2022

YANG HUICHEOL

행렬 계산기
프로젝트 제작

Matrix Calculator

CODE BY
양희철

Index :

1. Schedule
일정표

1. Schedule

2022
JULY

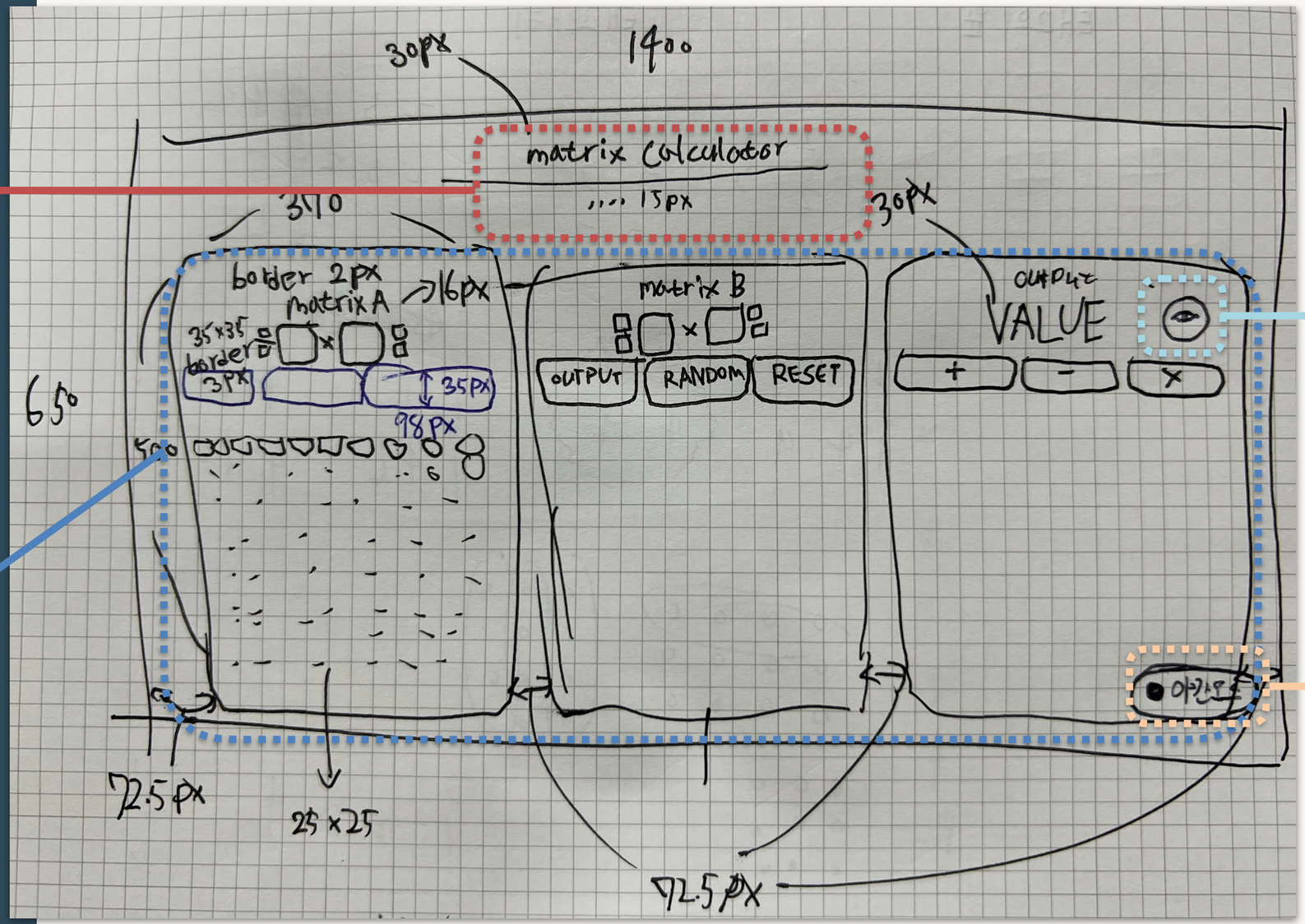
7월 10일	7월 11일	7월 12일	7월 13일	7월 14일	7월 15일	7월 16일
	프로젝트 시작 디자인 스케치 및 프로토타입 제작	**PUG, SASS 작성**	**본격적인 자바스크립트 코드 작성 및 기능 구현**		**마감**	
	행렬 계산기 레이아웃 구성 스케치, 디자인 컨셉 계획 및 Oven을 통한 프로토타입 제작	제작한 프로토 타입을 기반으로 PUG와 SASS를 작성 야간모드 버튼 제작 및 기능 구현	입력값 만큼 행, 열의 박스를 출력하는 솔루션 제작 출력 박스에 랜덤 값을 넣는 솔루션 및 입력값을 JS로 가져오는 솔루션 제작	텍스트 박스 안의 값들을 행렬 덧셈, 뺄셈 하는 기능 구현 행렬 곱셈 기능 구현	오류 검사 및 코드 보완 ppt 제작 및 제출	

2. Sketch
스케치

Sketch

Design Concept

난잡하지 않고 직관적이며 최대한 심플한 컨셉으로 구성



Title

최대한 깔끔한 상단 타이틀을
구성하기 위해
영어 제목과 한글 제목을
가운데에 얇은 선으로
구분해 둠

Layout

입력받을 2개의
Matrix A, B와 결과값을
출력하는 VALUE 박스를
각각의 파트로 나누어
직관적인 레이아웃 구성

Modal Button

출력되는 값들의 박스가
크기가 작기 때문에
출력값들을 크게 볼 수 있는
버튼 배치

Dark Mode

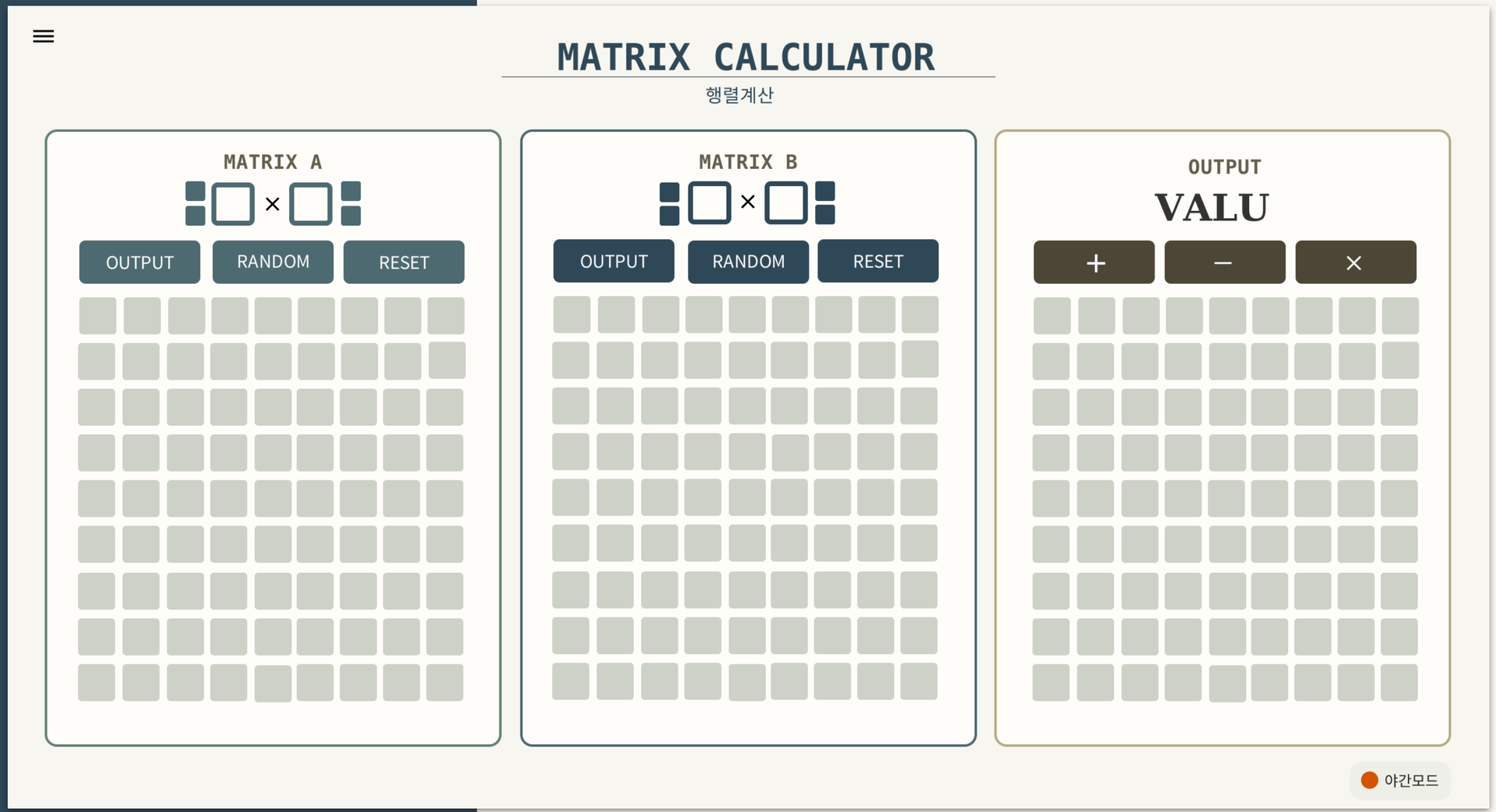
전체적으로
밝은 베이지톤으로
제작할 것이기 때문에
많은 숫자를 봐야하는
행렬계산기 특성상
눈의 피로를 줄이기 위해
야간모드 버튼 배치

Prototype

Design Tool

Kakao Oven

1. 전체적으로 직관적인 레이아웃 구성
2. 베이지 톤을 중심으로 구성
3. 부가적인 기능을 위해 좌 상단에 햄버거 메뉴 구성



COLOR

#2F4858

#4D6A70

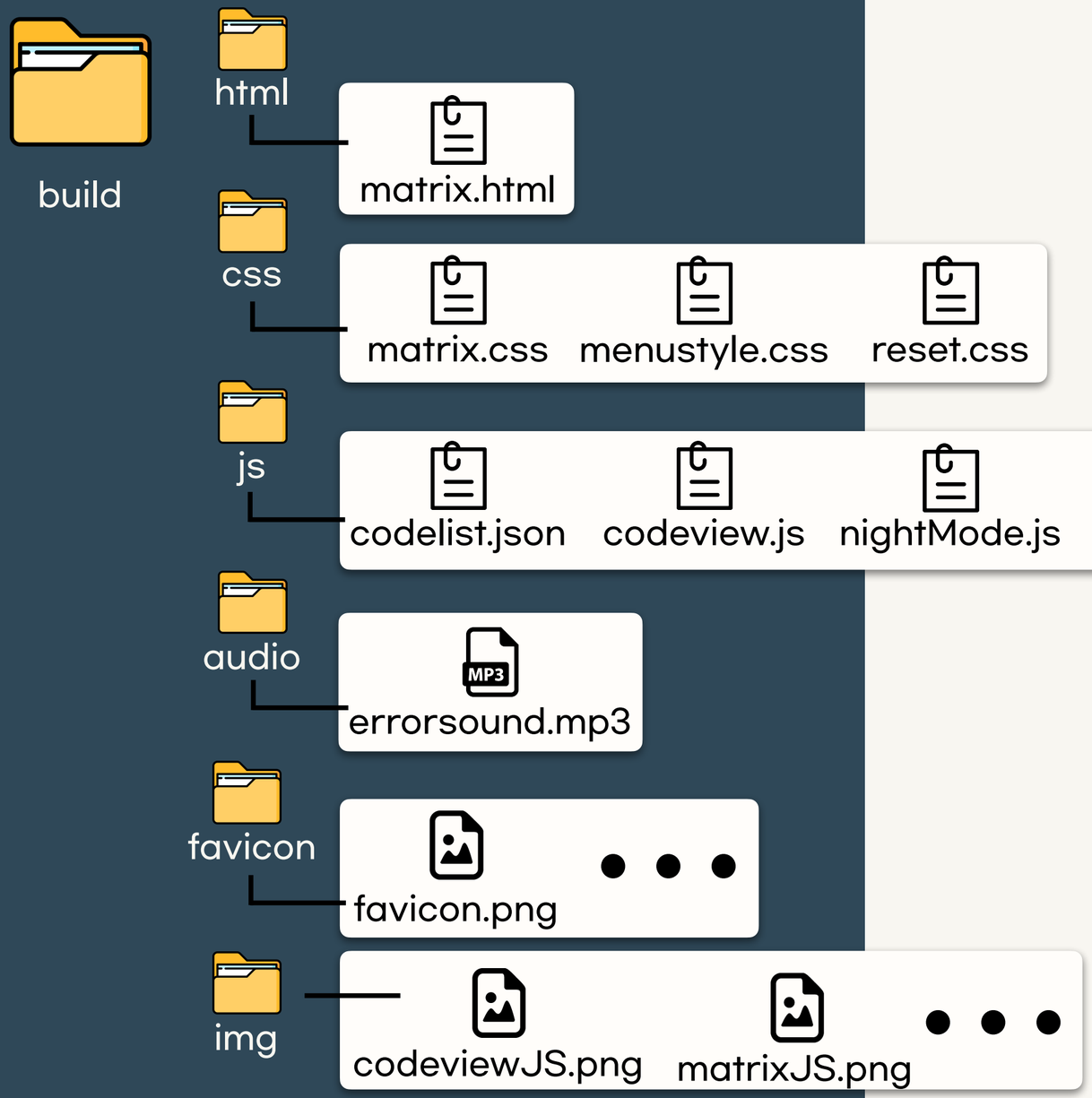
#4C4637

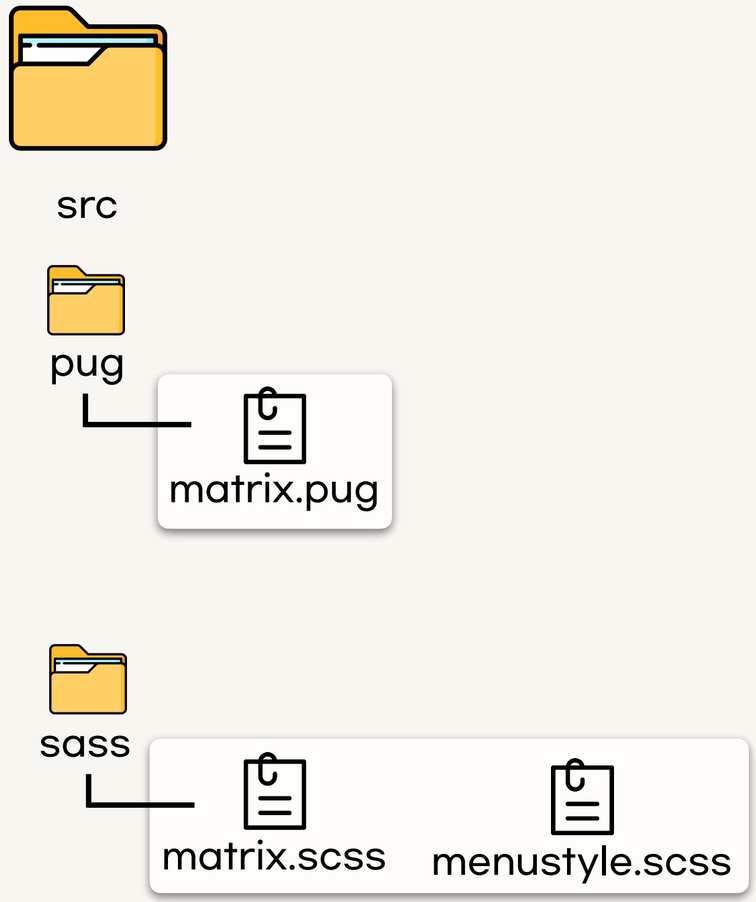
#F8F6F1

행렬 표의 형상으로
실제 행렬 계산기에 사용한
메인 컬러 4가지로 조합해
심플한 파비콘 제작

3. Folder Structure
폴더구조

Folder Structure :

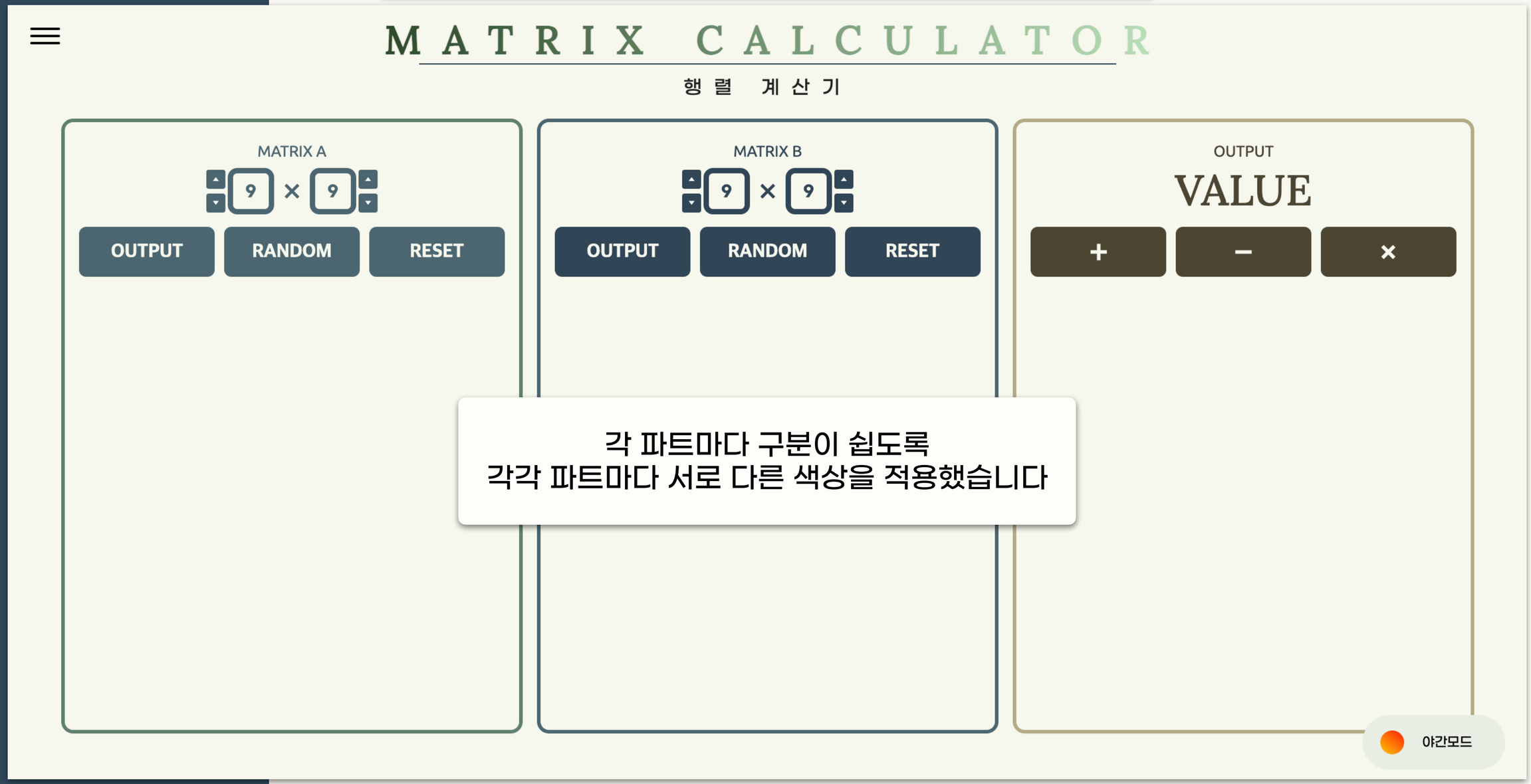




4. Design
디자인

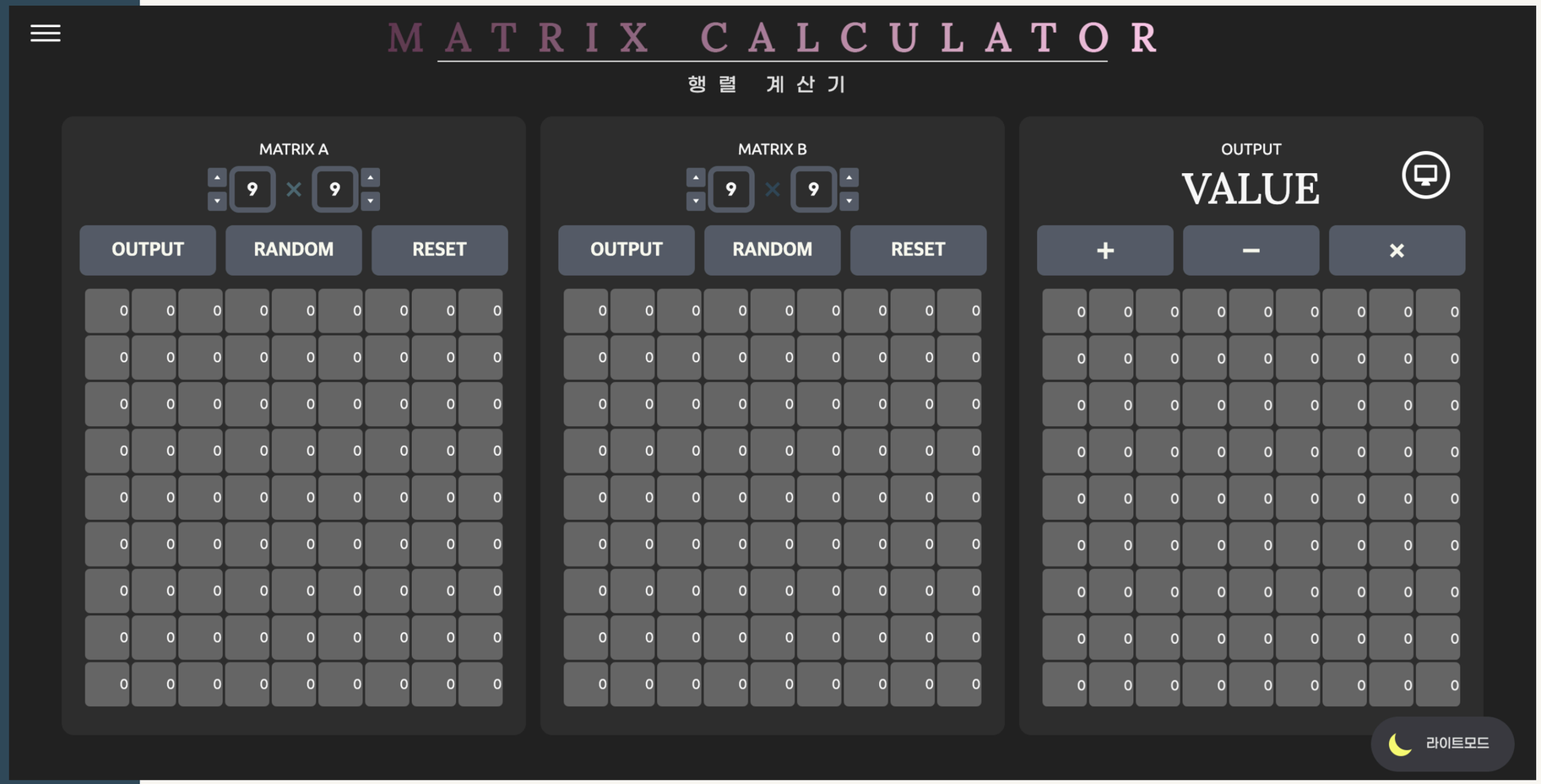
Design

전체적으로 정적인 웹페이지의 심심함을 덜기위해
색상이 변하는 타이틀과, 움직이는 선을 포인트로 주었습니다



각 파트마다 구분이 쉽도록
각각 파트마다 서로 다른 색상을 적용했습니다

Design



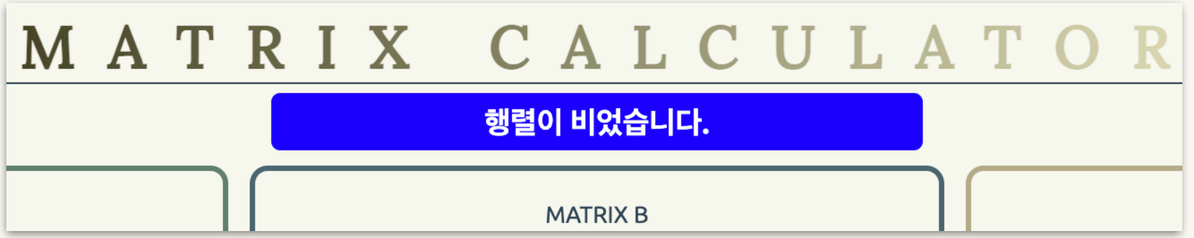
많은 숫자를 봐야하는 행렬계산기의 특성상
눈의 피로도를 줄이기 위해
다크모드를 제작했습니다.

Design

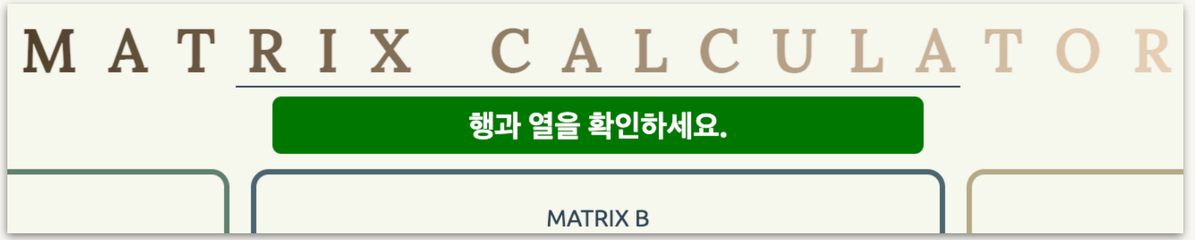
Error Design

강렬한 원색을 사용해
오류 내용을 사용자에게 강조

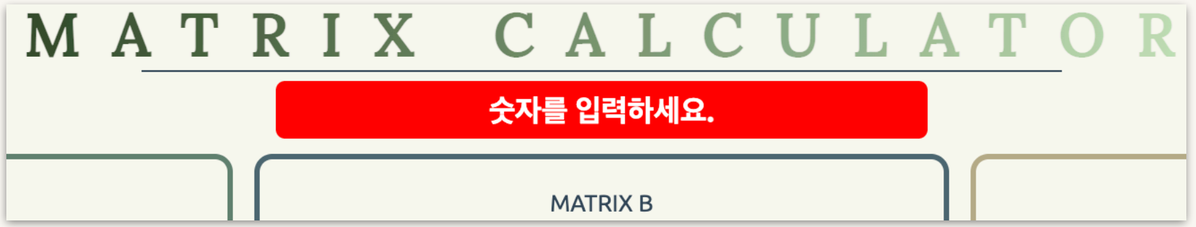
행렬이 없는 상태에서 연산을 하려고 하면
파란색의 경고 알림창 출력



행렬 연산을 하기에 행열의 수가 조건에 맞지 않으면
초록색의 경고 알림창 출력



행렬 표 안에 숫자가 비어있거나 문자를 입력할 경우
빨간색의 경고 알림창 출력



Design

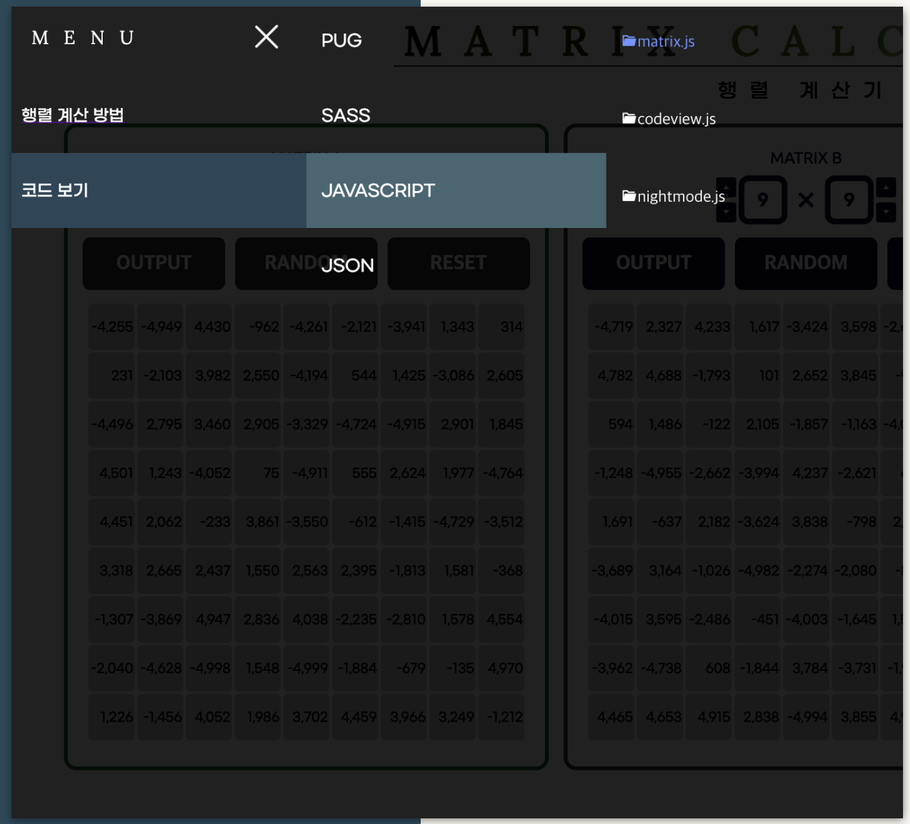
-4,618	-603	-4,259	97	-671	-969	-1,240	156	-1,465
4,118	2,616	-1,677	2,413	-3,915	-182	-1,953	1,834	3,117
2,553	-708	4,288	4,334	-1,324	300	1,842	712	3,191
1,251	1,396	-3,426	4,777		94	-3,857	1,547	4,782
605	-1,070	2,065	-4,309	-1,197	4,700	2,278	253	-2,373
3,260	-2,762	241	4,216	-4,479	3,105	925	4,065	2,995
3,459	-740	1,941	175	2,460	-2,680	2,235	-3,028	2,860
-3,483	-1,826	-1,606	-3,883	-1,934	-3,059	-2,133	2,730	2,645
1,343	2,164	2,383	138	-3,780	-1,288	3,356	-3,451	4,472

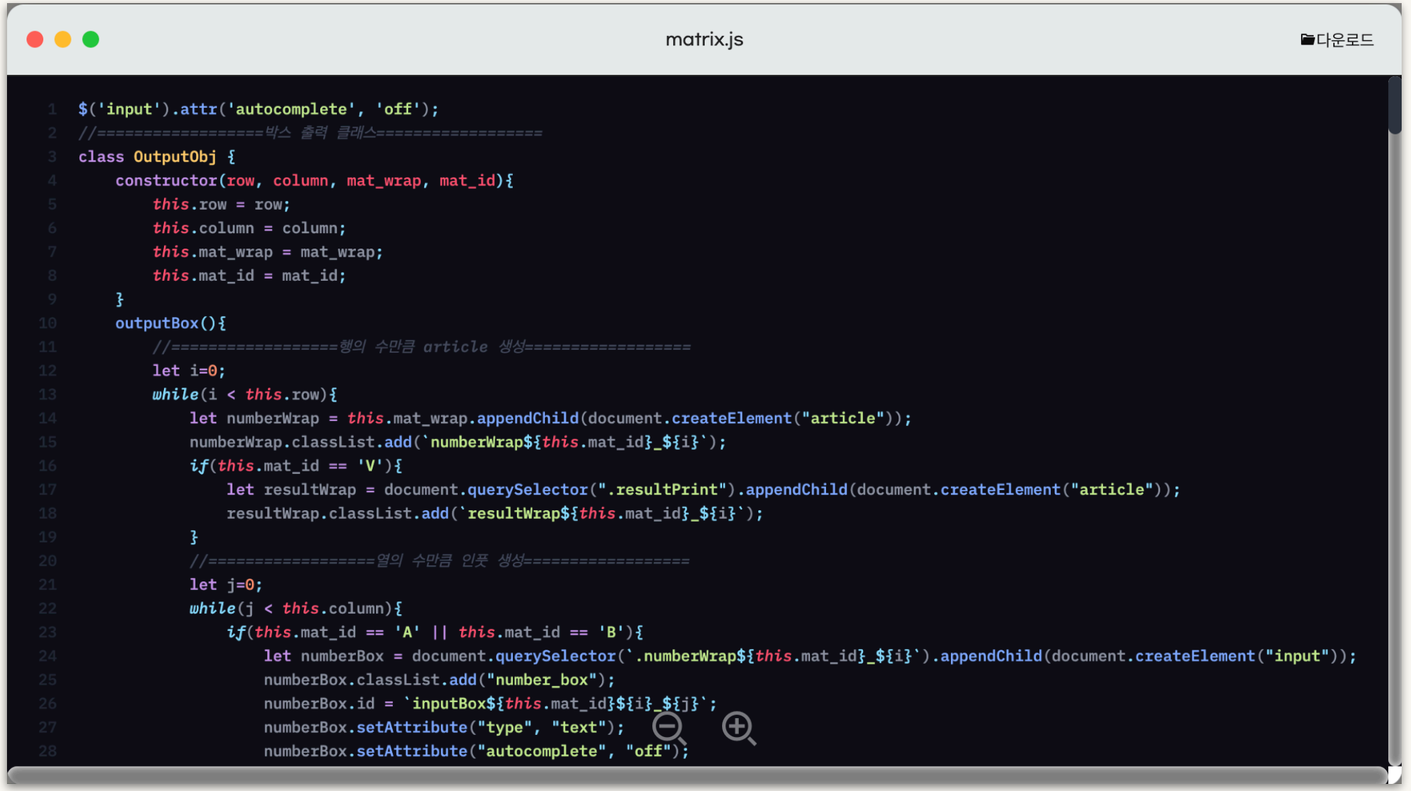
행렬안에 숫자가 비어있을 경우
사용자가 어느 위치인지 쉽게 알수 있도록
빈 공간을 붉은 배경으로 표시되며
값이 입력되면 색이 사라짐

Design

작은 행렬 표의 크기를 고려해
크게 볼수 있도록 모달창을 만들어
1억대의 자리까지 보일수 있도록 했습니다

Design





포트폴리오라는 점을 감안해
사이트 내에서 작성한 코드를 볼 수 있는
‘코드 보기’라는 메뉴를 제작했습니다

5. Code
코드

Code : JAVASCRIPT

matrix_html - matrix.js

```
class OutputObj {
    constructor(row, column, mat_wrap, mat_id){
        this.row = row;
        this.column = column;
        this.mat_wrap = mat_wrap;
        this.mat_id = mat_id;
    }
    outputBox(){
        //=================행의 수만큼 article 생성=================
        let i=0;
        while(i < this.row){
            let numberWrap = this.mat_wrap.appendChild(document.createElement("article"));
            numberWrap.classList.add(`numberWrap${this.mat_id}_${i}`);
            if(this.mat_id == 'V'){
                let resultWrap = document.querySelector(".resultPrint").appendChild(document.createElement("article"));
                resultWrap.classList.add(`resultWrap${this.mat_id}_${i}`);
            }
            //=================열의 수만큼 인풋 생성=================
            let j=0;
            while(j < this.column){
                if(this.mat_id == 'A' || this.mat_id == 'B'){
                    let numberBox = document.querySelector(`.numberWrap${this.mat_id}_${i}`).appendChild(document.createElement("input"));
                    numberBox.classList.add("number_box");
                    numberBox.id = `inputBox${this.mat_id}${i}_${j}`;
                    numberBox.setAttribute("type", "text");
                    numberBox.setAttribute("autocomplete", "off");
                    numberBox.setAttribute("onkeyup", `keyValCheck("inputBox${this.mat_id}${i}_${j}")`);
                    numberBox.value = 0;
                } else if(this.mat_id == 'V'){
                    function outputVal(who, mat_id) {
                        let resultBox = document.querySelector(`.${who}Wrap${mat_id}_${i}`).appendChild(document.createElement("div"));
                        resultBox.appendChild(document.createElement("p"));
                        resultBox.classList.add("value_box");
                        resultBox.id = `inputBox${i}_${j}`;
                    }
                    outputVal('number', this.mat_id);
                    outputVal('result', this.mat_id);
                }
                j++;
            }
            i++;
        }
        if($('.nightCheck').is(':checked')){
            $('.value_box').css('color', '#F6F6F6').css('background', '#656565');
            $('.number_box').css('color', '#F6F6F6').css('background', '#656565');
            $('article>div>p').css('color', '#F6F6F6').css('background', '#656565');
        }
    }
}
```

행렬 내 블럭을 생성하는 클래스

row(행의 수), **column**(열의 수), **mat_wrap**(블럭을 생성할 부모 태그), **mat_id**(블럭을 생성할 위치)를 매개값으로 전달 받아 클래스를 설계했습니다.

전달받은 **행과 열의 수만큼 반복문을 실행**하며 내부에서 입력받은 부모 태그에 자식 태그를 생성했습니다

이 과정에서 **mat_id가 식별자 역할**을 해 A혹은 B라면 input 태그를 생성하고 V라면 div 태그를 생성하도록 조건문을 만들었습니다

A는 matrix_a, **B는 matrix_b**, **V는 value**를 의미합니다.

Code

JAVASCRIPT

matrix_html - matrix.js

```
61  keyValCheck = mat_id => {        // bottom input
62      let inputNumber = $(`#${mat_id}`).val();
63      const resultFirst = /^\D\d|-0|^00|--$/g.test(inputNumber);
64      if(!resultFirst){
65          const resultSecond = /^-|-[0-9]+|[0-9]+$/g.test(inputNumber);
66          if(!resultSecond){
67              $(`#${mat_id}`).val('');
68              detailKeyTest();
69          } else{
70              if($(`#${mat_id}`).val() != '-'){
71                  $(`#${mat_id}`).val(parseInt(inputNumber).toLocaleString());
72              }
73          }
74      } else{
75          $(`#${mat_id}`).val('');
76          detailKeyTest();
77      }
78  }
79  //==================상세 키 입력 테스트==================
80  detailKeyTest = () => {
81      if(!(event.key == 'Backspace' || event.key == '-' || event.key == 'ArrowLeft' ||
82      event.key == 'ArrowRight' || event.key == 'Tab' || event.key == ',')){
83          warningTxt('errorNum');
84      }
85  }
```

문자열 입력 방지 정규식

행렬 내에 숫자가 아닌 **특수 문자나 문자열**이 입력되는것을 막기위한 함수 입니다.

각각의 블럭 요소들에 **onkeyup** 속성을 주어 값이 입력될 때마다 함수가 호출되도록 설계했습니다.

조건에 맞지 않는 값이 입력되면 **경고창 출력 함수를 호출**하는데 이때, 스페이스와 화살표와 같은 키가 눌렸을 때도 경고창이 출력되지 않도록 **상세 키 입력 테스트 함수**를 만들었습니다.

Code : JAVASCRIPT

matrix_html - matrix.js

```
223  // ==================행렬 곱셈 버튼 클릭시==================
224  multiClick = () => {
225      if(!document.querySelector(".number_box")){
226          warningTxt('errorMat');
227      } else{
228          if($(".numberWrapA_0>input").length == $("#matB_print>article").length){
229              let matA_test = numberTest('A');
230              let matB_test = numberTest('B');
231              if(!matA_test || !matB_test){
232                  warningTxt('errorNum');
233              } else {
234                  matMulti(arrPush('A'), arrPush('B'));
235              }
236          } else {
237              warningTxt('errorCal');
238          }
239      }
240  }
```

matrix_html - matrix.js

```
241  // ==================행렬의 값을 배열로 담는 함수==================
242  arrPush = mat_id => {
243      let matrix_arr = [];
244      let i=0;
245      while(i < $(`#mat${mat_id}_print>article`).length){
246          let matrix_set = [];
247          let j=0;
248          while(j < $(`.numberWrap${mat_id}_${i}>.number_box`).length){
249              let nowNum = $(`.numberWrap${mat_id}_${i}>#inputBox${mat_id}${i}_${j}`).val();
250              matrix_set.push(parseInt(nowNum.replace(/,/g, "")));
251              j++;
252          }
253          matrix_arr.push(matrix_set);
254          i++;
255      }
256      return matrix_arr;
257  }
```

곱셈 버튼 클릭시 발생 함수

곱셈 버튼 클릭시 우선적으로 **행렬이 만들어져 있는지 확인**한 뒤
행렬의 곱셈 조건이 맞는지 테스트 후에 행렬 내부에 비어있는 값의 여부를 확인합니다

그 후에 모든 조건이 맞으면 행렬에 입력되어있는 값들을 가져와 배열로 담아낸 후
만들어진 배열로 행렬 곱셈을 실행하는 함수를 호출합니다

Code

JAVASCRIPT

matrix_html - matrix.js

```
275  //==================행렬의 곱셈==================
276  matMulti = (matA, matB) => {
277      let calArr_value = [];
278      let i=0;
279      while (i < matA.length) {
280          let calArr_set = [];
281          let j=0;
282          while (j < matB[0].length) {
283              let sumArr = 0;
284              let k=0;
285              while (k < matB.length) {
286                  sumArr += (matA[i][k] * matB[k][j]);
287                  k++;
288              }
289              calArr_set.push(sumArr.toLocaleString());
290              j++;
291          }
292          calArr_value.push(calArr_set);
293          i++;
294      }
295      boxRequest(calArr_value);
296  }
```

matrix_html - matrix.js

```
308  // ==================최종 결과값 출력==================
309  valuePrint = valueArr => {
310      let k=0;
311      while(k < $("#matVal_print>article").length){    //행의 수만큼 반복
312          let l=0;
313          while(l < $(".numberWrapV_0>.value_box").length){    //열의 수만큼 반복
314              $(`#inputBox${k}_${l}>p`).text(valueArr[k][l]);  //결과값 입력
315              l++;
316          }
317          k++;
318      }
319      $(".windowBtn").css("visibility", "visible");
320  }
```

행렬 곱셈 함수

곱셈을 실행할 이중배열 2개를 **매개변수**로 전달받아
3중 반복문을 이용해 곱셈을 실행하고
새로운 배열에 결과 값을 담아
새로운 배열의 **행렬의 수만큼 박스를 생성하는 함수**를 호출해
박스를 생성하고 결과값을 출력하는 함수를 호출하는 방식으로
작성했습니다.

6. Review
마무리

Review

마치며...

행렬계산기를 제작하며 계산 과정에서 오류가 발생하지 않도록
여러가지 계산 상황을 고려하여 코드를 작성했고
이 과정에서 하나의 함수 내에서 모든 기능을 실행하는 것보다
각각 **고유의 기능을 하는 요소**들을 함수로 묶어
한개의 기능이 끝난 뒤에야 다음 기능을 실행하는 **동기식 코드를 사용**하고
동기와 비동기에 대해 조금 더 이해 할 수 있게 되었습니다.

사용자의 입장에서 **어떤 요소가 중요**한지,
어떤 부분이 사용자에게 **불편을 줄 수 있는지**, 그리고
행렬계산기의 특성을 위와 같은 요소들과 어떻게 **적절히 조합할지**
고민하며 사용자를 고려한 UI/UX를 디자인하며
사용자에게 편리한 디자인이 무엇인지에 대해 생각해볼수 있었습니다.

감사합니다!

monologue1935@gmail.com