라인컴퓨터아트학원

행렬계산기

웹표준기반 스마트UI/UX 디자인 콘텐츠 제작_A

장보라

목차

행렬계산기

행렬계산기란

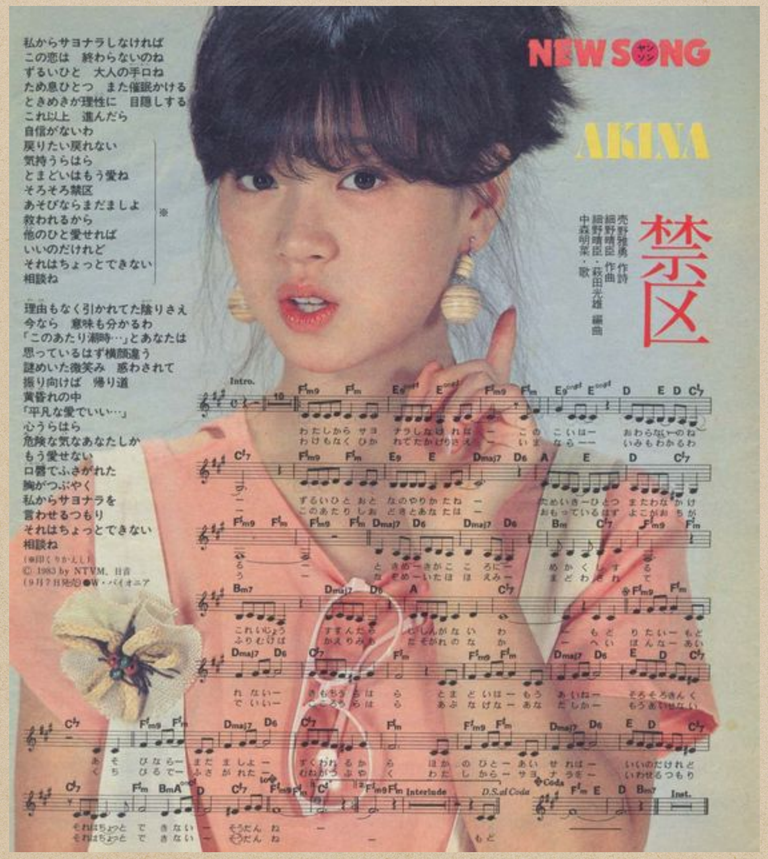
- 행렬의 생성 : 좌우에 행(y)의 개수와 열(x)의 개수를 입력받아 그만큼의 행과 열을 생성시킨다.
- 숫자 입력 : 직접 숫자를 입력하거나 랜덤 버튼을 누르면 생성된 행렬에 자동으로 숫자가 입력된다.
- 행렬 계산 : 생성된 행렬에 숫자를 입력한 후 연산 버튼을 누르면 밑에 결과 행렬이 생성된다.
- 행렬초기화 : 초기화 버튼을 누르면 입력값과 모든 행렬이 화면에서 지워진다.

- 덧셈, 뺄셈 공식 : 양 행렬의 각 좌표에 대응하는 값끼리 더하거나 뺀다.
- 곱셈 공식 : 좌측 행렬의 행에 우측 행렬의 열을 곱한다.

- 주의점
- 덧셈, 뺄셈의 경우 좌우 행렬의 행(y)의 개수와 열(x)의 개수가 일치해야한다.
- 곱셈의 경우 왼쪽 행렬의 행(y)의 개수와 오른쪽 행렬의 열(x)의 개수가 일치해야한다.

색상 선정

색상 및 폰트 선정 이유



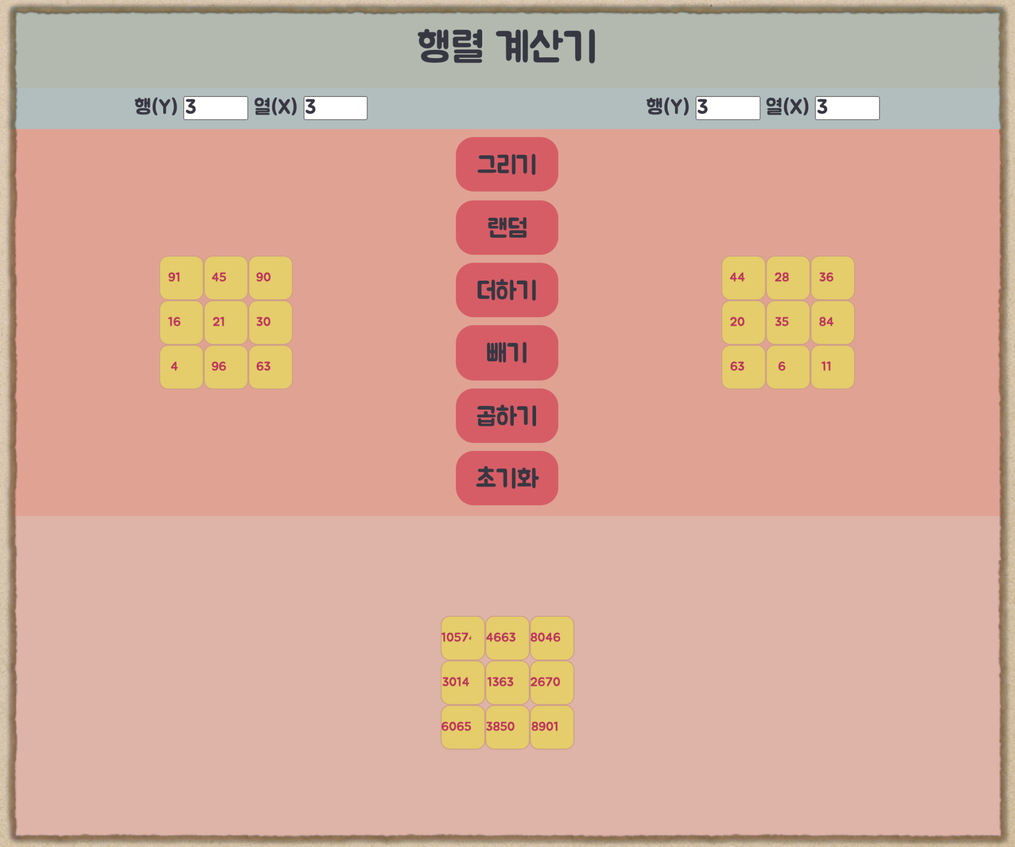
- 빈티지한 느낌의 느낌을 주기 위해 과거 일본 잡지인 좌측의 이미지에서 색상을 추출하여 사용하였다.
- 폰트는 자극적이지 않은 색상과 잘 어우러지도록 둥근 글씨체인 배달의민족 주아체를 사용하였다.

#B3B9B1
#B4BEBF
#DFA797
#DEB7AB
#D96069
#E7CD69
#3A3946

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

색상 배치

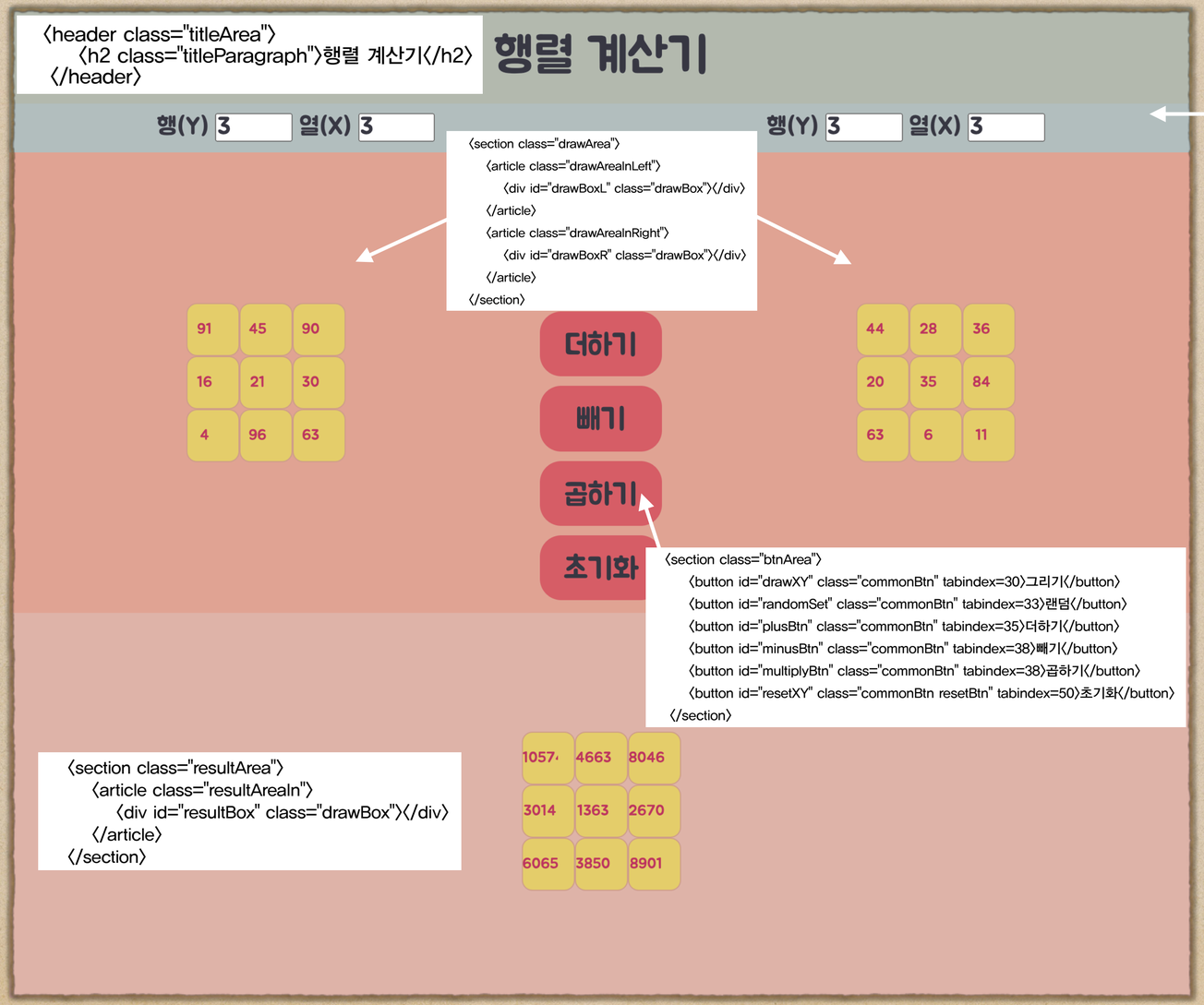
- 자연스러운 배치를 위하여 최상단인 행렬계산기 타이틀 영역에는 #B3B9B1색상을 사용하고, 입력칸의 영역에는 그보다 밝은 #B4BEBF색상을 사용하였다.
- 상단영역보다 넓게 사용되는 나머지 영역에 핑크계열의 #DFA797, #DEB7AB색상을 사용하여 전체적으로 칙칙해보이지 않게 하고, 버튼을 눈에 잘 띄도록 제일 진한 색인 #D96069색상을 사용하고 가운데에 배치하였다.
- 생성된 행렬에는 버튼과 같은 이유로 밝은 노란색인 #E7CD69색상을 사용하였다.
- 폰트에는 추출한 색상 중 진한 색인 #3A3946색상과 #E7CD69색상이 사용하였다.



http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

HTML

HTML




```
〈section class="inputArea"〉
    〈article class="leftInputArea"〉
        〈label〉행(Y)
            〈input id="inY1" class="commonIn" type="number" min="0" max="999" value="0" autofocus tabindex=10〉
        〈/label〉
        〈label〉열(X)
            〈input id="inX1" class="commonIn" type="number" min="0" max="999" value="0" tabindex=20〉
        〈/label〉
    〈/article〉
    〈article class="rightInputArea"〉
        〈label〉행(Y)
            〈input id="inY2" class="commonIn" type="number" min="0" max="999" value="0" tabindex=22〉
        〈/label〉
        〈label〉열(X)
            〈input id="inX2" class="commonIn" type="number" min="0" max="999" value="0" tabindex=25〉
        〈/label〉
    〈/article〉
〈/section〉
```

drawXY 객체

```
function drawXY(selectId, selectYAxis, selectXAxis) {
        this.inY = document.getElementById(selectYAxis).value;
        this.inX = document.getElementById(selectXAxis).value;
        this.boxSize = document.getElementById(selectId);
        this.matrixArray = [];
        this.plusArray = [];
        this.minusArray = [];

        this.matrixSetting = function (selMethod) {}
        this.drawBox = function () {}
        this.plusValue = function (inputLeftId, inputRightId) {}
        this.minusValue = function (inputLeftId, inputRightId) {}
        this.multiplyValue = function () {}
}
```

draw 객체를 이용하여 좌우행렬의 생성과 연산, 결과행렬의 생성을 한다.
draw객체의 안에는 행렬의 연산과 생성을 위한 변수의 선언과 행렬 계산을 위한 배열 생성 메소드 (matrixSetting), 객체 안에는 행렬 생성을 위한 메소드 (drawBox), 덧셈 메소드 (plusValue), 뺄셈 메소드 (minusValue), 곱셈 메소드 (multiplyValue) 등이 있다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

matrixSetting 메소드

```
this.matrixSetting = function (selMethod) {
    let k = 0;
    while (k < this.inY) {
    this.matrixArray.push([]);
    k++;
    }
    let j = 0;
    while (j < this.inY) {
        let i = 0;
        while (i < this.inX) {
        (selMethod == 'resetArray') && (this.matrixArray[j][i] = 0);
        (selMethod == 'randomArray') && (this.matrixArray[j][i] = Math.floor(Math.random() * 1000));
        i++;
        }
    j++;
    }
}
```

행렬 생성을 위한 2차원 배열을 생성하는 메소드이다.
매개변수의 값에 따라 배열의 값을 0으로 만들거나 배열의 값을 1에서 999까지의 랜덤한 숫자가 들어가게 한다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

drawBox 메소드

```
this.drawBox = function () {
      this.boxSize.innerHTML = "";
      this.boxSize.style.width = this.inX * 50 + "px";
      this.boxSize.style.height = this.inY * 50 + "px";
      this.boxSize.style.position = "absolute";
      this.boxSize.style.top = 50 + "%";
      this.boxSize.style.left = 50 + "%";
      this.boxSize.style.marginTop = -((this.inY * 50) / 2) + "px";
      this.boxSize.style.marginLeft = -((this.inX * 50) / 2) + "px";

      this.matrixArray.forEach((rowValue, rowIndex) => {
            rowValue.forEach((colValue, colIndex) => {
               this.boxSize.innerHTML +=
               '<input type="number" id="' + selectId + rowIndex + colIndex +
                  '" class="boxDeco" min="0" max=  "999" value="' + colValue + '">';
               });
            });
}
```

버튼이 눌리면 행렬이 생성되도록 하기 위해 객체의 매개변수로 입력된 태그의 스타일을 바꾸어준다.
생성된 행렬의 개수만큼 부모 태그의 크기도 커져야하기 때문에 부모의 크기에 그 수만큼 곱한다.
그리고 생성된 행렬을 가운데에 배치하지 위해 position 속성을 지정해준다.
forEach를 통해 matrixSetting 메소드로 만들어진 2차원 배열을 이용하여 innerHTML으로 input태그를 넣어준다. 이때 각 태그의 id는 다 달라야하기 때문에 생성된 배열의 값과 인덱스번호를 가져와서 id값을 지정해준다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

reset 함수

```
let resetF = () => {
    document.getElementById("inX1").value = document.getElementById("inY1").value = 0;
    document.getElementById("inX2").value = document.getElementById("inY2").value = 0;
    document.getElementById("drawBoxL").innerHTML = "";
    document.getElementById("drawBoxR").innerHTML = "";
    document.getElementById("resultBox").innerHTML = "";
}

let resetR = () => {
    document.getElementById("resultBox").innerHTML = "";
}
```

resetF 함수 : 초기화 버튼을 누르면 전체 화면이 지워지도록 각 태그의 값을 0이나 null로 만들어준다.

resetR 함수 : 이미 한번 연산이 된 상태에서 다른 연산을 할 경우 현재 생성되어 있는 결과값을 지워주기 위해 결과 영역의 값을 null로 만들어준다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

덧셈, 뺄셈 메소드

```
this.plusValue = function (inputLeftId, inputRightId) {
    let inY1V = Number(document.getElementById("inY1").value);
    let inX1V = Number(document.getElementById("inX1").value);
    let inY2V = Number(document.getElementById("inY2").value);
    let inX2V = Number(document.getElementById("inX2").value);
    if(inY1V==inY2V&&inX1V==inX2V){
     let i = 0;
     while (i <inY1V) {
        let j = 0;
        while (j <inX1V) {
            let leftBoxV = parseInt(document.getElementById(inputLeftId + i + j).value);
            let rightBoxV = parseInt(document.getElementById(inputRightId + i + j).value);
            this.plusArray.push(leftBoxV + rightBoxV);
            this.plusArray.forEach((value) => document.getElementById('resultBox' + i + j).value = value);
            j++;
        }
        i++;
        }
    }else{
        alert("덧셈은 왼쪽 행렬과 오른쪽 행렬의 행렬의 수가 같아야합니다.");
        resetR();
    }
}
```

덧셈, 뺄셈의 경우 좌우 행렬의 개수가 같아야 하기 때문에 if문을 통해 좌우 행의 값과 열의 값이 일치하는지 확인한 후 일치하면 이중 while문을 통해 행*열의 값만큼 반복해주고 일치하지 않으면 alert으로 안내와 함께 resetR함수를 이용하여 결과영역을 초기화한다. 초기화를 해줘야하는 이유는 if문의 조건을 만족할 경우를 위해 각 버튼이 눌리면 결과행렬이 생성되게 되어있기 때문에 객체 생성에 입력된 매개변수의 값만큼 결과행렬이 생겨버리기 때문이다.
if문의 조건을 만족하는 경우 변수에 현재 화면에 생성되어있는 값을 가져온 후 그 값을 더한 후 plusArray 배열에 넣어준다. 그리고 plusArray 배열의 길이만큼 값이 들어갈 수 있도록 forEach를 통해 결과행렬에 값을 직접 넣어준다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

곱셈 메소드 – 1

```
this.multiplyValue = function () {
    let inY1V = Number(document.getElementById("inY1").value);
    let inX1V = Number(document.getElementById("inX1").value);
    let inY2V = Number(document.getElementById("inY2").value);
    let inX2V = Number(document.getElementById("inX2").value)
    let matrixLeftArray = [];
    let matrixRightArray = [];
    let multiplyArray = [];
    let multiplyTemp=[];
    let mulTmp=0;
   for(let i=0; i<inY1V; i++){
        matrixLeftArray.push([]);
    }
    for(let j=0; j<inY2V; j++){
        matrixRightArray.push([]);
    }

       if(inX1V == inY2V){
    . . .

    }else{
          alert("곱셈은 왼쪽 행렬의 행의 수와 오른쪽 행렬의 열의 수가 같아야합니다." );
          resetR();
    }

}
```

곱셈의 경우 덧셈, 뺄셈과는 다르게 좌우 행렬의 배열을 통한 연산을 해야하므로 입력된 행(y)의 값만큼 좌우 행렬의 값이 들어갈 배열에 push함수를 사용하여 2차원 배열을 만든다.
곱셈의 경우에는 왼쪽 행렬의 행(y)의 개수와 오른쪽 행렬의 열(x)의 개수가 일치해야하기 때문에 if문을 사용한다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

곱셈 메소드 – 2

```
for(i=0; i<inY1V; i++){
    for(j=0; j<inX1V; j++){
        matrixLeftArray.forEach(function(arrL,index){
            if(index==i){
                arrL.push(Number(document.getElementById("drawBoxL"+i+j).value));
            }
        });
    }
}

for(i=0; i<inY2V; i++){
    for( j=0; j<inX2V; j++){
        matrixRightArray.forEach(function(arrR,index){
            if(index==i){
            arrR.push(Number(document.getElementById("drawBoxR"+i+j).value));
            }
        });
    }
}
```

앞 페이지의 if문의 조건을 만족하면 덧셈에서 했던 것과 같이 현재 화면의 좌우행렬에 입력되어 있는 값을 이중 for문 속에 좌우 배열의 행(y)값만큼 생성된 배열인 matrixLeftArray와 matrixRightArray를 forEach 속 if문을 통해 행(y)값과 matrixLeftArray, matrixRightArray배열의 인덱스번호가 같을 때 2차원 배열 중 안쪽 배열에 push해준다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

곱셈 메소드 – 3

```
for(i=0; i<inY1V; i++){
   for(j=0; j<inX2V; j++){
       for(let k=0; k<inY2V; k++){
           for(let l=0; l<inX1V; l++){
                  if(k==l){
                   mulTmp += matrixLeftArray[i][l]*matrixRightArray[k][j];
                  }
              }
        }
        multiplyTemp.push(mulTmp);
        document.getElementById('resultBox'+i+j).value=mulTmp;
        mulTmp=0;
      }
      multiplyArray.push(multiplyTemp);
      multiplyTemp=[];
    }
}
```

곱셈의 결과 행렬은 왼쪽 행렬의 행(y) X 오른쪽 행렬의 열(x)의 개수만큼 생성되어야 하기 때문에 2중 for문에서 곱해진 값을 곱셈 결과 배열에 넣어준다.
곱셈의 경우 4중 for문을 사용한다.왼쪽의 행(y)보다 오른쪽의 열(x)이 더 먼저 증가하기 때문에 안쪽 반복문에 쓰이는 j를 오른쪽 배열의 안쪽 배열(x)의 인덱스 번호로 쓰고, 반복문에 쓰이는 l의 경우 k와 값이 같아야 하지만 그보다 값이 더 커지면 안되기 때문에 if를 통해 조건문을 만들어준다. 하나의 행(y)과 하나의 열(x)의 계산이 끝나고 값이 들어가야하기 때문에 다음열, 다음 행(y)이 가기 전에 값을 넣어준다.
값을 넣어준 이후에는 초기화를 시켜서 이전의 값을 없애고 다음 배열에 들어갈 값이 들어가야 제대로 된 결과가 나온다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

이벤트리스너

```
this.addEventListener("click", (e) => {
        let boxL = new drawXY("drawBoxL", "inY1", "inX1");
        let boxR = new drawXY("drawBoxR", "inY2", "inX2");
        let boxP = new drawXY("resultBox", "inY1", "inX2");
        let boxM = new drawXY("resultBox", "inY1", "inX2");
        let boxMul = new drawXY("resultBox", "inY1",    "inX2")
        switch (e.target.id) {

        . . .다음장에 계속
    })
}
```

버튼이 클릭이 되었을 때 행렬이 생성이 되어야 하기 때문에 이벤트리스너 안에 각 객체를 생성한다.
그리고 어떤 버튼이 클릭되는지에 따라 다른 결과를 도출하기 위해 switch문을 사용한다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

이벤트리스너

```
switch (e.target.id) {


case 'drawXY':
        boxL.matrixSetting("resetArray");
        boxL.drawBox();
        boxR.matrixSetting("resetArray");
        boxR.drawBox();
        break;


 case 'randomSet':
       boxL.matrixSetting("randomArray");
       boxL.drawBox();
       boxR.matrixSetting("randomArray");
       boxR.drawBox();
       break;


    case 'resetXY':
         resetF();
         break;

});
```

그리기 버튼이 클릭 되었을 경우:
resetArray메소드를 통해 양 객체의 배열의 값을 초기화해준 후 drawBox메소드로 행렬을 생성한다.

랜덤 버튼이 클릭 되었을 경우:
randomtArray메소드를 통해 양 객체의 배열의 값을 랜덤으로 지정해준 후 drawBox메소드로 행렬을 생성한다.

초기화 버튼이 클릭 되었을 경우:
resetF함수를 통해 화면의 모든 값을 초기화한다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

이벤트리스너

```
switch (e.target.id) {


 case 'plusBtn':
      boxP.matrixSetting("resetArray");
      resetR();
      boxP.drawBox();
      boxP.plusValue("drawBoxL", "drawBoxR");
      break;

case 'minusBtn':
      boxP.matrixSetting("resetArray");
      resetR();
      boxP.drawBox();
      boxP.minusValue("drawBoxL", "drawBoxR");
      break;

 case 'multiplyBtn':
       boxMul.matrixSetting("resetArray");
       resetR();
       boxMul.drawBox();
       boxMul.multiplyValue();
       break;

 });
```

더하기 버튼이 클릭 되었을 경우 :
resetArray 메소드를 통해 양 객체의 배열의 값을 초기화해준 후 결과 영역을 초기화한 후 drawBox 메소드로 행렬을 생성한다. 그리고 plusValue 메소드를 이용하여 연산한다.

빼기 버튼이 클릭 되었을 경우 :
resetArray 메소드를 통해 양 객체의 배열의 값을 초기화해준 후 결과 영역을 초기화한 후 drawBox 메소드로 행렬을 생성한다. 그리고 minusValue 메소드를 이용하여 연산한다.

곱셈 버튼이 클릭 되었을 경우 :
resetArray 메소드를 통해 양 객체의 배열의 값을 초기화해준 후 결과 영역을 초기화한 후 drawBox 메소드로 행렬을 생성한다. 그리고 multiplyValue 메소드를 이용하여 연산한다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

총평

선생님께서 처음 행렬 계산기를 제작해오라고 하셨을 때는 행렬계산이 무엇인지조차 몰랐었다. 적당히 조사하고 코드를 작성을 시작했다. 그런데 나중에 실행을 해보니 곱셈의 공식을 제대로 파악을 한 상태에서 짠 것이 아니라서 오류가 발생했다. 이때 사전 조사의 중요성을 매우 절실히 깨달았다. 처음부터 제대로 조사하고 했으면 시간 낭비를 하지 않았을 것이다.

그러고 나서 행렬의 곱셈 계산 자체를 이해하는 데에 시간이 걸렸다. 같은 반 친구들의 설명을 들어도 이해가 잘 안 갔다. 하지만 제출 기일은 다가오고 있으니 완성을 해야 했다. 그래서 하고 싶지 않았던 남의 코드를 봤다. 그래도 이해가 가지 않아서 설명을 들었지만 그래도 이해가 가지 않았다. 그래서 나는 포기했다. 하지만 선생님이 우리들의 배열의 이해도의 심각성을 파악하시고 PHP에서 더 연습을 시켜주셨다. 그래서 그때 더 열심히 공부하고 연습을 했더니 자연스럽게 스스로 곱셈의 코드를 작성할 수 있었다. 다시 돌이켜보면 처음부터 더 배열의 활용에 대해 열심히 공부했었다면 스스로 더 빨리 문제를 해결할 수 있었을 것이다.

이번 경험을 통해 앞으로는 무언가를 활용하는 데에 있어서 그것에 대해 제대로 이해한 후에 제대로 활용할 수 있도록 노력할 것이다.

http://pager.kr/~c11st25/bora/portfolio/matrix/final_matrix_calculator.html

감사합니다.