함수형 프로그래밍(FP)과 함수형 반응형 프로그래밍(FRP)

#FP #FRP #Imperative_programming

함수형 반응형 프로그래밍 (FRP) 개요

FRP는 함수형 프로그래밍(FP)과 반응형 프로그래밍(RP) 두 가지 프로그래밍 패러다임의 교차점에 위치합니다. 이 개념을 더 깊이 이해하기 전에 몇 가지 기본 용어를 알아보겠습니다.

명령형 프로그래밍

전통적으로, 문제를 해결하는 방법을 설명하는 코드를 작성합니다. 각 코드 줄이 순차적으로 실행되어 원하는 결과를 도출하는 방식이 **명령형 프로그래밍**입니다. 명령형 패러다임은 프로그래머가 프로그램이 특정 작업을 해결하는 방법을 "어떻게" 작성할 지를 강요합니다.

```
let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9];
let numbersLessThanFive = [];

for (let i = 0; i < numbers.length; i++) {
    if (numbers[i] < 5) {
        numbersLessThanFive.push(numbers[i]);
    }
}

console.log(numbersLessThanFive); // Output: [1, 2, 3, 4]
```

위 예제에서는 원하는 출력을 생성하기 위해 일련의 명령을 순차적으로 실행합니다.

함수형 프로그래밍 (FP)

함수형 프로그래밍은 상태 변경과 데이터 변이를 피하면서 모든 것을 함수의 결과로 모델링하는 프로그래밍 패러다임입니다. 여기서 상태(state)는 프로그램 실행 중의 다양한 조합과 경우의 수를 나타내며, 데이터 변이(mutability)는 주어진 데이터셋이 프로그램 실행 중에 변할 수 있는 개념을 의미합니다.

```
let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9];
// 고차 함수를 이용하여, 로직을 처리.
let numbersLessThanFive = numbers.filter(num => num < 5);
/*

num => num < 5 에 집중.
이 함수는 불변성을 유지하는 순수함수로서, 5보다 큰지 작은 지를 판별하는데만 집중.

*/

console.log(numbersLessThanFive); // Output: [1, 2, 3, 4]
```

위 예제에서는 `filter` 함수를 사용하여 조건에 맞는 요소를 걸러내는 방식으로, `numbersLessThanFive` 배열은 `const`로 선언되어 불변성을 유지합니다.

함수형 프로그래밍 주요 개념

- **상태 변경과 데이터 변이 피하기**: 상태 변경과 데이터 변이를 피하여 예측 가능한 코드를 작성합니다.
- **순수 함수(Pure Functions)**: 같은 입력에 대해 항상 같은 출력을 반환하며, 외부 상태를 변경하지 않는 함수입니다.
- **불변성(Immutability)**: 데이터가 변경되지 않으며, 변경이 필요한 경우 새로운 데이터를 생성합니다.

반응형 프로그래밍 (Reactive Programming)

반응형 프로그래밍은 데이터 스트림과 변화의 전파를 다루는 선언형 프로그래밍 패러다임입니다. 이 패러다임을 통해 정적(예: 배열) 또는 동적(예: 이벤트 발생기) 데이터 스트림을 쉽게 표현할 수 있으며, 관련 실행 모델 내에서 연관된 종속성을 자동으로 전파할 수 있습니다.

반응형 프로그래밍에서는 데이터의 변경이 자동으로 전파된다는 것은 어떤 이야기일까요?

예를 들어, 명령형 프로그래밍에서 `a := b + c`는 `a`가 `b + c`의 결과로 즉시 할당된다는 것을 의미하며, 이후에 `b`와 `c`의 값이 변경되더라도 `a`의 값은 영향을 받지 않습니다. 하지만 반응형 프로그래밍에서는 `a`가 `b`나 `c`의 값이 변경될 때마다 자동으로 업데이트됩니다.

```
var b = 1;
var c = 2;
var a = b + c; b = 10;

console.log(a); // 3 (명령형 프로그래밍에서는 a의 값이 12가 아닌 3으로 유지됩니다.)
```

```
/*
이제 특별한 연산자 "$="가 있다고 가정해보겠습니다. 이 연산자는 초기화될 때뿐만 아니라 참조된 변수(b와 c)가 변경될 때도 자동으로 값을 변경합니다. */

var b = 1;
var c = 2;
var a $= b + c; b = 10;

console.log(a); // 12 (반응형 프로그래밍에서는 a의 값이 자동으로 업데이트됩니다.)`
```

반응형 프로그래밍의 주요 개념

1. **비동기 및 비차단(Asynchronous and Non-blocking)**:
   - 반응형 프로그래밍에서는 작업이 비동기적으로 실행되어 이전 작업이 완료될 때까지 기다릴 필요가 없습니다. 이는 대규모 동시 요청을 처리할 수 있게 하여 자원 활용을 최적화하고 성능을 향상시킵니다.
2. **이벤트 기반 아키텍처(Event-driven Architecture)**:
   - 반응형 시스템에서는 구성 요소들이 이벤트를 발생시키고 이에 반응하여 통신합니다. 이벤트는 사용자 입력, 데이터 변경, 시스템 알림 등 다양한 형태로 나타날 수 있습니다. 이러한 이벤트 기반 아키텍처는 느슨하게 결합되고 확장성이 뛰어난 시스템을 가능하게 합니다.
3. **반응형 스트림(Reactive Streams)**:
   - 반응형 프로그래밍은 종종 비동기적으로 처리되고 반응적으로 반응하는 데이터 항목의 시퀀스를 의미하는 반응형 스트림 개념을 사용합니다.
   - 반응형 스트림은 백프레셔(Backpressure) 처리 메커니즘을 제공하여, 소비자가 프로듀서로부터 데이터를 소비하는 속도를 제어하고 과부하와 자원 고갈을 방지합니다.

FRP의 활용

1. **사용자 인터페이스 (UI)**:
   - 사용자 입력, 애니메이션, 데이터 바인딩 등에서 실시간 반응성을 제공할 수 있습니다. 예를 들어, 사용자가 버튼을 클릭하면 즉시 UI가 업데이트되는 방식으로 동작할 수 있습니다.
2. **게임 개발**:
   - 게임 내에서 실시간으로 발생하는 이벤트와 상태 변화를 처리하는 데 유용합니다. 캐릭터의 움직임, 충돌 감지, 점수 업데이트 등을 실시간으로 반영할 수 있습니다.
3. **실시간 시스템**:
   - 주식 거래 시스템, 실시간 대시보드, 센서 데이터 처리 등 실시간으로 데이터 변화를 처리해야 하는 시스템에 적합합니다.

예제 (RxJS 사용)

반응형 프로그래밍을 실제로 적용하는 예로는 RxJS를 사용한 코드를 들 수 있습니다.

HTML 버튼 클릭 이벤트 스트림:

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

// 버튼 클릭 스트림 생성
const button = document.getElementById('myButton');
const clickStream = fromEvent(button, 'click');

// 클릭 이벤트를 스트림으로 처리
clickStream.pipe(
    map(event => event.clientX), // 클릭 이벤트에서 X 좌표 추출
    filter(x => x < 500) // X 좌표가 500 미만인 이벤트만 통과
).subscribe(x => console.log(`Clicked at: ${x}`));
```

위의 예제는 사용자가 버튼을 클릭할 때마다 비동기적으로 이벤트를 처리하며, 스트림을 통해 데이터를 변환하고 필터링하는 과정을 보여줍니다.

FRP의 이점

- **확장성**: 비동기 및 이벤트 기반 특성 덕분에 애플리케이션이 쉽게 확장됩니다.
- **반응성**: 실시간 피드백과 원활한 사용자 경험을 제공할 수 있습니다.
- **탄력성**: 비동기 및 비차단 I/O를 수용하여 더 높은 수준의 장애 대응 능력을 갖춥니다.

FRP의 단점

- **학습 곡선**: 명령형 프로그래밍에 익숙한 개발자에게는 비동기 및 이벤트 기반 프로그래밍을 이해하는 데 시간이 걸릴 수 있습니다.
- **복잡성**: 비동기 작업과 반응형 스트림을 효과적으로 관리하는 것은 복잡성을 증가시킬 수 있습니다.
- **디버깅 어려움**: 비동기 및 이벤트 기반 코드를 디버깅하는 것은 동기 코드보다 더 어려울 수 있습니다.

References & Terms

- Reactive programming - WikipediaChapter 1: Introduction to Reactive Programming | by Suman Maity | May, 2024 | Medium
- A quick introduction to Functional Reactive Programming (FRP) (freecodecamp.org)
- Chapter 1: Introduction to Reactive Programming | by Suman Maity | May, 2024 | Medium
- RxJS
