Hands On: Completing Event Handling

Let's complete event handling in this lesson.

WE'LL COVER THE FOLLOWING

The previous exercise showed that you can easily change an element in the DOM. However, instead of setting the background color of headings to red, the `<div>` section adjacent with the `<h2>` should be hidden or shown again.

In the upcoming exercise, you will learn how to do it with the operations of the DOM.

EXERCISE 6-3: Completing event handling

To change the event handler method, follow these steps:

Step 1:

Switch back to the code editor below, and open **hideandseek.js**.

```
var titles = document.getElementsByTagName('h2');
for (var i = 0; i < titles.length; i++) {
  var title = titles[i];

  title.innerHTML = '<span class="mono">- </span>'
    + title.innerHTML;
  title.addEventListener('click', onClick, true);
}

function onClick(evt) {
  var headerClicked = evt.currentTarget;
  headerClicked.setAttribute("style",
    "background-color: red;");
}
```

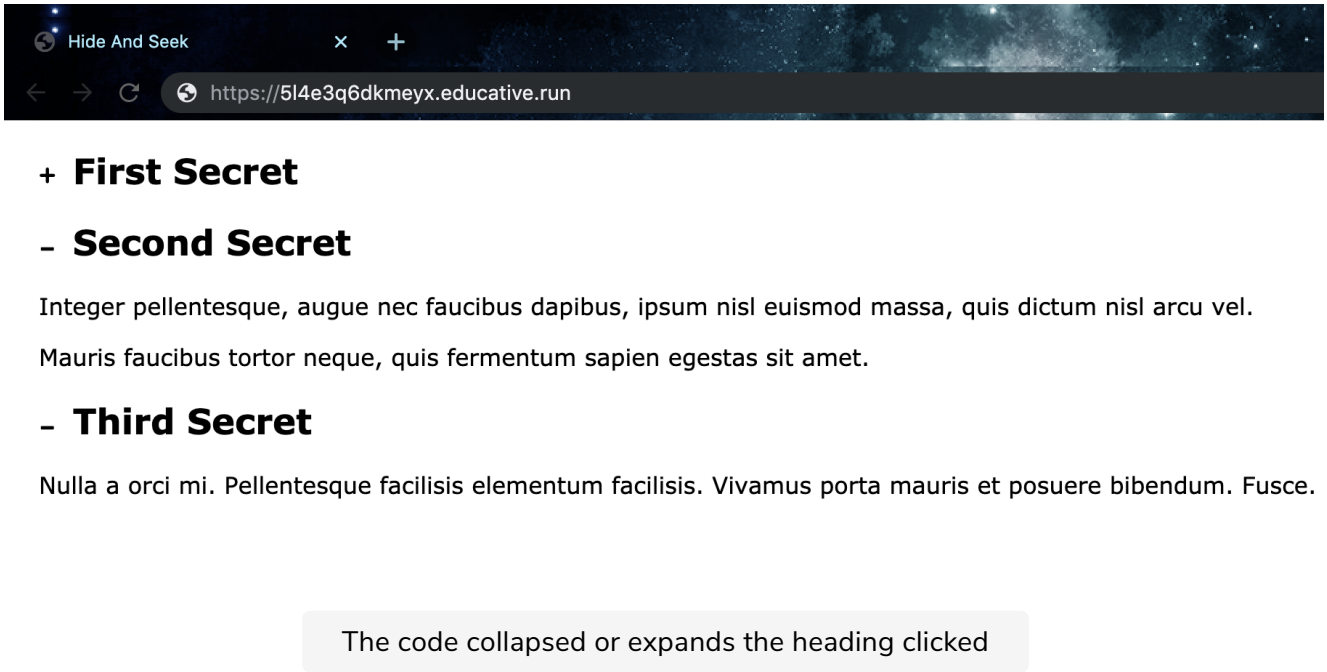
Step 2:

Remove the body of the `onClick()` function, and change it to this:

```
function onClick(evt) {
  var headerClicked = evt.currentTarget;
  var relatedDiv = headerClicked.nextElementSibling;
  var collapseMark = headerClicked.firstChild;
  var isCollapsed = collapseMark.innerText[0] == "+";
  collapseMark.innerText = isCollapsed ? "- " : "+ ";
  relatedDiv.setAttribute("style",
    isCollapsed ? "" : "display: none");
}
```

Step 3:

Take a look at the page in the browser. When you click any of the headings, the related details are now collapsed, as shown in the image below:



The code collapsed or expands the heading clicked

Step 4:

Step 4:

When the page appears, it would be better if all headings were collapsed by default. Add the highlighted lines to the `for` loop in **hideandseek.js**:

```
var titles = document.getElementsByTagName('h2');
for (var i = 0; i < titles.length; i++) {
  var title = titles[i];
  title.innerHTML = '<span class="mono">- </span>'
    + title.innerHTML;
  title.addEventListener('click', onClick, true);
  var relatedDiv = title.nextElementSibling;
  var collapseMark = title.firstChild;
  collapseMark.innerText = "+ ";
  relatedDiv.setAttribute("style", "display: none");
}
```

Step 5:

Turn to the browser again. This time all headings are collapsed. See if you can expand and collapse the headings.

Step 6:

If you take a look at the highlighted code lines in the previous step, you can observe that those lines carry out exactly the same operations as if you clicked the heading. Instead of repeating these operations, you can instruct the DOM to fire the click event. Change the highlighted lines in the previous step to these ones:

```
var evt = document.createEvent("mouseEvent");
evt.initEvent("click", true, false);
title.dispatchEvent(evt);
```

Step 7:

Check the page again. It will start and work exactly the same as in step 5. Close the browser.

HOW IT WORKS

In the first step you added this code to handle the click event of a heading:

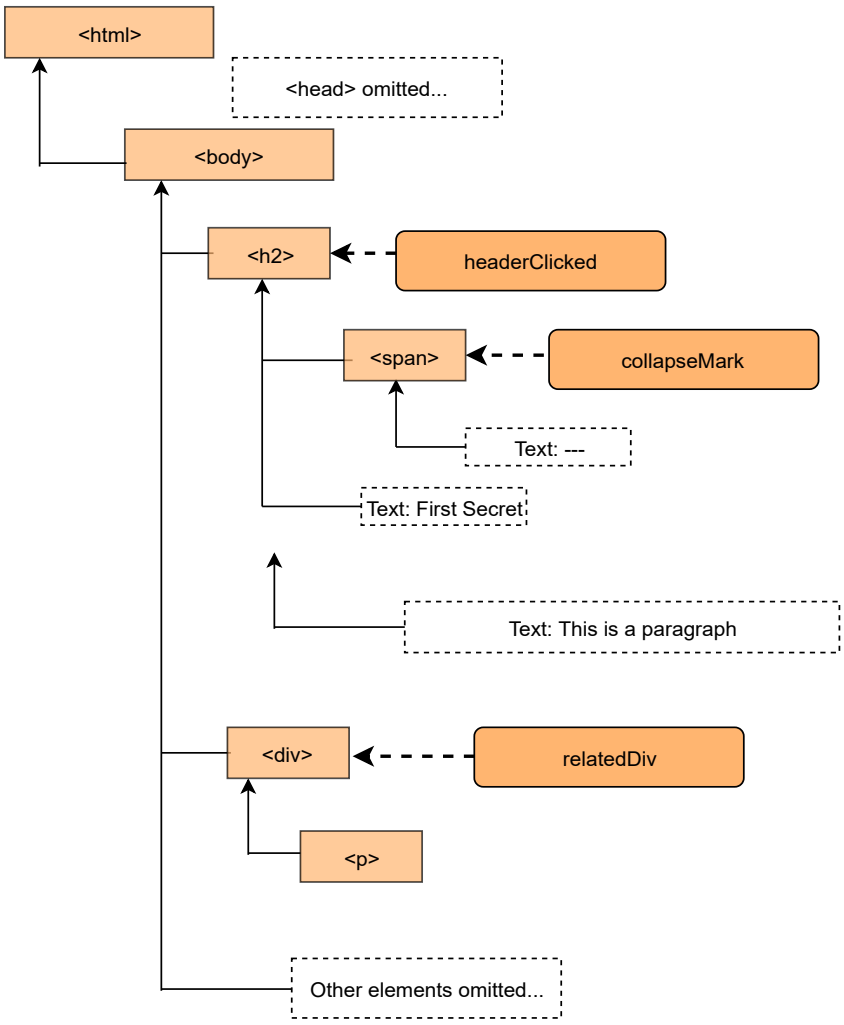
```
var headerClicked = evt.currentTarget;
```

```
var relatedDiv = headerClicked.nextElementSibling;
var collapseMark = headerClicked.firstChild;
var isCollapsed = collapseMark.innerText[0] == "+";

collapseMark.innerText = isCollapsed ? "- " : "+ ";
relatedDiv.setAttribute("style",
  isCollapsed ? "" : "display: none");
```

Line one stores the clicked element instance in the `headerClicked` variable. The `nextElementSibling` property evaluated on `headerClicked` (in **line two**) navigates to the subsequent `<div>` element directly following the `<h2>` represented by `headerClicked`. The `firstChild` property retrieves the first child element of the node it is evaluated on, so **line three** will retrieve the marker of the clicked heading.

Let's assume that the first `<h2>` element has been clicked. In this case, the variables will hold the DOM elements shown in the image below:



The document structure of the clicked heading

Line four sets the flag indicating whether the clicked heading is collapsed

Line four sets the flag indicating whether the clicked heading is collapsed. This flag is used in **line five** to switch the marker between the plus sign and the dash, which represent the collapsed and expanded states, respectively.

Line six uses the same flag to set the style attribute to `“display: node”` (it hides the `<div>` tag), or to an empty string (it shows the `<div>` tag).

The code added in **step four** uses the same navigation logic to set the default state of headings when the page is being loaded.

In **step six** you used this code to fire a click event:

```
var evt = document.createEvent("mouseEvent");
evt.initEvent("click", true, false);
title.dispatchEvent(evt);
```

When the user clicks an element on the page, the browser creates an event object and sends it to the event handler method set for the `onclick` event.

In **line one**, you use the `createEvent()` function of the document object to create a generic mouse event. The `initEvent()` function invoked on this new event object, with `“click”` as its first argument, specifies that this event should be triggered as if the mouse button has been clicked.

The other arguments of the method are not relevant in this context. **Line three** sends this event to the clicked heading with the `dispatchEvent()` method.

As the final result, these three lines force the click event to fire on the heading, just as if the user clicked the corresponding `<h2>` element.

You can find the completed implementation below:

Complete implementation

```
var titles = document.getElementsByTagName('h2');
for (var i = 0; i < titles.length; i++) {
  var title = titles[i];
  title.innerHTML = '<span class="mono">- </span>'
    + title.innerHTML;
  title.addEventListener('click', onClick, true);
  var evt = document.createEvent("mouseEvent");
  evt.initEvent("click", true, false);
  title.dispatchEvent(evt);
```

```
}

function onClick(evt) {

  var headerClicked = evt.currentTarget;
  var relatedDiv = headerClicked.nextElementSibling;
  var collapseMark = headerClicked.firstChild;
  var isCollapsed = collapseMark.innerText[0] == "+";
  collapseMark.innerText = isCollapsed ? "- " : "+ ";
  relatedDiv.setAttribute("style",
    isCollapsed ? "" : "display: none");
}
```

These exercises gave you a brief overview of the functionality covered by the Document Object Model.

It is time to dive a bit deeper into the details from the *next lesson*.