Maquenzie Garcia
COMP 484
Professor Marin
02 December 2025

1. Reproduce the Bug
   a. 

   b. 

   c. 5 + 6 = 56
   d. Added in two input fields A and B to represent two values. Rather than adding the numbers, they are being concatenated.

i.

```
/*For demo work, demonstrate concatenation as addition*/
const label = document.querySelector('.bug-example');
const num1Input = document.getElementById('num1');
const num2Input = document.getElementById('num2');
const button = document.querySelector('.bug-example button');

// Add click event to the button
button.addEventListener('click', onClick);

// Functions to get the input values
function getNumber1() {
  return num1Input.value;
}

function getNumber2() {
  return num2Input.value;
}

// Logic taken from google dev tools demo: https://developer.chrome.com/docs/devtools/javascript
function onClick() {
  if (inputsAreEmpty()) {
    label.textContent = "Please enter values in both input fields.";
    return;
  }
  updateLabel()
}

function inputsAreEmpty() {
  if (getNumber1() === "" || getNumber2() === "") {
    return true;
  } else {
    return false;
  }
}

function updateLabel() {
  const addend1 = getNumber1();
  console.log('addend1:', addend1);
  const addend2 = getNumber2();
  console.log('addend2:', addend2);
  const sum = addend1 + addend2;
  console.log('sum:', sum);
  label.textContent = addend1 + ' + ' + addend2 + ' = ' + sum;
}       You, 4 seconds ago • Uncommitted changes
```

ii.

```
*For demo work */
bug-example {
 text-align: center;
 margin-top: 50px;
 padding: 30px;
 background-color: white;
 border-radius: 15px;
 max-width: 600px;
 margin-left: auto;
 margin-right: auto;
```

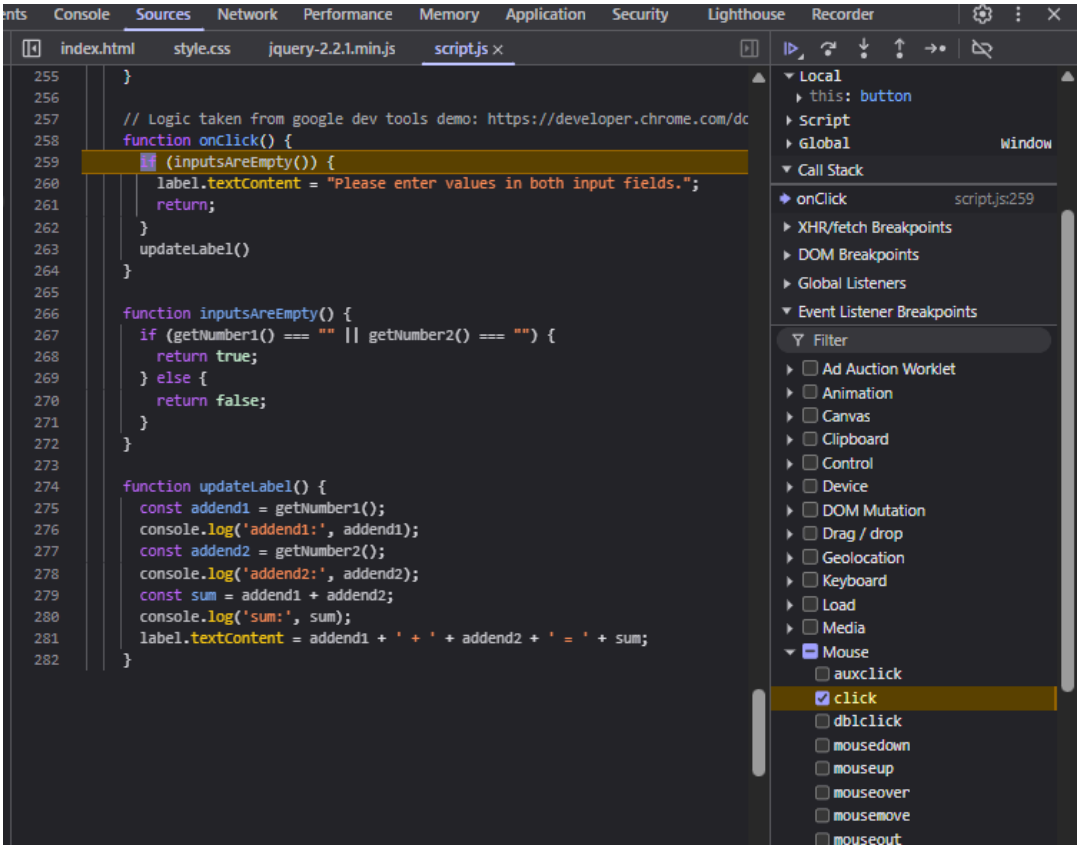
iii.

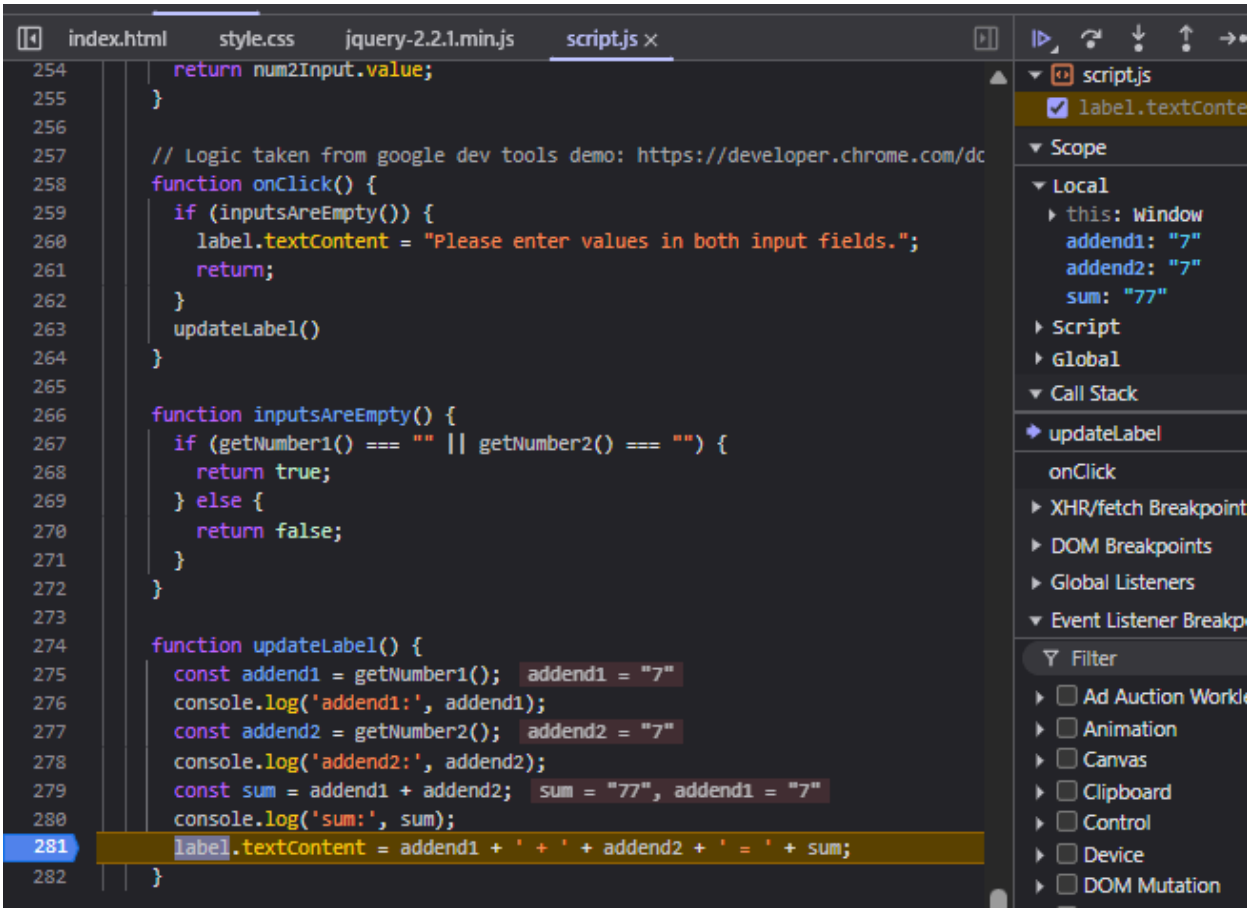
```
<div class="bug-example">
  <h2>Demo: Get Started Debugging JavaScript with Chrome DevTools</h2>
  <label for="num1">Number 1</label>
  <input placeholder="Number 1" id="num1">
  <label for="num2">Number 2</label>
  <input placeholder="Number 2" id="num2">
  <button>Add Number 1 and Number 2</button>
</div>
```

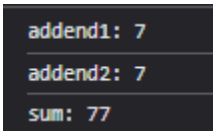
2. Pause the code, Step through code, Set a line-of-code breakpoint,

a.

b.

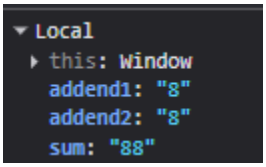


c. 

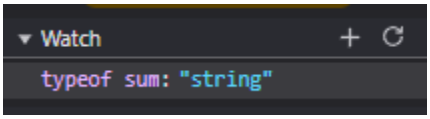
d. 

e. Code was paused, breakpoint added using event listener. We needed to move through the various functions by stepping into and stepping over the function calls.

3. Check variable values, (Method 1, 2,3)

a. 

b. Method 1 was inspecting the scope. In viewing my local vars, we see the different values for the addends and notice how they are wrapped in quotes- an indicator that our addition may be string concatenation.

c. 

d. Method 2 utilizes the ‘watch’ feature. Here we see that our sum variable is actually a string.

e.
```
> parseInt(addend1) + parseInt(addend2)
< 16
```

f. Method 3 involves the use of the console. If we parse the addends as ints, we see that the correct value is returned when added together (8 + 8 is 16 rather than 88).

4. Apply a fix

a. 4 + 6 = 10

b.
```
const sum = parseInt(addend1) + parseInt(addend2);
```

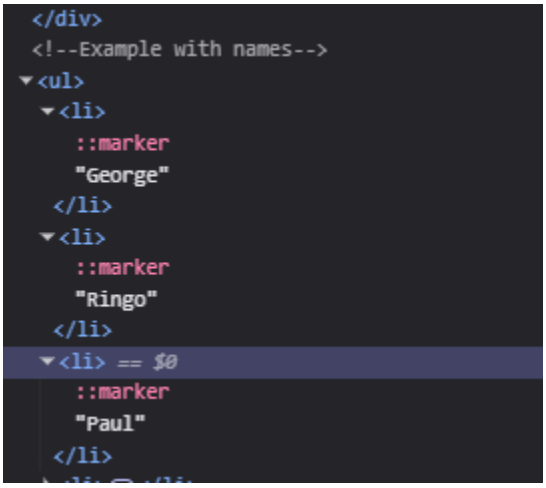
c. Added fix by parsing in the addends as integers.

5. Examples From (View DOM nodesLinks to an external site-> https://developer.chrome.com/docs/devtools/dom)
   a. Examples done below!
6. View DOM nodes (all sub parts)

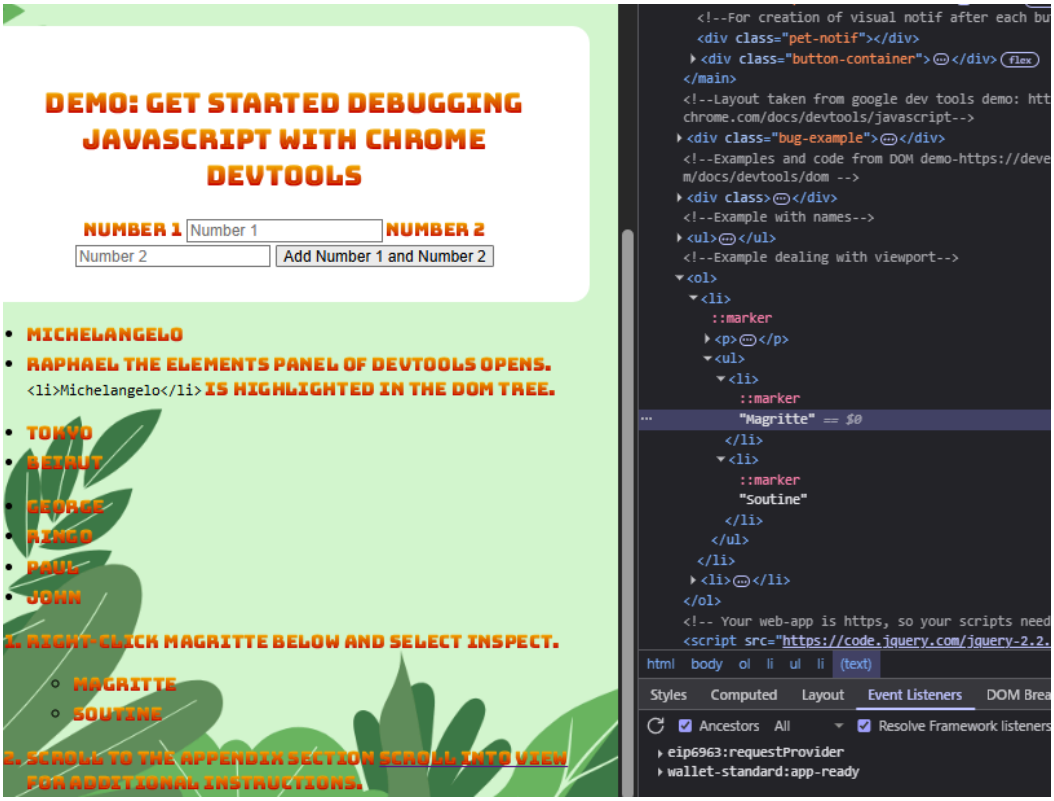
a.
```
<!--Example with painters-->
<ul>
  <li>
    ::marker
    "Michelangelo"
  </li>
  <li>…</li>
</ul>
<!--Example with the cities-->
<ul>
  <li>
    ::marker
    "Tokyo" == $0
  </li>
  <li>…</li>
</ul>
```

b.
```
--Examples and code from DOM demo-https://developer.chrome.com/docs/devt
iv class>
<!--Example with painters-->
<ul>
  <li>Michelangelo</li>
  <li>Raphael
  The <strong>Elements</strong> panel of DevTools opens.
  <code translate="no" dir="ltr">&lt;li&gt;Michelangelo&lt;/li&gt;</code
</ul>
<!--Example with the cities-->
<ul>
  <li>Tokyo</li>
  <li>Beirut</li>
</ul>
div>
```

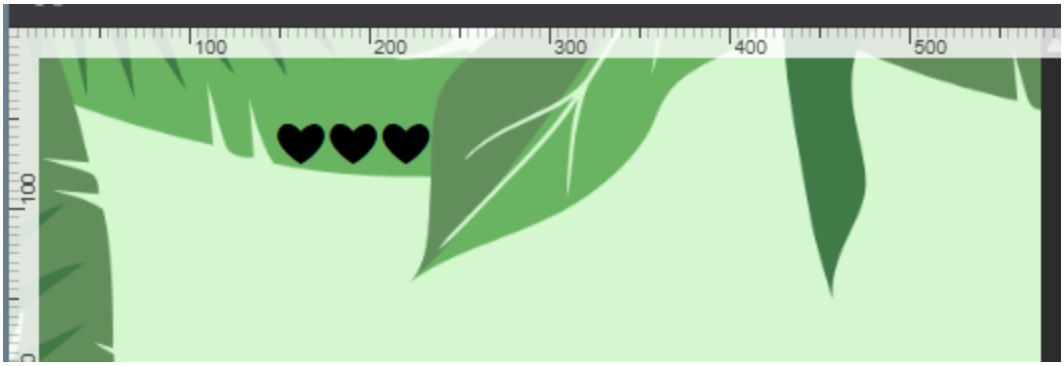
c. Added in the same lists from the tutorial. Inspected Michelangelo and Tokyo.

d. 

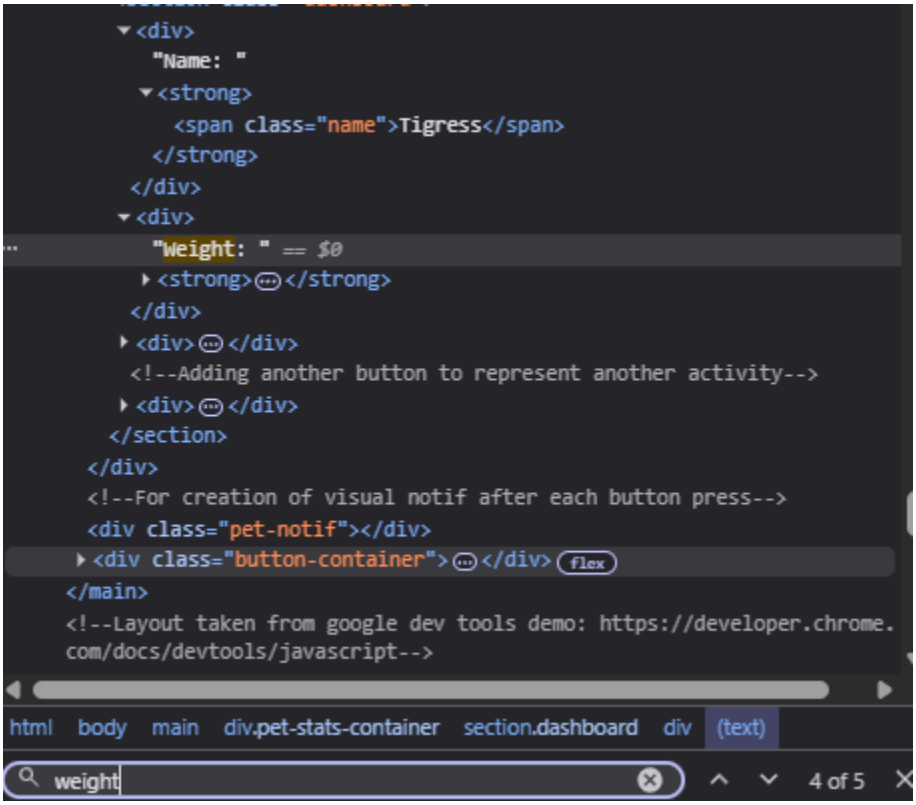
e. Using arrow keys to navigate through the list and open them up. Up/down arrow to navigate through the items, right/left to open and close the items themselves.

f. 

g. Scroll into view to take user back to point of interest.

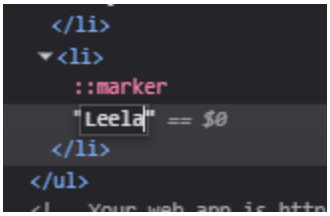
h. 

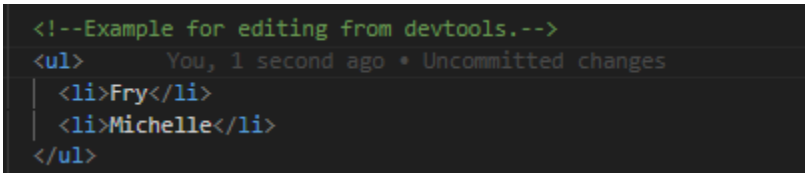
i. Addition of show rulers via command ctrl+shift+p.

j. 

k. Search feature. Searched for ‘weight’, which was an important variable in my project 2.

7. Edit the DOMLinks to an external site. (Edit the content, Edit attributes, Edit node type, Edit HTML)
    a. Edit the content

i. 

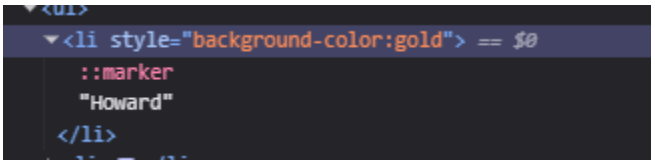


ii. 

iii. I added the sample list into my file and then used the devtools to directly edit the element from Michelle to Leela.

b. Edit Attributes

i. 

ii. 

iii. Added in a new attribute to the list element to make the background gold.

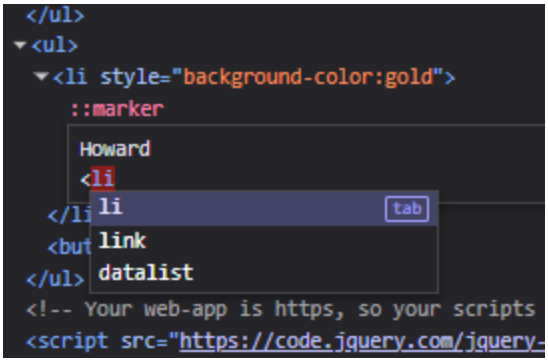
c. Edit Node Type

i. 

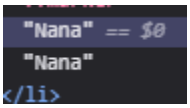
ii. 

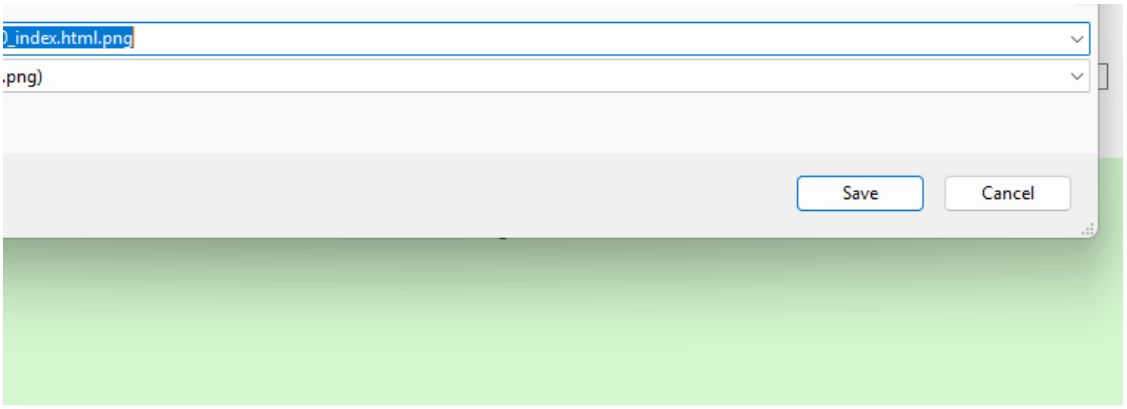
iii. Change the li element to a button element.

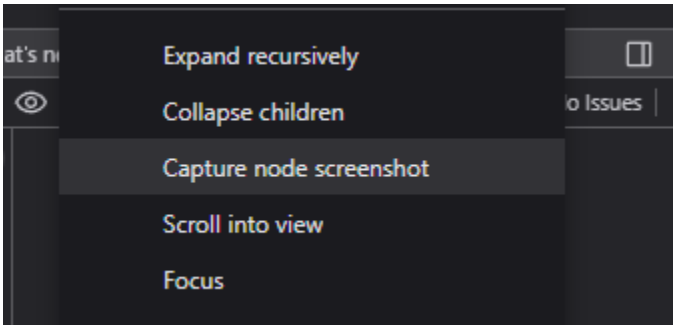
d. Edit HTML

i. 

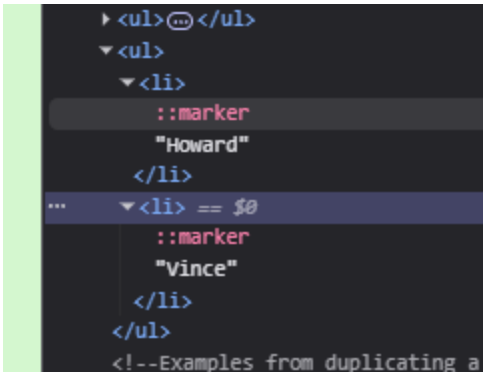
ii. Editing the elements as html using autocomplete within the dev tools.

8. (Duplicate a node, Capture a screenshot, Reorder DOM nodes, Force state, Hide a node, Delete a node)
   a. Duplicate a Node
      i. 

      ii. Right click to duplicate a value in the test list.
   b. Capture a Screenshot
      i. 

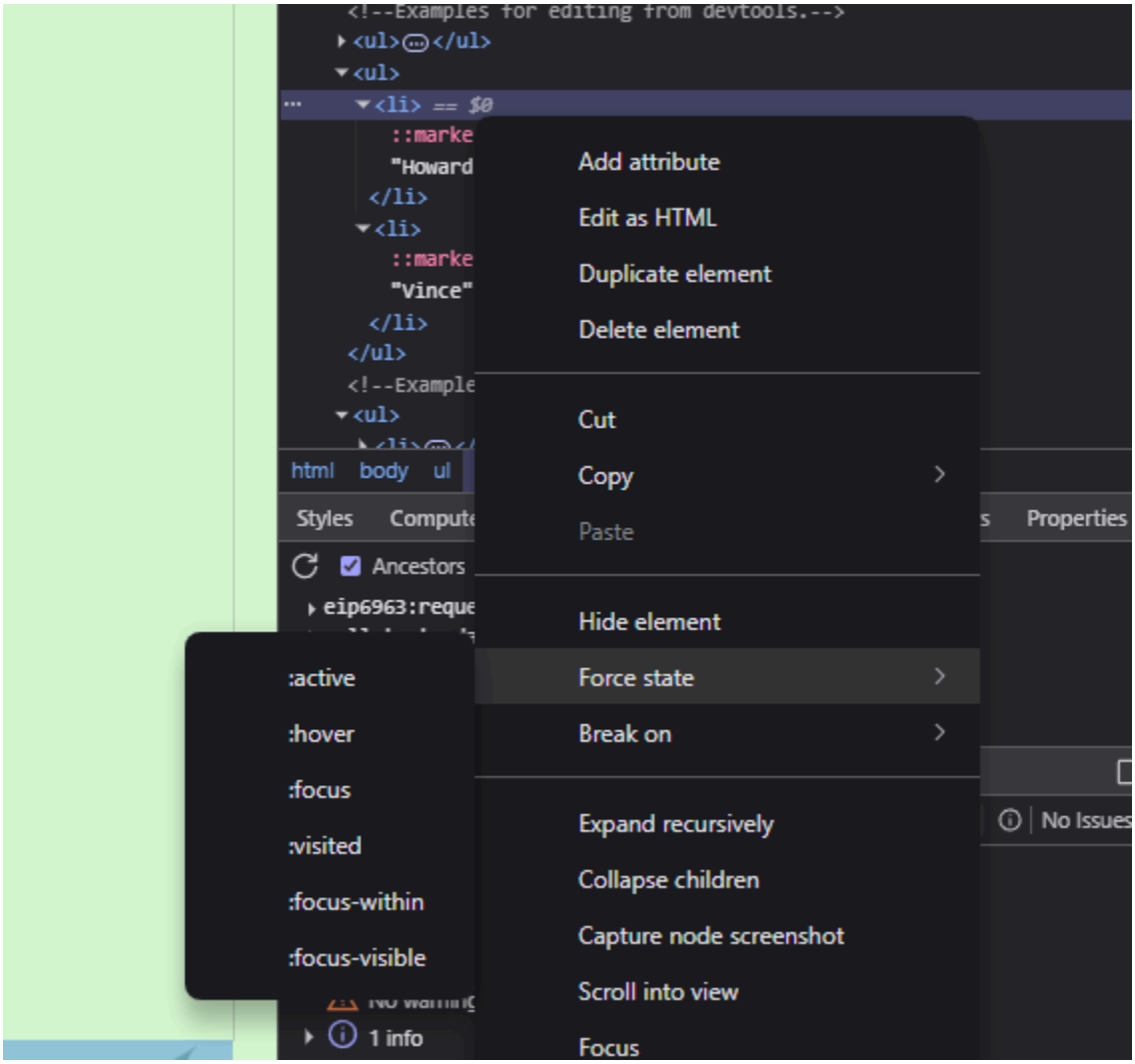
      ii. 

      iii. Capture node screenshot by right clicking and selecting the above option.
   c. Reorder DOM nodes

i. 

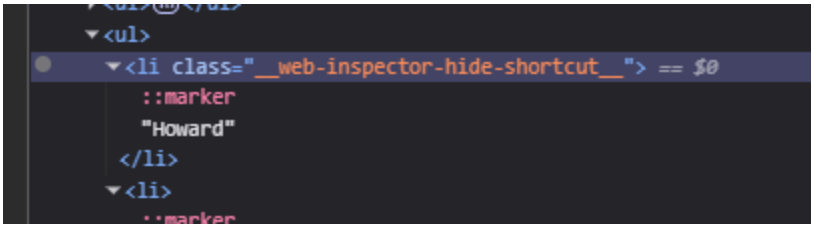
ii. Hold and drag the varying elements to change their order. I added the lists from the demos to my project, opened them with live server, opened dev tools on live server, and reordered them like so.

d. Force State

i. 

ii. Select any of the states from the dropdown for the nodes.

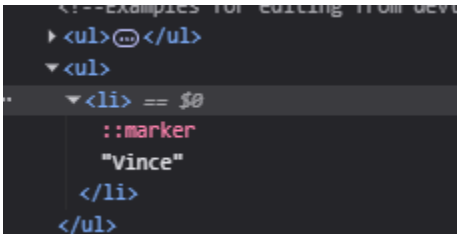
e. Hide a Node

i. 

ii. 

iii. With the sample lists, I hid the list element containing Howard.

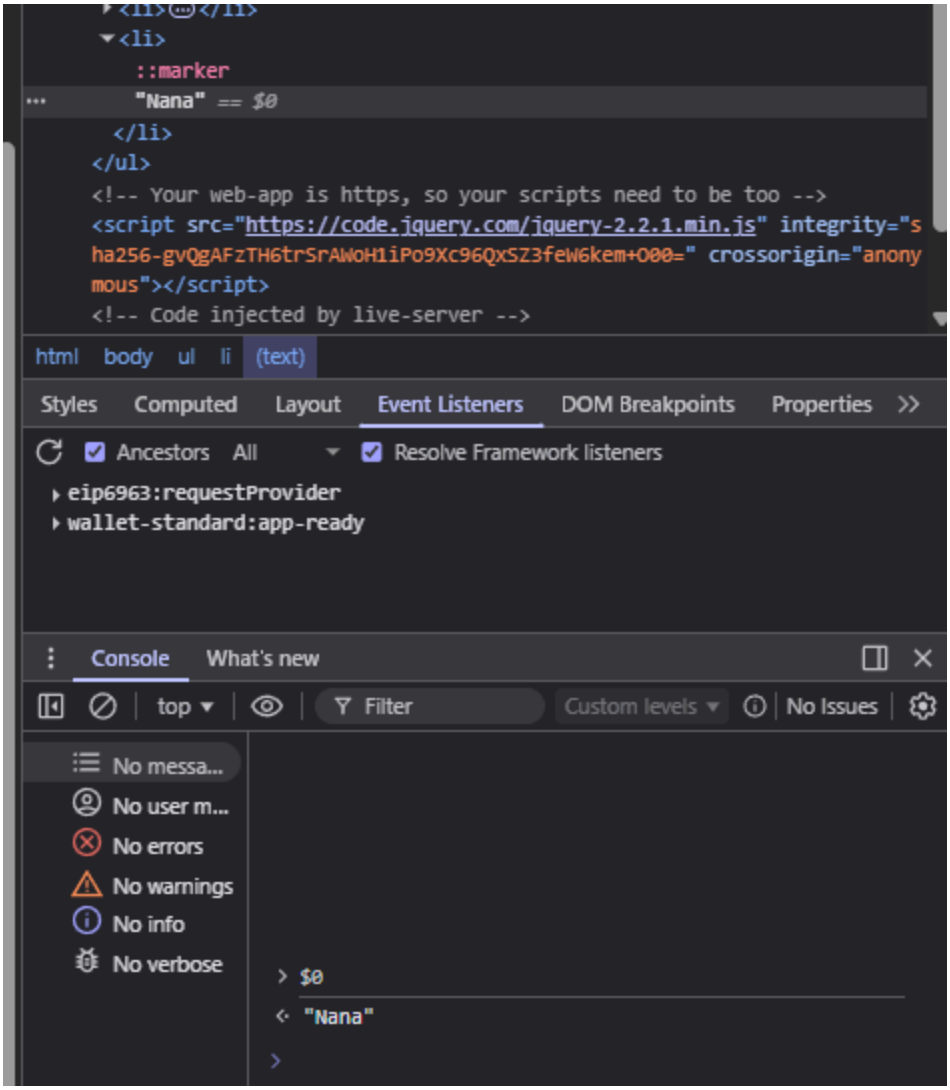
f. Delete a Node

i. 

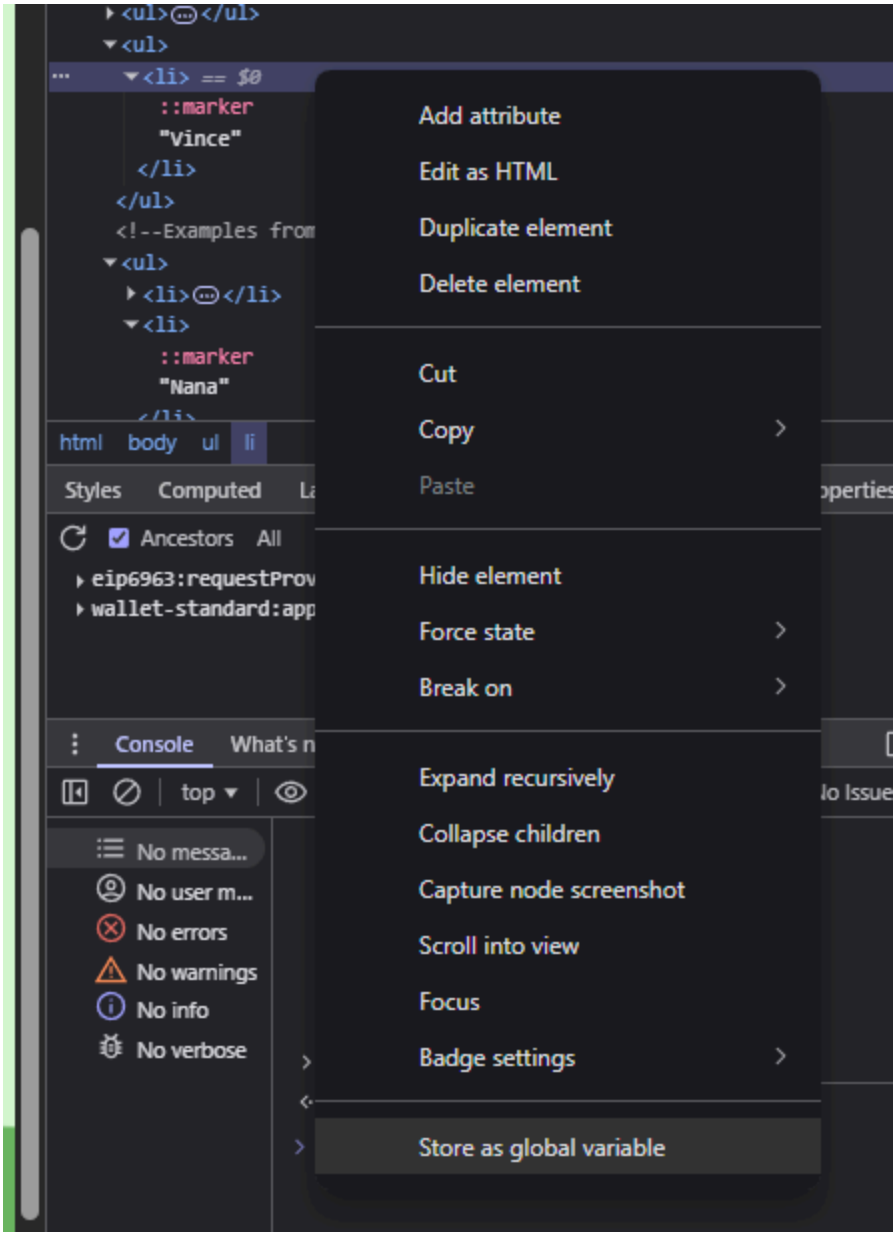
ii. 

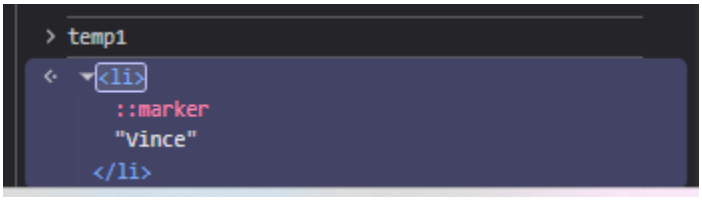
iii. The Howard element is now deleted.

9. Access Nodes in the Console

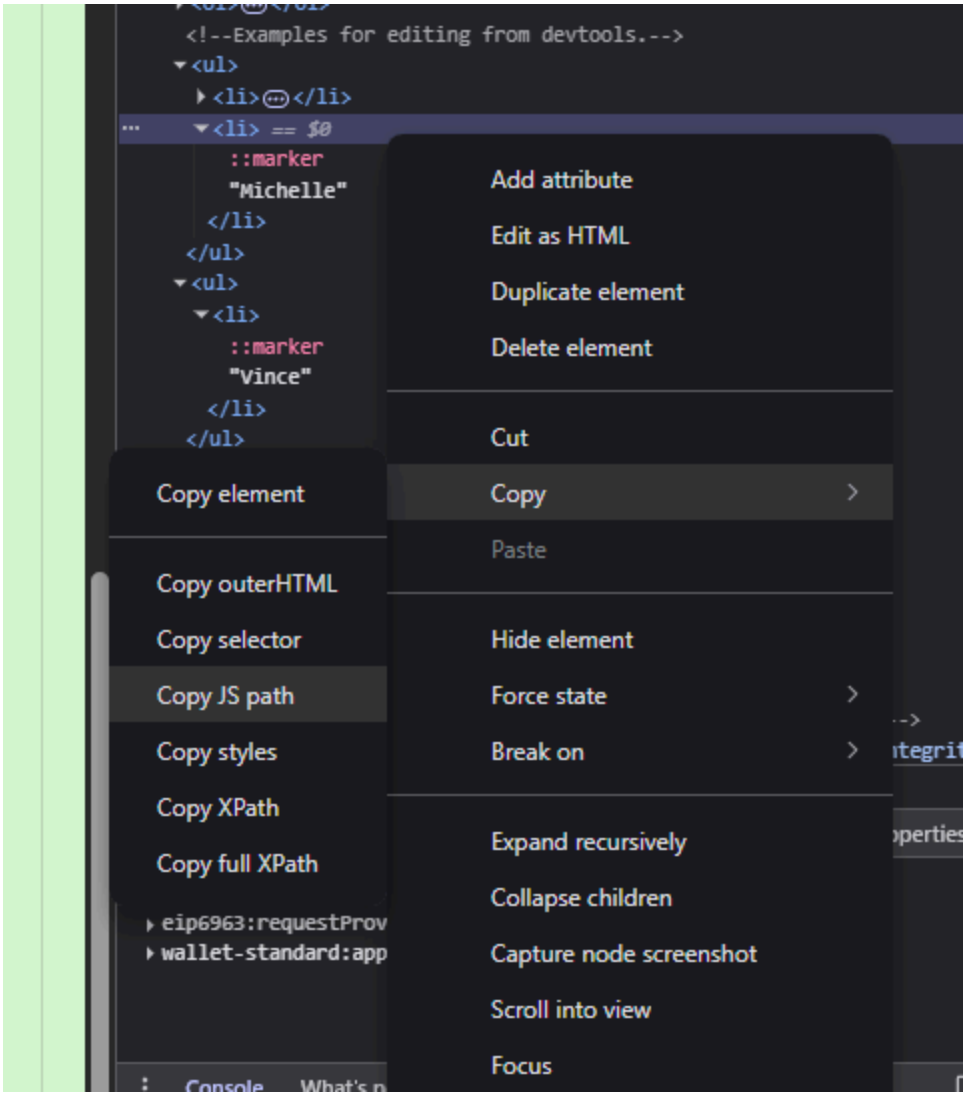
a. 

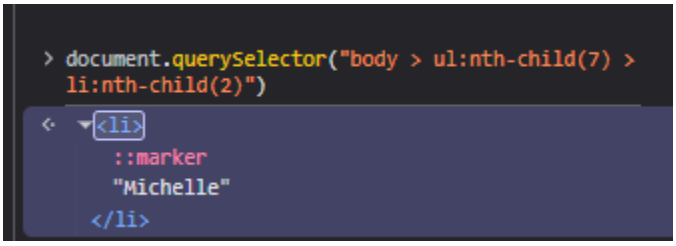
b. For this example, I added the various lists into my project. From there we can open up the project in live server, open dev tools, and select on a node. Selecting the node gives that '== $0' value, and typing in $0 into the console allows us to access that node.

c. 

d. 

e. We are also able to store nodes as global variables. Here, I selected a node from one of the sample lists I added. I stored the list element containing ‘Vince’ as a global variable, and now we are able to access it in the console as ‘temp 1’.

f. 

g. 

h. For this example, I took one of the lists I copied over and copied the js path. From there, I copy pasted the path into the console. Using the JS path, we are able to retrieve the node.