أكاديمية البرمجة
Coding Academy
powered by Simplon

JavaScript asynchronous

Simulating an Order Workflow Using setTimeout and Callbacks

create a program that processes an order through the following steps:

1. Placing the Order: Simulate the action of placing a customer's order.
2. Preparing the Order: Represent the time it takes to prepare the order in the kitchen.
3. Packaging the Order: Indicate the time it takes to package the order.
4. Delivering the Order: Simulate delivering the order to the customer.

Each step will take a certain amount of time, represented using setTimeout, and must execute only after the previous step is completed. You will use callbacks to ensure this sequential flow.

Task Steps:

1. Implement the following functions:
    - **placeOrder(order, callback)**
        - Logs the message: Placing order for: {order}.
        - After a delay of 2 seconds, logs: Order for {order} has been placed.
        - Executes the provided callback with the order.

    - **prepareOrder(order, callback)**
        - Logs the message: Preparing: {order}.
        - After a delay of 3 seconds, logs: {order} is ready.
        - Executes the provided callback with the order.

  - **packageOrder(order, callback)**
    - Logs the message: Packaging: {order}.
    - After a delay of 2 seconds, logs: {order} has been packaged.
    - Executes the provided callback with the order.

  - **deliverOrder(order, callback)**
    - Logs the message: Delivering: {order}.
    - After a delay of 4 seconds, logs: {order} has been delivered.
    - Executes the provided callback.

2. Write the **processOrder(order)** function:
   - This function initiates the order workflow by calling placeOrder.
   - Ensure that all functions are called in sequence using callbacks, and the process ends with logging: Order process completed successfully.

3. Test your implementation.