Amrita Vishwa Vidyapeetham
TIFAC-CORE in Cyber Security

20CYS312 - Principles of Programming Languages
Assignment-01: Exploring Programming Paradigms

M C Vivek Veera

21st January, 2024

Paradigm 1: Event-driven

Implementation:

- Event-driven programming is a programming paradigm in which the flow of the program is determined by external events.

- Program listens for events and then triggers a callback function when one of those events is detected.

- Control flow is determined mainly by events, such as mouse clicks or interrupts including timer.

Concepts:

Event Loop:

Call Stack: Keep track of the currently executing function.

Callback Queue: It is a FIFO data structure that holds callback functions waiting to be executed.

The **event loop** continuously checks the **call stack** and the **callback queue**. If the call stack is empty, it takes the first function from the callback queue and pushes it onto the call stack for execution..

Event Handlers:

Functions that will be triggered in response to interactions like clicking, hovering, focusing form inputs, and so on.
These functions are written by the programmer before hand.

Asynchrony:

Functions running in parallel with other functions are called **asynchronous.**

Language for Paradigm 1: RxJS

RxJS, short for Reactive Extensions for JavaScript, is a library that uses observable sequences to enable developers to work with asynchronous data streams.

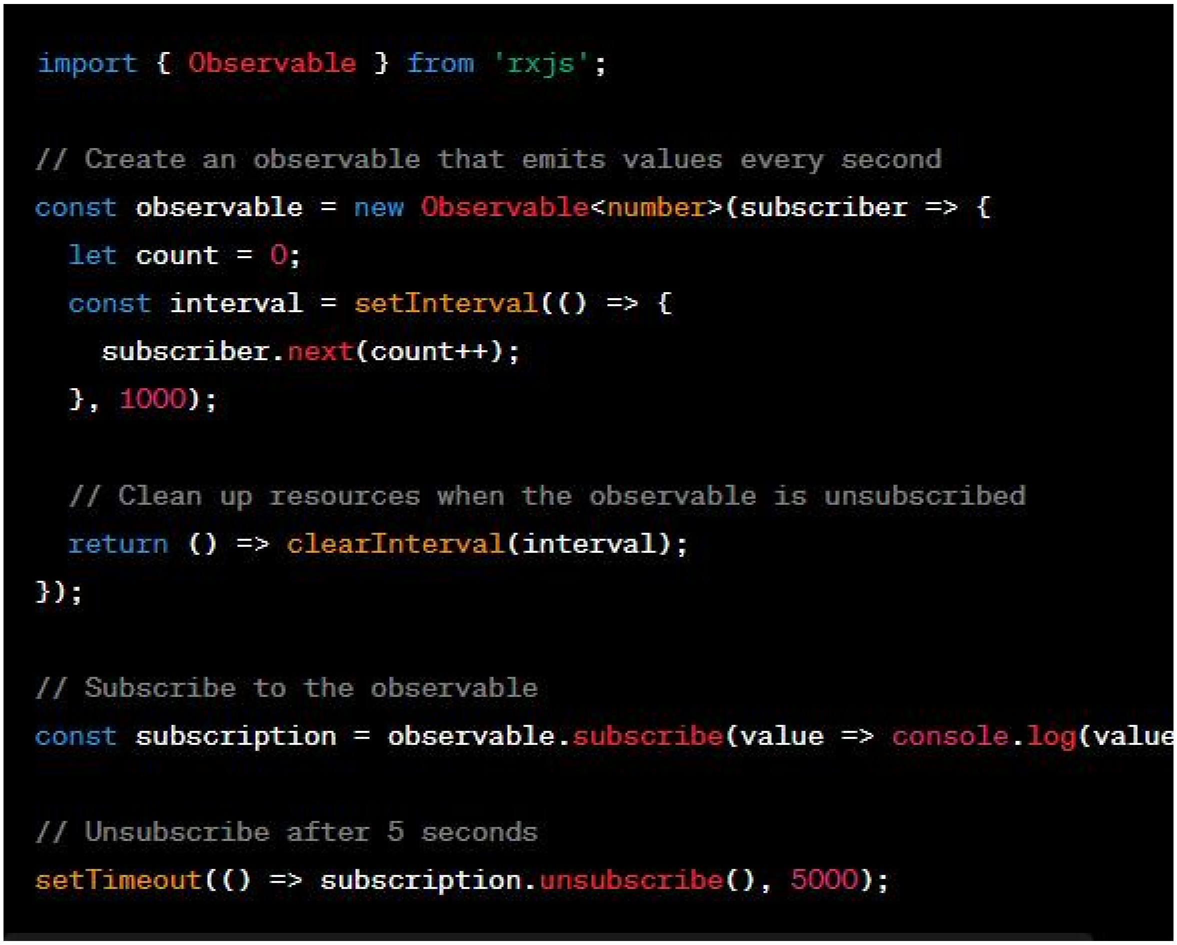
It’s essentially a set of tools that allows you to apply Reactive Programming principles in JavaScript.

RxJS revolves around four fundamental concepts: **Observables**, **Operators**, **Asynchronous Handling** and **Event Handling**.

Observables:

Observables represent sequences of values or events over time.

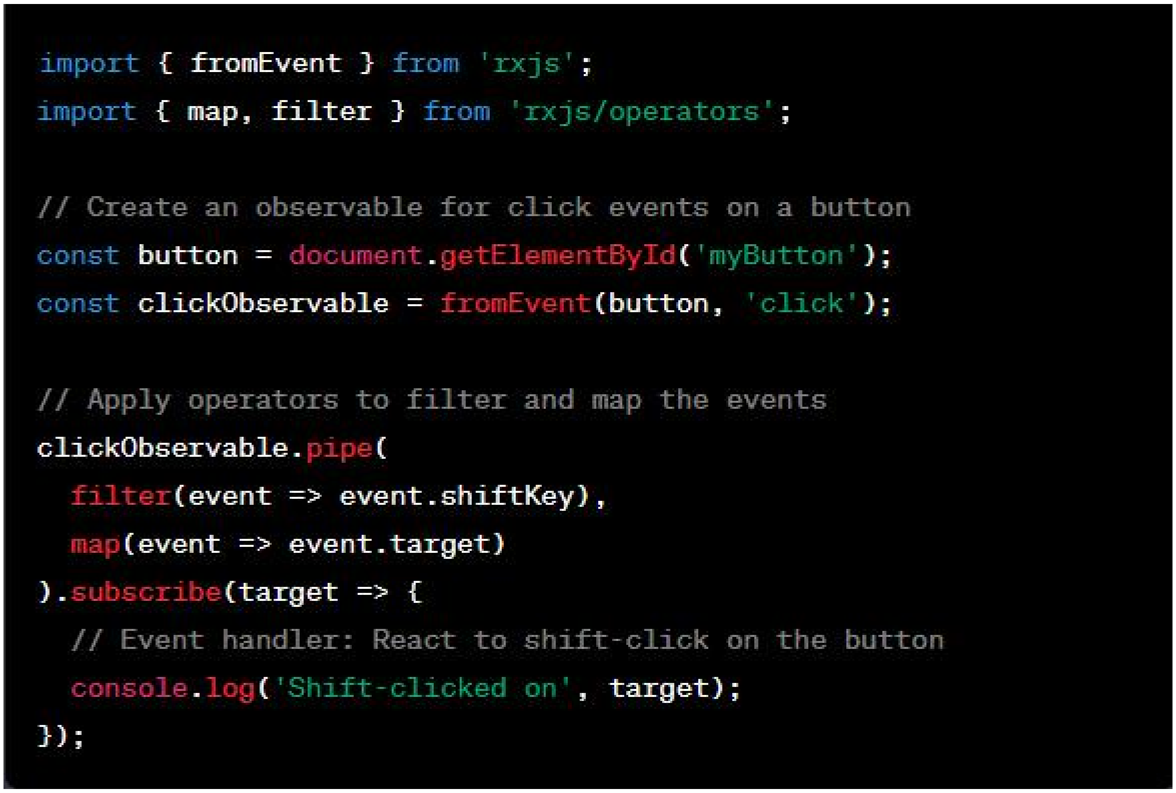
Observable streams can give multiple values asynchronously, allowing us to work with dynamic data sources.



```
import { Observable } from 'rxjs';

// Create an observable that emits values every second
const observable = new Observable<number>(subscriber => {
  let count = 0;
  const interval = setInterval(() => {
    subscriber.next(count++);
  }, 1000);

  // Clean up resources when the observable is unsubscribed
  return () => clearInterval(interval);
});

// Subscribe to the observable
const subscription = observable.subscribe(value => console.log(value

// Unsubscribe after 5 seconds
setTimeout(() => subscription.unsubscribe(), 5000);
```

Operators:

RxJS provides a large set of operators for transforming and manipulating observables. Eg. map, filter,etc.

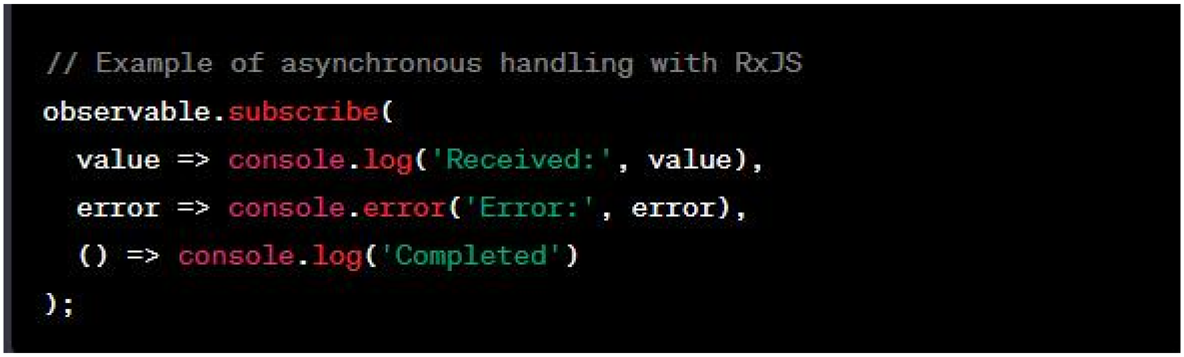


```
import { fromEvent } from 'rxjs';
import { map, filter } from 'rxjs/operators';

// Create an observable for click events on a button
const button = document.getElementById('myButton');
const clickObservable = fromEvent(button, 'click');

// Apply operators to filter and map the events
clickObservable.pipe(
  filter(event => event.shiftKey),
  map(event => event.target)
).subscribe(target => {
  // Event handler: React to shift-click on the button
  console.log('Shift-clicked on', target);
});
```

Asynchronous Handling:

RxJS simplifies the handling of asynchronous operations by using ‘subsrcibe’ method.

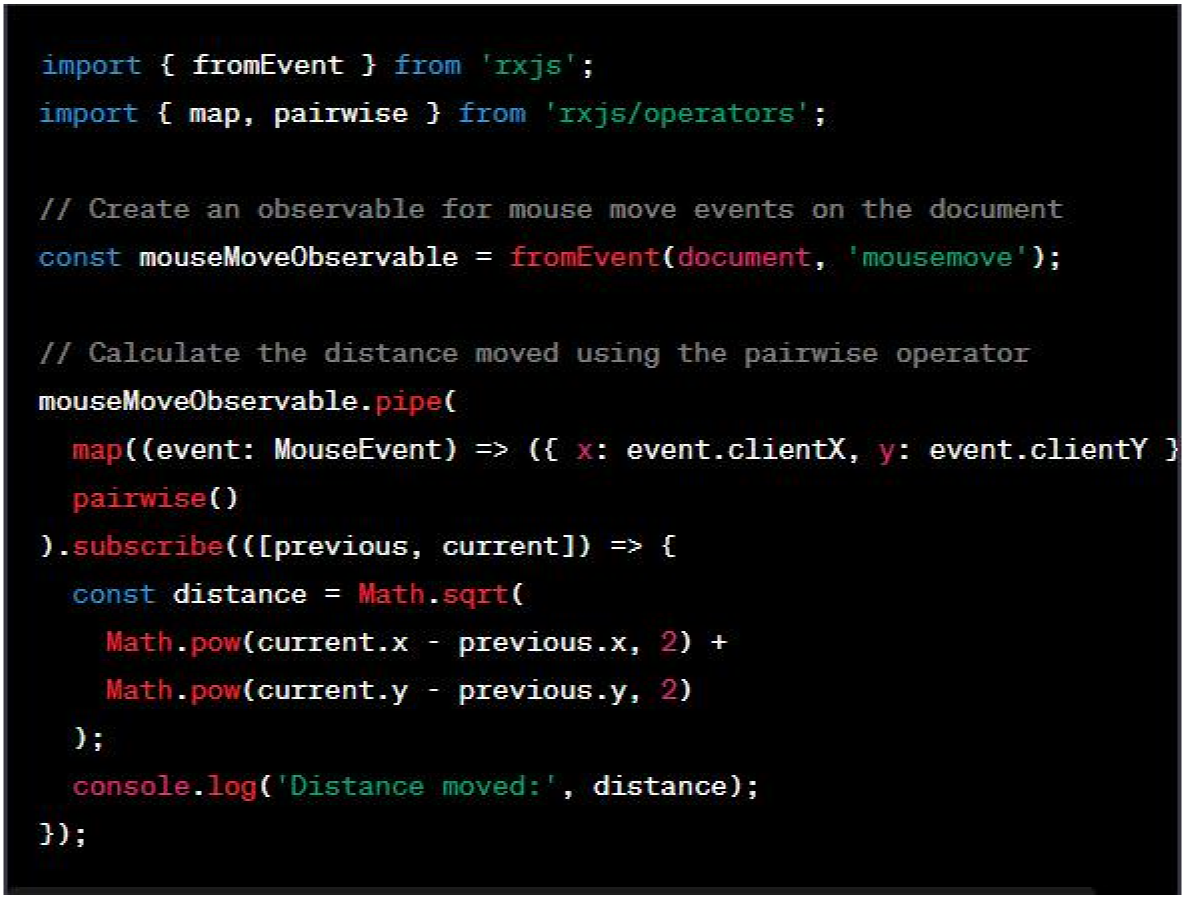
The subscribe metho initiates the execution of an observable and handle its emitted values.



```
// Example of asynchronous handling with RxJS
observable.subscribe(
  value => console.log('Received:', value),
  error => console.error('Error:', error),
  () => console.log('Completed')
);
```

Event Handling:

RxJS is extensively used for event handling, especially in web development where user interactionsis what determiness the application's behavior.



```
import { fromEvent } from 'rxjs';
import { map, pairwise } from 'rxjs/operators';

// Create an observable for mouse move events on the document
const mouseMoveObservable = fromEvent(document, 'mousemove');

// Calculate the distance moved using the pairwise operator
mouseMoveObservable.pipe(
  map((event: MouseEvent) => ({ x: event.clientX, y: event.clientY }
  pairwise()
).subscribe(([previous, current]) => {
  const distance = Math.sqrt(
    Math.pow(current.x - previous.x, 2) +
    Math.pow(current.y - previous.y, 2)
  );
  console.log('Distance moved:', distance);
});
```

Paradigm 2: Procedural

Implementation:

- Procedural programming is a programming paradigm containing a series of computational steps to be carried out.
- Any given procedure might be called at any point during a program's execution, including by other procedures or itself.

Concepts:

Scoping:

Variables often have limited scope within procedures.

Variables defined within a procedure are only accessible within that procedure, reducing the risk of side effects.

Sequential:

Code executes in a predefined order. The flows of control is determined by the order in which the statements are written.

It makes it logical and easier to understamd.

Functions:

Encapsulates a sequence of instructions wjich can be called and executed in a step by step way.

Modularization:

Breaking down the program into smaller, manageable procedures.

It make the code organized and maintainabe.

Language for Paradigm 2: PL/SQL

PL/SQL stands for “Procedural Language extensions to the Structured Query Language.

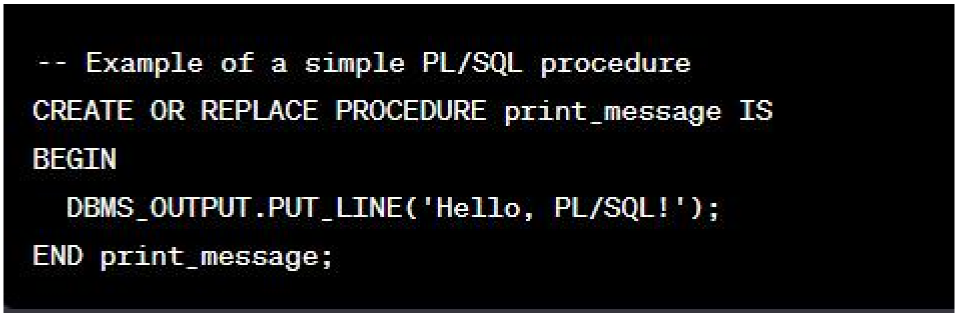
SQL is the language for both querying and updating relational databases.

PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements.

Revolves around four fundamental concepts: **Stored procedures, Cursors, Exception Handling** and **Triggers**.

Stored procedures:

Named blocks of code that can be called. It provides code reusability, modularity, and encapsulation.

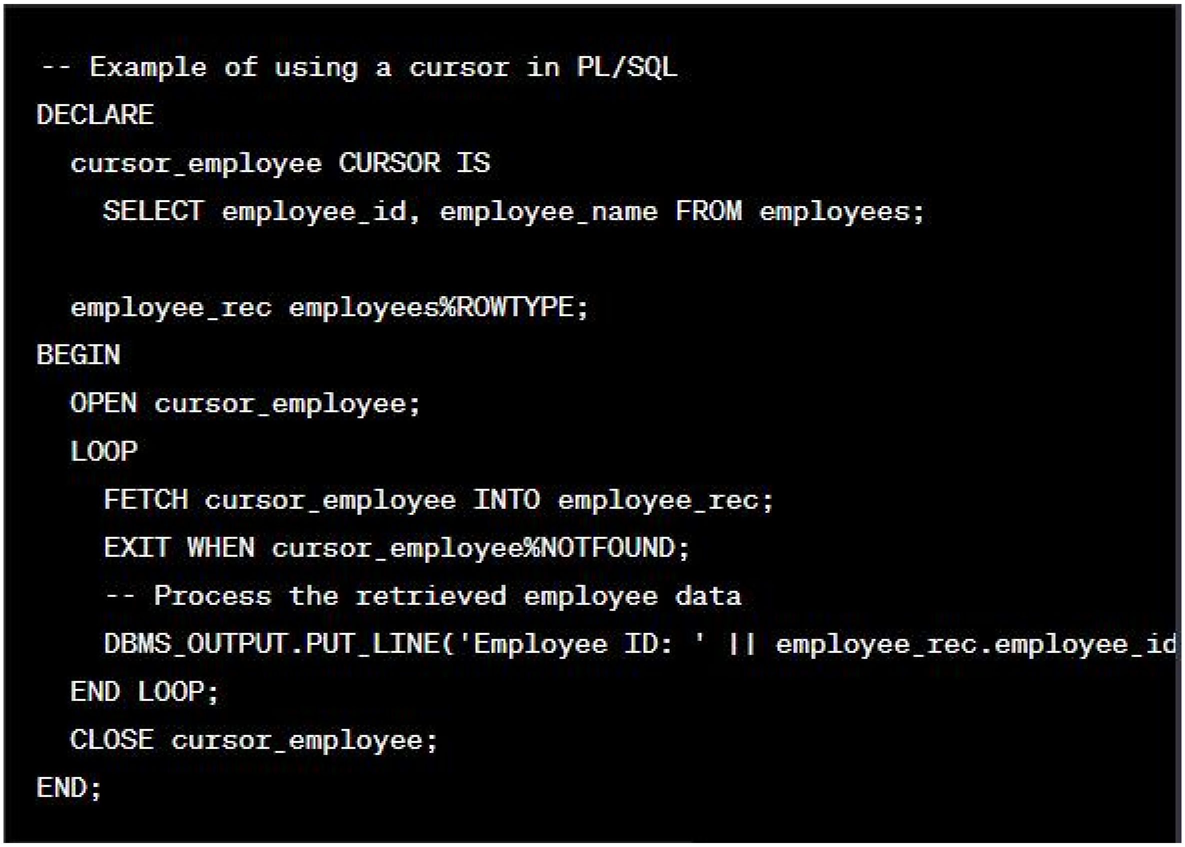


```
-- Example of a simple PL/SQL procedure
CREATE OR REPLACE PROCEDURE print_message IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('Hello, PL/SQL!');
END print_message;
```

Cursors:

Named control structure used by an application program to point to and select a row of data from a result set.

Needed to go through the data retrieved from the db.

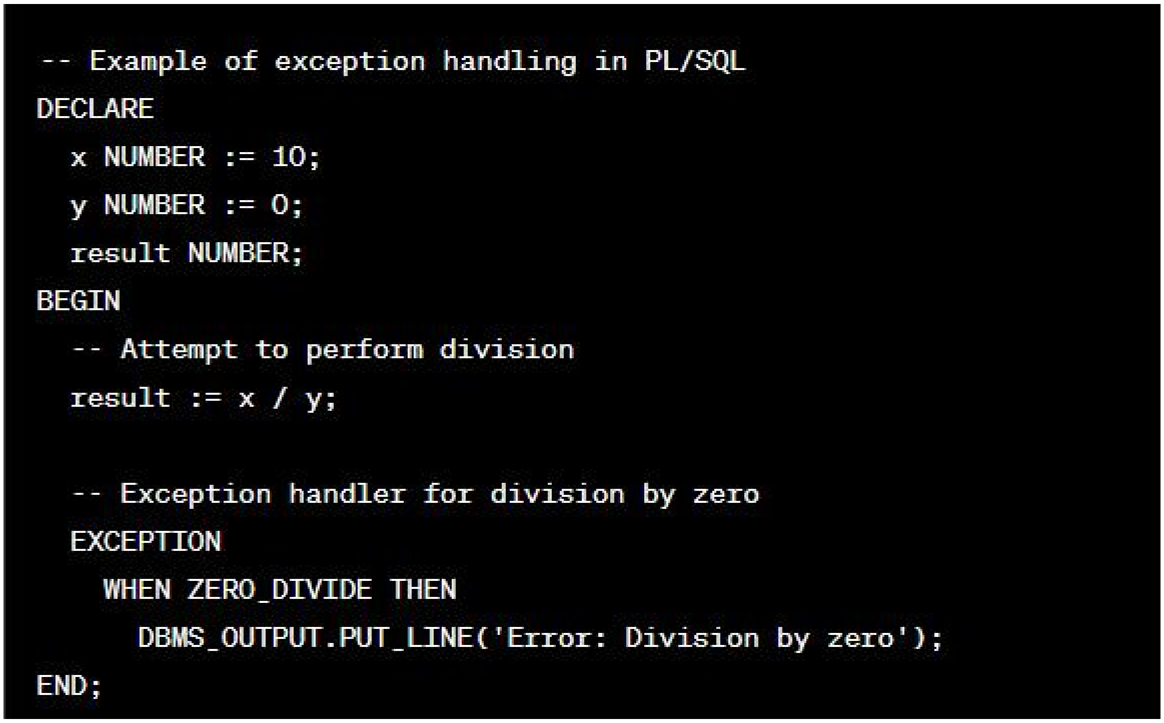


```
-- Example of using a cursor in PL/SQL
DECLARE
  cursor_employee CURSOR IS
    SELECT employee_id, employee_name FROM employees;

  employee_rec employees%ROWTYPE;
BEGIN
  OPEN cursor_employee;
  LOOP
    FETCH cursor_employee INTO employee_rec;
    EXIT WHEN cursor_employee%NOTFOUND;
    -- Process the retrieved employee data
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || employee_rec.employee_id
  END LOOP;
  CLOSE cursor_employee;
END;
```

Exception Handling:

Process of responding to unwanted or unexpected events when a computer program runs.

Any error encountered in a plsql program stops its execution so we can trap and recover from it by using an EXCEPTION.

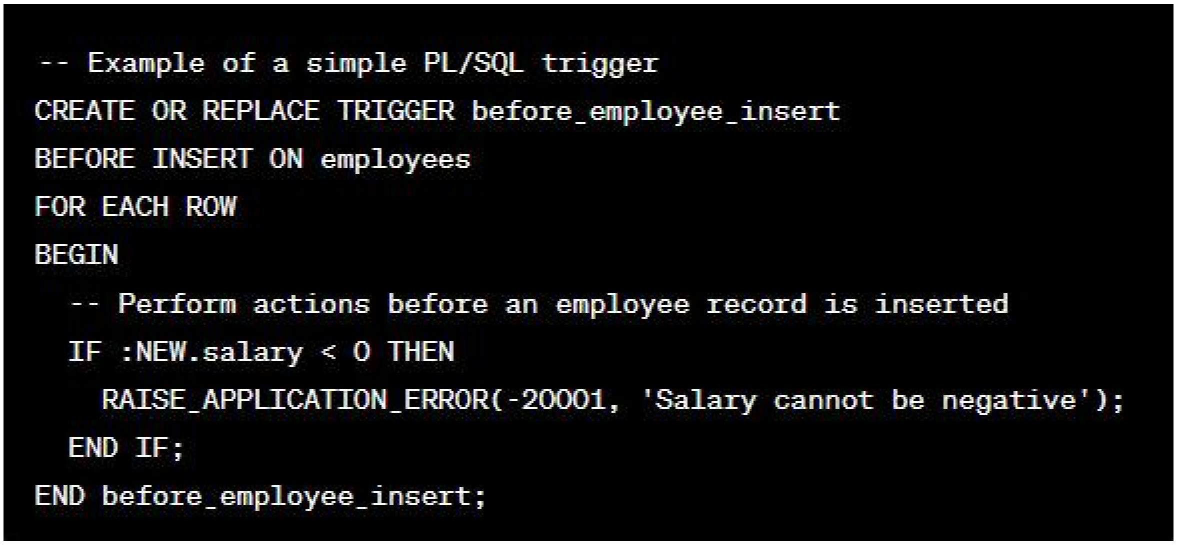


```
-- Example of exception handling in PL/SQL
DECLARE
  x NUMBER := 10;
  y NUMBER := 0;
  result NUMBER;
BEGIN
  -- Attempt to perform division
  result := x / y;

  -- Exception handler for division by zero
  EXCEPTION
    WHEN ZERO_DIVIDE THEN
      DBMS_OUTPUT.PUT_LINE('Error: Division by zero');
END;
```

Triggers:

Blocks of code that automatically execute when data is modifed like inserting deleting or updating.

Triggers are vital for enforcing business rules and maintaining data integrity.



```
-- Example of a simple PL/SQL trigger
CREATE OR REPLACE TRIGGER before_employee_insert
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
  -- Perform actions before an employee record is inserted
  IF :NEW.salary < 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Salary cannot be negative');
  END IF;
END before_employee_insert;
```

Analysis

Event-Driven(RxJS):

Strengths:

- Handles asynchronous operations by providing a reactive model as quick responses are crittical in many cases.
- Observables give better modularity which ensurres code organization and maintenance.
- RxJS being wide spread provides great community support and environment.
- It is also integrated with a popular framework like Angular.

Weaknesses:

- The asynchronous nature of event-driven applications can increase the software's complexity
- With the distributed nature of applications, it can be hard to trace an event from source to destination resulting in testing and debugging problems.

Procedural(PL/SQL):

Strengths:

- Efficient for tasks involving data manipulation and complex business logic.
- Clear step by st4ep exequtíon give clarity to follow thw logic.

Weaknesses:

- Inabitlity to reuse the code throughout the program.and to rewrite the same type of code many times.
- There is no data security, data is exposed and accessible to multiple procedures.

Comparison

Similarity:

They both structure their code into procedures, functions, or observables.

They both support concurrent tasks, eventdriven through reactive constructs and procedural through threads.

Differences:

In event driven the control flow is dynamically changed according to external events. Whereas procedural follows a predefined sequence.

Event driven is suited when the response is crucial. Whereas procedural is suited where step by step procedure is followed.

Challenges Faced

- Found it hard to grasp the concept of Observables
- Handling asynchronous workflows was complicated.
- Pl/SQL didn’t bother much but RxJS consumed a lot of time to take in.

Conclusion

Event-Driven (RxJS):

The reactive nature of RxJS and the use of observables offer a way to handle asynchronous operations. RxJSs graeat adoption in web development ensures support.

Procedural (PL/SQL):

PL/SQL, as a procedural language, excels in tasks involving data manipulation, business logic, and transactional processing. Its step by step execution flow makes it suitable for database dependent projects.

References

https://en.wikipedia.org

https://www.simplilearn.com

https://kinsta.com

https://www.cubesoftware.com

https://www.oracle.com

https://chat.openai.com