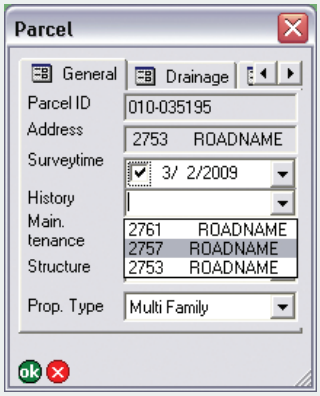
Little Effort, Big Savings

Add a fill-form-from-history tool to ArcPad in two steps

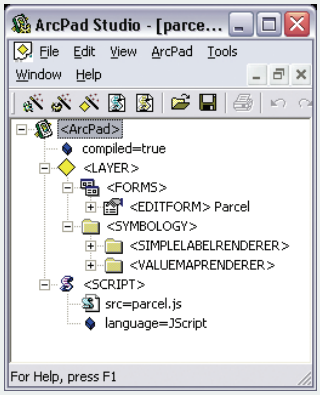
By Mel Yuanhang Meng, DLZ Ohio

How the Tool Was Developed

For a sanitary sewer improvement project, DLZ Ohio needed to inspect the condition of houses in the study area to evaluate the potential for storm water entering the sanitary sewer system. Initially, condition codes for whole blocks were assigned as project team members drove through neighborhoods in the study area. This process didn't take long but missed lots of details, and ratings were not consistent from one neighborhood to another. Was it possible to collect data for each property without blowing the budget?



The fill-form-from-history tool



The parcel APL file

ArcPad came to the rescue. Switching from a paper to an electronic format immediately freed the team from entering the address and other information that was already available for each house. However, compared with driving through neighborhoods, this was a much slower process. The project budget simply wouldn't cover this expense.

However, houses on the same block are very similar and many entries were repeated for every house. This discovery led the team to develop a fill-form-from-history tool that easily cut the time needed for inspecting a block by 50 to 80 percent. Selecting a previously inspected address listed in the History drop-down list populated the values from that address into the current form. For a form with 30 fields, the time savings provided by this tool were significant.

Add the Tool to Your Form in Two Steps

The fill-form-from-history tool was developed using ArcPad Studio 7 and tested on ArcPad 7. Any ArcPad form developed using an APL file and a JavaScript file can be easily modified to use this tool in just two steps.

1. Add a history combobox control in the form.
2. Reference the JavaScript file that defines a global variable that stores the history and functions that update the history control, read, and save form entries to the history.

The form associated with a shapefile is defined by an APL file (layer definition file or *.apl) and a JavaScript file (*.js) that defines the global variables and the functions. An APL file is automatically created when a shapefile is exported using the ArcPad Data Manager extension for ArcGIS Desktop. The APL file is later customized using ArcPad Studio.

An APL file is an XML file that defines the symbology data entry forms and references any additional script files. The APL file shares the same base name as the shapefile with an .apl file extension. Table 1 explains the control flow and event names for the fill-form-from-history tool. As a shapefile is loaded into ArcPad, the global variables and functions defined in the JavaScript file are automatically loaded. The function is called when the corresponding event is triggered.

Event (Event name)	**Action** (Function name)
Add shapefile into ArcPad (NA)	**Loading global variables** (NA)
Form is on load (onload)	**Populate the history control with previous entries** (draw_history)
Form is closed and saved (onok)	**Saving the entry into the global history variable** (save_history)
History control on select change (onselchange)	**Populate the form with values for the selected entry** (fill_form)

Table 1: Control flow of fill-form-from-history tool

What You Will Need
- ArcPad 7 or higher
- ArcPad Application Builder 7 or higher
- Sample data downloaded from *ArcUser Online*
- An unzipping utility such as WinZip

A simple customization to an ArcPad form greatly increased the efficiency of house inspections and allowed the author's consulting firm to collect more accurate data more rapidly and remain under budget for a project. This tool can easily be added to existing ArcPad forms following the instructions in this article.

Do It Yourself

The following instructions guide the reader through the process of installing the fill-form-from-history tool in an ArcPad form. Download frmhis.zip from *ArcUser Online* (www.esri.com/arcuser) and unzip it. This archive contains the sample data and application that illustrate the process of adding the fill-form-from-history tool.

Step 1:

Add a history Combobox control in the form.

1. Open the APL file for your form in ArcPad Studio.
2. Double-click the <EDITFORM> tag to display the form.
3. Click on the Controls palette and drag a Combobox control over to the form. Name the combo box History.
4. Add a label control and label the combo box History.
5. Double-click on the combo box and click the Events tab to bring up the Control Properties dialog box.
6. Scroll to the onselchange event and add the following event script: fill_form (ThisEvent.Object.ListIndex);.
7. Click OK and choose Form > Form Properties from the form's dialog box. Click the Events tab, select the onload event, and reference the draw history script draw_history();.
8. Select the onok event to reference the script save_history () script.

```
//*Configuration Section -- Start*

//prefix for control name. Prefix controls that you want to save in history with the character defined below.
//eg. a control with a name _structure will be saved while structure won't.
_flag = "_";
//history control definition. The name of the page it is on and the name of the control
//Below defines a control on PAGE1 with a name of history
_hctl = {
    page: "PAGE1",
    ctl: "history"
};
//The field used to identify a previous entry.
//Below uses the address as the identifier. It is the value of a control txtAddress on PAGE1
_hctlsrc = {
    page: "PAGE1",
    ctl: "txtAddress"
};

//*Configuration Section -- End*

_history = []; //global array storing the history of last 5 entered records.
_record = "desc"; //global variable, the key to retrieve the address from a record.
```

Listing 1: Global variables

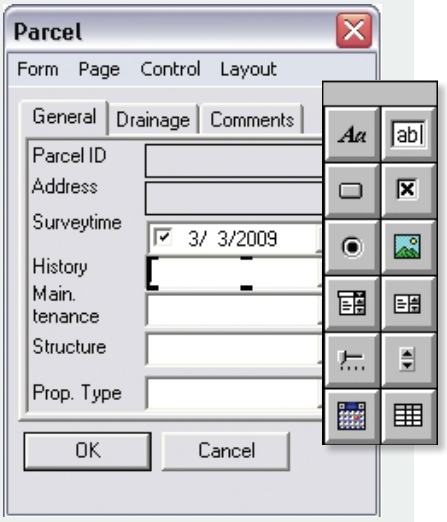
Step 2:

Reference the JavaScript file.

Its powerful features make JavaScript an ideal scripting language for this application. If VBScript is used instead, the code will be much longer and more complicated.

1. Copy the parcel.js file (which was included in the sample dataset) to the same folder that contains the shapefile and APL file for your project.
2. In the APL file, double-click on the <SCRIPT> tag and set the scr to parcel.js and language to JScript.
3. Open parcel.js using ArcPad Studio and inspect the code.

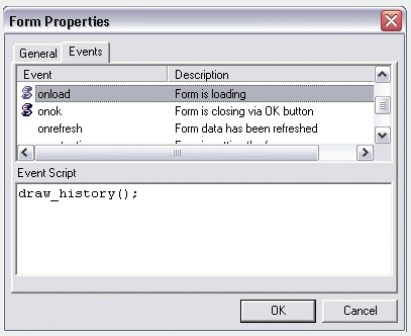
Continued on page 58

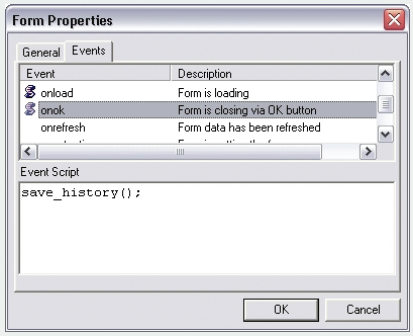


Adding the history Combobox control

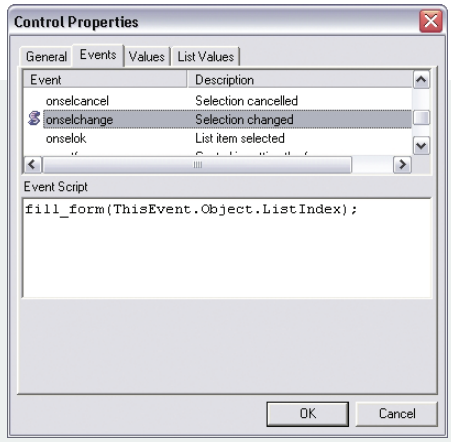
Little Effort, **Big Savings**

Continued from page 57





Setting the Form events



Setting the History control events

The configuration and global variables are shown in Listing 1. In this application, the _flag variable is used to filter the controls that need to be populated by the tool. Make sure the names of the controls that will be populated by the tool start with the underscore character "_".

Any ArcPad form developed using an APL file and a JavaScript file can be easily modified to use this tool in just two steps.

The save_history() function is called when the OK button of a form is clicked as shown in Listing 2. As the program loops through the controls page by page, it scans the name of the control. If the name starts with a character defined by the _flag variable, its value is saved in the associated values array (which is indexed by the control's name). Then the values array is pushed into the _history global variable. The program also checks the number of entries in the _history variable and removes older entries, if necessary.

As shown in Listing 3, the fill_form() function is fired when the selected item in the history control has changed. The index of the selected item will be passed to this function. Since the most recent entry is listed as the top item in the history control, the order of the items in the _history variable needs to be reversed to match the index from the history control.

As shown in Listing 4, the draw_history() function is fired as the form is loading. It populates the history control with previous entries.

```
//save the values of the current entry into the history variable
//triggered by the Form onok event
function save_history(){
    var ctls, i, j, pgs, pg, m;
    var values = {}; //An associate array to store all the values
    pgs = ThisEvent.Object.Pages; //Reference to all the pages
    for (j = 1; j < pgs.Count + 1; j++) {
        //loop all the pages
        pg = pgs.Item(j);//current page
        ctls = pg.Controls;//all the controls on the current page
        for (i = 1; i < ctls.Count + 1; i++) {
            //loop all the controls
            m = ctls.Item(i).name.toLowerCase();//In case the name control is not case sensitive.
            if (ctls.Item(i).name.substr(0, 1) == _flag) {
                //filter the controls, if it starts with "_", then remember it
                switch (ctls.Item(i).Type) {
                    //depending on the control's type, find the value need to be remembered.
                    //the data looks like, values[control name] =  "contro value"
                    case "COMBOBOX":
                        values[m] = ctls.Item(i).ListIndex;
                        break;
                    case "CHECKBOX":
                        values[m] = ctls.Item(i).Value;
                        break;
                    case "EDIT":
                        values[m] = ctls.Item(i).Value;
                        break;
                    case "RADIOBUTTON":
                        values[m] = ctls.Item(i).Value
                        break;
                }
            }
        }
    }
    //Now all the data are saved into the values variable
    //add the address to the values, so it can be populated for the history combobox
    values[_record] = pgs(_hctlsrc.page).Controls(_hctlsrc.ctl).Value;
     //Before saving values to the global history variable, check the length of the _history
variable.
    //Only keep most recent 5 entries
    if (_history.length > 4) {
        _history.shift();
.....//get rid of the oldest one
    }
    _history.push(values);
....//add the current one on the top
}
```

Listing 2: Global variables

```
//Populate the current form with values from the selected entry in the history combobox
// triggered by the history combox onselchange event
//--ind, index of the history
function fill_form(ind){
    var ctls, i, j, pgs, pg, r, f, m;
    //reference to the pages and loop each page
    pgs = ThisEvent.Object.Parent.Parent.Pages;
    for (i = 1; i < pgs.Count + 1; i++) {
        pg = pgs.Item(i);//current page
        //To change the values of a control, the page the control is on must be activated.
        //if it is not the first page, activate the page
        if (i > 1) {
            pg.Activate();
        }
        //as the most recent entry is listed as the first item in the in the combox
       //To match the an item in the _history array using the index from the combox, the order
need to be reversed.
        _history.reverse();
        r = _history[ind];//get the record from the _history
        _history.reverse();//reverse it back to normal order
        //reference to all the controls on the current page and loop through them
        //to populate values from history
        ctls = pg.Controls;
        for (j = 1; j < ctls.Count + 1; j++) {
            //get the name of the control and filter out those not saved.
            m = ctls.Item(j).name.toLowerCase();
            if (m.substr(0, 1) == _flag) {
                //if the control has ListIndex attribute, it must be a combobox
                if (ctls.Item(j).ListIndex) {
                    ctls.Item(j).SetFocus();
                    ctls.Item(j).ListIndex = r[m];
                }
                else {
                    //if it is not a combobox, it should have a Value field.
                    ctls.Item(j).Value = r[m]
                }
            }
        }
    }
    //Go back to the first page
    pgs.Item(1).Activate();
    }
```

Listing 3: The fill_form function

```
//Populate the history control with previous entries
//Triggered by Form onload event
function draw_history(){
    var ctl = ThisEvent.Object.Pages(_hctl.page).Controls(_hctl.ctl);
    //the history control
    var l = _history.length;
    ctl.Clear();//clear all existing items in it
    for (var i = 0; i < l; i++) {
        //populate the list, so that the last record are the first in the list
        r = _history[l - i - 1];
        ctl.AddItem(r[_record], i);//add the item in the list
    }
}
```

Listing 4: Draw_history function

Conclusion

A fill-form-from-history tool was developed for ArcPad using JavaScript. The tool can be easily adapted for existing ArcPad applications in only two steps. The advanced features of JavaScript achieve more functionality with less and simpler code, making it an excellent choice for ArcPad customization.

About the Author

Mel Yuanhang Meng is the CAD/GIS manager for DLZ Ohio, an architectural and engineering consulting firm. He works in the field of water/wastewater infrastructure design, maintenance, and modeling. His professional interests include asset management, field data collection, and Web-based GIS application development. He holds two master's degrees, one in geography and the other in environmental engineering, from the University of Cincinnati.

More Information

Working with ArcPad 7, a free Web course, and *Customizing ArcPad,* another Web course, are offered by ESRI. Visit www.esri.com/training to enroll.