進め方（一例）

- クライアント側か，サーバ側か，役割を決めよう
- それぞれ作成して，単体テストをしよう
  - どんなテストをすればよいか，考えてみよう
- それぞれのプログラムが連携できるか，結合テストをしよう
  - どんなテストをすればよいか，考えてみよう

- 自分たちでシステムを改造してみよう
  →次回をお楽しみに！