

○はじめに

< Javascriptに関する私の所感 > Day18現在

- ・イメージ

- ・メモリ管理が不要

- ・データ型を気にする必要がない

- ・特異的仕様

- ・比較演算子

- 厳密等価演算子 (なんじゃこりゃ)

- ・配列

- 次元数の異なる配列？(ビックリ)

- 『Javascriptでは多次元配列を直接サポートしていない』の記載

- (解釈) ⇒ 実は、配列の各要素に別の配列(先頭)を格納し 多次元配列を実現**

- ・オブジェクト

- 内部構成の変更が可能(キーの追加、削除)

- ・クロージャ、フィルタ、マップ・・・etc

○Presentation作成にあたり 織り込みたいこと

<講義内容から取り入れ>

配列

オブジェクト

API

クロージャ

フィルタ

マップ

<ブラウザ駆動 → 見た目の動き>

インターバル

## ○仕様

名称 **Hello, World** and Greeting (プログラミング言語入門コースの定番)

機能 **ゲスト名から 既定の呼出し順で挨拶分を表示する**

Welcome 1

Welcome 2, meet 1

Welcome 3, meet 1 and 2

Welcome 4, meet 1, 2 and 3

Welcome 5, meet 1, 2, 3 and 4

<Lesson 11のナイトメア>

- ・最初に'Welcome'と最終ゲスト
- ・ゲストが複数の場合は、'meet'の後に最初のゲスト
- ・2番目からは','で繋ぎ 最後から2番目の場合は'and'で繋ぐ

## 入力

必須 **ゲストの名前をカンマで区切り テキストボックスへキー入力**

オプション 出力②へのパラメータとして国を選択

## 出力

①Introduction **既定順で 画面へ表示する。但し、1行毎に1秒のインターバルを設ける**

②map 選択された国の挨拶で

③filter アルファベットのみ表示

④sort ゲスト名でソート

⑤array そのまま

○使用法

The screenshot shows a web browser window with the address bar displaying '3181085.github.io/presentation/'. The page has a purple background and contains the following text and elements:

- Header:** 'Hello, World and Greeing. Test!' (Note: 'Greeing' is misspelled as 'Greeing' in the image).
- Instruction:** 'Input guest names with CSV and click "SUBMIT"'.
- Japanese Instruction:** 'ゲスト名をCSVで入力しSUBMITを押下。' (Enter guest names in CSV and press SUBMIT).
- Form:** A text input field containing '1, 2, 3, 4, 5' and a 'SUBMIT' button.
- Buttons:** A row of buttons: 'Japanese' (with a dropdown arrow), 'array', 'sort', 'filter', 'map', and 'Introduction'. The 'array' button is circled in red.
- Output:** A list of welcome messages:
  - welcome 1
  - welcome 2, meet 1
  - welcome 3, meet 1 and 2
  - welcome 4, meet 1, 2 and 3
  - welcome 5, meet 1, 2, 3 and 4

Annotations (Callouts):

- Blue Callout:** 'テストデータを入力(キー操作を省)' (Enter test data (save key operations)).
- Pink Callout 1:** '①ゲスト名を入力したらSUBMITを押' (① After entering guest names, press SUBMIT).
- Pink Callout 2:** '②各処理を実行しゲスト名を表示す' (② Execute each process and display guest names).

## ○最後に気づいた点

### メモリ操作で間違った解釈

関数から配列を返すと参照渡し

```
function copy(array) {  
    let retArray = [];  
    retArray = array;  
    return retArray;  
}
```

newArray = copy(originalArray);                      ← newArrayには別のインスタンス？

間違った解釈：引数または、戻り値を受けるとそのインスタンス(実体)が生成される

⇒ 正: されない。originalArrayがそのまま返る(低レベル言語と同等仕様))

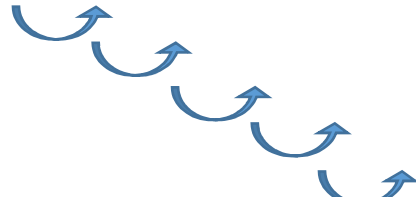
## オブジェクト

キー名を間違えてもエラーにならない

```
{  
    prefix: "Mr.",  
},  
{  
    perfix: "Hola",  
},
```

○アルゴリズム：実装したバブルソートのアルゴリズム

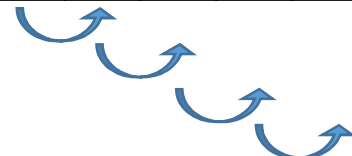
アドレス	0	1	2	3	4	5
値	5	1	3	2	4	6



内ループ  $j < n - 1 - i$

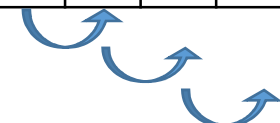
アドレス	0	1	2	3	4	5
値	1	3	2	4	5	6

← 1 回目のループで最後の要素が確定



アドレス	0	1	2	3	4	5
値	1	2	3	4	5	6

← 2 回目のループで最後から 2 番目の要素が確定



アドレス	0	1	2	3	4	5
値	1	2	3	4	5	6

← 3 回目のループで最後から 3 番目の要素が確定



アドレス	0	1	2	3	4	5
値	1	2	3	4	5	6

← 4 回目のループで最後から 4 番目の要素が確定



アドレス	0	1	2	3	4	5
値	1	2	3	4	5	6

← 5 回目のループで最後から 5 番目の要素が確定

外ループ  
 $i < n - 1$