Node.js 非同期処理 段階的課題 解答例

課題1：Promiseの基本

```
// wait.js
const delay = (ms) => new Promise(resolve => setTimeout(resolve, ms));

// または以下でもOK
// const delay = (ms) => {
//   return new Promise((resolve) => {
//     setTimeout(() => resolve(), ms);
//   });
// };

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

解説: `new Promise`でPromiseを作成し、`setTimeout`の後に`resolve`を呼び出す。`.then()`でPromiseが解決された後の処理を記述。

課題2：async/awaitの基本

```
// fetchData.js
const getData = async () => {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('Data from server');
    }, 1000);
  });
};

module.exports = getData;

// main.js
const getData = require('./fetchData');

(async () => {
  const result = await getData();
  console.log(result);
  console.log('Done!');
})();
```

解説: `async`関数内で`await`を使ってPromiseの解決を待つ。コードの可読性がPromiseのチェーンより優れている。

課題3：複数の非同期処理の並列実行

```
// tasks.js
const task1 = () => new Promise(resolve =>
  setTimeout(() => resolve('Task 1 complete'), 1000)
);

const task2 = () => new Promise(resolve =>
  setTimeout(() => resolve('Task 2 complete'), 2000)
);

const task3 = () => new Promise(resolve =>
  setTimeout(() => resolve('Task 3 complete'), 1500)
);

module.exports = { task1, task2, task3 };

// main.js
const { task1, task2, task3 } = require('./tasks');

Promise.all([task1(), task2(), task3()])
  .then(results => {
    console.log('All tasks completed:');
    console.log(results);
  })
  .catch(error => {
    console.error('An error occurred:', error);
  });

// または async/await で書く場合：
// (async () => {
//   try {
//     const results = await Promise.all([task1(), task2(), task3()]);
//     console.log('All tasks completed:');
//     console.log(results);
//   } catch (error) {
//     console.error('An error occurred:', error);
//   }
// })();
```

解説: `Promise.all()`は配列内のすべてのPromiseが解決されるまで待つ。1つでも失敗するとcatch()に制御が移る。実行時間は最も遅いタスク（task2の2秒）となる。

補足情報

Promise vs async/await

- **Promise**: チェーン的に `.then().then()` と書く
- **async/await**: より読みやすい、同期的なコードのように見える

よくある間違い

- `await`は`async`関数の中でしか使えない
- `Promise.all()`は**すべて**が成功する必要がある（1つ失敗で全て失敗）
- `setTimeout`をPromiseで包むことが重要