Core JavaScript #10

4. コールバック関数（1）

Kim Donghee

4. コールバック関数

目次

Quiz

Q. 以下のコードの実行結果で正しいのはどれか

```
var arr = ['スイカ','レモン','カルピス'];
arr = arr.map(function (index, current) {
  console.log(this);
  return current + 'サワー';
});
console.log(arr);
```

A.

① スイカ・レモン・カルピス / [スイカサワー・レモンサワー・カルピスサワー]

② window window window / [0サワー・1サワー・2サワー]

③ スイカ・レモン・カルピス / [0サワー・1サワー・2サワー]

④ window window window / [スイカサワー・レモンサワー・カルピスサワー]

コールバック関数とは

- コールバック関数
  - 他のコードにパラメータとして引き渡す関数
  - 引き受けたコードはコールバック関数を必要に応じて適切に実行

日常生活の例

チーム朝会に参加する２つのパターン

- メンバーA

```
朝10時行われることを覚えておく
-> 9時ぐらいに時間を確認し、朝会まであと1時間であることを確認
-> 9時50時ぐらいに時間を確認し、朝会まであと10分であることを確認
-> 9時55時ぐらいに時間を確認し、朝会まであと5分であることを確認
-> zoomのリンクを開いてMTGに参加する
```

- メンバーB

```
カレンダーにスケジュールをセットしておく
–> 別のタスクを進める
–> チーム朝会開始５分前の通知が来る
–> zoomのリンクを開いてMTGに参加する
```

メンバーAは時間を確認する行為を何回も行い、その結果で参加するかまいかを判断したが、メンバーBは**時間を確認する行為**（コールバック）をカレンダーに移譲し、カレンダーの通知が来た途端すぐにミーティングに参加した。

`callback`は`呼び出す`という意味の`call`と`戻ってくる・返る`という意味を持つ`back`を組み合わせて作られた合成語。

コールバック関数は関数またはメソッドに引数として引き渡す関数で、制御権も合わせて移譲される。

制御権

例を見て理解しよう。

呼び出す時点

```
var count = 0;
var callbackFunc = function () {
  console.log(count);
  if (++count > 4) clearInterval(timer);
};
var timer = setInterval(callbackFunc, 300);

// -- 実行結果 --
// 0 (0.3秒)
// 1 (0.6秒)
// 2 (0.9秒)
// 3 (1.2秒)
// 4 (1.5秒)
```

timerには`setInterval`のIDが割り当てられる。
`setInterval`に`callbackFunc`と`300`を引数として設定した。
`timer`を実行すると、0.3秒毎に無名関数が自動的に実行され、countを1増加させて4より大きい場合は終了する流れになる。

code	呼び出し元	制御権
callbackFunc();	ユーザ	ユーザ
setInterval(callbackFunc, 300);	setInterval	setInterval

このようにコールバック関数の制御権は書いたユーザでなくsetIntervalが持つことになる。

引数

```
var newArr = [10, 20, 30].map(function (current, index) {
  console.log(current, index);
  return current + 5;
});

// -- 実行結果 --
// 0 10
// 1 20
// 2 30
// [15, 25, 35]
```

配列の要素を順番で巡回しながらコールバック関数を実行する。

ログ出力後5を足す。

配列の `map` は以下のような仕組みになっている。

```
Array.prototype.map(callback[, thisArg])
```

`thisArg` は省略可能で、例のようにコールバックだけ渡すこともできるようになっている。
その時はグローバルオブジェクトがバインドされる。

```
var newArr = [10, 20, 30].map(function (index, current) {
  console.log(index, current);
  return current + 5;
});

// -- 実行結果 --
// 10 0
// 20 1
// 30 2
// [5, 6, 7]
```

コールバック関数のパラメータの名前を変えても第１引数を配列の要素、第２引数をindex番号として扱うことになっているため、indexに5を足して上書きするような結果になってしまう。

コールバックを受け取った関数は定義されている通り振舞うので、
呼び出し元がコールバック関数のパラメータに値を指定する制御権を持つ。

this

コールバック関数も関数のため、グロバールオブジェクトを参照するが
別途の値をthisとして指定場合はそれを参照することになる。

`Array` の `map` をコアな部分だけにすると以下のような形になる。

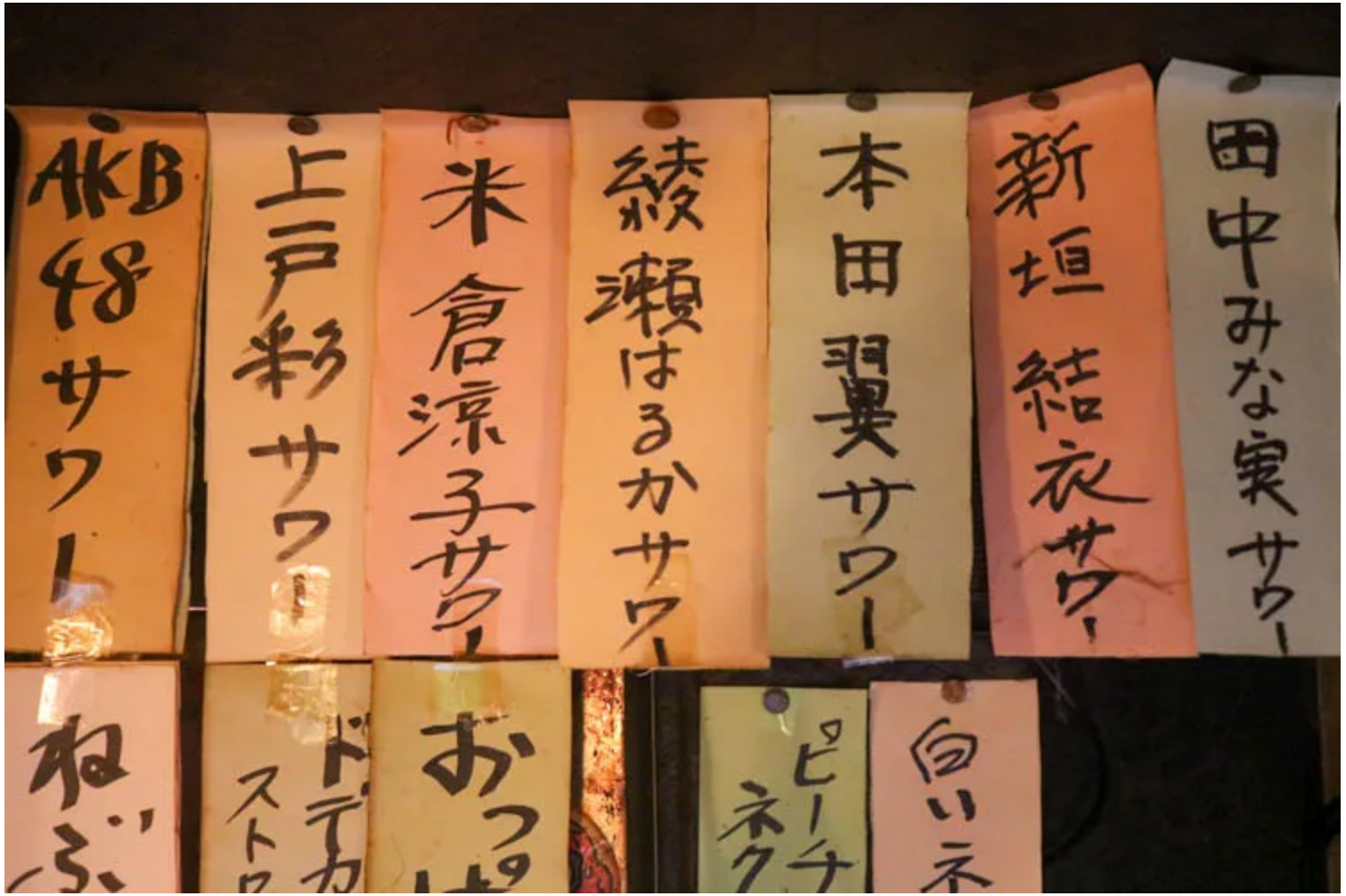
```
Array.prototype.map = function (callback, thisArg) {
  var mappedArr = [];
  for (var i = 0; i , this.length; i++) {
    var mappedValue = callback.call(thisArg || window, this[i], i, this);
    mappedArr[i] = mappedValue;
  }
  return arr;
}
```

`call` は第 1 引数で渡されたパラメータを `this` にバインドする。
第 1 引数を省略したらグロバールオブジェクトである `window` がセットされる。

```
var newArr = [10, 20, 30].map(function (current, index) {
  console.log(this);
  console.log(current, index);
  return current + 5;
});
// window
// 10 0
// window
// 20 1
// window
// 30 2
document.body.innerHTML += "<button id='a'>Click me</button>";
document.getElementById('a')
  .addEventListener('click', function (e) {
    console.log(this, e);
  }
);
// <button id='a'>Click me</button>
// MouseEvent { isTrusted: true, ... }
```

`addEventListener` のthisがボタンエレメントを指しているのは、
`map` と違って内部でコールバック関数を実行する際にthisが設定されていないと
呼び出し元をthisとして割り当てるからだ。

THE END



https://www.nomooo.jp/column/131107/

NEXT

4. コールバック関数

1. コールバック関数とは
2. 制御権
3. **コールバック関数は関数だ**
4. **コールバック内部のthisにbindする方法**
5. **コールバック地獄と非同期制御**

正解

Q. 以下のコードの実行結果で正しいのはどれか

```
var arr = ['スイカ','レモン','カルピス'];
arr = arr.map(function (index, current) {
  console.log(this);
  return current + 'サワー';
});
console.log(arr);
```

A.

① スイカ・レモン・カルピス / [スイカサワー・レモンサワー・カルピスサワー]

② window window window / [0サワー・1サワー・2サワー]

③ スイカ・レモン・カルピス / [0サワー・1サワー・2サワー]

④ window window window / [スイカサワー・レモンサワー・カルピスサワー]

正解

Q. 以下のコードの実行結果で正しいのはどれか

```
var arr = ['スイカ','レモン','カルピス'];
arr = arr.map(function (index, current) {
  console.log(this);
  return current + 'サワー';
});
console.log(arr);
```

A.

~~① スイカ・レモン・カルピス / [スイカサワー・レモンサワー・カルピスサワー]~~

② window window window / [0サワー・1サワー・2サワー]

~~③ スイカ・レモン・カルピス / [0サワー・1サワー・2サワー]~~

~~④ window window window / [スイカサワー・レモンサワー・カルピスサワー]~~