Node.jsモジュール 段階的課題 20問

レベル1：CommonJSの基本 (課題1-3)

課題1：モジュールのインポート

次の要件を満たすプログラムを作成してください。

- `math.js`というファイルを作成し、以下の関数をエクスポート：

```
exports.add = (a, b) => a + b;
```

- `main.js`というファイルを作成し、`math.js`から`add`関数を`require`でインポート
- インポートした関数を使って`add(5, 3)`の結果を出力

ポイント: requireの使い方とローカルモジュールのインポート

課題2：モジュールのエクスポート

次の要件を満たすプログラムを作成してください。

- `greet.js`というファイルを作成し、以下の関数を定義して`exports`でエクスポート：

```
exports.sayHello = (name) => `Hello, ${name}!`;
```

- `app.js`というファイルを作成し、`greet.js`をインポートして`sayHello('World')`の結果を出力

ポイント: exportsの使い方と関数エクスポート

課題3：複数のエクスポート

次の要件を満たすプログラムを作成してください。

- `utils.js`というファイルを作成し、以下の関数と変数を`exports`でエクスポート：

```
exports.PI = 3.14159;
exports.sum = (arr) => arr.reduce((a, b) => a + b, 0);
exports.average = (arr) => exports.sum(arr) / arr.length;
```

- `test.js`というファイルを作成し、`utils.js`からこれらをインポートして使用：
  - PIを出力
  - sum([1, 2, 3, 4])の結果を出力

- average([1, 2, 3, 4])の結果を出力

ポイント: 複数のプロパティのエクスポート

レベル2：ES Moduleの基本 (課題4-6)

課題4：ES Moduleのインポート

次の要件を満たすプログラムを作成してください。

- `math.mjs`というファイルを作成し、以下の関数をエクスポート：

```
export const multiply = (a, b) => a * b;
```

- `main.mjs`というファイルを作成し、`math.mjs`から`multiply`をインポート
- インポートした関数を使って`multiply(4, 5)`の結果を出力

ポイント: ES Moduleのimport構文（.mjs拡張子に注意）

課題5：ES Moduleのエクスポート

次の要件を満たすプログラムを作成してください。

- `logger.mjs`というファイルを作成し、以下の関数を`export`でエクスポート：

```
export const log = (message) => console.log(`[LOG] ${message}`);
```

- `app.mjs`というファイルを作成し、`logger.mjs`から`log`をインポートして`log('Hello ES Module')`を実行

ポイント: exportキーワードの使い方

課題6：デフォルトエクスポート

次の要件を満たすプログラムを作成してください。

- `config.mjs`というファイルを作成し、以下のオブジェクトを`export default`でエクスポート：

```
const config = {
  port: 3000,
  host: 'localhost'
};
export default config;
```

- `server.mjs`というファイルを作成し、`config.mjs`からデフォルトエクスポートをインポートして`config.port`を出力

ポイント: export defaultの使い方

レベル3：npmの基本 (課題7-9)

課題7：npm init

次の手順を実行してください。

- ターミナルで`npm init -y`コマンドを実行して`package.json`を作成
- 作成された`package.json`ファイルを開いて内容を確認
- `name`、`version`、`main`フィールドの値を確認して出力

ポイント: npm initコマンドとpackage.jsonの基本構造

課題8：パッケージのインストール

次の手順を実行してください。

- `npm install lodash`コマンドでlodashパッケージをインストール
- `program.js`というファイルを作成し、lodashを使って以下の処理を実行：

```
const _ = require('lodash');
const array = [1, 2, 3, 4, 5, 6];
console.log(_.chunk(array, 2)); // [[1,2], [3,4], [5,6]]
```

- プログラムを実行して結果を確認

ポイント: npm installとサードパーティパッケージの使用

課題9：package.jsonの編集

次の手順を実行してください。

- `package.json`を編集して`scripts`フィールドに以下を追加：

```
"scripts": {
  "start": "node index.js",
  "dev": "node --watch index.js"
}
```

- `index.js`ファイルを作成し、`console.log('Hello from npm script!');`と記述
- `npm start`コマンドを実行してスクリプトが動作することを確認

ポイント: package.jsonのscriptsフィールド

レベル4：モジュールの高度な使い方 (課題10-12)

課題10：module.exports

次の要件を満たすプログラムを作成してください。

- `library.js`というファイルを作成し、`module.exports`を使って以下のオブジェクトをエクスポート：

```
module.exports = {
  add: (a, b) => a + b,
  subtract: (a, b) => a - b,
  multiply: (a, b) => a * b,
  divide: (a, b) => a / b
};
```

- `calc.js`というファイルを作成し、`library.js`をインポートして各関数をテスト（例: add(10, 5)など）

ポイント: module.exports vs exportsの違い

課題11：条件付きエクスポート

次の要件を満たすプログラムを作成してください。

- `env.js`というファイルを作成し、環境変数に応じて異なる設定をエクスポート：

```
if (process.env.NODE_ENV === 'production') {
  module.exports = { apiUrl: 'https://api.prod.com', debug: false };
} else {
  module.exports = { apiUrl: 'http://localhost:3000', debug: true };
}
```

- `app.js`というファイルを作成し、`env.js`をインポートして設定を出力
- 環境変数を設定して動作を確認：`NODE_ENV=production node app.js`

ポイント: 動的エクスポートと環境変数の使用

課題12：サードパーティモジュールの使用

次の要件を満たすプログラムを作成してください。

- `npm install axios`でaxiosをインストール

- `api.js`というファイルを作成し、axiosを使ってGitHub APIからユーザー情報を取得：

```
const axios = require('axios');
axios.get('https://api.github.com/users/octocat')
  .then(response => {
    console.log('User:', response.data.login);
    console.log('Followers:', response.data.followers);
  })
  .catch(error => console.error('Error:', error.message));
```

- プログラムを実行して結果を確認

ポイント: HTTPリクエストを行うサードパーティモジュール

レベル5：ファイルシステムモジュール (課題13-15)

課題13：ファイルの読み込み

次の要件を満たすプログラムを作成してください。

- `sample.txt`というファイルを作成し、以下の内容を記述：

```
Hello, Node.js!
This is a sample text file.
```

- `read.js`というファイルを作成し、`fs.readFile`を使って`sample.txt`を読み込み、内容を出力

ポイント: fs.readFileの非同期ファイル読み込み

課題14：ファイルの書き込み

次の要件を満たすプログラムを作成してください。

- `write.js`というファイルを作成し、`fs.writeFile`を使って`output.txt`に以下の内容を書き込み：

```
This file was created by Node.js fs module.
Current time: ${new Date().toISOString()}
```

- 書き込み後にファイルが存在することを確認

ポイント: fs.writeFileのファイル書き込み

課題15：ディレクトリの操作

次の要件を満たすプログラムを作成してください。

- `dir.js`というファイルを作成し、`fs.mkdir`を使って`my-folder`というディレクトリを作成
- 作成後に`fs.readdir`を使って現在のディレクトリの内容をリスト表示

ポイント: fs.mkdirとfs.readdirのディレクトリ操作

レベル6：カスタムモジュールの作成 (課題16-18)

課題16：計算機モジュール

次の要件を満たすプログラムを作成してください。

- `calculator.js`というファイルを作成し、以下の関数をエクスポート：
  - `add(a, b)`: 加算
  - `subtract(a, b)`: 減算
  - `multiply(a, b)`: 乗算
  - `divide(a, b)`: 除算（0除算チェック付き）
- `useCalc.js`というファイルを作成し、各関数をテストして結果を出力

ポイント: 計算機モジュールの設計とエラー処理

課題17：ユーティリティモジュール

次の要件を満たすプログラムを作成してください。

- `stringUtils.js`というファイルを作成し、以下の関数をエクスポート：
  - `capitalize(str)`: 文字列の先頭を大文字に
  - `reverse(str)`: 文字列を逆順に
  - `isPalindrome(str)`: 回文かどうかを判定
- `testUtils.js`というファイルを作成し、各関数をテスト

ポイント: 文字列操作ユーティリティの作成

課題18：設定モジュール

次の要件を満たすプログラムを作成してください。

- `settings.mjs`というファイルを作成し、以下の設定をES Moduleでエクスポート：

```
export const database = {
  host: 'localhost',
  port: 5432,
  name: 'myapp'
};
export const server = {
```

```
  port: 3000,
  timeout: 5000
};
```

- `app.mjs`というファイルを作成し、これらの設定をインポートして使用

ポイント: 設定管理モジュールの作成

レベル7：モジュールのテストとデバッグ (課題19-20)

課題19：モジュールのテスト

次の要件を満たすプログラムを作成してください。

- 課題16で作成した`calculator.js`をインポート
- `testCalc.js`というファイルを作成し、以下のテストを実行：
  - add(2, 3) === 5
  - subtract(10, 4) === 6
  - multiply(3, 7) === 21
  - divide(15, 3) === 5
  - divide(10, 0) がエラーを投げることを確認
- 各テスト結果をconsole.logで出力

ポイント: モジュールの単体テスト

課題20：総合課題

次の要件を満たすプログラムを作成してください。

- `npm install moment`でmoment.jsをインストール
- `app.js`というファイルを作成し、以下の処理を実行：
  - fsモジュールで`data.txt`を作成し、日付情報を書き込み
  - moment.jsを使って現在の日付をフォーマット
  - カスタムモジュール（例: calculator.js）を使って計算
  - 結果をコンソールとファイルに出力
- プログラムを実行して総合的な動作を確認

ポイント: 複数のモジュールとnpmパッケージの統合

補足情報

学習の進め方

1. 各課題を順番に取り組んでください
2. エラーが出たら、エラーメッセージをよく読んで原因を考えてください
3. わからない場合は、該当するサンプルコード（3.nodejs_modules内）を参照してください

4. 余裕があれば、課題をアレンジして自分なりの機能を追加してみてください

参考：各課題と対応するサンプルコードの関連

- 課題1-3: 3.1.1/, 3.1.2/
- 課題4-6: 3.2.2/
- 課題7-9: 3.4/, 3.5/
- 課題10-12: 3.1.1/
- 課題13-15: 3.0.0/
- 課題16-18: 3.1.1/, 3.1.2/, 3.2.2/
- 課題19-20: 総合 c:\Users\user\nodejs\3.nodejs_modules\exercise\exercice.md