Vernier Coding Activities with JavaScript™

Go Direct® Sensors

VCJS-E

Vernier

Vernier Software & Technology • 888-VERNIER (888-837-6437)
info@vernier.com • vernier.com

Contents

Setting up a Simple Web Page (HTML)

INTRODUCTION

People use web pages to share information across the internet. You (along with an estimated 4.57 billion of your fellow human beings) have likely browsed the web to gather information, conduct research for an essay, or fact-check your science homework. The best web pages are easy to understand and navigate. But have you ever thought about what it takes to build a web page?

A typical web page uses a combination of three programming languages to create its look and feel:

- **HTML** sets up the landscape of a web page by defining the content and structure
- **CSS** modifies the presentation of the content by controlling colors, fonts, and general formatting (essentially a style guide).
- **JavaScript** enables the animation of web pages and creates the opportunity for user interaction.

Throughout this set of activities you will explore all three of these languages, one or two at a time. In this first activity you will explore how HTML is used to set the content of a web page.

PREPARATION

Each activity requires the following materials:

- computing device with internet access
- Visual Studio Code (download at **https://code.visualstudio.com/**)
- Vernier Go Direct Weather Sensor

PROCEDURE

Part I Making a Basic Web Page

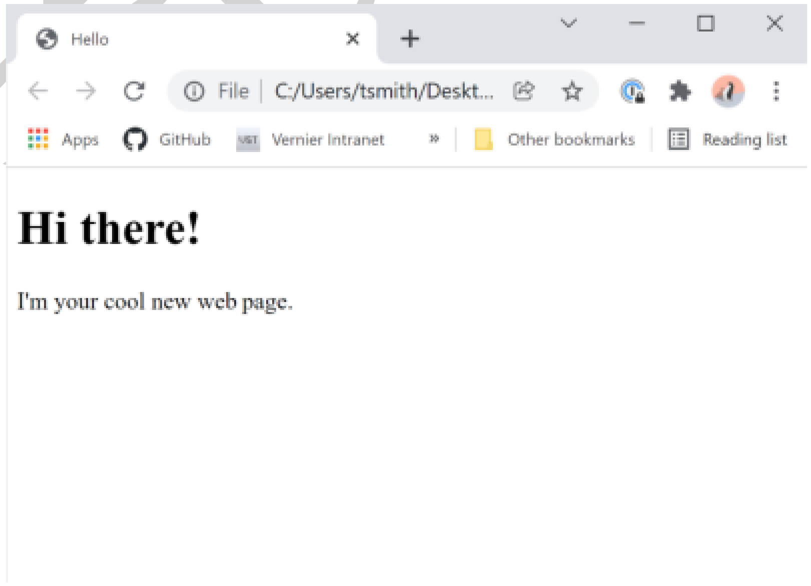
1. Launch Visual Studio Code and open a new file.

2. Enter the following code. If you have access to an electronic version of this document, you can copy/paste the code:

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="utf-8" />
      <meta http-equiv="X-UA-Compatible" content="IE=edge" />
      <meta name="viewport" content="width=device-width, initial-scale=1" />
      <title>Hello</title>
      <!--import the web page's stylesheet-->
      <link rel="stylesheet" href="/style.css" />
   </head>
   <body>
      <h1>Hi there!</h1>
      <p>
         I'm your cool new web page.
      </p>
   </body>
</html>
```

3. Save the file in a folder where you will store all of your code, giving appropriate names to the folder and file, such as "JavaScript Code" for the folder and "Activity 1 HTML.html" for the file.

4. Choose **Run Active File** from the **Terminal** menu. You should see a new web page open up as shown in Figure 1.



Hi there!

I'm your cool new web page.

Figure 1 A basic web page

5. On your web page, identify the three regions marked with A, B, and C in Figure 2. Note that your web page will not look exactly like Figure 2, but it will have all three elements.

   Look back at your code. Can you identify the code that creates each of these elements?

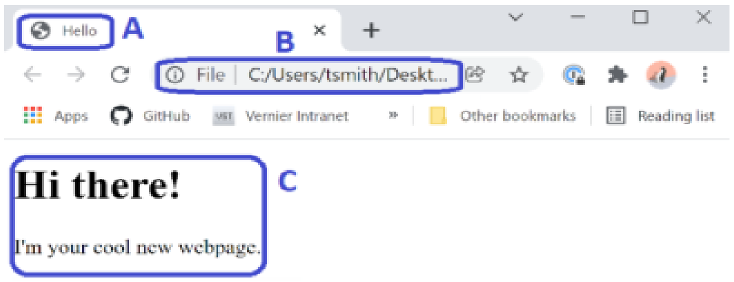


Figure 2 The anatomy of a web page

Part II Exploring HTML

6. Review the HTML code and use the following table to understand how the code functions.

HTML and HTML Tags

An "element" on a web page is anything that is displayed on the page (e.g., text, images, or videos). In HTML, an element is defined by start and end tags (< … > and </ … >); the tags include text that define how the element is displayed. For example

<h1>Introduction</h1>

<h1> indicates the start of an element, 'Introduction' is the text that will be displayed, and </h1> indicates the end of the element. The appearance of the text is defined by the CSS. All elements that use <h1>...</h1> will have the same appearance.

Here are a few of the elements that are used to get started. Take a moment to read through this list.

`<!DOCTYPE HTML>`	This is a declaration necessary for each HTML document and is found at the very beginning of the code.
`<html lang=en>...</html>`	Everything in an HTML document needs to be written between the html tags. In the example to the left, "lang=en" is defined in this tag. This specifies the language for the HTML code. In this case it is English.
`<head>...</head>`	Meta information about the HTML page is held between the head tags. This is where you can import libraries to be used in your code. Metadata is used by browsers (how to display content or reload page), search engines (keywords), and other web services.
`<title>...</title>`	The effect of a title depends on where it is used. When it is placed in the head tags it is used to specify the name of the web page that appears in the browser. In our code, this is "Hello" Refer to A in Figure 2. When the title tag is within the body, the text appears on the web page itself. Find "Hi there!" in the code and on the web page.

HTML and HTML Tags (continued)

`<link rel="stylesheet" href="/style.css" />`
 	
Not used, but will show up in later activities:
`<script src="/script.js" defer></script>`
	These tags reference and import the code found in the style.css and script.js tabs.
	
	CSS and JS can both be coded directly into an HTML document. However, it is often easier to navigate between the different languages if they are in separate linked sections.

`<body>...</body>`	This is where all the elements (headings, paragraphs, images, tables, etc.) of an HTML page are defined and contained.
`<h1>...</h1>`	These tags define an element as a level 1 heading.
`<p>...</p>`	These tags define an element as a paragraph.
`<span>...</span>`	Span tags denote an element that does not start on a new line and only takes up as much width as necessary.
`<!--comment-->`	Comments entered between the formatted brackets shown here are ignored when the code is executed. Comments are used to document the function of a line or section of code.
`<br>`	This tag inserts a line break in the web page.

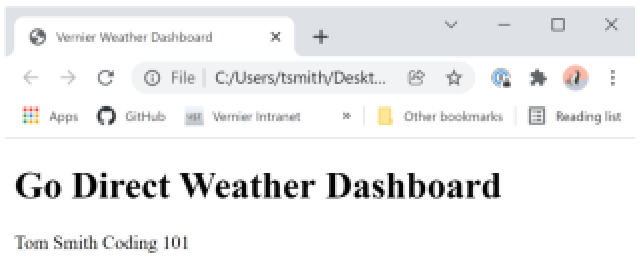
7. Make the following edits to the code:

   a. In the <head> section, edit the title of your web page to an appropriate name. For example, the web page in Figure 3 is titled "Vernier Weather Dashboard". You can watch the change happen dynamically as you type.

   b. In the <body> section of code, edit the <h1> element to display a descriptive name. The example page in Figure 3 displays “Go Direct Weather Dashboard".

   c. Change the <p> element so that your name and the title of your class are displayed.

   d. Save your changes and run the program again.



Go Direct Weather Dashboard

Tom Smith Coding 101

Figure 3 Example of a completed web page made in Activity 1

Choosing Sensor Channels

INTRODUCTION

Over the course of these activities, you have experienced the basics of creating a web page, connecting a Vernier Go Direct device, and displaying the default sensor information (including sensor data). Go Direct devices often have several sensors in addition to the default sensor. The Go Direct Weather Sensor has a total of 11 sensors including temperature, pressure, and wind speed.

For this activity, your Weather Dashboard will continue to have the existing elements that allow you to:

- Connect to a Go Direct device
- Obtain and display live readouts from the default sensor
- Disconnect from the sensor and reset the connection so you can connect again
- Change the visibility of HTML elements (like buttons) so that those visible are relevant

In addition, you will build these capabilities:

- Display a list of all of available sensors
- Select from the list of available sensors
- Display the reading from the selected sensor(s)

For this activity, you will use the following approach:

1. Create global variables and modify the existing selectDevice function in the JavaScript code.
2. Add code to the selectDevice function to display the available channels.
3. Finally, create a function that prompts you to choose a channel and start collecting data.

PROCEDURE

Part I Create Global Variables, Modify selectDevice()

1. Save your project from Activity 7 with a new recognizable name.
2. Create two global variables in your JavaScript code. For consistency, place these with the other global variables declared at the top of your code:

   a. `let enabledSensors;`

   b. `let sensor;`

Part II Add Code to Display the Channels

In this part, you will add code to display the available channels from your connected sensor and remove the code that reads data and displays it on the web page. The functionality of reading and displaying data will be added to the function that allows you to select the sensor channel. This is a complicated maneuver so you are provided with a chunk of code to cut and another chunk to paste into a specific location.

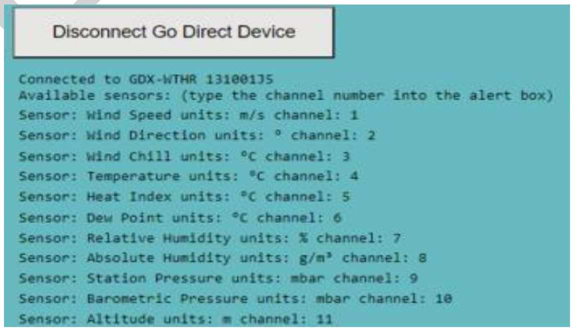
3. Remove the following code. This code activated the default sensor channels and began data collection:

```
//enable access to the default sensors
   gdxDevice.enableDefaultSensors();
   //create a constant enabledSensors that correlates with the enabled
   sensors of the gdx Device
      const enabledSensors = gdxDevice.sensors.filter(s => s.enabled);
      console.log("sensors are enabled");
      //function that runs for each of the enabled sensor measurements
      enabledSensors.forEach(sensor =>{
      //trigger a set of actions to occur whenever the value the sensor
      detects changes
         sensor.on("value-changed", sensor => {
         document.getElementById("data").innerHTML = `\n
${sensor.value.toFixed(2)} ${sensor.unit}`;
});
      });
```

4. In its place, insert the following code; the code causes a list of the available sensor channels to be displayed using the output element, and then calls the chooseChannel function that you will create in a later step.

```
//enable all sensors to output channel info
   enabledSensors = gdxDevice.sensors.filter(s => (s.enabled = true));
   output.textContent += `\n Available sensors: (type the channel number
into the alert box) `;
   enabledSensors.forEach(sensor => {
      output.textContent += `\n\n Sensor: ${sensor.name} units
${sensor.unit} channel: ${sensor.number} `;
   });
   // wait 1 seconds before starting the chooseChannel function
   setTimeout(chooseChannel, 1000);
```

5. Run this code and connect to your sensor to see the sensor channel information displayed on the web page.



```
Connected to GDX-WTHR 131001J5
Available sensors: (type the channel number into the alert box)
Sensor: Wind Speed units: m/s channel: 1
Sensor: Wind Direction units: ° channel: 2
Sensor: Wind Chill units: °C channel: 3
Sensor: Temperature units: °C channel: 4
Sensor: Heat Index units: °C channel: 5
Sensor: Dew Point units: °C channel: 6
Sensor: Relative Humidity units: % channel: 7
Sensor: Absolute Humidity units: g/m³ channel: 8
Sensor: Station Pressure units: mbar channel: 9
Sensor: Barometric Pressure units: mbar channel: 10
Sensor: Altitude units: m channel: 11
```

Figure 1 List of available sensor channels

6. You may find that the first line (or two) of this list of channels is obscured by the Disconnect Sensor button. Use your mouse to scroll up and down to observe the behavior of the web page. The buttons remain fixed in the window due to the position property ("fixed") set in CSS code for buttons. Experiment with other position properties such as "relative" or "sticky". Change your CSS code or add space to your web page to accommodate all of your elements.[1]

Part III Create a Function to Select a Sensor and Display Data

7. Create a new function named chooseChannel that will allow you to select a sensor channel and then begin to display data to your web page. The location of this function is not critical except that it should not be embedded in another function. We suggest placing it immediately after the selectDevice function.

```
function chooseChannel () {
   try {
      //prompt the user for a channel input choice
      const channel = prompt("choose a sensor channel");
      //set the desired sensor according to the channel selection
      const sensor = gdxDevice.getSensor(parseInt(channel));
      //display the sensor channel name selected
      output.textContent = `\n\n Selected sensor: ${sensor.name}`;
      sensor.setEnabled(true); // enable the sensor
      console.log("sensor on");
      //push the sensor data to the "data" element on the web page
      if (gdxDevice.on('device-closed')){
         output.textContent = `\n Disconnected from ` + gdxDevice.name;
      }
      else{
         sensor.on("value-changed", sensor => {
         document.getElementById("data").innerHTML = `\n
         ${sensor.value.toFixed(
            2
         )} ${sensor.unit}`;
      })};
         }
      catch (err) {
         console.error(err)
      }
}
```

Coding Tip

The prompt for choosing a channel returns the entered content as a ‘string'. The parseInt() function in `sensorChannel = gdxDevice.getSensor(parseInt(channel));` converts that string to an integer value expected by this line of code.

[1] The simplest solution may be to insert line breaks `<br>` to move elements down the screen.

9. Run the program to verify that it works correctly. Troubleshoot as necessary. See Figure 2 for an example of the page we created.

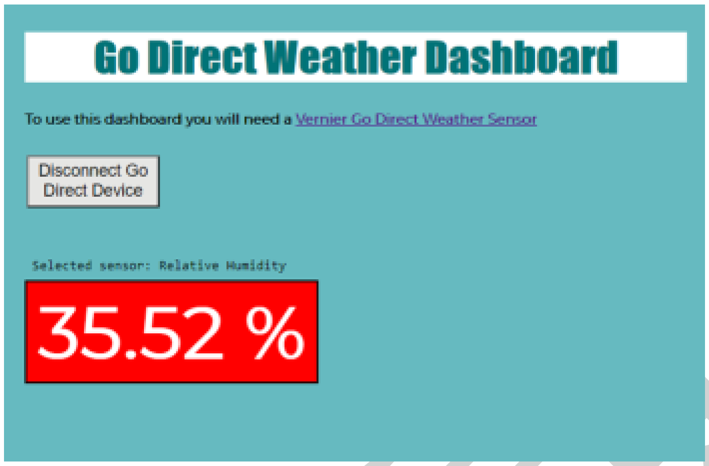

Go Direct Weather Dashboard
To use this dashboard you will need a Vernier Go Direct Weather Sensor
Disconnect Go Direct Device
Selected sensor: Relative Humidity
35.52 %

Figure 2 Displaying relative humidity reading

EXTENSION

Create code that allows you to change sensor channels without disconnecting and reconnecting the sensor.