코드 수선 모임

12월 4일(목)

선언적 프로그래밍

짝수만 필터링, 각 짝수의 제곱을 구한 후, 그 합계를 계산하는 예제

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let sumOfSquares = 0;

for (let i = 0; i < numbers.length; i++) {
    if (numbers[i] % 2 === 0) {
        const square = numbers[i] * numbers[i];
        sumOfSquares += square;
    }
}

```

명령형

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

const isEven = (num: number): boolean => num % 2 === 0;

const square = (num: number): number => num * num;

const sum = (acc: number, current: number): number => acc + current;

const sumOfSquares = numbers
    .filter(isEven)  // 짝수만 필터링
    .map(square)     // 제곱을 구함
    .reduce(sum, 0); // 합계를 계산

```

선언형 (함수형 프로그래밍으로 구현..)

선언적 프로그래밍

선언형 프로그래밍(Declarative Programming)은 "어떻게(How)"가 아니라 **"무엇(What)"**을 기술하는 방식입니다.
즉, **프로그램이 수행해야 할 목적과 결과를 중심으로 코드를 작성**하고, 그 목적을 달성하기 위한 내부 실행 로직은 추상화하거나 위임합니다.

같이 읽어볼 내용

선언적 프로그래밍에 대한 착각과 오해

문법이 아닌 사고방식의 전환이 만드는 진정한 선언적 코드

함수를 사용하면 선언적이다 (X)

React 등 선언적인 도구를 사용하면 선언적이다 (X)

map, filter, reduce등 고차 함수를 사용하면 선언적이다 (X)

선언적인 사고방식을 사용해야 선언적이다 (O)

선언이라는 개념

“물 2컵을 끓인다, 면을 넣고 3분간 끓인다, 스프를 넣고 1분간 더 끓인다, 그릇에 담는다”라는 **시간의 흐름**에 따른 행동 지침

“라면 = 삶은 면 + 스프 + 뜨거운 물의 조합”이라는 재료들 사이의 **관계**

$$f(x)=2x+1$$

“x에 2를 곱하고 1을 더하라”는 명령이 아니다. 이는 ” x와 $f(x)$ 사이에는 이런 관계가 있다”는 것

선언적인 방식으로 작성된 것들

```jsx
function UserProfile({ user }) {
  return (
    <div className="user-profile">
      <h2>{user.name}</h2>
      <p>{user.email}</p>
      {user.avatar && (
        <img src={user.avatar} alt={`${user.name}의 아바타`} />
      )}
    </div>
  );
}
```

jsx -> 데이터와 UI구조 사이 관계만을 표현 (user.name을 h2에 붙이는 코드가 없음)

선언적과 절차적은 상대적

A. 비즈니스, 프레젠테이션레벨(개별 컴포넌트, 페이지 컴포넌트),
B. 각 단계가 독립적으로 정의되고 테스트 가능할 때

```
// 변환 관계를 명확히 선언
const processUserData = (rawUsers: RawUser[]) =>
  rawUsers
    .filter(isActiveUser)
    .map(normalizeUserData)
    .map(addComputedFields)
    .sort(byLastLoginDate);
```

tsx

```tsx
// 좋은 예: 명확한 레벨 분리

// 비즈니스 레벨 - 선언적
const ShoppingCart = ({ items, onCheckout }) => (
  <div className="shopping-cart">
    <ItemList items={items} />
    <TotalPrice items={items} />
    <CheckoutButton onClick={() => onCheckout(items)} />
  </div>
);

// 프레젠테이션 레벨 - 선언적
const ItemList = ({ items }) => (
  <ul className="item-list">
    {items.map(item => (
      <ItemCard key={item.id} item={item} />
    ))}
  </ul>
);

// 인프라 레벨 - 절차적이어도 OK
function calculateTotalWithTax(items, taxRate) {
  let subtotal = 0;

  for (const item of items) {
    subtotal += item.price * item.quantity;
  }

  const tax = subtotal * taxRate;
  return subtotal + tax;
}
```

인프라 레벨과 성능 최적화에서는 절차적 접근이 적절

선언적인 것 같은 코드 vs. 선언적인 코드