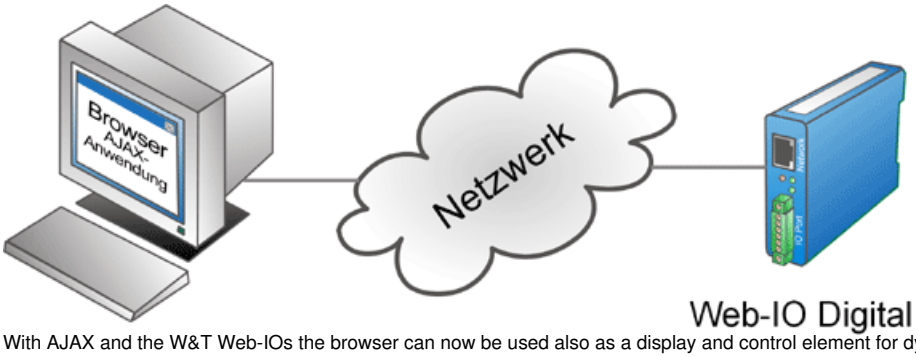
Application for the Web-IO Digital:

Web-IO Digital - Visualize in the browser using AJAX

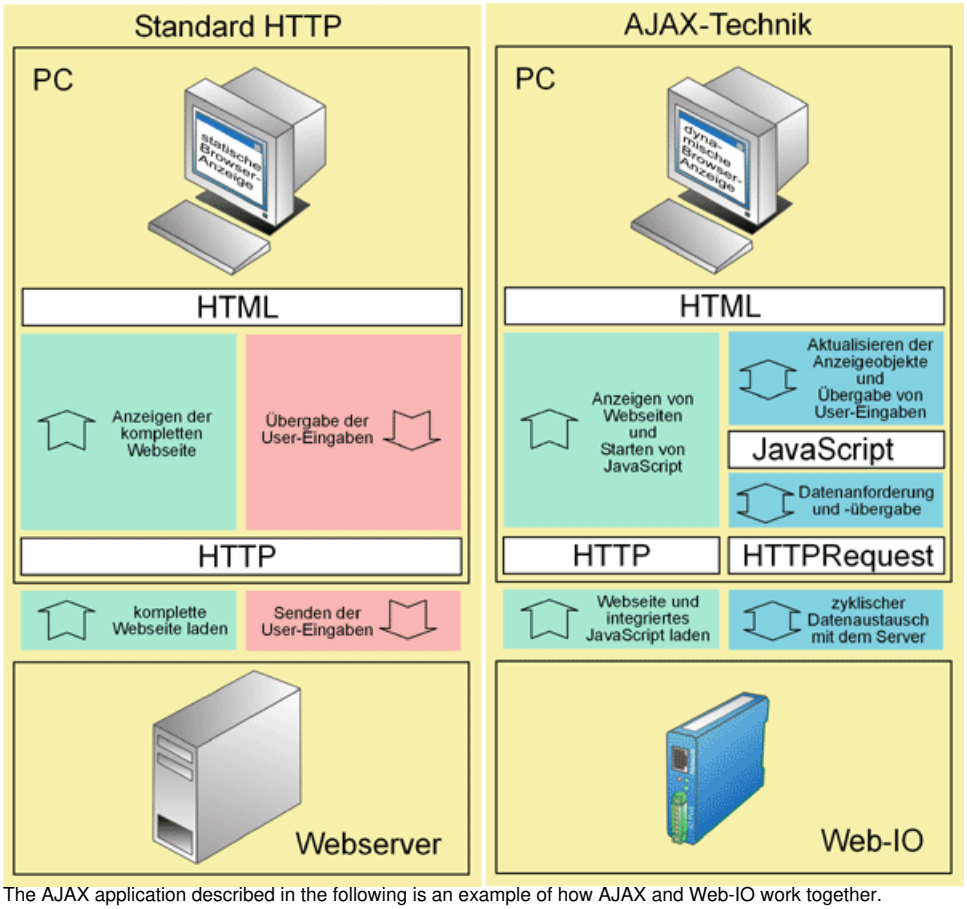
Product overview
Application overview

The internet browser is today a part of every modern operating system. Whether Internet Explorer, Firefox, Opera, Netscape or Safari - the browser is valued as a versatile display element when surfing the internet.



With AJAX and the W&T Web-IOs the browser can now be used also as a display and control element for dynamic, technical applications.

AJAX stands for Asynchronous JavaScript and XML, whereby the core functionality of AJAX is in being able to continue to communicate with the server after loading a Web site to the browser. Web pages which are constructed in standard HTML can only be refreshed by completely reloading them. AJAX-based JavaScripts on the other hand can exchange or modify individual display elements after the fact.

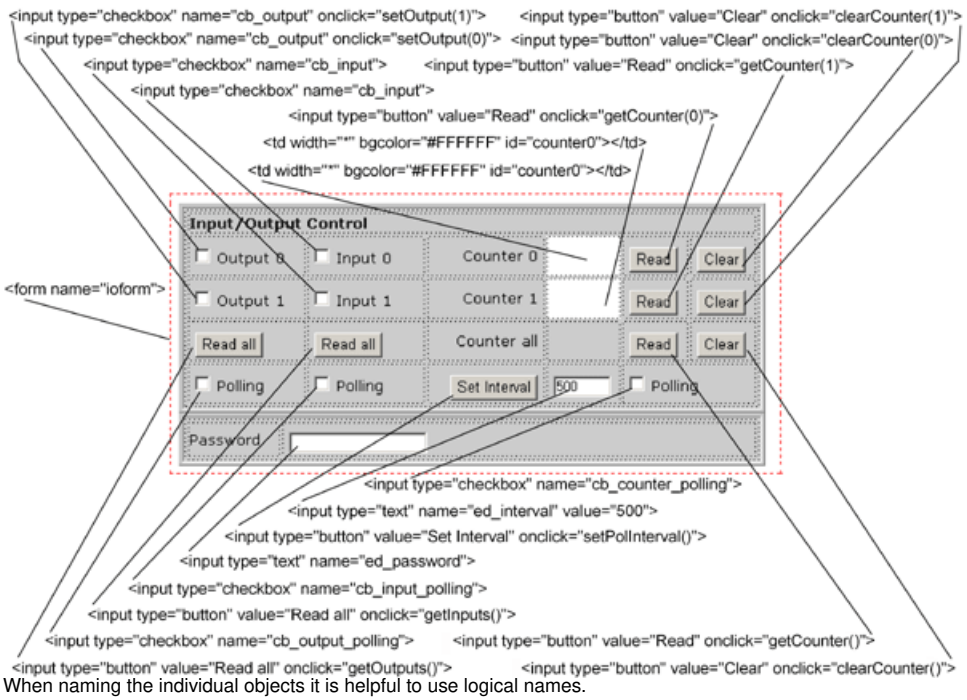


The AJAX application described in the following is an example of how AJAX and Web-IO work together.

Preparations

- with power,
- configured the inputs and outputs,
- connected it to your network,
- assigned it an IP address - which withWuTility is no problem.
- The Web-IO via Web-based management configured for AJAX operation (activate HTTP header)

Combining the various operating elements and display objects on the Web page



When naming the individual objects it is helpful to use logical names.

1. HTML underlying structure of the Web page Placing the operating and display elements
For better structuring the individual elements are placed in tables and the whole is declared as a form. The <user.htm> tag is not part of standard HTML syntax and is used by the Web-IO for identifying the Web page. The tag is filtered out before uploading to the browser.

```
<user.htm>
 <html>
  <head>
   <title>Web-IO AJAX-Client</title>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <STYLE type=text/css>
    TD {
    COLOR: #000000;
    FONT-FAMILY: Verdana,Arial,Helvetica;
    FONT-SIZE: 10pt;
    }
   </STYLE>
  </head>
  <body bgcolor="#FFFFFF" text="#000000" link="#000000">
   <form name="ioform">
    <table width="500" border="1" height="144" bgcolor="#CCCCCC">
     <tr>
      <td>
       <table width="500">
        <tr>
         <td colspan="6">
          <b>Input/Output Control &lt;/b>
         </td>
        </tr>
        <tr>
         <td width="100">
          <input type="checkbox" name="cb_output" onclick="setOutput(0)">Output 0
         </td>
         <td width="100">
          <input type="checkbox" name="cb_input">Input 0
         </td>
         <td width="100">
          <div align="right">Counter 0 </div>
         </td>
         <td width="*" bgcolor="#FFFFFF" id="counter0">
         </td>
         <td width="55">
          <input type="button" value="Read" onclick="getCounter(0)">
         </td>
         <td width="55">
          <input type="button" value="Clear" onclick="clearCounter(0)">
         </td>
        </tr>
        <tr>
```

```
    <td width="100">
     <input type="checkbox" name="cb_output" onclick="setOutput(1)"> Output 1
    </td>
    <td width="100">
     <input type="checkbox" name="cb_input"> Input 1
    </td>
    <td width="100">
     <div align="right">Counter 1 </div>
    </td>
    <td width="*" bgcolor="#FFFFFF" id="counter1">
    </td>
    <td width="55">
     <input type="button" value="Read" onclick="getCounter(1)">
    </td>
    <td width="55">
     <input type="button" value="Clear" onclick="clearCounter(1)">
    </td>
   </tr>
   <tr>
    <td width="100" height="29">
     <input type="button" value="Read all" onclick="getOutputs()">
    </td>
    <td width="100" height="29">
     <input type="button" value="Read all" onclick="getInputs()">
    </td>
    <td width="100" height="29">
     <div align="right">Counter all &lt;/div>
    </td>
    <td width="*" height="29">
    </td>
    <td width="55" height="29">
     <input type="button" value="Read" onclick="getCounter()">
    </td>
    <td width="55" height="29">
     <input type="button" value="Clear" onclick="clearCounter()">
    </td>
   </tr>
   <tr>
    <td width="100">
     <input type="checkbox" name="cb_output_polling">Polling
    </td>
    <td width="100">
     <input type="checkbox" name="cb_input_polling"> Polling
    </td>
    <td width="100">
     <div align="right">
     <input type="button" value="Set Interval" onclick="setPolInterval()">
     </div>
    </td>
    <td width="*" >
     <input type="text" name="ed_interval" value="500" maxlength="6" size="9">
    </td>
    <td colspan="2">
     <input type="checkbox" name="cb_counter_polling"> Polling
    </td>
   </tr>
  </table>
 </td>
</tr>
</table>
<table width="500" border="1" bgcolor="#CCCCCC" >
 <tr>
  <td height="35">
   <table width="500">
    <tr>
     <td width="78">Password</td>
     <td width="410">
      <input type="text" name="ed_password">
     </td>
    </tr>
   </table>
  </td>
 </tr>
</table>
```

```
  </form>
 ....
```

2. Global JavaScript declarations General variables and functions
Although the Web page is structured for the Web-IO 2xDigital Input, Web-IO 2xDigital Output, the JavaScripts are prepared for Web-IOs with more IOs. This is the purpose of the HexToInt function, which converts the hexadecimal strings into whole numbers.

The variable SendString is used later for sending data to the Web-IO.

```
...
var MAXIO=2;
var SendString;

function HexToInt(HexStr)
 {
 var TempVal;
 var HexVal=0;
 for( i=0; i<HexStr.length;i++)
  {
  if (HexStr.charCodeAt(i) > 57)
   {
   TempVal = HexStr.charCodeAt(i) - 55;
   }
  else
   {
   TempVal = HexStr.charCodeAt(i) - 48;
   }
  HexVal=HexVal+TempVal*Math.pow(16, HexStr.length-i-1);
  }
 return HexVal;
 }
```

3. Process user operation
The variable SendString is filled with a command which depends on which operating element the user has clicked on or modified.

Setting the outputs
The user sets the outputs by using the two check boxes cb_output. When the function is invoked the output no. is passed.

The DataRequest function which is then invoked is used for data exchange with the Web-IO and is will be described later in greater detail.

```
  function setOutput(OutputNr)
   {
   if (ioform.cb_output[OutputNr].checked==true)
    {
    SendString='outputaccess'+OutputNr+'?PW='+ioform.ed_password.value+'&State=ON&';
    }
   else
    {
    SendString='outputaccess'+OutputNr+'?PW='+ioform.ed_password.value+'&State=OFF&';
    }
   DataRequest(SendString);
   }
```

Querying output/input status
The user can request the status of the outputs and inputs by clicking on the corresponding button.

```
  function getOutputs()
   {
   DataRequest('output?PW='+ioform.ed_password.value+'&');
   }

  function getInputs()
   {
   DataRequest('Input?PW='+ioform.ed_password.value+'&');
   }
```

Read clear counters
Also the counter states of the input counters can be read or cleared. The parameter sent is the number of the counter you want to read or clear. If no parameter is sent, the Web-IO reads or clears all counters.

```
  function getCounter(CounterNr)
   {
   if (CounterNr==undefined)
    {
    DataRequest('counter?PW='+ioform.ed_password.value+'&');
    }
    else
    {
    DataRequest('counter'+CounterNr+'?PW='+ioform.ed_password.value+'&');
    }
   }

  function clearCounter(CounterNr)
   {
   if (CounterNr==undefined)
    {
    DataRequest('counterclear?PW='+ioform.ed_password.value+'&');
    }
   else
    {
    DataRequest('counterclear'+CounterNr+'?PW='+ioform.ed_password.value+'&');
    }
   }
```

4. Communication with the Web-IO

Data exchange with the Web-IO and refreshing of the Web page after it has already been loaded.

- The function shown here contains the essence of AJAX.
- The DataRequest function sends to the Web-IO the selected commands which are passed in the SendString in the background, invisible to the user. The integrated function DataReceived accepts the replies from the Web-IO.
- The replies from the Web-IO have a specific structure depending on the type.
- For the outputs: output;<binary value of the output status in hexadecimal format>
- For the inputs: input;<binary value of the input status in hexadecimal format>
- For the counters: counterx;<decimal counter state>
- or counter;<decimal counter state 0 >; <decimal counter state 0 >;.....if you want to read all counters at the same time.
- Depending on the reply received, the receive function branches accordingly and refreshes the display of the objects in the browser window.

```
function DataRequest(SendString)
 {
 var xmlHttp;
 try
  {// Internet Explorer
  if( window.ActiveXObject )
   {
   xmlHttp = new ActiveXObject("Microsoft.XMLHTTP" );
   }
  // Mozilla, Opera und Safari
  else if(window.XMLHttpRequest)
   {
   xmlHttp = new XMLHttpRequest();
   }
  }
 // loading of xmlhttp object failed
 catch( excNotLoadable )
  {
  xmlHttp = false;
  alert("no knowen browser");
  }
 if (xmlHttp)
  { xmlHttp.onreadystatechange = DataReceived;
  xmlHttp.open("GET", SendString, true);
  xmlHttp.setRequestHeader("Cache-Control", "no-store, no-cache, must-revalidate");
  xmlHttp.setRequestHeader("Expires", "Sat, 05 Nov 2005 00:00:00 GMT");
  xmlHttp.setRequestHeader("Pragma","no-cache");
  xmlHttp.send(null);
  }
function DataReceived()
 {
 var HexVal;
 var ReceiveStr;
 if (xmlHttp.readyState == 4)
  {
  if (xmlHttp.status == 200)
   {
   ReceiveStr = xmlHttp.responseText;
   //Input handling
   if (ReceiveStr.substring(0,5)=='input')
    {
    HexVal=HexToInt(ReceiveStr.substring(6,10));
    for (i=0;i<MAXIO;i++)
     {
     if ((HexVal & Math.pow(2,i)) == Math.pow(2,i))
      {
      ioform.cb_input[i].checked = true;
      }
     else
      {
      ioform.cb_input[i].checked = false;
      }
     }
    }
   //Output handling
   if (ReceiveStr.substring(0,6)=='output')
    {
    HexVal=HexToInt(ReceiveStr.substring(7,11));
    for (i=0;i<MAXIO;i++)
     {
     if ((HexVal & Math.pow(2,i)) == Math.pow(2,i))
      {
      ioform.cb_output[i].checked = true;
      }
      else
      {
      ioform.cb_output[i].checked = false;
      }
     }
    }
   //Counter handling
   if (ReceiveStr.substring(0,7)=='counter')
    {
    var slength=ReceiveStr.length;
    if (ReceiveStr.substring(7,8)==';')
     {
     countervalue=ReceiveStr.split(';');
     for (i=0;i<MAXIO;i++)
      {
      document.getElementById('counter'+i).innerHTML = '<a>'+countervalue[i+1]+'<\/a>';
      }
```

```
      }
     else
      {
      if (ReceiveStr.substring(9,10)==';')
       {
       i=(ReceiveStr.substring(7,9));
       document.getElementById('counter'+i).innerHTML = '<a>'+ReceiveStr.substring(10,slength)+'<\/a>';
       }
       else
       {
       i=(ReceiveStr.substring(7,8));
       document.getElementById('counter'+i).innerHTML = '<a>'+ReceiveStr.substring(9,slength)+'<\/a>';
       }
      }
     }
    }
   }
  }
```

Although the example shown is tailored to the Web-IO 2x Digital Input, 2x Digital Output, the DataRequest function is also capable of use for Web-IOs having more IOs.

5. Polling

Cyclical polling of particular values - In order to enable automatic refreshing of the display, the JavaScript Interval function is used. Depending on the check boxes for output, input and counter polling, the corresponding information is obtained from the Web-IO at a set interval.

```
var pollingtimer = window.setInterval("Polling()", 500);
function Polling()
 {
 if (ioform.cb_output_polling.checked==true)
  {
  DataRequest('output?PW='+ioform.ed_password.value+'&');
  }
 if (ioform.cb_input_polling.checked==true)
  {
  DataRequest('input?PW='+ioform.ed_password.value+'&');
  }
 if (ioform.cb_counter_polling.checked==true)
  {
  DataRequest('counter?PW='+ioform.ed_password.value+'&');
  }
 }
```

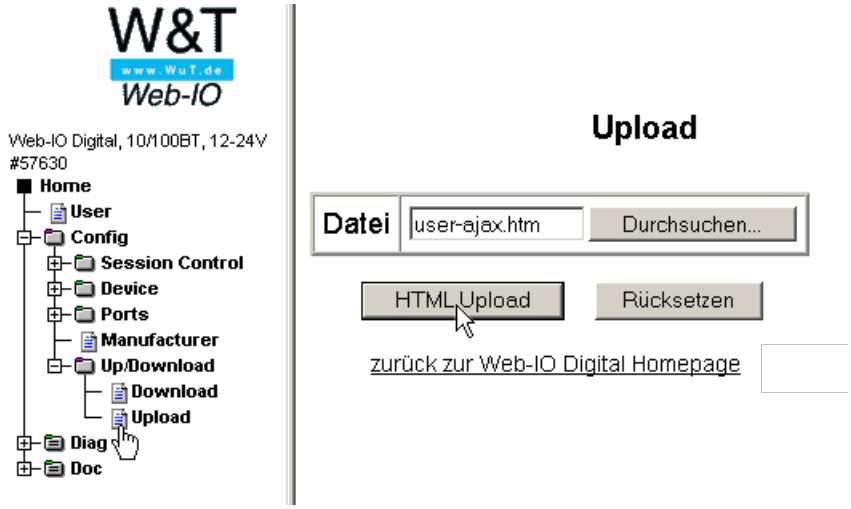
The desired interval can be entered in the corresponding text field and is adjusted by clicking on the Set Interval button.

```
function setPolInterval()
 {
 var intervaltime=parseInt(ioform.ed_interval.value)
 clearInterval(pollingtimer);
 pollingtimer = window.setInterval("Polling()", intervaltime);
 }
```

Loading the Web page into the Web-IO

AJAX-based Web pages only function if the data to be refreshed come from the same server as the Web page. It is not possible to load the Web page from Server A but retrieve the data from Server B.

When programming of the Web page is finished, it must be uploaded to the Web-IO. This is easy to do with just a few mouse clicks from the Web interface of the Web-IO:



The example supports all common functions of the Web-IO, optimized for theWeb-IO 2x Digital Input, 2x Digital Output PoE. For the other Web-IO models you may have to make adaptations to the example Web page. Additional program examples for socket programming can be found on the tool pages for the Web-IO. A detailed description for the socket interface of the Web-IO Digital models can be found in the reference manual.

Download program example

You don't have a Web-IO yet but would like to try the example out sometime?

No problem: We will be glad to send you the Web-IO Digital 2xInput, 2xOutput at no charge for 30 days. Simply fill out a sample ordering form, and we will ship the Web-IO for testing on an open invoice. If you return the unit within 30 days, we will simply mark the invoice as paid.

To sample orders

W&T
www.WuT.de

We are available to you in person:

Wiesemann & Theis
GmbH
Porschestr. 12
42279 Wuppertal
Phone: +49 202/2680-110 (Mon.-Fri. 8 a.m. to 5 p.m.)
Fax: +49 202/2680-265
info@wut.de

Data Privacy