Enhancing JavaScript Skills using JSFiddle

Estimated Time: 60 minutes

Introduction:

This lab is designed to provide you with practical exercises to enhance your JavaScript programming skills using JS Fiddle. The focus is on solving real-world logical problems, encouraging you to think and write efficient code and execute them in JS Fiddle to see the Output. By the end of this lab, you will have improved your ability to implement solutions for varied scenarios and gained confidence in your coding abilities.

Objective:

- Develop problem-solving skills using JavaScript
- Practice writing and debugging logical programs
- Understand how to implement real-world solutions using loops, functions, and conditional logic
- Strengthen coding practices on platforms like JSFiddle

Exercise 1: Calculate total sales amount

Problem:
You are working for an online store. Your task is to write a JavaScript code snippet that calculates the total sales amount for a given set of sales transactions.

Input details:

- An array of objects representing sales transactions. Each object has the following properties:
  - `item`: Name of the product (string)
  - `quantity`: Number of units sold (integer)
  - `price`: Price per unit (float)

Output details:

- A single number representing the total sales amount

Steps to implement:

1. Define an array of sales transactions with at least 3 sample objects
2. Write a function `calculateTotalSales` that takes this array as input
3. Use a loop to iterate through the array and calculate the total sales amount
4. Print the total sales amount to the console

▼ Click here to view hints

- The `sales` array contains objects, each representing a sales transaction with `item`, `quantity`, and `price` properties.
- Use a loop to go through each object in the `sales` array.
- For each object, multiply the `quantity` by the `price` to get the total for that item.
- Accumulate the totals in a variable to get the overall sales amount.
- Return the accumulated total and display it using `console.log`.

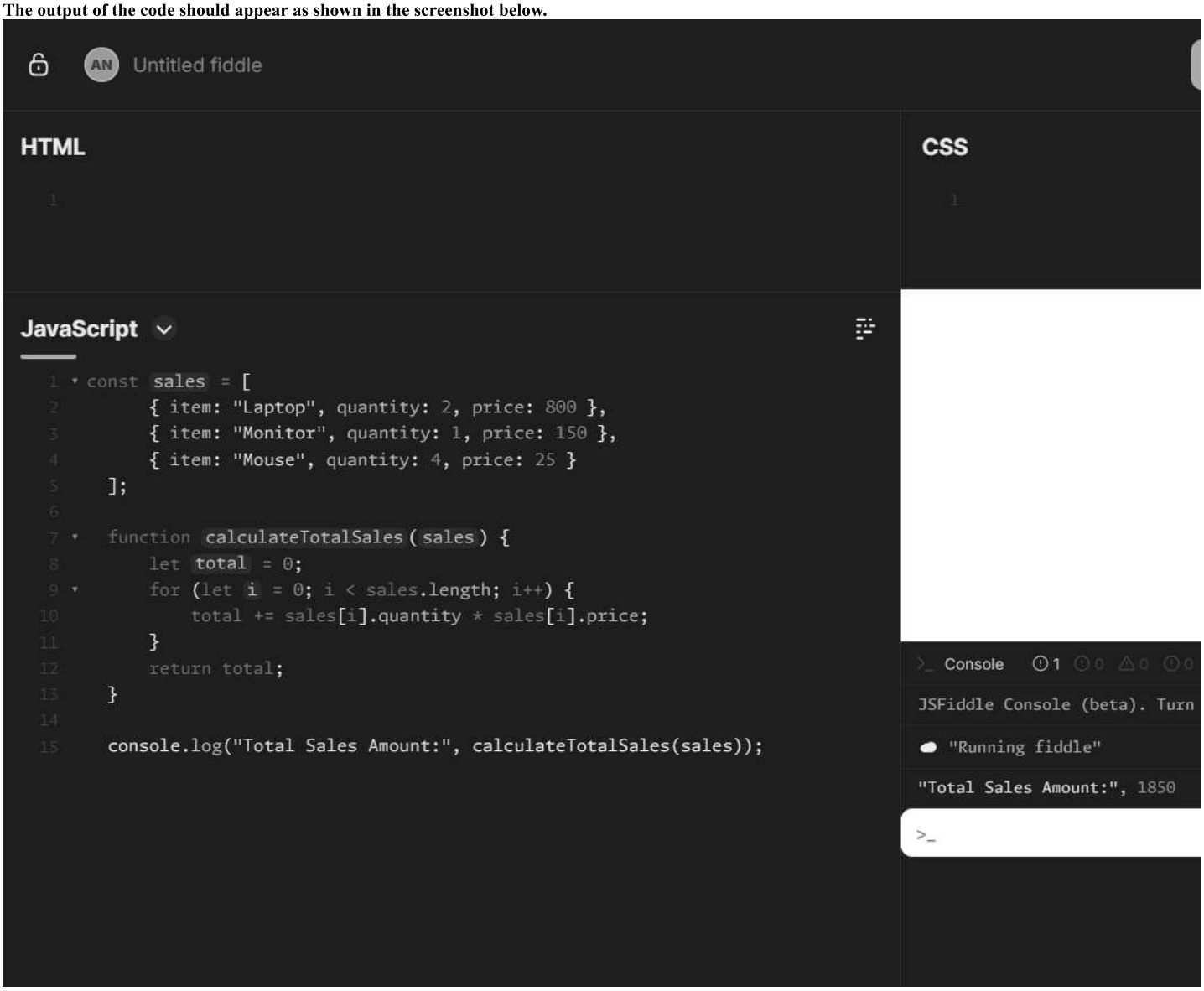
▼ Click here to see the solution code

```
const sales = [
    { item: "Laptop", quantity: 2, price: 800 },
    { item: "Monitor", quantity: 1, price: 150 },
    { item: "Mouse", quantity: 4, price: 25 }
];
function calculateTotalSales(sales) {
    let total = 0;
    for (let i = 0; i < sales.length; i++) {
        total += sales[i].quantity * sales[i].price;
    }
    return total;
}
console.log("Total Sales Amount:", calculateTotalSales(sales));
```

Write the program on JSFiddle:

- Go to JSFiddle
- Write the code in the JavaScript section
- Execute the program by clicking the `Run` button and check the results in the console section

The output of the code should appear as shown in the screenshot below.



Exercise 2: Generate an order receipt

Problem:
Write a JavaScript program that generates a receipt for a customer's order. The receipt should include each item's name, quantity, price, and total cost.

Input details:

- An array of objects representing ordered items. Each object has:
  - `item`: Name of the product (string)
  - `quantity`: Quantity ordered (integer)
  - `price`: Price per unit (float)

Output details:

- A detailed receipt showing each item's details and the grand total

Steps to implement:

1. Define an array of ordered items with at least 3 sample entries
2. Write a function `generateReceipt` that takes this array as input
3. Use a loop to iterate through the items and calculate the total for each item and the grand total
4. Print the receipt in a formatted string

▼ Click here to view hints

- The `orders` array contains objects, each with `item`, `quantity`, and `price` properties representing items ordered.
- Use a loop to iterate through each object in the `orders` array.
- For each item, calculate the total by multiplying `quantity` and `price`.
- Accumulate the totals in a variable `grandTotal`.
- Print each item's details and total cost using a formatted string.
- Finally, print the grand total after iterating through the array.

▼ Click here to see the solution code

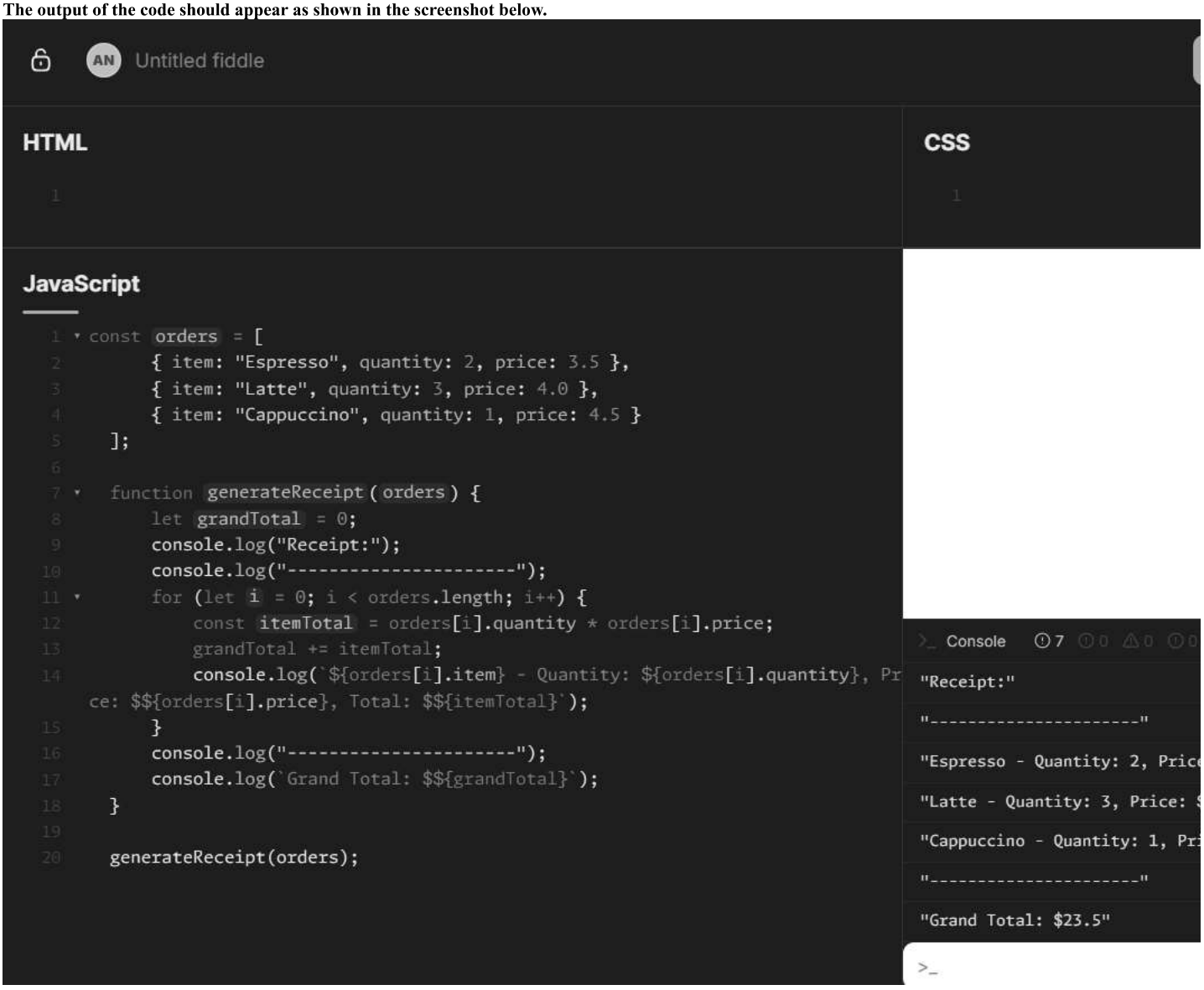
```
const orders = [
    { item: "Espresso", quantity: 2, price: 3.5 },
    { item: "Latte", quantity: 3, price: 4.0 },
    { item: "Cappuccino", quantity: 1, price: 4.5 }
];
```

```
function generateReceipt(orders) {
    let grandTotal = 0;
    console.log("Receipt:");
    console.log("----------------------");
    for (let i = 0; i < orders.length; i++) {
        const itemTotal = orders[i].quantity * orders[i].price;
        grandTotal += itemTotal;
        console.log(`${orders[i].item} - Quantity: ${orders[i].quantity}, Price: $${orders[i].price}, Total: $${itemTotal}`);
    }
    console.log("----------------------");
    console.log(`Grand Total: $${grandTotal}`);
}
generateReceipt(orders);
```

Write the program on JSFiddle:

- Go to JSFiddle
- write the code in the JavaScript section
- Execute the program by clicking the `Run` button and check the results in the console section

The output of the code should appear as shown in the screenshot below.



Exercise 3: Validate passwords

Problem:
Write a JavaScript program to validate a list of passwords. A password is valid if:

- It contains only alphanumeric characters (letters and numbers)
- It must be at least 8 characters long, but no more than 20 characters

Input details:

- An array of passwords (strings)

Output details:

- A message indicating whether each password is valid or invalid

Steps to implement:

1. Define an array of sample passwords

2. Write a function `validatePasswords` that takes this array as input
3. Use a loop to iterate through the passwords and check each against the validation criteria
4. Log whether each password is valid or invalid

▼ Click here to view hints

- The `passwords` array contains strings that need to be validated based on the given rules.
- Use a regular expression to define the pattern for valid passwords. The pattern should allow only alphanumeric characters and be between 8 and 20 characters long.
- Use the `test` method to check if each password matches the regular expression.
- Iterate over the `passwords` array and log each password as either valid or invalid based on the result of the regex check.

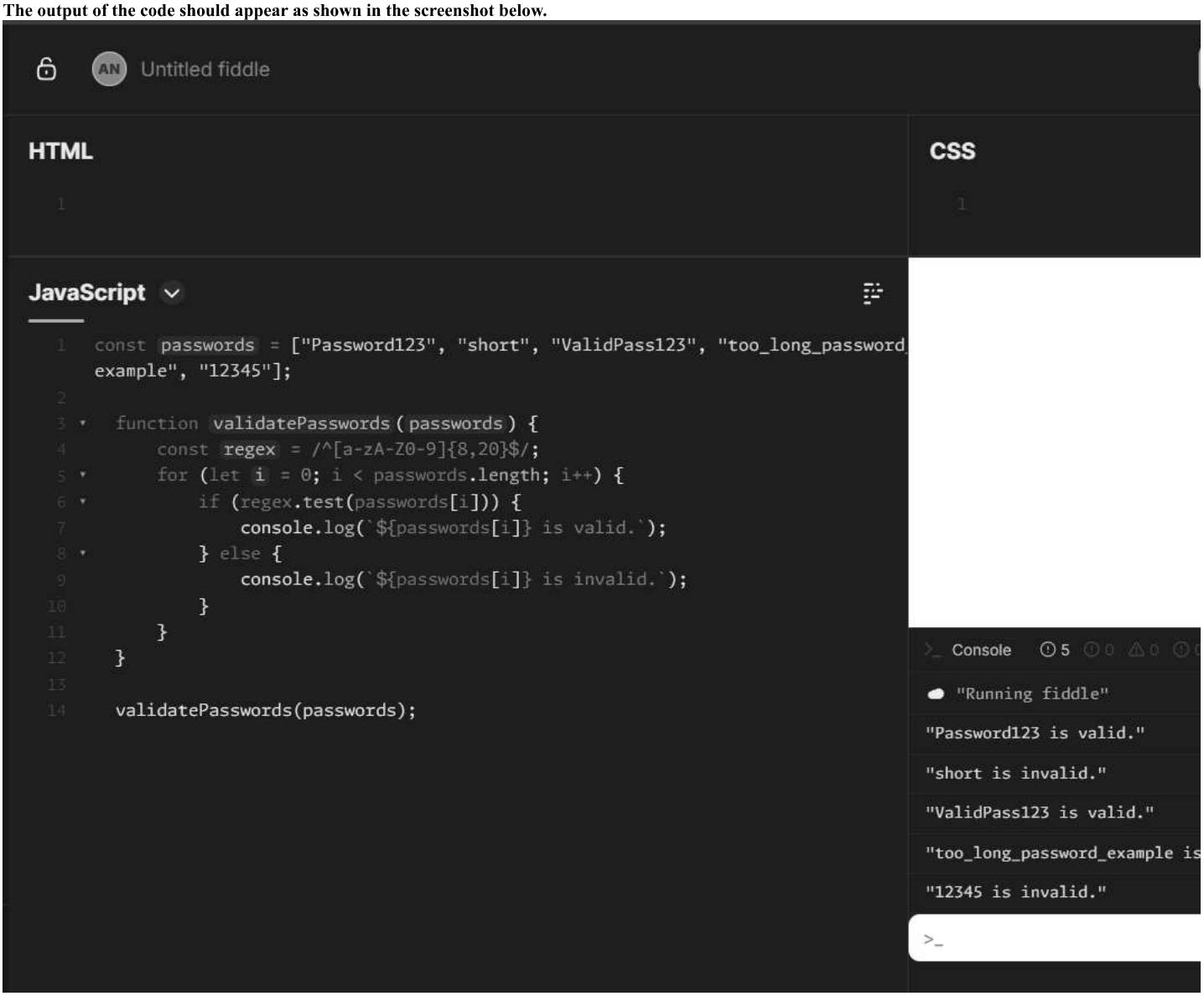
▼ Click here to see the solution code

```
const passwords = ["Password123", "short", "ValidPass123", "too_long_password_example", "12345"];
function validatePasswords(passwords) {
    const regex = /^[a-zA-Z0-9]{8,20}$/;
    for (let i = 0; i < passwords.length; i++) {
        if (regex.test(passwords[i])) {
            console.log(`${passwords[i]} is valid.`);
        } else {
            console.log(`${passwords[i]} is invalid.`);
        }
    }
}
validatePasswords(passwords);
```

Write the program on JSFiddle:

- Go to [JSFiddle](https://jsfiddle.ne
- write the code in the JavaScript section
- Execute the program by clicking the `Run` button and check the results in the console section

The output of the code should appear as shown in the screenshot below.



Exercise 4: Track product stock levels

Problem:
You are working for an online retail company. Your task is to write a JavaScript program that tracks the stock levels of various products in the inventory. The program should check if a product is in stock and log an appropriate message.

Input details:

- An array of objects representing products. Each object contains:
  - `product`: Name of the product (string)
  - `stock`: Number of units available in stock (integer)

Output details:

- A message for each product indicating whether the product is in stock or out of stock.

Steps to implement:

1. Define an array of product objects with at least 3 sample products
2. Write a function `checkStockLevels` that takes this array as input
3. Use a loop to iterate through the array and check the stock level for each product
4. Print a message indicating if the product is "In Stock" or "Out of Stock"

▼ Click here to view hints

- The `products` array contains objects with `product` and `stock` properties.
- Use a loop to iterate through the `products` array and check the stock level for each product.
- If the stock is greater than 0, log "In Stock"; otherwise, log "Out of Stock".

▼ Click here to see the solution code

```
const products = [
    { product: "Laptop", stock: 5 },
    { product: "Headphones", stock: 0 },
    { product: "Smartphone", stock: 3 }
];
function checkStockLevels(products) {
    for (let i = 0; i < products.length; i++) {
        if (products[i].stock > 0) {
            console.log(`${products[i].product} is In Stock.`);
        } else {
            console.log(`${products[i].product} is Out of Stock.`);
        }
    }
}
checkStockLevels(products);
```

Write the program on JSFiddle:

- Go to JSFiddle
- Write the code in the JavaScript section
- Execute the program by clicking the `Run` button and check the results in the console section

The output of the code should appear as shown in the screenshot below.

Untitled fiddle

HTML

CSS

JavaScript

```javascript
const products = [
        { product: "Laptop", stock: 5 },
        { product: "Headphones", stock: 0 },
        { product: "Smartphone", stock: 3 }
    ];

    function checkStockLevels(products) {
        for (let i = 0; i < products.length; i++) {
            if (products[i].stock > 0) {
                console.log(`${products[i].product} is In Stock.`);
            } else {
                console.log(`${products[i].product} is Out of Stock.`);
            }
        }
    }

    checkStockLevels(products);
```

Console 3

```
"Running fiddle"
"Laptop is In Stock."
"Headphones is Out of Stock."
"Smartphone is In Stock."
```

Conclusion:

Through these exercises, you have practiced solving intermediate-level problems using JavaScript. Each task focused on different aspects of logical thinking, from validation to string manipulation. Continue practicing similar challenges to further enhance your programming skills and confidence.

Author

Rajashree Patil

Skills Network