휴먼컴퓨터 인터페이스

과제 #2. WebUI 계산기 구현

2020.05.31

컴퓨터 소프트웨어 학과

2016726009 | 임현우

01 요구조건과 제약조건 만족도 및 추가 구현사항 Summary

1단계

	템플릿 프로젝트에 기초	레이아웃 관련 코드 입력	로그인 페이지 재구성
충족 여부	○	○	○
세부 사항	▷ 템플릿 프로젝트(web_ui_v1.zip)에 기초해서 주어진 코드의 흐름에 맞추어 구현하였으며 리소스도 그대로 사용	▷ WidgetTypes, Alignment, maxSize, minSize에 빠진 부분을 입력	▷ initWidgets() 함수에서 Container, Row, Column을 생성해 상대적 위치 기반으로 위젯 생성

01 요구조건과 제약조건 만족도 및 추가 구현사항 Summary

2단계

	수식 계산을 위한 객체 정의	MyPushButton 클래스 정의	initWidgets() 함수 재정의
충족 여부	○	○	○
세부 사항	▷ WebUI.parser 라는 이름의 math.parser() 객체를 선언해 사용	▷ PushButton을 상속한 MyPushButton을 선언 ▷ handleButtonPushed 멤버 함수 추가하여 버튼이 눌렸을 때의 이벤트를 처리	▷ 제목, 화면, 버튼을 각각 Container로 감싸 새롭게 레이아웃을 구성

01 요구조건과 제약조건 만족도 및 추가 구현사항 Summary

3단계

	새로운 상호작용 위젯 정의	새로운 레이아웃 위젯 정의	외부 라이브러리, 리소스 사용
충족 여부	○	○	○
세부 사항	▷ 계산기의 버튼을 사용할 때 어려움이 없도록 설명창을 띄우도록 하는 새로운 Text 위젯 정의	▷ 어떠한 위젯의 위치에 종속적인 다른 위젯을 생성하기 위해 새로운 위치기반 레이아웃을 정[redacted]	▷ html2canvas 라이브러리를 사용해 캔버스를 스크린샷 하도록 함

01 요구조건과 제약조건 만족도 Summary

제약 조건

1	클라이언트 측 스크립트만 사용	○
2	오픈소스 라이브러리 허용(HTML2CANVAS 사용하였음)	○
3	추가 리소스 사용 권장(HTML2CANVAS 라이브러리는 인터넷을 통해 임포트하였음)	○
4	구글 크롬 웹 브라우저 호환 필수	○

Chapter 1

WebUi 라이브러리 완성

01 레이아웃 관련 코드 입력(빠진 부분 채워 넣기)

WidgetTypes

```
WebUI.WidgetTypes = {
    UNDEFINED:      "undefind",
    TEXT:           "text",
    IMAGE:          "image",
    PUSH_BUTTON:    "push_button",
    TEXT_FIELD:     "text_field",
    SWITCH:         "switch",
    CONTAINER:      "container",
    ROW:            "row",
    COLUMN:         "column"
};
```

WidgetTypes의 기존 목록에 CONTAINER, ROW, COLUMN을 추가하였다.

01 레이아웃 관련 코드 입력(빠진 부분 채워 넣기)

Alignment

```
WebUI.Alignment = {
    CENTER:         "center",
    LEFT:           "left",
    RIGHT:          "right",
    TOP:            "top",
    BOTTOM:         "bottom"
};
```

정렬에 필요한 CENTER, LEFT, RIGHT, TOP, BOTTOM 요소를 정의하였다.

01 레이아웃 관련 코드 입력(빠진 부분 채워 넣기)

maxSize, minSize

```
WebUI.maxSize = function(size1, size2) {
    let max_size = {width: 0, height0: 0};
    max_size.width = (size1.width > size2.width) ? size1.width : size2.width;
    max_size.height = (size1.height > size2.height) ? size1.height : size2.height;

    return max_size;
}

WebUI.minSize = function(size1, size2) {
    let min_size = {width:0, height:0};
    min_size.width = (size1.width < size2.width) ? size1.width : size2.width;
    min_size.height = (size1.height < size2.height) ? size1.height : size2.height;

    return min_size;
}
```

강의자료에 주어진 대로 maxSize, minSize함수를 정의하였고, return문이 포함되지 않아 오류가 발생하는 부분이 있어 return 문을 추가함.

01 레이아웃 관련 코드 입력(빠진 부분 채워 넣기)

layout, measure

```
WebUI.Widget.prototype.layout = function() {
    this.measure();
    this.arrange(this.position);
}
WebUI.Widget.prototype.measure = function() {
    if(this.children.length > 0){
      this.size_children = {width: 0, height: 0};
      this.children.forEach(child => {
        let size_child = child.measure();
        this.size_children = this.extendSizeChildren(this.size_children, size_child);
      });
      this.size = WebUI.maxSize(this.desired_size, this.size_children);
    }
    else{
      this.size.width += this.padding * 2;
      this.size.height += this.padding * 2;
    }
    return this.size;
}
```

강의자료에 주어진 대로 layout, measure 함수를 채워 넣었다. Measure 함수는 this에 해당하는 위젯의 자식들의 크기를 모두 구해 this.size_children 변수에 변화를 주는 기능을 한다.

01 레이아웃 관련 코드 입력(빠진 부분 채워 넣기)

arrange

```
WebUI.Widget.prototype.arrange = function(position) {
    //arrange this
    this.moveTo(position);
    this.visual_items.forEach(item => {WebUI.canvas.add(item);});

    //arrange children
    if(this.children.length > 0){
      let left_spacing = 0, top_spacing = 0;

      if(this.size.width > this.size_children.width){
        let room_width = this.size.width - this.size_children.width;

        if(this.horizontal_alignment == WebUI.Alignment.LEFT)
          left_spacing = this.padding;
        else if(this.horizontal_alignment == WebUI.Alignment.CENTER)
          left_spacing = this.padding + room_width / 2.0;
        else if(this.horizontal_alignment == WebUI.Alignment.RIGHT)
          left_spacing = this.padding + room_width;
      }

      if(this.size.height > this.size_children.height){
        let room_height = this.size.height - this.size_children.height;

        if(this.vertical_alignment == WebUI.Alignment.TOP)
          top_spacing = this.padding;
        else if(this.vertical_alignment == WebUI.Alignment.CENTER)
          top_spacing = this.padding + room_height/ 2.0;
        else if (this.vertical_alignment == WebUI.Alignment.BOTTOM)
          top_spcing = this.padding + room_height;
      }

      let next_position = {left: position.left + left_spacing, top: position.
      this.children.forEach(child => {
        child.arrange(next_position);
        next_position = this.calcNextPosition(next_position, child.size);
      });
    }
}
```

강의자료에 주어진 대로 arrange함수를 채워 넣었다. Arrange함수는 children의 크기에 기반해 위치와 여백을 지정해주는 기능을 한다.

02 로그인 페이지 재구성

initWidgets

```
WebUI.initWidgets = function() {
    WebUI.app = new WebUI.Row({
      children: [
        new WebUI.Container({
          desired_size: {width: 400, 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.Container({
              desired_size:{width:130, height:80},
              horizontal_alignment: WebUI.Alignment.CENTER,
              vertical_alignment: WebUI.Alignment.CENTER,
              children: [ new WebUI.Image("resources/HTML5.png", {width: 100, height: 80}) ]
            }),
            new WebUI.Container({
              desired_size:{width:130, height:80},
              horizontal_alignment: WebUI.Alignment.CENTER,
              vertical_alignment: WebUI.Alignment.CENTER,
              children: [ new WebUI.Image("resources/CSS3.png", {width:100, height:80}) ]
            }),
            new WebUI.Container({
              desired_size:{width:130, height:80},
              horizontal_alignment: WebUI.Alignment.CENTER,
              vertical_alignment: WebUI.Alignment.CENTER,
              children: [ new WebUI.Image("resources/JS.png", {width: 100, height: 80}) ]
            })
          ]
        }),
```

캡쳐된 부분은 initWidgets 함수 내부 중 하나의 Column 만을 생성하는 부분이다. 이처럼 위젯을 생성할 때 위젯의 절대적인 위치를 지정해주지 않는 모습을 보인다.

02 로그인 페이지 재구성

Container

```
WebUI.Container = function(properties) {
    WebUI.Widget.call(this, properties);

    this.type = WebUI.WidgetTypes.CONTAINER;
}

WebUI.Container.prototype = Object.create(WebUI.Widget.prototype);
WebUI.Container.prototype.constructor = WebUI.Container;

WebUI.Container.prototype.extendSizeChildren = function(size, child_size) {
    // implement this
    if(size.width < child_size.width) size.width = child_size.width;
    if(size.height < child_size.height) size.height = child_size.height;
    return size;
}

WebUI.Container.prototype.calcNextPosition = function(position, size) {
    // implement this
    let next_left = position.left;
    let next_top = position.top;
    return {left: next_left, top: next_top};
}
```

Container의 생성자와 멤버 함수이다. extendSizeChildren 함수와 calcNextPosition 함수를 이용해 자식들의 크기에 따라 Container의 크기가 결정되고, 자식의 위치 또한 자동으로 배정한다.

02 로그인 페이지 재구성

Row

```
WebUI.Row = function(properties) {
    WebUI.Widget.call(this, properties);
    this.type = WebUI.WidgetTypes.ROW;
}

WebUI.Row.prototype = Object.create(WebUI.Widget.prototype);
WebUI.Row.prototype.constructor = WebUI.Row;

WebUI.Row.prototype.extendSizeChildren = function(size, child_size) {
    // implement this
    //add
    if(size.width < child_size.width) size.width = child_size.width;
    size.height += child_size.height;
    return size;
}

WebUI.Row.prototype.calcNextPosition = function(position, size) {
    // implement this
    //add
    let next_left = position.left;
    let next_top = position.top + size.height;
    return {left: next_left, top: next_top};
}
```

Row의 생성자와 멤버 함수이다.
extendSizeChildren 함수와 calcNextPosition 함수를 이용해 자식들의 크기를 구하고, 그 크기를 바탕으로 세로 방향으로 자식들을 늘어뜨리는 작동을 한다.

02 로그인 페이지 재구성

Column

```
WebUI.Column = function(properties) {
    WebUI.Widget.call(this, properties);

    this.type = WebUI.WidgetTypes.COLUMN;
}

WebUI.Column.prototype = Object.create(WebUI.Widget.prototype);
WebUI.Column.prototype.constructor = WebUI.Column;

WebUI.Column.prototype.extendSizeChildren = function(size, child_size) {
    // implement this
    //add
    size.width += child_size.width;
    if(size.height < child_size.height) size.height = child_size.height;
    return size;
}

WebUI.Column.prototype.calcNextPosition = function(position, size) {
    // implement this
    //add
    let next_left = position.left + size.width;
    let next_top = position.top;
    return {left: next_left, top: next_top};
}
```

Column의 생성자와 멤버 함수이다. extendSizeChildren 함수와 calcNextPosition 함수를 이용해 자식들의 크기를 구하고, 그 크기를 바탕으로 가로 방향으로 자식들을 늘어뜨이는 작동을 한다.

03 최종 결과

과제#1과 비슷한 레이아웃을 갖지만 위치를 직접 지정하는 방식이 아닌 상대적 위치지정 방식을 적용한 최종 결과 화면 예시

Chapter 2

WebUI 기본 계산기 구현

01 수식 계산을 위한 객체 정의

WebUI.parser

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

Math.parser 객체를 생성하고 WebUI.parser 가 참조하도록 하여 추후에 수식 계산을 위한 객체로 사용 하였다.

02 MyPushButton 클래스 정의

MyPushButton

```
WebUI.MyPushButton = function(label, desired_size, properties) {
    WebUI.PushButton.call(this, label, desired_size, properties);

    this.onPushed = WebUI.MyPushButton.handleButtonPushed;
}
WebUI.MyPushButton.prototype = Object.create(WebUI.PushButton.prototype);
WebUI.MyPushButton.prototype.constructor = WebUI.MyPushButton;
```

MyPushButton을 선언하고 PushButton을 상속 받도록 작성된 코드이다. 필요해 의해서 label과 desired_size를 생성시에 입력받도록 하였다.

This.onPushed 에 hadleButtonPushed 이벤트 핸들러를 연결 하였다.

02 MyPushButton 클래스 정의

MyPushButton

```
WebUI.MyPushButton.handleButtonPushed = function(){
  if(expression == "0"){
    expression="";
  }
  if(this.label == "CL"){
    WebUI.clear_func();
  }
  else if(this.label == "EV"){
    WebUI.eval_func();
  }
  else{
    WebUI.mainView.setLabel(expression+=this.label);
  }
}
```

WebUI.MyPushButton.handleButtonPushed 함수의 정의이다. Push된 MyButton의 label에 따라서 각각 다른 동작을 하도록 지정하였다. EV 라고 적힌 버튼은 계산, CL이라고 적힌 버튼은 문자 지우기, 그 외의 버튼들은 버튼의 label이 WebUI.mainView에 적히도록 하였으며 expression 변수는 수식 계산을 위해 존재한다.

03 initWidgets 함수 재정의

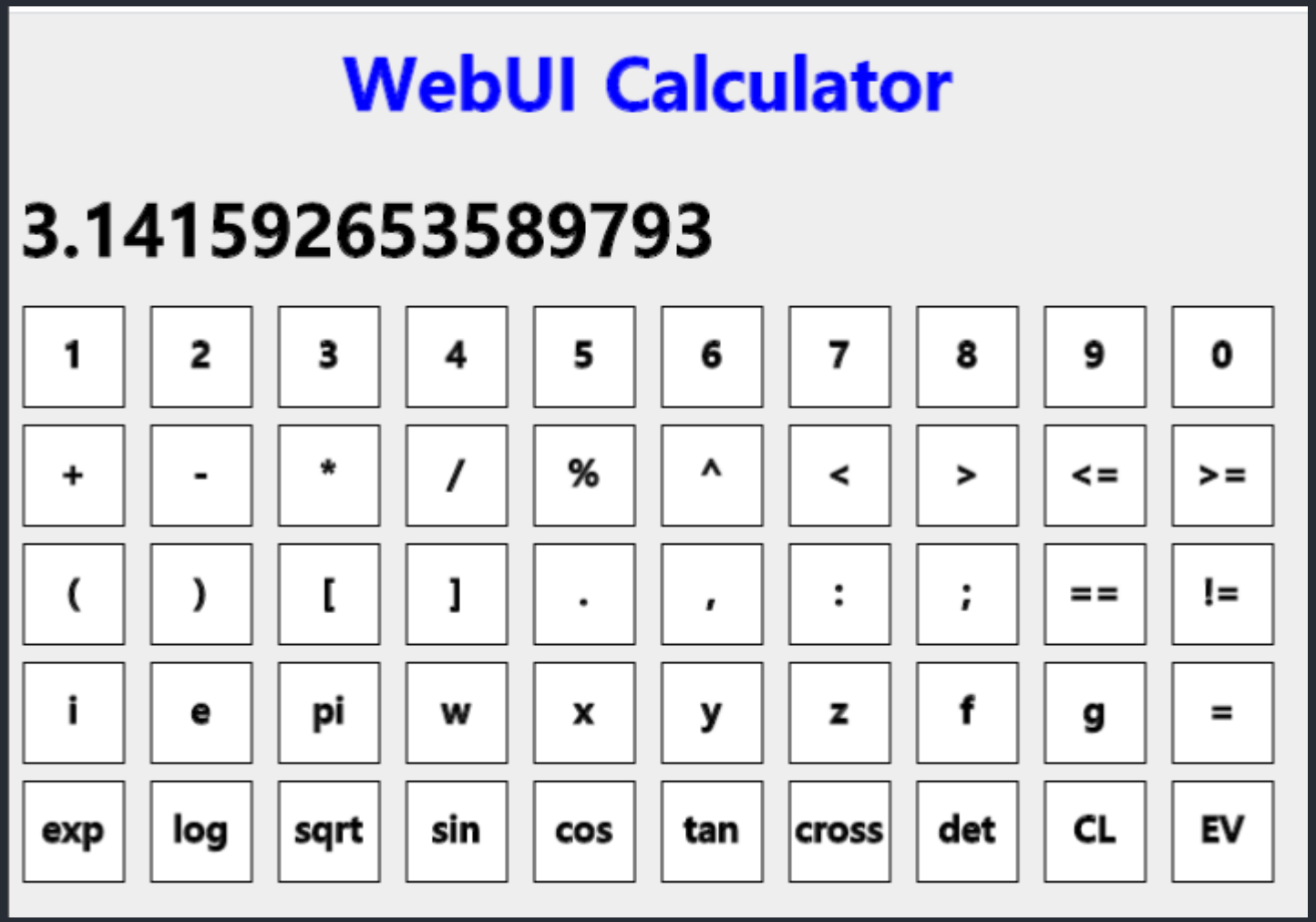
initWidgets

```
WebUI.initWidgets = function() {
    WebUI.app = new WebUI.Row({
      children: [
        new WebUI.Container({
          desired_size: {width: 700, height: 80},
          horizontal_alignment: WebUI.Alignment.CENTER,
          vertical_alignment: WebUI.Alignment.CENTER,
          children: [ new WebUI.Text("WebUI Calculator",40,"bold","blue") ]
        }),
        new WebUI.Container({
          desired_size: {width:700, height:80},
          horizontal_alignment: WebUI.Alignment.LEFT,
          vertical_alignment: WebUI.Alignment.CENTER,
          children: [ WebUI.mainView = new WebUI.Text("",40,"bold","black") ]
        }),
        new WebUI.Column({
          children:[
            new WebUI.Container({
              desired_size:{width:70, height:60},
              horizontal_alignment: WebUI.Alignment.CENTER,
              vertical_alignment: WebUI.Alignment.CENTER,
              children: [ new WebUI.MyPushButton("1", {width:55, height: 55}]
            }),
            new WebUI.Container({
              desired_size:{width:70, height:60},
              horizontal_alignment: WebUI.Alignment.CENTER,
              vertical_alignment: WebUI.Alignment.CENTER,
              children: [ new WebUI.MyPushButton("2", {width: 55, height: 55]
            }),
```

과제에 제시된 조건에 맞게 제목과 화면을 Container가 감싼 Text 위젯으로, 각 버튼들은 모두 Row와 Column에 감싸져 있는 형태로 생성하였다.

화면에 해당하는 Text위젯은 실시간으로 text에 변화가 있어야 하므로 WebUI.mainView라는 변수가 참조하도록 생성하였다.

04 최종 결과



과제에 제시된 것과 동일한 결과 화면을 갖도록 하였고 다음 화면은 pi 버튼을 누른 후 EV 버튼을 누른 결과 화면이다. Math.parser가 제대로 동작함을 알 수 있다.

Chapter 3

WebUI 확장 계산기 구현

01 보다 편리한 기능/인터페이스 제공 – 새로운 상호작용 위젯

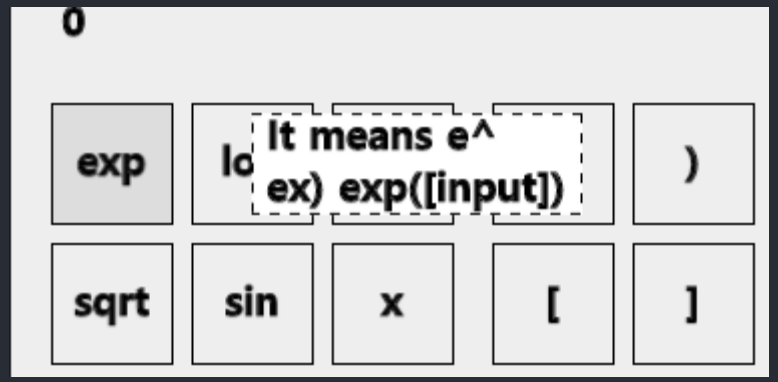
WebUI.Info_Box

```
WebUI.Info_Box.prototype.initVisualItems = function(properties) {
  let text = new fabric.Text(this.label, {
      left:       this.position.left,
      top:        this.position.top,
      selectable: false,
      fontFamily: this.font_family,
      fontSize:   this.font_size,
      fontWeight: this.font_weight,
      textAlign:  this.text_align,
      stroke:     this.text_color,
      fill:       this.text_color,
  });

  //
  let bound = text.getBoundingRect();
  this.position.left = bound.left;
  this.position.top = bound.top;
  this.size.width = bound.width;
  this.size.height = bound.height;


  let line1 = new fabric.Line([this.position.left,this.position.top,this.position.left+this.size.width,this.position.top],{
    strokeDashArray: [5,5],
    stroke: 'black'
  });
  let line2 = new fabric.Line([this.position.left+this.size.width,this.position.top,this.position.left+this.size.width,this.position.top+this.size.height],{
    strokeDashArray: [5,5],
    stroke: 'black'
  });
  let line3 = new fabric.Line([this.position.left+this.size.width,this.position.top+this.size.height,this.position.left,this.position.top+this.size.height],{
    strokeDashArray: [5,5],
    stroke: 'black'
  });
```

WebUI.Info_Box의 initVisualItems 함수이다. Info_Box는 fabric.Text, fabric.Line, fabric.Rect 를 visual items로 갖는다. 따라서 텍스트, 점선, 배경에 해당하는 사각형을 시각적 요소로 갖는다. WebUI.Info_Box는 뒤에 선언될 WebUI.Abs_Container안에 존재할 예정이므로 parent의 size와 position에 기반에 Info_Box의 위치를 결정한다.

01 보다 편리한 기능/인터페이스 제공 – 새로운 상호작용 위젯

WebUI.Info_Box



Exp에 마우스를 올려놓으면 handleMouseEnter 이벤트가 발생하게 되고, handleMouseEnter 이벤트 핸들러가 setInteraval 함수를 호출해 시간을 측정하고 일정 시간이 지나면 WebUI.Info_Box 가 표시되도록 하였다. 일정 시간이 지나기 전에 handleMouseExit 이벤트가 발생하면 시간 측정을 종료해 WebUI.Info_Box가 출력되지 않는다.

02 보다 편리한 기능/인터페이스 제공 – 새로운 레이아웃 위젯

WebUI.ABS_Container

```
WebUI.Abs_Container = function(properties){
  WebUI.Widget.call(this, properties);
  this.type = WebUI.WidgetTypes.ABS_CONTAINER;
}

WebUI.Abs_Container.prototype = Object.create(WebUI.Widget.prototype);
WebUI.Abs_Container.prototype.constructor = WebUI.Abs_Container;

WebUI.Abs_Container.prototype.extendSizeChildren = function(size, child_size) {

    if(size.width < child_size.width) size.width = child_size.width;
    size.height += child_size.height;
    return size;
}

WebUI.Abs_Container.prototype.calcNextPosition = function(position, size) {

    let next_left = position.left;
    let next_top = position.top + size.height;
    return {left: next_left, top: next_top};
}
```

WebUI.Abs_Container는 기본적으로 Container의 특징을 갖는다. 하지만 다른 Container들이 Row와 Column에 속해서 위치가 지정되는 반면, Abs_Container는 특정 위젯의 위치를 기반으로 위치가 결정된다.

02 보다 편리한 기능/인터페이스 제공 – 새로운 레이아웃 위젯

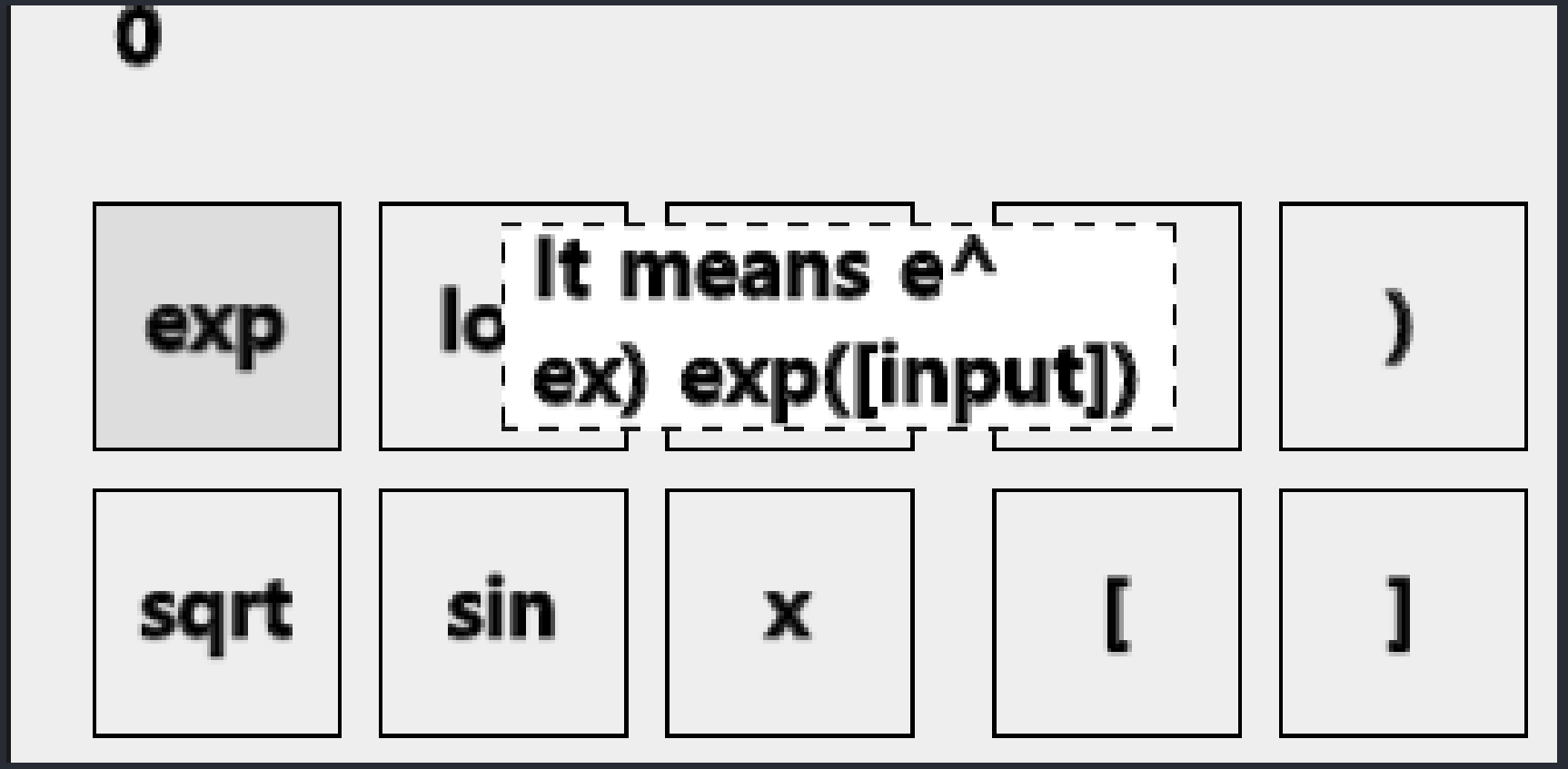
WebUI.ABS_Container

```
var startTimer = function(widget,label){
    WebUI.intervalReturn = setInterval(function(){
    WebUI.tmrCount++;
    if(WebUI.tmrCount == 3 && label != ""){
      WebUI.abs_container = new WebUI.Abs_Container({
        position: {left:widget.position.left+100, top:widget.position.top},
        desired_size: {width:60, height:60},
        horizontal_alignment: WebUI.Alignment.CENTER,
        vertical_alignment: WebUI.Alignment.CENTER,
        children: [ new WebUI.Info_Box(label,{width:60, height:60}) ]
      });

      WebUI.abs_container.children.forEach(widget => {
          widget.initVisualItems();
      });
      WebUI.abs_container.layout();
      WebUI.canvas.requestRenderAll();
    }
  }, 500);
}
```

startTimer 함수는 MyPushButton.prototype.handleMouseEnter이벤트 핸들러에서 호출되는 함수이며, 500밀리 세컨드마다 한 번 호출되고 호출될 때마다 tmrCount 변수를 1씩 증가시킨다. 그러다가 tmrCount 변수가 3이 되면 Abs_Container를 생성하고, 그 내부에 Info_Box를 생성한다. Info_Box의 label을 mouseEnter 이벤트를 발생시킨 버튼의 label에 따라서 결정되며, 버튼의 설명에 대한 문자열을 갖는다.

02 보다 편리한 기능/인터페이스 제공 – 새로운 레이아웃 위젯

WebUI.ABS_Container



03 보다 편리한 기능/인터페이스 제공 – 새로운 상호작용 위젯

Screen shot

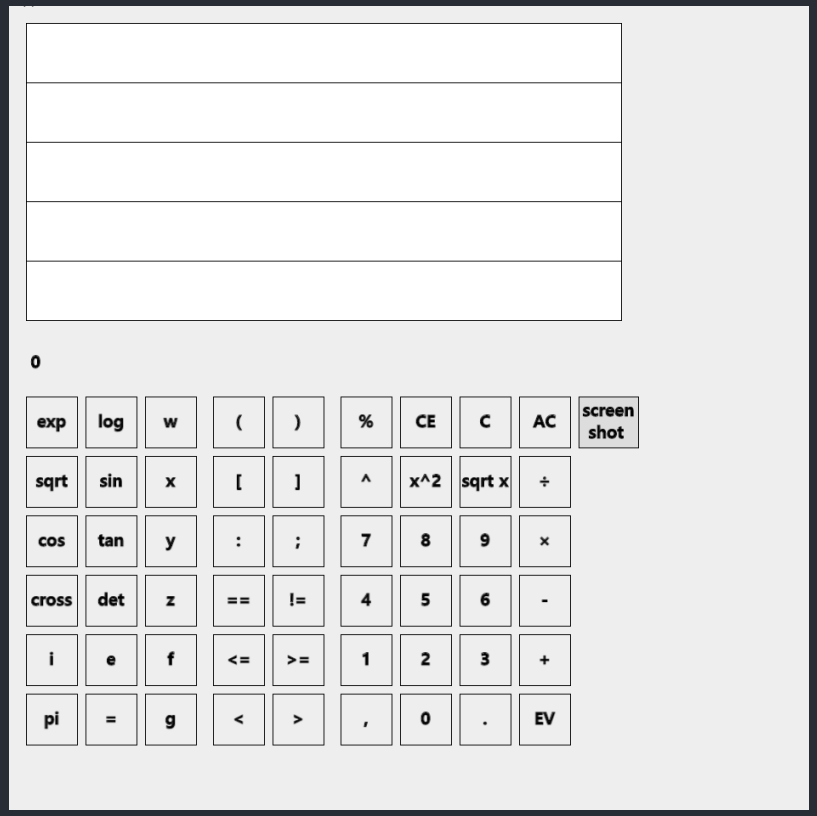
```
else if(this.label == "screen\n shot"){
  html2canvas(document.querySelector("#capture")).then(canvas=>{
    saveAs(canvas.toDataURL('image/png'),"calculator image.png");
  });
}
```

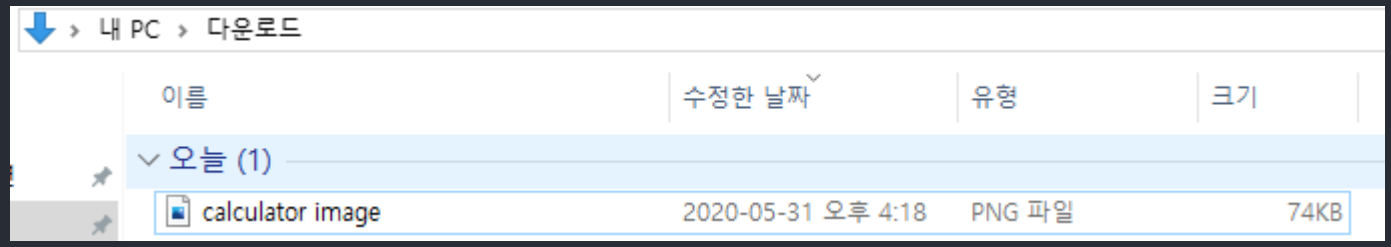
```
var saveAs = function(uri, filename){
  var link = document.createElement('a');
  if(typeof link.download == 'string'){
    link.href = uri;
    link.download = filename;
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);
  }
  else{
    window.open(uri);
  }
}
```

Screen shot 이라고 적인 버튼을 클릭 시 html2canvas 라이브러리의 함수를 이용해 canvas 전체를 캡처해 calculator image.png 로 저장한다.

03 보다 편리한 기능/인터페이스 제공 – 새로운 상호작용 위젯

Screen shot





왼쪽은 저장된 이미지, 위는 다운로드 폴더에 저장된 이미지의 캡쳐이다.

04 보다 편리한 기능/인터페이스 제공 – 새로운 상호작용 위젯

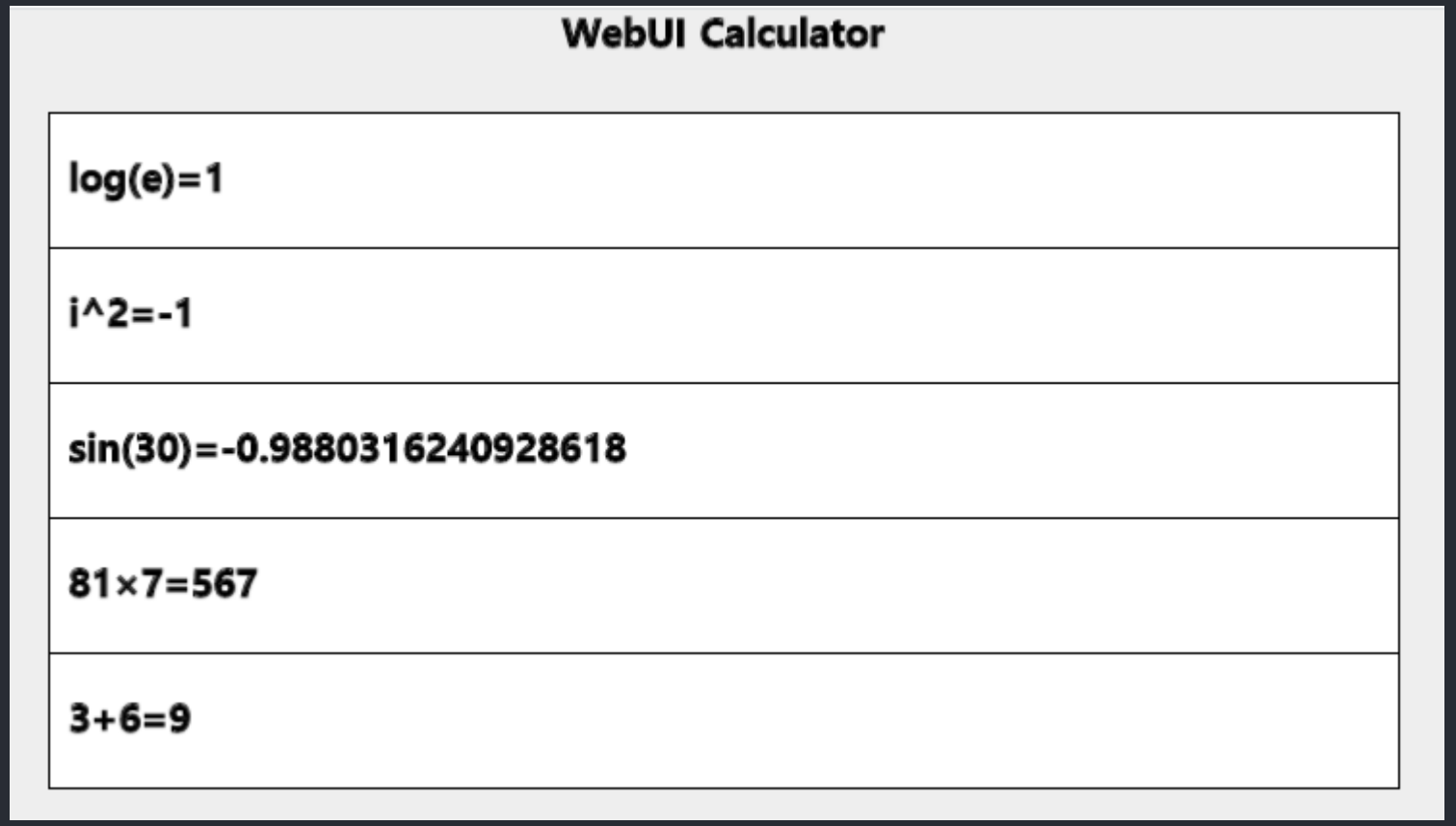
History View

```
WebUI.addHistory = function(express, result){
  let history = express + "=" + result;
  let index;
  for(let i=4; i>=0; i--){
    if(WebUI.historyView[i].visual_items[1].text == ""){
      index = i;
      break;
    }
    index = -1;
  }
  if(index != -1){
    WebUI.historyView[index].visual_items[1].text = history;
  }
  else{
    for(let i=0;i<4;i++){
      WebUI.historyView[i].visual_items[1].text =
        WebUI.historyView[i+1].visual_items[1].text;
    }
    WebUI.historyView[4].visual_items[1].text = history;
  }
}
```

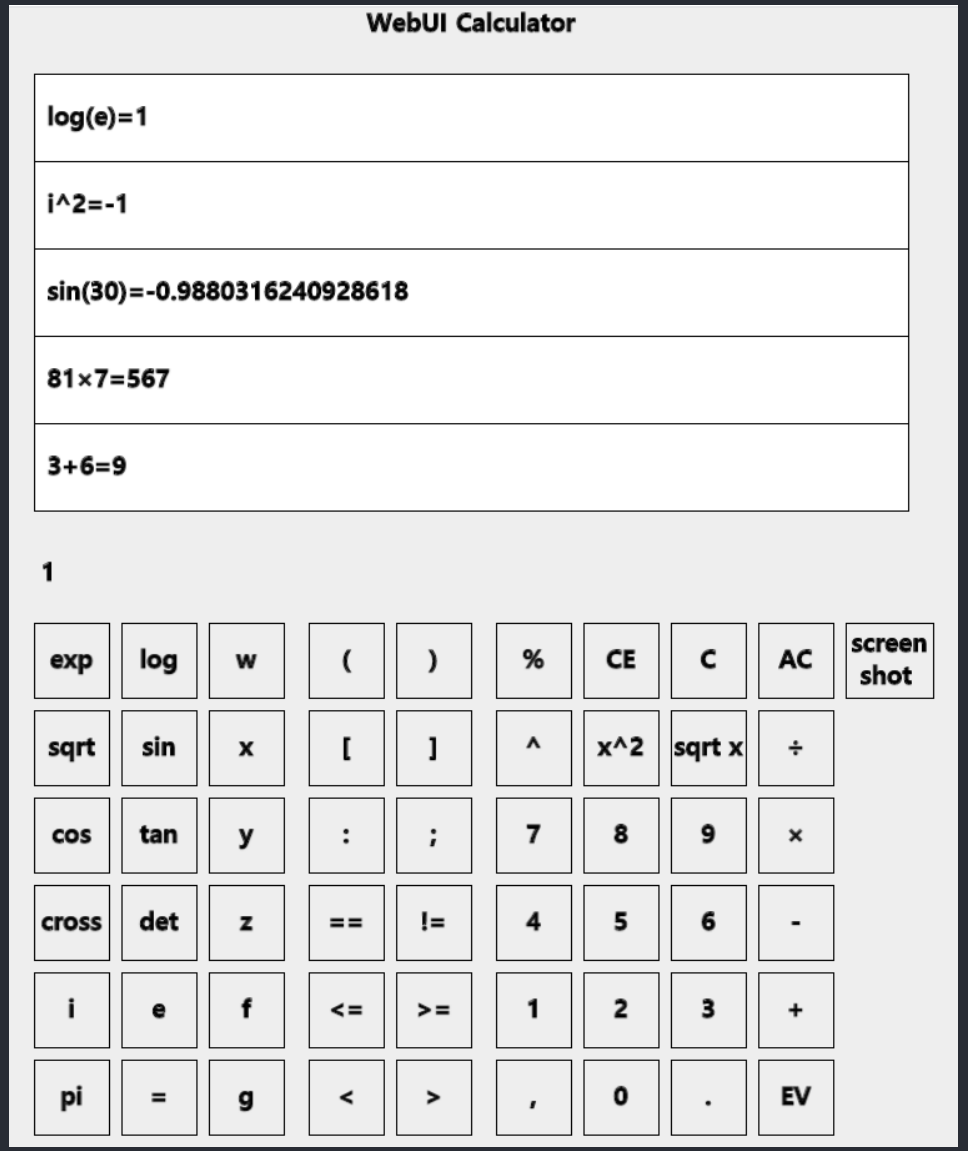
addHistory 함수는 WebUI.historyView의 각 텍스트를 계산의 결과가 저장되도록 바꿔주는 함수이다. 비어있는 historyView가 있으면 비어있는 historyView에 계산결과를 저장하고, 비어있는 것이 없으면 위에서 부터 계산과정을 지우고 아래쪽에 새로운 계산 결과를 추가한다.

04 보다 편리한 기능/인터페이스 제공 – 새로운 상호작용 위젯

History View



05 최종 결과화면


WebUI Calculator
log(e)=1
i^2=-1
sin(30)=-0.9880316240928618
81×7=567
3+6=9
1
exp log w ( ) % CE C AC screen shot
sqrt sin x [ ] ^ x^2 sqrt x ÷
cos tan y : ; 7 8 9 ×
cross det z == != 4 5 6 -
i e f <= >= 1 2 3 +
pi = g < > , 0 . EV

논의

① MyPushButton, Abs_Container, Info_Box, historyView 등을 새롭게 정의하고 페이지의 스크린샷을 저장하는 기능이 잘 구현되었다. 버튼 간의 간격도 기능별로 구분되도록 하여 사용성을 높였다.

② 자체적인 평가를 하자면 과제에서 요구하는 모든 요건을 충족하였으며, 새로운 상호작용 위젯과 레이아웃 위젯을 정의하였기 때문에 양호한 수준이라고 보여진다.

③ 향후 개선점

- 어떠한 부분에서 발생하는 오류인지 모르겠으나 계산기를 계속해서 사용하면 클릭 하는 위치와 프로그램이 받아들이는 위치에 오차가 발생하는 오류를 발견하였다. 하지만 해결하지 못하였다.
- Png나 jpg 이미지를 사용하고 싶었으나 배경색을 처리하지 못하여서 오히려 계산기의 디자인적인 느낌을 반감시키는 것 같아 리소스를 많이 사용하지 못하였다.

감사합니다