In the RPG code, set the NegAmt indicator on when AMOUNT is less than zero.

You can use the concept of binding to hide and display elements dynamically (*visibility* property), dynamically change the position of an element (*left* and *top* properties), create dynamic images (*image source* property on an image widget), display custom tool tips (*tool tip* property), set focus on a field dynamically (*set focus* property), and much more.

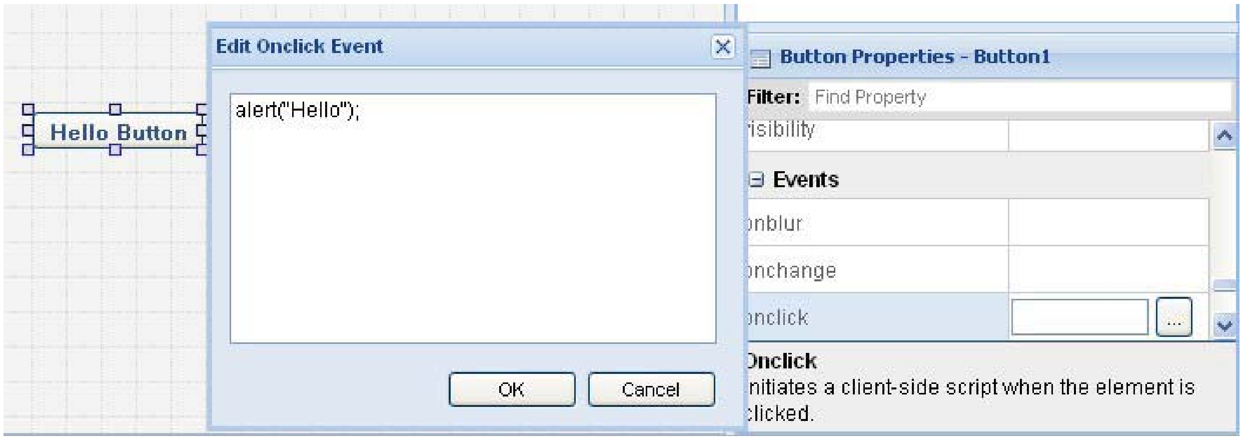
Using Events

Sample code:

Display file source:	PUISAMPLES/QDDSSRC(DROPD005D)
RPGLE source:	PUISAMPLES/QRPGLESRC(DROPD005R)

Widgets in Profound UI can also respond to events, such as *onclick* or *onchange*. This allows us to add even more dynamic functionality to the Web application using client-side JavaScript code. There are two ways to write code for JavaScript events -- inline or in an external JavaScript file. When using an external file, you will generally place it on the IFS somewhere in the /www/profoundui/htdocs/profoundui/userdata/ folder. You should then reference the external JavaScript file from /www/profoundui/htdocs/profoundui/userdata/html/start.html using a <script> tag.

If the JavaScript functionality is fairly simple, writing the code inline may be preferred. Simply prompt the event in the Properties Window, and type the code.



```
alert("Hello");
```

Onclick
nitiates a client-side script when the element is clicked.

Profound UI provides a number of useful JavaScript API to make coding easier. The two most commonly used API are described below:

- **pui.click()** – submits a response to the server. This API can accept an optional parameter containing an ID or an object reference to a button, or a similar element, that has a bound response property to be sent to the server. Sample usage:

  ```
  pui.click();  // send response to server
  ```

pui.click(“Exit”);   // send response to server by triggering the Exit button.

- **getObj()** – retrieves a reference to the DOM (browser’s Document Object Model) element given an element ID. The function can be used to retrieve and set the content or CSS style of an element. Sample usage:

  getObj(“CustBox”).value = “”; // clear the customer number box

  if (Number(getObj(“Amount”).innerHTML) < 0) getObj(“Amount”).style.color = “Red”; // if the content of the output field with the ID of Amount is less than 0, make the output field appear in red

Note: JavaScript and element IDs are case sensitive.

Note: In addition to attaching events to widgets, you may also attach events, such as *onload* and *onsubmit*, to the screen itself.