TOMETA SOFTWARE
www.tometasoftware.com

Ajax Tutorial: Drag & Drop

White Paper

Abstract
This Ajax tutorial explains how to easily add the popular drag and drop functionality to your web site.

Tometa creates custom software for you

Tometa Software designs and develops robust software solutions for virtually all industries including in-house (vertical market) and retail software, some of which is on the shelves at your local software store. We focus our unique combination of creative, technical, and problem-solving skills on meeting our client's objectives. Because of our clarity of purpose, commitment to process, and broad professional skill sets, we are able to provide our clients with world-class solutions that are functionally superior and fully aligned with our client's strategic focus.

Balancing development speed, quality and cost is what we are all about. Tometa combines agile development practices with fixed pricing, so you know what the cost, end product, and delivery time table look like–up front. If we underestimate the effort, we complete the overrun on our dime. Simple as that. That’s why large enterprise firms like Alcoa and NASDAQ choose Tometa.

Tometa’s agile development expertise and low-overhead US location keep our prices comparable to offshore vendors – without offshore challenges. Using a fixed pricing model, we provide upfront visibility into a project’s ultimate costs, end product and delivery schedule. Our clients like knowing that we have “skin in the game” – a fixed price that aligns our goals with yours, incenting us to get the job done right and fast.

Lastly, as a Microsoft Certified Gold Partner, Tometa Software, can customize its products or create custom web, client/server, and traditional applications. With programming experience in C#, C++, Visual Basic, CGI, HTML, RPG, Delphi, Java and many others; Tometa Software is uniquely positioned to meet your needs as a development firm.

Check us out today

If you're into Web Development, chances are you've used Ajax on a few occasions. For those of you who haven't heard about this exciting and relatively new technology, let me give you a quick low down.

Ajax is all the rage in Web application Development. It is such an exciting technology that it's already found use in Google's most popular applications, such as Gmail and Google Maps. What's it all about? To put it in simple terms: it's mainly a combination of JavaScript, HTML, XML and Document Object Model (DOM)… with a twist. You see, with conventional JavaScript when you fetch and send information to the server, you need to wait for a response before the user can do anything on the Webpage. Once the browser receives this information, the webpage will blink a few times and refresh.

Sounds tedious, awkward and slow? Well, fear not for Ajax takes it all away: no waiting for the server, no re-freshing and no re-loading. Ajax works "behind the scenes" doing all the "roundtrips" and interaction with the server. This enables the update of portions of the Web page without the need to reload or wait for a server response. This makes for a richer and faster browsing experience. To get a taste of Ajax, visit Google Maps and zoom in. Use your cursor to scroll up and down. See how everything happens almost instantly, with no waiting for the page to reload. That's the magic of Ajax.

Now that you're familiar with what Ajax can achieve, let us put this new technology to some use and learn a few titbits about the language – or shall I say combination of languages. This tutorial will walk you through writing a simple Ajax program to drag and drop a box effortlessly anywhere in the Window. I will be explaining every portion of the code and what it does as we progress through the tutorial. This will give you a chance to digest the different elements involved in the Ajax language.

Drag & Drop: Let the fun begin

First, let us start with some simple HTML code to better explain what we're set to achieve through the rest of the Ajax code that will follow.

```
<html >
<head>
            <title>Ajax Drag & Drop!</title>
        <style type="text/css">
</head>

<div id="dragbox"
            style="position:absolute; left:200px; top:200px; filter:
alpha(opacity=100); opacity:1;"
            onmousedown="dragPress(event);">Hey! Please Drag Me!</div>
```

```
<div id="dropbox"
           style="left:400px; top:400px;
                              width:100px; height:100px;">Drop Me
Here!</div>


</body>
</html>
```

Concentrate on the CSS inside the <div> tags for now. What we have here are two boxes. The first one, dragbox, is the box we want the user to drag around the window. The second box, dropbox, will act as a place to drop the dragbox.

Now to the piece of code that might be confusing you:

onmousedown="dragPress(event);"

dragPress(event) is an event handler that is called whenever a user clicks on our dragbox with either button. What's an event handler? It's simply a special attribute that associates an object – in our case the dragbox – with an event – mouse click -. Pretty simple, huh?

How do we implement the event handler? What attributes and actions do we want to associate with this event?

We will change the opacity of the dragbox when the event takes place. This will make it easier to distinguish our dragbox between an idle state and the occurrence of an event. Furthermore, we want to implement further event handlers for moving and releasing the dragbox.

Changing the Opacity

As mentioned previously, we want to change the opacity of the dragbox to mark the start of the event. This is implemented in the listing that follows:

```
function setOpacity(node,val) {
          if (node.filters) {
              try {
                      node.filters["alpha"].opacity = val*100;
                    } catch (e) { }
                    } else if (node.style.opacity) {
                      node.style.opacity = val;
                                                              }
          }
```

Within the Document Object Model (DOM), the HTML document is viewed as a tree structure. Every element that contains data is a node. For example the attribute <div> is a node. The listing above is now self-explanatory: we set the opacity of the dragbox through *node.filters* or *node.style.opacity* and then assign a user-defined value - *val* - to it.

Adding and removing Event Listeners

We want to add new event listeners when moving the dragbox. We also want to remove even listeners once the user releases the dragbox.

```
Evt.addEventListener = function (target,type,func,bubbles) {
                              if (document.addEventListener) {

          target.addEventListener(type,func,bubbles);
                              } else if (document.attachEvent) {
                                        target.attachEvent("on"+type,func,bubbles);
                              } else {
                                        target["on"+type] = func;
                              }
                    };

                    Evt.removeEventListener = function
(target,type,func,bubbles) {
                              if (document.removeEventListener) {

          target.removeEventListener(type,func,bubbles);
                              } else if (document.detachEvent) {

          target.detachEvent("on"+type,func,bubbles);
                              } else {
                                        target["on"+type] = null;
                              }
                    };
```

The *addEventListener* has a different implementation across browsers. The listing above is no more than cross-scripting for Internet Explorer, Netscape and other browsers through *AddEventListener* and *attachEvent* respectively. If we come across a browser that doesn’t support *addEventListner*, we simply set the type of event to null.
The exact same code logic applies to *removeEventListener*.

Getting the dragbox position

Accessing HTML tags is a straightforward exercise in Ajax. A node object denotes HTML tags in a given document. All you have to do is specify which element you want to access. It is as simple as that!

The listing that follows accesses the coordinates of the dragbox anywhere in the Window. This will come in handy for communicating with the server later on.

```
function getX(node) {
```

```
                    return parseInt(node.style.left);
            }

function getY(node) {
                    return parseInt(node.style.top);
            }

function getWidth(node) {
                    return parseInt(node.style.width);
            }

function getHeight(node) {
                    return parseInt(node.style.height);
            }
```

Setting the dragbox position

When the user drags and releases the box around, we want the position of the box to change accordingly. To do that, we have to set the left and top positions of our dragbox inside the <div id=dragbox>. As we have seen, accessing the elements of an HTML tag is pretty simple. We follow the same method to dynamically set the position of the dragbox.

```
function setX(node,x) {
                    node.style.left = x + "px";
            }

function setY(node,y) {
                    node.style.top = y + "px";
            }
```

For the new position of our box, we need to get the position of the dragbox at the time of the *onmouseclick* event. This is implemented in the following listing:

```
var deltaX = evt.x - getX(dragbox);
var deltaY = evt.y - getY(dragbox);
```

Now, we're ready to set the dragbox positions as the user drags the box.

```
function dragMove(evt) {
evt = new Evt(evt);
setX(dragbox,evt.x - deltaX);
setY(dragbox,evt.y - deltaY);
evt.consume();
            }
```

The *Evt(evt)* and *consume()* functions are pretty simple. The *Evt(evt)* function checks the event propagated in the document and gets the coordinates of the window. These are variables for which we need to set the new position of the dragbox.

```
function Evt(evt) {
            this.evt = evt ? evt : window.event;
            this.source = evt.target ? evt.target : evt.srcElement;
            this.x = evt.pageX ? evt.pageX : evt.clientX;
            this.y = evt.pageY ? evt.pageY : evt.clientY;
        }
```

The *consume* function cancels the bubbling effect of the event. Remember the tree structure in DOM? When an event occurs, it propagates from the target element upwards towards its parent. *stopPropagation* or *cancelBubble* will stop the propagation of the event. Note again that the extended lines of code are no more than a try at cross-browser compatibility.

```
Evt.prototype.consume = function () {
            if (this.evt.stopPropagation) {
                this.evt.stopPropagation();
                this.evt.preventDefault();
            } else if (this.evt.cancelBubble) {
                this.evt.cancelBubble = true;
                this.evt.returnValue  = false;
            }
        };
```

Releasing the dragbox:

Once the user releases the dragbox, we want the opacity to change and the event handler to be removed. Since we've implemented the functions, it's only a matter of plugging in a few parameters.

```
function dragRelease(evt) {
            evt = new Evt(evt);
            setOpacity(dragbox,1);

    Evt.removeEventListener(document,"mousemove",dragMove,false);

    Evt.removeEventListener(document,"mouseup",dragRelease,false);
            if (droppedOnDropBox(evt)) {
                dragboxDropped(evt);
            }
        }
```

XMLHttpRequest

Ajax establishes a connection between the client and the server using the XMLHttpRequest object. A connection is implemented differently across the browser spectrum. The listing below will use alternative code to handle most browsers. If the browser doesn't support Ajax, the user will get an alert to let him know that his browser is not Ajax-enabled.

```
function createClient() {
                              try {
                                        client = window.XMLHttpRequest ? new
XMLHttpRequest() :

          new ActiveXObject("Microsoft.XMLHTTP");
                              } catch (e) {
                                        alert("Sorry, your browser is not AJAX-
enabled!");
                              }
                    }
```

Getting a Response from the Server

We want to get a message from the server when the user drags the box on the dropbox. We want the user to drop the box *above* the dropbox. The following listing will get the coordinates of the dragbox and return true if they are greater than the coordinates of the dropbox. Else, it will return false.

```
function droppedOnDropBox(evt) {
                              var dropbox =
document.getElementById("dropbox");
                              var x = getX(dropbox);
                              var y = getY(dropbox);
                              var width = getWidth(dropbox);
                              var height = getHeight(dropbox);
                              return evt.x > x &&
                                          evt.y > y &&
                                          evt.x < x + width &&
                                          evt.y < y + height;
                    }
```

The next function will open a connection between the client and the server. Doing so is very simple in Ajax. See the listing below:

```
function dragBoxDropped(evt) {
                              client.onreadystatechange = callback;
                              client.open("get","server.php",true);
```

```
                    client.send(null);
          }
```

The callback function will wait for a response from the server. If there is a response, it will be shown to the user. Otherwise, an error message will be displayed.

```
function dragBoxDropped(evt) {
                    client.onreadystatechange = callback;
                    client.open("get","server.php",true);
                    client.send(null);
          }
```

Putting it all together in the main function

Now, we can implement all the code in our main function. One the user clicks and drags the box, the opacity will be changed and the position of the dragbox will be adjusted. Ajax will take care of all the interaction with the server. This will pan out smoothly for the user, as he will drag & drop smoothly without noticing a thing!

```
function dragPress(evt) {
                    evt = new Evt(evt);
                    box = evt.source;
                    setOpacity(box,.7);
                    deltaX = evt.x - getX(box);
                    deltaY = evt.y - getY(box);

     Evt.addEventListener(document,"mousemove",dragMove,false);

     Evt.addEventListener(document,"mouseup",dragRelease,false);
          }
```

I hope you've enjoyed this tutorial. It's simple enough yet shows the power of Ajax in assuring a rich and smooth experience for the user. The language itself is very intuitive and easy to learn. Check the full listing below and toy with the code. You can add more boxes and display more information to the user. With just a few hours of practice under your belt, you will be loving the ease and excitement of coding with Ajax!

Appendix: Full Code Listing

```
<html>
<head>
          <title>Drag & Drop Ajax Tutorial!</title>
          <style type="text/css">

                    body {
                              font:10px Verdana,sans-serif;
                    }

                    #box {
                              z-index:100;
                              width:100px; height:100px;
                              border:1px solid silver;
                              background: #eee;
                              text-align:center;
                              color:#333;
                    }

                    #dropbox {
                              position:absolute;
                              border:1px solid red;
                              background:orange;
                              text-align:center;
                              color:#333;
                    }

          </style>
          <script type="text/javascript">

                    var dragbox;
                    var deltaX, deltaY;
                    var client;
                    createClient();


                    function createClient() {
                              try {
                                        client = window.XMLHttpRequest ? new
XMLHttpRequest() :

          new ActiveXObject("Microsoft.XMLHTTP");
                              } catch (e) {
                                        alert("Sorry, your browser is not Ajax-
enabled!");
```

```
          }
}

function setOpacity(node,val) {
          if (node.filters) {
                    try {
                              node.filters["alpha"].opacity =
val*100;
                    } catch (e) { }
          } else if (node.style.opacity) {
                    node.style.opacity = val;
          }
}

function getX(node) {
          return parseInt(node.style.left);
}

function getY(node) {
          return parseInt(node.style.top);
}

function getWidth(node) {
          return parseInt(node.style.width);
}

function getHeight(node) {
          return parseInt(node.style.height);
}

function setX(node,x) {
          node.style.left = x + "px";
}

function setY(node,y) {
          node.style.top = y + "px";
}

function Evt(evt) {
          this.evt = evt ? evt : window.event;
          this.source = evt.target ? evt.target : evt.srcElement;
          this.x = evt.pageX ? evt.pageX : evt.clientX;
          this.y = evt.pageY ? evt.pageY : evt.clientY;
}

Evt.prototype.toString = function () {
          return "Evt [ x = " + this.x + ", y = " + this.y + " ]";
};

Evt.prototype.consume = function () {
				if (this.evt.stopPropagation) {
					this.evt.stopPropagation();
					this.evt.preventDefault();
				} else if (this.evt.cancelBubble) {
					this.evt.cancelBubble = true;
					this.evt.returnValue  = false;
				}
			};

			Evt.addEventListener = function (target,type,func,bubbles) {
				if (document.addEventListener) {

		target.addEventListener(type,func,bubbles);
				} else if (document.attachEvent) {
					target.attachEvent("on"+type,func,bubbles);
				} else {
					target["on"+type] = func;
				}
			};

			Evt.removeEventListener = function
	(target,type,func,bubbles) {
				if (document.removeEventListener) {

		target.removeEventListener(type,func,bubbles);
				} else if (document.detachEvent) {

		target.detachEvent("on"+type,func,bubbles);
				} else {
					target["on"+type] = null;
				}
			};

			function dragPress(evt) {
				evt = new Evt(evt);
				dragbox = evt.source;
				setOpacity(dragbox,.7);
				deltaX = evt.x - getX(dragbox);
				deltaY = evt.y - getY(dragbox);

		Evt.addEventListener(document,"mousemove",dragMove,false);

		Evt.addEventListener(document,"mouseup",dragRelease,false);
			}

			function dragMove(evt) {
				evt = new Evt(evt);
				setX(dragbox,evt.x - deltaX);
				setY(dragbox,evt.y - deltaY);
				evt.consume();
```

```
                    }

                    function dragRelease(evt) {
                              evt = new Evt(evt);
                              setOpacity(box,1);

          Evt.removeEventListener(document,"mousemove",dragMove,false);

          Evt.removeEventListener(document,"mouseup",dragRelease,false);
                              if (droppedOnDropBox(evt)) {
                                        dragBoxDropped(evt);
                              }
                    }

                    function droppedOnDropBox(evt) {
                              var dropbox = document.getElementById("dropbox
");
                              var x = getX(dropbox);
                              var y = getY(dropbox);
                              var width = getWidth(dropbox);
                              var height = getHeight(dropbox);
                              return evt.x > x &&
                                          evt.y > y &&
                                          evt.x < x + width &&
                                          evt.y < y + height;
                    }

                    function dragBoxDropped(evt) {
                              client.onreadystatechange = callback;
                              client.open("get","server.php",true);
                              client.send(null);
                    }

                    function callback() {
                              if (client.readyState == 4) {
                                        if (client.status == 200) {
                                                  alert(client.responseText);
                                                  createClient();
                                        } else {
                                                  alert("Sorry, there seems to be a
problem retrieving the response:\n" +
                                                            client.statusText);
                                                  createClient();
                                        }
                              }
                    }

          </script>
</head>
<body onload="windowLoaded(event);">
```

```
<div id="dragbox"
            style="position:absolute; left:200px; top:200px; filter:
alpha(opacity=100); opacity:1;"
            onmousedown="dragPress(event);">Hey! Drag Me Please!</div>

<div id="dropbox"
            style="left:400px; top:400px;
                                    width:100px; height:100px;">Drop Me Here,!</div>


</body>
</html>
```