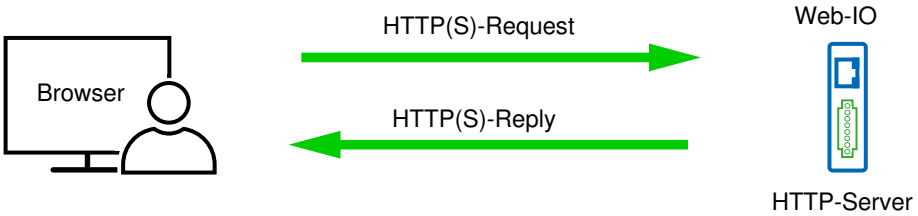
Tutorial zum Web-IO Digital:

Web-IO Digital aus eigener Webseite mit AJAX steuern

Die Web-IO Digital 4.0-Modelle bieten die Möglichkeit, eine eigene Webseite zu gestalten und ins Web-IO zu laden. So kann die Oberfläche zum Schalten und Überwachen nach eigenen Bedürfnissen aufgebaut werden.

Kernstück solcher dynamischer (selbst-aktualisierender) Webseiten ist die Technik, mit Hilfe von JavaScript über HTTP-Requests Informationen aus dem Web-IO abzurufen und die bereits geladene Webseite inhaltlich an das Prozessabbild des Web-IO anzupassen.

Diese Programmiertechnik bezeichnet man als *Asynchronous JavaScript and XML*, kurz *AJAX*.

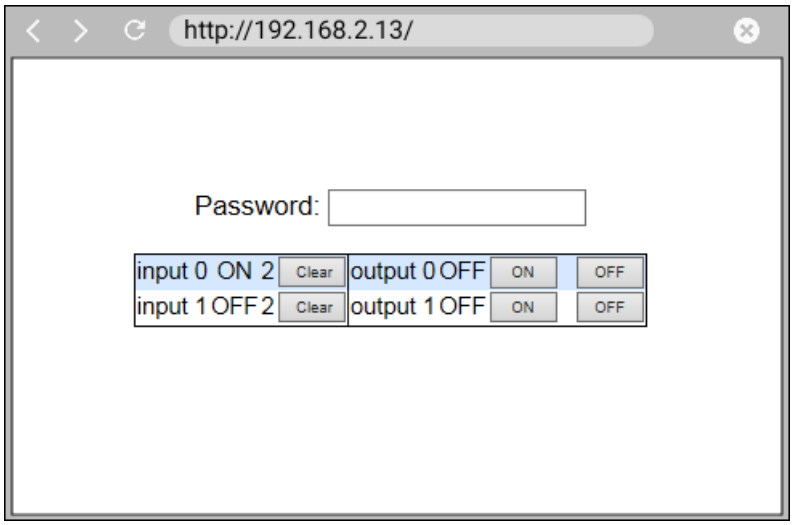


Das folgenden Beispiel zeigt den Aufbau einer sehr einfachen Webseite und die notwendigen Funktionen zur Anzeige der IO-Zustände für das Web-IO Digital 4.0 2xIn, 2xOut.

Vorbereitungen

- Web-IO mit Spannung versorgen und IOs verdrahten
- Web-IO mit dem Netzwerk verbinden
- IP-Adressen vergeben
- Beim Web-IO im Bereich *Kommunikationswege >> Web-API* den Punkt *HTTP-Request erlauben* aktivieren und die *Outputs zum Schalten freigeben*

Grundaufbau der eigenen Webseite



Das HTML-Grundgerüst dieser Seite sieht so aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
<title>Web-IO Digital, User</title>
<style type="text/css">
  * { font-family:arial; }
  table { font-size:14px; text-align:center; padding-left:5px; padding-right:5px;}
  .borderLeft { border-left:1px solid #000000; }
  .button { font-size:9px; width:40px; }
  .name { font-size:20px; font-weight:bold; text-align:center }
  .table { background-color:#d6e8ff; border-collapse:collapse; border:1px solid #000000; }
  .whiteBack { background-color:#ffffff; }
</style>
<script language="JavaScript" type="text/javascript">

 ........
 ........

</script>
</head>
<body onload="CommandLoop()">
  <form>
  <div align="center">
    <h2>Web-IO 0</h2>
    <span>Password: </span>
    <input id="pw" type="password" maxlength="31" size="20"">
  </div>
  </form>
  <table align="center" class="table">
  <tr>
    <td>input 0</td>
    <td id="input0">OFF</td>
    <td id="counter0">0</td>
    <td>
      <input class="button" onclick="clearCounter(0);" type="button" value="Clear">
    </td>
    <td class="borderLeft">output 0</td>
    <td id="output0">OFF</td>
    <td>
      <input class="button" onclick="setOutput(0,1);" type="button" value="ON">
      <input class="button" onclick="setOutput(0,0);" type="button" value="OFF">
    </td>
  </tr>
  <tr class="whiteBack">
    <td>input 1</td>
    <td id="input1">OFF</td>
    <td id="counter1">0</td>
    <td>
      <input class="button" onclick="clearCounter(1);" type="button" value="Clear">
    </td>
    <td class="borderLeft">output 1</td>
    <td id="output1">OFF</td>
    <td>
      <input class="button" onclick="setOutput(1,1);" type="button" value="ON">
      <input class="button" onclick="setOutput(1,0);" type="button" value="OFF">
    </td>
  </tr>
  </table>
</body>
</html>
```

Auf den *head*-Bereich und die Style-Informationen wollen wir hier nicht näher eingehen. Die Inhalte des *script*-Bereichs werden im weiteren Verlauf beschrieben. Wichtig sind im *body*-Bereich *id*-Benennungen, über die in den JavaScript-Funktionen die Objekte bzw. Tabellenzellen angesprochen werden. Den Buttons wird über *onclick* die aufzurufende Funktion zugeteilt. Als Parameter wird die Nummer der Counter bzw. der Outputs übergeben. Bei den Outputs gibt der zweite Parameter den Status (0=OFF, 1=ON) an.

Bei Einhaltung dieser Systematik kann das Beispiel ohne Änderung der JavaScript-Funktionen durch Hinzufügen weiterer Tabellenzeilen an weitere Web-IO-Modelle angepasst werden.

Globale Variablen und Funktionen im JavaScript

Zunächst müssen einige globale Variablen deklariert werden.

```
var MaxI = 2;
var MaxO = 2;
var ApplicationStep = 0;
var Interval = 500;

// Converting hexadecimal string to Integer
function HexToInt(HexStr)
{ var TempVal;
  var HexVal=0;
  for( var 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;
}
```

HTTP-Requests versenden und HTTP-Replys empfangen

```
// Sending command lines to Web-IO and receiving IO State
function DataRequest(SendString)
{ var xmlHttp;
  if( window.ActiveXObject )       // Internet Explorer
  { xmlHttp = new ActiveXObject( "Microsoft.XMLHTTP" );
  }
  else if(window.XMLHttpRequest )  // Mozilla, Opera und Safari
  { xmlHttp = new XMLHttpRequest();
  }
  if (xmlHttp)
  { xmlHttp.onreadystatechange = function()
    { if (xmlHttp.readyState == 4)
      { if (xmlHttp.status == 200)
        { if (xmlHttp.responseText.length > 0)
          { updateDisplay(xmlHttp.responseText);
          }
          xmlHttp=null;
        }
      }
    }
    xmlHttp.open("GET", SendString, true);
    xmlHttp.setRequestHeader("If-Modified-Since", "Thu, 1 Jan 1970 00:00:00 GMT");
    xmlHttp.send(null);
  }
}
```

Die Abwicklung von HTTP-Request und HTTP-Reply übernimmt die Funktion DataRequest. Leider nutzt der Internet Explorer für die Verarbeitung von HTTP-Requests andere Mechanismen als andere Browser. Deshalb prüft die Funktion zunächst, in welchem Browser die Webseite ausgeführt wird. Der als SendString übergebene HTTP-Request wird dann an den Server bzw. an das Web-IO gesendet, von dem auch die Webseite geladen wurde. Die Antwort, also der HTTP-Reply, wird dann an die Funktion UpdateDisplay übergeben.

Zyklische Abfrage von Inputs, Outputs und Countern

```
// Preparing command lines for cycle sending
function CommandLoop()
{ var CommandString ='';
  var IOPassword = document.getElementById('pw').value;
  ApplicationStep++;
  switch(ApplicationStep)
  { case 1:
      CommandString = 'input';
    break;
    case 2:
      CommandString = 'output';
    break;
    case 3:
      CommandString = 'counter';
      ApplicationStep = 0;
    break;
  }
  DataRequest(CommandString + '?PW=' + IOPassword + '&');
  maintimer = setTimeout("CommandLoop()", Interval);
}
```

Die Funktion CommandLoop wird über den Parameter *onload* im *body*-Tag gestartet, sobald die Webseite vollständig geladen ist. Die Funktion schickt über die Funktion DataRequest abwechselnd die HTTP-Requests für die Abfrage von Inputs, Outputs und Countern an das Web-IO und ruft sich mit zeitlicher Verzögerung wieder selbst auf. So werden kontinuierlich im festen Intervall die IO-Zustände aktualisiert.

Schalten der Outputs

```
// Set Output to ON (requested from User)
function setOutput(OutNo, OutVal)
{ var IOPassword = document.getElementById('pw').value;
  var CommandString = 'outputaccess'+OutNo+'?PW='+IOPassword+'&State=OFF&';
  if(OutVal>0)
  { CommandString = 'outputaccess'+OutNo+'?PW='+IOPassword+'&State=ON&';
  }
  DataRequest(CommandString);
}
```

Durch Klick auf die entsprechenden Buttons wird die Funktion setOutput aufgerufen und die Nummer des Outputs sowie der zu schaltende Zustand übergeben. Über die Funktion DataRequest wird der notwendige HTTP-Request an das Web-IO geschickt.

Löschen der Counter

```
// Set Counter to 0 (requested from display)
function clearCounter(CounterNo)
{ var IOPassword = document.getElementById('pw').value;
  DataRequest('counterclear'+CounterNo+'?PW='+IOPassword+'&');
}
```

Durch Klick auf die entsprechenden Buttons wird die Funktion clearCounter aufgerufen und die Nummer des Inputs übergeben.

Aktualisieren der Webseiteninhalte

```
// Dynamic update of the display depending on IO State
function updateDisplay(ReceiveStr)
{ var HexVal;
  var State;
  var ReceiveData = ReceiveStr.split(';')
  // Display Intput State
  if (ReceiveData[ReceiveData.length - 2].substring(0, 1) == 'i')
  { HexVal = HexToInt(ReceiveData[ReceiveData.length - 1]);
    for (var i = 0; i < MaxI; i++)
    { State = false;
      if ((HexVal & Math.pow(2, i)) == Math.pow(2, i))
      { State = true;
      }
      document.getElementById('input'+i).firstChild.data = ( !State ) ? 'OFF' : 'ON';
    }
  }
  // Display Output State
  if (ReceiveData[ReceiveData.length - 2].substring(0, 1) == 'o')
  { HexVal = HexToInt(ReceiveData[ReceiveData.length - 1]);
    for (var i = 0; i < MaxO; i++)
    { State = false;
      if ((HexVal & Math.pow(2, i)) == Math.pow(2, i))
      { State = true;
      }
      document.getElementById('output'+i).firstChild.data = ( !State ) ? 'OFF' : 'ON';
    }
  }
  //Display Counter
  if (ReceiveData.length - MaxI - 1 >= 0)
  { if (ReceiveData[ReceiveData.length - MaxI - 1].substring(0, 1) == 'c')
    { for (var i = 0; i < MaxI; i++)
      { document.getElementById('counter' + i).innerHTML = ReceiveData[ReceiveData.length - MaxI + i]
      }
    }
  }
  //Display cleared Counter
  if (ReceiveData[ReceiveData.length - 2].substring(0, 1) == 'c')
  { document.getElementById('counter'+ ReceiveData[ReceiveData.length - 2].substring(7,
ReceiveData[ReceiveData.length - 2].length)).innerHTML = ReceiveData[ReceiveData.length - 1];
  }
}
```

Als Antwort auf einen HTTP-Request sendet das Web-IO immer das eigentliche Kommando und getrennt mit Semikolon einen oder mehrere Parameter.

Für eine Input-Abfrage ist dies z.B. `input;1`.

Die Funktion `UpdateDisplay` zerlegt mittels `split` den übergebenen String in ein String-Array. Für das Input-Beispiel ergibt sich ein Array mit der ersten Feldvariablen `input` und der zweiten `1`. Über den ersten Buchstaben der ersten Feldvariablen wird festgestellt, ob die Antwort einen Input, Output oder die Counter betrifft.

Die zweite Feldvariable beinhaltet für Input bzw. Output das Bit-Muster für den Schaltzustand der IOs (Bit0=Input0, Bit1= Input1, ...). Da der Wert in hexadezimaler Schreibweise übergeben wird, muss er zunächst über die Funktion `HexToInt` in eine Dezimalzahl gewandelt werden.

Über eine UND-Verknüpfung des Input-Wertes mit der Zweierpotenz, die dem Bit-Wert der einzelnen Inputs bzw. Outputs entspricht, wird festgestellt, ob der betreffende Input (Output) gleich 0 oder 1, also OFF oder ON ist. Das erfolgt in einer Schleife, die entsprechend der Anzahl der IOs durchlaufen wird.

Mittels der JavaScript-Funktion `document.getElementById` wird das zu aktualisierende Anzeigeobjekt identifiziert und entsprechend dem aktuellen Schaltzustand angepasst.

Die gleichen Mechanismen werden für die Aktualisierung der Counter verwendet, wobei die Umrechnung hexadezimal in dezimal entfällt, da die Zählerstände der Counter dezimal übergeben werden.

Das Beispiel unterstützt die wichtigsten Funktionen des Web-IO, die über HTTP-Requests verfügbar sind, optimiert für das Web-IO Digital 4.0 2xIn, 2xOut. Für die anderen Web-IO-Modelle müssen ggf. Anpassung am Aufbau der Webseite und am JavaScript-Teil vorgenommen werden. Eine detaillierte Beschreibung zur Nutzung von HTTP-Requests für die Web-IO Digital-Modelle finden Sie in der Request-Kommandoübersicht oder im Programmierhandbuch zum Web-IO.

Programmbeispiel herunterladen

Produkte

Web-IO 4.0 Digital
2xIn, 2xOut

Bei Bedarf auch über PoE zu versorgen

Web-IO 4.0 Digital
12xIn, 12xOut

12x Eingänge,
12x Ausgänge

Weitere Web-IOs

Alle W&T Web-IO Digital 24V

Wir sind gerne persönlich für Sie da:

Wiesemann & Theis GmbH
Porschestr. 12
42279 Wuppertal
Tel.: 0202/2680-110 (Mo-Fr. 8-17 Uhr)
Fax: 0202/2680-265
info@wut.de