Now setting the style on the element in question will function as expected in all modern browsers.

Setting an Element’s class Attribute

After reading the previous section about setting an element’s inline style via JavaScript, you’ve probably decided that simply setting the element’s class attribute must be the easiest way to go. Unfortunately, that is not correct. As with setting an element’s inline style, quirks also exist when setting an element’s class dynamically via JavaScript.

As you’ve probably guessed by now, Internet Explorer is the oddball amongst other modern browsers, although the workaround is rather simple. Browsers such as Firefox and Safari allow you to set an element’s class attribute using the element’s setAttribute method, like so:

```javascript
var element = document.getElementById("myElement");
element.setAttribute("class", "styleClass");
```

Oddly enough, Internet Explorer does not set the element’s class attribute when using the setAttribute method and class as the attribute name. Instead, Internet Explorer recognizes the className attribute when used in conjunction with the setAttribute method.

The complete workaround for this situation is to use both class and className as the attribute names when using the element’s setAttribute method, like so:

```javascript
var element = document.getElementById("myElement");
element.setAttribute("class", "styleClass");
element.setAttribute("className", "styleClass");
```

Most modern browsers will use the class attribute name and ignore className, and Internet Explorer will do the opposite.

Creating Input Elements

Input elements provide a way for a user to interact with the page. HTML natively has a limited set of input elements, including single-line text boxes, multiline text areas, select boxes, buttons, check boxes, and radio buttons. You’ll likely want to create some of these input elements dynamically using JavaScript as part of your Ajax implementations.

You create single-line text boxes, buttons, check boxes, and radio buttons all with an input element; only the value of the type attribute differs. Select boxes and text areas have their own unique tags. Creating input elements dynamically via JavaScript is straightforward (except for radio buttons, which are explained in the “Creating Radio Buttons” section), as long as you follow a few simple rules. You can easily create select boxes and text areas using the `document.createElement` method, passing to `document.createElement` the element’s tag name such as select or textarea.

Single-line text boxes, buttons, check boxes, and radio buttons are a little trickier because they all share the same element name of input and differ only in the value of the type attribute. So, to create these elements, you’ll need to use the `document.createElement` method followed by the element’s setAttribute method to set the value of the type attribute. This is not difficult, but it does require an extra line of code.