휴먼 컴퓨터 인터페이스

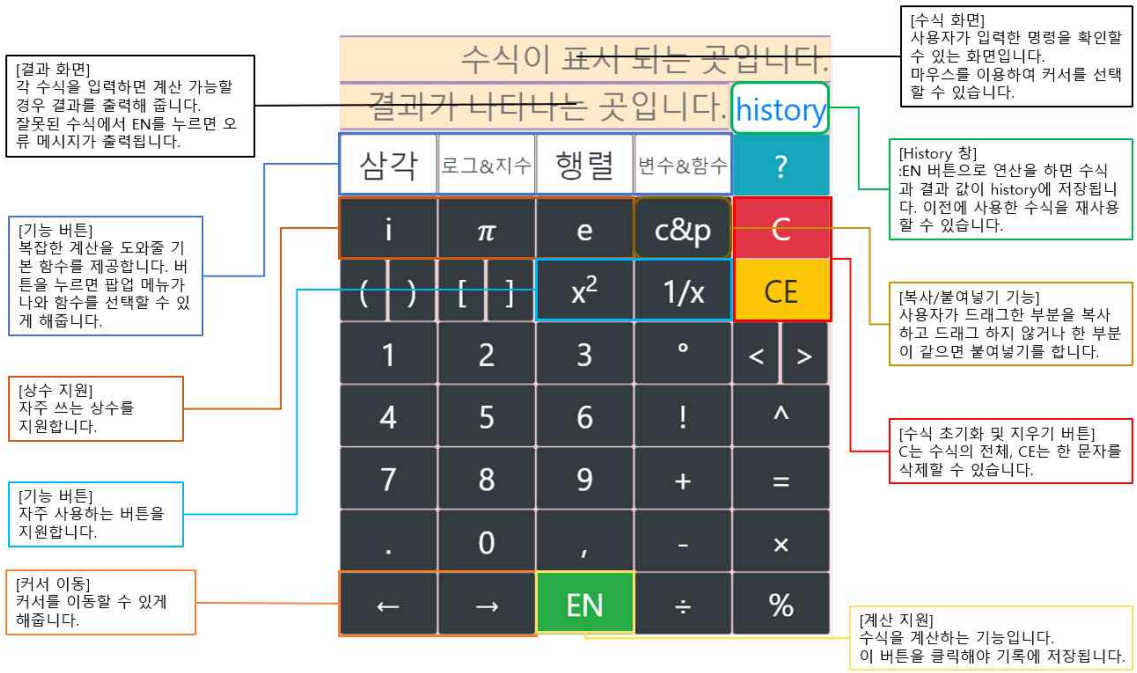
과제 #1. 기본 공학용 계산기

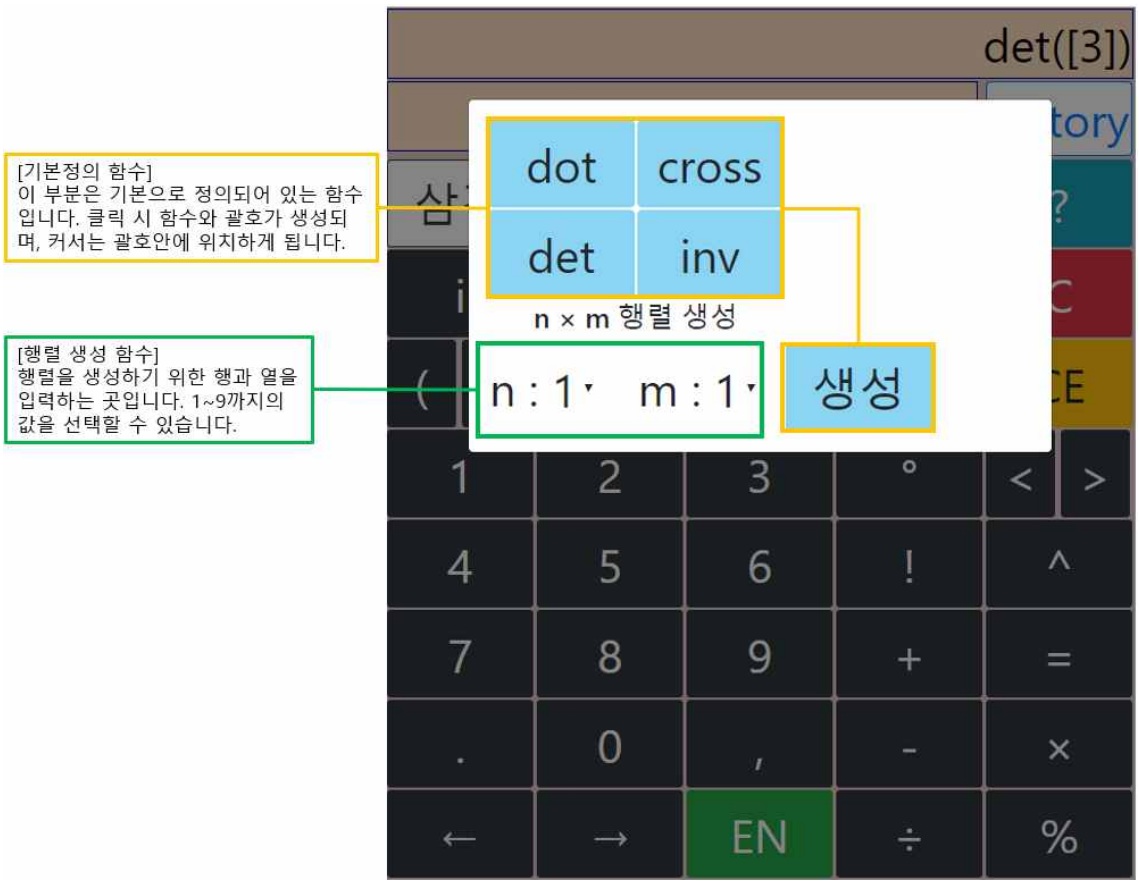
소프트웨어학부 2018203090 김광호

	기능	구현 여부
기능적 최소 요구조건	정수, 실수, 복소수의 표현과 그 기본 연산 산술연산 (+, -, *, /, %, ^), 비교연산 (==, !=, >, <, >=, <=)	O
	벡터, 행렬의 표현과 그 기본 연산 - 벡터: 내적(n차원), 외적(3차원) - 행렬: 곱셈, 역행렬(inverse), 행렬식(determinant)	O
	자주 사용되는 상수 및 함수 지원 - 상수: pi, e, i - 함수: sin, cos, tan, exp, log, sqrt	O
	결과 출력 - 올바른 입력 ➔ 수식의 결과 값 - 잘못된 입력 ➔ 오류 메시지	O
	변수, 함수 정의 및 사용 - 변수: 최소 3개 (예. x, y, z) - 함수: 최소 2개 (예. f, g)	O
인터페이스 최소 요구조건	팝업 메뉴: 기능의 그룹화, 단계별 선택지의 최소화	O
	백 스페이스: 입력 수식의 마지막 문자 지우기	O
	도움말: 인터페이스 사용법 설명	O
	벡터, 행렬 입력 간소화: 괄호 쌍 입력의 불편함 최소화	O
	복사 & 붙여넣기: 일부 영역 선택하여 보관 및 재사용	O
	히스토리: 과거 입출력 내역의 확인 및 재사용	O
추가 구현기능	초기에 정의되어있는 함수의 경우 그 함수의 시작 괄호를 삭제하면 함수 전체가 삭제된다. (안에 값이 빈 함수의 마지막 괄호를 삭제해도 전체 삭제가 된다.) (함수를 한 문자씩 삭제하지 않아도 된다.)	O
	log 중 밑이 2, e, 10인 로그 지원, 각도(deg) 지원 쌍곡선함수 : sinh, cosh, tanh 역삼각함수: arcsin,arccos, arctan 역쌍곡선함수 :arcsinh, arccosh, arctanh	O
	커서 이동 지원 (←, →) 및 커서 선택 지원	O
	수식의 전체 삭제 지원	O
	수식의 변경 때마다 계산 가능하면 결과 값 출력 (계산 못 할 경우 이전 값 출력)	O
	드래그를 통해 수식이나 결과 값의 부분 복사 지원	O
	자주 사용하는 수식인 (x)^2, 1/(x) 지원	O
	초기에 정의되어있는 함수 사용 시 괄호를 자동으로 입력해주고 괄호 안에 커서를 놓아 준다	O
	ln, log, lg, √를 수식에서 보여주고, 계산할 때 해당하는 함수로 변환해 계산한다.	O

오픈소스	사용한 기능
bootstrap, jquery, popper.js	Modal 창 기능, 각 버튼의 CSS
MathJax.js, math.js	수식 계산

사용자 인터페이스의 구성 요소 및 사용 방법





상호작용 방식에 대한 구현 설명

1. 수식 입력 화면 제어

입력 부분에서 직접 커서를 지정하는 부분입니다.

```
input.addEventListener('click', function () {
    let a = input.selectionStart;
    if (a == input.selectionEnd) {
        input.value = leftInput + rightInput;

        leftInput = input.value.substr(0, a);
        rightInput = input.value.substr(a);
```

leftInput과 rightInput은 커서를 기준으로 왼쪽, 오른쪽에 있는 수식들을 의미합니다.

input.selectionStart로 커서가 지정된 위치를 기준으로 leftInput과 rightInput를 나눕니다.

커서 이동 버튼 부분 코드입니다.

```
cursorLeft.addEventListener('click', function () {
    rightInput = leftInput.substr(leftInput.length - 1, 1) +
    rightInput;
    leftInput = leftInput.substr(0, leftInput.length - 1);
    input.value = leftInput + rightInput;
    inputFocus();
```

```
cursorRight.addEventListener('click', function () {
    leftInput = leftInput + rightInput.substr(0, 1);
    rightInput = rightInput.substr(1, rightInput.length - 1);
    input.value = leftInput + rightInput;
    inputFocus();
```

커서 이동시 각 leftInput, rightInput의 값을 수정합니다.

커서를 포커스하는 부분입니다.

```
function inputFocus() {
    input.focus();
    input.setSelectionRange(leftInput.length, leftInput.length);
}
```

각 수식을 입력할 때마다 알맞게 포커스 합니다.

2. 결과 화면 출력

계산 불가능한 문자들(π, ln)을 변경하는 부분입니다.

```
function getReplaceInput() {
    let t = input.value
    return t.replace(/π/g, 'pi').replace(/√/g, 'sqrt').replace(/÷/g, '/').replace(/×/g, '*').replace(/ln/g, 'log').replace(/lg/g, 'log2').replace(/
    log/g, 'log10').replace(/arcsin/g, 'asin').replace(/arccos/g, 'acos').replace(/arctan/g, 'atan').replace(/arcsinh/g, 'asinh').replace(/arccosh/g,
     'acosh').replace(/arctanh/g, 'atanh').replace(/°/g, 'deg');
```

각 문자에 대응되는 알맞은 수식으로 대체합니다.
(정규표현식으로 모든 문자에 대해 실행합니다.)

각 수식 입력마다 계산하는 부분입니다.

```
function calculateResult() {
    try {
        console.log("input.value: " + input.value);
        let replaceInput = getReplaceInput();
        let tmp = calFormula.eval(replaceInput).toString().replace('deg', '°');
        if (tmp.split(" ")[0] != "function")
            result.value = tmp;
    }
    catch (e) {
        console.log('계산할 수 없는 수식이므로 작동하지 않는다.');
    }
    finally {
        if (input.value == "")
            result.value = "";
    }
}
```

대체된 값을 가지고 계산을 수행합니다. 계산이 잘되면 결과 값(deg는 ˚로 대체)에 넣습니다. 계산 수행이 불가하면 에러가 발생합니다. 이때 예외처리를 통해 log만 찍고 넘어갑니다. 마지막으로 지우기 연산으로 입력값이 없을경우 결과값에 빈 문자열을 대입합니다.

3. Histroy 창

history 모달 창에 각 수식 및 결과를 넣는 과정입니다.

```
historyList.innerHTML += "<tr><button type='button' class='btn btn-outline-secondary history_list' data-dismiss='modal'
onclick=\"leftInput='" + input.value + "'; rightInput=''; input.value='" + leftInput + rightInput + "'; result.value = '" + answer +
"'\">" + value + "</button></tr>";
```

EN 버튼 클릭 시 history 모달 창 내부에 버튼 태그로 대입하고 onclick 함수를 직접 넣어 각 버튼이 독립적으로 존재하도록 했습니다.

모달 창 지정하는 부분입니다.

```
<button type="button" class="btn btn-outline-primary" data-toggle="modal" data-target="#history"
    style="width:125px; height: 64px; margin:0px; padding:0px;">
    history
</button>
```

data-toggle를 modal로 설정하고, data-target를 설정합니다.

target으로 설정된 부분입니다.

```
<div class="modal fade" id="history" tabindex="-1" role="dialog" aria-labelledby="history" style="top:50px">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-body">
                <table id="history-tablelist">
                </table>
            </div>
        </div>
    </div>
</div>
```

<table> 태그안에 js에 의해서 태그들이 채워집니다.

(모달 창 기능은 도움말이나, 기본 정의 함수에도 똑같이 적용됩니다.)

4. 수식 초기화 및 지우기

전체 삭제하는 부분입니다.

```
clear.addEventListener('click', function () {
    input.value = result.value = leftInput = rightInput = "";
    inputFocus();
    console.log("clear : input.value = " + input.value);
});
```

input, result, leftInput, rightInput 값 모두 빈 문자열로 초기화합니다.

한 문자만을 삭제하는 부분입니다. (기본 정의 함수 제외)

```
let check = true;
for (let i = 0; i < backSpaceList.length; i++) {
    let checkMatch = leftInput.match(backSpaceList[i]);
    if (checkMatch != null && checkMatch.index == (leftInput.length - backSpaceList[i].length - 1).toString()) {
        if(leftInput.substr(leftInput.length - 1, 1) == "(") {
            console.log('match : ' + backSpaceList[i]);
            leftInput = leftInput.substr(0, leftInput.length - backSpaceList[i].length - 1);
            if (rightInput.substr(0, 1) == ")")
                rightInput = rightInput.substr(1, rightInput.length - 1);

            check = false;
            break;
        }
    }
    else if (checkMatch != null && checkMatch.index == (leftInput.length - backSpaceList[i].length - 2).toString()) {
        if (leftInput.substr(leftInput.length - 2, 2) == "()") {
            console.log('match : ' + backSpaceList[i]);
            leftInput = leftInput.substr(0, leftInput.length - backSpaceList[i].length - 2);

            check = false;
            break;
        }
    }
}
if (check) {
    let piCheck = leftInput.match('π');
    if (piCheck != null && piCheck.index == (leftInput.length - 2).toString())
        leftInput = leftInput.substr(0, leftInput.length - 2);
    else
        leftInput = leftInput.substr(0, leftInput.length - 1);
}
input.value = leftInput + rightInput;
```

backSpaceList에는 기본으로 정의된 함수 이름들이 들어가 있습니다. 현재 삭제하려는 것이 기본 함수일 경우에는 괄호를 포함해 다 삭제하도록 구현하였습니다. 즉 함수 + “(”인 상태나 함수 + “()”인 상태에서 backSpace 연산이 함수까지 다 삭제가 됩니다. 기본 함수가 아닐 경우 한 문자만을 지웁니다. (π는 문자 2개로 인식해서 따로 연산하도록 했습니다.)

5. 복사 / 붙여넣기

복사와 붙여넣기 기능을 하는 부분입니다.

```
txt = window.getSelection();
console.log("window.getSelection : txt = " + txt);

if (txt.toString() == "" || txt.toString() == copyString) {
    leftInput += copyString;
    input.value = leftInput + rightInput;

    calculateResult();
    inputFocus();
    console.log("paste : input.value = " + input.value);
}
else {
    copyString = txt.toString();

    inputFocus();
    console.log("copy : copyString = " + copyString);
}
```

window.getSelection()으로 드래그된 부분을 가져옵니다. 드래그된 부분이 빈 문자열이거나 이미 복사된 문자열이라면 붙여넣기를 실행합니다. 그렇지 않다면 copyString이라는 전역변수에 문자열을 저장합니다.

6. 기본 기능 함수

x^2 기능을 제공하는 부분입니다.

```
exponential.addEventListener('click', function(){
    leftInput += "(";
    rightInput = ")^2"+ rightInput;
    input.value = leftInput + rightInput;

    inputFocus();
});
```

()^2를 만들고, 커서는 괄호 안에 위치하도록 해줍니다.

1/x 기능을 제공하는 부분입니다.

```
exponential.addEventListener('click', function(){
    leftInput += "(";
    rightInput = ")^2"+ rightInput;
    input.value = leftInput + rightInput;

    inputFocus();
});
```

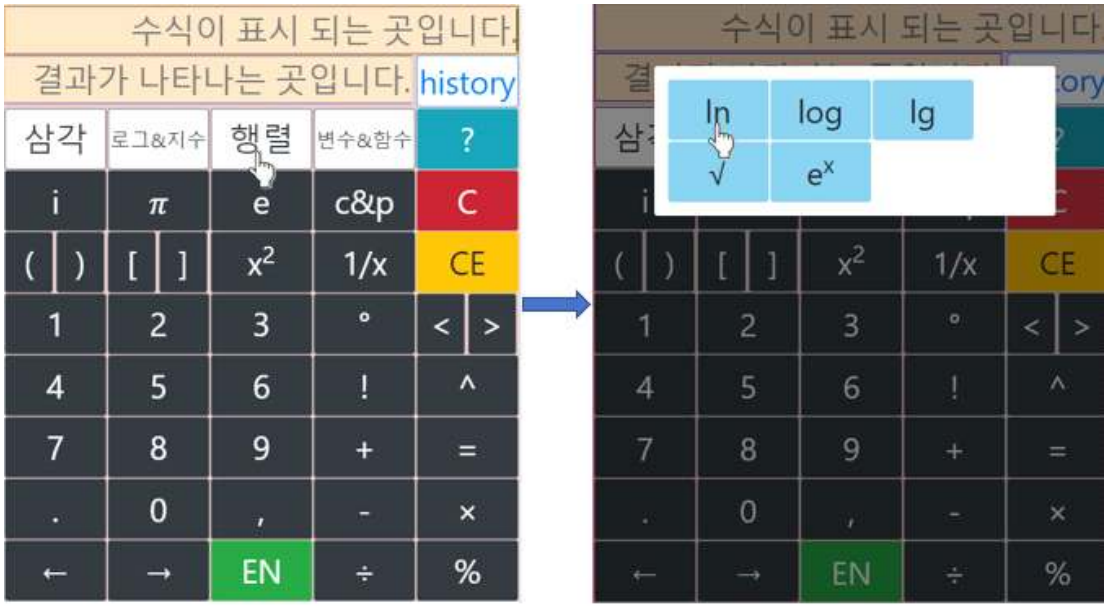
1/()를 만들고, 커서는 괄호 안에 위치하도록 해줍니다.

7. 기능 버튼

기본 정의 함수 클릭 시 실행되는 부분입니다.

```
for (let i = 0; i < specialOperations.length; i++) {
    specialOperations[i].addEventListener('click', function () {
        leftInput += specialOperations[i].value + "(";
        rightInput = ")" + rightInput;
        input.value = leftInput + rightInput;
        calculateResult();
        inputFocus();
        console.log("specialOperations[" + i + "] '" + specialOperations[i].value + "' : input.value = " + input.value);
    });
}
```

함수와 함께 괄호도 생성합니다. 커서는 괄호 안에 위치하도록 구현했습니다.



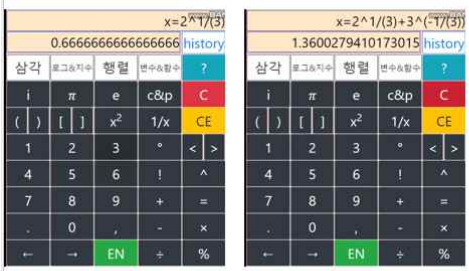
실제 문제에 대한 사용 예시

$4^{\log_2 3} = ?$

1. 4를 먼저 입력합니다.
2. ^연산자를 입력합니다.
3. 로그와 지수에 대한 함수가 있는 로그&지수 버튼을 클릭합니다.
4. lg는 밑이2인 log의 축약 형태이므로 lg를 클릭합니다.
5. 마지막으로 3를 입력합니다.
6. EN버튼을 눌러 계산완료를 하고 기록에도 저장합니다.

$x = 2^{\frac{1}{3}} + 3^{-\frac{1}{3}}$ 일때,
$2x^3 - 6x + 5 = ?$

1. 변수 설정을 위해 변수&함수를 클릭합니다.
2. 'X' 변수를 클릭합니다.
3. 'X'에 값을 대입하기 위해 '='를 입력하고 상수도 입력합니다.
4. 1/(x)의 경우 기본 제공 함수를 이용합니다.
5. 커서가 괄호 안에 존재하므로 바로 값을 입력합니다.



6.마찬가지 방법으로 뒤에 필요한 수식을 적습니다.
7. EN 버튼을 눌러 변수에 값을 저장합니다.

1. 함수를 정의하기 위해서 f를 클릭합니다.
2. 함수 인자에 x를 대입합니다.
3. f(x)를 정의합니다.
4. EN버튼을 눌러 f(x)의 정의를 완료합니다.
5. f(x)로 답을 구합니다.

1. 행렬 값은 det를 이용해서 구할 수 있습니다. 그래서 det를 클릭합니다.

2. 행렬을 만들기 위해 행렬을 다시 클릭합니다.

3. 4 * 4 행렬을 생성합니다.

4. 행렬이 생성 되었으므로 커서 이동과 직접 커서 조작으로 값들을 입력합니다.
(같은 행일 경우 커서 이동 버튼으로 다른 행으로 이동할 경우 직접 커서 조작으로 하면 편리합니다.)

5. 값들을 다 입력하면 EN 버튼으로 계산을 완료합니다.

1. 행렬의 n승을 계산하기 위한 함수를 정의하기 위해 g를 클릭합니다.

2. 필요한 인자는 하나 이므로 변수 x를 클릭합니다.

3. 3 * 3 행렬을 생성합니다.

4. 행렬에 값을 대입합니다.

5. EN 버튼을 눌러 함수 정의를 완료합니다.

1. 함수 f(x)를 정의 하기 위해 f(x)를 입력합니다.
(이미 f와 x가 정의 되어있으므로 값이 나옵니다.)

2. 행렬의 첫 번째 행만 가져오기 위한 행렬을 만듭니다.

3. g(x)와 연산하기 위한 초기 행렬을 생성합니다.

4. EN 버튼을 눌러 함수 정의를 완료합니다.

5. f(5)로 값을 계산합니다.

논의
성공적으로 구현이 됐다고 생각한 부분은 입력의 커서를 움직이고 드래그로 복사 붙여넣기 기능을 제공하는 것입니다. 긴 입력의 경우 커서 이동 버튼을 많이 눌러야 하지만 마우스 클릭으로 빠르게 이동할 수 있습니다. 수식의 전체는 history를 통해 재사용이 가능합니다. 수식의 일부분을 복사하도록 지원하는 것은 사용자가 할 수 있는 것이 좀 더 많아졌다는 것을 나타낸다고 생각합니다. 하지만 입력의 커서 같은 경우 웹에서 편리하지만 모바일로 넘어갈 경우 키보드가 나타난다는 단점이 있습니다. 이를 위해 readonly 속성을 추가하면 커서가 보이지 않는 단점이 있습니다. 그래서 이 부분에 대한 처리는 스마트폰 자체에서 키보드 기능을 꺼야 할 것 같습니다.

행렬 모달 창의 경우 보통 행렬 함수를 이용하면 바로 행렬을 이용하는 게 더 사용자 편의에 좋다고 생각합니다. 하지만 지금 제 기능으로는 함수를 입력하고 다시 행렬 버튼을 눌러서 행렬을 만들어야 합니다. 그래서 이 부분이 약간 아쉽다고 생각합니다. 행렬 생성을 부분을 밖으로 빼기에는 비슷한 기능끼리 묶는 느낌이 사라지기 때문입니다. 다른 측면에서 보자면 행렬을 사용 안 할 경우는 UI 적으로 괜찮다고 생각합니다.

화면의 해상도가 모바일 640 * 960, 컴퓨터 660 * 744 기준이 가장 적당한 비율입니다. 하지만 역시 두 개의 비율이 맞지 않아 문제가 있다고 생각합니다. 크기를 어떻게 줘야 할지는 많이 생각해 봐야 할 것 같습니다. 해상도가 달라 여러 문제가 발생하는 것은 피할 수 없지만 적어도 최소화는 하고 싶기 때문입니다.

과제를 할 때 저는 기본적인 Javascript만을 이용하려 했습니다. 기본적인 javscript로 충분히 구현 가능하다고 생각했기 때문입니다. 하지만 생각과는 다르게 약간 더 복잡해진 것 같습니다. 다음 과제에서는 jQuery를 사용할 계획인데 얼마나 더 쉽게 할 수 있을지 기대가 됩니다.

일단 제 계산기는 한눈에 봤을 때 직관적이라고 생각합니다. 종류별로 색을 다르게 해서 헷갈리지 않고 사용할 수 있으며, 대략적으로는 무슨 기능을 하겠다고 알 수 있다고 생각합니다. 예를 들어 전체 삭제 버튼은 빨간색으로 위험 표시를 한 문자 삭제의 경우 노란색으로 경고 표시를 주었습니다.

저는 사용자가 예외를 발생시키지 않도록 하고 싶었습니다. 그래서 함수를 지울 때 전체를 지우게 하는 등 여러 조치를 했습니다. 하지만 삼각 함수의 경우 너무 많은 함수가 한 번에 보여서 사용자가 뭔가 누르기 싫어 할 수 있을 것 같다는 생각이 들었습니다. 하지만 각 단어를 분리할 경우 괄호를 자동으로 넣어주는 기능을 수행하기 힘들고 모달 창의 한번 입력하면 사라진다는 특징이 사라지는 것도 마음에 걸렸습니다. 디자인적으로 좋다고는 할 수 없다고 생각되는 데 기능적으로는 좋다고 생각됩니다.

제 계산기는 무엇인가 많이 제공하는 것 같지만 편리하다고는 말하지 못할 것 같습니다. 직접 문제를 풀면서, 팝업 기능으로 여러 번 클릭해야 한다는 것은 처음 한 번에 눈에 들어 오지만 갈수록 클릭 횟수가 증가한다는 단점이 있습니다. 사용자가 자주 사용하는 함수를 밖으로 뺄 수 있는 기능을 다음에는 구현해보고 싶습니다.

Youtube 링크 : https://youtu.be/VmG7y1TWYyk

사진에 나오는 마우스 이미지 출처: https://kor.pngtree.compngtree.com