# Todo4 解説（図解つき）

このドキュメントは src/practice/Todo4.jsx の処理を図で整理しながら技術的に解説します。必要ならこのMarkdownをWord(.docx)に変換してご利用ください。

## 1. コンポーネントの全体像

Todo4 (Function Component)  
├─ state: todos (Array<{ text: string, done: boolean }>)  
├─ state: inputs (string)  
├─ addTodo(): 追加処理  
├─ deleteTodo(index): 削除処理  
└─ render(): 入力欄 + 追加ボタン + 一覧

## 2. データフロー（図）

ユーザー入力  
 │ onChange(e)  
 ▼  
inputs (state) ──→ 入力欄の value に反映  
 │ Enter/クリック  
 ▼  
addTodo()  
 │ 不変更新([...todos, 新アイテム])  
 ▼  
todos (state) ──→ 一覧(map)に反映

* inputs: 入力欄のソース・オブ・トゥルース（Controlled Component）
* todos: 描画用配列。新しい配列を作って差し替える（不変更新）

## 3. 追加処理（addTodo）のポイント

if (inputs.trim() === '') return;

* ガード節(Guard Clause): 不正条件（空/空白のみ）を先頭で弾いて早期リターン
* String.prototype.trim(): 先頭/末尾の空白文字を除去
* 厳密等価(===): 暗黙の型変換のない安全な比較

setTodos([...todos, { text: inputs, done: false }]);  
setInputs('');

* 不変更新: スプレッド構文で新配列を作成 → Reactが差分検出し再レンダリング
* 入力欄クリア: Controlled ComponentなのでUIも即座に空になる

## 4. 削除処理（deleteTodo）のポイント

setTodos(todos.filter((\_, index) => index !== deleteIndex));

* Array.prototype.filter(): 非破壊メソッド（元配列を変更しない）
* 指定 index の要素を除外した新配列に置き換える → 不変更新

## 5. Controlled Component（入力欄）

<input value={inputs} onChange={e => setInputs(e.target.value)} />

* 表示値は常に state 由来（valueにinputsを束縛）
* onChangeでUI→stateの一方向データフロー

## 6. key と差分計算

{todos.map((todo, index) => (  
 <div key={index}> ... </div>  
))}

* サンプルでは簡素化のため index を key に使用
* 実運用では安定したIDをkeyにするのが推奨

## 7. 差分検出の概念図

旧: todos = [A, B, C]  
新: todos' = [...todos, D] → [A, B, C, D]  
  
旧配列 !== 新配列 なので参照の違いで変更を検出  
map内の key に基づき必要部分のみ再描画

## 8. Todo4.jsx 抜粋

import React,{ useState } from 'react';  
  
function Todo4(){  
 const [todos, setTodos] = useState([]);  
 const [inputs, setInputs] = useState('');  
  
 const addTodo = () => {  
 if (inputs.trim() === '') return;  
 setTodos([...todos, { text: inputs, done: false }]);  
 setInputs('');  
 }  
 const deleteTodo = (deleteIndex) => {  
 setTodos(todos.filter((\_, index) => index !== deleteIndex));  
 }  
 return (  
 <div>  
 <input value={inputs} onChange={(e) => setInputs(e.target.value)} />  
 <button onClick={addTodo}>追加</button>  
 {todos.map((todo, index) => (  
 <div key={index}>  
 {todo.text}  
 <button onClick={() => deleteTodo(index)}>削除</button>  
 </div>  
 ))}  
 </div>  
 );  
}  
export default Todo4;

## 9. Word(.docx) への変換

* このMarkdownをWordで開く/貼り付ければレイアウト付きドキュメントとして保存可能
* あるいは Pandoc を利用して変換:
  + pandoc Todo4\_Explanation.md -o Todo4\_Explanation.docx