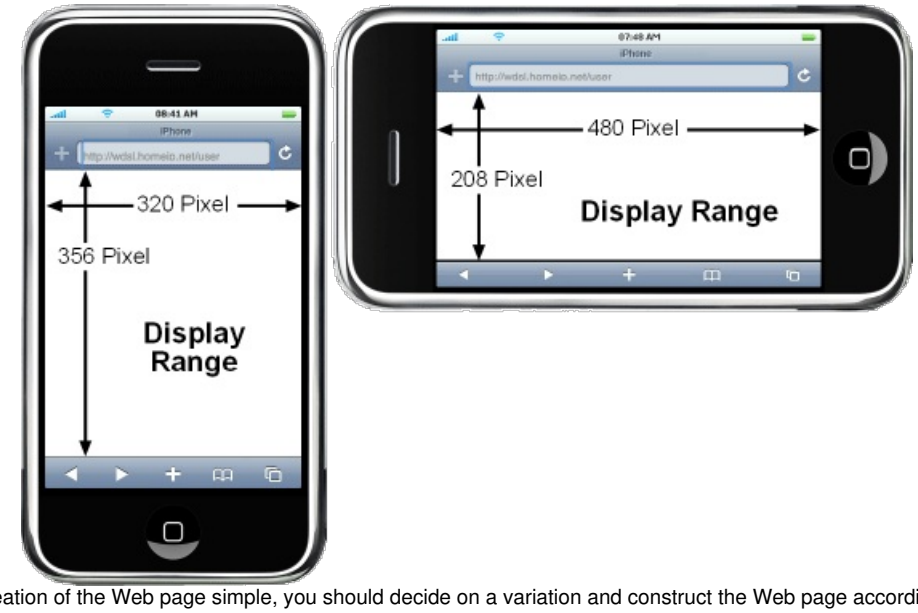
Application for the Web-IO Digital:

# iPhone optimized Web pages for Web-IO applications

The techniques shown here refer to the Apple iPhone, but they can be adapted to any Internet-capable cell phone by changing the display size to the corresponding display resolution. The only prerequisite is support of JavaScript and in particular HTTP Request Objects (AJAX technology).

As for the iPhone, there are two variations:

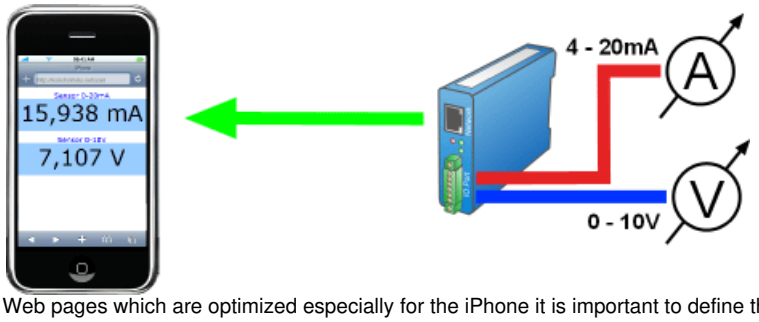



To keep creation of the Web page simple, you should decide on a variation and construct the Web page accordingly.

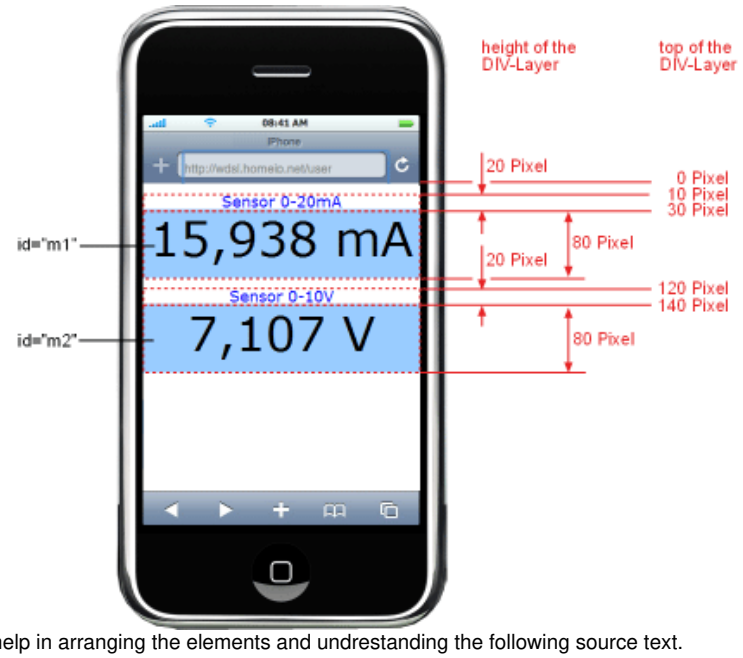
**If there is WLAN access, the example shown functions on all iPod Touch models!**

## Example: Display measurement values from industry and building technology with the iPhone

In this example we will measure currents or voltages using a Web-IO 2xAnalog IN and display them as a dynamic Web site on the iPhone. By connecting appropriate transducers virtually any physical property can be measured and dynamically displayed on the iPhone or cellular phone.




As already noted, for Web pages which are optimized especially for the iPhone it is important to define the HTML elements size and position to pixel accuracy.




The drawing should help in arranging the elements and undrestanding the following source text.

```
<user.htm>
<html>
 <head>
  <title>Aktuelles Klima</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  <meta name="viewport" content="width=320" />
  <style type="text/css">
   <!--
   * { font-family:Verdana, Helvetica; }
   .description{ position:absolute;
      text-align:center;
      font-size:18px;
      left:0px;
      height:20px;
      width:320px;
      }
   .value  { position:absolute;
      background-color:#99CCFF;
      text-align:center;
      font-size:55px;
      left:0px;
      height:80px;
      width:320px;
      }
   -->
  </style>
 </head>
```

The source text begins with the Web-IO specific tag <user.htm>, which the Web-IO needs for internally associating into the file system. The <user.htm> tag is not passed from the Web-IO to the browser.

Then follows the normal <head> area of the Web page. Important is the specifier " viewport". This limits the normal zoom-out of the Web page to 320 pixels, so that the Web page can always be displayed in full and in the desired scaling.

Further on in the source text DIV elements for description and display are created. Corresponding CSS-Style information is contained in the head for these elements. Among other things, height and left position are defined.

```
<body>
<div class="description" style="top:10px; ">
 <w&t_tags=sensor1>
</div>
<div id="m1" class="value" style="top:30px;">
 <w&t_tags=m1>
</div>
<div class="description" style="top:120px; ">
 <w&t_tags=sensor2>
</div>
<div id="m2" class="value" style="top:140px; ">
 <w&t_tags=m2>
</div>
</body>
</html>
```

Then follows the <body> area of the Web page. Here the CSS Classes and the top specifiers, are used to bring the individual DIV elements into position. For the DIV elements that still need to be changed after loading the Web page are assigned an ID. As the display text special W&T tags are entered. These tags are replaced by the currently valid value by the Web-IO when the page is loaded into the browser.

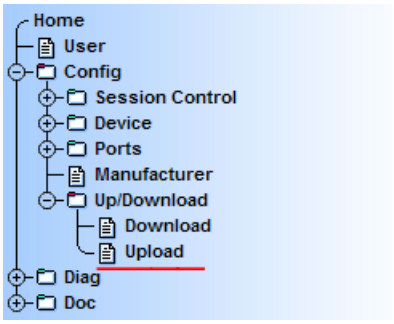
```
<script language="JavaScript" type="text/javascript">
function DataRequest(sendstring)
{
var xmlHttp = window.ActiveXObject ? new ActiveXObject("Microsoft.XMLHTTP") : xmlHttp = new XMLHttpRequest();
if (xmlHttp)
 {
 xmlHttp.onreadystatechange = function()
  {
  if (xmlHttp.readyState == 4)
   {
   if (xmlHttp.status == 200)
    {
    var ReceiveData = xmlHttp.responseText.split(";");
    document.getElementById('m1').innerHTML = ReceiveData[ReceiveData.length-2];
    document.getElementById('m2').innerHTML =ReceiveData[ReceiveData.length-1];
    xmlHttp=null;
    }
   }
  }
  xmlHttp.open("GET", sendstring, true);
  xmlHttp.setRequestHeader("Connection", "close");
  xmlHttp.setRequestHeader("If-Modified-Since", "Thu, 1 Jan 1970 00:00:00 GMT");
  xmlHttp.send(null);
  maintimer = setTimeout("DataRequest('single')", 5000);
 }
}
DataRequest('single');
</script>
```

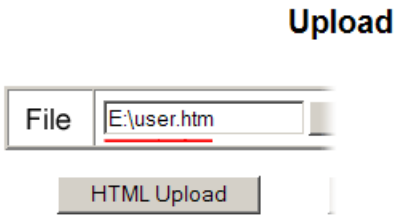
The core of the Web page is the JavaScript section, which is contained in the <head> area of the Web page. Here the DataRequest function is used to send the command *single* to the Web-IO. The Web-IO sends the values for temperature, relative humidity and barometric pressure semi-colon delineated. The split statement separates the values and writes them to a variable array.

In this method document.getElementById() the corresponding DIV elements are selected for display and the current values entered.

A timer causes this to happen cyclically every 5000ms.

The Web page is now finished and just needs to be uploaded to the Web-IO.




**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 Analog 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

Download program example

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