

Task 10: Implementing Debouncing and Throttling

Objective:

Create two separate functions to demonstrate the concepts of debouncing and throttling. Debouncing should ensure that a function is executed only after a specified delay after the last invocation. Throttling should ensure that a function is executed at most once in a specified time interval.

Pre-requisites:

- Basic JavaScript (variables, functions)
- Understanding of higher-order functions
- JavaScript Closures
- Event handling

Concepts Covered:

- Higher-order functions
- JavaScript closures
- Debouncing
- Throttling
- Event handling

Setup:

Install Node.js:

- Ensure Node.js is installed on your machine. You can download it from nodejs.org.

Tasks:

1. Define Debounce Function:

- **Task:**
 - Define a function named `debounce`.
 - The `debounce` function should take two arguments: the function to be debounced and the delay in milliseconds.
 - The debounced function should only execute after the specified delay has passed since the last invocation.
- **Outcome:**
 - Ensure the debounced function executes correctly after the specified delay.

2. Define Throttle Function:

- **Task:**
 - Define a function named `throttle`.
 - The `throttle` function should take two arguments: the function to be throttled and the interval in milliseconds.
 - The throttled function should only execute at most once every specified interval.

- **Outcome:**
 - Ensure the throttled function executes correctly at most once every specified interval.

Instructions:

- **Perform the following tasks:**
 - Write the required code in `index.js`.
 - Run the file using Node.js to ensure the code executes without errors and demonstrates the use of debouncing and throttling.

Example Input:

1. Debounce:

- Call the debounced function with the message "Start". Wait 500ms, call it with the message "Continue". Wait 1500ms, call it with the message "End".

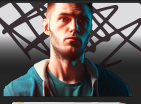
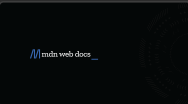



Expected Output:

- Only "End" should be logged after 2000ms.
- #### 2. Throttle:
- Call the throttled function with the message "First". Wait 500ms, call it with the message "Second". Wait 1000ms, call it with the message "Third". Wait 3000ms, call it with the message "Fourth".

Expected Output:

- "First" should be logged immediately, "Third" after 1000ms, and "Fourth" after 3000ms.

Resources:

-  **Debounce and Throttling: What They Are and When to Use Them**
If you are a web developer, you might have encountered situations where you need to optimize t...
Publisher Medium Author Bablu Singh Published 8/20/2023
-  **Closures - JavaScript | MDN**
A closure is the combination of a function bundled together (enclosed) with references to i...
 <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures>
-  **Event reference | MDN**
Events are fired to notify code of "interesting changes" that may affect code execution. Th...
 <https://developer.mozilla.org/en-US/docs/Web/Events>

Videos:



GitHub Instructions:

1. Open in Visual Studio Code:

- After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, Visual Studio Code (VSCode) will open the repository directly.
- If prompted, select "Open" or "Allow" to open the repository in VSCode.

2. Open the Terminal in VSCode:

- In VSCode, open a terminal by selecting Terminal > New Terminal from the top menu.

3. Complete the Task:

- In VSCode, write your solution in the `index.js` file.

4. Run and Test Your Code:

- In the VSCode terminal, navigate to the directory containing `index.js`.
- Run your code to ensure it works correctly. Use the following commands:

```
node index.js debounce "Start" 500
node index.js debounce "Continue" 1500
node index.js debounce "End" 2000

node index.js throttle "First" 0
node index.js throttle "Second" 500
node index.js throttle "Third" 1000
node index.js throttle "Fourth" 3000
```

5. Commit Your Changes:

- In the VSCode terminal, add your changes to git:

```
git add index.js
```

- Commit your changes with a meaningful message:

```
git commit -m "Completed task 10"
```

6. Push Your Changes to Your Repository:

- Push your changes to your forked repository:

```
git push origin main
```

7. Create a Pull Request:

- Go to your repository on GitHub.
- Click on the "Pull Requests" tab.
- Click the "New Pull Request" button.
- Ensure the base repository is the original template repository and the base branch is `main`.
- Ensure the head repository is your forked repository and the compare branch is `main`.
- Click "Create Pull Request".
- Add a title and description for your pull request and submit it.

Summary of Commands:

```
# Open in Visual Studio Code

# Open terminal in VSCode

# Complete the task by editing index.js

# Navigate to the directory containing index.js
cd path/to/your/index.js

# Run your code
node index.js debounce "Start" 500
node index.js debounce "Continue" 1500
node index.js debounce "End" 2000

node index.js throttle "First" 0
node index.js throttle "Second" 500
node index.js throttle "Third" 1000
node index.js throttle "Fourth" 3000

# Add, commit, and push your changes
```

```
git add index.js
git commit -m "Completed task 10"
git push origin main
```

Create a pull request on GitHub

Need Help?



Task 10 Solution

Code Hints — Debounce Function: — Ensure your debounce function correctly delays the exe...



w3o NFThing

Last Edited 7/3/2024