Node.js 非同期処理 段階的課題 3問

課題1：Promiseの基本

次の要件を満たすプログラムを作成してください。

- `wait.js`というファイルを作成
- `delay`という関数を定義し、`Promiseで2秒後に解決するようにする
- その関数を使って以下を実行：

```
delay(2000)
  .then(() => console.log('2秒経過しました！'))
  .catch(err => console.error(err));
```

ポイント: `new Promise`とresolve、その後の`.then()`の使い方

課題2：async/awaitの基本

次の要件を満たすプログラムを作成してください。

- `fetchData.js`というファイルを作成
- `getData`という非同期関数を定義し、1秒後にデータを返すようにする
- `main.js`で`getData`をインポートして、`await`を使って呼び出し、結果を出力
- 結果に「Done!」と表示

ポイント: `async`と`await`キーワード、非同期関数の基本的な使い方

課題3：複数の非同期処理の並列実行

次の要件を満たすプログラムを作成してください。

- `tasks.js`というファイルを作成
- 以下の3つの関数を定義：
  - `task1()`: 1秒後に'Task 1 complete'を返す
  - `task2()`: 2秒後に'Task 2 complete'を返す
  - `task3()`: 1.5秒後に'Task 3 complete'を返す
- `main.js`で以下を実行：
  - `Promise.all()`を使って3つのタスクを並列実行
  - すべてのタスクが完了したら、結果を配列で出力
  - エラーハンドリングも追加

ポイント: `Promise.all()`による複数のPromise処理

解き方のコツ

- 課題1: `new Promise((resolve, reject) => {...})`でPromiseを作成
- 課題2: `async function`でasync関数を定義し、その中で`await`を使用
- 課題3: タスク関数内で`setTimeoutをPromiseで包む`ことでPromiseを返す