

CALCULATOR

이정임

개발 기간 : 2024.08.21 – 2024.08.30

목차

1. 계산기 소개

p. 3-10

2. JAVASCRIPT

p. 11-18

3. 후기

p. 19-21

1. 계산기 소개

1. 계산기 소개

[계산기 바로가기](#)

디자인 테마

사용자의 눈이 편안할 수 있도록 따뜻한 색을 사용하여 디자인을 맞췄습니다.

사용자의 편의성을 위해 가독성이 좋은 색을 사용했습니다.

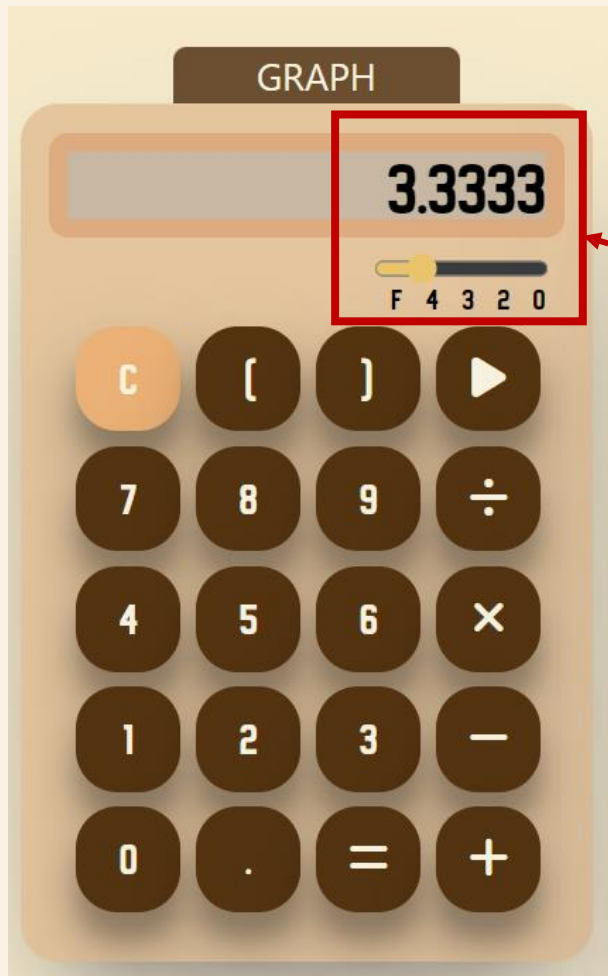
디자인 포인트

실제 계산기와 비슷한 느낌을 주기 위해 버튼과 배경에 입체감을 주었습니다.



1. 계산기 소개

[계산기 바로가기](#)



소수점의 자릿수를
사용자가 직접 설정하여
원하는 결과값을
얻을 수 있게 하였습니다.

잘못된 수식을 넣으면
Try again 이라는 오류가
나오고 수식을 다시 입력할
수 있게 하였습니다.

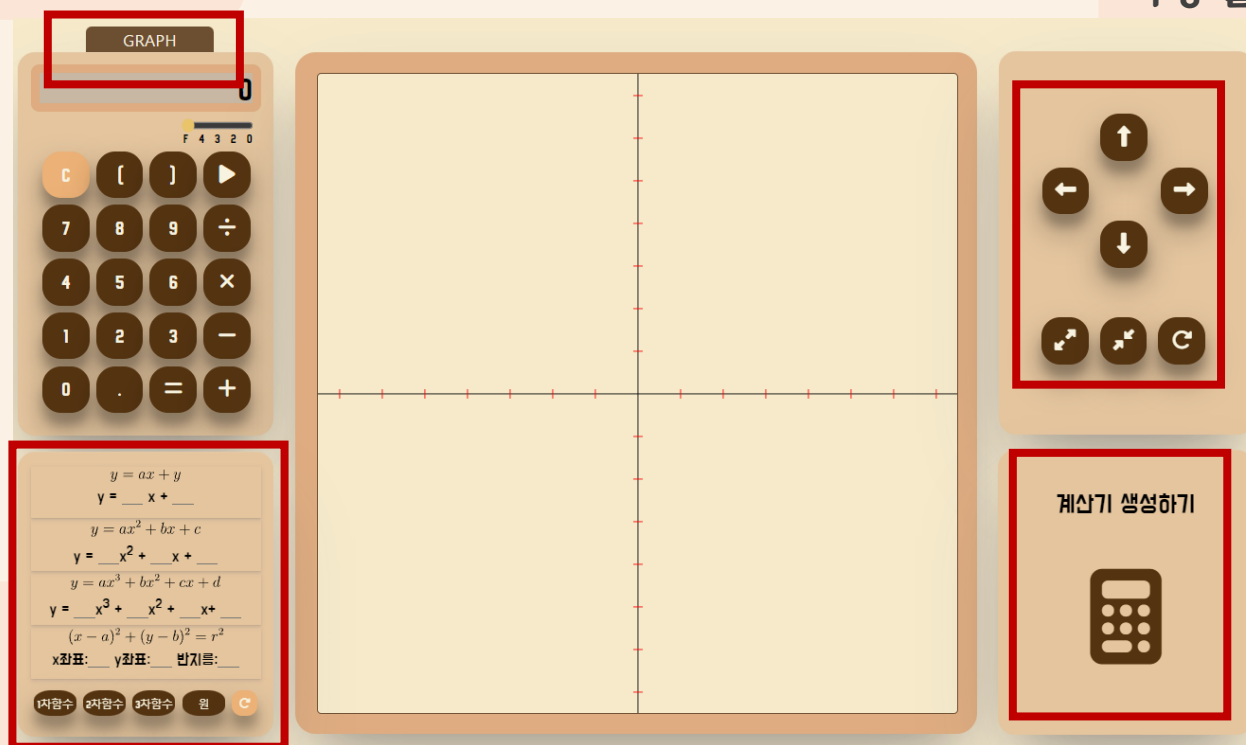


1. 계산기 소개

계산기 바로가기

GRAPH 버튼을 클릭하면
canvas 가 나옵니다

Canvas를
확대, 축소, x축, y축
이동을 할 수 있습니다.



좌표를 적고
함수 버튼을
누르면 그래프
가 생성됩니다

계산기 생성하기
버튼을 누르면
새로운 계산기가
생성됩니다.

1. 계산기 소개

계산기 바로가기

The calculator interface displays several input fields for mathematical formulas. A red dashed arrow points from the first two fields to the first text box, and a green dashed arrow points from the third and fourth fields to the second text box. An orange dashed arrow points from the reset button to the third text box.

$y = ax + y$
 $y = \underline{1} x + \underline{1}$

$y = ax^2 + bx + c$
 $y = \underline{2} x^2 + \underline{2} x + \underline{2}$

$y = ax^3 + bx^2 + cx + d$
 $y = \underline{3} x^3 + \underline{3} x^2 + \underline{3} x + \underline{3}$

$(x - a)^2 + (y - b)^2 = r^2$
x좌표: 2 y좌표: 2 반지름: 2

1차함수 2차함수 3차함수 원

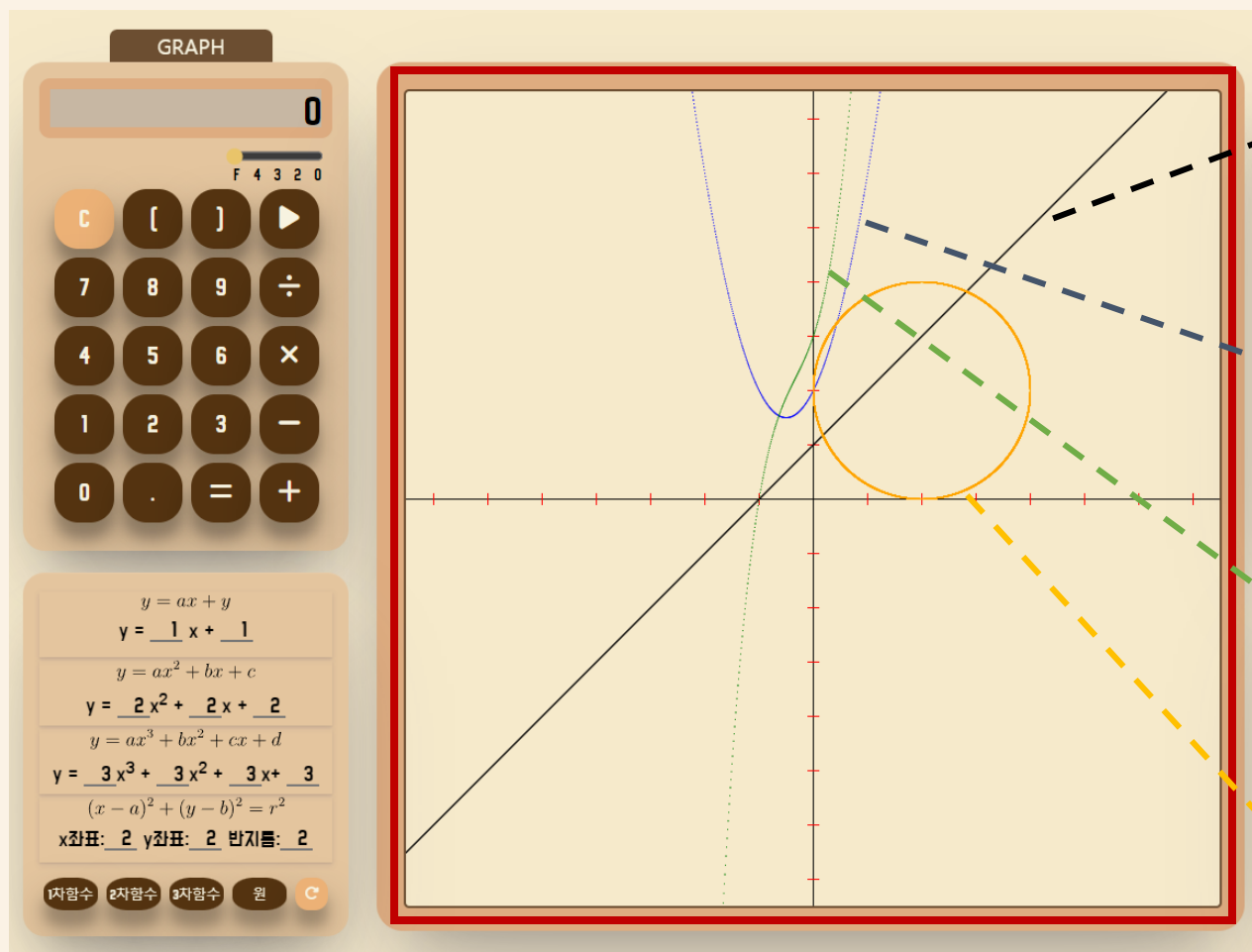
그리고 싶은 함수를
입력합니다.

버튼을 눌러서
Canvas에 함수를
출력합니다.

리셋 버튼을 누르면
입력했던 값과
Canvas 위에 출력되었던
함수를 모두 지웁니다.

1. 계산기 소개

계산기 바로가기



1차 함수는 검은색으로 나오게 하였습니다.

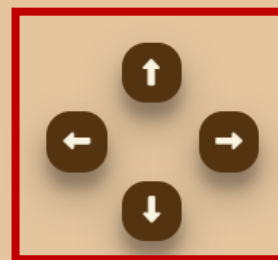
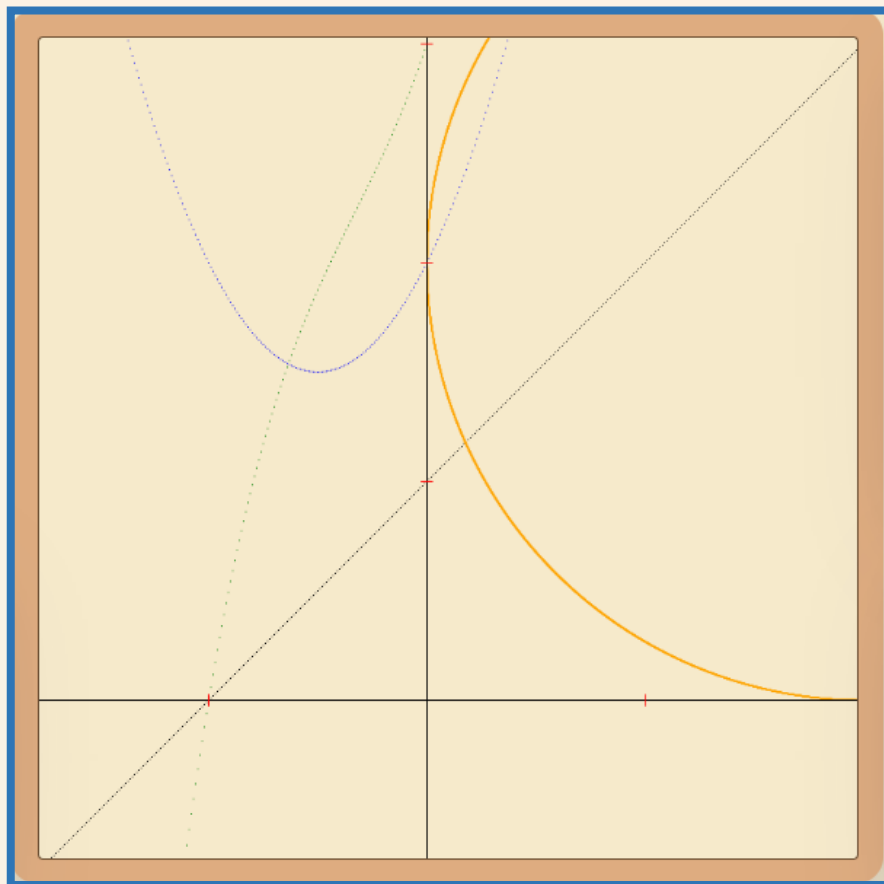
2차 함수는 파란색으로 나오게 하였습니다.

3차 함수는 초록색으로 나오게 하였습니다.

원은 주황색으로 나오게 하였습니다.

1. 계산기 소개

계산기 바로가기



자세하게 보고 싶은
함수 위치로 이동시킵니다.



그래프를 확대, 축소하여
자세하게 볼 수 있습니다.

계산기 생성하기

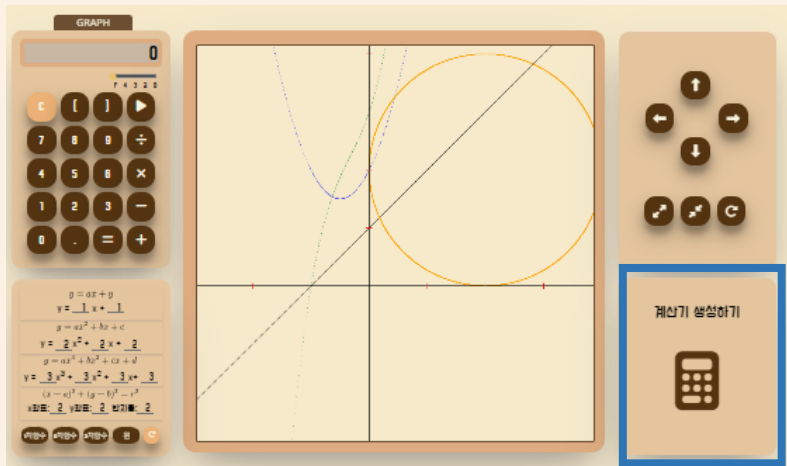


리셋 버튼을 누르면
원래의 그래프 크기와
위치로 돌아옵니다.

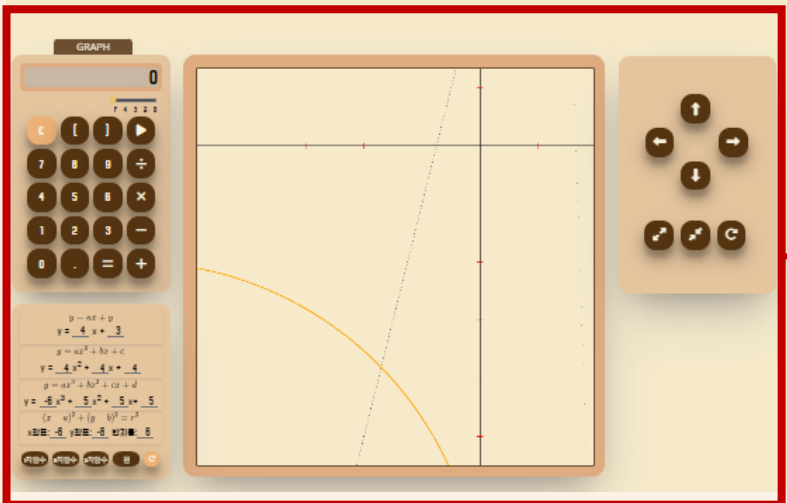
크게 확대한 그래프의
모습입니다.

1. 계산기 소개

계산기 바로가기



계산기 생성하기 버튼을 누르면
아래에 새로운 계산기가
생성됩니다.



새롭게 생성된 계산기와
위쪽 계산기는 따로 동작하기
때문에 그래프를 비교하며
볼 수 있습니다.



2. JAVASCRIPT

2. JAVASCRIPT

계산기 바로가기

```
1 <body>
2   <div id="CalculatorArea">
3     <div id="Calculator"></div>
4   </div>
5   <script>
6     class RealCalculator {
7       constructor(id) {
8         this.id = id;
9       }
10      displayCalculator(targetDom) {
11        let HTML = `
12          <!--html 시작 부분-->
```

Body에 계산기의 영역을 지정하여 그 안에 순서대로 생성되도록 하였습니다.

Class 를 사용하여 계산기의 복제를 용이하게 하였습니다.

HTML 부분과 CSS 부분을 javascript 안에 넣어 웹컴포넌트 방식을 사용해 보았습니다.

2. JAVASCRIPT

계산기 바로가기

중첩클래스를 이용하여
Deck 을 만들었습니다.

```
//Deck 영역
1  inputData() {
2    class Deck {
3      constructor(id) {
4        this.id = id;
5        this.storage = [];
6      }
7    }
8    pushItem(item) {
9      this.storage.push(item);
10   }
11   shiftItem() {
12     return this.storage.shift();
13   }
14   popItem() {
15     return this.storage.pop();
16   }
17 }
18 let tempString = "";
19 const CalculatorDeck = new Deck(Calculator);
```



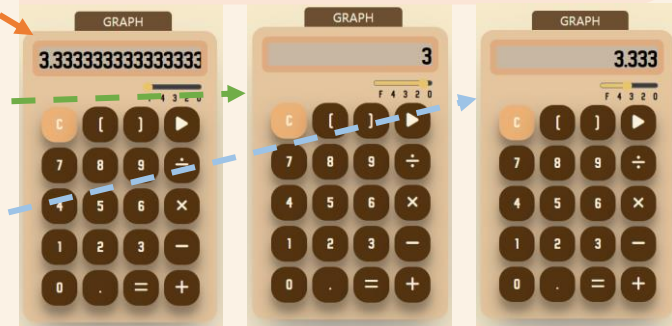
새로운 계산기가 생성 될 때
Class도 새로 만들어 집니다.

2. JAVASCRIPT

계산기 바로가기

```
1 range.addEventListener("input", (e) => {  
2     switch (range.value) {  
3         case "0":  
4             this.displayData(eval(tempString));  
5             break;  
6         case "1":  
7             this.displayData(eval(tempString).toFixed(4));  
8             break;  
9         case "2":  
10            this.displayData(eval(tempString).toFixed(3));
```

Switch, toFixed 를
이용해서 소수점 자릿수를
설정하였습니다.



```
    catch (err) {  
        this.displayData("Try again");  
    }
```

Try catch 를 이용하여
수식 오류를 잡을 수 있게
하였습니다.

2. JAVASCRIPT

계산기 바로가기

```
canvasControl() {  
  class MakeCanvas {  
    constructor(id) {  
      this.id = id;  
      this.canvas = document.getElementById(`canvas${this.id}`);  
      this.pen = this.canvas.getContext("2d");  
      this.canvasWidth = this.canvas.width;  
      this.canvasHeight = this.canvas.height;  
      this.canvasWidthHalf = this.canvas.width / 2;  
      this.canvasHeightHalf = this.canvas.height / 2;  
      this.scaleFactor = 50;  
      this.basicY = 0;  
      this.basicX = 0;  
    }  
  }  
}
```

중첩클래스를
이용하여
Canvas를
control
하는 부분을
나누었습니다.

Canvas를 축소 확대 하거나 이동할 때
x 좌표, y 좌표, scaleFactor를 조정하여 canvas를
그리는 방식으로 설계하였습니다.

2. JAVASCRIPT

계산기 바로가기

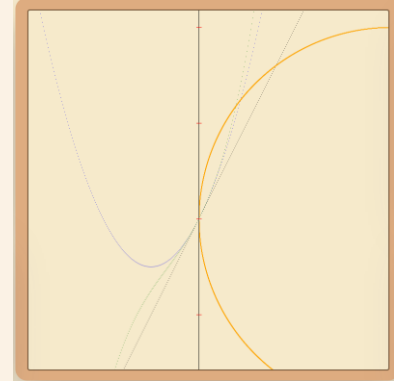
```
1 document
2 .getElementById(`moveBtn3${this.id}`)
3 .addEventListener("click", () => {
4   this.basicX += 10;
5   if (this.basicX > 490) {
6     this.basicX = 490;
7   }
8 }
```

```
1 document
2 .getElementById(`moveBtn5${this.id}`)
3 .addEventListener("click", () => {
4   this.scaleFactor += 10;
5   if (this.scaleFactor > 200) {
6     this.scaleFactor = 200;
7   }
8 }
```

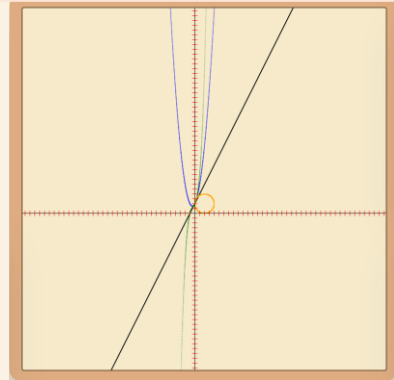
```
1 document
2 .getElementById(`moveBtn2${this.id}`)
3 .addEventListener("click", () => {
4   this.basicX -= 10;
5   if (this.basicX < -490) {
6     this.basicX = -490;
7   }
8 }
```

x 값과 y 값, scaleFactor
값을 증가, 감소를 설정하여
그래프가 무한대로 가지 않게
막았습니다.

최대 scaleFactor



최소 scaleFactor



2. JAVASCRIPT

계산기 바로가기

```
1 document
2   .getElementById(`moveBtn7${this.id}`)
3   .addEventListener("click", () => {
4     this.scaleFactor = 50;
5     this.basicY = 0;
6     this.basicX = 0;
7     this.pen.clearRect(
8       0,
9       0,
10      this.canvas.width,
11      this.canvas.height
12    );
13    this.equation_1();
14    this.equation_2();
15    this.equation_3();
16    this.equation_4();
17    this.printCross();
18    this.printScaleFactor();
19  });
20 }
```

리셋 버튼을 누르면
x 값과 y 값, scaleFactor 값을
초기화 해줍니다.

그래프를 다시 그리는 함수를
호출해서 원래 초기 상태의
그래프로 돌아가게 됩니다.

2. JAVASCRIPT

계산기 바로가기

```
1 class NewCalculator {
2   constructor(id) {
3     this.id = id;
4     this.indexId = 1;
5     this.tempArray = [];
6   }
7   makeCalculatorControl() {
8     const CalculatorArea = document.getElementById("CalculatorArea");
9     document
10      .getElementById("makeCalculBtn")
11      .addEventListener("click", () => {
12        this.indexId++;
13        const calcArea = document.createElement("div");
14        calcArea.id = `calcArea${this.indexId}`;
15        //CalculatorArea.appendChild(calcArea);
16        document.body.appendChild(calcArea);
17        const cal = new RealCalculator(`Calculator${this.indexId}`);
18        cal.controlFun(
19          document.getElementById(`calcArea${this.indexId}`)
20        );
21        this.tempArray.push(cal);
22      });
23   }
24 }
```

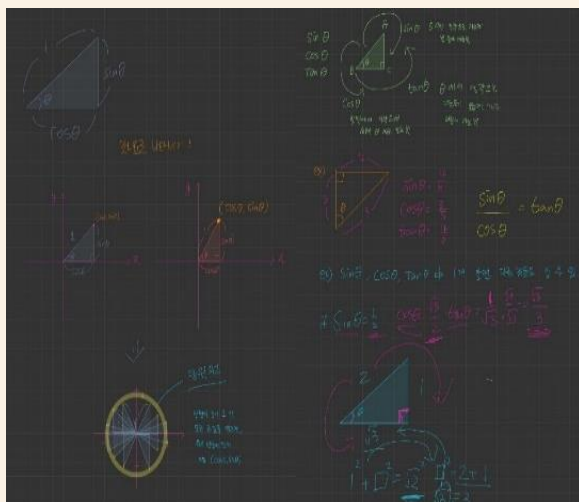
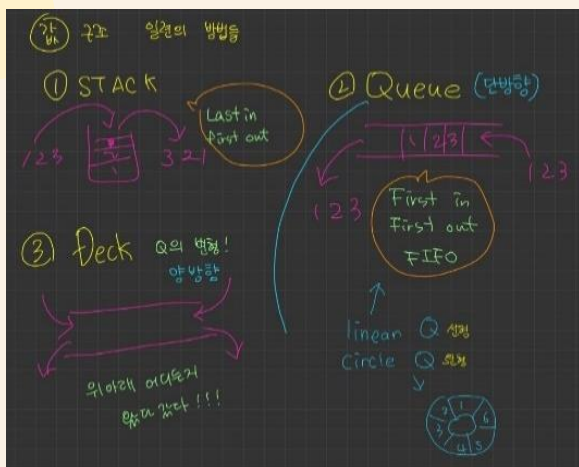
계산기 생성 버튼을
누르면 새로운 class 가
생기도록 하였습니다.

Class의 id 를 다르게
설정하기 위해서
Index Id를 증가시켜
Id 중복을 막았습니다.

3. 후기

3. 후기

계산기 바로가기



구조화 능력 향상

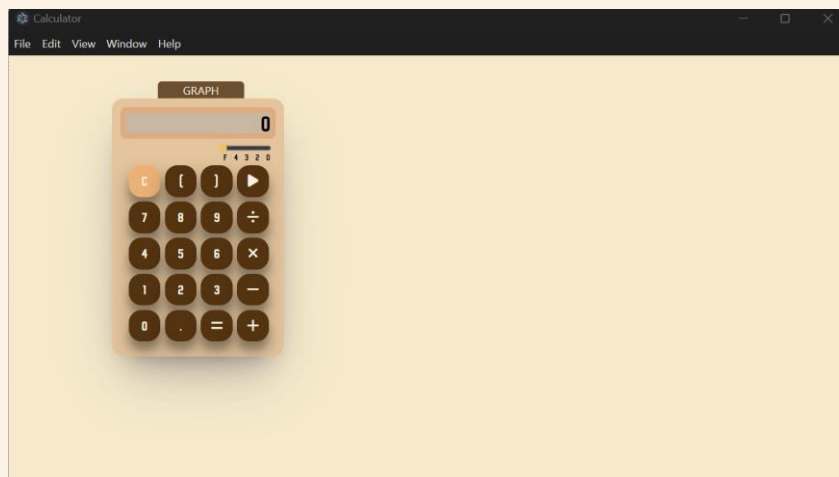
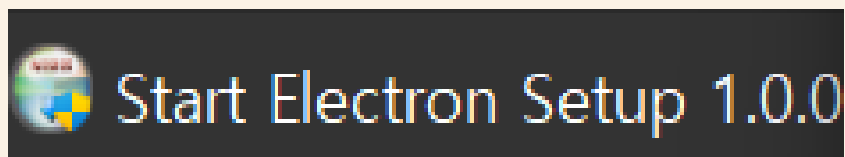
deck 구조를 사용해본 경험을 통해 자료 구조화의 중요성을 알게 되었고, 앞으로도 다양한 자료 구조(예: Stack, Queue, Deck 등)를 학습하고 실제 프로젝트에 적용할 계획입니다. 이를 통해 데이터를 더 효율적으로 관리하고 처리할 수 있는 능력을 기르며, 나아가 더 복잡한 프로그램 개발에도 도전할 예정입니다.

계속적인 자기 개선

중학교 때 배운 수학(Sin, Cos 등)을 다시 복습한 것처럼, 앞으로도 학습했던 내용들을 주기적으로 복습하고 실전에 적용하는 연습을 계속할 것이고 모르는 부분이 생기면 적극적으로 공부할 것입니다. 이를 통해 기존 지식을 더 깊이 있게 이해하고 새로운 개념을 습득하는 데 도움을 줄 수 있는 학습 루틴을 구축할 것입니다.

3. 후기

계산기 바로가기




Electron으로 만들기

처음엔 브라우저만을 통해 프로그램을 실행했지만, Electron을 활용한 실행에 도전하게 되었습니다. Node.js에 대한 기초 지식만으로 Electron을 구동하는 것은 예상보다 복잡하고 어려웠지만, 그 과정에서 스스로의 한계를 극복할 수 있는 귀중한 기회가 되었다고 생각합니다.

앞으로의 계획

Electron의 내부 구조와 기능을 더 깊이 이해하여, 더 복잡한 데스크탑 애플리케이션을 개발할 수 있는 역량을 강화하겠습니다.

Node.js의 비동기 처리와 서버 관리 능력을 더욱 향상시켜, 백엔드와 프론트엔드의 통합된 애플리케이션을 원활하게 구현하겠습니다. 이러한 목표들을 통해 더욱 완성도 높은 애플리케이션을 개발하고, 나아가 개발자로서 한 단계 더 성장하고자 합니다.



감사합니다.