CODE EXERCISE

JAVASCRIPT ES6+ features

CODING EXERCISES TEST YOUR SKILLS

Introduction:

These exercises cover a range of ES6+ features, providing both practical examples and conceptual understanding. They can be used as a teaching resource or for self-study to reinforce learning of modern JavaScript.

Comprehensive set of 10 JavaScript exercises, specifically tailored to explore and master the 10 JavaScript exercises. This is a perfect resource for both beginners looking to get a grip on modern JavaScript and seasoned devs aiming to refresh their knowledge. 🌟

Each exercise is meticulously crafted with:

- A clear problem statement 🧩
- Helpful hints and concept explanations 📚
- Detailed solutions with code 🖥️

From understanding let and const, to diving into arrow functions, template literals, destructuring, and much more, these exercises are designed to give you a hands-on experience with real-world JavaScript coding scenarios. 🌐

Here's a glimpse of the topics covered:

- Let and Const Variables
- Arrow Functions
- Template Literals
- Destructuring Objects and Arrays
- Spread Operator and Rest Parameters
- Default Parameters
- Classes and Inheritance
- Promises and Async/Await

Whether you're looking to crack your next coding interview or just want to level up your JavaScript skills, these exercises are the perfect starting point. 💪

Exercise 1: Let and Const

Problem Statement:

Create two variables using let and const. The let variable should be named age and set to 30. The const variable should be named name and set to "Alice". Try to reassign the name variable and observe what happens.

Hint/Explanation:

ES6 introduced let and const for declaring variables. let is used for variables that can change, while const is for variables that should remain constant.

Solution:

```
let age = 30;
const name = "Alice";

// Uncomment the line below to see the error
// name = "Bob"; // This will throw an error
```

Exercise 2: Arrow Functions

Problem Statement:

Convert the following function into an arrow function.

```
function add(a, b) {
   return a + b;
```

}

Hint/Explanation:

Arrow functions provide a concise syntax and lexically bind the this value.

Solution:

const add = (a, b) => a + b;

Exercise 3: Template Literals

Problem Statement:

Use a template literal to print "Hello, Alice! Your age is 30." using the variables name and age.

Hint/Explanation:

Template literals allow embedded expressions and multi-line strings.

Solution:

console.log(`Hello, ${name}! Your age is ${age}.`);

Exercise 4: Destructuring Objects

Problem Statement:

Given an object { firstName: "Alice", lastName: "Johnson" }, use object destructuring to extract firstName and lastName into variables.

Hint/Explanation:

Destructuring allows you to unpack values from arrays or properties from objects.

Solution:

```
const person = { firstName: "Alice", lastName: "Johnson" };
const { firstName, lastName } = person;
```

Exercise 5: Destructuring Arrays

Problem Statement:

Given an array [1, 2, 3, 4, 5], use array destructuring to create variables first and second for the first two elements.

Hint/Explanation:

Array destructuring works similarly to object destructuring but with array elements.

Solution:

```
const numbers = [1, 2, 3, 4, 5];
const [first, second] = numbers;
```

Exercise 6: Spread Operator

Problem Statement:

Combine two arrays [1, 2, 3] and [4, 5, 6] into a new array using the spread operator.

Hint/Explanation:

The spread operator allows an iterable (like an array) to be expanded in places where zero or more arguments or elements are expected.

Solution:

```
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const combined = [...arr1, ...arr2];
```

Exercise 7: Rest Parameters

Problem Statement:

Write a function that takes multiple arguments and returns their sum using rest parameters.

Hint/Explanation:

Rest parameters allow us to represent an indefinite number of arguments as an array.

Solution:

```
const sum = (...numbers) => numbers.reduce((acc, current) => acc + current, 0);
```

Exercise 8: Default Parameters

Problem Statement:

Create a function greet(name, greeting = "Hello") that greets a person. If no greeting is provided, it should default to "Hello".

Hint/Explanation:

Default function parameters allow named parameters to be initialized with default values if no value or undefined is passed.

Solution:

```
const greet = (name, greeting = "Hello") => `${greeting}, ${name}!`;
```

Exercise 9: Classes and Inheritance

Problem Statement:

Create a class Animal with a constructor setting the name property. Then, create a subclass Dog that extends Animal and adds a bark method.

Hint/Explanation:

ES6 classes are syntactical sugar over JavaScript's existing prototype-based inheritance. The extends keyword is used for class inheritance.

Solution:

```
class Animal {
   constructor(name) {
      this.name = name;
   }
```

```
}

class Dog extends Animal {
   bark() {
      return `Woof! My name is ${this.name}`;
   }
}
```

Exercise 10: Promises and Async/Await

Problem Statement:

Create a function waitAndReturn that returns a Promise which resolves with the string "Done" after 2 seconds. Then, use async/await to call this function and log the result.

Hint/Explanation:

Promises are used for asynchronous computations. Async/await is syntactic sugar for working with Promises in a more synchronous fashion.

Solution:

```
const waitAndReturn = () => new Promise(resolve => setTimeout(() =>
resolve("Done"), 2000));
async function run() {
   const result = await waitAndReturn();
   console.log(result);
```

```
}
run();
```

Feel free to reach out if you have any questions or need further explanations on any of the exercises. Let's code and learn together! 🤝

#JavaScript #ES6 #WebDevelopment #Coding #Programming #LearningToCode #FrontendDevelopment #WebDesign #SoftwareEngineering #CodeNewbies #100DaysOfCode #Developers