



MATRIX CALCULATOR^{+ - ×}

박경호

INDEX



SCHEDULE _ page 3

SKETCH & DRAWING _ page 4 ~ 5

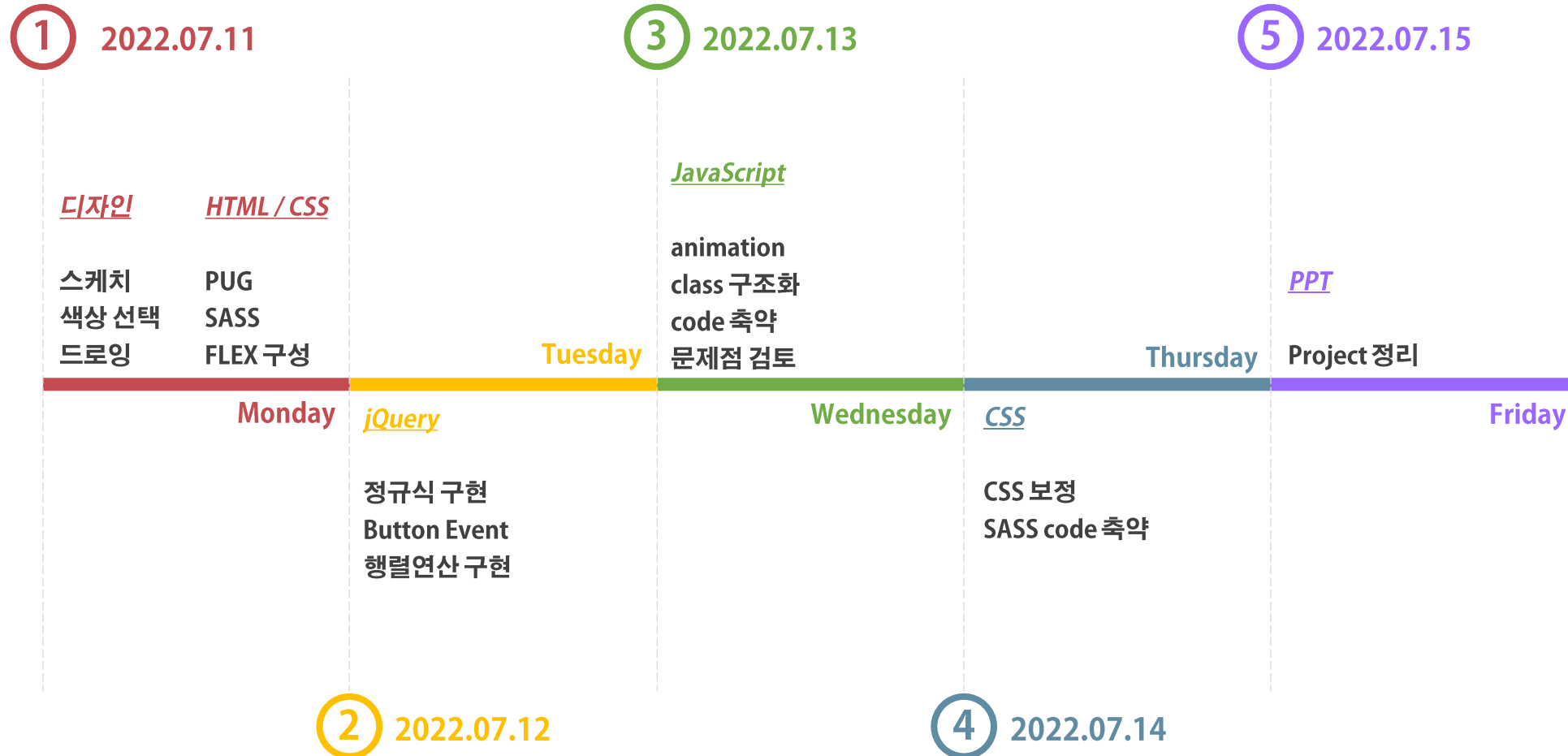
DESIGN LAYOUT _ page 6

FOLDER STRUCTURE _ page 7

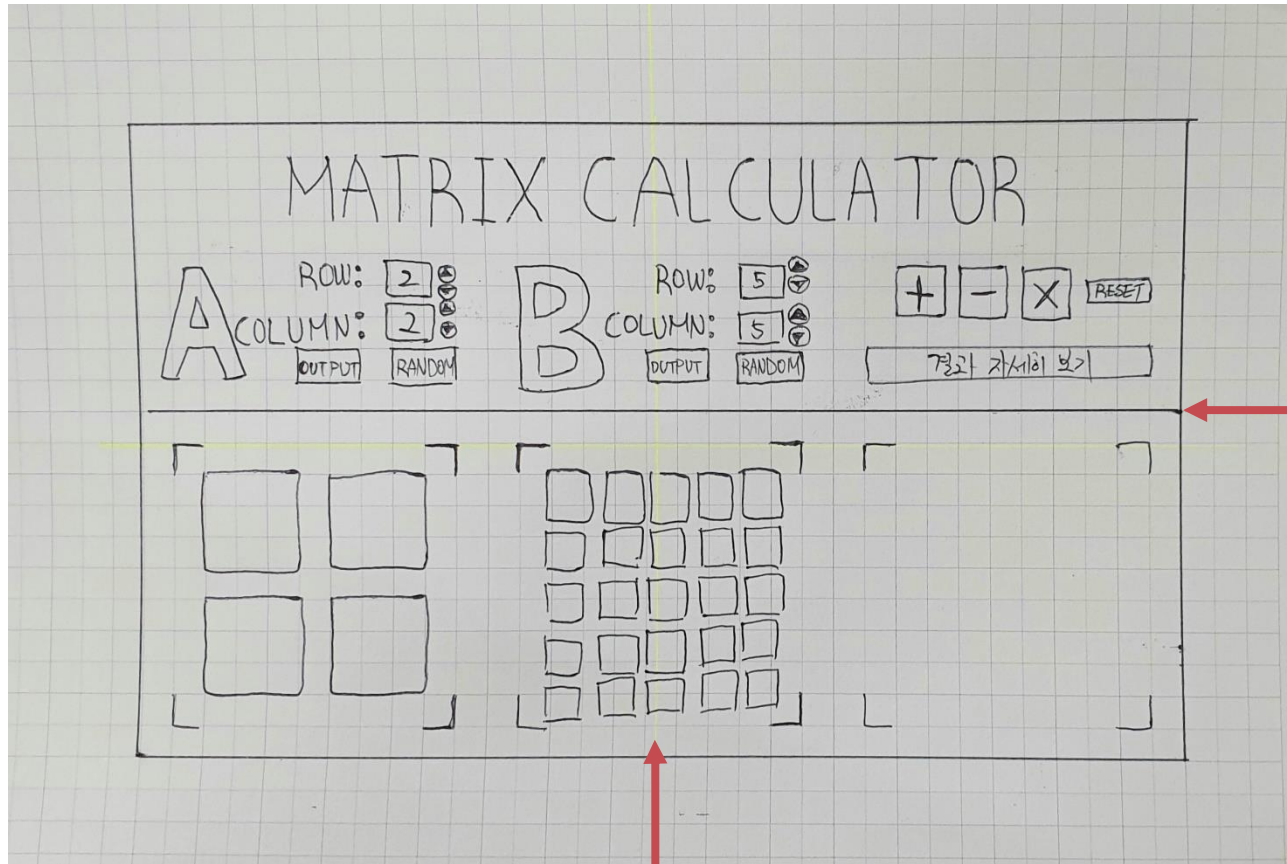
CODE _ page 8 ~ 15

REVIEW _ page 16

SCHEDULE



SKETCH



입력과 출력 사이에 구분선을 주어
출력 숫자에 집중될 수 있도록 함

flex 구성으로 Matrix A, Matrix B,
Result Box를 일정하게 정렬

DRAWING



OVEN을 활용한 Drawing

MATRIX CALCULATOR

A
ROW:
COLUMN:

B
ROW:
COLUMN:

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

값의 가독성을 위해
낮은 채도의 Color Palette 사용

#C44C51

#FFB6B8

#FFEFB6

#A2B5BF

#5F8CA3

DESIGN LAYOUT



MATRIX CALCULATOR

A

ROW : 5
COLUMN : 5

OUTPUT RANDOM

3,939	6,225	7,164	8,483	4,686
6,755	4,003	4,004	4,711	5,761
1,671	5,705	206	2,471	3,302
3,887	7,167	6,630	5,965	7,696
6,322	1,981	9,671	-887	7,428

B

ROW : 5
COLUMN : 5

OUTPUT RANDOM

3,347	-326	9,622	318	4,585
6,289	270	5,977	6,213	745
9,818	5,137	6,268	3,398	2,676
-885	2,143	-63	5,025	2,619
5,888	8,363	-723	6,695	2,705

+ - ×

RESET

결과 자세히 보기 ▼

7,286	5,899	16,786	8,801	9,271
13,044	4,273	9,981	10,924	6,506
11,489	10,842	6,474	5,869	5,978
3,002	9,310	6,567	10,990	10,315
12,210	10,344	8,948	5,808	10,133

HEADER AREA

Animation 적용

INPUT AREA

행렬 & Random 값 생성
연산 및 초기화 버튼
결과값 자세히 보기

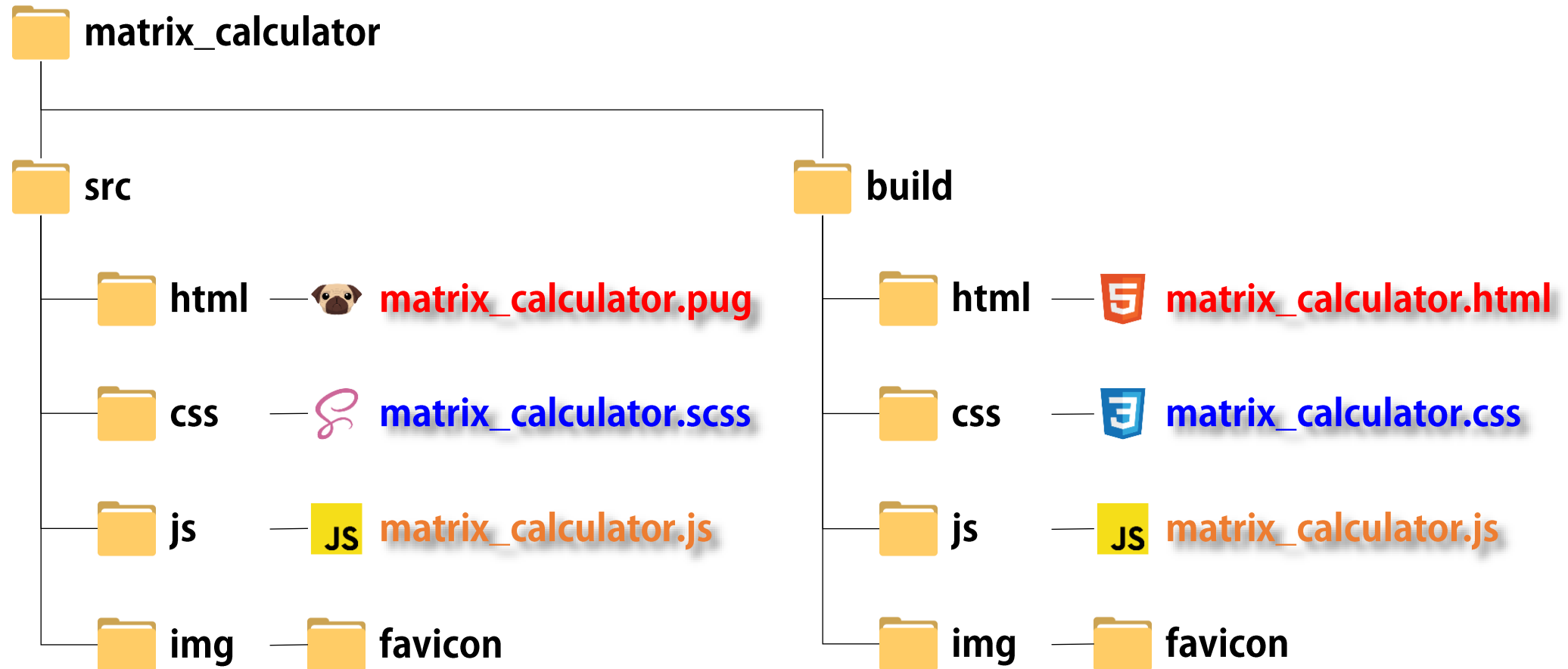
OUTPUT AREA

직접 계산 값 입력 가능
연산 결과 확인

BACKGROUND

눈이 내리는 듯한 효과 적용

FOLDER STRUCTURE



```
@keyframes header_bar {
  0% { left: -5rem; width: 5rem; }
  50% { left: calc(50% - 37.5rem); width: 75rem; background: $color_2; }
  100% { left: 100%; width: 5rem; }
}

@keyframes header_color {
  0% { color: $color_4; }
  50% { color: $color_1; text-shadow: 0.5rem 0.5rem 0.5rem $color_4; }
  100% { color: $color_4; }
}

@keyframes rect_left {
  0% {top: -5rem}
  80% {top: -5rem}
  100% {top: 8rem}
}

@keyframes rect_right {
  0% {top: 8rem}
  80% {top: 8rem}
  100% {top: -5rem}
}

@keyframes rect_top {
  0% {right: -5rem}
  19% {right: -5rem}
  80% {right: 75rem}
  100% {right: 75rem}
}
```

제목에 action bar와 color change
효과를 적용하여 제목을 강조

MATRIX CALCULATOR

MATRIX CALCULATOR

CODE *JavaScript & jQuery*



//=====배경 눈내리는 효과=====//


```
let snow_key = 0;
let snow_action = () => {
  let snow_size = (Math.random()*(5-1+1)+1).toFixed(3);
  let to_position = Math.floor(Math.random()*(120+20+1)-20);
  let from_position = Math.floor(Math.random()*(120+20+1)-20);
  let snow_opacity = Math.floor(Math.random()*(60-40+1)+40);
  let snow_duration = Math.floor(Math.random()*(30000-10000+1)+10000);
  let snow_effect = [
    {top: "-20%", left: `${to_position}%`, color: '#FFEFB6', opacity: `${snow_opacity}%`, fontSize: `${snow_size}rem`},
    {opacity: '40%'},
    {top: `calc(115rem - ${snow_size}rem)`, left: `${from_position}%`, color: '#FFB6B8', opacity: `0%`, fontSize: `${snow_size}rem`}
  ];
  let snow_timing = {
    duration: snow_duration,
    iterations: 1
  }
  $('body').append(` class="snow_${snow_key} xi-snow-crystal"></i>`);
  let snow_target = $('.snow_${snow_key}');
  snow_target[0].animate(snow_effect, snow_timing);
  snow_key++;
  let snow_delete = () => {
    snow_target.remove();
  }
  setTimeout(snow_delete, snow_duration);
  setTimeout(snow_action, snow_duration);
};

let i = 0;
while(i < Math.floor(Math.random()*(70-30+1)+30)){
  let snow_roof_time = Math.floor(Math.random()*(30000+1));
  setTimeout(snow_action, snow_roof_time);
  i++;
}
```

배경에 자연스럽게 눈이 내리는 효과 구현

```
class Matrix_calculator{ //=====MATRIX CALCULATOR CLASS 선언==
  constructor(){
  }
  box_control(){ //=====MATRIX CALCULATOR 동작 메소드=====
  }
}
```

JavaScript & jQuery CLASS 구조



```
box_control(){ //=====MATRIX CALCULATOR 동작 메소드=====//
  class New_inner { //=====내부박스 클래스=====//
    constructor (row, column){ -
    }
    reg_input(e){ //=====입력창 정규식 메소드=====// -
    };
    reg_box(e, count){ //=====내부박스 정규식 메소드=====// -
    };
    create_inner(box, type, keep_data){ //=====내부박스 생성 메소드=====
    }
    random_inner(box, keep_data){ //=====랜덤숫자 적용 메소드=====
    }
    data_push(box){ //=====A, B DATA get 메소드=====// -
    };
    data_calculation(box, set_calculator, data_arr_a, data_arr_b){ //=====
    }
    input_effect(){ //=====EVENT 메소드=====// -
    }
    click_effect(element){ //=====Click EVENT 메소드=====//
    }
  }
}
```

```
click_effect(element){ //=====Click EVENT 메소드=====
  class Css_animation{ //=====CSS 효과 Class 선언=====
    constructor(first_percent = -30, second_percent = -200, roof_time = 1000){
    }
    get css_value(){ -
    }
    set css_value([set_percent_1, set_percent_2, set_time]){ -
    }
    css_input(line){ -
    };
    css_error(){ -
    };
    zoom_result(){ -
    }
    button_effect(){ -
    }
  }
}
```

CODE *JavaScript & jQuery*



행과 열 생성 값에 1~9사이의 숫자가
아니면 공백으로 전환 class method

```
reg_input(e){ //=====입력창 정규식 메소드=====//
  const input_doc = new RegExp(/[^\d-]/g, '');
  $(e).val( (input_doc.test($(e).val()) ? ('') : ($(e).val())));
  if(1 < $(e).val().length){
    $(e).val('');
  };
  $(e).attr('value', `${$(e).val()}`);
};
```

A

ROW :

COLUMN :

OUTPUT RANDOM

9,999	-999	0
-------	------	---

생성된 행렬이 -999 ~ 9,999 사이의
숫자가 아니면 공백으로 만들며
세 번째 자리에 콤마 생성되는
class method

```
reg_box(e, count){ //=====내부박스 정규식 메소드=====//
  const reg_doc = new RegExp(/^-?\d*$/gm);
  const reg_doc_sort = new RegExp(/^\d0|-0/gm, '');
  reg_doc_sort.test($(e).val()) && $(e).val('');
  let sort_reg = reg_doc.test($(e).val());
  if(sort_reg == false){
    $(e).val('');
  } else if(3 < $(e).val().length) {
    let num_reg = Number($(e).val());
    $(e).val(num_reg.toLocaleString('ko-KR'));
  }
  (count < $(e).val().length) && $(e).val('');
};
```

적용 시에도 1회 기존에 입력한 값을 보존

Figure 1 displays two grids of numbers. The left grid is a 3x3 matrix with the following values:

8,510	724	2,556
9,711	1,878	5,555
496	9,936	3,120

The right grid is a 7x7 matrix with the following values:

8,510	724	2,556	0	0	0	0
9,711	1,878	5,555	0	0	0	0
496	9,936	3,120	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Both grids are labeled 'A' and have 'ROW : 3' and 'COLUMN : 3' for the left grid, and 'ROW : 7' and 'COLUMN : 7' for the right grid. The grids are generated using a 'RANDOM' process, as indicated by the 'RANDOM' button.

```

create_inner(box, type, keep_data){ //=====내부박스 생성 메소드=====//
$(box).empty();
function css_inner(inner_root, large_box) {
    `(${box}>${inner_root}).css({width: `${100 / large_box}%`, height: "100%", fontSize: `${9 / large_box}rem`
    `inset ${0.6 / large_box}rem ${0.6 / large_box}rem 0rem #FFEFB6, ${0.6 / large_box}rem ${0.6 / large_box}rem
}
let row_box = Number(this._row) + 1.1;
let col_box = Number(this._column) + 1.1;
let i = 0;
while(i < this._row){
    $(box).append(`<div></div>`);
    (box != '#inner_result' && keep_data[i] == undefined) && keep_data.push([]); // 빈배열 행 추가(결과값 제외)
    if(this._row >= this._column){
        `(${box}>div).css("width", `100%`).css("height", `${100 / row_box}%`);
    }else {
        `(${box}>div).css("width", `100%`).css("height", `${100 / col_box}%`);
    };
    let j = 0;
    while(j < this._column){
        if(type == "input"){
            `(${box}>div:nth-child(${i+1}))`.append(`<input type="text" name="num_box" placeholder="0">`);
            if(keep_data[i][j] == undefined){
                `(${box}>div:nth-child(${i+1})>input:nth-child(${j+1}))[0].value = '0';
            } else {
                `(${box}>div:nth-child(${i+1})>input:nth-child(${j+1}))[0].value = keep_data[i][j];
            } // 기존 입력값 확인후 값 유지
            this.reg_box(`(${box}>div:nth-child(${i+1})>input:nth-child(${j+1}))[0]`, 5);
        }else {
            `(${box}>div:nth-child(${i+1}))`.append(`<div><input type="text" name="num_box" placeholder="0" readonl
        });
        if(this._row >= this._column){
            css_inner(`div>input`, row_box);
            css_inner(`div>div`, row_box);
            `(${box}>div>div input).css("font-size", `${9 / row_box}rem`);

```

행렬의 +, -, × 연산 Method

행렬의 +, -, × 연산 및 연산 된 값을
정규식을 통해 3자릿수 마다 콤마 생성

+

-

×

RESET

결과 자세히 보기 ▼

2,813	684	18,455	2,101	845
4,167	9,168	4,693	17,600	8,484
2,873	3,642	8,843	13,505	12,055
507	10,993	9,538	3,650	10,334
11,817	7,893	1,986	9,751	2,985

```
data_calculation(box, set_calculator, data_arr_a, data_arr_b){ //=====행렬연산 메소드=====
    this._data = [];
    let max_length = 0;
    max_length = ($(`#inner_first>div`).length > `${`#inner_second>div:first-child>input`}.length) ? (
        `${`#inner_first>div`}.length
    ) : (
        `${`#inner_second>div:first-child>input`}.length
    );
    `${`#detail_box>p`}.remove();
    let i = 0;
    while(i < `${`#inner_first>div`}.length){
        let j = 0;
        while(j < `${`#inner_second>div:first-child>input`}.length){
            if( set_calculator == 'plus' ){
                `${`${box}>div:nth-child(${i+1})>div:nth-child(${j+1})>input`}.attr('value', `${Number(data_arr_a[i][j])
            } else if( set_calculator == 'minus' ) {
                `${`${box}>div:nth-child(${i+1})>div:nth-child(${j+1})>input`}.attr('value', `${Number(data_arr_a[i][j])
            } else if (set_calculator == 'multiply') {
                let k = 0;
                let math_multiply = 0;
                while(k < `${`#inner_second>div`}.length){
                    math_multiply += ( Number(data_arr_a[i][k]) * Number(data_arr_b[k][j]) );
                    k++;
                }
                `${`${box}>div:nth-child(${i+1})>div:nth-child(${j+1})>input`}.attr('value', `${math_multiply}`);
            }
            let get_result = `${`${box}>div:nth-child(${i+1})>div:nth-child(${j+1})>input`}[0];
            this.reg_box(get_result, 99); //=====result 내부박스 정규식=====//
        }
    }
```



좌, 우로 움직이는 모션

마우스 오버 시 해당 값만 확대

8,370,643	38,306,007	55,160,717	94,535,145	65,715,612
30,590,206	155,674,990	86,416,688	131,028,246	89,013,009
19,300,943	86,406,653	65,772,314	106,762,353	71,561,011
18,149,423	50,483,771	29,172,551	76,843,009	26,653,721
15,808,118	64,395,256	113,574,047	71,414,046	39,264,253

결과 자세히 보기 버튼 클릭 시
전체 값만 따로 확인할 수 있는 창 생성

좌, 우로 움직이는 모션
마우스 오버 시 해당 값만 확대

```
zoom_result(){
  let keyframes_result = [
    { left : `${this.first_percent}%`, width: 'auto' },
    { left : `${this.second_percent}%`, width: 'auto' },
    { left : `${this.first_percent}%`, width: 'auto' }
  ];
  let result_timing = {
    duration: this.roof_time,
    iterations: Infinity
  };
  $(`#inner_result>div>div>input`).each(function(k, v){
    if(v.value.length > 6){
      v.animate(keyframes_result, result_timing);
      $(v).on('mouseenter', function(){
        $(v).parent().parent().parent().css('overflow', 'visible');
        $(v).parent().css('overflow', 'visible').css('z-index', 1);
        $(v).css({transform: 'scale(2.5)', textShadow: '0.1rem 0.1rem 0 #C44C51, 0 -0.1rem #C44C51, 0 0 0.5rem black'

```

결과 자세히 보기 버튼 클릭 시
전체 값만 따로 확인할 수 있는 창 생성

```
//=====결과값 자세히 보기=====
let show_flag = 1;
$(`#detail_result`).on('click', () => {
  if(show_flag){
    $(`#detail_box`).css({width: '114rem', height: '50rem',
    $(`html, body`).css({height: '115rem', overflowY: 'visible');
    $(`body, html`).animate({scrollTop: $(document).height()});
    $(`.close_detail`).css('visibility', 'visible');
    show_flag = 0;
  } else {
    $(`#detail_box`).css({width: '114rem', height: '0rem',
    $(`html, body`).css({height: '61rem', overflowY: 'hidden');
    $(`body, html`).animate({scrollTop: 0}, 1000);
    $(`.close_detail`).css('visibility', 'hidden');
    show_flag = 1;
  }
});
```

REVIEW



행렬 계산기 사이트를 제작하기에 앞서 행렬 계산을 할 때에 필요한 기능을 확인하고 사용자가 불편함을 느낄 수 있는 부분을 고려해 스케치 및 코드 작업을 하였습니다.

pug, sass를 활용해 사이트의 기본 구조작업을 빠르게 할 수 있었으며 flex 구조를 통해 레이아웃의 위치를 보다 손쉽게 잡을 수 있었습니다.

code 작성 완료 후 JavaScript & jQuery를 class로 정리하는 과정에서 겪은 Event의 중복 발생 및 값의 전달에 대한 문제를 해결함으로써 code의 logic 순서를 이해하는데 큰 도움이 되었습니다.

THANK YOU+-×

박경호