Practical Use Cases for Generators

Maaike van Putten

Lead Trainer & Software Developer

@brightboost | www.brightboost.nl

Use Cases

Generators are used for situations where there's a lot of data, memory needs to be considered, and iteration is required.

Overview

Possibilities with generators:

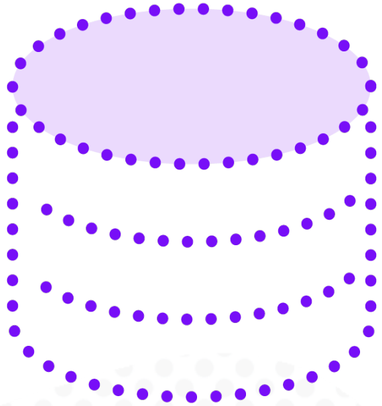
- Lazy evaluation
- Data pagination
- Data streaming
- Generators with databases or external APIs
- Design patterns: observer and mediator pattern

Lazy evaluation

Lazy Evaluation and Efficiently Handling Memory

Big advantage of generators

Lazy evaluation means calculating values on demand



Generators are lazy by nature.

Demo

Node Express project

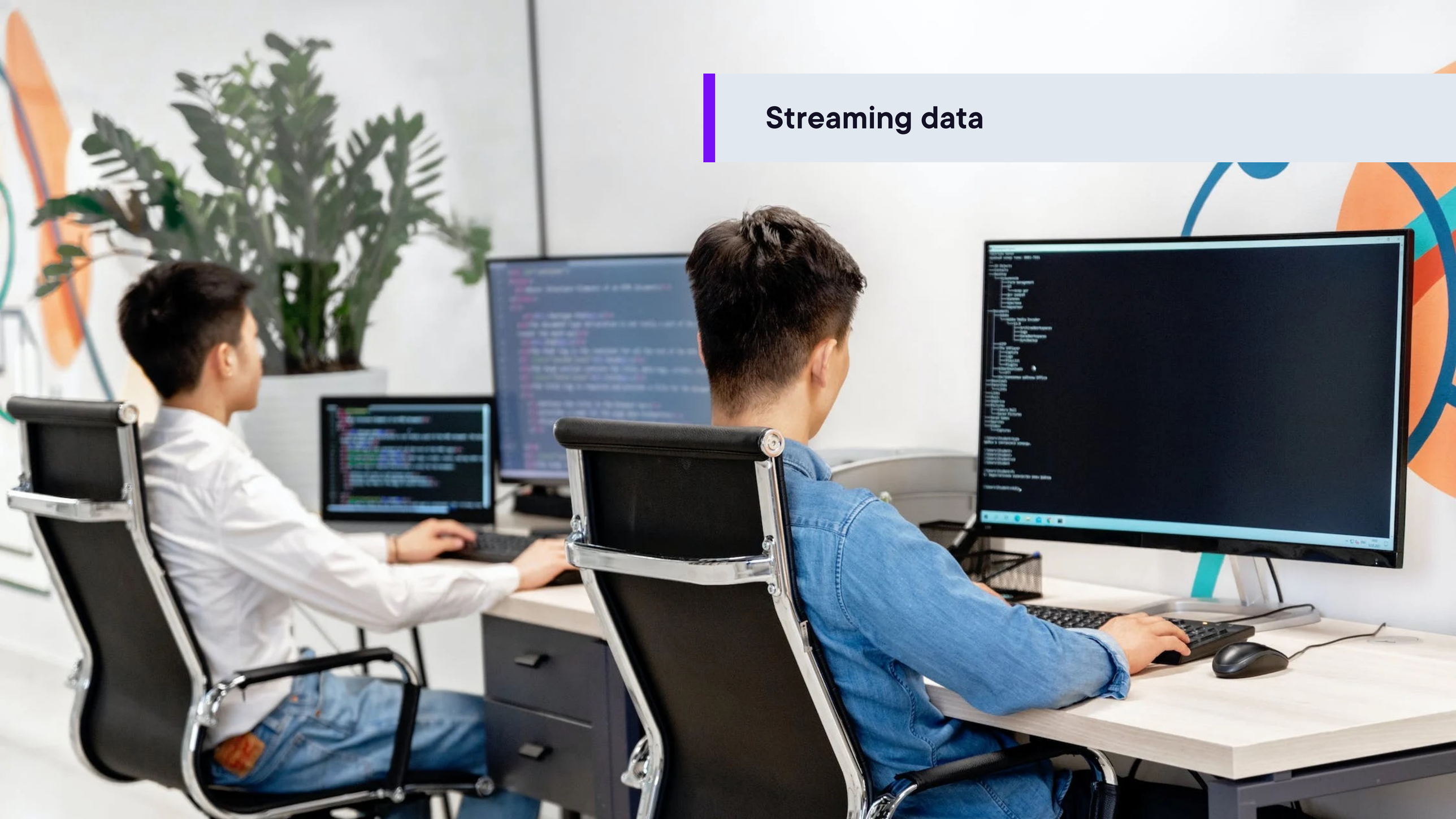
Lazy evaluation: generator that processes a large dataset

Display the products on demand

Load more products on scroll with the use of a generator

Because calling the database multiple times is typically worse for performance than the small dataset in the memory.

Streaming data



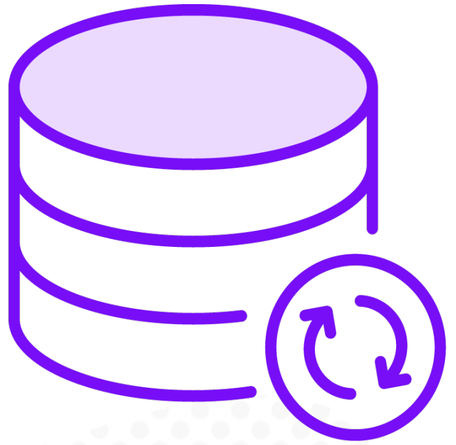
Data Streaming

Close to the nature of generators

Dealing with data in chunks

Lazy evaluation

Memory optimization



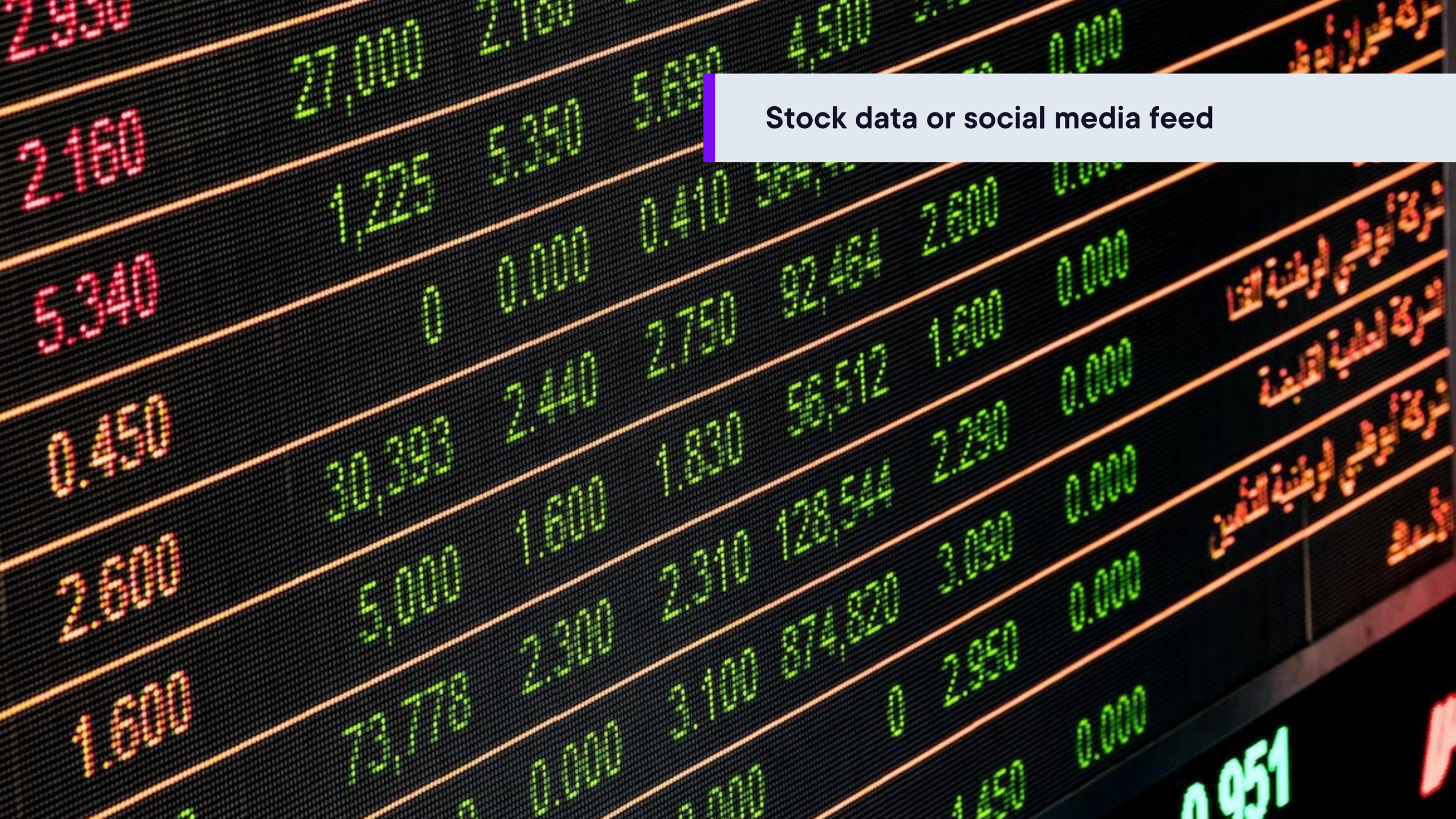
Demo

Web socket that is sending data

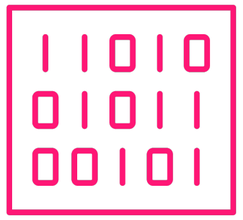
Web socket utilizes a generator

Log the information in the console on the frontend

This is typically more complex.

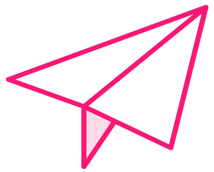

Stock data or social media feed

Low-Level Processing



Receive the bytes

Convert the bytes

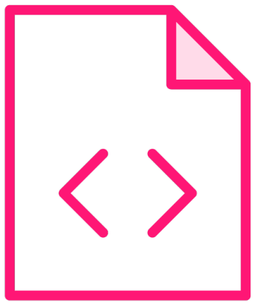


Yield the result

Using the Result of Processing the Data

Display on the website

Write to a database



Write to a file

Generators and databases

Combining Generators with Databases and APIs

Useful when working with big datasets

Retrieving and processing chunk by chunk

Demo

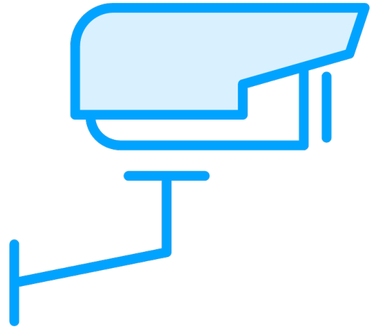
Fetching results from the database in batches to make sure it fits in the memory

Processing the data

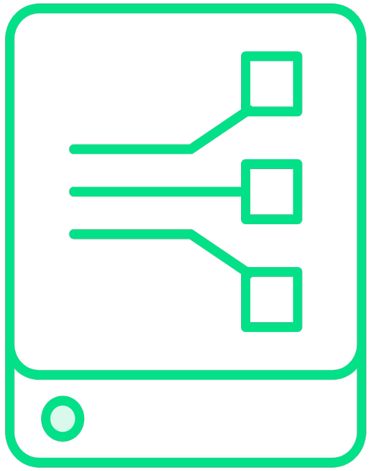
Writing the results to a CSV file

Design patterns

Observer Pattern and Mediator Pattern



Observer Pattern



Mediator Pattern

Observer Pattern

Subject maintains a list of its dependents

Dependents are called observers

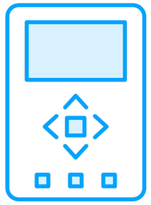
Subject notifies observers of state changes

Event handling systems

Demo

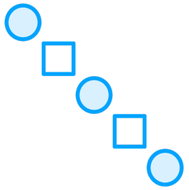
Integrate generators in the observer pattern

Mediator Pattern

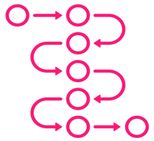


Used to control and manage communication between different objects in a system

Instead of the objects referring to each other directly, a mediator object is used



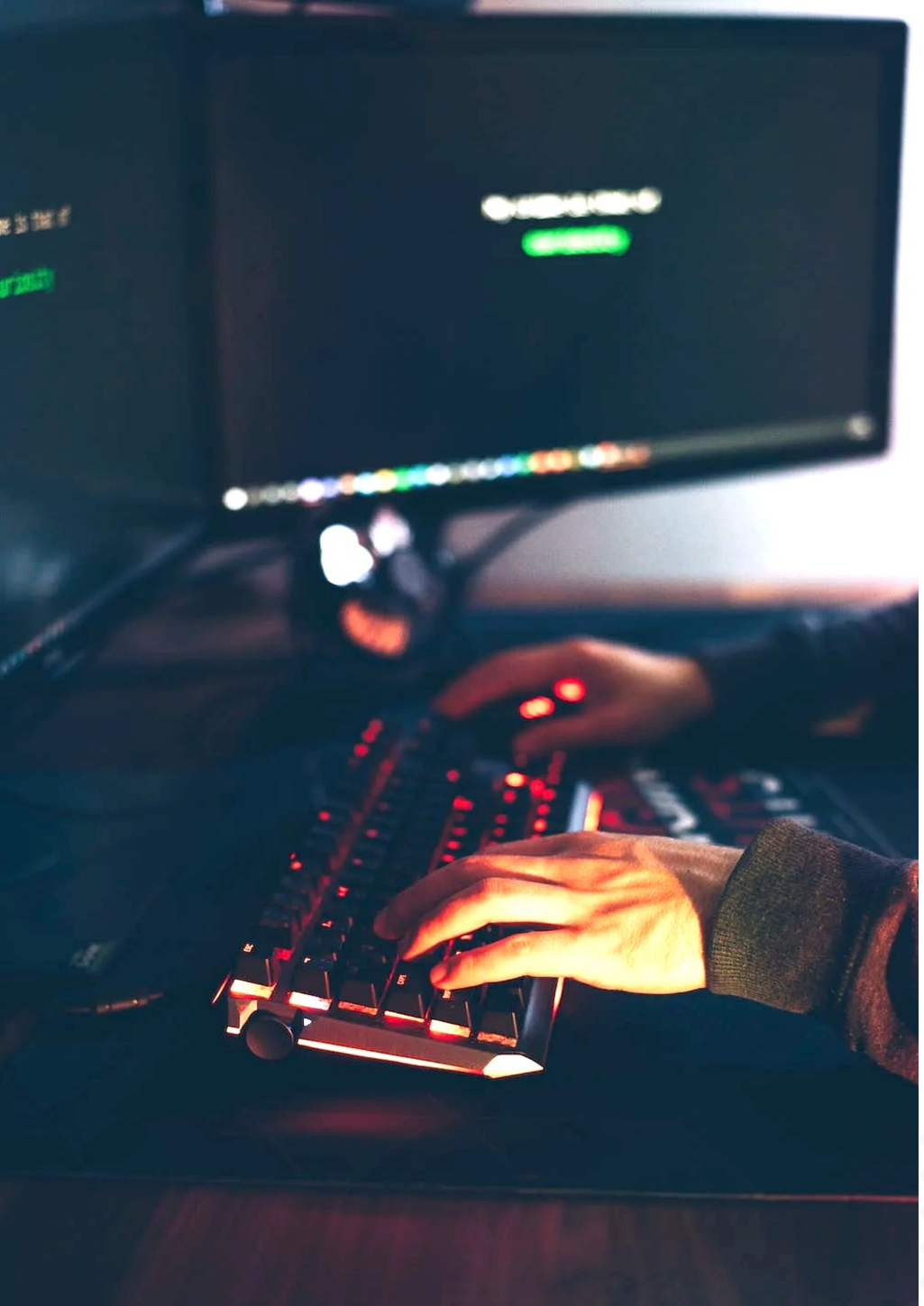
Pattern is essential for reducing coupling between classes and objects



Generators can be used to achieve this

Demo

Using a generator to manage the communication between components in the Mediator class



Generators can be used in the observer and mediator pattern

They add a layer of control and flexibility

Manage sequence of operations

Module summary

Summary

Possibilities with generators:

Lazy evaluation

Data pagination

Data streaming

Generators with databases or external APIs

Design patterns: observer and mediator pattern

Up Next:

Best Practices and Production Concerns