React Tail Game Documentation

Introduction:

This is a React-based memory game where players can match pairs of cards with identical images. The game consists of a welcome page where the user enters their name, a game board with cards, and a timer to track the time taken to complete the game.

Gameplay Walkthrough

1. Welcome Page

- When the application loads, the user is greeted with the "React Tiles" title and a prompt to enter their name on the welcome page.
- The user types their name in the input field and clicks the "Start Game" button.

2. Game Board

- Upon clicking "Start Game," the game board appears with a grid of face-down cards.
- The user's name is displayed at the top of the game board, and a score and timer section is shown.

3. Gameplay

- The player clicks on a card to reveal its image. If it's the first card in a turn, its image stays visible.
- If it's the second card in a turn, the game checks if the two cards have identical images.
    - If they match, they remain face up, and the player scores a point.
    - If they don't match, the cards flip back face down after a brief delay.
- The player continues flipping cards and making matches until all pairs are found.

4. Game Completion

- When all pairs are matched, the game displays a "Game Finished!" message along with the player's score, time taken, and total turns.
- The user can choose to restart the game by reloading the page.

Components:

1. App (App.js)

The **App** component serves as the main entry point for the application. It manages the user's name and controls the transition between the welcome page and the game board.

State:

- **userName**: Represents the user's name.

Methods:

- **handleNameSubmit(name)**: Handles the submission of the user's name and sets it in the state.

Render:

- Conditionally renders either the **WelcomePage** or the **Game** component based on whether the user's name is set.

2. WelcomePage (WelcomePage.js)

The **WelcomePage** component is responsible for capturing the user's name and initiating the start of the game.

State:

- **userName**: Stores the user's name input.

Methods:

- **handleNameChange(e)**: Updates the **userName** state as the user types.
- **handleSubmit()**: Invokes the **onNameSubmit** prop to send the user's name to the parent component.

Render:

- Displays a form with an input for the user's name and a "Start Game" button.

3. Game (Game.js)

The **Game** component orchestrates the memory game, including managing game state, handling user interactions, and rendering the game board.

State:

- **timer**: Tracks the time elapsed in seconds.
- **isGameFinished**: Indicates whether the game has been completed.
- **turnNo**: Tracks the number of turns.
- **pairsFound**: Tracks the number of pairs matched.
- **numClicksWithinTurn**: Counts the number of clicks within a turn.
- **firstId** and **secondId**: Store the IDs of the clicked cards within a turn.

Lifecycle Methods:

- **componentDidMount()**: Initializes the game and starts the timer.
- **componentWillUnmount()**: Clears the timer interval to prevent memory leaks.
- **componentDidUpdate(prevProps, prevState)**: Checks if all pairs are found and stops the timer if true.

Methods:

- **initGame()**: Initializes the game state and generates a new set of memory cards.
- **startTimer()**: Initiates the timer interval.
- **stopTimer()**: Stops the timer interval and sets **isGameFinished** to true.
- **getCardViews()**: Generates **CardView** components for each card in the game.
- **clearCards(id1, id2)**: Clears the cards if they are not a match.
- **onCardClicked(id, image)**: Handles card click events and determines if cards match or not.

Render:

- Renders the game board, user information, and game status.
- Conditionally renders the game finished screen when all pairs are found.

4. CardView (CardView.js)

The **CardView** component represents an individual card on the game board. It handles user clicks and displays the card's state.

Props:

- **id**: Unique identifier for the card.
- **image**: The image associated with the card.
- **imageUp**: Indicates whether the card is facing up.
- **matched**: Indicates whether the card is matched.
- **onClick**: Callback function for card click events.

Methods:

- **onClick()**: Handles the click event on the card, triggering the parent's **onClick** prop.

Render:

- Renders the card's view, displaying the image or a blank space based on its state.

5. MemoryCards (MemoryCards.js)

The **MemoryCards** class is responsible for managing the set of cards, including generating the initial card set, flipping cards, and checking for matches.

Properties:

- **cards**: An array containing the card objects.
- **NUM_IMAGES**: The total number of unique images in the game.

Methods:

- **generateCardSet()**: Generates a new set of cards with matching pairs and shuffles them.
- **getCard(id)**: Retrieves a card object by its ID.
- **flipCard(id, imageUp)**: Changes the state of a card to either face up or face down.
- **setCardAsMatched(id, matched)**: Marks a card as matched.
- **cardsHaveIdenticalImages(id1, id2)**: Checks if two cards have identical images.

Code Explanation

1. Initialization:

- **App Component (App.js):**
  - Initially, the **App** component sets up the state with **userName** initialized to **null**.
  - It conditionally renders either the **WelcomePage** or the **Game** component based on whether **userName** is set.
- **WelcomePage Component (WelcomePage.js):**
  - The user enters their name in the input field and clicks the "Start Game" button.
  - The **handleNameChange** function updates the **userName** state as the user types.
  - The **handleSubmit** function ensures a non-empty name and triggers the **onNameSubmit** prop to send the user's name to the parent (**App**) component.

2. Game Initialization:

- **Game Component (Game.js):**
  - When the user's name is submitted, the **Game** component is rendered.
  - The **componentDidMount** lifecycle method initializes the game, generating a set of memory cards using the **MemoryCards** class, and starts the timer.
- **MemoryCards Class (MemoryCards.js):**
  - The **generateCardSet** method creates a set of cards with matching pairs, shuffles them, and stores them in the **cards** array.

3. Gameplay:

- **Game Component (Game.js):**
    - The **getCardViews** method generates **CardView** components based on the current state of the memory cards.
- **CardView Component (CardView.js):**
    - Each **CardView** represents an individual card on the game board.
    - The **onClick** method handles user clicks on the card, triggering the **onCardClicked** prop in the parent (**Game**) component.
- **Game Component (Game.js):**
    - The **onCardClicked** method manages the game logic:
        - Flips cards based on user clicks.
        - Checks for matches and updates the game state accordingly.
        - Controls the timer and tracks the number of turns and pairs found.

4. Game Completion:

- **Game Component (Game.js):**
    - The **componentDidUpdate** lifecycle method checks if all pairs are found and stops the timer, triggering the game completion.
- **Game Completion Screen (Game.js):**
    - When all pairs are matched, the game displays a completion screen with the user's name, score, time taken, and total turns.

5. Timer and Score:

- **Game Component (Game.js):**
    - The timer increments every second and is displayed on the screen.
    - The score represents the number of pairs found.

6. Restarting the Game:

- The user can restart the game by reloading the page.