PORTFOLIO

C20ST21

Matrix Calculator

(행 렬 계 산 기)

허기범

프로세스 목차

(PROCESS INDEX)

프로젝트

프로젝트의 목적 및 소개

슬로건

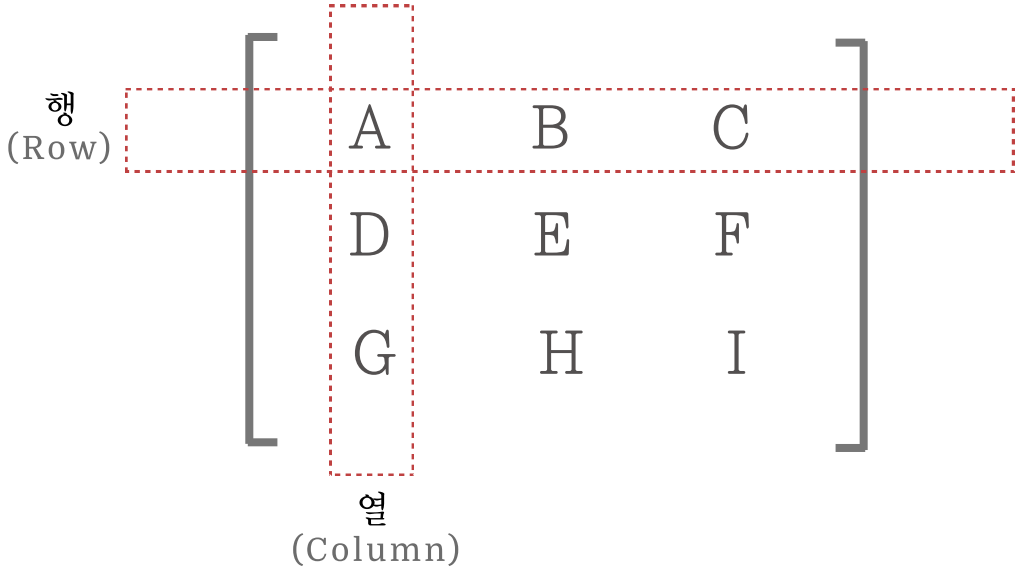
'행렬 연산을 손쉽게'

행렬 계산기
Matrix Calculator

↓

행렬에 대해 정확히 이해하고,
연산 규칙을 찾아내어
일련의 계산 과정을 코딩

행렬이란?



행렬(matrix)은 수 또는 다항식 등을 직사각형 모양으로 배열한 것이다.
가로를 행(Row), 세로를 열(Column)이라 하며,
행렬에 배열된 값을 성분(Entry)이라고 한다.

행렬의 i행 j열에 위치하는 성분을 행렬의 (i, j) 성분이라고 하며,

보다 간단하게 A_{ij}와 같이 표시한다.

프로젝트

프로젝트의 목적 및 소개

행렬의 덧셈과 뺄셈

$$A=\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad B=\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$A+B=\begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \end{pmatrix}$$

$$A-B=\begin{pmatrix} a_{11}-b_{11} & a_{12}-b_{12} \\ a_{21}-b_{21} & a_{22}-b_{22} \end{pmatrix}$$

연산에 대한 설명

- 행렬의 덧셈과 뺄셈은 행의 수와 열의 수가 같은 행렬끼리만 가능하다.

  행의 개수는 행의 개수끼리, 열의 개수는 열의 개수끼리 같아야 한다.

- 행과 열의 자리 값이 대응하는 성분끼리 더하거나 빼서,

  같은 위치에 결과값을 배치한다.

프로젝트

프로젝트의 목적 및 소개

행렬의 곱셈

$$A \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} B \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$A*B \begin{pmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} \end{pmatrix}$$

연산에 대한 설명

- 행렬의 곱셈은 A행렬의 열의 수와 B행렬의 행의 수가 같을 때에만 정의된다.

  앞 행렬의 크기가 m × n 이고, 뒷 행렬의 크기가 n × r 인 경우, 곱셈 결과 나오는 행렬의 크기는 m × r 이 된다.

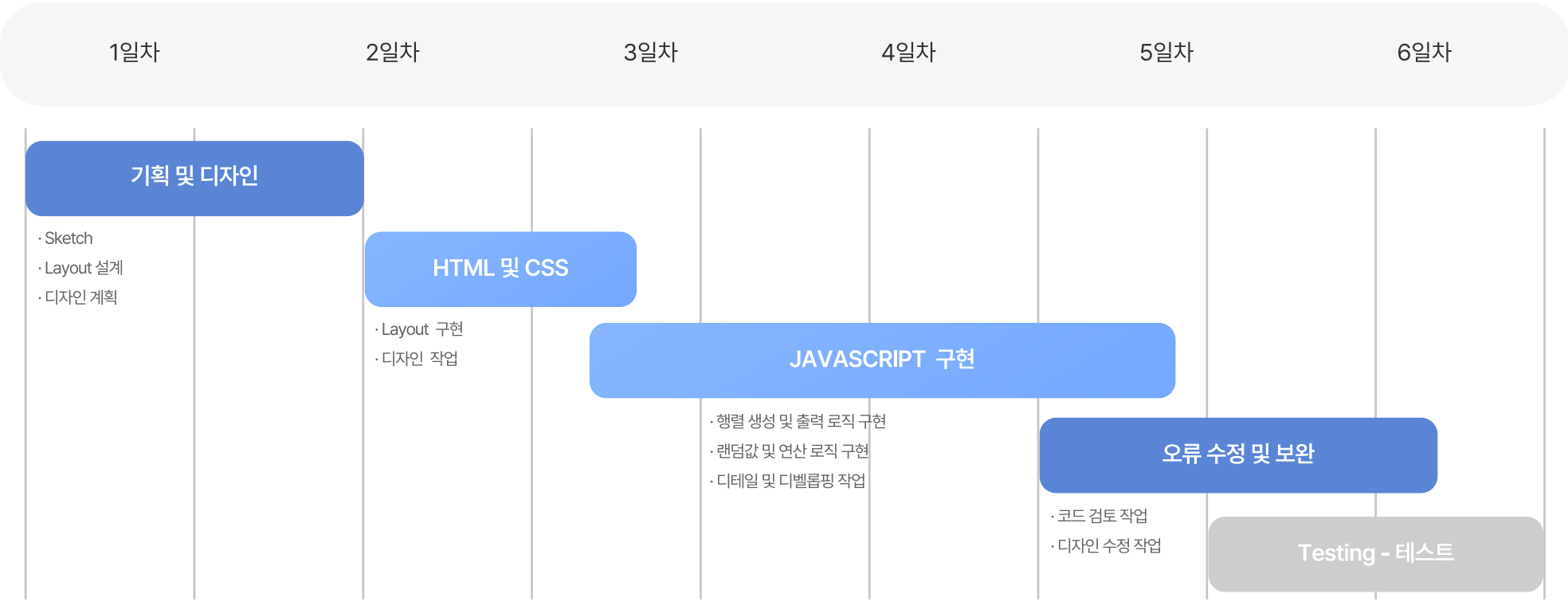
- 행렬의 곱은 A 성분의 열 자릿값과 B 성분의 행 자릿값이 같은 성분끼리 곱한 결과들을 더한 뒤,

  (A성분의 행 자릿값, B성분의 열 자릿값) 위치에 결과값을 배치한다.

프로젝트

프로젝트 일정

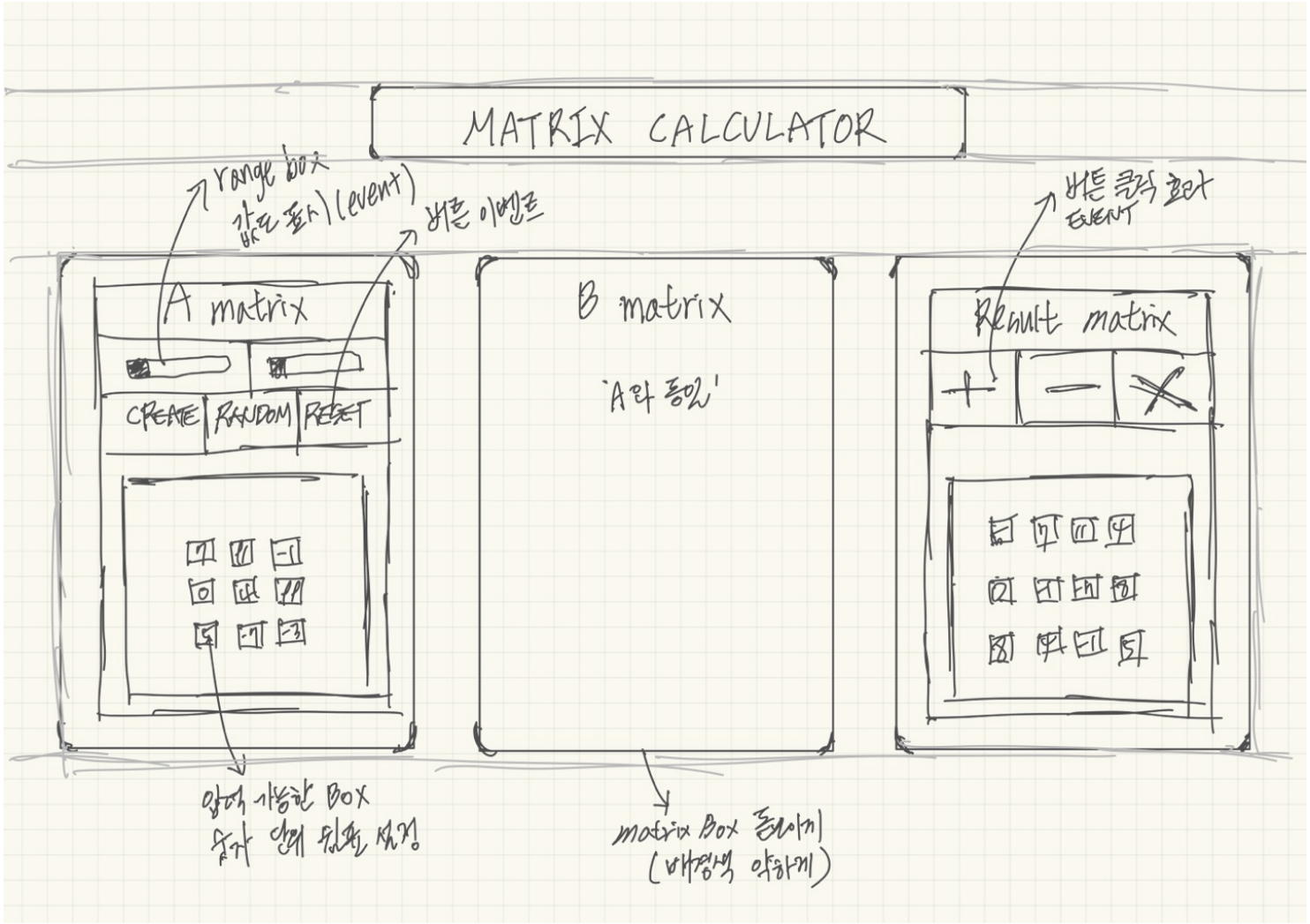
Project Schedule



디자인

디자인 설계 및 스케치

기본 스케치



디자인

디자인 설계 및 스케치

주요 키워드

Keyword 1

색상

연산 기능 작동 및
결과값에 집중 가능한
채도 낮은 컬러 선택

Keyword 2

배치

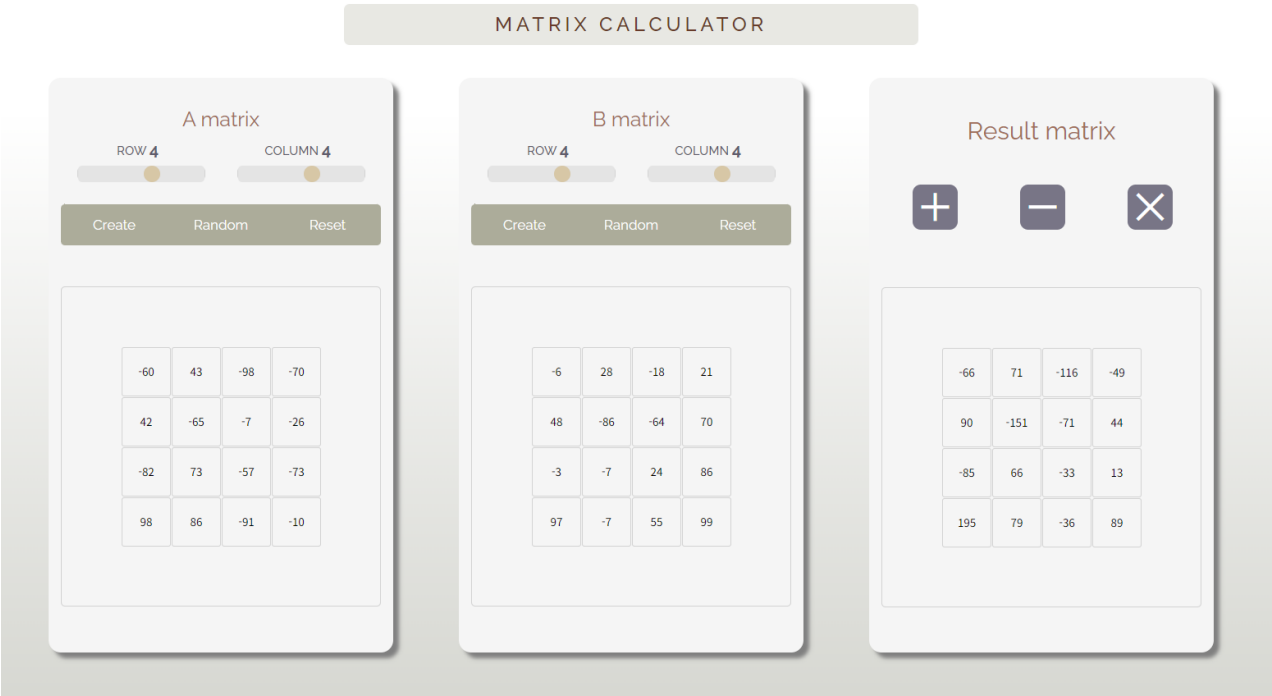
크게 세 부분으로 나누어,
가시성을 높이는데 집중

Keyword 3

간소화

기능 이외의 디테일을
최소화하여,
사용 방법에 대한 직관성을 높임

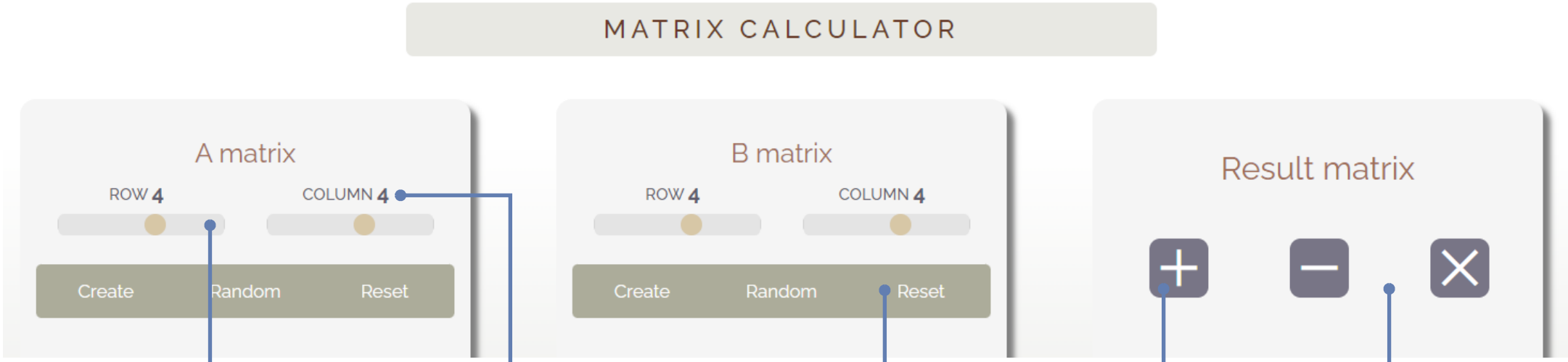
전체 디자인



디자인

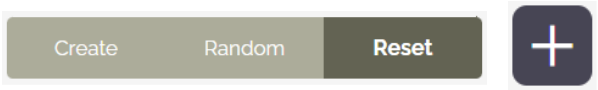
디자인 설계 및 스케치

세부 디자인 설명



입력 방법을 범위 선택자로 제공하여,
마우스의 사용만으로도 손쉽게 사용 가능케 함.

입력 값이 실시간으로 변하는
이벤트를 추가하여 편의성 높임.



버튼 선택 시,
크기 및 색상에 변화를 주는 효과를 통해
동적인 느낌을 구현함.

핵심 기능 부분에
디테일을 최소화 하여,
직관적인 사용을 유도함.

디자인

디자인 설계 및 스케치

세부 디자인 설명

'값'이 보여지는 부분에
간결한 폰트와 사이즈를 설정하여,
최대한의 가시성을 추구

입력값의 범위를 설정하고,
직접 값을 입력할 수 있도록 하여
기능 극대화

높은 값이 출력되었을 때,
가독성을 높이기 위해 세자리 단위마다
쉼표(,)를 삽입하여 표시

결과값의 변형 및 오류를 방지하기 위해
사용자의 직접 입력을 제한함

코드

코드분석

A, B(입력값) 매트릭스 객체

```
//matrix A
const matrixAoperation = {
    matrixArow: 0,      ▶ 행렬의 행값
    matrixAcolumn: 0,   ▶ 행렬의 열값
    matrixA: [],
    matrixAset: function () {  입력된 행렬의 행값, 열값을 저장
        this.matrixArow = document.querySelector('#matrixAinputX').value;
        this.matrixAcolumn = document.querySelector('#matrixAinputY').value;
    },
    matrixAmake: function () {  행 입력값과 열 입력값(value)을 Array의 형태로 변환
        let matrixAtempColumn = [];
        this.matrixA = [];
        for (let i = 0; i < this.matrixArow; i++) {
            for (let j = 0; j < this.matrixAcolumn; j++) {
                matrixAtempColumn.push('0');  ▶ default value를 '0'으로 설정
            }
            this.matrixA.push(matrixAtempColumn);
            matrixAtempColumn = [];
        }
    },
    matrixAvisualize: function () {  Array를 실제 Box의 형태로 만들어 나타냄
        document.querySelector('#matrixAoutputBox').innerHTML = '';
        for (let i = 0; i < this.matrixA.length; i++) {
            for (let j = 0; j < this.matrixA[0].length; j++) {
                                                                ▲ 실제 행렬의 성분값(행렬값)의 형태로 id 설정
                document.querySelector('#matrixAoutputBox').innerHTML += `<input id="aCell${i}${j}" class="matrixCell" type="number" value="0">`;
                document.querySelector('#matrixAoutputBox').style.width = (61 * this.matrixAcolumn) + 'px';
                document.querySelector('#matrixAoutputBox').style.height = (61 * this.matrixArow) + 'px';  ▶ Box의 width값을 설정하여,
                                                                                                            실제 행렬의 형태를 구현 (행을 나눔)
            }
        }
    },
```

코드

코드분석

A, B(입력값) 매트릭스 객체

```
matrixArandom: function () {  랜덤값을 생성
    for (let i = 0; i < this.matrixArow; i++) {
        for (let j = 0; j < this.matrixAcolumn; j++) {
            this.matrixA[i][j] = Math.floor(Math.random() * 199) - 99;  ▶ 랜덤값의 범위는 -99부터 99까지로 설정
            document.getElementById(`aCell${i}${j}`).value = this.matrixA[i][j];
        }
    }
},
matrixAreset: function () {  리셋값을 생성
    for (let i = 0; i < this.matrixArow; i++) {
        for (let j = 0; j < this.matrixAcolumn; j++) {
            this.matrixA[i][j] = '0';                    ▶ 행렬값에 따른 Array를 다시 default('0') 값으로 변환
            document.getElementById(`aCell${i}${j}`).value = this.matrixA[i][j];
        }
    }
},
getNumber: function () {  입력된 value 값을 Array의 형태로 저장
    for (let i = 0; i < this.matrixArow; i++) {
        for (let j = 0; j < this.matrixAcolumn; j++) {
            this.matrixA[i][j] = document.getElementById(`aCell${i}${j}`).value;
        }
    }
    //console.log(this.matrixA);
},
inputKeyValue: function () {  키 입력 제한
    document.querySelector('#matrixAoutputBox').addEventListener('input', function (event) {
        if (event.target.value > 100) {
            event.target.value = 99;          ▶ 범위 밖의 값을 범위 내의 최대값과 최소값으로 제한
        } if (event.target.value < -99) {
            event.target.value = -99;
        }
    })
```

코드

코드분석

A, B(입력값) 매트릭스 객체

```
document.querySelector('#matrixAsetBtn').addEventListener(
    'click',
    function (event) {
        matrixAoperation.matrixAset();
        matrixAoperation.matrixAmake();
        matrixAoperation.matrixAvisualize();
    }
)
document.querySelector('#matrixArandomBtn').addEventListener(
    'click',
    function (event) {
        matrixAoperation.matrixArandom();
    }
)
document.querySelector('#matrixAresetBtn').addEventListener(
    'click',
    function (event) {
        matrixAoperation.matrixAreset();
    }
)
matrixAoperation.inputKeyValue();
```

CREATE 버튼 event
▶ 입력값에 따라 하단 output area에 box 생성

RANDOM 버튼 event
▶ box 내에 랜덤값 생성

RESET 버튼 event
▶ box 내에 값들을 default value('0')로 초기화

입력값 제한 함수 호출

코드

코드분석

RESULT(결과값) 매트릭스 객체

```
////////matrixResult
const resultOperation = {
    matrixResultRow: 0,     ▶ result 행렬의 행값
    matrixResultColumn: 0,  ▶ result 행렬의 열값
    matrixResult: [],
    matrixResultSum: function () {   행렬의 덧셈 연산(뺄셈도 동일함)
        matrixAoperation.getNumber();
        matrixBoperation.getNumber();   ▶ A, B 매트릭스 자료(value) 호출
        let totalSumArray = [];
        if (matrixAoperation.matrixArow == matrixBoperation.matrixBrow &&        ▶ 덧셈 기본조건(행렬이 같은 경우)을 확인
            matrixAoperation.matrixAcolumn == matrixBoperation.matrixBcolumn) {     (조건에 부합한 경우 함수 실행)
            for (let i = 0; i < matrixBoperation.matrixB.length; i++) {
                totalSumArray.push([]);  ▶ 덧셈 결과값을 Array 형태로 변환
                                                  행렬의 덧셈 연산 작동 부분(for문을 이용해 Array값을 연산)
                for (let j = 0; j < matrixBoperation.matrixB[0].length; j++) {
                    totalSumArray[i][j] = (Number(matrixAoperation.matrixA[i][j]) + Number(matrixBoperation.matrixB[i][j]));
                }
                //console.log(totalSumArray);
                this.matrixResult = totalSumArray;
                document.getElementById('alertBox').style.display = "none";
                document.getElementById('matrixResultOutputBox').style.display = "block";
                document.getElementById('alertBox1').style.display = "none";
                //console.log(this.matrixResult);
            }
        } else {
            document.getElementById('alertBox').style.display = "block";
            document.getElementById('matrixResultOutputBox').style.display = "none";
            document.getElementById('alertBox1').style.display = "none";
        }
    },
```

A matrix와 B matrix의
행과 열이 모두 같아야 합니다.

▶ 덧셈 기본조건(행렬이 같은 경우)을 확인
(조건에 부합하지 않은 경우 위와같이 오류문구를 출력)

코드

코드분석

RESULT(결과값) 매트릭스 객체

```
matrixResultMulti: function () {
    matrixAoperation.getNumber();
    matrixBoperation.getNumber();
    let totalMultiArray = [];
    if (matrixAoperation.matrixAcolumn == matrixBoperation.matrixBrow) {
        //console.log(matrixAoperation.matrixA);
        for (i = 0; i < matrixAoperation.matrixA.length; i++) {
            totalMultiArray.push([]);
            for (j = 0; j < matrixBoperation.matrixB[0].length; j++) {
                let tempMulti = 0;
                for (k = 0; k < matrixAoperation.matrixA[0].length; k++) {
                    tempMulti += (Number(matrixAoperation.matrixA[i][k]) * Number(matrixBoperation.matrixB[k][j]));
                }
                totalMultiArray[i].push(tempMulti);
                this.matrixResult = totalMultiArray;
                //console.log(this.matrixResult);
                document.getElementById('alertBox1').style.display = "none";
                document.getElementById('alertBox').style.display = "none";
                document.getElementById('matrixResultOutputBox').style.display = "block";
            }

        }
    } else {
        document.getElementById('alertBox1').style.display = "block";
        document.getElementById('matrixResultOutputBox').style.display = "none";
        document.getElementById('alertBox').style.display = "none";
    }
},
```

행렬의 곱셈연산

▶ A, B 매트릭스 자료(value) 호출

행렬의 곱셈 연산 작동 부분
→ 행렬 곱셈의 기본 조건(m*n / n*r)에 따라,
for문을 삼중으로 사용하여 3개의 값(m,n,r)을 연산

▼ 곱셈 기본조건(A의 열과 B의 행이 같은 경우)을 확인
(조건에 부합한 경우 함수 실행)

▶ 곱셈 결과값을 Array 형태로 변환

A matrix의 열과 B matrix의 행이
일치하지 않아 연산이 불가능합니다.

▶ 곱셈 기본조건(A의 열과 B의 행이 같은 경우)을 확인
(조건에 부합하지 않은 경우 위와같이 오류문구를 출력)

코드

코드분석

RESULT(결과값) 매트릭스 객체

```
    matrixResultvisualize: function () {  행렬 시각화
        document.querySelector('#matrixResultOutputBox').innerHTML = '';
        //console.log(this.matrixResult.length);
        let commaValue = 0;
        for (let i = 0; i < this.matrixResult.length; i++) {
            for (let j = 0; j < this.matrixResult[0].length; j++) {

                commaValue = resultOperation.matrixResult[i][j].toLocaleString();
                document.querySelector('#matrixResultOutputBox').innerHTML += `<input id="resultCell${i}${j}" class="matrixCell" type="text" value="${commaValue}" readonly>`;
                //console.log(resultOperation.matrixResult[i][j]);
                document.querySelector('#matrixResultOutputBox').style.width = (61 * this.matrixResult[0].length) + 'px';
                document.querySelector('#matrixResultOutputBox').style.height = (61 * this.matrixResult.length) + 'px';
            }
        }
    }
}
document.querySelector('#resultSumBtn').addEventListener(  (+) 버튼 선택시 덧셈 연산 실행 후 시각화
    'click',
    function () {
        resultOperation.matrixResultSum();
        resultOperation.matrixResultvisualize();
    }
)
document.querySelector('#resultSubBtn').addEventListener(  (-) 버튼 선택시 뺄셈 연산 실행 후 시각화
    'click',
    function () {
        resultOperation.matrixResultSub();
        resultOperation.matrixResultvisualize();
    }
)
document.querySelector('#resultMultiBtn').addEventListener( (x) 버튼 선택시 곱셈 연산 실행 후 시각화
    'click',
    function () {
        resultOperation.matrixResultMulti();
        resultOperation.matrixResultvisualize();
    }
)
```

▶ A, B 매트릭스의 행렬값 (m*n / n*r) 에 따라 m*r의 형태로 box 생성

리뷰

프로젝트 후기

느낀점

첫 프로젝트였던 만큼, 과정의 한 부분 부분보다는
넓은 범위에서 느낀 성과들이 많았습니다.

프로젝트가 발전되어 가는 단계, 흐름을 직접 겪어 보면서
다음 프로젝트는 어떻게 효율적으로 진행시켜 나갈 수 있을지도 생각해 볼 수 있었고,

완성도에 대해 확신할 수 없어서 힘들었지만
끝까지 코드를 마무리 한 후에는 뿌듯함 혹은 자신감도 느꼈으며,
문제해결에 대한 방법 또한 작게나마 배울 수 있었습니다.

관련 경험이 없어 가장 막연했던 디자인 단계에서도,
아이디어를 위해 고민해보는 시간을 가질 수 있었던 자체가 좋은 경험이었습니다.

보완 체크리스트

- [x] 프로젝트 초반, 프로그램 구현에 필요한 기능을 빨리 이해하기 위해 노력하고, 자신감 가지기
- [x] 코드 진행 과정 중, 이번 프로젝트 동안 배웠던 문제해결 능력을 활용하고 더욱 함양하기
- [x] 객체 지향(class)을 활용하기
- [x] 디자인 뿐만 아니라 코드도 체계적으로 진행하기 (스케치 등 단계적으로 진행되는 프로세스)

감 사 합 니 다

THANK YOU