Node.js構文 段階的課題 解答例

レベル1：変数とスコープ

課題1：変数宣言の基本

```
const myName = '佐々木博幸';
let age = 25;
console.log('名前:', myName);
console.log('年齢:', age);

age = age + 1;
// または age++;
console.log('1年後の年齢:', age);
```

解説: constは再代入不可、letは再代入可能。myNameをconstで宣言したため、後から変更できない。

課題2：スコープの理解

```
if (true) {
    var x = 10;
    let y = 20;
    const z = 30;
}
console.log(x);  // 10 が出力される
// console.log(y);  // ReferenceError: y is not defined
// console.log(z);  // ReferenceError: z is not defined
```

解説:

- `var` は関数スコープなので、ifブロックの外からもアクセス可能
- `let` と `const` はブロックスコープなので、ifブロックの外からアクセスできない

課題3：変数の再代入

```
let score = 0;
console.log('初期値:', score);

score = score + 10;
console.log('10加算後:', score);

score = score * 5;
console.log('5倍後:', score);

// または以下のように書くこともできる
// score += 10;
// score *= 5;
```

出力結果:

```
初期値: 0
10加算後: 10
5倍後: 50
```

レベル2：演算子とデータ型

課題4：計算プログラム

```
const num1 = 7;
const num2 = 3;

const add = num1 + num2;
const sub = num1 - num2;
const mul = num1 * num2;
const div = num1 / num2;

console.log('加算:', add, '10より大きい:', add > 10);
console.log('減算:', sub, '10より大きい:', sub > 10);
console.log('乗算:', mul, '10より大きい:', mul > 10);
console.log('除算:', div, '10より大きい:', div > 10);
```

出力結果:

```
加算: 10 10より大きい: false
減算: 4 10より大きい: false
乗算: 21 10より大きい: true
除算: 2.333... 10より大きい: false
```

課題5：型の確認

```
console.log(typeof "JavaScript");        // string
console.log(typeof 42);                  // number
console.log(typeof true);                // boolean
console.log(typeof null);                // object（これは言語の仕様上のバグ）
console.log(typeof undefined);           // undefined
console.log(typeof []);                  // object
console.log(typeof {name: "Taro"});      // object
```

解説:

- `null` のtypeofが"object"になるのはJavaScriptの古いバグだが、互換性のため修正されていない
- 配列もオブジェクトとして扱われる

課題6：等価演算子の違い

```
const num = 10;
const str = "10";

console.log(num == str);   // true
console.log(num === str);  // false
```

解説:

- `==` は型変換を行ってから比較（"10"が数値10に変換される）
- `===` は型も含めて厳密に比較（型が違うのでfalse）
- 基本的には `===` を使うことが推奨される

レベル3：文字列とテンプレートリテラル

課題7：自己紹介プログラム

```
const name = '田中太郎';
const age = 25;
const hobby = 'プログラミング';

console.log(`私の名前は${name}です。年齢は${age}歳で、趣味は${hobby}です。`);

// 従来の方法（参考）
// console.log('私の名前は' + name + 'です。年齢は' + age + '歳で、趣味は' + hobby + 'です。')
```

出力結果:

```
私の名前は田中太郎です。年齢は25歳で、趣味はプログラミングです。
```

課題8：複数行の文字列

```
const postalCode = '060-0001';
const prefecture = '北海道';
const city = '札幌市中央区';

// 方法1: \nを使う
const address1 = `〒${postalCode}\n${prefecture}\n${city}`;
console.log(address1);

console.log('---');

// 方法2: テンプレートリテラル内で改行
const address2 = `〒${postalCode}
${prefecture}
${city}`;
console.log(address2);
```

出力結果:

〒060-0001
北海道
札幌市中央区

レベル4：オブジェクトの基本

課題9：学生オブジェクト

```
const student = {
    name: '山田花子',
    age: 18,
    grade: 3,
    club: 'サッカー部'
};

// ドット記法
const studentName = student.name;
console.log('名前:', studentName);

// ブラケット記法
const studentGrade = student['grade'];
console.log('学年:', studentGrade);
```

出力結果:

```
名前: 山田花子
学年: 3
```

課題10：ネストしたオブジェクト

```
const company = {
    name: '株式会社サンプル',
    address: {
        prefecture: '北海道',
        city: '札幌市'
    },
    employees: 150
};

console.log('市区町村:', company.address.city);
// またはブラケット記法で
console.log('市区町村:', company['address']['city']);
```

出力結果:

```
市区町村: 札幌市
```

課題11：オブジェクトのプロパティ省略記法

```
const title = 'JavaScript入門';
const author = '山田太郎';
const year = 2024;

// プロパティ省略記法
const book = {
    title,
    author,
    year
};

console.log(book);

// 従来の方法（参考）
// const book = {
//     title: title,
//     author: author,
//     year: year
// };
```

出力結果:

```
{ title: 'JavaScript入門', author: '山田太郎', year: 2024 }
```

レベル5：配列の操作

課題12：配列の基本操作

```
const numbers = [10, 20, 30, 40, 50];

console.log('配列の長さ:', numbers.length);
console.log('最初の要素:', numbers[0]);
console.log('最後の要素:', numbers[numbers.length - 1]);

console.log('全要素:');
for (let i = 0; i < numbers.length; i++) {
    console.log(`  ${i}番目: ${numbers[i]}`);
}
```

出力結果:

```
配列の長さ: 5
最初の要素: 10
最後の要素: 50
全要素:
  0番目: 10
  1番目: 20
  2番目: 30
  3番目: 40
  4番目: 50
```

課題13：配列のmap

```
const prices = [100, 200, 300, 400, 500];

const pricesWithTax = prices.map(function(price) {
    return price * 1.1;
});

console.log('元の価格:', prices);
console.log('税込価格:', pricesWithTax);

// アロー関数を使った書き方（参考）
// const pricesWithTax = prices.map(price => price * 1.1);
```

出力結果:

```
元の価格: [ 100, 200, 300, 400, 500 ]
税込価格: [ 110, 220, 330, 440, 550 ]
```

課題14：配列のfilter

```
const scores = [45, 78, 92, 35, 88, 60, 95, 42];

const passScores = scores.filter(function(score) {
    return score >= 60;
});

console.log('全ての点数:', scores);
console.log('合格点(60点以上):', passScores);

// アロー関数を使った書き方（参考）
// const passScores = scores.filter(score => score >= 60);
```

出力結果:

```
全ての点数: [ 45, 78, 92, 35, 88, 60, 95, 42 ]
合格点(60点以上): [ 78, 92, 88, 60, 95 ]
```

レベル6：関数

課題15：関数の基本

```
function calculateArea(width, height) {
    return width * height;
}

const area1 = calculateArea(5, 10);
console.log('5 × 10 の面積:', area1);

const area2 = calculateArea(8, 3);
console.log('8 × 3 の面積:', area2);
```

出力結果:

```
5 × 10 の面積: 50
8 × 3 の面積: 24
```

課題16：アロー関数

```
// 長方形の面積をアロー関数で
const calculateArea = (width, height) => {
    return width * height;
};

const area = calculateArea(5, 10);
console.log('面積:', area);

// 簡略版（returnを省略）
const calculateArea2 = (width, height) => width * height;
console.log('面積2:', calculateArea2(5, 10));

// 引数が1つで2倍にする関数（最も省略した形）
const double = x => x * 2;
console.log('10の2倍:', double(10));
```

出力結果:

```
面積: 50
面積2: 50
10の2倍: 20
```

課題17：デフォルト引数

```
function greet(name = 'Guest') {
    return `Hello, ${name}!`;
}

// 引数あり
const message1 = greet('田中');
console.log(message1);

// 引数なし（デフォルト値が使われる）
const message2 = greet();
console.log(message2);
```

出力結果:

```
Hello, 田中!
Hello, Guest!
```

レベル7：高度な配列操作とスプレッド構文

課題18：スプレッド構文で配列を結合

```
const fruits1 = ['apple', 'banana'];
const fruits2 = ['orange', 'grape'];

const allFruits = [...fruits1, ...fruits2, 'melon'];

console.log('fruits1:', fruits1);
console.log('fruits2:', fruits2);
console.log('結合した配列:', allFruits);
```

出力結果:

```
fruits1: [ 'apple', 'banana' ]
fruits2: [ 'orange', 'grape' ]
結合した配列: [ 'apple', 'banana', 'orange', 'grape', 'melon' ]
```

課題19：分割代入

```
const userInfo = ['田中太郎', 25, 'Tokyo', 'developer'];

const [name, age, city, job] = userInfo;

console.log('名前:', name);
console.log('年齢:', age);
console.log('都市:', city);
console.log('職業:', job);
```

出力結果:

```
名前: 田中太郎
年齢: 25
都市: Tokyo
職業: developer
```

レベル8：総合課題

課題20：学生管理システム

```
// 1. 学生データの作成
const students = [
    { name: '佐藤一郎', age: 18, score: 85 },
    { name: '鈴木花子', age: 19, score: 45 },
    { name: '田中太郎', age: 18, score: 92 },
    { name: '山田次郎', age: 20, score: 55 },
    { name: '伊藤美咲', age: 19, score: 78 }
];

// 2-1. 60点以上の学生を返す関数
function getPassStudents(studentList) {
    return studentList.filter(student => student.score >= 60);
}

// 2-2. 平均点を返す関数
function getAverageScore(studentList) {
    const total = studentList.reduce((sum, student) => sum + student.score, 0);
    return total / studentList.length;
}

// 2-3. 学生情報を整形する関数
function formatStudentInfo(student) {
    return `名前: ${student.name}, 年齢: ${student.age}歳, 点数: ${student.score}点`;
}

// 3. 実行
console.log('=== 全学生 ===');
students.forEach(student => {
    console.log(formatStudentInfo(student));
});

console.log('\n=== 合格者（60点以上）===');
const passStudents = getPassStudents(students);
passStudents.forEach(student => {
    console.log(formatStudentInfo(student));
});

console.log('\n=== 統計情報 ===');
```

```
const average = getAverageScore(students);
console.log(`全体の平均点: ${average.toFixed(1)}点`);
console.log(`合格者数: ${passStudents.length}人 / 全体: ${students.length}人`);
```

出力結果:

```
=== 全学生 ===
名前: 佐藤一郎, 年齢: 18歳, 点数: 85点
名前: 鈴木花子, 年齢: 19歳, 点数: 45点
名前: 田中太郎, 年齢: 18歳, 点数: 92点
名前: 山田次郎, 年齢: 20歳, 点数: 55点
名前: 伊藤美咲, 年齢: 19歳, 点数: 78点

=== 合格者（60点以上）===
名前: 佐藤一郎, 年齢: 18歳, 点数: 85点
名前: 田中太郎, 年齢: 18歳, 点数: 92点
名前: 伊藤美咲, 年齢: 19歳, 点数: 78点

=== 統計情報 ===
全体の平均点: 71.0点
合格者数: 3人 / 全体: 5人
```

解説:

- `filter`：条件に合う要素だけを抽出
- `reduce`：配列の要素を集計（合計を計算）
- `forEach`：配列の各要素に対して処理を実行
- `toFixed(1)`：小数点第1位まで表示

応用例（課題20の別解）

アロー関数を使った簡潔な書き方

```
const students = [
    { name: '佐藤一郎', age: 18, score: 85 },
    { name: '鈴木花子', age: 19, score: 45 },
    { name: '田中太郎', age: 18, score: 92 },
    { name: '山田次郎', age: 20, score: 55 },
    { name: '伊藤美咲', age: 19, score: 78 }
];

// 関数をアロー関数で定義
const getPassStudents = studentList => studentList.filter(s => s.score >= 60);

const getAverageScore = studentList =>
    studentList.reduce((sum, s) => sum + s.score, 0) / studentList.length;

const formatStudentInfo = s =>
    `名前: ${s.name}, 年齢: ${s.age}歳, 点数: ${s.score}点`;

// 実行
console.log('=== 合格者リスト ===');
getPassStudents(students).forEach(s => console.log(formatStudentInfo(s)));

console.log('\n=== 統計 ===');
console.log(`平均点: ${getAverageScore(students).toFixed(1)}点`);
```

学習のポイント

よく使う配列メソッド

- `map()`：配列の各要素を変換して新しい配列を作る
- `filter()`：条件に合う要素だけを抽出
- `reduce()`：配列を集約して1つの値にする
- `forEach()`：各要素に対して処理を実行（戻り値なし）

関数の書き方

```
// 通常の関数
function add(a, b) {
    return a + b;
}

// アロー関数（完全版）
const add = (a, b) => {
    return a + b;
};

// アロー関数（省略版）
const add = (a, b) => a + b;

// 引数が1つの場合はカッコも省略可能
const double = x => x * 2;
```

オブジェクトと配列の分割代入

```
// 配列
const [a, b, c] = [1, 2, 3];

// オブジェクト
const { name, age } = { name: 'Taro', age: 20 };

// 配列の一部をスキップ
const [first, , third] = [1, 2, 3];

// 残りをまとめて取得
const [head, ...tail] = [1, 2, 3, 4, 5];
```