

Creating Custom Commands in FormsViewer

Last Updated: January 2025

Introduction

FormsViewer allows you to create custom commands that extend the qRules command set. These custom commands can be used in form rules just like built-in commands.

You can learn more about creating and using custom commands by watching this webinar:

[Create a Custom qRules Command](#)

Command Registration Basics

Registering a Command

To create a custom command, you use the qRules() method from your FormsViewer instance:

```
var qr = instance.qRules();  
qr.registerCommand(name, definition, action);
```

The registerCommand method takes three parameters:

1. name - The command name (will be prefixed with “Custom” when used in your form)
2. definition - Command parameter definitions (optional)
3. action - The function that executes when the command runs

Defining Parameters

The definition parameter tells FormsViewer what parameters your command accepts:

```
// Simple parameter list  
{ parameters: ['paramName1', 'paramName2'] }  
  
// Parameters with descriptions  
{ parameters: [  
  'simpleParam',  
  {  
    name: 'complexParam',  
    description: 'What this parameter does'  
  }  
] }
```

You can omit the definition parameter if your command doesn't need parameters.

Working with Parameters

Your command receives a paramProvider object that lets you access parameter values:

```
function(params) {  
    // Get parameters with default values if not provided  
    var text = params.get('paramName', '');           // string  
    var flag = params.getBool('flagName', false);    // boolean  
    var count = params.getInt('countName', 0);       // integer  
    var value = params.getNumber('valueName', 0);    // number  
}
```

Returning Results

Your command can return results in several ways:

1. Simple Values

```
return "some text";           // Returns text  
return 42;                    // Returns number  
return true;                  // Returns boolean
```

2. Status Objects

```
return {  
    success: true,           // Required  
    result: "some value",    // Optional  
    error: "error message"  // Optional  
};
```

3. Async Results (Promises)

```
return somePromise.then(function(result) {  
    return result;  
});
```

4. No Result

```
return null; // Command completed successfully
```

Complete Example

Here's a simple command that concatenates two strings:

```
qr.registerCommand(  
    'Concat2Strings',  
    { parameters: ['String1', 'String2'] },  
    function(params) {  
        return params.get('String1') + params.get('String2');  
    }  
);
```

This creates a command that can be used in rules as: CustomConcat2Strings(/String1="Hello", /String2=" World")

Error Handling

Your command can handle errors in two ways:

1. Throw an error:

```
throw new Error('Something went wrong');
```

2. Return an error object:

```
return {  
    success: false,  
    error: 'Something went wrong'  
};
```

FormsViewer Custom Command Examples

Example 1: Multiple Command Types

```
(function () {  
    "use strict";  
  
    function delay(ms) {  
        return new Promise(function (resolve) {  
            setTimeout(resolve, ms);  
        });  
    }  
  
    function addCommands(instance) {  
        var qr = instance.qRules();  
  
        // Example 1: Command with URL handling and error checks  
        qr.registerCommand(  
            'SetBackground',  
            { parameters: ['url', 'reset'] },  
            function (params) {  
                var url = params.get('url');  
                var reset = params.getBool('reset');  
  
                // Error handling examples  
                if (!(reset || url)) {  
                    throw new Error('Must specify /url or /reset=true');  
                }  
                if (reset && url) {  
                    return {  
                        success: false,  
                        error: 'Cannot specify /url and /reset=true together.'  
                    };  
                }  
            }  
        );  
    }  
})
```

```

        };
    }

    var style = document.body.style;

    if (reset) {
        style.backgroundImage = null;
    } else {
        style.backgroundImage = 'url("' + url.replace(/"/g, '\\\\') + '"

    style.backgroundSize = 'contain';
    style.backgroundPosition = 'center center';
    }

    return { success: true };
}
);

// Example 2: Mathematical operation
qr.registerCommand(
    'Power',
    {
        parameters: [
            'base',
            { name: 'exponent', description: 'The exponent' },
        ],
    },
    function (params) {
        var base = params.getNumber('base');
        var exponent = params.getNumber('exponent');

        return Math.pow(base, exponent);
    }
);

// Example 3: String manipulation
qr.registerCommand(
    'ReverseString',
    { parameters: ['value'] },
    function (params) {
        return params.get('value').split('').reverse().join('');
    }
);

// Example 4: Asynchronous operation
qr.registerCommand(
    'Wait',
    { parameters: ['ms'] },
    function (params) {

```

```

        var ms = params.getInt('ms');

        return delay(ms)
            .then(function () {
                return ms;
            });
    }
};

// Example 5: Simple string concatenation
qr.registerCommand(
    'Concat2Strings',
    { parameters: ['String1', 'String2'] },
    function (params) {
        return params.get('String1') + params.get('String2');
    }
);

// Example 6: Command with no definition
qr.registerCommand(
    'NoDefinition',
    function () {
        return 'NoDefinition';
    }
);

// Example 7: Command with null definition
qr.registerCommand(
    'NullDefinition',
    null,
    function () {
        return 'NullDefinition';
    }
);

// Example 8: Command with empty definition
qr.registerCommand(
    'EmptyDefinition',
    {},
    function () {
        return 'EmptyDefinition';
    }
);
};

formsViewer.getInstances().forEach(addCommands);
})();

```

Example 2: URL Opening Command

```
(function () {
    "use strict";

    function registerCommand(instance) {
        var qr = instance.qRules();

        qr.registerCommand(
            'OpenUrl',
            { parameters: ['url', 'newtab'] },
            function (params) {
                var url = params.get('url');
                var newTab = params.getBool('newtab');

                if (newTab) {
                    window.open(url);
                } else {
                    window.location = url;
                }

                return { success: true };
            }
        );
    }

    formsViewer
        .getInstances()
        .forEach(registerCommand);
})();
```

Key Features Demonstrated

The Example 1 and Example 2 code implementations demonstrate the following features:

1. **Parameter Types:**
 - String parameters (url, value, String1, String2)
 - Boolean parameters (reset, newtab)
 - Numeric parameters (base, exponent, ms)
2. **Error Handling:**
 - Throwing errors
 - Returning error objects
 - Parameter validation
3. **Return Types:**
 - Simple values
 - Success/error objects
 - Promises

- No return value
- 4. **Command Definitions:**
 - With parameters
 - With parameter descriptions
 - Without parameters
 - Null definition
 - Empty definition
- 5. **Browser Integration:**
 - DOM manipulation
 - Window control
 - Async operations