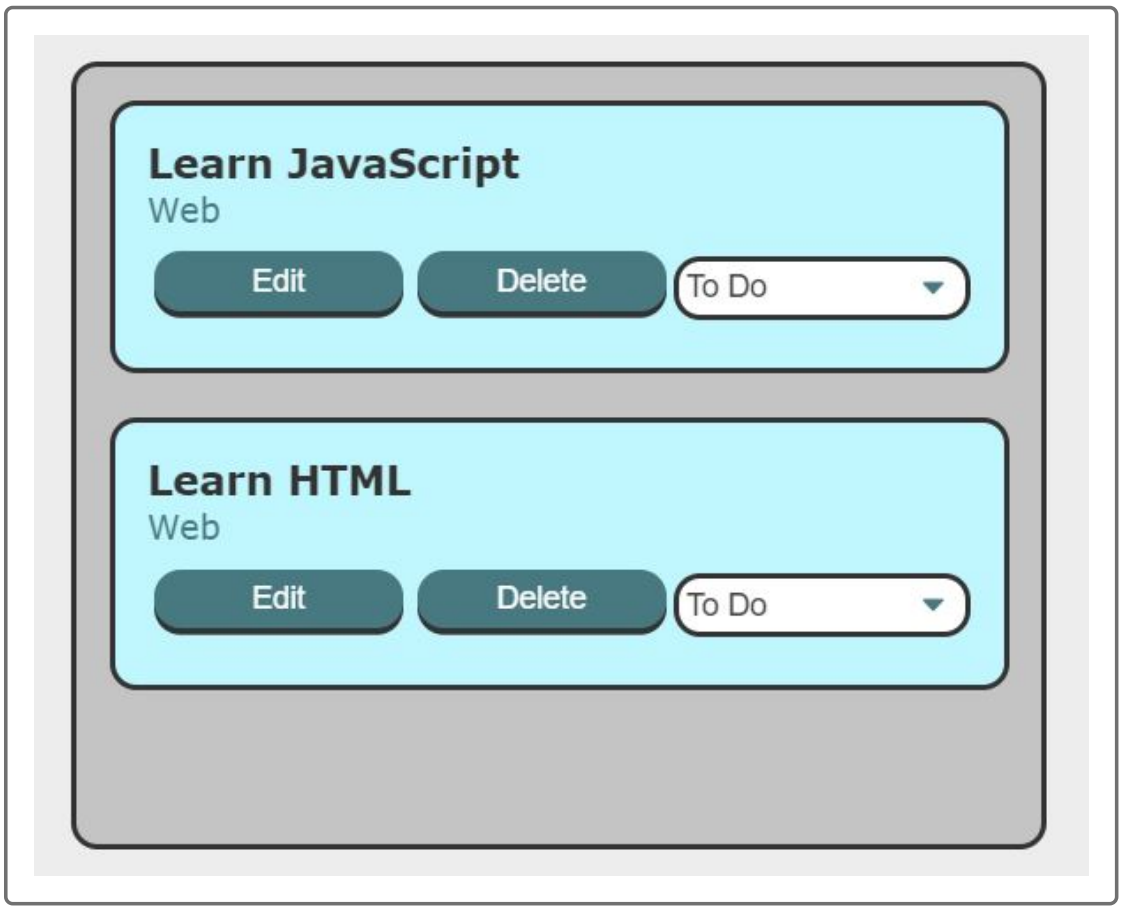
4.3.5 Apply Unique IDs to Tasks

Before we can implement the JavaScript logic to update tasks, we need to formalize a way to uniquely identify each task that's created. Consider the following example:



If you were to click the Delete button under "Learn JavaScript," how would you programmatically know which task the button was referring to? This is where assigning an ID to each task would help keep track of what's what.

CONNECT THE DOTS

Later on in your web development journey, you'll start using actual databases like MySQL to store your app's information. In database design, unique IDs are crucial in ensuring that the correct data is being updated or deleted. Databases can be set up to create that ID for you, but understanding their value now will better prepare you for the future.

There are a few ways we could generate this ID:

- Use `Math.random()` to (hopefully) create a unique ID each time we use it.
- Get the current time in milliseconds and call that an ID.
- Create a counter that increments by one each time a task is created.

The third option more closely matches what a MySQL database would do, and it doesn't require any new JavaScript methods, so we'll go with that.

At the top of `script.js`, declare a new variable:

```
var taskIdCounter = 0;
```

```
  // add task id as a custom attribute
  listItemEl.setAttribute("data-task-id", taskIdCounter);

  var taskInfoEl = document.createElement("div");
  taskInfoEl.className = "task-info";
  taskInfoEl.innerHTML = "<h3 class='task-name'>" + taskDataObj.name +
  listItemEl.appendChild(taskInfoEl);

  tasksToDoEl.appendChild(listItemEl);

  // increase task counter for next unique id
  taskIdCounter++;
};
```

Because we're making HTML elements in JavaScript, we needed to use the DOM method `setAttribute()` to add our task ID. The `setAttribute()` method can be used to add or update any attribute on an HTML element, but the only attribute we need for now is `data-task-id`, which we set to the current value of `taskIdCounter`.

PAUSE

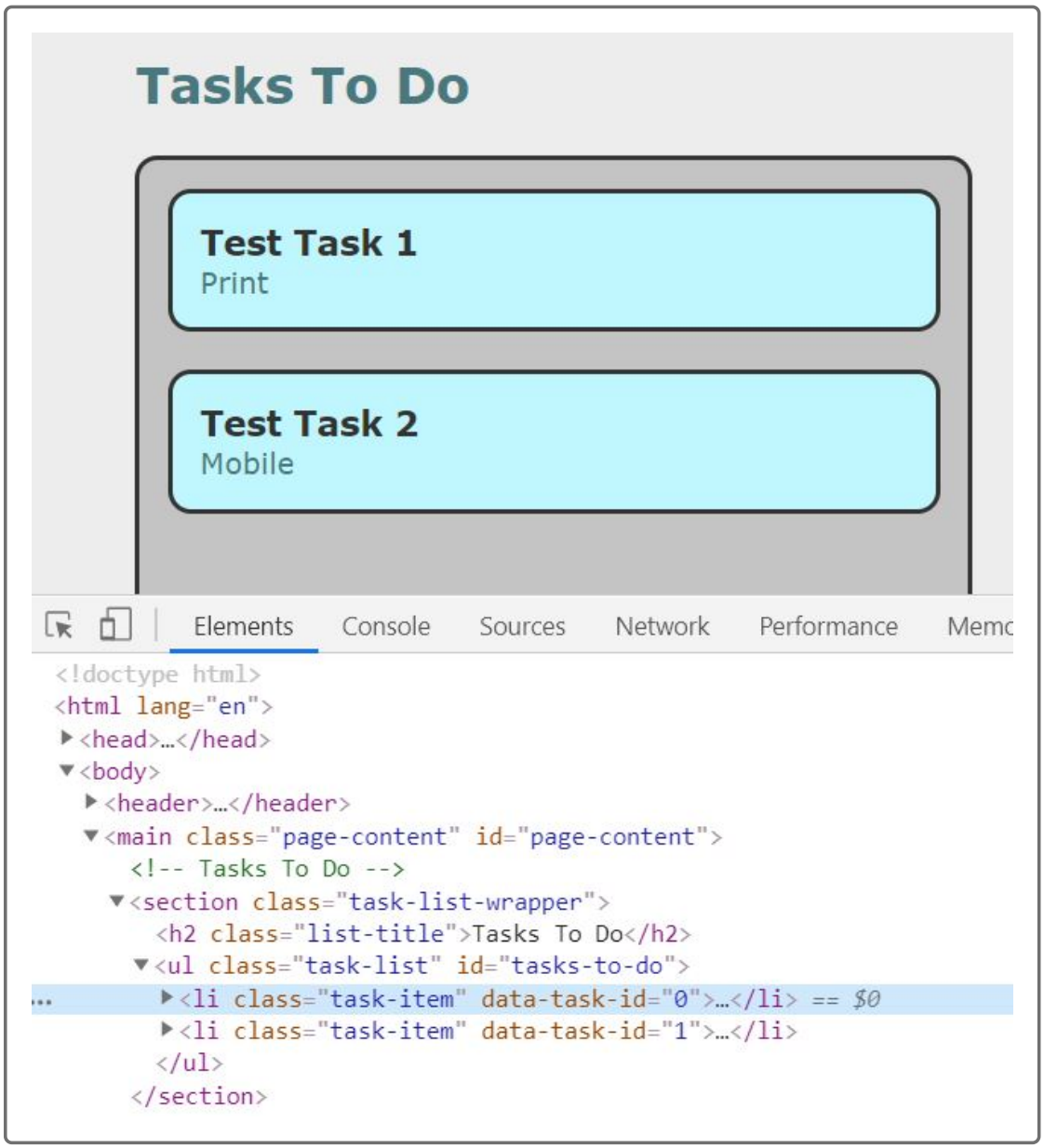
When the first task is created, what will the value of its `data-task-id` attribute be?

Zero, because the `taskIdCounter` starts at zero.

Hide Answer

At the bottom of the `createTaskEl()` function, we then increment the counter by one (`taskIdCounter++`) to keep the IDs unique.

Save `script.js` and refresh the browser. Make a few tasks, then use the DevTools to inspect the DOM. You'll know everything's working correctly if you can see `data-task-id` attributes in the Elements panel:



In the `createTaskEl()` function, we'll use this `taskIdCounter` variable to assign an ID to the current task being created. How would we attach the ID to an HTML element, though? There's always the `id` attribute, but we should be mindful that this attribute already serves a different purpose. The browser uses `id` to keep track of elements for CSS selectors, links, event listeners, etc. It wouldn't be appropriate to use it for a counter value.

That's where `data-*` attributes come in. Also known as **custom data attributes**, these allow developers to store extra information about an HTML element without conflicting with any of the built-in attributes.

Watch the following video for an example of how to use `data-*` attributes:

For Taskinator, we'll add a `data-task-id` attribute to each `<li>` element. In the `createTaskEl()` function, update the code as follows:

```
var createTaskEl = function(taskDataObj) {
  var listItemEl = document.createElement("li");
  listItemEl.className = "task-item";
```