ServiceNow Application Developer

Server-side Scripting > Exercise: JavaScript Debugger

In this exercise you will practice using the JavaScript Debugger.

Create the Business Rule

1. Create a Business Rule.
   a. In Studio, click the **Create Application File** link.
   b. In the *Filter...* field enter the text **Business** OR select **Server Development** from the categories in the left hand pane.
   c. Select **Business Rule** in the middle pane as the file type, then select the **Create** button.

2. Configure the Business Rule:

   Name: **NeedIt Debugging Practice**
   Table: **NeedIt [x_<your_company_code>_needit_needit]**
   Active: **Selected (checked)**
   Advanced: **Selected (checked)**

3. Switch to the **When to run** section and continue configuring the Business Rule:

   When: **Before**
   Insert: **Selected (checked)**
   Update: **Selected (checked)**

4. Click the **Submit** button.

5. Switch to the **Advanced** section.

6. Copy this script and paste it into the *executeRule* function in the Script field. Do not overwrite the template; paste the script after the *Add your code here* comment.

```
current.short_description = "This text set by the Debugging Business Rules
business rule.";
var myNum = current.state;

// The function in this try/catch is not defined
try{
    thisFunctionDoesNotExist();
}
catch(err){
    gs.error("NeedIt App: a JavaScript runtime error occurred - " + err);
}

// This function is not defined and is not part of a try/catch
   thisFunctionAlsoDoesNotExist();

// getNum and setNum demonstrate JavaScript Closure
var x = 7;

function numFunc(){
    var x = 10;
    return{
        getNum: function(){
            return x;
        },
        setNum: function(newNum){
            x = newNum;
        }
    };
}
var callFunc = numFunc();
callFunc.getNum();
callFunc.setNum(2);
callFunc.getNum();
```

7. Click the **Update** button.

Testing

1. In Studio, open the JavaScript Debugger by clicking the **JavaScript Debugger** button () in the Script Editor toolbar.

> **NOTE**: The JavaScript Debugger opens in a new window. If nothing happens when you click the *JavaScript Debugger* button, check your browser's pop-up blocker settings.

2. Click in the gutter to set three breakpoints:

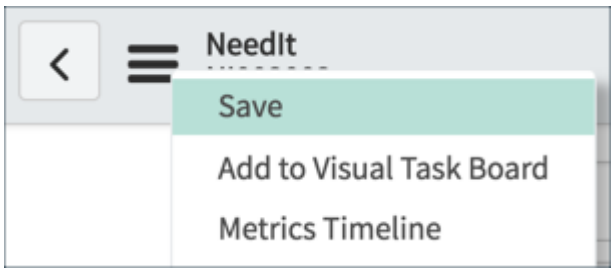
```
1  (function executeRule(current, previous /*null when
   async*/) {
2
3  current.short_description = "This text set by the
   Debugging Business Rules business rule.";
4  var myNum = current.state;
5
6
7  // The function in this try/catch is not defined
8  try{
9      thisFunctionDoesNotExist();
10 }
11 catch(err){
12     gs.error("NeedIt App: a JavaScript runtime error
   occurred - " + err);
13 }
14
15 // This function is not defined and is not part of a
   try/catch
16    thisFunctionAlsoDoesNotExist();
17
18 // getNum and setNum demonstrate JavaScript Closure
19 var x = 7;
20
```

3. Switch to the main ServiceNow browser window.

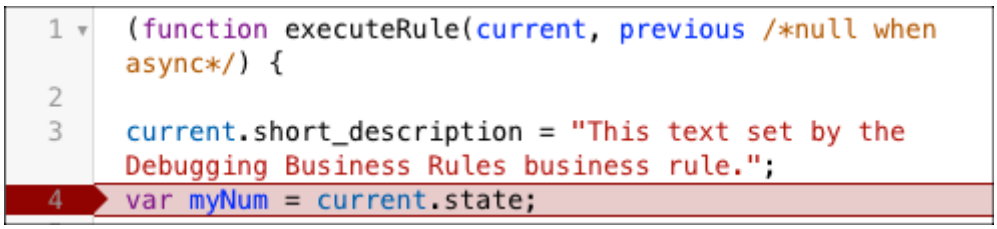
4. In the Application Navigator, open **NeedIt > Open**.

5. Open a record of your choice for editing. Notice the value in the *Short description* field.

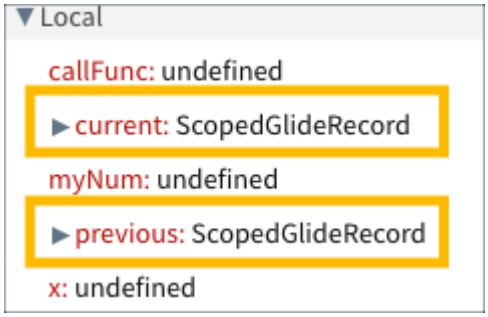
6. Make a change to any record field except *Short description*. Use the **Additional actions** menu (☰) to **Save** the record.



7. Return to the JavaScript Debugger window. The script execution should be stopped at the first breakpoint.



8. Examine the *Local* variables and their values. Open both the **current** and **previous** objects and look at the values of the *short_description* property.



   - **QUESTION**: Why are the property values different for the *previous* and *current* objects?

9. Examine the value for the *myNum* variable.

   - **QUESTION**: Does the *myNum* variable have a value? Should it?

10. Click the **Resume** button (▶) to move to the next breakpoint.

11. The script should be paused at the second breakpoint. Examine the *local* variables. The *myNum* variable should have a value.

12. Click the **Resume** button (▶) to move to the next breakpoint.

    - **QUESTION**: Did the JavaScript Debugger stop at the third breakpoint? Why not?

13. Force the Business Rule to execute again. When stopped at breakpoints, try the **Step over** ( ), **Step into** ( ), and **Step out of** ( ) buttons.

14. In the JavaScript Debugger, remove the breakpoints by clicking on the blue arrow for each breakpoint.

15. If you are not going to do the optional *Testing Closures* part of this exercise, make the *NeedIt Debugging Practice* Business Rule inactive.

Testing Closures (Advanced Topic - Optional)

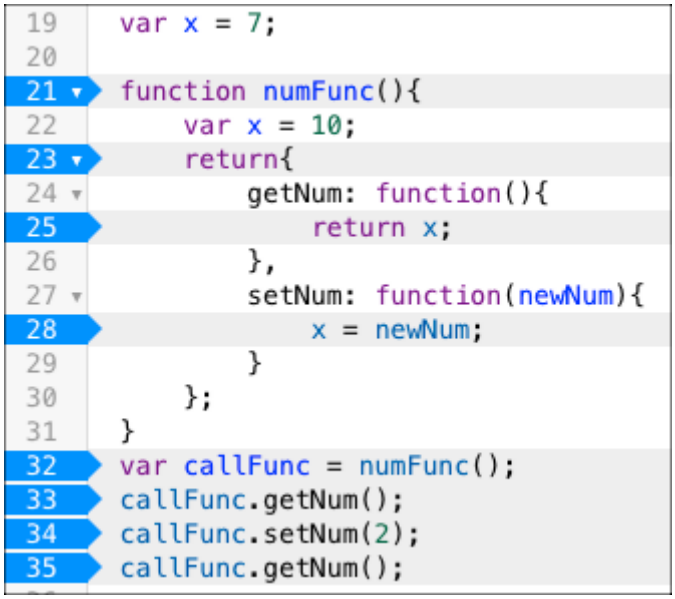
1. Return to Studio and comment out or delete these lines of the script:

```
// current.short_description = "This text set by the Debugging Business Rules business rule.";
// var myNum = current.state;


// // The function in this try/catch is not defined
// try{
//  thisFunctionDoesNotExist();
// }
// catch(err){
//  gs.error("NeedIt App: a JavaScript runtime error occurred - " + err);
// }

// // This function is not defined and is not part of a try/catch
//    thisFunctionAlsoDoesNotExist();
```

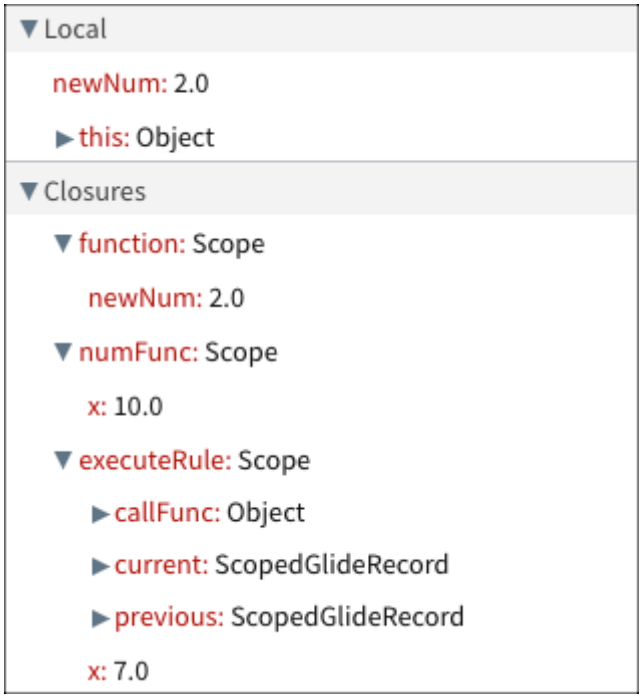
2. Set new breakpoints:



```
19  var x = 7;
20
21  function numFunc(){
22      var x = 10;
23      return{
24          getNum: function(){
25              return x;
26          },
27          setNum: function(newNum){
28              x = newNum;
29          }
30      };
31  }
32  var callFunc = numFunc();
33  callFunc.getNum();
34  callFunc.setNum(2);
35  callFunc.getNum();
```

3. Edit or create a *NeedIt* record to trigger the *NeedIt Debugging Practice* Business Rule.

4. Open the *Local* and *Closures* variables. Use the **Resume** button (▶) to move between breakpoints. Watch the value of *x* as you move through the breakpoints. Use your knowledge of JavaScript closures to explain why *x* can have different values at the same time.



5. Remove the breakpoints and make the *NeedIt Debugging Practice* Business Rule inactive.