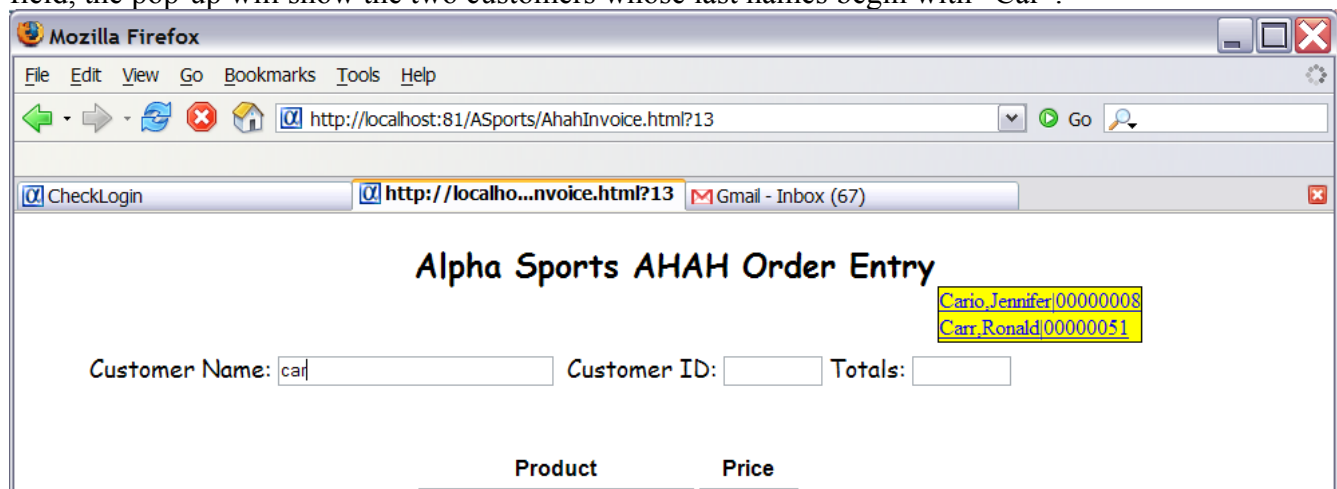


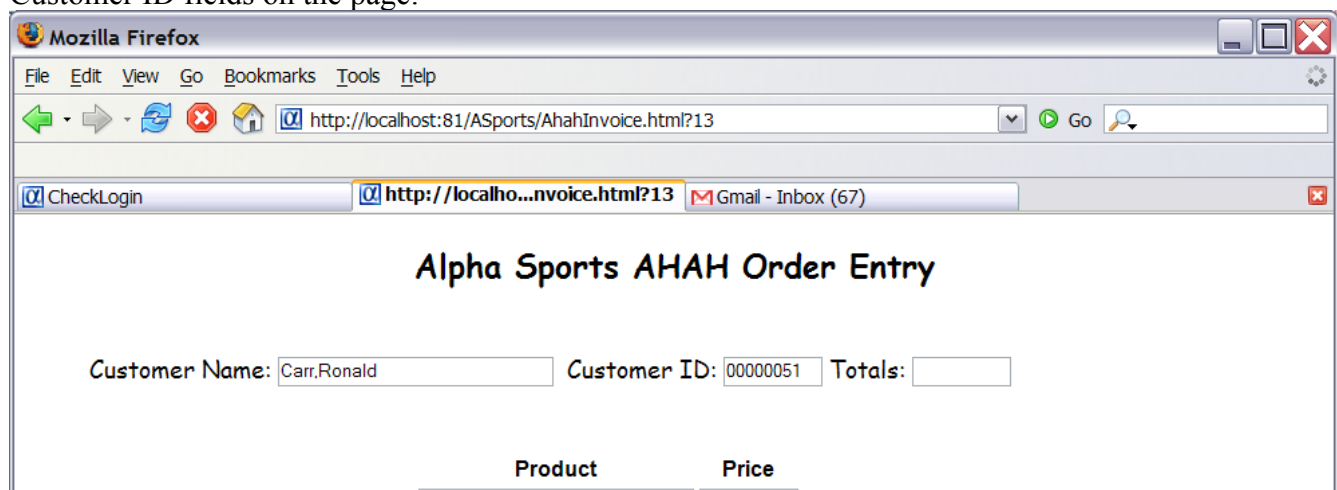
Using Javascript, CSS, and AHAH to Make Your Web Pages More Interactive

by Dr. Peter Wayne

The standard Web server model is that the user requests a page, the page is sent to the user, the user makes a change, sends the information back to the server, and the server then delivers a new page to the user. Each round-trip is time-consuming and lacks the immediate feedback of a desktop application. I'm going to show how you can use CSS, Javascript, and a technology called "Asynchronous HTML and HTTP" or AHAH to make your pages more responsive. Our example will use a stripped-down invoice form for the ubiquitous Alpha Sports database; I'll show you how to provide immediate pop-up help as the user types. For example, if you type "car" in the **customer name** field, the pop-up will show the two customers whose last names begin with "Car":

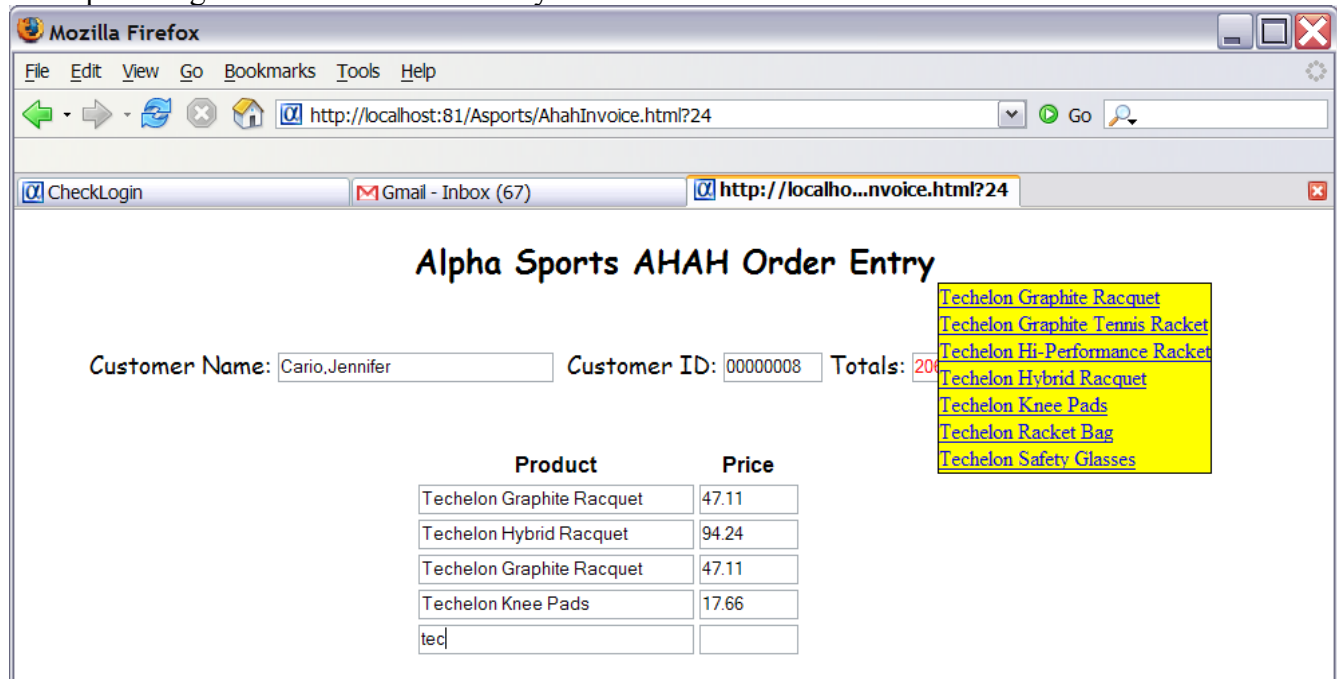


All you need do is click on the second name in the pop-up list to fill in the Customer Name and Customer ID fields on the page:



Similarly, if you start to type "tec" in the Product field, you'll get a list of all the products whose

description begin with “tec” from which you can make a selection:



Your Basic HTML

I'm going to start with a typical HTML page to display an order entry form. For simplicity, in this form I'll only show the customer name and ID, the products ordered and their pricing. Here is the HTML:

```

<html>
<head>
<meta name="generator" content="Alpha Five HTML Editor Version 8 Build 1173-3034">
<title></title>
</head>
<body>

<form method="post" action="doInvoice.a5w">
<h2>Alpha Sports AHAH Order Entry</h2>
<div class="top">
Customer Name:
<input id="custName" name="custName" size="30" maxlength="30" >&nbsp;
Customer ID:
<input id="custId" name="custId" size="8" maxlength="8">
Totals: <input id="totals" size="8" maxlength="8" >
</div>

<div class="middle">
<table id="items">

<thead><tr><th>Product</th><th>Price</th></tr></thead>
<tr id="row1">
<td><input id="prod1" name="prod1" size="30"></td>

```

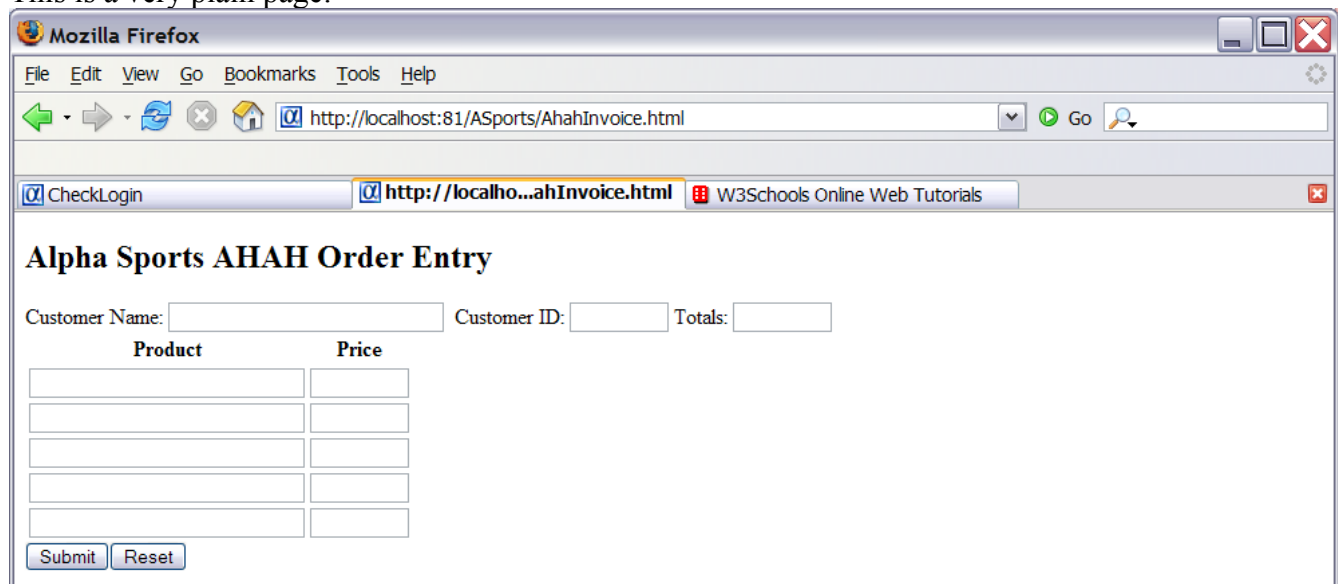
```

<td><input id="price1" name="price1" size="8"></td>
</tr>
<tr id="row2">
<td><input id="prod2" name="prod2" size="30"></td>
<td><input id="price2" name="price2" size="8"></td>
</tr>
<tr id="row3">
<td><input id="prod3" name="prod3" size="30"></td>
<td><input id="price3" name="price3" size="8"></td>
</tr>
<tr id="row4">
<td><input id="prod4" name="prod4" size="30"></td>
<td><input id="price4" name="price4" size="8"></td>
</tr>
<tr id="row5">
<td><input id="prod5" name="prod5" size="30"></td>
<td><input id="price5" name="price5" size="8"></td>
</tr>

</table>
</div>
<div id="hintField" class="hint"></div>
<div class="bottom">
<input type="submit" value="Submit"><input type="reset" value="Reset">
</div>
</form>
</body></html>

```

This is a very plain page:



Add some style

We'll add styling to the page to improve its appearance. The preferred way to style the page is with an

externally linked stylesheet. In the basic HTML I created several divisions (**div**) to which we can attach styles; that lets us change the style at will while leaving the basic layout unchanged. Here's the stylesheet:

```
h2 {
font-family: Comic Sans MS;
text-align: center;
}

div.top{
font-family: Comic Sans MS;
font-size: 18px;
font-style: normal;
font-weight: normal;
margin-left: 5%;
margin-top: 5%;
}

th{
font-family: Arial;
font-size: 16px;
font-style: normal;
font-weight: bold;
max-width: 20%;
}

div.middle{
margin-left: 30%;
margin-top: 5%;
}

div.total{
font-family: Comic Sans MS;
margin-left: 5%;
font-size: 16px;
font-weight: normal;
}

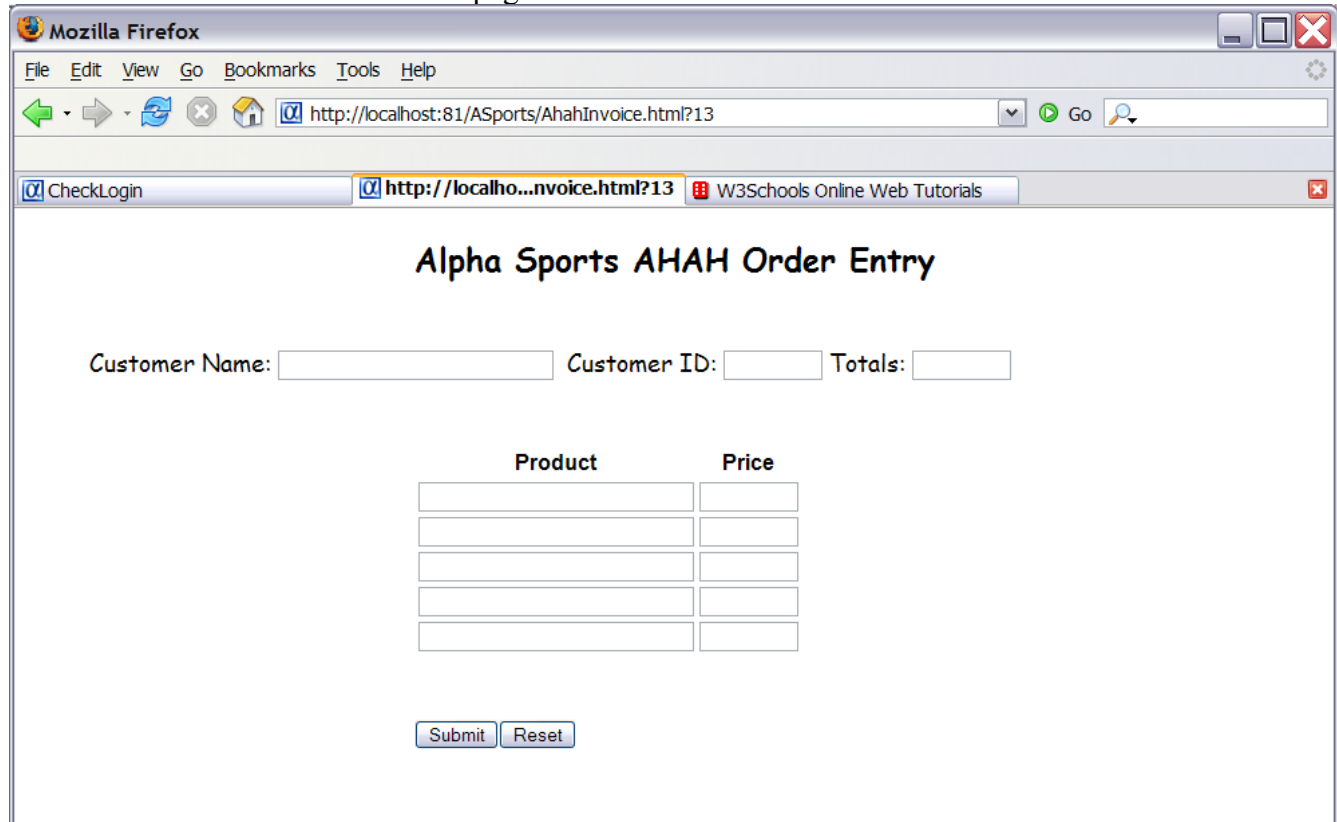
div.bottom{
font-family: Arial;
font-size: 18px;
margin-left: 30%;
margin-top: 5%;
}

div.hint{
position: absolute;
left: 70%;
top: 10%;
z-index: -1;
visibility: hidden;
border: 1px solid black;
background-color: yellow;
}

input#totals{
```

```
color: red;
}
```

And here's what results when the stylesheet is applied, by adding the line `<link rel="stylesheet" type="text/css" href="Sports.css">` to the `<head>` section of the HTML page:



Notice that the section of the page delimited by `<div id="hintField" class="hint"></div>` is invisible. We'll use this `div` section to display the "hint list" as the user types.

The communication with the server is done through Javascript. I'll present *how* communication with the server occurs first, and once we have that under our belt, we'll discuss *what* is sent to the server.

AHAH – the How

Asynchronous communication with the server is done in one of two ways: either through an XMLHttpRequest, for standards-compliant browsers (Firefox, Mozilla, Netscape, Safari, and others), or through an ActiveXObject, for Internet Explorer. In either case our Javascript will send a message to the server and then listen for a response, taking action when the response is received. I copied, um, adapted, freely from several web sites, the most helpful being www.crackajax.net and binnyva.blogspot.com.

The AHAH-specific functions are below:

```
function ahah(url) {
document.getElementById(hint).innerHTML="Fetching data...";
```

```

if (window.XMLHttpRequest) {
  req=new XMLHttpRequest();
} else if (window.ActiveXObject) {
  req=new ActiveXObject("Microsoft.XMLHTTP");
}
if (req !== undefined) {
  req.onreadystatechange = function() {ahahDone(url)};
  req.open("GET",url, true);
  req.send("");
}
}

function ahahDone(url){
if (req.readyState===4) {
  if (req.status===200) {
    document.getElementById(hint).innerHTML = req.responseText;
  } else {
    document.getElementById(hint).innerHTML = 'AHAH Error:\n' + req.statusText;
  }
}
}
}

```

The first Javascript function, **ahah()**, initiates an XML request object, either in the standards-compliant way or the Microsoft way. The XML request object has events and properties, and one of its events is the “readystatechange” event. When the “readystate” of the XML request object changes – that is, when the server sends a message back to the browser – the function **ahahDone()** is called. There are a series of state changes that **ahahDone** monitors, but we're only interested in the last state of **4** which means that the process is complete. Then we check the **status** returned by the server. We're all familiar with the **404** status of “File not found” and the status of **500** of “Server error”; a status of **200** is the green light, A-OK, all clear status. When we get the status of 200 we then perform a trick of inserting what the server sent us directly into the **hintField** that we placed on the original web page. What could be simpler?

Stop a moment to absorb this. The client's browser sends a request to the server and initiates a background task to wait for an answer. In AHAH, the answer will come as a properly formatted snippet of HTML which the browser can just place on the page in the **div** we reserved for hints!

Calling the AHAH functions – the What

Now that we have some idea how to send data to the server without submitting an entire HTML form, how can we trap keystrokes and send them to the server to check for matches? We need to expand our Javascript to attach code to the **onkeyup** events in the relevant input fields.

The easiest way to attach a function to an event in Javascript is to specify the function in-line in the HTML, but this has the drawback of mixing code with layout; every time you change the code, you have to re-edit (and re-publish, in Alpha Five) the HTML page. It's much cleaner if you keep your basic layout (HTML), styling (CSS), and code (Javascript) files separate – you can tweak one without worrying about the others. In the Javascript code that we'll reference from this page, we insert these lines:

```
var hint;
```

```
function getCustomer(aField) {
var url="getCustomer.a5w?custNm=";
var custNm = aField.value;
if (custNm.length>2) { //require at least 3 letters before searching
document.getElementById(hint).style.zIndex= +1;
document.getElementById(hint).style.visibility= 'visible';
url += escape(custNm);
var myRandom = parseInt(Math.random()*99999999,10);
url += "&rand="+myRandom; // prevents browser from using its cache
ahah (url);
} else {
document.getElementById(hint).style.visibility= 'hidden';
document.getElementById(hint).style.zIndex = -1;
}
}

function init(){
hint = 'hintField';
document.getElementById('custName').onkeyup = function() {getCustomer(this);};
}

window.onload=init;
```

What does this script fragment do? Let's start at the end. The statement,

```
window.onload=init;
```

tells the browser to run the **init** function when the document (web page) finishes loading. Note the syntax is **window.onload=init** and not **window.onload=init()**; the former, correct syntax tells the browser to execute the **init** function when the document has loaded, whereas the latter syntax attaches the *result* of a function call to the **onload** event – something that generally makes no sense.

Moving up one function, our **init** function attaches the **getCustomer()** function to the **onkeyup** event of the **custName** field on the web page. The syntax takes a little getting used to, but there's nothing here that is totally foreign to an Xbasic programmer – it's no different than attaching a function to the **OnPush** event of a button on an Alpha Five form. Note that I've set up the function to reference the calling object (field) by including itself as a parameter to the function, using the alias of **this**.

Finally, we come to the top function in our little pyramid of functions, **getCustomer()**. This function collects the keystrokes typed in the **custName** field; when there are three or more, it passes them to our **ahah** function, along with the name of the server-side page that will receive the keystrokes and a random number to make sure the browser cache won't try to supply a result (I'm not sure this is necessary, but it can't hurt). For example, if the user types “rab” in the **custName** field, then the **getCustomer()** function will create a call to **ahah** of the form

```
ahah ("getCustomer.a5w?custNm=rab&rand=6765")
```

I know we've covered a lot of ground, but here's where the light should be dawning. We have one function that's checking keystrokes in the **custName** field; when three characters have been entered, the three characters are passed to the server's **getCustomer.a5w** page by our **ahah** function. The user never had to press a “submit” key, and indeed the user may be continuing to type.

The Server-Side Script

Clearly our next task is to write a server-side script to provide the hints back to the user. Our first rendition of **getCustomer.a5w** will only provide a list of possible matches:

```
<%a5
if eval_valid("custNm") then
    custList=table.external_record_content_get("[PathAlias.adb_path]\customer",\
"lastname-', '-firstname-'|'-Customer_Id-'  


There's nothing very exotic about this script. It receives the variable custNm from the user and looks for matches in the customer.dbf table in Alpha Sports.


```

We can do better, though, than just provide a list of possible matches – we can let the user click on a match and insert the customer name and customer ID into the appropriate fields on the web page. To do this, we gussy up our HTML by returning, not just a simple list, but a list of references to more Javascript:

```
<%a5
if eval_valid("custNm") then
    custList=table.external_record_content_get("[PathAlias.adb_path]\customer",\
"lastname-', '-firstname-'|'-Customer_Id", "lastname", \
"left (lastname, "+len (custNm) +")="+quote (custNm)
    custList=*for_each(entry, \
"<a href=\"javascript:insertCust('"+ word(entry,1,"|")+
"', '"+word(entry,2,"|")+\"'\>"+entry+"</a><br />", \
custList)
    ? custList
end if
%>
```

There are four bolded lines above, which are just one long expression. The two middle bolded lines should both be on the same line in your script; they had to be broken up to fit on a printed page.

Our revised **getCustomer.a5w** script returns a list of names and customer IDs contained within HTML links. Selecting a link, though, does not take us to a new URL but instead runs a (so far undefined) Javascript function, **insertCust()**, which will take the chosen customer information and insert it into the appropriate fields on the web page. The Javascript **insertCust()** function – which, like all Javascript functions, is interpreted by the client's browser – is not terribly exciting. It takes the values handed it, inserts the values where they belong, and then hides the whole hint box:

```
function insertCust (aCustNm, aCustID) {
document.getElementById ("custName").value=aCustNm;
document.getElementById ("custId").value=aCustID;
document.getElementById (hint).style.zIndex=-1;
document.getElementById (hint).style.visibility='hidden';
}
```

The full Monty

There are other Javascript functions and server-side scripts needed to make this little demo actually

work, and I won't hold anything back: I'll reproduce them here and you can cut and paste them as you see fit. You'll notice I haven't used any Webserver components – they're not necessary. You can, if you like, add Javascript and AHAH functions to your Webserver components, or you can do what I've done here: use basic HTML, external CSS and Javascript files, and server-side A5W scripts that are separate from the actual presentation.

Here is **Sports.js**, the file containing all the Javascript for this demo. In addition to hints for the products, I've included a Javascript function to compute a running total as new items are ordered:

```
// Sports.js
var hint;
var req;

function ahah(url) {
document.getElementById(hint).innerHTML="Fetching data...";
if (window.XMLHttpRequest) {
    req=new XMLHttpRequest();
} else if (window.ActiveXObject) {
    req=new ActiveXObject("Microsoft.XMLHTTP");
}
if (req !== undefined) {
    req.onreadystatechange = function() {ahahDone(url)};
    req.open("GET",url, true);
    req.send("");
}
}

function ahahDone(url){
if (req.readyState==4) {
    if (req.status==200) {
        document.getElementById(hint).innerHTML = req.responseText;
    } else {
        document.getElementById(hint).innerHTML = 'AHAH Error:\n' + req.statusText;
    }
}
}

function insertCust(aCustNm,aCustID) {
document.getElementById("custName").value=aCustNm;
document.getElementById("custId").value=aCustID;
document.getElementById(hint).style.zIndex=-1;
document.getElementById(hint).style.visibility='hidden';
}

function insertProduct(thisRow, description, retail) {
document.getElementById(hint).style.zIndex=-1;
document.getElementById(hint).style.visibility= 'hidden';
document.getElementById('prod'+thisRow).value=description;
document.getElementById('price'+thisRow).value=Number(retail).toFixed(2); // 2
decimal positions
sumTotal();
}

function sumTotal() {
```

```
var tot=0;
for (var i=1;i<6;i++){
  var price=parseFloat(document.getElementById("price"+i).value);
  if (!isNaN(price)){
    tot+=price;
  }
}
document.getElementById("totals").value=tot.toFixed(2);
}

function getCustomer(aField){
var url="getCustomer.a5w?custNm=";
var custNm = aField.value;
if (custNm.length>2) { //require at least 3 letters before searching
  document.getElementById(hint).style.zIndex= +1;
  document.getElementById(hint).style.visibility= 'visible';

  url += escape(custNm);
  var myRandom = parseInt(Math.random()*99999999,10);
  url += "&rand="+myRandom; // prevents browser from using its cache
  ahah (url);
} else {
  document.getElementById(hint).style.visibility= 'hidden';
  document.getElementById(hint).style.zIndex = -1;
}
}

function getProduct(aField){
var url="getProduct.a5w?product=";
var prodNm = aField.value;
if (prodNm.length>2) {
  document.getElementById(hint).style.zIndex+=1;
  document.getElementById(hint).style.visibility= 'visible';

  url += escape(prodNm)+"&aField="+aField.id;
  var myRandom = parseInt(Math.random()*99999999,10);
  url += "&rand="+myRandom;
  ahah(url);
} else {
  document.getElementById(hint).style.visible= 'hidden';
  document.getElementById(hint).style.zIndex = -1;
}
}

function init(){
hint = 'hintField';
document.getElementById('custName').onkeyup = function() {getCustomer(this)};

var totals=document.getElementById("totals");
totals.onfocus=function() {blur()};

var amtfield;
for (var i=1;i<6;i++){
```

```

amtfield=document.getElementById("price"+i);
amtfield.onchange = function() {sumTotal();};
document.getElementById('prod'+i).onkeyup = function() {getProduct(this);};
}
}

window.onload=init;

```

And the only change to **AhahInvoice.html**, the main html file, is the addition of references to Javascript and css files in the header:

```

<html>
<head>
<meta name="generator" content="Alpha Five HTML Editor Version 8 Build 1173-3034">
<link rel="stylesheet" type="text/css" href="Sports.css">
<script language="javascript" src="Sports.js"></script>
<title></title>
</head>
<body>

<form method="post" action="doInvoice.a5w">
<h2>Alpha Sports AHAH Order Entry</h2>
<div class="top">
Customer Name:
<input id="custName" name="custName" size="30" maxlength="30" >&nbsp;   
Customer ID:
<input id="custId" name="custId" size="8" maxlength="8">
Totals: <input id="totals" size="8" maxlength="8" >
</div>

<div class="middle">
<table id="items">

<thead><tr><th>Product</th><th>Price</th></tr></thead>
<tr id="row1">
<td><input id="prod1" name="prod1" size="30"></td>
<td><input id="price1" name="price1" size="8"></td>
</tr>
<tr id="row2">
<td><input id="prod2" name="prod2" size="30"></td>
<td><input id="price2" name="price2" size="8"></td>
</tr>
<tr id="row3">
<td><input id="prod3" name="prod3" size="30"></td>
<td><input id="price3" name="price3" size="8"></td>
</tr>
<tr id="row4">
<td><input id="prod4" name="prod4" size="30"></td>
<td><input id="price4" name="price4" size="8"></td>
</tr>
<tr id="row5">
<td><input id="prod5" name="prod5" size="30"></td>
<td><input id="price5" name="price5" size="8"></td>
</tr>

```

```

</table>
</div>
<div id="hintField" class="hint"></div>
<div class="bottom">
<input type="submit" value="Submit"><input type="reset" value="Reset">
</div>
</form>
</body></html>

```

You've already seen **getCustomer.a5w** and **Sports.css**, so I won't reproduce them again, but here comes **getProduct.a5w**:

```

<%a5
if eval_valid("product") then
    prodList=table.external_record_content_get("[PathAlias.adb_path]\product",\
"trim(description)+'|'+retail","description","left(description,"+len(product)+")="+
quote(product))
    prodList=*for_each(entry,\
"<a name=\""+word(entry,1,"|")+\"\"
href=\"javascript:insertProduct('"+right(aField,1)+
"', '"+word(entry,1,"|")+\"', '"+word(entry,2,\"|")+\"')\">"+word(entry,1,\"|")+
\"</a><br/>\", \
prodList)
    ? prodList
end if
%>

```

Let me caution you again that everything in boldface above should appear on one line in your code editor.

I'll also include a **doInvoice.a5w** page. In a real application, this page would take some action on the submitted invoice. Here, it only parrots back the submitted data:

```

<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html><head>
<meta http-equiv=Content-type content="text/html; charset=unicode"><%a5
' doInvoice.a5w
if eval_valid("custId") then
html=<<%txt%
<table>
<tr><td colspan="2">You submitted these values:</td></tr>
<tr><td>Customer: %txt%+custName+</td><td>Customer ID: "+custId+
"</td></tr>"+crlf()

for i=1 to 5
html=html+\  

"<tr><td>"+eval("prod"+i)+"</td><td>"+eval("price"+i)+"</td></tr>"+crlf()
next
html=html+\  

"</table>"

? html
end if
%>

```

```
<meta content="MSHTML 6.00.5730.11" name=GENERATOR></head>  
<body></body></html>
```

I hope I've stimulated your appetite for including Javascript in your Alpha Five Webserver projects. If you want to see more Javascript examples, the best overall learning guide I have found is at w3schools.com. An extremely valuable resource for tracking down silly typos and undeclared variables in your Javascript is at jshint.com. And although I already mentioned the examples at www.crackajax.net and binnyva.blogspot.com I have to give thanks to them again for providing the basic framework of the AHAH functions used here.