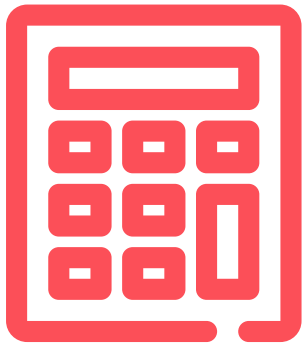
MATRIX



CALCULATOR

서동현

T. 010.5699.3299

E. tjehdgus9503@gmail.com

INDEX

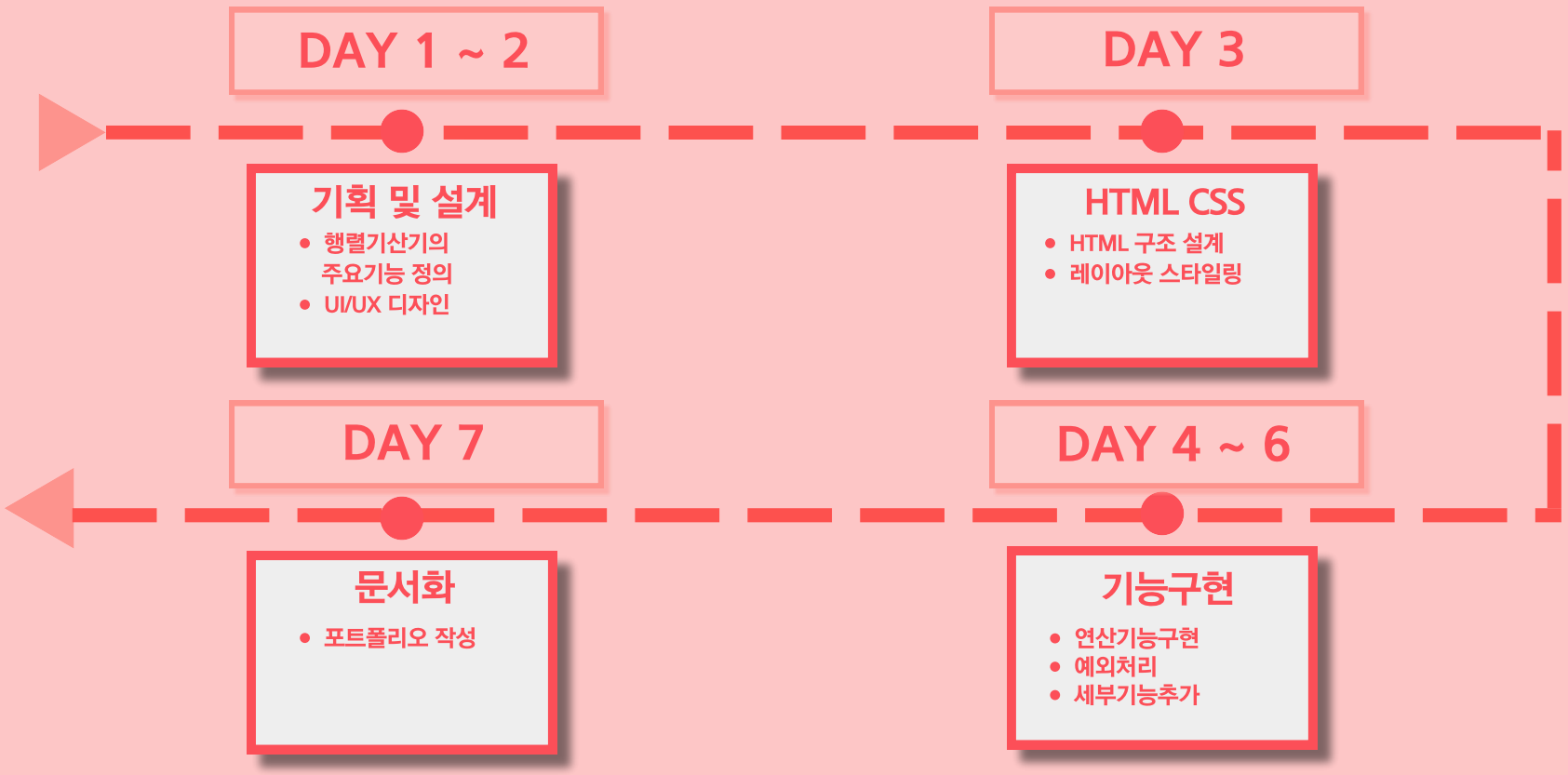
OVERVIEW

- 제가 구축한 행렬계산기는 간단한 행렬 연산을 통해 행렬 연산의 개념을 쉽게 이해할 수 있도록 도와줍니다.

- 사용자 친화적인 인터페이스를 통해 누구나 편리하게 사용할 수 있습니다.

PERIOD 2024.07.15 ~ 2024.07.21

SCHEDULE


DAY 1 ~ 2
기획 및 설계
• 행렬기산기의 주요기능 정의
• UI/UX 디자인
DAY 3
HTML CSS
• HTML 구조 설계
• 레이아웃 스타일링
DAY 4 ~ 6
기능구현
• 연산기능구현
• 예외처리
• 세부기능추가
DAY 7
문서화
• 포트폴리오 작성

OPERATION

행렬(MATRIX)이란?

1개 이상의 수나 식을 직사각형의 배열로 나열한 것

$$0_{m\times n} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathrm{Mat}(m, n; R)$$

OPERATION

행렬의 구성요소

행(ROW) 열(COLUMN)

$$0_{m\times n} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathrm{Mat}(m, n; R)$$

OPERATION

덧셈과 뺄셈

두 행렬의 크기가 같아야만 연산가능

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

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

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

OPERATION

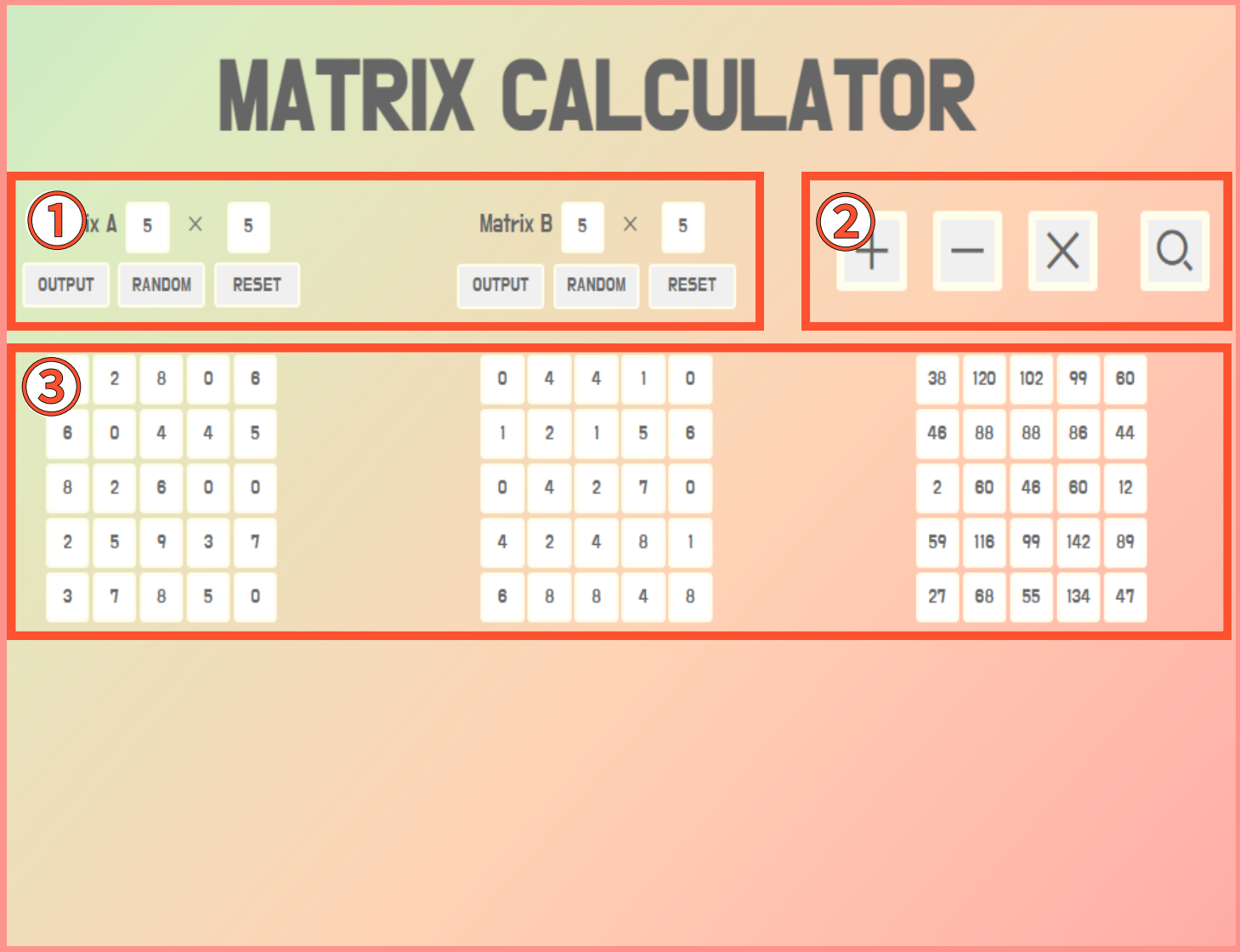
곱셈

A의 열의 숫자(n) B의 행의 숫자(n)가 같을 경우 연산가능

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}+b_{11} \times a_{12}+b_{21} & a_{11}+b_{12} \times a_{12}+b_{22} \\ a_{21}+b_{11} \times a_{22}+b_{21} & a_{21}+b_{12} \times a_{22}+b_{22} \end{bmatrix}$$

LAYOUT



① **OUTPUT 버튼**
지정된 행과 열을 화면에 표시

RANDOM 버튼
화면에 표시된 행렬에 0~9까지 값을 랜덤 입력

RESET 버튼
화면에 표시된 행렬을 화면에서 삭제합니다.

② **+, −, * 버튼**
행렬의 덧셈, 뺄셈, 곱하기 연산 수행

돋보기 버튼
자세히보기 모달창 표시

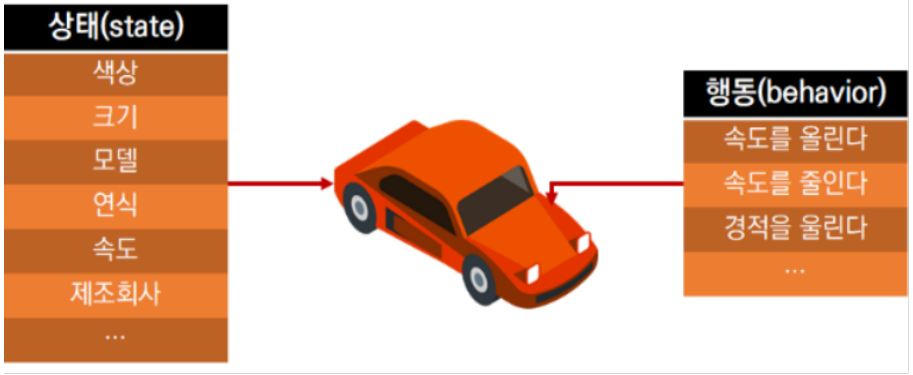
③ 지정된 행렬을 표시
연산결과를 표시

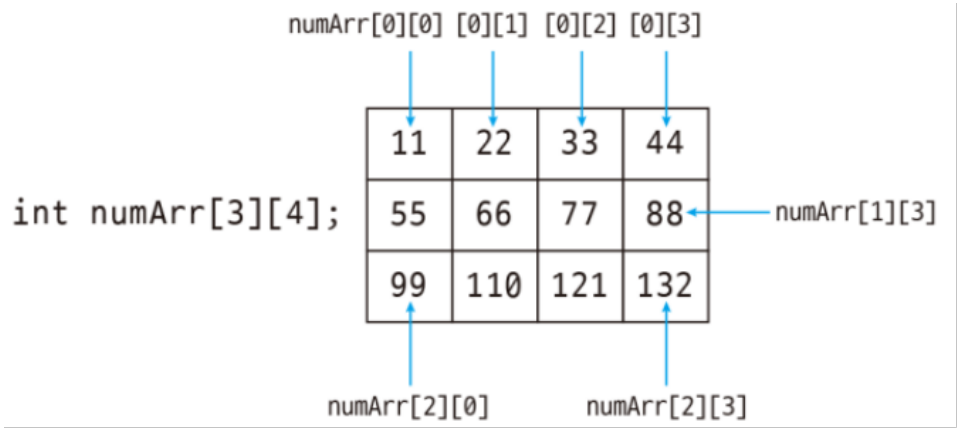
CODE

코드 리뷰에 앞서...

사용자 정의 객체와 2차원 배열을 사용하여 프로그램을 작성했습니다.

전역 변수 사용을 최소화 하고, 코드 재사용성에 중점을 두고 코드를 작성했습니다.





CODE

```
const matrixCal = {
    leftMatrix: [],
    rightMatrix: [],
    resultMatrix: [],
    leftValue: { row: 0, col: 0 },
    rightValue: { row: 0, col: 0 },
    resultValue: { row: 0, col: 0 },
    setValue: function (position) {···
    matrixSet: function (position, operationType = "") {···
    visualize: function (position) {···
    areaReset: function (position) {···
    calc: function (operation) {···
    randomSet: function (position) {···
    exceptionPopup: function(errMessage){···
    controlExpandBtn: function( value ){···
}
```

matrixCal 객체로 이벤트 리스너를 제외한 모든 기능을 수행할 수 있도록 객체를 디자인했습니다.

CODE

```
//LEFT OUTPUT BTN EVENT
document.getElementById("leftOutputBtn").addEventListener(
    'click',
    function () {
        matrixCal.setValue("left");
        matrixCal.matrixSet("left");
        matrixCal.visualize("left");
    }
);
//RIGHT OUTPUT BTN EVENT
document.getElementById("rightOutputBtn").addEventListener(
    'click',
    function () {
        matrixCal.setValue("right");
        matrixCal.matrixSet("right");
        matrixCal.visualize("right");
    }
);
```

matrix 객체의 메소드를 사용하여 코드의 재사용성을 높였습니다.

CODE

```
setValue: function (position) {
    if (position == "left") {
        this.leftValue.row = document.getElementById("leftRowValue").value;
        this.leftValue.col = document.getElementById("leftColValue").value;
    } else if (position == "right") {
        this.rightValue.row = document.getElementById("rightRowValue").value;
        this.rightValue.col = document.getElementById("rightColValue").value;
    } else { alert("Something wrong!! --> SetValue Method") }
},
```

SetValue 메소드는 HTML 콘텐츠 값(행, 열)을 받아오는 메소드입니다.

이 메소드는 파라미터로 값을 받아올 HTML 요소를 결정합니다.

명시되지 않은 파라미터 값이 입력될 경우, alert 창이 뜨도록 예외 처리를 하였습니다.

CODE

```
matrixSet: function (position, operationType = "") {
    let tempV = [];
    if (position === "left") {
        this.leftMatrix = [];

        for (y = 0; y < this.leftValue.row; y++) {
            for (let x = 0; x < this.leftValue.col; x++) {
                tempV.push(0);
            }
            this.leftMatrix.push(tempV);
            tempV = [];
        }
        console.log(this.leftValue.col, this.leftValue.row, this.leftMatrix);
    }
```

matrixSet 메소드는 입력된 데이터의 행과 열 값을 전달받아 2차원 배열을 만들어주는 메소드입니다.

코드의 재사용을 위해 역할을 매개변수로 처리하여 분배하였습니다.

자기 참조 키워드인 this를 사용하여 2차원 배열의 값을 생성하고, 초기값으로 0을 입력합니다.

CODE

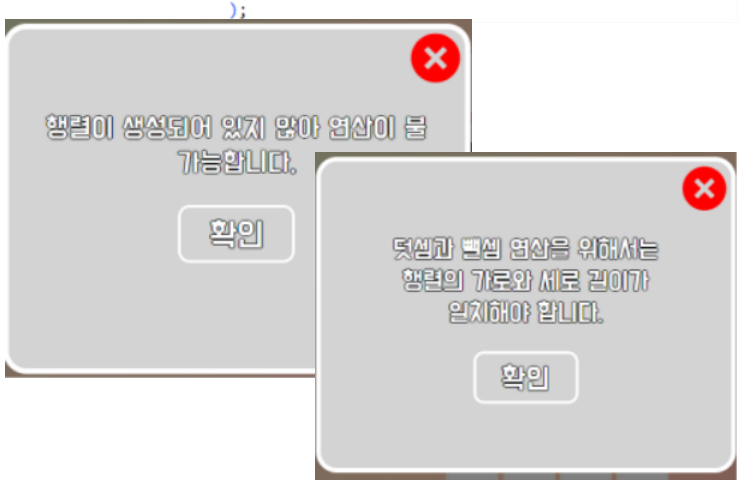
```
visualize: function (position) {
    const resultBox = document.getElementById("printResultBox");
    if (position === "left") {
        const printLeftBox = document.getElementById("printLeftBox");
        printLeftBox.innerHTML = '';
        printLeftBox.style.width = 54 * Number(this.leftValue.col) + "px";
        for (let y = 0; y < this.leftValue.row; y++) {
            for (let x = 0; x < this.leftValue.col; x++) {
                printLeftBox.innerHTML += '<input id="LeftCell' + y + x +
                        '" class="inputBoxTag" type="number" value="' +
                        this.leftMatrix[y][x] + '">';
            }
        }
    }
```

visualize 메소드는 객체에 입력된 2차원 배열의 데이터를 HTML 요소로 시각화해주는 메소드입니다.

CODE

```
calc: function (operation) {
        if (this.leftMatrix.length === 0 || this.rightMatrix.length === 0) {
        this.exceptionPopup("행렬이 생성되어 있지 않아 연산이 불가능합니다.");
        return;
    }
    if (operation == "plus" || operation == "minus") {
    if (this.leftValue.col != this.rightValue.col || this.leftValue.row != this.rightValue.row) {
        this.exceptionPopup("덧셈과 뺄셈 연산을 위해서는 <br>행렬의 가로와 세로 길이가<br> 일치해야 합니다.");
        return;
    }
        else {
            this.matrixSet("result", operation);
            this.visualize("result");
        }
    } else if (operation == "multiple") {
        console.log(this.leftValue.col, this.rightValue.row);
        if (this.leftValue.col != this.rightValue.row) {
    this.exceptionPopup("곱셈 연산은 A 행렬의 열 길이와 <br>B 행렬의 행 길이가 같아야 합니다.");
            return;
        } else {
            this.matrixSet("result", operation);
            this.visualize("result");
        }
    }
},
```

```
//PLUS BTN
document.getElementById("plusBtn").addEventListener(
    'click',
    function () {
        matrixCal.Calc("plus");
    }
);
```



calc 메소드는 연산 수행 전에 행렬의 연산이 불가능한 경우 exceptionPopup 메소드를 실행합니다.

그 후 matrixSet 메소드에 "result"와 operation을 인자값으로 넘겨줍니다.

CODE

```
// - + * 연산부분
else if (position === "result") {
    this.resultMatrix = [];
    this.resultValue.row = this.leftValue.row;
    this.resultValue.col = this.rightValue.col;

    for (let y = 0; y < this.resultValue.row; y++) {
        for (let x = 0; x < this.resultValue.col; x++) {
            if (operationType == "plus") {
                tempV.push(Number(this.leftMatrix[y][x]) + Number(this.rightMatrix[y][x]));
            }
            else if (operationType == "minus") {
                tempV.push(Number(this.leftMatrix[y][x]) - Number(this.rightMatrix[y][x]));
            } else if (operationType == "multiply") {
                tempV[x] = 0;
                for (let k = 0; k < this.leftValue.col; k++) {
                    tempV[x] += this.leftMatrix[y][k] * this.rightMatrix[k][x];
                }
            }
            else {
                alert("Something Wrong!! --> Matrixset ")
            }
        }
        this.resultMatrix.push(tempV);
        tempV = [];
    }
}
else { alert("Something wrong!! --> matrixInit") }
```

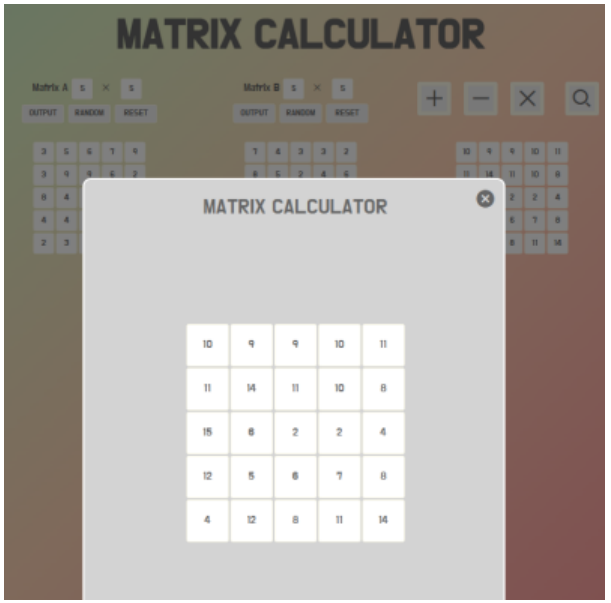
result와 operation 정보를 받은 matrixSet 메소드는 operation 값이 덧셈 또는 뺄셈인 경우

2중 for문을 사용하여 2차원 배열을 연산합니다.

곱셈인 경우에는 3중 for문을 사용하여 연산을 수행합니다.

CODE

```
//MODAL EXPAND BTN
document.getElementById("modalExpandBtn").addEventListener(
    'click',
    function(){
        document.getElementById("modalArea").style.display ="block";
    }
)
document.getElementById("modalCloseBtn").addEventListener(
    'click',
    function(){
        document.getElementById("modalArea").style.display ="none";
    }
)
```



연산 결과를 자세히 보고 싶은 경우, 돋보기 버튼을 클릭하면 자세히 보기 모달창이 표시됩니다.

모달창에 있는 데이터는 결과 데이터를 visualize할 때 함께 생성되며, 버튼이 클릭되기 전에는

display: none 상태로 숨겨져 있다가, 버튼 클릭 시 display 속성을 변경하여 표시되도록 구현했습니다.

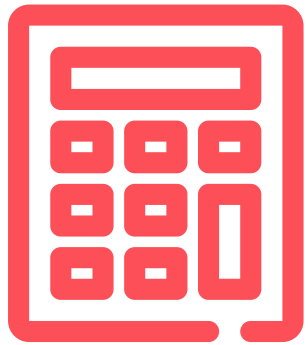
REVIEW

2주간 학습한 배열과 사용자 정의 객체를 이용해 행렬 계산기를 만들면서 느낀 점입니다.

첫 개인 프로젝트를 진행하면서, 리얼 월드의 문제를 프로그램으로 구현하는 것이 정말 어렵다는 것을 실감했습니다. VS Code를 사용하지 않고 리눅스 기본 편집기인 VIM을 사용하면서 작업을 하다 보니 기존에 사용하던 편의 기능을 사용할 수 없어 불편하기도 했지만, 직접 알파벳 하나하나를 타이핑하면서 HTML 요소, CSS, JavaScript의 문법에 대해 좀 더 깊게 생각할 수 있는 시간을 가졌습니다.

행렬의 계산 로직을 구현하기 위해 2차원 배열을 사용하면서 배열에 대한 이해도가 많이 향상되었습니다. 또한, 전역 변수 대신 사용자 정의 객체를 사용하여 프로그래밍하면서 객체의 강력함을 느낄 수 있었습니다. 앞으로 학습하게 될 클래스 기반의 복제 가능한 객체로 프로그램을 작성해 보고 싶다는 생각이 들었습니다.

이러한 프로젝트를 수행하면서 기본 프로그래밍 지식을 차근차근 쌓아간다면,
언젠가는 뛰어난 프로그래머가 될 수 있을 것이라고 믿습니다.



THANK YOU