You must also be careful regarding the order of the `document.createElement` and `setAttribute` statements with regard to where you add the newly created input element to its parent. In some browsers, you can add the newly created element only to its parent element after the element has been created and the `type` attribute has been correctly set. For instance, the following code snippet may behave unexpectedly in some browsers:

```javascript
document.getElementById("formElement").appendChild(button);
button.setAttribute("type", "button");
```

To avoid potentially erratic behavior, be sure to create the input element and set all its attributes, especially the `type` attribute, before adding it to the parent, like so:

```javascript
var button = document.createElement("input");
button.setAttribute("type", "button");
document.getElementById("formElement").appendChild(button);
```

Following this simple rule can help eliminate difficult-to-diagnose problems that may arise later.

### Adding Event Handlers to Input Elements

Adding an event handler to an input element should be as easy as using the `setAttribute` method and specifying the event handler name and the name of the desired function handler, right? Wrong. The standard way of setting an element’s event handler uses the element’s `setAttribute` method; it uses the event name as the attribute name and the function handler as the attribute value, as follows:

```javascript
var formElement = document.getElementById("formElement");
formElement.setAttribute("onclick", "doFoo()");
```

The previous code works in all modern browsers except Internet Explorer. To set an element’s event handler using JavaScript in Internet Explorer, you must reference the desired event handler via dot notation from the element and assign it to an anonymous function that calls the desired event handler, like so:

```javascript
var formElement = document.getElementById("formElement");
formElement.onclick = function() { doFoo(); };
```

Note how the `onclick` event handler is referenced via dot notation from `formElement`. The `onclick` event handler is assigned to an anonymous function that simply calls the desired event handler, in this case `doFoo`.

Fortunately, this technique is supported by Internet Explorer and apparently other modern browsers, so you have no reason to avoid setting a form element’s event handler dynamically via JavaScript.