1. 요구조건 0단계

1-1. 구현요약

구현사항		구현여부
위젯 관련 코드 입력	Widget, Text, Image, PushButton, TextField	O
	Switch	O
로그인 페이지 구성	initWidgets()	O
	layoutWidgets	O

1-2. 구현 방법

1-2-1. 위젯관련코드 입력(Widget, Text, Image, PushButton, TextField)

- 강의 교안 코드 참고하여 그대로 사용.

1-2-2. 위젯관련코드 입력(Switch)

```
// Switch widget
WebUI.Switch = function(is_on, desired_size) {
    // IMPLEMENT HERE!
    WebUI.Widget.call(this);
    this.type = WebUI.WidgetTypes.SWITCH;
    this.is_on = false;
    this.desired_size = desired_size;
    this.fill_color = 'rgb(142,142,147)';
    this.stroke_color = 'rgb(142,142,147)';

    this.margin = 10;
}

WebUI.Switch.prototype = Object.create(WebUI.Widget.prototype);
WebUI.Switch.prototype.constructor = WebUI.Switch;
```

- Switch 버튼을 만들 때 필요한 요소들을 정의.

```
WebUI.Switch.prototype.initVisualItems = function() {
    // IMPLEMENT HERE!
    let boundary = new fabric.Rect({
        left: this.position.left + this.desired_size.width/4,
        top: this.position.top,
        width: this.desired_size.width/2,
        height: this.desired_size.height,
        fill: this.fill_color,
        stroke: this.stroke_color,
        selectable: false
    });

    let left_Circle = new fabric.Circle({
        left: this.position.left,
        top: this.position.top,
        radius: this.desired_size.width/4,
        fill: this.fill_color,
        stroke: this.stroke_color,
        selectable: false
    });

    let right_Circle = new fabric.Circle({
        left: this.position.left + this.desired_size.width/2,
        top: this.position.top,
        radius: this.desired_size.width/4,
        fill: this.fill_color,
        stroke: this.stroke_color,
        selectable: false
    });

    let switch_Circle = new fabric.Circle({
        left: this.position.left + 0.1 * (this.desired_size.width/4),
        top: this.position.top + 0.1 * (this.desired_size.width/4),
        radius: 0.9 * (this.desired_size.width/4),
        fill: 'white',
        stroke: 'white',
        selectable: false
    });
```

```
    this.size = this.desired_size;

    this.visual_items.push(left_Circle);
    this.visual_items.push(right_Circle);
    this.visual_items.push(boundary);
    this.visual_items.push(switch_Circle);
    this.is_resource_ready = true;
}
```

- Switch 버튼 모양 정의. 가운데 정사각형을 그리고 양쪽에 원2개를 그림.

- 반지름이 0.9인 이동버튼을 그림.

```
WebUI.Switch.prototype.handleMouseDown = function() {
    // IMPLEMENT HERE!
    if (!this.is_on) {
        this.visual_items[0].set('fill', 'rgb(48,209,88)');
        this.visual_items[0].set('stroke', 'rgb(48,209,88)');
        this.visual_items[1].set('fill', 'rgb(48,209,88)');
        this.visual_items[1].set('stroke', 'rgb(48,209,88)');
        this.visual_items[2].set('fill', 'rgb(48,209,88)');
        this.visual_items[2].set('stroke', 'rgb(48,209,88)');
        this.visual_items[3].set('stroke', 'rgb(48,209,88');
        this.visual_items[3].animate('left', '+=50', {
            onChange: WebUI.canvas.renderAll.bind(WebUI.canvas),
            duration: 100,
            easing: fabric.util.ease.easeOutBounce
        });
        this.is_on = true;
    }

    else{
        this.visual_items[0].set('fill', 'rgb(142,142,147)');
        this.visual_items[0].set('stroke', 'rgb(142,142,147)');
        this.visual_items[1].set('fill', 'rgb(142,142,147)');
        this.visual_items[1].set('stroke', 'rgb(142,142,147)');
        this.visual_items[2].set('fill', 'rgb(142,142,147)');
        this.visual_items[2].set('stroke', 'rgb(142,142,147)');
        this.visual_items[3].set('stroke', 'rgb(142,142,147');
        this.visual_items[3].animate('left', '-=50', {
            onChange: WebUI.canvas.renderAll.bind(WebUI.canvas),
            duration: 100,
            easing: fabric.util.ease.easeOutBounce
        });

        this.is_on = false;
    }
    return true;
}
```

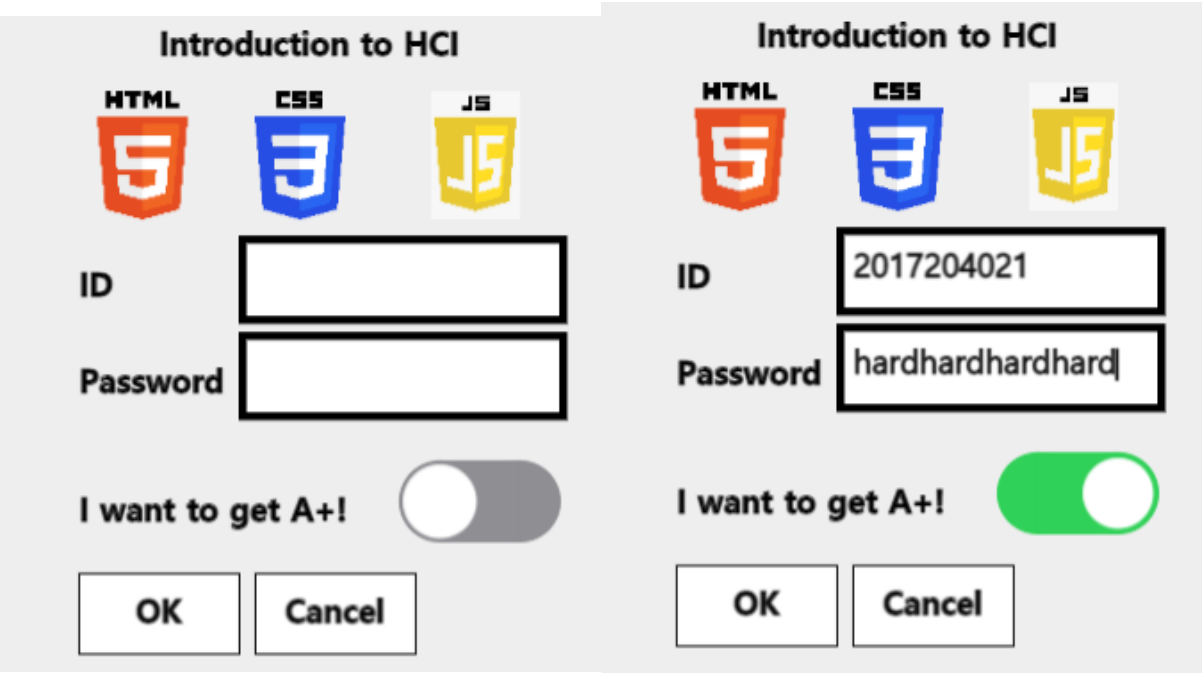
- Switch버튼의 이동이벤트를 정의.

- is_on을 확인하여 버튼 색상 변경과 버튼의 이동을 구현해줌.

1-2-3. 로그인 페이지 구성

- 강의 교안 코드 참고하여 그대로 사용.

1-3. 최종결과



- 왼쪽은 스위치가 꺼진화면, 오른쪽은 켜진화면.

- ID와 Password를 입력받을 시, 보여지는 공간 이상으로는 입력할 수 없고, 엔터키를 통해 빠져나올 수 있음.

- 하지만 한글입력시 공간 이상으로 입력 가능.

- OK버튼과 Cancel버튼은 마우스를 올렸을 때, 테두리가 두꺼워지고, 클릭시 아래로 내려가는 시각효과가 들어가 있음.

2. 요구조건 1단계

2-1. 구현요약

구현사항		구현여부
위젯 관련 코드 복사 (0단계와 동일)		O
레이아웃 관련 코드 입력	WidgetTypes, Alignment, Widget, maxSize, minSize…	O
로그인 페이지 구성	initWidgets()	O
	Container, Row, Column	O

2-2. 구현 방법

2-2-1. 위젯 관련 코드 입력(0단계와 동일)

- 0단계와 동일한 과정.

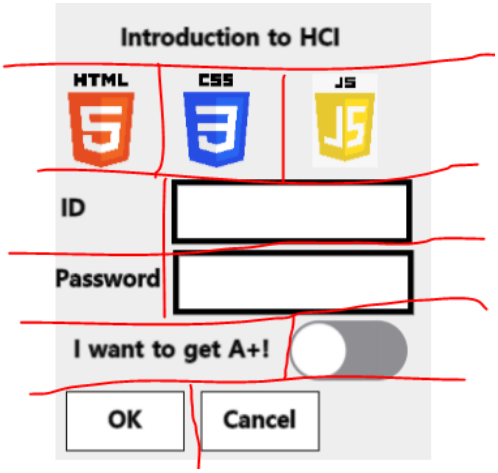
2-2-2. 레이아웃 관련 코드 입력

- 강의 교안 코드 참고하여 그대로 사용.

2-2-3. 로그인 페이지 구성 구현 코드

```
WebUI.initWidgets = function() {
    // INITIALIZE WIDGETS HERE
    WebUI.app = new WebUI.Row({
        children: [
            new WebUI.Container({
                desired_size: {width: 300, height: 60},
                horizontal_alignment: WebUI.Alignment.CENTER,
                vertical_alignment: WebUI.Alignment.CENTER,
                children: [ new WebUI.Text("Introduction to HCI") ],
            }),

            new WebUI.Column({
                children: [
                    new WebUI.Image("resources/HTML5.png",
                        {width: 100, height: 80}),
                    new WebUI.Image("resources/CSS3.png",
                        {width: 100, height: 80}),
                    new WebUI.Image("resources/JS.png",
                        {width: 100, height: 80})
                ]
            }),

            new WebUI.Column({
                children: [
                    new WebUI.Container({
                        desired_size: {width: 100, height: 50},
                        horizontal_alignment: WebUI.Alignment.LEFT,
                        vertical_alignment: WebUI.Alignment.CENTER,
                        children: [ new WebUI.Text("ID") ],
                    }),
                    new WebUI.TextField("", {width: 200, height: 50})
                ]
            }),

            new WebUI.Column({
                children: [
                    new WebUI.Container({
                        desired_size: {width: 100, height: 50},
                        horizontal_alignment: WebUI.Alignment.LEFT,
                        vertical_alignment: WebUI.Alignment.CENTER,
                        children: [ new WebUI.Text("Password") ],
                    }),
                    new WebUI.TextField("", {width: 200, height: 50})
                ]
            }),

            new WebUI.Column({
                children: [
                    new WebUI.Container({
                        desired_size: {width: 200, height: 50},
                        horizontal_alignment: WebUI.Alignment.CENTER,
                        vertical_alignment: WebUI.Alignment.CENTER,
                        children: [ new WebUI.Text("I want to get A+!")
                    }),
                    new WebUI.Container({
                        desired_size: {width: 100, height: 50},
                        horizontal_alignment: WebUI.Alignment.CENTER,
                        vertical_alignment: WebUI.Alignment.CENTER,
                        children: [ new WebUI.Switch(false, {width: 100
                    })
                ]
            }),

            new WebUI.Column({
                children: [
                    new WebUI.Container({
                        desired_size: {width: 120, height: 50},
                        horizontal_alignment: WebUI.Alignment.CENTER,
                        vertical_alignment: WebUI.Alignment.CENTER,
                        children: [new WebUI.PushButton("OK", {width: 1
                    }),
                    new WebUI.Container({
                        desired_size: {width: 120, height: 50},
                        horizontal_alignment: WebUI.Alignment.LEFT,
                        vertical_alignment: WebUI.Alignment.CENTER,
                        children: [ new WebUI.PushButton("Cancel", {widt
                    })
                ]
            })
        ]
    });
}
```

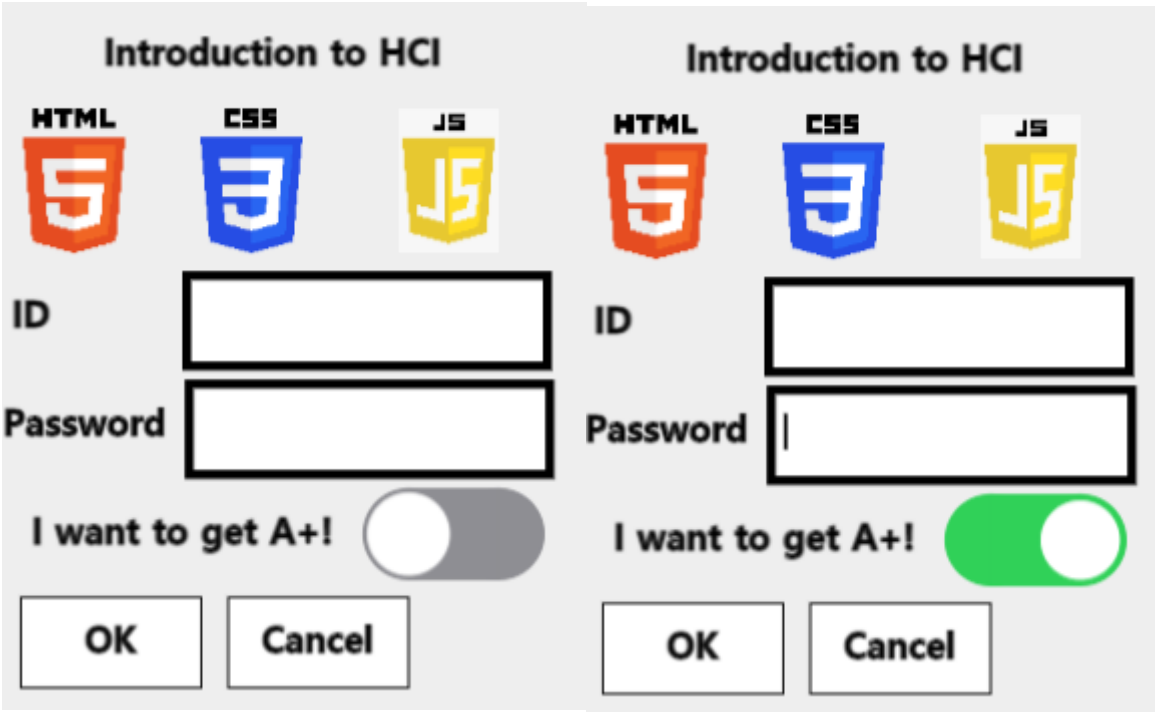
2-2-4. 로그인 페이지 레이아웃 구성



- 6개의 Row로 나눈후, 각각 1,3,2,2,2,2개의 Column으로 각각 나눠 배치하였음.

- 각각의 요소들을 container안에 넣어 구역내에서 가운데 정렬을 해줌.

2-3. 최종결과



- 왼쪽은 스위치가 꺼진화면, 오른쪽은 켜진화면.

- ID와 Password를 입력받을 시, 보여지는 공간 이상으로는 입력할 수 없고, 엔터키를 통해 빠져나올 수 있음.

- 하지만 한글입력시 공간 이상으로 입력 가능.

- OK버튼과 Cancel버튼은 마우스를 올렸을 때, 테두리가 두꺼워지고, 클릭시 아래로 내려가는 시각효과가 들어가 있음.

- 직접 좌표를 지정해주어야 하는 0단계와 달리 전체적 틀만 정해서 편하게 구현할 수 있었음.

3. 요구조건 2단계

3-1. 구현요약

구현사항		구현여부
수식 계산을 위한 객체 정의		O
CalcButton 클래스 정의	PushButton으로부터 상속	O
	handleButtonPushed 멤버 함수 추가	O
	this.onPushed코드 추가	O
initWidgets()함수 재정의	제목: Container > Text	O
	화면: Container > Text	O
	버튼: Row > Column > CalcButton	O

3-2. 구현 방법

3-2-1. 수식 계산을 위한 객체 정의

```
WebUI.parser = math.parser();
```

- WebUI.parser = math.parser();를 이용한 객체 정의.

3-2-2. CalcButton 클래스 PushButton으로부터 상속

```
WebUI.CalcButton = function(label, desired_size){
    WebUI.PushButton.call(this);
    this.type = WebUI.WidgetTypes.CALC_BUTTON;
    this.label = label;
    this.desired_size = desired_size;
    this.is_pushed = false;
    this.stroke_color = 'black';
    this.fill_color = 'white';
    this.onPushed = WebUI.CalcButton.handleButtonPushed;
}

WebUI.CalcButton.prototype = Object.create(WebUI.PushButton.prototype);
WebUI.CalcButton.prototype.constructor = WebUI.CalcButton;
```

- 0~1단계의 PushButton으로부터 상속받아 CalcButton 클래스 생성.

3-2-3. CalcButton 클래스 handleButtonPushed 멤버 함수 추가

```
WebUI.CalcButton = function(label, desired_size){
    WebUI.PushButton.call(this);
    this.type = WebUI.WidgetTypes.CALC_BUTTON;
    this.label = label;
    this.desired_size = desired_size;
    this.is_pushed = false;
    this.stroke_color = 'black';
    this.fill_color = 'white';
    this.onPushed = WebUI.CalcButton.handleButtonPushed;
}
```

- this.onPushed = WebUI.CalcButton.handleButtonPushed; 로 멤버함수를 추가해줌.

3-2-4. CalcButton 클래스 this.onPushed 코드 추가

```
WebUI.CalcButton.handleButtonPushed = function() {
    let parser = math.parser();

    result = WebUI.calc_result.label;
    if(result == "0"){
        result = "";
    }
    if(this.label == "EV"){
        try{
            result = parser.eval(WebUI.calc_result.label).toString();
            var tokens = result.split(' ');
            if(tokens[0] == "function")
            {
                result = tokens[0];
            }
        }
        catch(e)
        {
            result = "ERROR"
        }
    }

    else {
        if (this.label == "CL"){
            result = "0";
        }
        else {
            result += this.label;
        }

    }

    WebUI.calc_result.setLabel(result);

};
```

- EV버튼 입력시 수식 계산결과 반환, CL버튼 입력시 결과값을 다시 0으로 바꿈.

- 계산 오류시 ERROR 출력.

```
WebUI.Text.prototype.setLabel = function(new_label) {
    let text = this.visual_items[0];
    text.set('text', new_label);

    this.label = new_label;

    WebUI.canvas.requestRenderAll();
}
```

- 결과값 라벨 반환은 Text.setLabel함수를 이용한 반환.

3-2-6. initWidgets() 재정의

```
new WebUI.Container({
    desired_size: {width: 600, height: 60},
    horizontal_alignment: WebUI.Alignment.CENTER,
    vertical_alignment: WebUI.Alignment.CENTER,
    children: [ new WebUI.Text("WebUI Calculator", 40, "blue") ],
}),
```

- 제목 : Container > Text로 정의해줌.

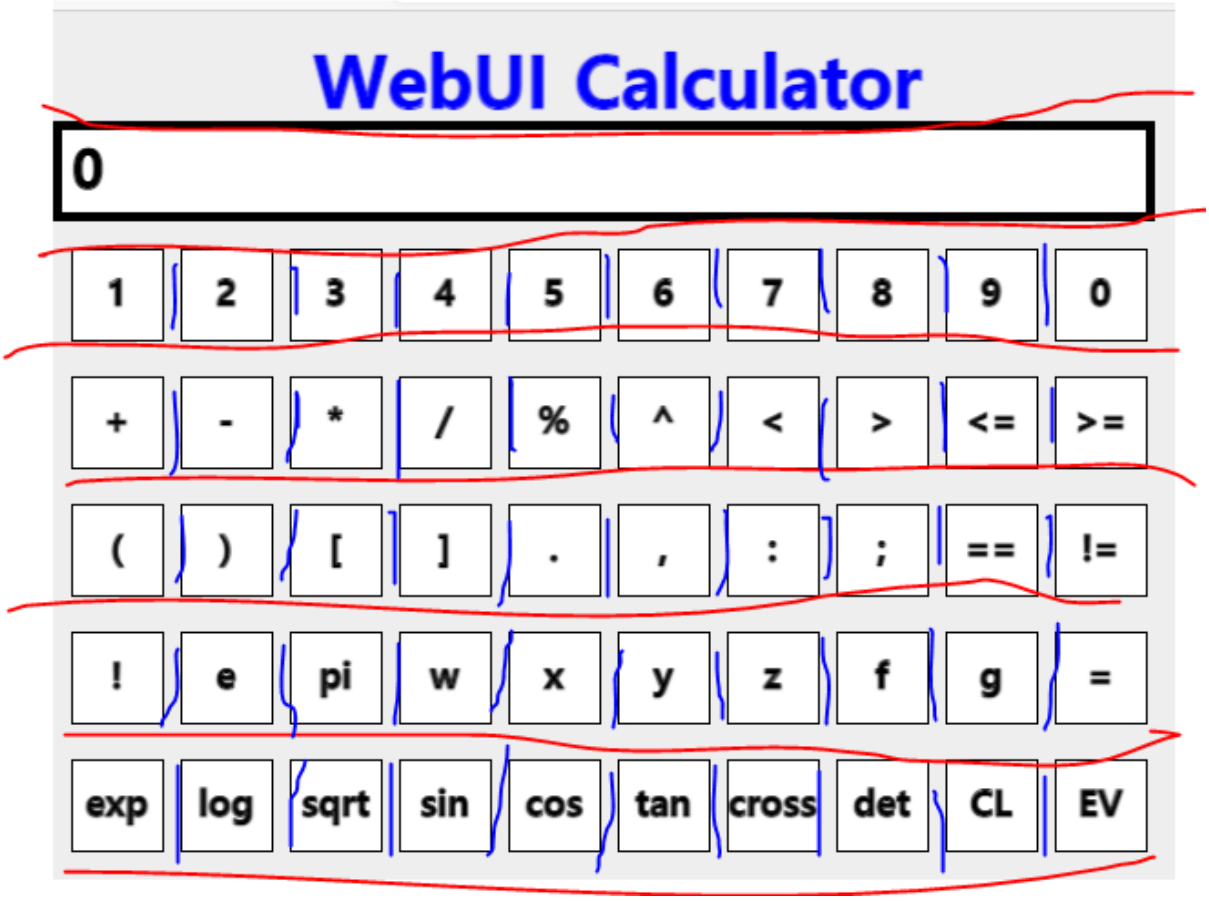
```
WebUI.calc_result = new WebUI.Text(final_result, 30, "black")
WebUI.app = new WebUI.Row({
    children: [
        new WebUI.Container({
            desired_size: {width: 600, height: 60},
            horizontal_alignment: WebUI.Alignment.CENTER,
            vertical_alignment: WebUI.Alignment.CENTER,
            children: [ new WebUI.Text("WebUI Calculator", 40, "blue") ],
        }),

        new WebUI.Container({
            desired_size: {width: 600, height: 60},
            horizontal_alignment: WebUI.Alignment.CENTER,
            vertical_alignment: WebUI.Alignment.CENTER,
            children: [ new WebUI.Resultbox("", {width: 600, height: 50})
                WebUI.calc_result]
        }),
```

- 화면 수식 : ResultBox 위젯을 따로 만들어줬었고, 그 안쪽에 Container > Text로 수식이 나오는

화면을 정의해줌.

```
WebUI.app = new WebUI.Row({
    children: [
        new WebUI.Container({
            desired_size: {width: 600, height: 60},
            horizontal_alignment: WebUI.Alignment.CENTER,
            vertical_alignment: WebUI.Alignment.CENTER,
            children: [ new WebUI.Text("WebUI Calculator", 40, "blue") ],
        }),

        new WebUI.Container({
            desired_size: {width: 600, height: 60},
            horizontal_alignment: WebUI.Alignment.CENTER,
            vertical_alignment: WebUI.Alignment.CENTER,
            children: [ new WebUI.Resultbox("", {width: 600, height: 50})
                WebUI.calc_result]
        }),

        new WebUI.Column({
            children: [
                new WebUI.Container({
                    desired_size: {width: 100, height: 70},
                    horizontal_alignment: WebUI.Alignment.CENTER,
                    vertical_alignment: WebUI.Alignment.CENTER,
                    vertical_alignment: WebUI.Alignment.CENTER,
                    children:[]
                }),
                new WebUI.Container({
                    desired_size: {width: 50, height: 70},
                    horizontal_alignment: WebUI.Alignment.CENTER,
                    vertical_alignment: WebUI.Alignment.CENTER,
                    children: [new WebUI.CalcButton("1", {width: 50, heig
                }),
                new WebUI.Container({
                    desired_size: {width: 50, height: 70},
                    horizontal_alignment: WebUI.Alignment.CENTER,
                    vertical_alignment: WebUI.Alignment.CENTER,
                    children: [ new WebUI.CalcButton("2", {width: 50, hei
                }),
                new WebUI.Container({
                    desired_size: {width: 50, height: 70},
                    horizontal_alignment: WebUI.Alignment.CENTER,
                    vertical_alignment: WebUI.Alignment.CENTER,
                    children: [ new WebUI.CalcButton("3", {width: 50, hei
                }),
```

- 버튼 : Row > Column > Container > CalcButton으로 입력버튼을 정의해주었다.

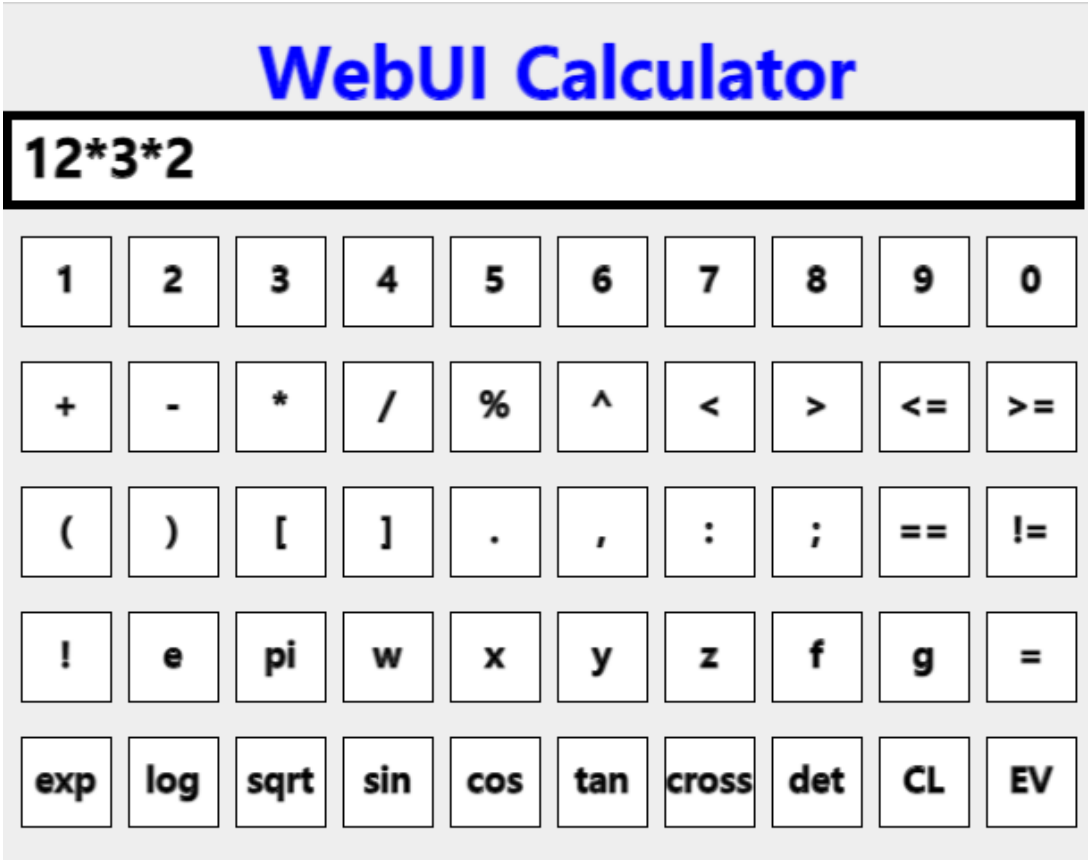


- 전체를 빨간색 선 기준으로 Row로 나눈후, 파란색 선 기준으로 Calcbutton들을 Column에 Container안에 넣어 레이아웃을 구성하였음.

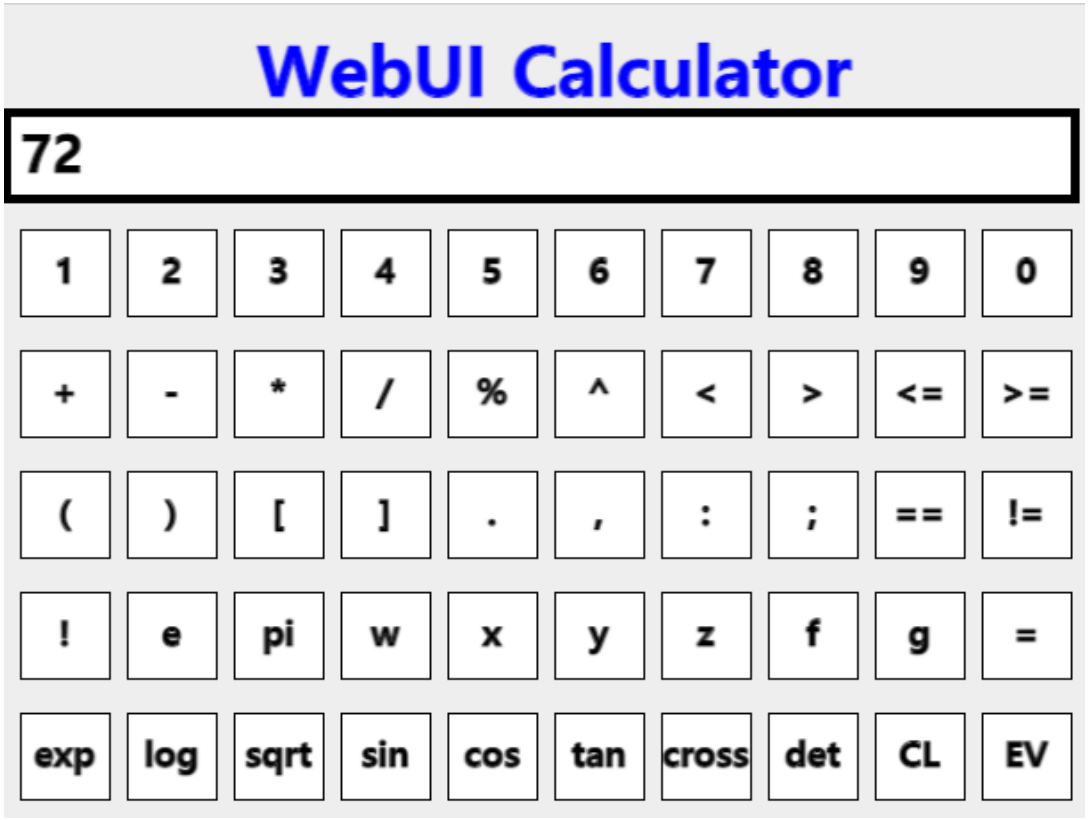
```
WebUI.Resultbox = function(label, desired_size) {
    WebUI.Widget.call(this);

    this.type = WebUI.WidgetTypes.TEXT_FIELD;
    this.label = label;
    this.desired_size = desired_size;
    this.margin = 10;

    this.stroke_color = 'black';
    this.fill_color = 'white';
    this.stroke_width = 5;

    this.font_family = 'System';
    this.font_size = 20;
    this.font_weight = 'normal';
    this.text_align = 'left';
    this.text_color = 'black';
}

WebUI.Resultbox.prototype = Object.create(WebUI.Widget.prototype);
WebUI.Resultbox.prototype.constructor = WebUI.TextField;

WebUI.Resultbox.prototype.initVisualItems = function() {
    let boundary = new fabric.Rect({
        left: this.position.left,
        top: this.position.top,
        width: this.desired_size.width,
        height: this.desired_size.height,
        fill: this.fill_color,
        stroke: this.stroke_color,
        strokeWidth: this.stroke_width,
        selectable: false
    });

    this.size = this.desired_size;

    //
    this.visual_items.push(boundary);
    this.is_resource_ready = true;
}
```

- 수식 결과 텍스트박스를 표시해줄 Resultbox위젯. 3단계에서도 사용.

3-3. 최종결과



- 마우스 버튼 클릭을 이용한 수식 입력과 EV눌러 결과를 출력한 계산기의 모습-



4. 요구조건 3단계

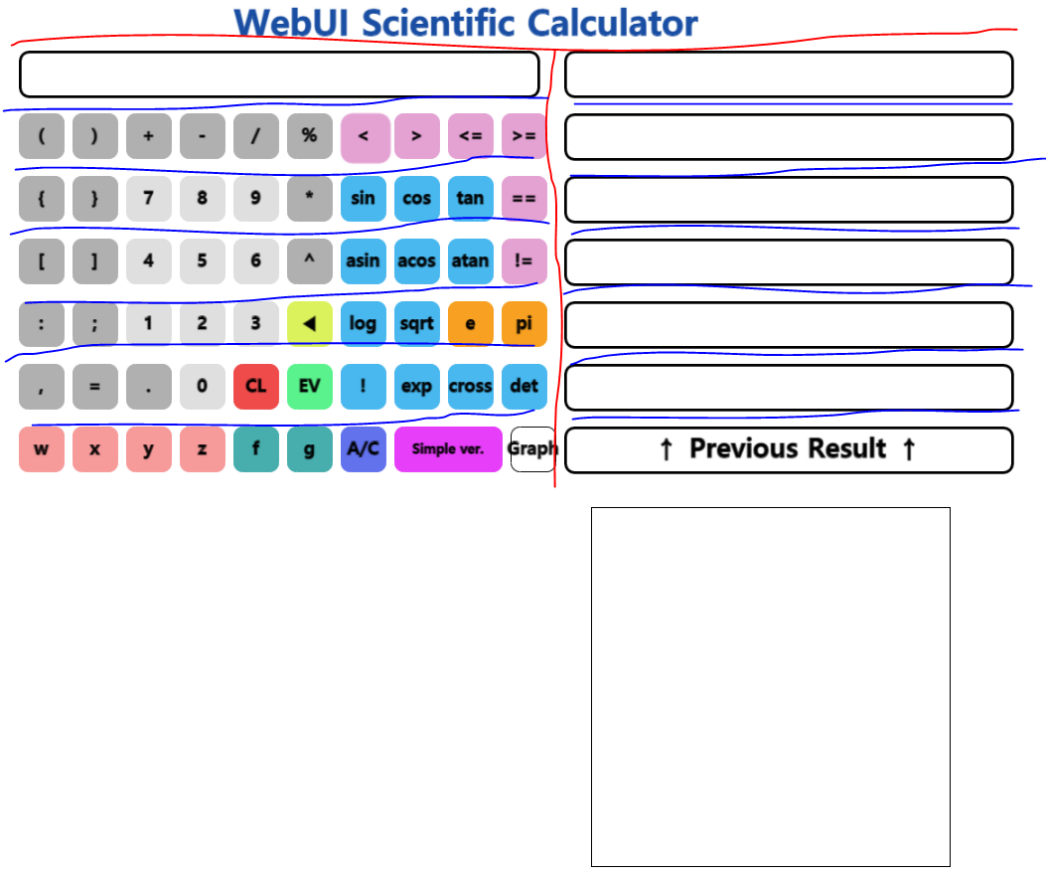
4-1. 구현요약

구현사항		구현여부
편리한 기능/인터페이스 제공		O
새로운 위젯	ChangeButton(mode)	O
	GraphPlot	O

4-2. 구현방법

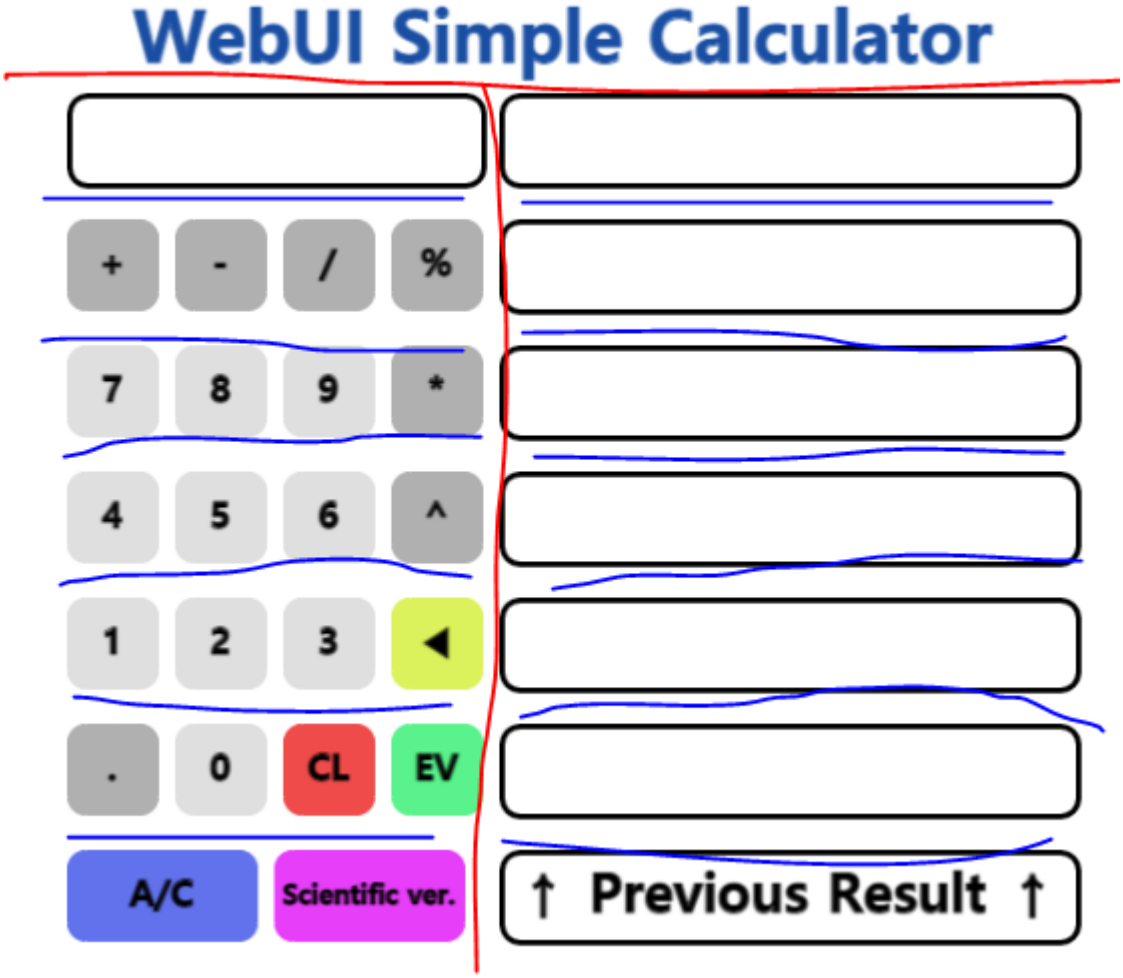
4-2-1. 레이아웃구성

-> 공학용 계산기



- 제목과 계산기 부분은 Row로 나누고, 다시 계산기 버튼과 이전결과 출력창을 Column으로 나눈 후, 각각 Row를 통해 한번더 나누고(파란색 선), 계산기 버튼을 다시 각각 Column으로 나누었다.

-> 간편 계산기



- 공학용과 마찬가지고 제목과 계산기 부분을 Row로 나누고, 다시 계산기 버튼과 이전결과 출력창을 Column으로 나눈 후, 각각 Row를 통해 한번더 나누고(파란색 선), 계산기 버튼을 다시 각각 Column으로 나누었다.

4-2-2. 추가 기능.

=== 백스페이스 ===

```
else if (this.label == "◀"){
    result = result.slice(0, -1);

    WebUI.calc_result.setLabel(result);
}
```

- 기존 CalcButton.handleButtonPushed 에 글자를 지워주는 slice기능을 이용해 백스페이스를 구현함.

=== 모드 변경 ===

- 다음 버튼을 이용하여 공학용, 간편 계산기 간 변경 가능.

```
WebUI.ChangeButton = function(label, desired_size, color){
    WebUI.PushButton.call(this);
    this.type = WebUI.WidgetTypes.CHANGE_BUTTON;
    this.label = label;
    this.desired_size = desired_size;
    this.is_pushed = false;
    this.stroke_color = color;
    this.fill_color = color;
    this.font_size = 15;


    this.onPushed = WebUI.ChangeButton.handleButtonPushed;
}

WebUI.ChangeButton.prototype = Object.create(WebUI.PushButton.prototype);
WebUI.ChangeButton.prototype.constructor = WebUI.ChangeButton;
var Calcmode = "Scientific";

WebUI.ChangeButton.handleButtonPushed = function() {
    if(Calcmode == "Scientific"){
        Calcmode = "Simple";
        WebUI.widgets = [];
        WebUI.canvas.clear();
        WebUI.initWidgets();
        WebUI.initVisualItems();
        WebUI.layoutWhenResourceReady();
        console.log(Calcmode);
    }

    else{
        Calcmode="Scientific";
        WebUI.widgets = [];
        WebUI.canvas.clear();
        WebUI.initWidgets();
        WebUI.initVisualItems();
        WebUI.layoutWhenResourceReady();
        console.log(Calcmode);
    }
};
```

- PushButton을 상속받아 ChangeButton위젯 새로 생성.

- 버튼클릭 시, 현재 계산기 모드를 저장하고있는 Calcmode변수를 변경. (Scientific, Simple)

```
if (Calcmode == "Scientific")
{

    WebUI.app = new WebUI.Row({
        children: [


            new WebUI.Container({
```

```
else{

    WebUI.app = new WebUI.Row({
        children: [


            new WebUI.Container({
                desired_size: {width: 5
                horizontal_alignment: W
                vertical_alignment: Web
```

- Calcmode == Scientific일경우, WebUI.initWidget 함수에 레이아웃이 공학용 계산기 형태로 나오도록 작성. (왼쪽)

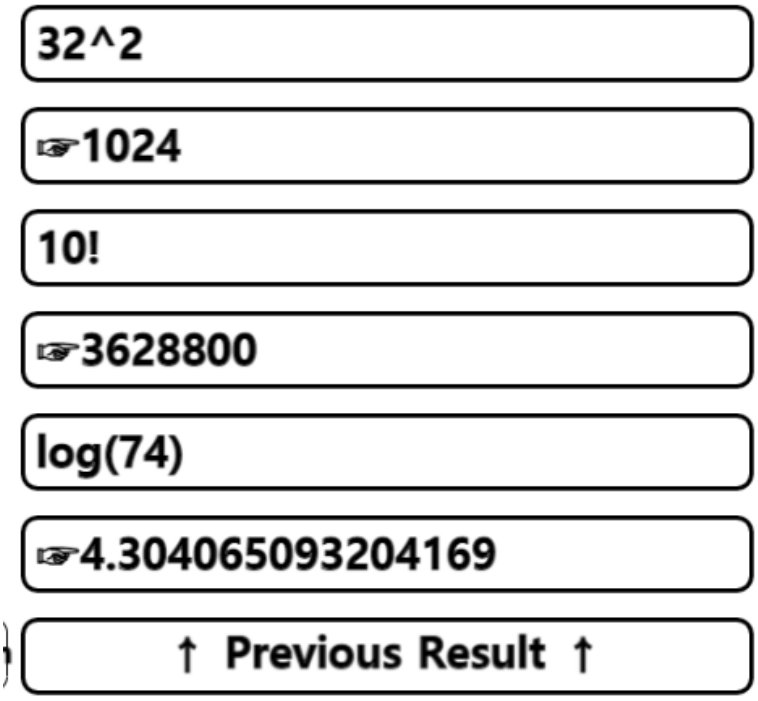
- Calcmode == Simple일경우, WebUI.initWidget 함수에 레이아웃이 간단 계산기 형태로 나오도록 작성. (오른쪽)

=== 이전 수식, 결과 저장 및 시각화 기능 ===

- 계산기 오른쪽에 있으며, 수식과 결과값 순으로 총 2행씩 저장되며, 계산이 진행될수록 아래부터 저장되어 위로 올라간다.

- 최대 3개까지의 수식과 결과값 저장 가능.

- 현재 입력되고 있는 수식은 calc_result에, 계산이 완료될때마다 수식과 결과값은 previous_result1~6에 차례대로 2칸씩 이동하며 저장된다.



=== A/C, 모두 지우기 버튼 구현 ===


```
else if (this.label == "A/C"){
    WebUI.calc_result.setLabel("");
    WebUI.previous_result1.setLabel("");
    WebUI.previous_result2.setLabel("");
    WebUI.previous_result3.setLabel("");
    WebUI.previous_result4.setLabel("");
    WebUI.previous_result5.setLabel("");
    WebUI.previous_result6.setLabel("");
}
```

- 기존 CalcButton.handleButtonPushed에 글자를 지워주는 slice기능을 이용해 백스페이스를 구현함.

- 현재 입력되던 수식부터, 저장되어있는 수식들과 결과값들을 모두 삭제한다.

=== GraphPlot 사용자 정의함수 그래프 시각화 ===

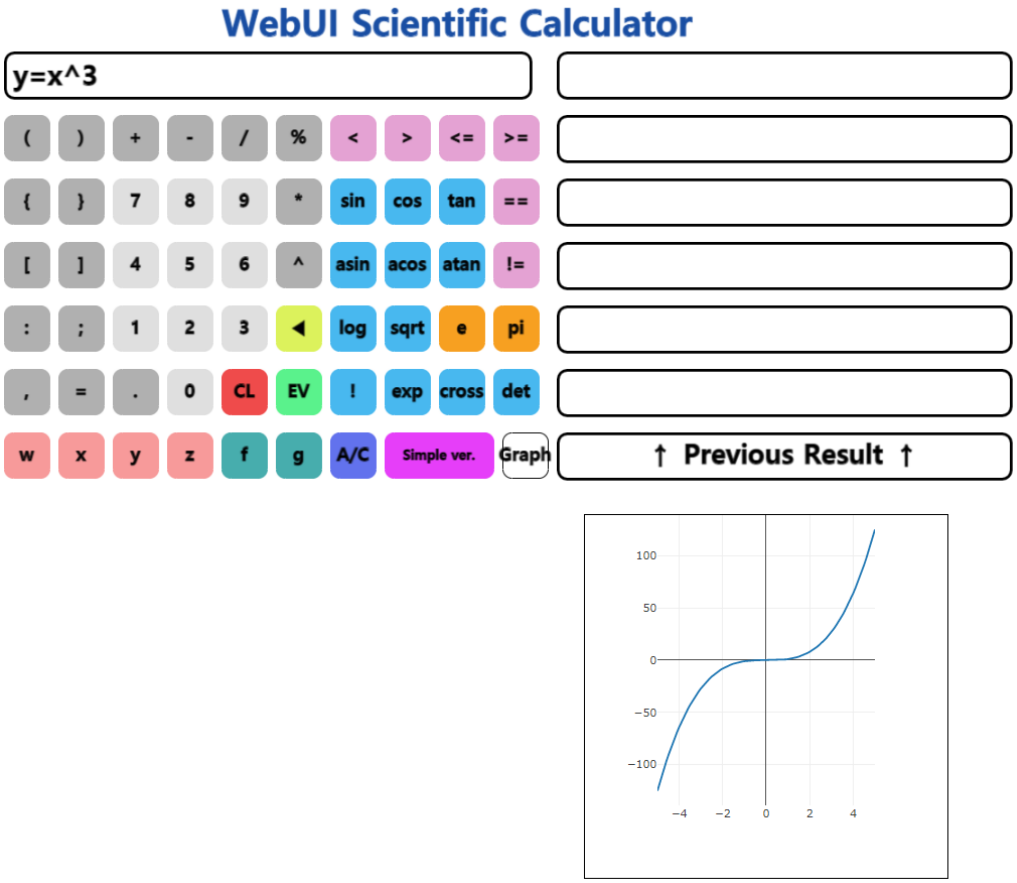
- Plotly 라이브러리 사용.

```
WebUI.GraphPlot = function(label, desired_size, properties) {
    WebUI.PushButton.call(this, label, desired_size, properties);
    this.fill_color='white';
    this.type = WebUI.WidgetTypes.GRAPH_PLOT;
    this.label = label;
    this.desired_size = desired_size;
    this.is_pushed = false;


    this.onPushed = WebUI.GraphPlot.StateButtonPushed;
}
WebUI.GraphPlot.prototype = Object.create(WebUI.PushButton.prototype);
WebUI.GraphPlot.prototype.constructor = WebUI.PushButton;
WebUI.GraphPlot.StateButtonPushed = function(){
    let func = '';
    let i = -5;
    let j = -5;
    let x_range = [];
    let y_range = [];
    while (i<5){
        x_range.push(i);
        i = i + 0.1;
    }
    let parser = math.parser();
    try{
        func = WebUI.calc_result.label;
        func_list = func.split('=');
        if(func_list[0]=='f(x)'||func_list[0]=='y'){
            cal = func_list[1];

            while (j<5){
                y = cal.replace('x',j);
                //console.log(y);
                y = parser.eval(y);
                //console.log(y);
                y_range.push(y);
                j = j + 0.1;
            }
            console.log(y_range);
        }
    }catch(e){
        if(result != 'function')
            {
                WebUI.calc_result.setTextField('error');
            }
    }
```

```
    GRAPHDRAWING = document.getElementById('graphdrawing');
    Plotly.plot(GRAPHDRAWING, [{
        x: x_range,
        y: y_range }],
        { margin: { t: 0 } }
      );
}
```

- GraphPlot 위젯을 PushButton으로부터 상속받아 생성.

- 공학용 계산기에서 f(x) 또는 y로 시작하는 간단한 수식을 입력한뒤 Graph버튼을 누르면 오른쪽 아래 공간에 그래프 생성. 아래 예시는 y=x^3을 입력한 뒤, Graph버튼을 누른 모습.



- 그래프 상자 크기 정보 설정. html, <style> 안에 정의

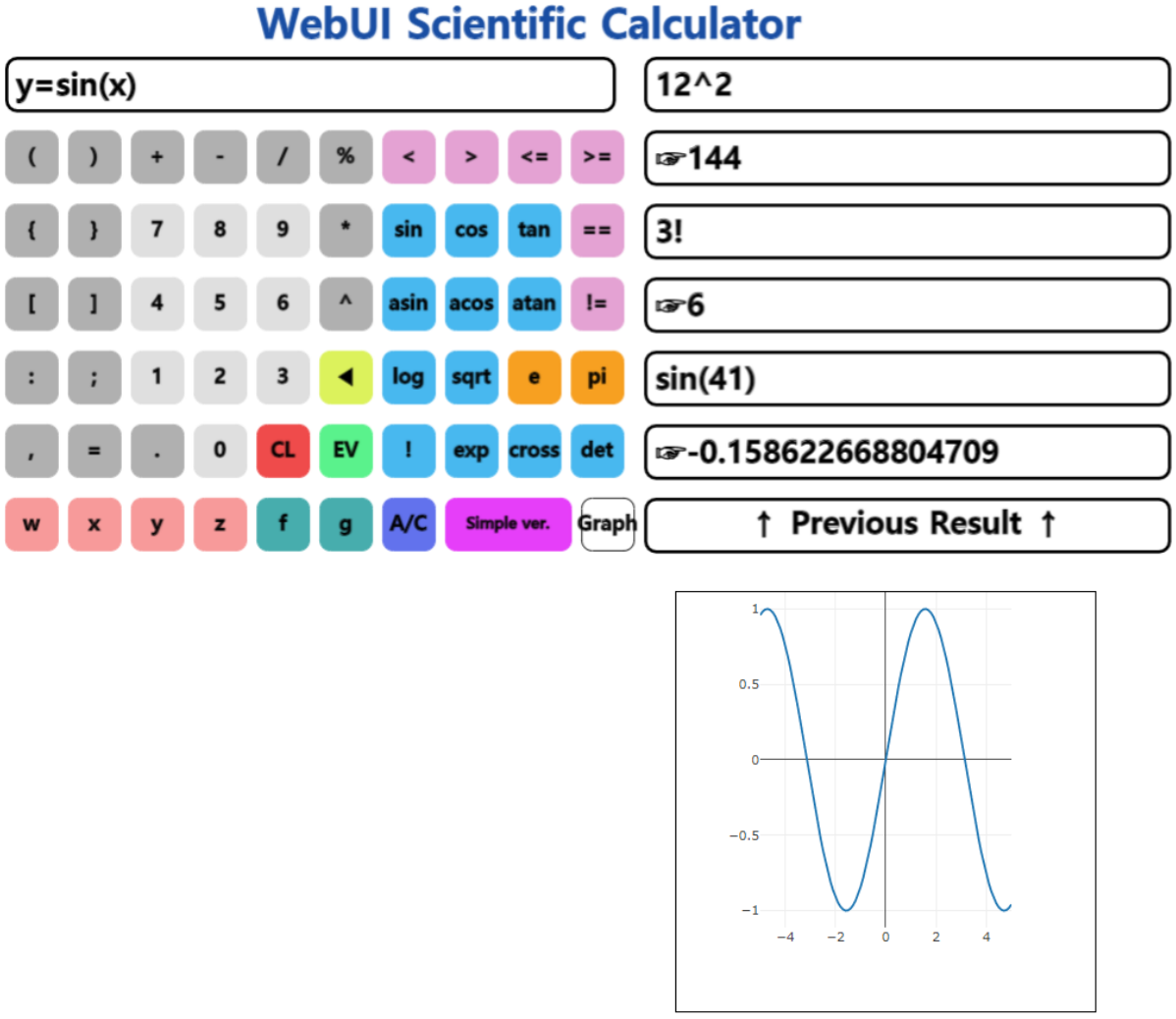
```
#graphdrawing {
                position: relative;
                top:0px;
                left:650px;
                display:inline-block;
                width:400px;
                height:400px;
                border:1px solid black;
            }
```

- 그래프 위치는 html, <body>에 를 이용해 표현.

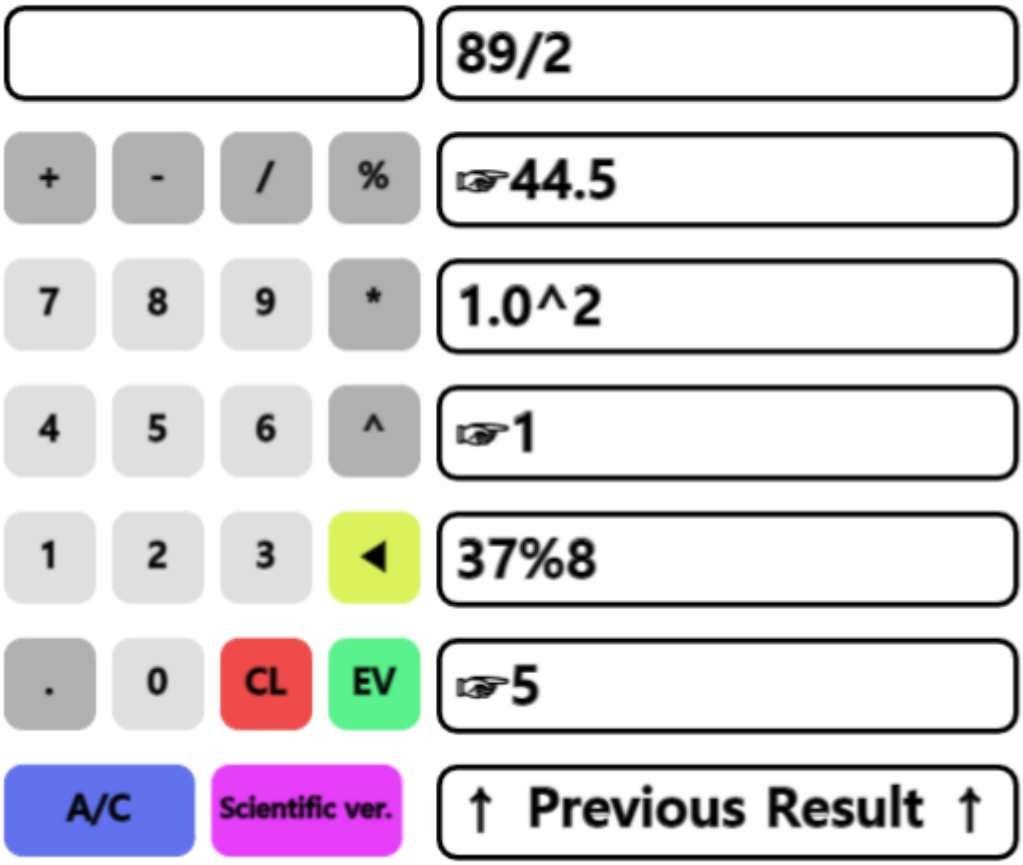
- <script src="http://cdn.plot.ly/plotly-latest.min.js"></script>

```
<body>
    <script src='https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min
.js' type='text/javascript'></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/fabric.js/2.4.6/fabric
.min.js" type="text/javascript"></script>
    <script src="math.js" type = "text/javascript"></script>
    <script src="web_ui_v3.js" type="text/javascript"></script>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>

    <canvas id="c">
        Canvas not supported
    </canvas>
    <div id="graphdrawing"></div>
<script>
```

4-3. 최종결과



WebUI Simple Calculator



5. 성공/실패 부분

- 0, 1, 2단계를 거치면서 기본적 레이아웃, WebUI 위젯을 구현하는 방법에 적응하여 3단계 과정을 진행하는데에 큰 불편함과 문제점이 발생하지 않았음.

- 그래프 그리는 공간 왼쪽이 너무 비어 보여서 그곳을 MathJax를 이용한 수식그리기로 채우고 싶었지만, 시간상 결국 구현해내지 못했다.

- 그래프는 기본적인 그래프밖에 그리지 못하지만, 처음에 html이랑 js 사이에서 어떻게 표현해야 하나 고민을 많이 했는데 구현에 성공하여 좋았음.

- 간단 계산기 모드에서 그래프 박스를 지우고 싶었지만 구현하지 못해 아쉬웠음.

6. 평가, 개선점

기호와 숫자의 역할에 따라 버튼의 색을 달리하여 한눈에 보기 편한 것 같아 좋다. 하지만, Previous Result가 저장되는 공간이 너무 딱딱하게 나누어져 있는 느낌이다. stroke테두리 없이 표현하였다면 더 좋았을 것 같다. 또한, 계산기 전체적으로 테두리와 배경을 주어 계산기 한 개 자체로의 통일감을 주었다면 디자인적으로 더 좋았을 것 같다.