JavaScript におけるシャローコピーとディープコピー

概要

JavaScriptでオブジェクトをコピーする際、「シャローコピー」と「ディープコピー」という2つの異なる方法があります。これらの違いを理解することは、特に複雑なデータ構造を扱う際に非常に重要です。

シャローコピー（浅いコピー）

シャローコピーは、最上位のプロパティだけをコピーします。オブジェクトの中にネストされたオブジェクトがある場合、そのネストされたオブジェクトは参照のみがコピーされます。つまり、ネストされたオブジェクトは元のオブジェクトと新しいオブジェクトの間で共有されます。

シャローコピーの作成方法：

1. `Object.assign()` メソッド
2. スプレッド構文 `{...obj}`
3. `Array.slice()` （配列の場合）
4. `Array.from()` （配列の場合）
5. スプレッド構文 `[...array]` （配列の場合）

シャローコピーのサンプルコード：

```
// シャローコピーの例
const originalObj = {
  name: '田中',
  age: 30,
  address: {
    city: '東京',
    district: '新宿区',
  },
  hobbies: ['読書', '旅行'],
};
---
// 方法1: Object.assign()
const copyWithAssign = Object.assign({}, originalObj);

// 方法2: スプレッド構文
const copyWithSpread = { ...originalObj };
---
// 元のオブジェクトとコピーを比較
console.log(originalObj === copyWithAssign); // false - 別のオブジェクト
console.log(originalObj.address === copyWithAssign.address); // true - 同じオブジェクトの参照

// ネストされたオブジェクトを変更すると、両方のオブジェクトに影響する
originalObj.address.district = '渋谷区';
console.log(copyWithAssign.address.district); // "渋谷区" - 変更が反映される

// 配列のシャローコピー
const originalArray = [1, 2, [3, 4]];
const arrayCopy = [...originalArray];

originalArray[2].push(5);
console.log(arrayCopy[2]); // [3, 4, 5] - ネストされた配列は共有されている
```

ディープコピー（深いコピー）

ディープコピーは、オブジェクト内のすべてのレベルの値を完全にコピーします。ネストされたオブジェクトも含め、すべての値が完全に新しいコピーになります。

ディープコピーの作成方法：

1. `structuredClone()` (モダンブラウザとNode.js 17以降で使用可能)
2. JSON.parse(JSON.stringify()) - ただし関数やDate、MapなどのJSONに変換できない値は扱えない制限がある

ディープコピーのサンプルコード：

```
// ディープコピーの例
const originalObj = {
  name: '田中',
  age: 30,
  address: {
    city: '東京',
    district: '新宿区',
  },
  hobbies: ['読書', '旅行'],
  birthDate: new Date('1993-05-21'),
};
---
// structuredClone を使ったディープコピー
const deepCopy = structuredClone(originalObj);

// 元のオブジェクトとコピーを比較
console.log(originalObj === deepCopy); // false - 別のオブジェクト
console.log(originalObj.address === deepCopy.address); // false - 別のオブジェクト
---
// ネストされたオブジェクトを変更しても、他方には影響しない
originalObj.address.district = '渋谷区';
console.log(deepCopy.address.district); // "新宿区" - 変更が反映されない
---
// 配列のディープコピー
const originalArray = [1, 2, [3, 4]];
const deepArrayCopy = structuredClone(originalArray);

originalArray[2].push(5);
console.log(deepArrayCopy[2]); // [3, 4] - 変更が反映されない
---
// Dateオブジェクトもきちんとコピーされる
console.log(deepCopy.birthDate instanceof Date); // true
```

シャローコピーとディープコピーの比較

比較点	シャローコピー	ディープコピー
コピー範囲	最上位のプロパティのみ	すべてのレベルの値
ネストされたオブジェクト	参照のみコピー	完全にコピー
パフォーマンス	速い	遅い（データ量による）
実装の複雑さ	単純	やや複雑
変更の独立性	部分的	完全

実用的な使用例

シャローコピーの使用例：

- 単純なデータ構造のコピー
- パフォーマンスが重要な場合
- ネストされたオブジェクトへの変更を意図的に共有したい場合

ディープコピーの使用例：

- 複雑なデータ構造の完全な独立したコピーが必要な場合
- イミュータブルなデータ操作を行いたい場合
- 副作用を避けたい場合
- オリジナルデータの保存が必要な場合

まとめ

- **シャローコピー**：最上位のプロパティのみをコピーし、ネストされたオブジェクトは参照をコピーします。
- **ディープコピー**：すべてのレベルの値を完全にコピーし、元のオブジェクトとは完全に独立したコピーを作成します。
- 適切なコピー方法の選択は、データ構造の複雑さや、コピー後のデータの使用方法によって異なります。

現代のJavaScriptでは、`structuredClone()`がディープコピーを作成する最も簡単で効率的な方法として推奨されています。