Task 11: Implementing the Observer Pattern for a Notification System

Level: Advanced

Description:

Create an observer pattern implementation to handle a simple notification system. The notification system should allow multiple observers to subscribe to notifications and be notified when an event occurs.

Objective:

Implement an observer pattern to create a notification system where observers can subscribe to events, and the system will notify all subscribed observers when an event occurs.

Pre-requisites:

- Basic JavaScript (variables, functions, objects)
- JavaScript Closures
- Observer pattern
- Event handling

Concepts Covered:

- Observer pattern
- JavaScript closures
- Event handling
- Encapsulation and state management

Setup:

Install Node.js:

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

Tasks:

1. **Define the Notification System Module:**
    - **Task:**
        - Define a module named `notificationSystem`.
        - The `notificationSystem` module should encapsulate a list of observers.
        - The module should expose the following public methods:
            - `subscribe(observer)` : Adds an observer to the list.
            - `unsubscribe(observer)` : Removes an observer from the list.
            - `notify(data)` : Notifies all subscribed observers, passing the data to each observer's update method.
        - Ensure that the list of observers is not directly accessible from outside the module.
    - **Outcome:**
        - Ensure the module correctly manages the list of observers and notifies them appropriately.

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 the observer pattern.

Example Input Sets and Expected Output:

1. **Set 1:**
   - **Input:**
     - Subscribe `observer1` and `observer2` to `notificationSystem`.
     - Notify with data "Event A".
     - Unsubscribe `observer2`.
     - Notify with data "Event B".
   - **Expected Output:**

   ```
   Observer 1 received: Event A
   Observer 2 received: Event A
   Observer 1 received: Event B
   ```

2. **Set 2:**
   - **Input:**
     - Subscribe `observer1` to `notificationSystem`.
     - Notify with data "Event X".
     - Subscribe `observer2`.
     - Notify with data "Event Y".
   - **Expected Output:**

   ```
   Observer 1 received: Event X
   Observer 1 received: Event Y
   Observer 2 received: Event Y
   ```

3. **Set 3:**
   - **Input:**
     - Subscribe `observer2` to `notificationSystem`.
     - Notify with data "Event 100".
     - Unsubscribe `observer2`.
     - Notify with data "Event 101".
   - **Expected Output:**

   ```
   Observer 2 received: Event 100
   ```

4. **Set 4:**
    - **Input:**
        - Subscribe `observer1` and `observer2` to `notificationSystem`.
        - Notify with data "Event Hello".
        - Unsubscribe both `observer1` and `observer2`.
        - Notify with data "Event Goodbye".
    - **Expected Output:**

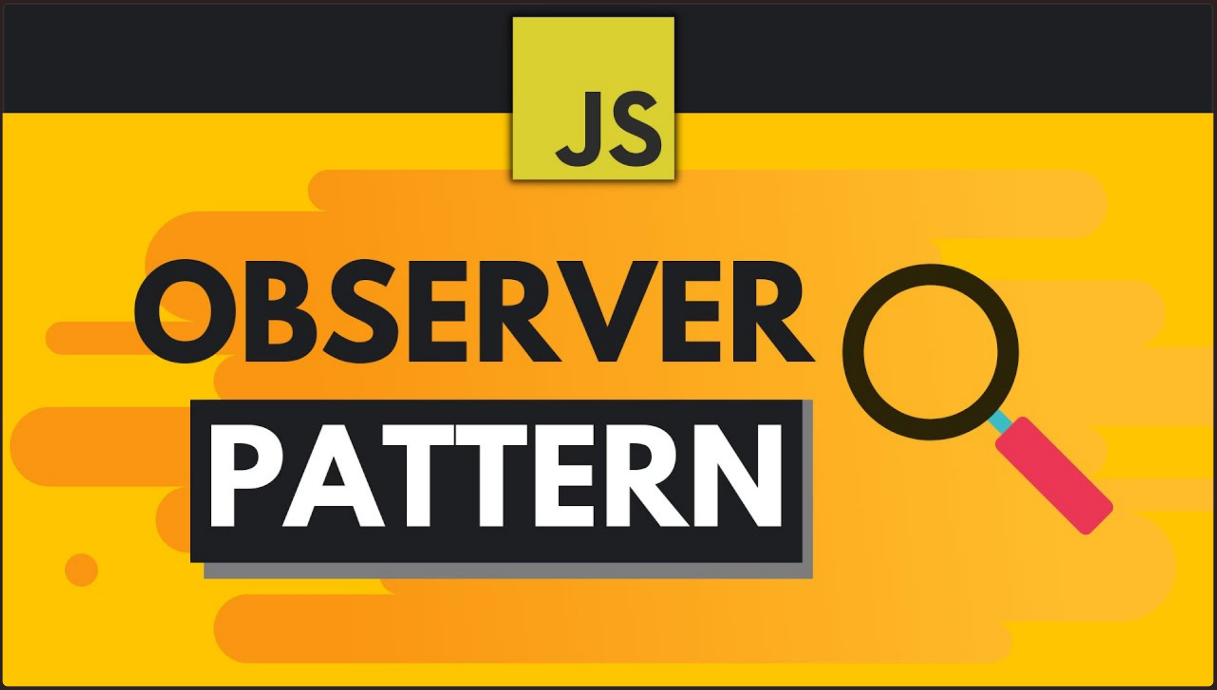
```
Observer 1 received: Event Hello
Observer 2 received: Event Hello
```

Resources:

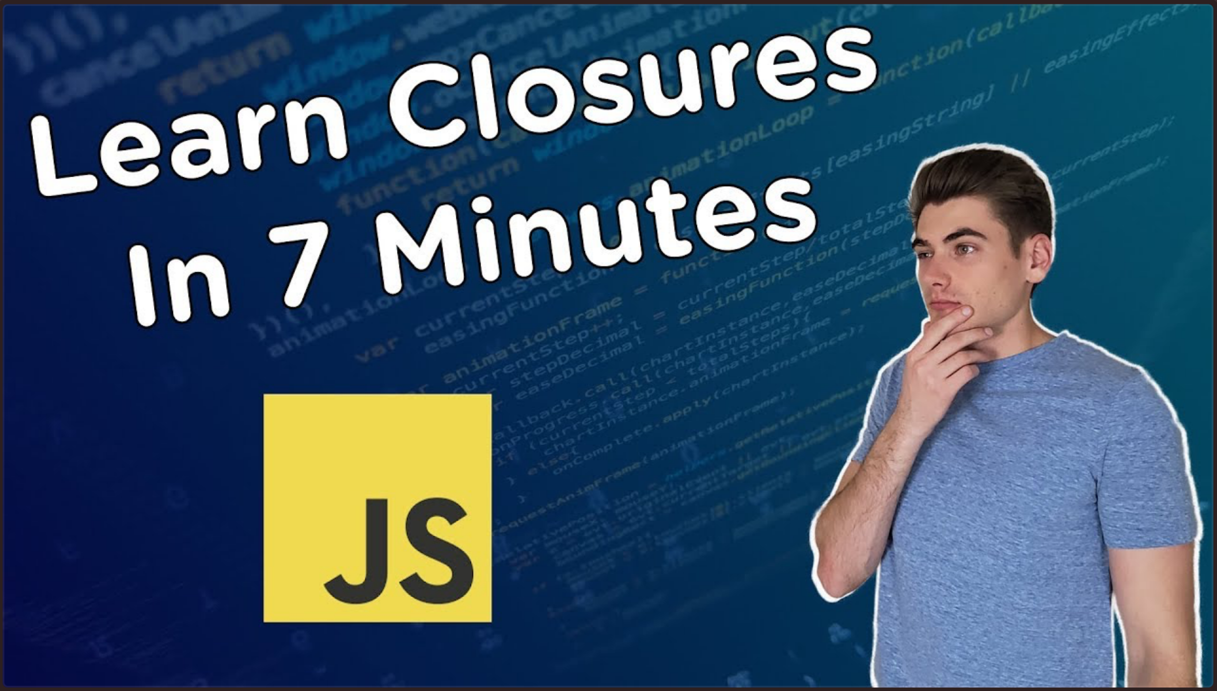
- Closures - JavaScript | MDN
  A closure is the combination of a function bundled together (enclosed) with references to i...
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures
-

  Observer
  Observer is a behavioral design pattern that lets you define a subscription mechanism to...
  https://refactoring.guru/design-patterns/observer
- Event handling (overview) - Event reference | MDN
  Events are signals fired inside the browser window that notify of changes in the browser or...
  https://developer.mozilla.org/en-US/docs/Web/Events/Event_handlers

Videos:



DevSage 8:12min 53,011 Views 1,667 Likes



Web Dev Simplified 6:56min 321,122 Views 10,697 Likes

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
   ```

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 11"
   ```

6. **Push Your Changes to Your Repository:**
   - Push your changes to your forked repository:

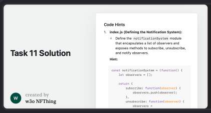
   ```
   git push origin main
   ```

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

# Add, commit, and push your changes
git add index.js
git commit -m "Completed task 11"
git push origin main

# Create a pull request on GitHub
```

Need Help?



Task 11 Solution

created by
w3o NFThing

Code Hints

1. **index.js (Defining the Notification System):**
   - Define the `notificationSystem` module that encapsulates a list of observers and exposes methods to subscribe, unsubscribe, and notify observers.

   Hint:

```
const notificationSystem = (function() {
  let observers = [];

  return {
    subscribe: function(observer) {
      observers.push(observer);
    },
    unsubscribe: function(observer) {
      observers =
```

Task 11 Solution

w3o NFThing Last Edited 7/3/2024