By David Wall

ITworld |
MAY 7, 2002

In JavaScript, pretty much everything you do with respect to your graphical user interface (GUI) has to do with detecting and reacting to events. When the user clicks a button, to cite a popular example, the click is represented as an event object of the click type and is passed to whatever event handler (if any) is assigned to react to events of that kind. Events bubble upwards through the Document Object Model (DOM) -- from, say, a button to its form to its document to its browser -- and event handlers may react to it at more than one level, if you want to set things up that way.

The newer versions of the DOM, as implemented in Microsoft Internet Explorer 6, allow you to use JavaScript to define and execute synthetic events. Synthetic events are event objects that are created ex nihilo by the JavaScript programmer, as opposed to "real" events that come into being as a result of an action within the browser window. Synthetic events can, for example, simulate a selection from a drop-down list box

followed by a click on a particular button, or a click on a particular region followed by a drag between two pairs of coordinates.

Synthetic events aren't helpful when creating auto-running demonstrations, since the events appear to take place for no reason (i.e., the mouse pointer doesn't move) and, unless you put in delay statements, the events occur in such quick succession that you can't tell what's happening. Rather, synthetic events are nice for creating test drivers that verify the behavior of interfaces. You can set up your application so that a single button click triggers a whole series of events, which will be run in an absolutely consistent way. This would enable you to adjust your application to behave as required in response to certain user behavior.

To create a synthetic event, you use the createEvent() method of the Document object.

```
var e = new Event;
e = document.createEvent("MouseEvents");
```

The createEvent() method takes any of several strings as arguments; "MouseEvents" describes a set of events associated with the mouse pointer. A DOM reference will give you a full roster of legal arguments.

When you've created an event, you need to initialize it before you turn

it loose. Here's how to do that:

```
e.initEvent("click", true, false);
```

The "click" argument is one that's significant to the "MouseEvents" set mentioned earlier -- again, consult that reference. The two Boolean values refer to whether the event bubbles up through the DOM and whether the event may be canceled with the preventDefault() method.

When you've configured your event, you can apply it to an element in the local DOM this way:

```
document.myForm.myButton.dispatchEvent(e);
```

That simulates a click on the DOM element called myButton.