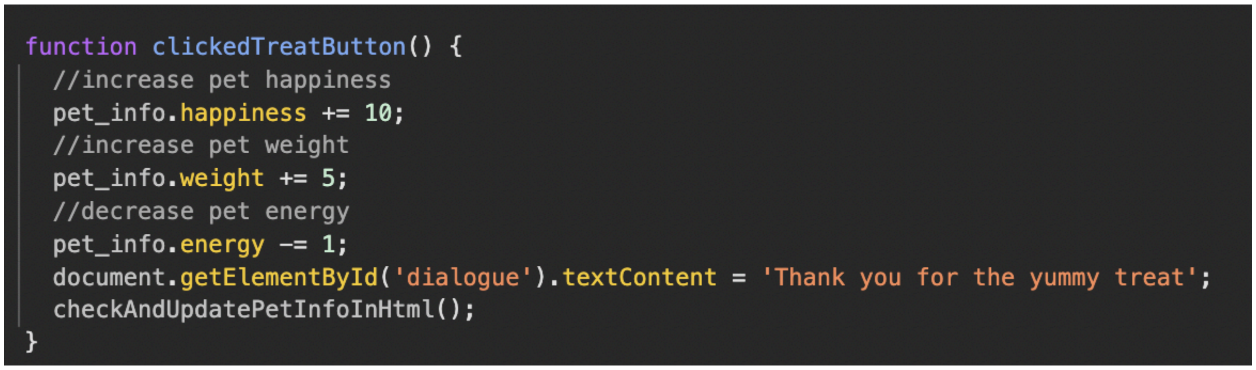
Jason Tran
COMP 484

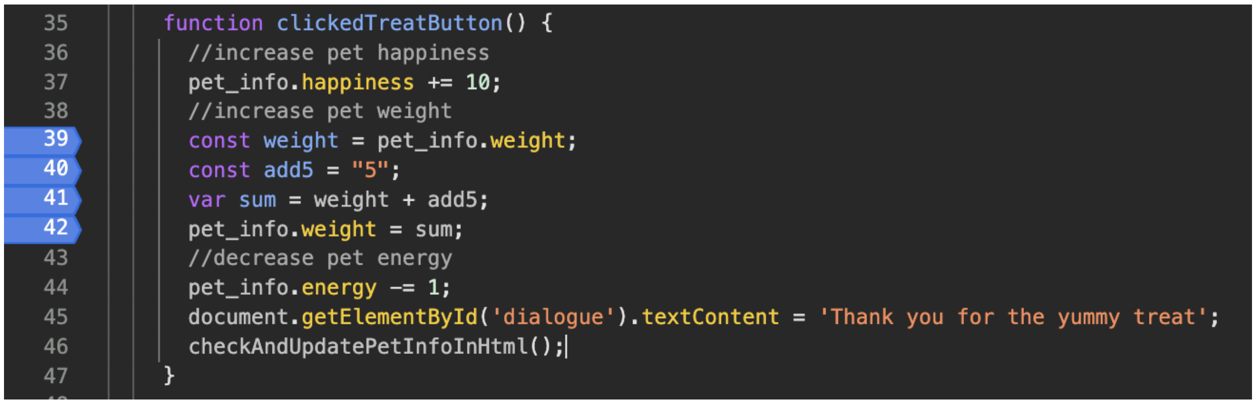
Chrome Dev Tools Documentation

First we start off with the Debug JavaScript on the Chrome Dev Tools
Here I will modify my clickedTreatButton to be outputting an incorrect value after each update and adding breakpoints to check on modified values instead of using a console.log
Original Code:



```
function clickedTreatButton() {
  //increase pet happiness
  pet_info.happiness += 10;
  //increase pet weight
  pet_info.weight += 5;
  //decrease pet energy
  pet_info.energy -= 1;
  document.getElementById('dialogue').textContent = 'Thank you for the yummy treat';
  checkAndUpdatePetInfoInHtml();
}
```

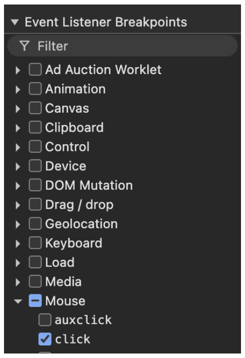
Modified Code:



```
35  function clickedTreatButton() {
36    //increase pet happiness
37    pet_info.happiness += 10;
38    //increase pet weight
39    const weight = pet_info.weight;
40    const add5 = "5";
41    var sum = weight + add5;
42    pet_info.weight = sum;
43    //decrease pet energy
44    pet_info.energy -= 1;
45    document.getElementById('dialogue').textContent = 'Thank you for the yummy treat';
46    checkAndUpdatePetInfoInHtml();
47  }
```

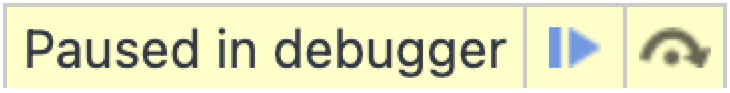
This code is outputting an incorrectly concatenated value instead of adding the values together. Also has breakpoints in between.

Then we follow with adding an Event Listener Breakpoint on mouse click which will automatically pause when any click event listener executes

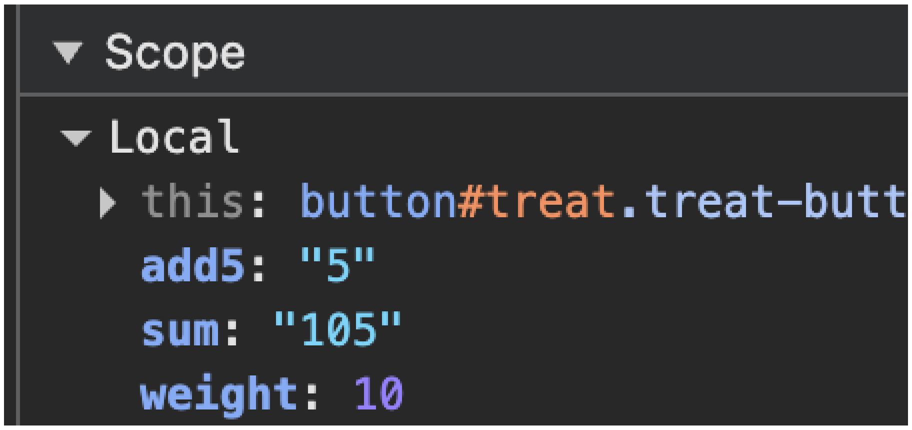


Event Listener Breakpoints
Filter
- [ ] Ad Auction Worklet
- [ ] Animation
- [ ] Canvas
- [ ] Clipboard
- [ ] Control
- [ ] Device
- [ ] DOM Mutation
- [ ] Drag / drop
- [ ] Geolocation
- [ ] Keyboard
- [ ] Load
- [ ] Media
- Mouse
  - [ ] auxclick
  - [x] click

With this each breakpoint pauses the code step by step on each breakpoint also including our initial button press because of our event listener breakpoint

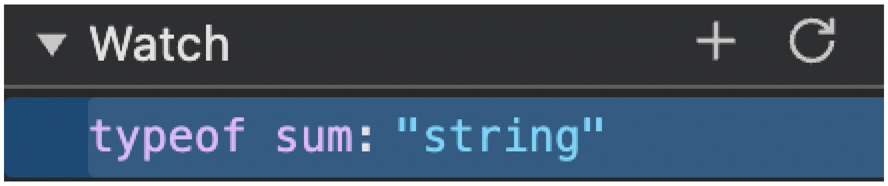


By inspecting our local scope we can see an issue that has been occurring where our add5 value is being read as a string and our sum value is being read as a string



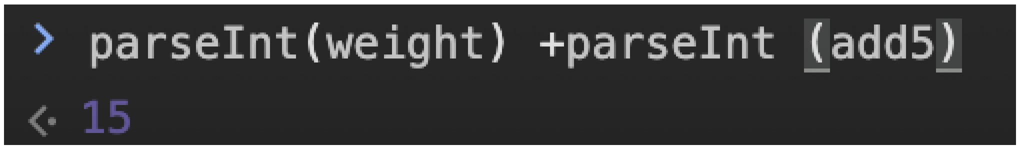
```
▼ Scope
  ▼ Local
    ▶ this: button#treat.treat-butt
      add5: "5"
      sum: "105"
      weight: 10
```

While using our Watch tab and monitoring the typeof our sum variable we can see that it is being stored as a string and not as an integer value. We can see that our bug is revolving around our sum value being a string



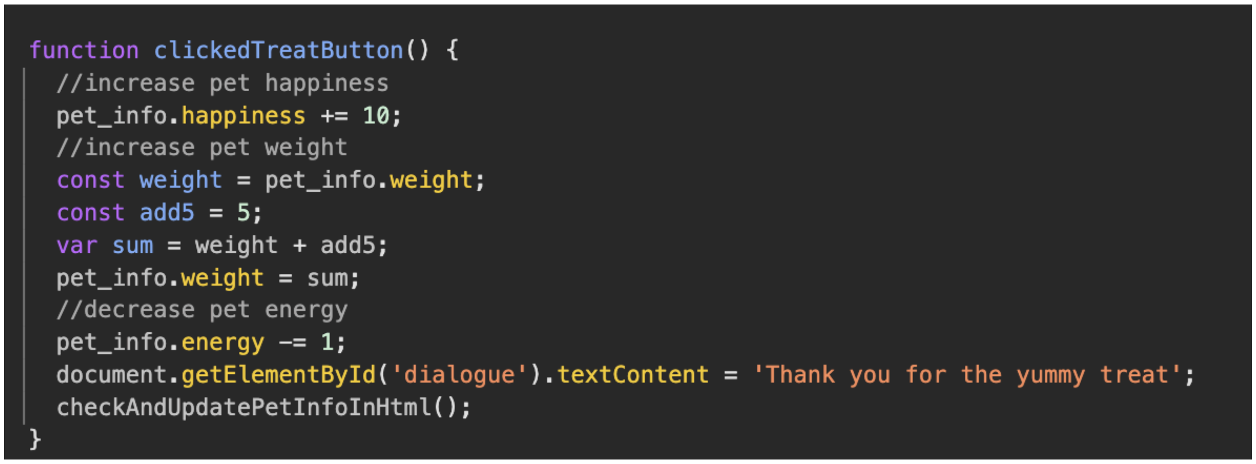
```
▼ Watch
    typeof sum: "string"
```

In addition we can use our console to parse the integer values to double check that when done properly we would get the proper value and when parsed we do get a proper addition



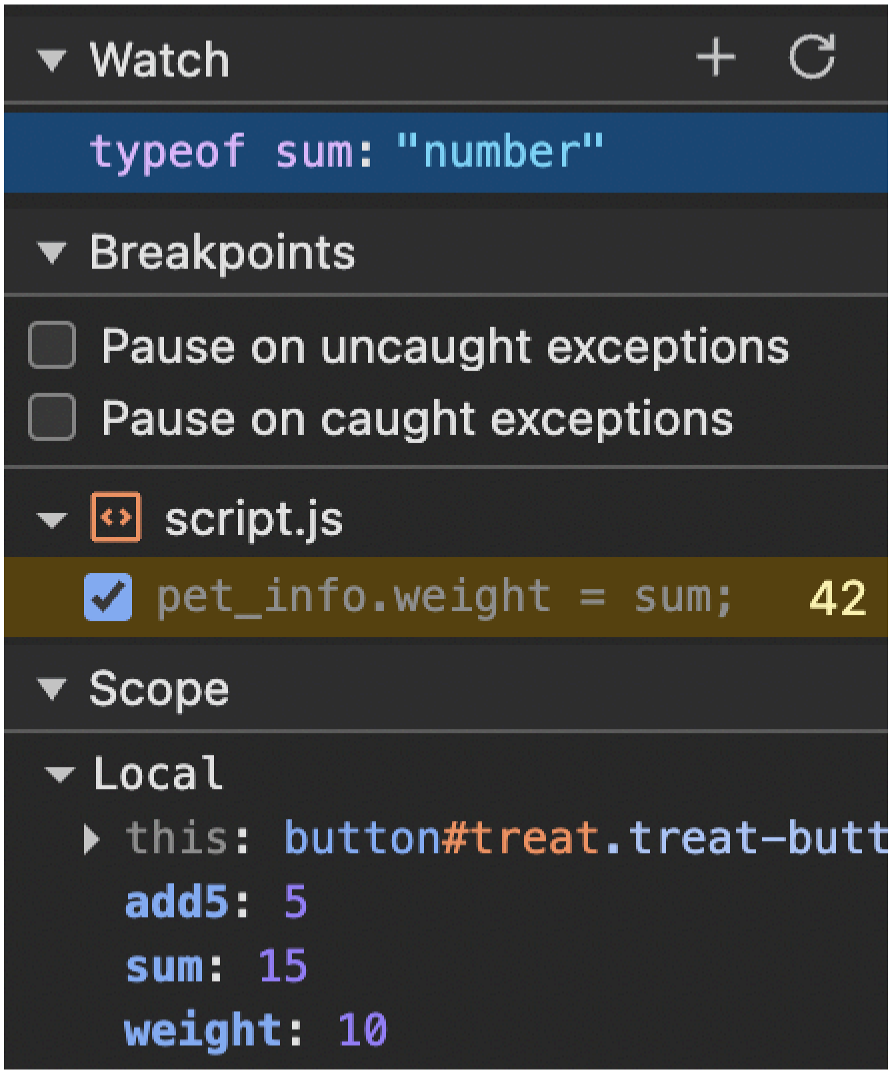
```
> parseInt(weight) +parseInt (add5)
<· 15
```

By modifying the code again and removing the quotation marks on the 5 in the add5 variable



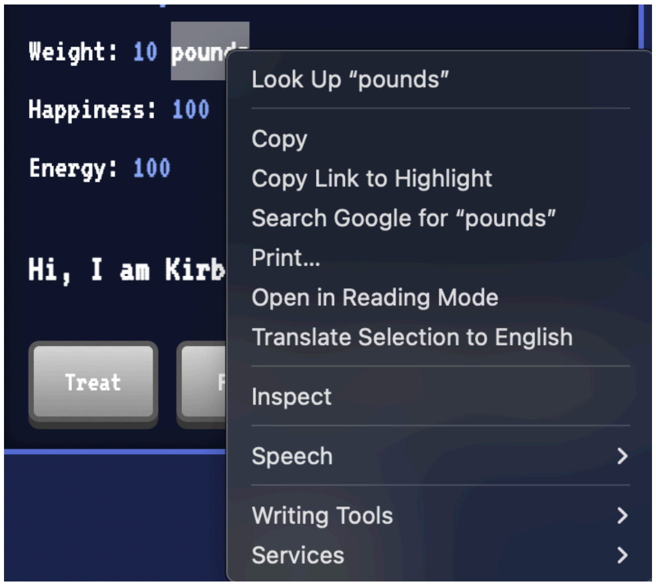
```
function clickedTreatButton() {
  //increase pet happiness
  pet_info.happiness += 10;
  //increase pet weight
  const weight = pet_info.weight;
  const add5 = 5;
  var sum = weight + add5;
  pet_info.weight = sum;
  //decrease pet energy
  pet_info.energy -= 1;
  document.getElementById('dialogue').textContent = 'Thank you for the yummy treat';
  checkAndUpdatePetInfoInHtml();
}
```

We can see our sum is now of the proper type number and the local scope has the correct values being outputted with that simple change.

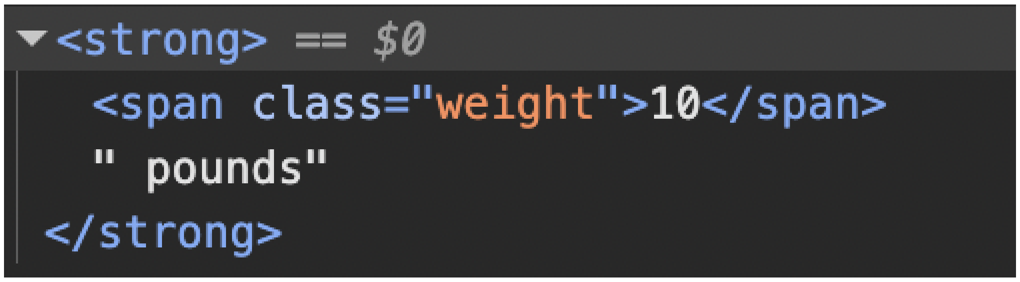


```
▼ Watch
    typeof sum: "number"
▼ Breakpoints
  ☐ Pause on uncaught exceptions
  ☐ Pause on caught exceptions
▼ script.js
    ☑ pet_info.weight = sum;   42
▼ Scope
 ▼ Local
   ▶ this: button#treat.treat-butt
     add5: 5
     sum: 15
     weight: 10
```

Chrome Tools DOM
Starting off with using the Chrome Dev tools to change a dom we will right click and inspect a node in this case will be the word pounds in our pets weight category

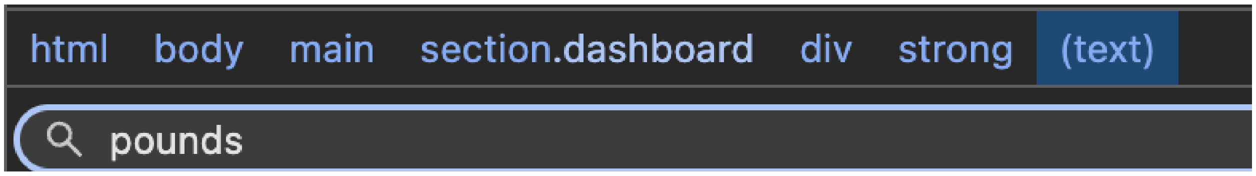


Where it would be highlighted in its strong tag

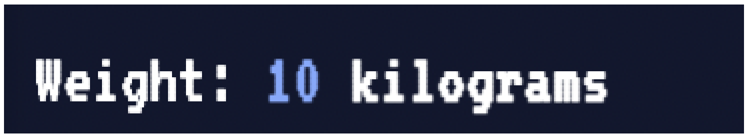


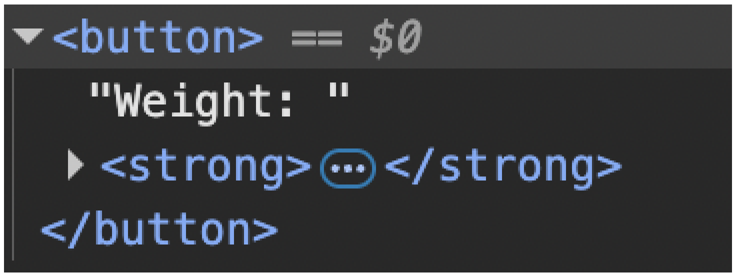
```
<strong> == $0
  <span class="weight">10</span>
  " pounds"
</strong>
```

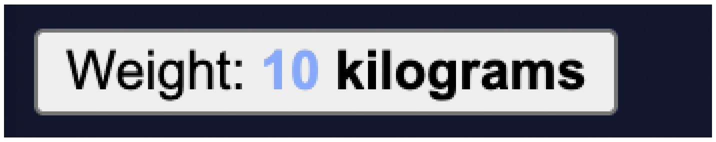
We can search for specific phrases and headings with Ctrl F as well



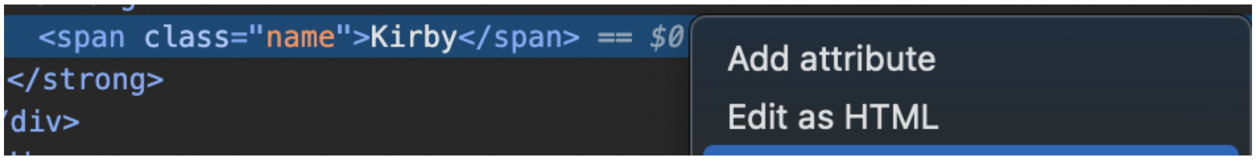
We can and edit attributes like the names and types of

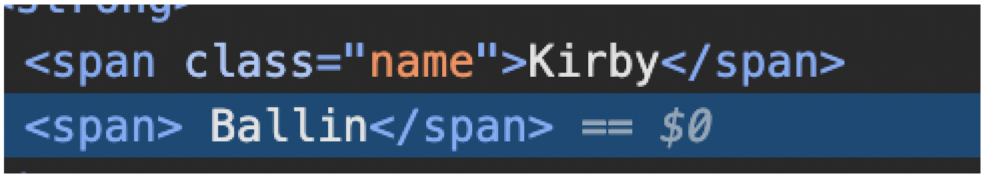


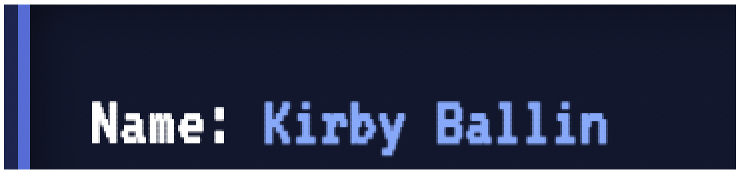

```
<button> == $0
  "Weight: "
  <strong>…</strong>
</button>
```


Weight: 10 kilograms

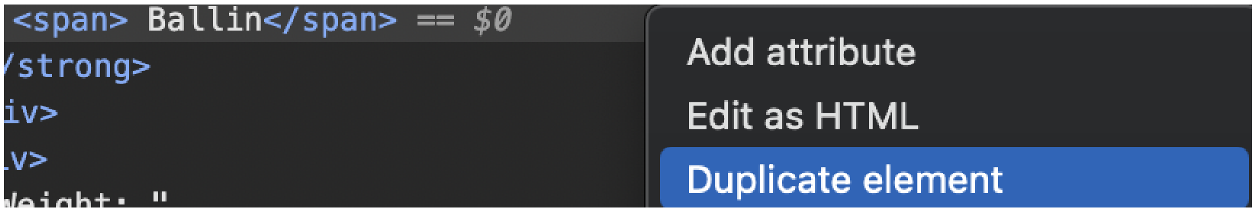
That was editing the DOM now we can go into editing the HTML of a page by adding new tags and new phrases into the html

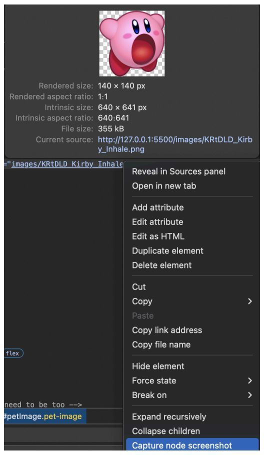

```
<span class="name">Kirby</span> == $0
</strong>
/div>
```


```
<span class="name">Kirby</span>
<span> Ballin</span> == $0
```


Name: Kirby Ballin

We can duplicate elements on each node


```
<span> Ballin</span> == $0
/strong>
iv>
v>
```

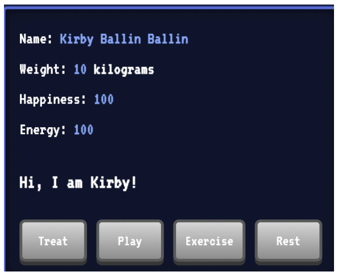


Rendered size: 140 × 140 px
Rendered aspect ratio: 1:1
Intrinsic size: 640 × 641 px
Intrinsic aspect ratio: 640:641
File size: 355 kB
Current source: http://127.0.0.1:5500/images/KRtDLD_Kirby_Inhale.png

```
="images/KRtDLD_Kirby_Inhale
need to be too -->
#petImage.pet-image
```

By Inspecting a Page Image we can go into the URL in the HTML to capture the node screenshot and save the image

Original



Name: Kirby Ballin Ballin

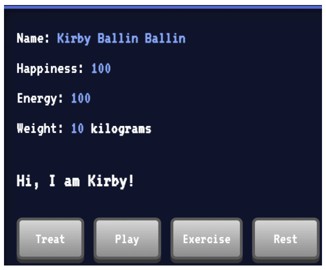
Weight: 10 kilograms

Happiness: 100

Energy: 100

Hi, I am Kirby!

Modified



Name: Kirby Ballin Ballin

Happiness: 100

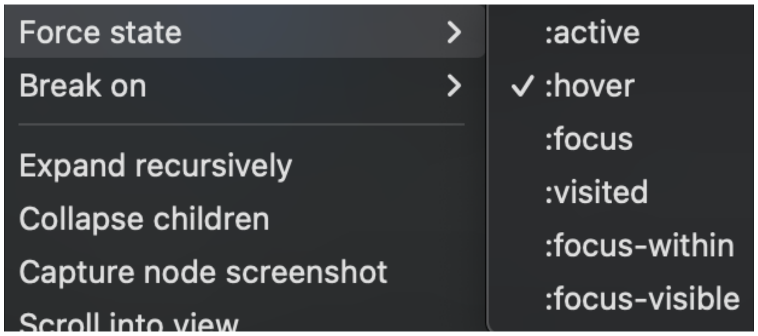
Energy: 100

Weight: 10 kilograms

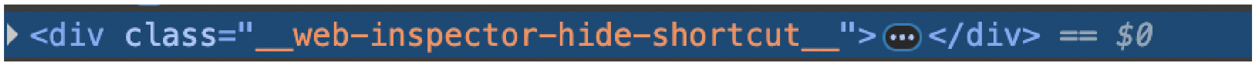
Hi, I am Kirby!

By dragging and dropping nodes in the DOM tree we can replace values as you can see in the modified screenshot i moved the weight value to go all the way onto the bottom the list

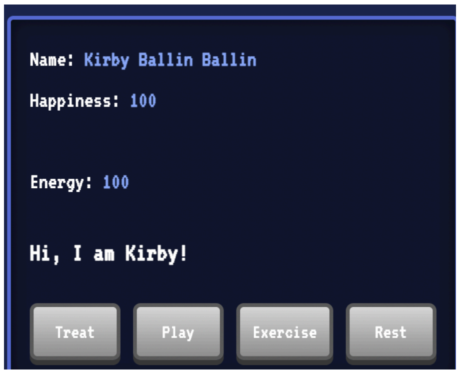
We can force states of items in our DOM tree to remain having certain affects such as hovering and focus



By selecting the Hide Element Option we can hide elements until we press the H key on similar note we can also use the Delete node option to delete a specific node and undo with Ctrl Z

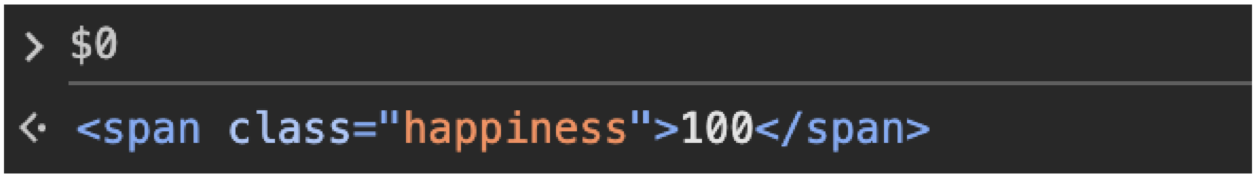


```
<div class="__web-inspector-hide-shortcut__">…</div> == $0
```



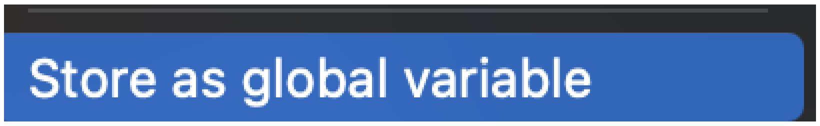
```
Name: Kirby Ballin Ballin
Happiness: 100

Energy: 100

Hi, I am Kirby!
```

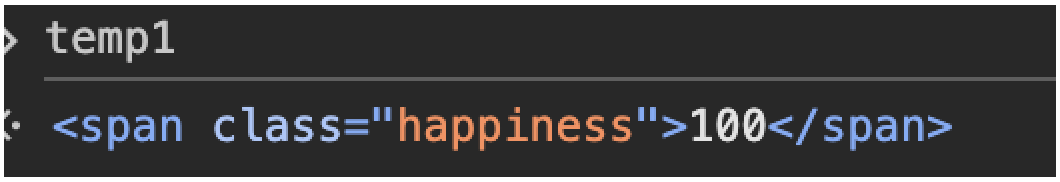
By typing $0 in the console while we have an item inspected we can view the node within our console



```
> $0
<· <span class="happiness">100</span>
```

We can refer back to nodes many times with global variables when selecting a DOM node you can store as a global variable and the console will store it within a temp

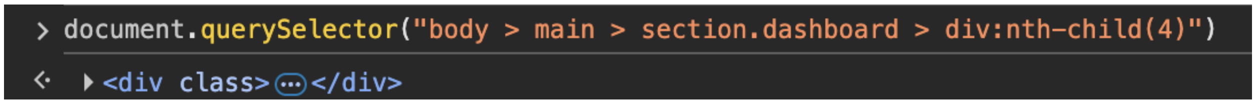




```
> temp1
<· <span class="happiness">100</span>
```

Along the same lines of using the console you can copy the JS Path a node in the DOM Tree has and it will result in a document.querySelector() expression to be copied on the clipboard and the console can read that expression

document.querySelector("body > main > section.dashboard > div:nth-child(4)")



```
> document.querySelector("body > main > section.dashboard > div:nth-child(4)")
<·  ▸<div class>…</div>
```