🎓 Reactコンポーネント分割講義

1. コンポーネントを分ける理由

- 読みやすくなる
- 保守しやすくなる
- 再利用しやすくなる

2. 分割の判断軸はこれ！

✅ 汎用性

→ いろんな場所で再利用できるパーツは切り出そう

✅ 責務の分離

→ コンポーネントが「何をするか」が明確なら分けるべき！

3. 責務の分離：ToDo一覧の例

🔽 一覧全体（`ToDoList.tsx`）の責務

- ToDoを繰り返し表示する
- 一覧の構造や枠組みを担当

🔽 一覧の中の1行（`ToDoItem.tsx`）の責務

- タイトル、状態、日付の表示
- 1つのToDoの見た目を担当

✅ サンプルコード：ToDoList.tsx

```
type ToDoListProps = {
  todos: TTodo[];
};

export const ToDoList: React.FC<ToDoListProps> = ({ todos }) => {
  return (
    <div>
      <h2 className="text-xl font-bold mb-2">ToDo一覧</h2>
      <ul>
        {todos.map((todo) => (
          <ToDoItem key={todo.id} todo={todo} />
        ))}
      </ul>
    </div>
  );
};
```

✅ サンプルコード：ToDoItem.tsx

```tsx
type ToDoItemProps = {
  todo: TTodo;
};

export const ToDoItem: React.FC<ToDoItemProps> = ({ todo }) => {
  return (
    <li className="border-b py-2">
      <p
        className={todo.isCompleted ? 'line-through text-muted-foreground' : ''}
      >
        {todo.title}
      </p>
      <p className="text-sm text-muted-foreground">{todo.createdAt}</p>
    </li>
  );
};
```

4. 汎用性の例：Buttonコンポーネント（ShadCN UI）

```tsx
import { Button } from '@/components/ui/button';

type CustomButtonProps = {
  children: React.ReactNode;
  onClick?: () => void;
  variant?: 'default' | 'outline' | 'destructive';
};

export const CustomButton = ({
  children,
  onClick,
  variant = 'default',
}: CustomButtonProps) => {
  return (
    <Button onClick={onClick} variant={variant}>
      {children}
    </Button>
  );
};
```

✅ 汎用化ポイント

- `variant` を渡して見た目を変更可能
- `children` でボタン内のラベルを柔軟に対応
- `onClick` は使う側に任せる

🎯 まとめ

- コンポーネントを分ける基準は「**汎用性**」と「**責務の分離**」
- mapの中で使うもの、1つの役割しかないものは積極的に分割
- 再利用する前提のものは汎用化して切り出す！