📘 React 学習メモ（2025/04/10）

🔁 【1】.map() と memos.length の理解

● .map() の役割

- 配列の要素を1つずつ処理し、UIに展開する

```
{memos.map((memo, index) => (
  <li key={index}>{memo}</li>
))}
```

● memos.length

- 現在のメモの数を表示するのに使用

```
<p>現在のメモ数：{memos.length}</p>
```

🧪 【2】JSON.stringify() の使い所

● デバッグ表示用

- 配列やオブジェクトの中身を見やすく整形して表示

```
<pre>{JSON.stringify(memos, null, 2)}</pre>
```

- null: replacer指定なし
- 2: インデント2スペースで表示

✂️ 【3】削除機能の実装（handleDelete）

● 削除処理の中身

```
const handleDelete = (indexToRemove) => {
  setMemos(memos.filter((_, index) => index !== indexToRemove));
};
```

● 解説

- `.filter()` → 条件に一致した要素だけ残す
- `_`：使わないけど受け取る変数（ここでは中身）
- `index !== indexToRemove` → 削除されたindex以外を残す

📐 【4】Reactの作り方の順番を理解

🔻 基本の実装フロー

1. UIを分解（何を表示したいか？）
2. 状態（useState）を考える
3. JSXだけ先に書いて仮組み
4. イベント（onClickなど）で状態を変える
5. `.map()` でリスト表示
6. コンポーネントに分割（props設計）

🔄 【5】props分割での整理（役割分担）

● 親：MemoApp.jsx

```
<MemoList memos={memos} onDelete={handleDelete} />
```

- 状態とイベント関数を持つ
- 子に「データ」と「関数」を渡す

● 子：MemoList.jsx

```
<MemoItem
  key={index}
  memo={memo}
  index={index}
  onDelete={onDelete}
/>
```

- `.map()` で表示を繰り返す
- 孫（MemoItem）にpropsを渡す

● 孫：MemoItem.jsx

```
function MemoItem({ memo, onDelete, index }) {
  return (
    <li>
      {memo}
      <button onClick={() => onDelete(index)}>削除</button>
    </li>
```

```
    );
}
```

- 表示とボタンのみ担当
- 削除処理は親から渡された関数を呼ぶだけ

📦 【6】propsの書き方と意味

● propsの分割代入

```
function MemoItem({ memo, onDelete, index }) {
```

- `props.memo` などと書かなくてもOK
- コードが短く、読みやすくなる

● よくあるpropsパターン

名前	説明
`memo`	表示する文字列
`index`	メモの番号（識別子）
`onDelete`	親の削除関数（indexで指定）

🧱 【7】最終構成のファイル分け

```
components/
├── MemoApp.jsx      // 状態と処理を持つ親
├── MemoList.jsx     // リスト表示（子）
└── MemoItem.jsx     // 1つのメモ行（孫）
```

✅ 今日のまとめ

- Reactの流れと役割分担を理解
- 状態の保持・更新 → 表示 → イベント → 分割まで経験
- `props` の流れと 関数の受け渡し に慣れてきた

→ 明日は応用（編集・チェック機能、propsからの逆流）に挑戦！