Clean Code

3장 함수

2021-09-30 이호찬

함수

함수: 어떤 프로그램이든 가장 기본적인 단위

- 함수가 읽기 쉽고 이해하기 쉬운 이유는 무엇일까?
- 의도를 분명히 표현하는 함수를 어떻게 구현할 수 있을까?
- 함수에 어떤 속성을 부여해야 처음 읽는 사람이 프로그램 내부를 직관적으로 파악할 수 있을까?

작게 만들어라!

함수는 읽고 이해하기 쉬워야한다.

블록과 들여쓰기: 함수에서 들여쓰기 수준은 1단이나 2단을 넘어서면 안된다.

```
42  const dropAppIconButtons = (event: MouseEvent): void => {
43    const draggingTarget = $('.dragging') as HTMLButtonElement;
44    const eventTarget = event.target as HTMLElement;
45    event.preventDefault();
46
47    if (draggingTarget === eventTarget) {
48      return;
49    }
50
51    if (eventTarget.classList.contains('draggable')) {
52      const draggedDragzone = draggingTarget?.parentNode as HTMLDivElement;
53      const eventTargetDragzone = eventTarget?.parentNode as HTMLDivElement;
54
55      model.switchAppDataOrder(eventTarget.innerText, draggingTarget.innerText);
56
57      draggedDragzone.appendChild(eventTarget);
58      eventTargetDragzone.appendChild(draggingTarget);
59      eventTarget.classList.toggle('dragenter', false);
60    } else if (eventTarget.classList.contains('dragzone')) {
61      const draggedDragzone = draggingTarget?.parentNode as HTMLDivElement;
62
63      if (eventTarget?.children[0]) {
64        model.switchAppDataOrder((eventTarget.children[0] as HTMLButtonElement).innerText, draggingTarget.innerText);
65
66        draggedDragzone.appendChild(eventTarget.children[0]);
67        eventTarget.appendChild(draggingTarget);
68        eventTarget.children[0].classList.toggle('dragenter', false);
69      } else {
70        throw Error(HOME_DRAG_ERROR);
71      }
72    }
73  };
```

Home-page.ts

```
const dropAppIconButtons = (event: MouseEvent): void => {
  event.preventDefault();

  if (checkDraggingTarget()) {
    return;
  }

  if (checkIconDraggable()) {
    dropIconDraggableSpace();
  } else if (checkIconDragzone()) {
    dropIconDragzoneSpace();
  } else {
    throw Error(HOME_DRAG_ERROR);
  }
}
```

Home-page.ts (Re-refactored)

작게 만들어라!

함수는 읽고 이해하기 쉬워야한다.

블록과 들여쓰기: 함수에서 들여쓰기 수준은 1단이나 2단을 넘어서면 안된다.

```
42  const dropAppIconButtons = (event: MouseEvent): void => {
43    const draggingTarget = $('.dragging') as HTMLButtonElement;
44    const eventTarget = event.target as HTMLElement;
45    event.preventDefault();
46
47    if (draggingTarget === eventTarget) {
48      return;
49    }
50
51    if (eventTarget.classList.contains('draggable')) {
52      const draggedDragzone = draggingTarget?.parentNode as HTMLDivElement;
53      const eventTargetDragzone = eventTarget?.parentNode as HTMLDivElement;
54
55      model.switchAppDataOrder(eventTarget.innerText, draggingTarget.innerText);
56
57      draggedDragzone.appendChild(eventTarget);
58      eventTargetDragzone.appendChild(draggingTarget);
59      eventTarget.classList.toggle('dragenter', false);
60    } else if (eventTarget.classList.contains('dragzone')) {
61      const draggedDragzone = draggingTarget?.parentNode as HTMLDivElement;
62
63      if (eventTarget?.children[0]) {
64        model.switchAppDataOrder((eventTarget.children[0] as HTMLButtonElement).innerText, draggingTarget.innerText);
65
66        draggedDragzone.appendChild(eventTarget.children[0]);
67        eventTarget.appendChild(draggingTarget);
68        eventTarget.children[0].classList.toggle('dragenter', false);
69      } else {
70        throw Error(HOME_DRAG_ERROR);
71      }
72    }
73  };
```

```
const dropAppIconButtons = (event: MouseEvent): void => {
  event.preventDefault();

  if (checkDraggingTarget()) {
    return;
  }

  if (checkIconDraggable()) {
    dropIconDraggableSpace();
  } else if (checkIconDragzone()) {
    dropIconDragzoneSpace();
  } else {
    throw Error(HOME_DRAG_ERROR);
  }
}
```

한가지만 해라!

- 함수는 한 가지를 해야한다.
- 의미 있는 이름으로 다른 함수를 추출할 수 있다면 그 함수는 여러가지 작업을 하고 있다는 뜻.

```
const updateNavigationTime = () => {
    setInterval(() => {
        const navTimeElement = $('.nav__time') as HTMLSpanElement;
        const currentDate = getCurrentDate();
        notifyAlarm(currentDate);
        navTimeElement.innerText = currentDate;
    }, 1000);
};

const initController = (): void => {
    updateNavigationTime();
    window.addEventListener('popstate', () => historyRouter(window.location.pathname));
    homePageController();
};

export default initController;
```

한가지만 해라!

```
const updateNavigationTime = () => {
    setInterval(() => {
        const navTimeElement = $('.nav__time') as HTMLSpanElement;
        const currentDate = getCurrentDate();
        notifyAlarm(currentDate);
        navTimeElement.innerText = currentDate;
    }, 1000);
};
```

```
const updateNavigationTime = () => {
    const navTimeElement = $('.nav__time') as HTMLSpanElement;
    const currentDate = getCurrentDate();
    navTimeElement.innerText = currentDate;
}

const navigationController = () => {
    setInterval(() => {
        updateNavigationTime();
        notifyAlarm();
    }, 1000);
};
```

함수 당 추상화 수준은 하나로

- 근본 개념과 세부사항을 뒤섞기 시작하면, 사람들이 함수에 세부사항을 점점 더 추가한다.
- 내려가기 규칙: 한 함수 다음에는 추상화 수준이 한 단계 낮은 함수가 온다.

```
const initController = (): void => {
    updateNavigationTime();
    window.addEventListener('popstate', () => historyRouter(window.location.pathname));
    homePageController();
};

export default initController;
```

```
const initController = (): void => {
    updateNavigationTime();
    historyController();
    homePageController();
};

export default initController;
```

Switch 문

Switch 문의 문제점

1. 함수가 길다.
2. '한 가지' 작업만 수행하지 않는다.
3. SRP(Single Responsibility Principle)를 위반한다.
4. OCP(Open Closed Principle)를 위반한다.
   - 새 유형을 추가할 때마다 코드를 변경한다.
5. 동일한 구조의 함수가 여러개 존재할 수 있다.

```
const createCar = (model) => {
   switch (model) {
      case 'Cayman':
         return createCaymanCar();
      case 'Boxster':
         return createBoxsterCar();
      case 'Panamera':
         return createPanameraCar();
      default:
         return createPanameraCar();
   }
}
```

ABSTRACT FACTORY로 해결

```
// Abstract Factory
class CarFactory {
  // Factory Method
  createCar(model) {
    let car = null;
    switch(model) {
      case 'Cayman':
        car = new Cayman();
        break;
      case 'Boxster':
        car = new Boxster();
        break;
      case 'Panamera':
        car = new Panamera();
        break;
      default:
        car = new Cayman();
        break;
    }
    if(typeof car.printModel === 'undefined') {
      car.printModel = () => {
        console.log(`This car model is: ${car.model}`);
      }
    }
    return car;
  }
}
```

```
// Car를 상속받는 Caymen, Boxster, Panamera 팩토리
// 각 팩토리에 맞게 오버라이딩하여 재구현
class Car {
  constructor(name) {
    this.model = name;
  }
  // Abstract Product
  createDoor() {}
  createHood() {}
}
```

```
// Concrete Factory
class Cayman extends Car{
  constructor() {
    super('Cayman');
  }
  // Concrete Product
  createDoor(side) {
    return new CaymanDoor(side, this.constructor.name);
  }
  // Concrete Product
  createHood() {
    return new CaymanHood(this.constructor.name);
  }
}

// Concrete Factory
class Boxster extends Car{
  constructor() {
    super('Boxster');
  }
  // Concrete Product
  createDoor(side) {
    return new BoxsterDoor(side, this.constructor.name);
  }
  // Concrete Product
  createHood() {
    return new BoxsterHood(this.constructor.name);
  }
}

// Concrete Factory
class Panamera extends Car{
  constructor() {
    super('Panamera');
  }
  // Concrete Product
  createDoor(side) {
    return new PanameraDoor(side, this.constructor.name);
  }
  // Concrete Product
  createHood() {
    return new PanameraHood(this.constructor.name);
  }
}
```

```
const factory = new CarFactory();

// Client
const boxsterCar = factory.createCar('Boxster');
```

서술적인 이름을 사용하라

- 코드를 읽으면서 짐작했던 기능을 각 루틴이 그대로 수행한다면 깨끗한 코드이다.

```
const updateNavigationTime = () => {
    setInterval(() => {
        const navTimeElement = $('.nav__time') as HTMLSpanElement;
        const currentDate = getCurrentDate();
        notifyAlarm(currentDate);
        navTimeElement.innerText = currentDate;
    }, 1000);
};
```

```
const navigationController = () => {
    setInterval(() => {
        updateNavigationTime();
        notifyAlarm();
    }, 1000);
};
```

함수 인수

- 함수에서 이상적인 인수 개수는 0개(무항)다.
- 다음은 1개(단항)고, 다음은 2개(이항)다.
- 3개(삼항)는 가능한 피하는 편이 좋다.
- 플래그 인수는 피하자.
- 또한, 테스트 관점에서 보면 인수는 더 어렵다.

함수 인수 (단항)

1. 함수에 인수 1개는 넘기는 best practice

- 인수에 질문을 던지는 경우: Boolean fileExists(“MyFile”)
- 인수를 뭔가로 변환해 결과를 반환하는 경우: InputStream fileOpen(“MyFile”)
- 출력 인수가 없고, 입력 인수만 존재하는 함수 형식이 이벤트인 경우: passwordAttemptFailedNtimes(int attempts)

2. 함수에 인수가 2개 이상일 경우

- actual 다음에 expected가 온다는 순서를 인위적으로 기억해야한다.

```
assert.equal(actual, expected, [message]);
```

```
it('padding 0 제거', () => {
    for (let i = 0; i < 100; i++) {
        assert.equal(atoi('0'.repeat(i) + '123456789hello'), 123456789);
    }
});
```

함수 인수

3. 함수에 인수가 2~3개 필요하다면 일부를 독자적인 클래스 변수로 선언할 가능성을 짚어보아야한다.

```
Circle makeCircle(double x, double y, double radius);
Circle makeCircle(Point center, double radius);
```

4. 가변 인수

```
void monad(Integer… args);
void dyad(String name, Integer… args);
void triad(String name, int count, Integer… args);
```

```
void monad(…args);
void dyad(name, …args);
void triad(name, count, …args);
```

5. 동사 키워드

write(name) 대신 writeField(name)를 사용할 경우 ‘이름name’이 ‘필드field’라는 사실이 분명히 드러난다.

부수 효과를 일으키지 마라.

```
7  const NotifyAlarm = (currentDate: string): void => {
8    const currentHour = atoi(currentDate.split(' ')[3] as string) as string;
9    const currentMinute = atoi(currentDate.split(' ')[4] as string) as string;
10   const currentSecond = atoi(currentDate.split(' ')[5] as string) as string;
11
12   if (currentSecond !== '0') {
13     return;
14   }
15   const alarmData = model.getLocalStorageAlarmData('alarmData');
16   alarmData?.forEach(({ meridiem, hour, minute }): void => {
17     if (
18       (meridiem === 'pm' || meridiem === 'am') &&
19       hour === (meridiem === 'pm' ? String(+currentHour - 12) : currentHour) &&
20       minute === currentMinute
21     ) {
22       alert(`[알림] ${meridiem === 'pm' ? '오후' : '오전'} ${currentHour}시 ${currentMinute}분입니다.`);
23       model.removeLocalStorageAlarmData('alarmData', meridiem, hour, minute);
24       renderAlarmList();
25     }
26   });
27 };
28
29 const updateNavigationTime = () => {
30   setInterval(() => {
31     const navTimeElement = $('.nav__time') as HTMLSpanElement;
32     const currentDate = getCurrentDate();
33     NotifyAlarm(currentDate);
34     navTimeElement.innerText = currentDate;
35   }, 1000);
36 };
```

이름만 봐서는 updateNavigationTime 함수가 알람 시간을 공지해준다는 사실이 드러나지 않는다.

따라서 NotifyAlarm 에 대한 side effect가 발생할 위험에 처할 수 있다.

updateNavigationTime보다는 updateNavigationTimeAndNotifyAlarm이 더 나은 선택지다.

앞에서는 navigationController로 사용했다.

부수 효과를 일으키지 마라.

- 함수에서 상태를 변경해야 한다면 함수가 속한 객체 상태를 변경하는 방식을 택한다.

```
void appendFooter(report: object);
```



```
report.appendFooter()
```

명령과 조회를 분리하라.

- 코드만 봐서는 애매모호한 함수를 명령과 조회를 분리시켜 혼란을 애초에 없앨 수 있다.
- If (set(…)){…} 코드를 “username 속성이 unclebob으로 설정되어있다면…”으로도 읽을 수 있다.

```
if (set(“username”, “unclebob”)) {
    …
}
```



```
if (attributeExists(“username”)) {
    setAttribute(“username”, “unclebob”);
    …
}
```

오류 코드보다 예외를 사용하라!

```
if (deletePage(page) === E_OK) {
   if (registry.deleteReference(page. Name) === E_OK) {
      if (configKeys.deleteKey(page.name.makeKey()) === E_OK) {
         logger.log(“page deleted”);
      } else {
         logger.log(“configKey not deleted”);
      }
   } else {
      logger.log(“deleteReference from registry failed”);
   }
} else {
   logger.log(“delete failed”);
   return E_ERROR;
}
```

```
try {
   deletePage(page);
   registry.deleteReference(page.name);
   configKeys.deleteKey(page.name.makeKey());
} catch {
   logger.log(e.getMessage());
}
```

Try/Catch 블록 뽑아내기

```
const delete = (page) => {
   try {
      deletePageAndAllReferrences(page);
   }
   catch (e) {
      logError(e);
   }
}

const deletePageAndAllReferences = (page) {
   deletePage(page);
   registry.deleteReference(page.name);
   configKeys.deleteKey(page.name.makeKey());
}

const logError = (e) => {
   logger.log(e.getMessage());
}
```

- try/catch 블록을 별도 함수로 뽑아내는 편이 좋다.
- 오류 처리도 한 가지 작업이다.

반복하지 마라!

```
const optionWrapper = ({ option, value }: { option: string | number; value: string | number }): string
=> {
   return `<option value="${value}">
               ${option}
            </option>`;
};

const selectWrapper = (selectId: string, options: string): string => {
   return `<select id="${selectId}" class="alarm__select">
               ${options}
            </select>`;
};
```

구조적 프로그래밍

데이크스트라
- 모든 함수와 함수 내 모든 블록에 입구와 출구가 하나만 존재해야한다.
- 즉, 함수는 return 문이 하나여야한다.

구조적 프로그래밍의 목표와 규율은 함수가 작을 경우 별다른 이익을 제공하지 못한다.
때로는 return, break, continue의 여러 차례 사용이 단일 입/출구 규칙보다 의도를 표현하기 쉬워진다.

그외의 goto문은 작은 함수에서는 피하자.

함수는 어떻게 짜죠?

1. 길고 복잡한 함수를 짠다.

2. 서투른 코드도 단위 테스트 케이스를 만든다.

3. 리펙토링 한다.
   - 코드를 다듬고, 함수를 만들고, 이름을 바꾸고, 중복을 제거한다. 메서드를 줄이고 순서도 바꾼다.

4. 이와중에도 코드는 항상 단위 테스트를 통과한다.

프로그래머는 시스템을 구현할 프로그램이아니라 풀어갈 이야기로 여긴다.

감사합니다.