Final Assessment - Basic Javascript

October 2, 2024NEW

1.
CC-06 | NazarAF**OP** — Today at 1:36 PM

React to Post
Follow

2.
CC-06 | NazarAF**OP** — Today at 1:37 PM

Writing Comments

```
// Username : nazar_af_bangkit
/*
 Goal tahun ini:
 1. Belajar JavaScript.
 2. Menjadi Front-End atau Back-End Developer.
*/
```

Code Style

```
const books = {};

function getBooks() {
  return books;
}

function getBookById(id) {
  const book = books[id];

  if (!book) return null;

  return book.id;
}

function saveBook(book) {
  books[book.id] = book;
}

saveBook({ id: "book-1", name: "Book 1" });
const myBooks = getBooks();
const myBook = getBookById("book-1");

console.log(myBooks);
console.log(myBook);
```

Writing Test

```
import { sum } from "./index.js";
import { strict as assert } from "node:assert";
import test from "node:test";

test("sum should return the sum of two numbers", t => {
  // Test case 1: Sum of positive numbers
  assert.equal(sum(2, 3), 5, "2 + 3 should equal 5");

  // Test case 2: Sum of negative numbers
  assert.equal(sum(-2, -3), -5, "-2 + -3 should equal -5");

  // Test case 3: Sum of positive and negative number
  assert.equal(sum(2, -3), -1, "2 + (-3) should equal -1");
  assert.equal(sum(-2, 3), 1, "(-2) + 3 should equal 1");

  // Test case 4: Sum with zero
  assert.equal(sum(0, 5), 5, "0 + 5 should equal 5");
  assert.equal(sum(5, 0), 5, "5 + 0 should equal 5");

  // Test case 5: Sum of number and string number
  assert.equal(sum(1, "2"), "12", "1 + '2' should equal '12'");
  assert.equal(sum("1", 2), "12", "'1' + 2 should equal '12'");

  // Test case 6: Sum with string number
  assert.equal(sum("1", "2"), "12", "1 + '2' should equal '12'");
});
```

OOP

```
// Inventory
class Inventory {
  constructor() {
    this.items = [];
  }

  // Method for add item
  addItem(item) {
    this.items.push(item);
  }

  // Method for remove item
  removeItem(id) {
    this.items = this.items.filter(item => item.id !== id);
  }

  // Method for view item
  listItems() {
    return this.items.map(item => item.displayDetails()).join("\n");
  }
}

export default Inventory;

// Item
class Item {
  constructor(id, name, quantity, price) {
    this.id = id;
    this.name = name;
    this.quantity = quantity;
    this.price = price;
  }

  // Method for update detail item
  updateDetails(name, quantity, price) {
    this.name = name;
    this.quantity = quantity;
    this.price = price;
  }

  // Method for view detail
  displayDetails() {
    return `ID: ${this.id}, Name: ${this.name}, Quantity: ${this.quantity}, Price: ${this.price}`;
  }
}

export default Item;

// Main
import Item from "./Item.js";
import Inventory from "./Inventory.js";

const inventory = new Inventory();

const item1 = new Item(1, "Laptop", 10, 1000);
const item2 = new Item(2, "Mouse", 50, 20);

inventory.addItem(item1);
inventory.addItem(item2);

console.log("Initial Inventory:");
console.log(inventory.listItems());
/**
 * Output yang diharapkan:
 * Initial Inventory:
 * ID: 1, Name: Laptop, Quantity: 10, Price: 1000
 * ID: 2, Name: Mouse, Quantity: 50, Price: 20
 */

item1.updateDetails("Laptop", 8, 950);
console.log("\nInventory after update:");
console.log(inventory.listItems());

/**
 * Output yang diharapkan:
 * Inventory after update:
 * ID: 1, Name: Laptop, Quantity: 8, Price: 950
 * ID: 2, Name: Mouse, Quantity: 50, Price: 20
 */

inventory.removeItem(2);
console.log("\nInventory after removal:");
console.log(inventory.listItems());

/**
 * Output yang diharapkan:
```

```
 * Inventory after removal:
 * ID: 1, Name: Laptop, Quantity: 8, Price: 950
 */
```

Recursive

```
// Factorial
function factorial(n) {
  return n === 0 || n === 1 ? 1 : n * factorial(n - 1);
}

export default factorial;

// Fibonacci
function fibonacci(n) {
  return n === 0 ? 0 : n === 1 ? 1 : fibonacci(n - 1) + fibonacci(n - 2);
}

export default fibonacci;

// Main
import factorial from "./factorial.js";
import fibonacci from "./fibonacci.js";

const numberForFactorial = 5;
console.log(`Faktorial dari ${numberForFactorial} adalah ${factorial(numberForFactorial)}`);
/**
 * Output yang diharapkan:
 * Faktorial dari 5 adalah 120
 */

const numberForFibonacci = 10;
const fibArray = [];
console.log(`Deret Fibonacci hingga elemen ${numberForFibonacci} adalah:`);
for (let i = 0; i <= numberForFibonacci; i++) {
  // console.log(fibonacci(i));

  fibArray.push(fibonacci(i));
  console.log(`[ ${fibArray.join(", ")} ]`);
}

/**
 * Output yang diharapkan:
 * Deret Fibonacci hingga elemen  10 adalah:
 * [ 0 ]
 * [ 0, 1 ]
 * [ 0, 1, 1 ]
 * [ 0, 1, 1, 2 ]
 * [ 0, 1, 1, 2, 3 ]
 * [ 0, 1, 1, 2, 3, 5 ]
 * [ 0, 1, 1, 2, 3, 5, 8 ]
 * [ 0, 1, 1, 2, 3, 5, 8, 13]
 * [ 0, 1, 1, 2, 3, 5, 8, 13, 21]
 * [ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ]
 * [ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ]
 */
```

Full Coverage Test

```
import sum from "./index.js";
import { strict as assert } from "node:assert";
import test from "node:test";

test("sum should return the sum of two numbers", t => {
  // Test case 1: Sum of positive numbers
  assert.equal(sum(2, 3), 5, "2 + 3 should equal 5");

  // Test case 2: Sum of negative numbers
  assert.equal(sum(-2, -3), 0, "-2 + -3 should equal 0");

  // Test case 3: Sum of positive and negative number
  assert.equal(sum(2, -3), 0, "2 + (-3) should equal 0");
  assert.equal(sum(-2, 3), 0, "(-2) + 3 should equal 0");

  // Test case 4: Sum with zero
  assert.equal(sum(0, 5), 5, "0 + 5 should equal 5");
  assert.equal(sum(5, 0), 5, "5 + 0 should equal 5");

  // Test case 5: Sum of number and string number
  assert.equal(sum(1, "2"), 0, "1 + '2' should equal 0");
  assert.equal(sum("1", 2), 0, "'1' + 2 should equal 0");

  // Test case 6: Sum with string number
  assert.equal(sum("1", "2"), 0, "1 + '2' should equal 0");
});
```

Real World Scenario

```
function generateUniqueId() {
  return `_${Math.random().toString(36).slice(2, 9)}`;
}

// TODO: buatlah variabel yang menampung data orders
let orders = [];

// TODO: selesaikan fungsi addOrder
function addOrder(customerName, items) {
  const totalPrice = items.reduce((total, item) => total + item.price, 0);

  orders.push({
    id: generateUniqueId(),
    customerName,
    items,
    totalPrice,
    status: "Menunggu",
  });
}

// TODO: selesaikan fungsi updateOrderStatus
function updateOrderStatus(orderId, status) {
  const order = orders.find(order => order.id === orderId);
  if (order) order.status = status;
}

// TODO: selesaikan fungsi calculateTotalRevenue dari order yang berstatus Selesai
```

```
function calculateTotalRevenue() {
  return orders.filter(order => order.status === "Selesai").reduce((total, order) => total +
order.totalPrice, 0);
}

// TODO: selesaikan fungsi deleteOrder
function deleteOrder(id) {
  orders = orders.filter(order => order.id !== id);
}

export { orders, addOrder, updateOrderStatus, calculateTotalRevenue, deleteOrder };
```