JavaScript 配列（Array）学習資料

01. 要素の追加: push() vs concat()

- **解説**: push() は元の配列を直接書き換え（破壊的）、concat() は元の配列を残したまま新しい配列を返します。
- **コード**:

JavaScript

```
let 果物 = ["りんご", "バナナ"];
果物.push("みかん"); // 元の配列が変わる
let 新リスト = 果物.concat("ぶどう"); // 新しい配列を作る
```

- **実行結果**:
  - 元の配列: ["りんご", "バナナ", "みかん"]
  - 新リスト: ["りんご", "バナナ", "みかん", "ぶどう"]

02. ループ処理: map() vs forEach()

- **解説**: map() は各要素を加工して新しい配列を作りますが、forEach() は単に処理を繰り返すだけで何も返しません。
- **コード**:

JavaScript

```
let 数字 = [1, 2, 3];
```

let map 結果 = 数字.map(n => n * 2);

let each 結果 = 数字.forEach(n => n * 2);

- **実行結果**:
    - map 結果: [2, 4, 6]
    - each 結果: undefined

03. 条件抽出: filter()

- **解説**: 条件に合う（true を返す）要素だけを集めて、新しい配列を作成します。
- **コード**:

JavaScript

let 年齢 = [15, 20, 25, 12];

let 大人 = 年齢.filter(n => n >= 18);

- **実行結果**:
    - 大人: [20, 25]

04. 検索の違い: find() vs filter()

- **解説**: find() は最初に見つかった「値」を返し、filter() は見つかったもの「すべてが入った配列」を返します。
- **コード**:

JavaScript

let 名簿 = ["田中", "佐藤", "田中"];

let 一人 = 名簿.find(n => n === "田中");

let 全員 = 名簿.filter(n => n === "田中");

- **実行結果**:
    - 一人: "田中"
    - 全員: ["田中", "田中"]

05. 配列の初期化: length = 0

- **解説**: length プロパティを 0 に設定することで、最も効率的に配列を空にできます。
- **コード**:

JavaScript

let 買い物 = ["パン", "牛乳"];

買い物.length = 0;

- **実行結果**:
    - 買い物: []