JavaScript Case Studies

Basic Syntax and Structure

Problem:

Create a simple JavaScript program that prints 'Hello, World!' to the console.

Solution:

```
console.log('Hello, World!');
```

Variables and Data Types

Problem:

Declare a variable for storing a user's name and print it to the console.

Solution:

```
let userName = 'John Doe';
console.log(userName);
```

Operators

Problem:

Write a program that calculates the area of a rectangle given its width and height.

Solution:

```
let width = 5;
let height = 10;
let area = width * height;
console.log('Area of the rectangle: ' + area);
```

Control Structures

Problem:

Write a program that prints 'Even' if a number is even and 'Odd' if a number is odd.

Solution:

```
let number = 7;
if (number % 2 === 0) {
   console.log('Even');
```

```
} else {
  console.log('Odd');
}
```

Functions

Problem:

Create a function that takes two numbers as arguments and returns their sum.

Solution:

```
function add(a, b) {
  return a + b;
}
console.log(add(3, 4)); // Output: 7
```

Objects and Arrays

Problem:

Create an object to store information about a book and print the book's title.

Solution:

```
let book = {
  title: 'JavaScript: The Good Parts',
  author: 'Douglas Crockford',
  year: 2008
};
console.log(book.title);
```

ES6 and Beyond

Problem:

Use destructuring to extract and print values from an array.

Solution:

```
let [firstName, lastName] = ['John', 'Doe'];
console.log(firstName); // Output: John
console.log(lastName);  // Output: Doe
```

DOM Manipulation

Problem:

Change the text of a paragraph with the id 'myParagraph' to 'Hello, World!'.

Solution:

```
document.getElementById('myParagraph').innerText = 'Hello, World!';
```

Asynchronous JavaScript

Problem:

Create a function that fetches data from an API and logs the response to the console using async/await.

Solution:

```
async function fetchData() {
  let response = await fetch('https://api.example.com/data');
  let data = await response.json();
  console.log(data);
}
fetchData();
```

Error Handling

Problem:

Write a program that tries to parse a JSON string and handles any errors that occur.

Solution:

```
let jsonString = '{"name":"John", "age":30}';
try {
  let user = JSON.parse(jsonString);
  console.log(user);
} catch (error) {
  console.error('Invalid JSON string');
}
```

Modules

Problem:

Create a module that exports a function to calculate the square of a number and import it in another file.

Solution:

```
export function square(x) {
  return x * x;
}

import { square } from './square.js';
console.log(square(5)); // Output: 25
```

APIs and AJAX

Problem:

Make a GET request to an API and display the data on a webpage.

Solution:

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => {
    document.getElementById('dataContainer').innerText = JSON.stringify(data);
  })
  .catch(error => console.error('Error:', error));
```

JavaScript Design Patterns

Problem:

Implement the Singleton pattern in JavaScript.

Solution:

```
let Singleton = (function() {
  let instance;

  function createInstance() {
    let object = new Object('I am the instance');
    return object;
  }

  return {
    getInstance: function() {
      if (!instance) {
        instance = createInstance();
      }
      return instance;
    }
```

```
  };
})();

let instance1 = Singleton.getInstance();
let instance2 = Singleton.getInstance();

console.log(instance1 === instance2); // Output: true
```

Testing and Debugging

Problem:

Write a unit test for a function that multiplies two numbers using Jest.

Solution:

```
function multiply(a, b) {
  return a * b;
}
module.exports = multiply;

const multiply = require('./multiply');
test('multiplies 3 and 4 to equal 12', () => {
  expect(multiply(3, 4)).toBe(12);
});
```

Frameworks and Libraries

Problem:

Create a simple React component that displays 'Hello, World!'.

Solution:

```
import React from 'react';
import ReactDOM from 'react-dom';

function HelloWorld() {
  return <h1>Hello, World!</h1>;
}

ReactDOM.render(<HelloWorld />, document.getElementById('root'));
```

Node.js and Backend Development

Problem:

Create a simple Node.js server that responds with 'Hello, World!'.

Solution:

```
const http = require('http');

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, World!\n');
});

server.listen(3000, '127.0.0.1', () => {
  console.log('Server running at http://127.0.0.1:3000/');
});
```

TypeScript

Problem:

Create a TypeScript function that takes a string and returns its length.

Solution:

```
function getStringLength(str: string): number {
  return str.length;
}
console.log(getStringLength('Hello, TypeScript!')); // Output: 17
```

Web Performance Optimization

Problem:

Optimize a function to reduce its execution time using memoization.

Solution:

```
function memoize(fn) {
  let cache = {};
  return function(...args) {
    let key = JSON.stringify(args);
    if (cache[key]) {
      return cache[key];
    } else {
```

```
        let result = fn(...args);
        cache[key] = result;
        return result;
      }
    };
}

function slowFunction(num) {
  for (let i = 0; i < 1000000000; i++) {}
  return num * 2;
}

let fastFunction = memoize(slowFunction);
console.log(fastFunction(5)); // Faster on subsequent calls
```

Working with JSON

Problem:

Convert a JavaScript object to a JSON string and parse it back to an object.

Solution:

```
let user = {
  name: 'John',
  age: 30
};
let jsonString = JSON.stringify(user);
console.log(jsonString);

let parsedUser = JSON.parse(jsonString);
console.log(parsedUser);
```

Event Handling

Problem:

Create a button that changes its text when clicked.

Solution:

```
<button id='myButton'>Click me</button>
<script>
  document.getElementById('myButton').addEventListener('click', function() {
    this.innerText = 'Clicked!';
```

```
    });
</script>
```