go to site

하노이 타워

Tower of Hanoi

AI기반 챗봇(GPT) 활용한 웹개발(Python)전문가 과정 I 허기범

go to site

목차

Contents

go to site

01 프로젝트 개요

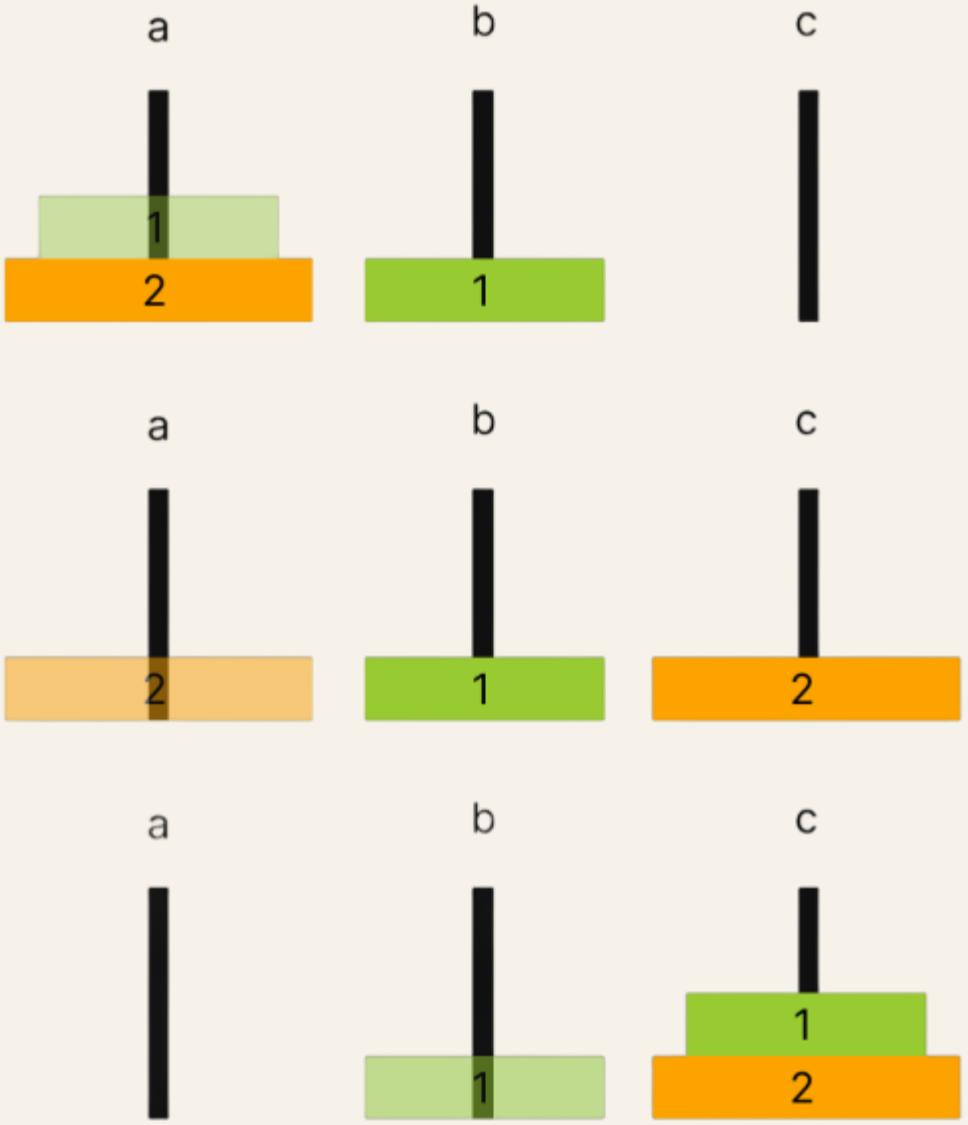
01 프로젝트 개요

프로젝트 소개

"하노이의 탑"

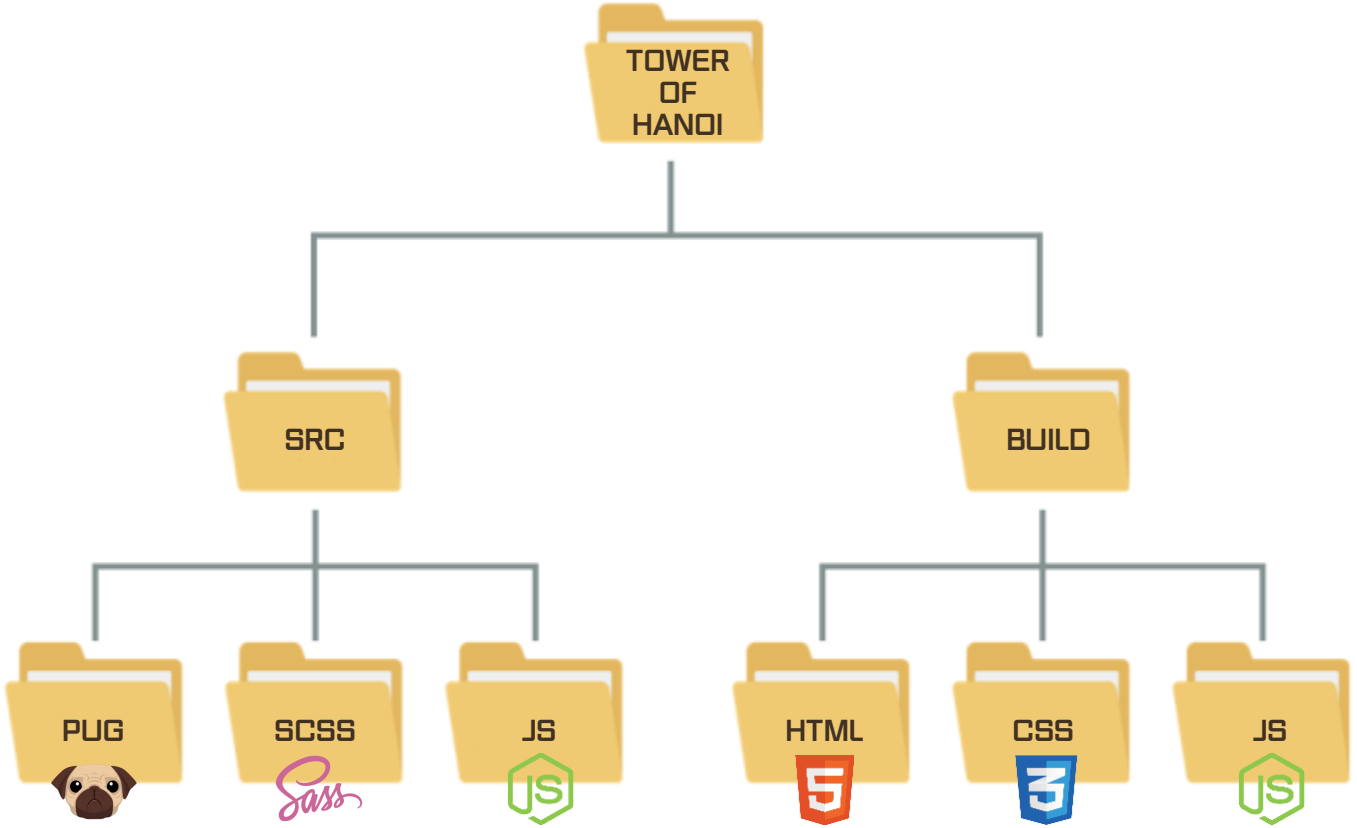
3개의 기둥이 있고, 가장 왼쪽 기둥에 쌓여진 원반들을 가장 오른쪽 기둥에 같은 순서대로 옮기는 게임.

단, 2가지의 규칙을 가지고 있다.
a. 한 번에 하나씩 이동해야 한다.
b. 작은 원반 위에 큰 원반을 올릴 수 없다.



01 프로젝트 개요

사용 언어 및 파일 구조



01 프로젝트 개요

개발 일정

DAY 01

패턴 분석 & JAVASCRIPT

- 하노이의 탑 패턴 (알고리즘) 분석
- 패턴 코드화작업 (JAVASCRIPT)

DAY 02

디자인 작업

- 스케치 작업
- PUG & SASS 학습 및 구성
- Animation 작업

DAY 03

JAVASCRIPT

- JAVASCRIPT 코드 구성
- 기능 추가 및 오류 해결

DAY 04

프로젝트 마무리

- 코드 정리 및 마무리 점검
- PPT 문서화 작업

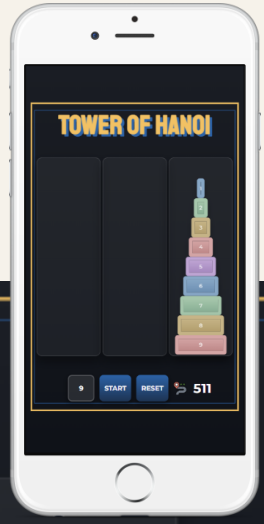
go to site

02 기획 및 설계

02 기획 및 설계

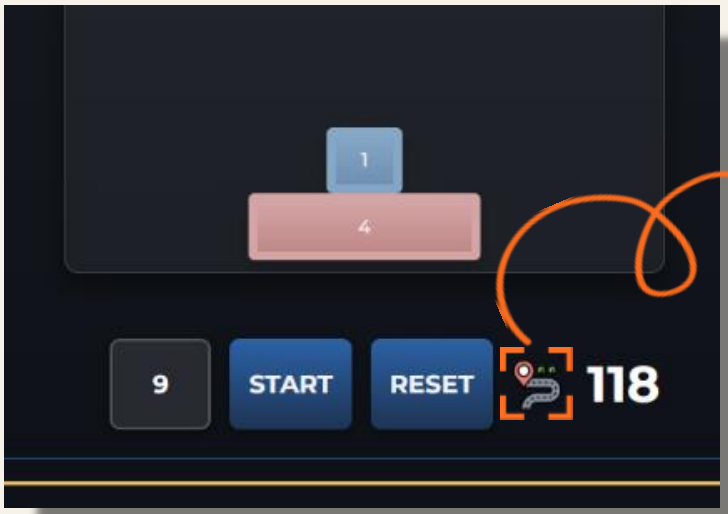
디자인 및 기능

모바일 반응형 디자인

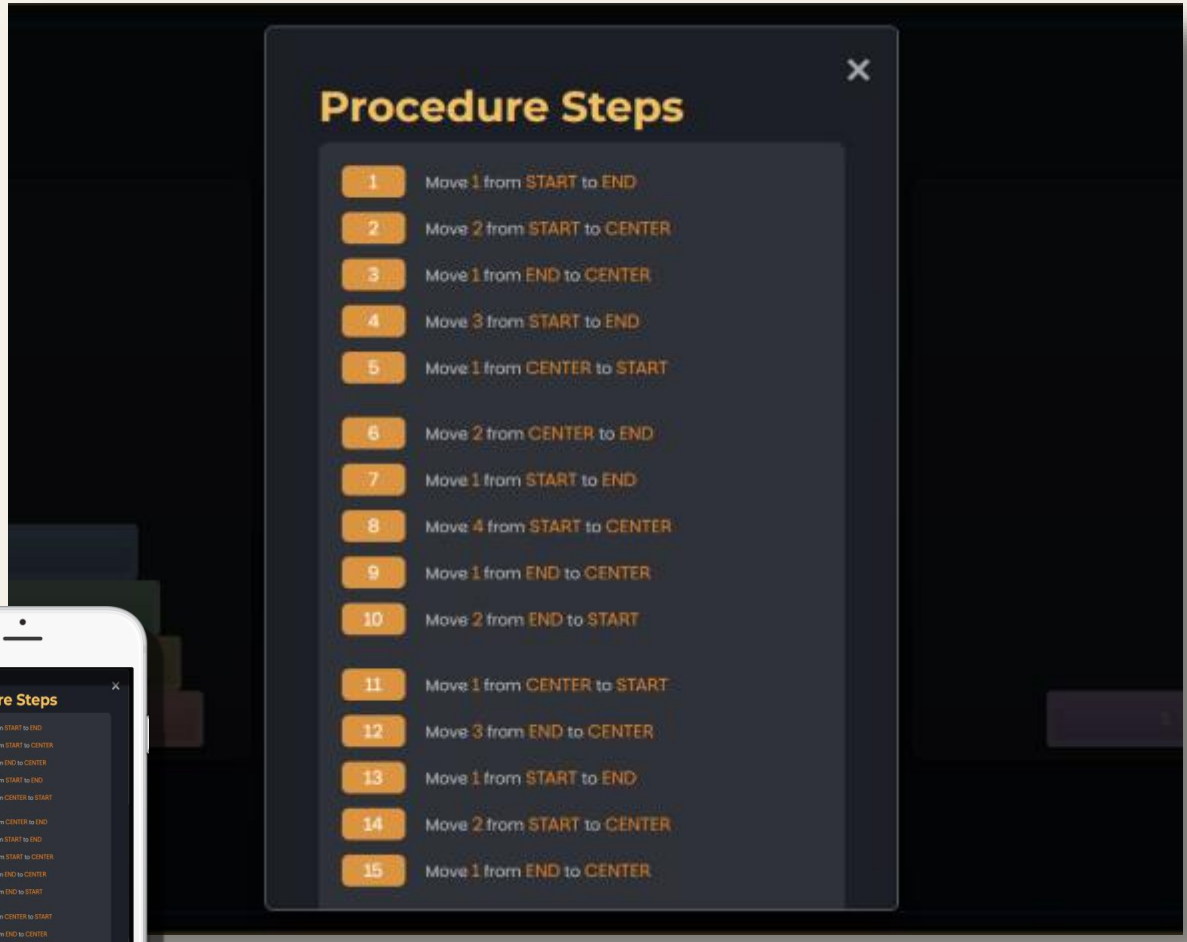


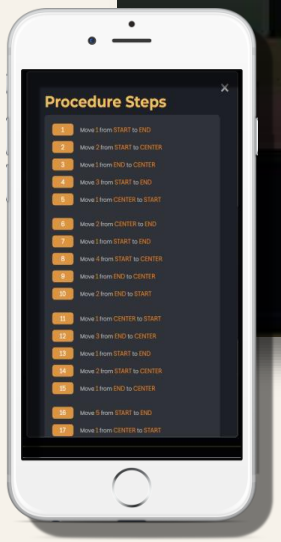
02 기획 및 설계

디자인 및 기능



하단의 해당 버튼 클릭 시,
모달창을 통해 전체 경로를
한 눈에 확인할 수 있게 함.





go to site

03 코드 분석

03 코드 분석

Pug

간결함, 가독성 향상을 위해 pug 활용.

mixin 문법 등을 활용하여 중복을 줄이고, 재사용 및 수정 등을 용이하게 함.

```pug
doctype html
html(lang="ko")
  head
    meta(charset="UTF-8")
    meta(name="viewport" content="width=device-width, initial-scale=1.0")
    title Tower of Hanoi
    link(href="https://fonts.googleapis.com/css2?family=Montserrat:ital")
    link(rel="icon", href="./images/lighthouse_1874219.png")
    link(rel="stylesheet", href="//cdn.jsdelivr.net/npm/xeicon@2.3.3/xeicon.min.css")
    link(rel="stylesheet", href="./tower_of_hanoi.css")
    script(src="./tower_of_hanoi.js")
  body
    .outerWrap
      .innerWrap
        h1.headerArea Tower of Hanoi
        .mainArea
          +towerBox('firstBox')
          +towerBox('secondBox')
          +towerBox('thirdBox')
        .footerArea
          input.discNum#discNum(type="number", min="1", max="10", placeholder="1 ~ 9")
          button.startBtn#startBtn START
          button.resetBtn#resetBtn RESET
          .navBtn#navBtn
          .countNum#countNum
        .modal#modal
          .modalContent
            span.closeBtn#closeBtnrtBtn#startBtn#sta
              i.xi-close-min
            h2 Procedure Steps
            .procedureSteps#procedureSteps

mixin towerBox(idName)
  .towerBox(id=idName)
```

03 코드 분석

Sass

```scss
$mainColorYellow : #F1C164;
$mainColorBlue : #2D63A7;
$mobile: 600px;
```

변수 기능 활용.

```scss
.outerWrap {
  width: 95%;
  height: 95%;
  (..생략)

  .innerWrap {
    width: 98%;
    height: 96%;
    (..생략)

    .headerArea {
      font: {
        family: 'Staatliches', cursive;
        size: 8rem;
      }
      text: {
        align: center;
        shadow: 0px 4px 6px rgba(0, 0, 0, 0.6);
        shadow: 5px 5px $mainColorBlue;
      }
```

프로젝트 특성(여러 개의 디스크 등)에 따라,
@mixin - @include를 활용하여
반복되는 디자인 요소들에 같은 속성을 간결하게 적용.

```scss
@mixin discDesign($width, $color1, $color2) {
  width: $width;
  height: 10%;
  border: 7px solid $color1;
  border-radius: 5px;
  text-align: center;
  display: flex;
  justify-content: center;
  align-items: center;
  background: linear-gradient(to bottom, $color1, $color2);
  box-shadow: 0 8px 12px rgba(0, 0, 0, 0.2);
  transition: transform 0.2s ease;
  font-size: 1.5rem;
}

.disc1 {
  @include discDesign($width: 13%, $color1: #87A7C6, $color2: #6E90B3)
}

.disc2 {
  @include discDesign($width: 22%, $color1: #A5C6AB, $color2: #89AF95)
}
...생략
```

중첩 구조를 활용하여,부모-자식 관계를 명확히 하고,
font, text 등의 속성을 그룹화하여 가독성을 향상 시키고 효율을 높임.

03 코드 분석

Javascript

Deque 자료 구조

코드 구조를 처음 설계하고 기획하는 단계에서
디스크 세팅 기능에는 큐(Queue),
디스크 이동 기능에는 스택(Stack) 자료 구조가
적절하다고 판단하였고,

이외의 모든 세부 기능을 고려하고 종합하여,
Deque 자료 구조를 활용하는 것이
가장 효율적이라고 생각하여 결정함.

```
1  class Deque {
2      constructor(id) {
3          this.id = id;
4          this.storage = [];
5          this.length = 0;
6      }
7      shiftItem() {
8          return this.storage.shift();
9      }
10     unshiftItem(item) {
11         this.storage.unshift(item);
12     }
13     popItem() {
14         return this.storage.pop();
15     }
16     pushItem(item) {
17         this.storage.push(item);
18     }
19     countDeque() {
20         this.length = this.storage.length;
21         return this.length;
22     }
23 }
```

03 코드 분석

Javascript

Hanoi Class의 속성 및 settingInput() 메소드

Deque Class의 인스턴스로 각 타워에 해당하는 배열을 속성값으로 생성하고,

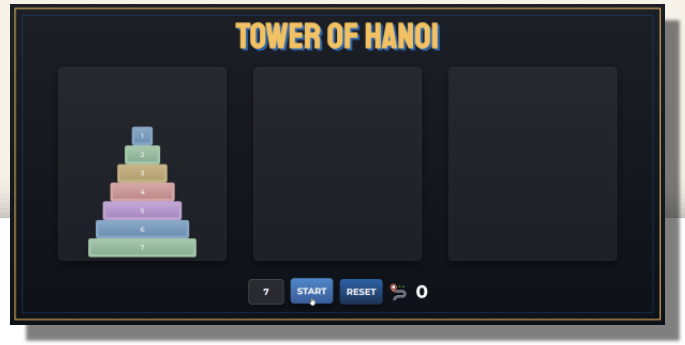
사용자의 입력값을 제어하는 메소드를 추가.

```
1  class Hanoi {
2      constructor(id) {
3          this.id = id;
4          this.firstArr = new Deque('firstArr');
5          this.secondArr = new Deque('secondArr');
6          this.thirdArr = new Deque('thirdArr');
7          this.procedureArr = [];
8          this.totalDisc = 0;
9          this.movingEvent = null;
10     }
11     settingInput() {
12         document.getElementById('discNum').addEventListener('input', (e) => {
13             if (e.target.value < 1) {
14                 e.target.value = 1
15             } else if (e.target.value > 9) {
16                 e.target.value = 9
17             }
18         })
19     }
```

go to site

03 코드 분석

Javascript



'START' 버튼 클릭 시,

기존 데이터를
모두 초기화하고,
첫 번째 타워에
디스크를 추가.

실행한 알고리즘에 따라
시각화 실행.
(디스크 이동 시작)

```
settingDisc() {
    document.getElementById('startBtn').addEventListener('click', (e) => {

        this.totalDisc = Number(document.getElementById('discNum').value);
        console.log(this.totalDisc);
        document.getElementById('firstBox').innerHTML = '';
        document.getElementById('secondBox').innerHTML = '';
        document.getElementById('thirdBox').innerHTML = '';
        this.firstArr.storage = [];
        this.secondArr.storage = [];
        this.thirdArr.storage = [];
        this.procedureArr = [];
        document.getElementById('countNum').innerHTML = 0;
        for (let i = 1; i <= this.totalDisc; i++) {
            this.firstArr.pushItem(i)
        }
        this.visualizeDisc();
        this.algorithm();
        clearTimeout(this.movingEvent);
        this.moveDisc();
        const bodyMiddle = (document.body.offsetHeight / 2) - (window.innerHeight / 2);
        window.scrollTo({ top: bodyMiddle, behavior: 'smooth' });
    })
}
```

03 코드 분석

Javascript

디스크 시각화

Tower의 상태를 실시간으로 반영하여 화면에 출력.
각 타워의 디스크 배열을 기반으로 요소를 생성하여 시각적으로 표현.

```
visualizeDisc() {

    this.firstArr.countDeque();
    this.secondArr.countDeque();
    this.thirdArr.countDeque();
    document.getElementById('firstBox').innerHTML = '';
    document.getElementById('secondBox').innerHTML = '';
    document.getElementById('thirdBox').innerHTML = '';

    for (let i = 0; i < this.firstArr.length; i++) {
        document.getElementById('firstBox').innerHTML += `<div class=disc${this.firstArr.storage[i]}>${this.firstArr.storage[i]}</div>`;
    }
    for (let i = 0; i < this.secondArr.length; i++) {
        document.getElementById('secondBox').innerHTML += `<div class=disc${this.secondArr.storage[i]}>${this.secondArr.storage[i]}</div>`;
    }
    for (let i = 0; i < this.thirdArr.length; i++) {
        document.getElementById('thirdBox').innerHTML += `<div class=disc${this.thirdArr.storage[i]}>${this.thirdArr.storage[i]}</div>`;
    }
}
```

03 코드 분석

Javascript

하노이의 탑 알고리즘

재귀적으로 디스크를 이동하며 경로를 procedureArr 속성에 저장.

```
algorithm() {
    const getProcedure = (Num, from, to, via) => {
        if (Num === 1) {
            (this.procedureArr).push([Num, from, to]);
        } else {
            getProcedure(Num - 1, from, via, to);
            (this.procedureArr).push([Num, from, to]);
            getProcedure(Num - 1, via, to, from);
        }
    }
    getProcedure(this.totalDisc, this.firstArr, this.thirdArr, this.secondArr);
}
```

go to site

03 코드 분석

Javascript

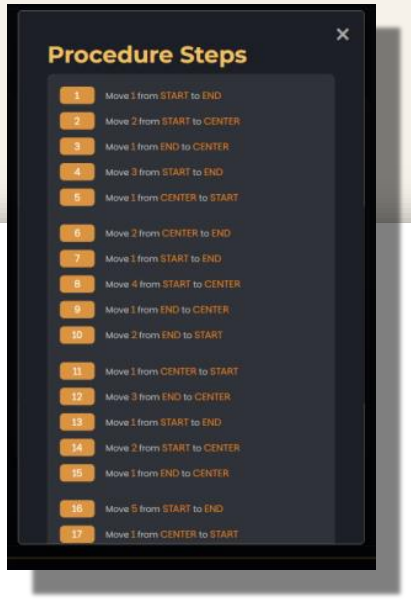
디스크 이동

procedureArr에 저장된 이동 경로를 기반으로 디스크를 한 단계씩 이동.
애니메이션 효과(classList.add/remove)를 추가하여 시각적 효과 강화.


```
moveDisc() {
    this.movingEvent = (
        setTimeout(
            () => {
                const countNumElement = document.getElementById('countNum');
                const countNum = Number(countNumElement.innerHTML);

                this.procedureArr[countNum][2].storage.unshift((this.procedureArr[countNum][1].storage.shift()));
                document.getElementById('countNum').innerHTML = countNum + 1;

                countNumElement.classList.add('animated');
                setTimeout(() => {
                    countNumElement.classList.remove('animated');
                }, 350);

                this.visualizeDisc();
                if (countNum < this.procedureArr.length) {
                    this.moveDisc();
                }
            },
            500
        )
    )
}
```

03 코드 분석

Javascript



모달 이벤트

사용자가 전체 경로 아이콘을 클릭하면 모달 창을 열고 경로를 표시.

단계별 이동 경로를 리스트로 보여줌.

```
modalEvent() {
    navBtn.addEventListener('click', (e) => {
        modal.style.display = 'flex';
        document.querySelector('.modalContent').scrollTop = 0;

        let inputString = '';
        if (this.procedureArr[0] === undefined) {
            inputString = '<span>! 로직을 먼저 실행해주세요.</span>'
            procedureSteps.innerHTML = inputString;
        } else {
            inputString += '<ul>';
            for (let i = 0; i < this.procedureArr.length; i++) {
                let discName = this.procedureArr[i][0];
                let discFrom = naming[this.procedureArr[i][1].id];
                let discTo = naming[this.procedureArr[i][2].id];
                if (i % 5 == 0) {
                    inputString += `</li><li class="devideList"><span class="colored badge ">
                    ${i + 1}</span> Move <span>${discName}</span> from <span>${discFrom}
                    </span> to <span>${discTo}</span><br>`
                } else {
                    inputString += `<li><span class="colored badge">${i + 1}
                    </span> Move <span>${discName}</span> from <span>${discFrom}
                    </span> to <span>${discTo}</span> <br>` }
            } inputString += '</ul>';
            procedureSteps.innerHTML = inputString;
```

04 리뷰

go to site

04 리뷰

지난 프로젝트에서 느꼈던 점들을 바탕으로,
이번 프로젝트에서는 기획 단계에서 코드를 설계하는 시간을 충분히 가졌습니다.

하노이의 탑 알고리즘에 대해서 충분히 학습한 후에
코드를 효율적으로 전개시킬 수 있는 방향을 잡으려 노력했고,
결론적으로는 유의미한 시간이었던 것 같습니다.

패턴이나 로직에 대해 완전히 이해하는 것을 가장 우선시하면서,
코드 작업보다 먼저 수반되어야 할 내용, 과정에 대해 직접 느껴볼 수 있었습니다.

설계 단계에서부터 Class를 활용해 코드를 구조화하려 했고,
Deque와 같은 자료구조를 활용해 효율적으로 하노이의 탑 알고리즘을 구현하는 데 집중했습니다.
그 고민들이 이 프로젝트의 모든 단계를 체계적으로 만들었고,
그 단계에 맞춰 완성까지 일관된 좋은 흐름으로 진행된 부분이 가장 뿌듯한 점으로 남아있습니다.

또, 이번 프로젝트에서는 프로젝트의 특성상 반복되는 요소나 속성들이 많다고 판단했기 때문에
Pug와 Sass를 활용하면 효율적인 가능할 것이라고 생각했습니다.
익숙해지는 데 시간이 조금 필요했지만 기초부터 차근차근 학습하면서
처음으로 프로젝트에 적용시켜 볼 수 있었습니다.

마지막으로, 재귀 함수가 포함된 알고리즘을 다루면서 재귀 함수의 원리를 더 깊이 이해할 수 있었고,
구현 과정에서 발생한 여러 오류를 해결해 나가면서 문제 해결 능력도 키울 수 있었습니다.

go to site

Thank you

감사합니다.

허기범