Activity 2: Displaying Sensor Information

Introduction

In Activity 1 you formatted a web page using HTML. In this activity, you will use the Vernier JavaScript library to connect a sensor to your web page. You will also use a combination of HTML and JavaScript to display basic sensor information on your web page.

Procedure

Importing the Vernier Go Direct Library

1. Open your code from Activity 1.
2. Remix the Activity and make a change to the name, such as act2-*yourname*
3. Between the head tags <head>...</head>, add a pair of script tags. Define its source as "https://unpkg.com/@vernier/godirect/dist/godirect.min.umd.js". This imports the Vernier Go Direct library of commands. This library provides the code that allows the communication between the sensor and a web page. It should look like the following:

```
<script src="https://unpkg.com/@vernier/godirect/dist/godirect.min.umd.js"></script>
```

4. Add a comment above this line to explain the purpose of the code.

```
<!--import gdx library -->
```

Coding Tip

A JavaScript library is a library of pre-written JavaScript that allows for easier development of JavaScript-based applications.

Creating Button Elements with HTML

1. In your HTML file, within the body and below the <h1> tags, create a button with the element "Begin Live Reading" using the <button> tags.
    a. Type this code:

```
<button id="select_device">Select Go Direct Device </button>
```

- The id="select_device" provides a unique id in order to reference the element elsewhere in your code.
- The text "Select Go Direct Device" will appear on the button when you launch the web page.

2. Immediately below your button tags, create a pair of <pre> tags with the id set to output.

- <pre> tag stands for preformatted text. It specifies the format for output from the element as fixed-width font, set spaces, and line breaks.
- Your code should look like this:

```
<pre id="output"> </pre>
```

3. Run this program by clicking on the Show… In a New Window button within Glitch. You should see a button on your web page with the name that you have provided in your code.

Coding Tip

> Add comments as you go through your code so others can follow your logic and you can remember what the code is designed to accomplish.

Connecting your Sensor to the Device Using JavaScript

There are two ways you can integrate JavaScript into your code:

- You can code JavaScript directly in the HTML program by inserting script tags (<script>...</script>). All the code between these tags is interpreted as JavaScript.
- You can code all of your JavaScript in the script.js tab in Glitch and call it from within your HTML file. This is the approach used in these instructions. It should make the distinction between the coding languages more clear.

The JavaScript code is imported into the index.html file using this line of code:

```
    <script src="/script.js" defer></script>
```

Note: Confirm that this code is still present in your index.html file within the head tags.

1. Open the script.js file.
2. Replace the existing code and comments with the following code (you may want to cut and paste it if you have access to an electronic version of this document):

```
//create two variables to represent elements in the html code
    const selectDeviceBtn = document.querySelector('#select_device');
    const output = document.querySelector('#output');

//create an async function to run when selectDevice is clicked
    const selectDevice = async () => {
      try {
        const gdxDevice = await godirect.selectDevice();
        // print name (order code and serial number)
        output.textContent = `\n Connected to `+gdxDevice.name;
      } catch (err) {
        console.error(err);
      }
    };
    selectDeviceBtn.addEventListener('click', selectDevice);
```

Programming Tip

> Commenting in JavaScript is different from HTML. There are two methods for adding comments in JavaScript:
>
> - Typing // before your comment works when comments are confined to a single line.
> - More extensive comments (multiple lines) can be contained between these characters /* … */.
>
> The /* … */ approach is also helpful when you are troubleshooting your program. You can effectively "turn off" sections of code to simplify the program.

3. Turn on your sensor by pressing the power button. The Bluetooth LED will blink red.
4. Run the code by clicking on the Show… button and selecting In a New Window. You should be prompted to select a sensor from a list of available sensors. If you are unsure which sensor is yours, match the serial number found on the label of the sensor.
5. Once you have selected your sensor, you should see the LED on the sensor blink green and the web page should display the sensor order code and serial number.

Analyzing your Code

In the code above, you created several variables. You can think of variables as buckets containing information. JavaScript designates variables by using the keyword "const" or "let". "const" designates a variable whose value is set once and not changed. The value of "let" variables can be changed throughout the execution of the code.

- The code const selectDeviceBtn = document.querySelector('#select_device'); refers to an element in the html named "select_device". Whenever you see code that starts with "document" it will address an element on the webpage.
- Look at the index.html code and see to which element it is linked.

Programming Tip

> Be mindful when naming variables so that the name is descriptive of the function it serves.This makes it easier for yourself and others to understand the code when troubleshooting the program.

- You created a function called "selectDevice":

```
const selectDevice = async () => {
  try {
    const gdxDevice = await godirect.selectDevice();
    // print name and serial number
    output.textContent = `\n Connected to `+gdxDevice.name;
```

```
    } catch (err) {
      console.error(err);
    }
  };
```

- Note that the content of the function is contained within curly brackets and ends with a semicolon.
- Other single lines of code end with a semicolon, as well.
- The code within the “try” section of code runs when the function is called.

- `const gdxDevice = await godirect.selectDevice();` This code causes the program to wait until an available Go Direct sensor is selected. Then the program assigns the sensor to the variable “gdxDevice”.

  The function also includes code in the form of “try {...} catch (err) {...}”. This is a troubleshooting tool that creates a flag when an error is encountered. Without this code, a program that contains an error may simply stop operating without giving any indication where in the program the error occurred. Errors that are caught in this code are logged in the console (“console.error(err)”).

- The function is called by the last line of code in the script.js file:
  `selectDeviceBtn.addEventListener('click', selectDevice);`
  The Event Listener waits for the button to be clicked and then executes the function “selectDevice”. This code is always active in the background, waiting for the button on the web page to be clicked.

Try this:

- Identify all the variables you created in this section of code.
- Identify the two lines of code that are in the try { … } portion of the code. Explain what you think the code does.
- What is the name of the function that is executed when the event listener is activated?

There are several additional pieces of information that you can obtain from the sensor. Here is a partial list:

```
      output.textContent += `\n Order Code: `+gdxDevice.orderCode;
      output.textContent += `\n Typical Measurement Period:
`+gdxDevice.measurementPeriod + ` ms/sample`;
      output.textContent += `\n Battery Level: `+await
gdxDevice.getBatteryLevel() +`%`;
      output.textContent += `\n Charging: ` +(await
gdxDevice.getChargingState() === 1 ? `yes` : `no`);
```

Notice that the Battery Level and Charging information require an “await” term. The sensor has to be queried in a way that takes some time for this information to be transmitted.

Add code to display at least two of these additional pieces of information, and run your program to observe the effect. You will need to add an additional output element for each new element you want displayed on your web page.

Click Show > In a New Window, to run your web page. You should see the information you coded displayed on your web page.

Congratulations! You have successfully completed Activity 2!