공통 API 가이드

Last update: 2024/07/02

www.softbase.co.kr

목차

공통 API 가이드 1
화면 스크립트 1

공통 API 가이드

컴포넌트 기본 공통 API 및 기본/특수 속성 샘플 가이드 화면이다.

API(Application Programming Interface)는 컴포넌트가 제공하는 기능을 제공하는 함수이다.

API는 “ 컴포넌트이름”.“API이름 ” 형식으로 일반javaScript 함수 호출 방식으로 개발한다.

일반 속성은 컴포넌트가 제공하는 특성을 의미하며, prop API를 통해서 값을 구하거나, set~으로 시작되는 API를 통해서 값을 저장한다.

특수 속성은 컴포넌트가 제공하는 특수한 속성을 의미하며, “ 컴포넌트이름 ” . “ 속성이름 ” 형식으로 값을 구하거나 저장한다.

관련 속성으로 value, innertctrl, screen 배열, member 배열이 있다.

관련 API로 prop, val, valbyname, columnprop, columnpropbyname, itemprop, itempropbyname이 있다.

템플릿 위치: /HTML5/COMPONENT/COMMON/api_basic

템플릿 파일

- api_basic.xml
- api_basic.js
- 새창으로 실행

화면 스크립트

```
// "value" 버튼 이벤트
function btn_value_on_mouseup(objInst)
{
    factory.consoleprint("변경전 fld_data 값 = " + this.fld_data.value);
    this.fld_data.value = "NEW";
    factory.consoleprint("변경후 fld_data 값 = " + this.fld_data.value);
}

// "val" 버튼 이벤트
function btn_val_on_mouseup(objInst)
{
    factory.consoleprint("변경전 fld_data 값 = " + this.fld_data.val());
    this.fld_data.val("NEW");
    factory.consoleprint("변경후 fld_data 값 = " + this.fld_data.val());
}

// "prop" 버튼 이벤트
function btn_prop_on_mouseup(objInst)
```

```
{
    var nPropValue, bPropValue, sPropValue, aPropValue;

    nPropValue = this.fld_data.prop("border"); // 숫자형
    sPropValue = this.fld_data.prop("default_value"); // 문자형
    bPropValue = this.fld_data.prop("enable"); // 불린형
    aPropValue = this.fld_data.prop("font"); // 배열형

    screen.alert(
        "border 속성값 = " + nPropValue + ", " + typeof nPropValue + "\n" +
        "default_value 속성값 = " + sPropValue + ", " + typeof sPropValue +
"\n" +
        "enable 속성값 = " + bPropValue + ", " + typeof bPropValue + "\n" +
        "font 속성값 = " + aPropValue.join(", ") + ", " + typeof aPropValue);
}

// "GRID val" 버튼 이벤트
function btn_grid_val_on_mouseup(objInst)
{
    var row_index, column_index, row_count, column_count;

    row_count = this.grd.getrowcount();
    column_count = this.grd.getcolumncount();

   // 그리드 컬럼 인덱스를 기준으로 값을 구함
    for (row_index = 0; row_index < row_count; row_index++) {
        for (column_index = 0; column_index < column_count; column_index++)
{
            factory.consoleprint(row_index + ":" + column_index + "=" +
                this.grd.val(row_index, column_index));
        }
    }

   // 그리드 컬럼 인덱스를 기준으로 값을 저장
    for (row_index = 0; row_index < row_count; row_index++) {
        for (column_index = 0; column_index < column_count; column_index++)
{
            this.grd.val(row_index, column_index, "NEW", false, true, true,
true);
        }
    }

    this.grd.refresh();
}

// "GRID valbyname" 버튼 이벤트
function btn_grid_valbyname_on_mouseup(objInst)
{
    var row_index, column_index, row_count, column_count;

    row_count = this.grd.getrowcount();
```

```
  // 그리드 컬럼 이름을 기준으로 값을 구함
    for (row_index = 0; row_index < row_count; row_index++) {
        factory.consoleprint(row_index + ":" + "COL_A" + "=" +
            this.grd.valbyname(row_index, "COL_A"));
        factory.consoleprint(row_index + ":" + "COL_B" + "=" +
            this.grd.valbyname(row_index, "COL_B"));
    }

  // 그리드 컬럼 이름을 기준으로 값을 저장
    for (row_index = 0; row_index < row_count; row_index++) {
        this.grd.valbyname(row_index, "COL_A", "NEW", false, true, true,
true);
        this.grd.valbyname(row_index, "COL_B", "NEW", false, true, true,
true);
    }

    this.grd.refresh();
}

// "GRID columnprop" 버튼 이벤트
function btn_grid_columnprop_on_mouseup(objInst)
{
    var column_count, column_index;

    column_count = this.grd.getcolumncount();
    for (column_index = 0; column_index < column_count; column_index++) {
    // 컬럼 인덱스를 기준으로 컬럼 속성을 구함
        factory.consoleprint(column_index + " column input_type = " +
            this.grd.columnprop(column_index, "input_type"));
    }
}

// "GRID columnpropbyname" 버튼 이벤트
function btn_grid_columnpropbyname_on_mouseup(objInst)
{
    var column_count, column_index;

  // 컬럼 이름을 기준으로 컬럼 속성을 구함
    factory.consoleprint("COL_A" + " column input_type = " +
        this.grd.columnpropbyname("COL_A", "input_type"));
    factory.consoleprint("COL_B" + " column input_type = " +
        this.grd.columnpropbyname("COL_B", "input_type"));
}

// "GRID itemprop" 버튼 이벤트
function btn_grid_itemprop_on_mouseup(objInst)
{
    var row_index, column_index, item_prop_value;

    row_index = 0;
    column_index = 0;
  // 컬럼 인덱스를 기준으로 아이템 속성값 반환
    factory.consoleprint("item input_type = " +
        this.grd.itemprop(row_index, column_index, "input_type");
}

// "GRID itempropbyname" 버튼 이벤트
function btn_grid_itempropbyname_on_mouseup(objInst)
{
    var row_index;

    row_index = 0;

  // 컬럼 이름을 기준으로 아이템 속성값 반환
    factory.consoleprint("item input_type = " +
        this.grd.itempropbyname(row_index, "COL_A", "input_type");
}

// "XDATASET val" 버튼 이벤트
function btn_xdatasetval_on_mouseup(objInst)
{
    var row_index, column_index, row_count, column_count;

  // 데이터셋 행/열 갯수를 구함
    row_count = this.DS_DATA.getrowcount();
    column_count = this.DS_DATA.getcolumncount();

  // 데이터셋 컬럼 인덱스를 기준으로 값을 구함
    for (row_index = 0; row_index < row_count; row_index++) {
        for (column_index = 0; column_index < column_count; column_index++)
{
            factory.consoleprint(row_index + ":" + column_index + "=" +
                this.DS_DATA.val(row_index, column_index));
        }
    }

  // 데이터셋 컬럼 인덱스를 기준으로 값을 저장
    for (row_index = 0; row_index < row_count; row_index++) {
        for (column_index = 0; column_index < column_count; column_index++)
{
            this.DS_DATA.val(row_index, column_index, "NEW", false);
        }
    }

  // 데이터셋에 링크된 컴포넌트에 이벤트 발생(Refresh) 처리
    this.DS_DATA.fireevent();
}

// "XDATASET valbyname" 버튼 이벤트
function btn_xdatasetvalbyname_on_mouseup(objInst)
{
    var row_index, column_index, row_count, column_count;
    row_count = this.DS_DATA.getrowcount();

  // 데이터셋 컬럼 이름을 기준으로 값을 구함
    for (row_index = 0; row_index < row_count; row_index++) {
        factory.consoleprint(row_index + ":" + "COL_A" + "=" +
            this.DS_DATA.valbyname(row_index, "COL_A"));
        factory.consoleprint(row_index + ":" + "COL_B" + "=" +
            this.DS_DATA.valbyname(row_index, "COL_B"));
    }

  // 데이터셋 컬럼 이름을 기준으로 값을 저장
    for (row_index = 0; row_index < row_count; row_index++) {
        this.DS_DATA.valbyname(row_index, "COL_A", "NEW", false);
        this.DS_DATA.valbyname(row_index, "COL_B", "NEW", false);
    }

  // 데이터셋에 링크된 컴포넌트에 이벤트 발생(Refresh) 처리
    this.DS_DATA.fireevent();
}

// "TAB innerctrl 속성" 버튼 이벤트
function btn_tabinnerctrl_on_mouseup(objInst)
{
    var tab_item_index, tab_item_count;

    // UDC 화면이 링크되지 않은 경우 제외
    if (!this.tab_udc.innerctrl) { return; }

  // 링크된 UDC 화면의 udc_api 속성에서 정의된 함수 호출
    this.tab_udc.innerctrl.udc_api_a("param_a");
}

// "screen 배열 속성" 버튼 이벤트
function btn_tabscreen_on_mouseup(objInst)
{
    var tab_item_index, tab_item_count;

  // 탭 아이템 갯수를 구함
    tab_item_count = this.tab_linkscreen.gettabitemcount();
    for (tab_item_index = 0; tab_item_index < tab_item_count;
tab_item_index++) {
    // 화면이 링크되지 않은 탭 아이템은 제외
        if (!this.tab_linkscreen.screen[tab_item_index]) { continue; }

    // 링크되 화면 오브젝트의 "getscreenurl" 함수 호출
        factory.consoleprint(tab_index + " screen url = " +
            this.tab_linkscreen.screen[tab_item_index].getscreenurl());
    }
}

// "member 배열 속성" 버튼 이벤트
function btn_tabmember_on_mouseup(objInst)
```

```
{
    var tab_item_index, tab_item_count;

  // 탭 아이템 갯수를 구함
    tab_item_count = this.tab_linkscreen.gettabitemcount();
    for (tab_item_index = 0; tab_item_index < tab_item_count;
tab_item_index++) {
    // 화면이 링크되지 않은 탭 아이템은 제외
        if (!this.tab_linkscreen.member[tab_item_index]) { continue; }

    // 링크되 화면의 "member_func_a" 함수 호출
        this.tab_linkscreen.member[tab_item_index].member_func_a("param_a");
    }
}
```