Debugging JavaScript

In this lesson, we will commence our debugging journey in JavaScript.
Let's begin!

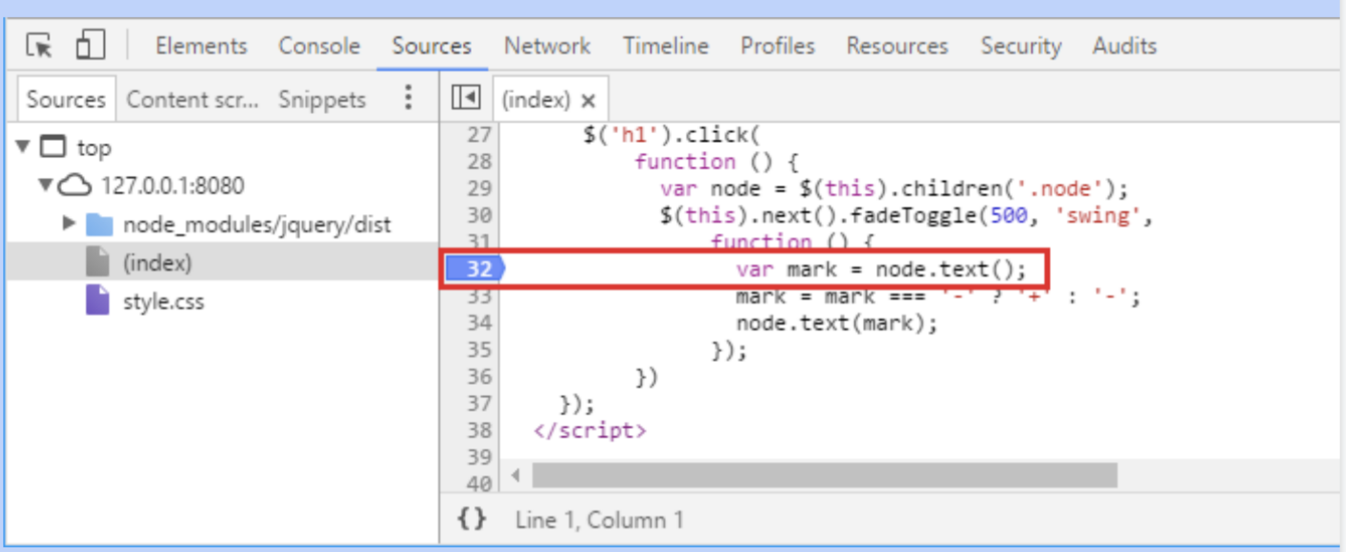
WE'LL COVER THE FOLLOWING

Debugging JavaScript with Chrome dev tools

To discover the simplicity of JavaScript debugging in Chrome, follow these steps:

Step 1:

Display your page in Chrome (the page you created in the previous exercise), and turn on the **Developer Tools** 🛠️ pane (with `F12` (press `fn` key on Mac) or `Ctrl+Shift+I` or `Cmd+Opt+J` for Mac). Select the Source tab and in the source file hierarchy, click (index). To the right from the source file hierarchy, the **index.html** file is displayed. Scroll down to the `var` mark line, and click the line number to set a breakpoint, as shown in the figure below.

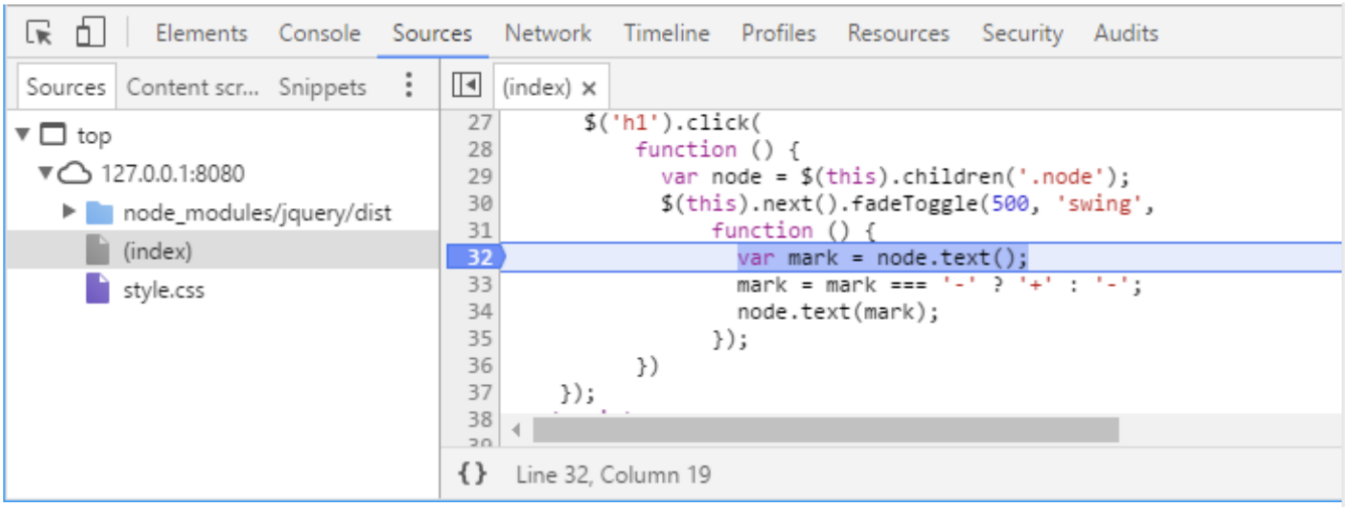


```
  $('h1').click(
    function () {
     var node = $(this).children('.node');
     $(this).next().fadeToggle(500, 'swing',
       function () {
        var mark = node.text();
        mark = mark === '-' ? '+' : '-';
        node.text(mark);
       });
    })
 });
</script>
```

Setting a breakpoint on JavaScript code

Step 2:

Click one of the first-level headings. As soon as the nested content is faded out, Chrome suspends the code execution because the execution flow reaches the breakpoint you set. This is indicated by the border around the shaded code line, as shown in the figure below.

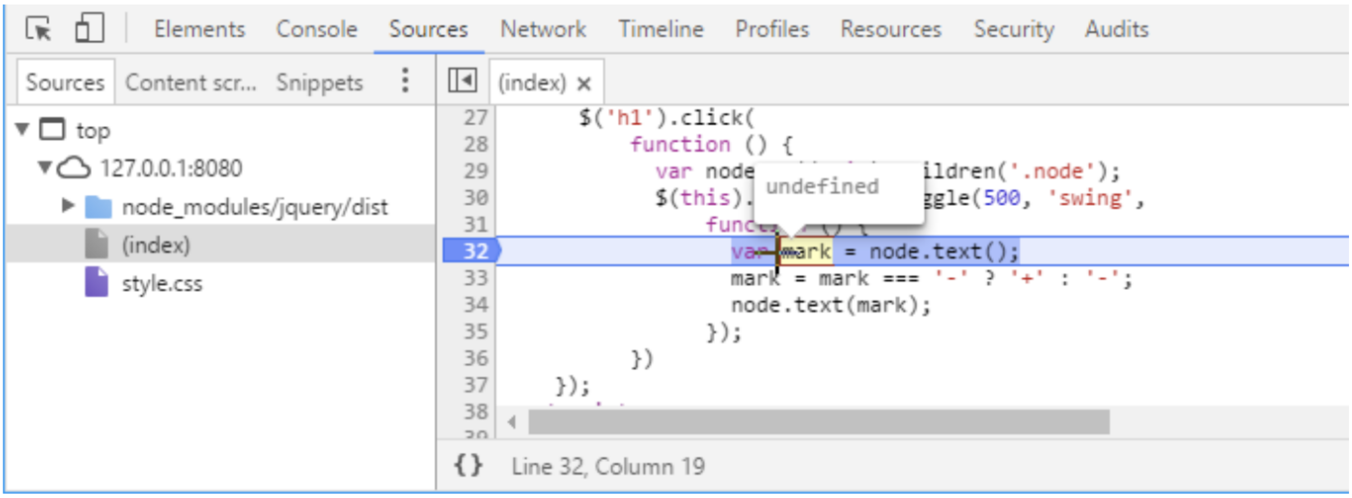


```
  $('h1').click(
    function () {
     var node = $(this).children('.node');
     $(this).next().fadeToggle(500, 'swing',
       function () {
        var mark = node.text();
        mark = mark === '-' ? '+' : '-';
        node.text(mark);
       });
    })
 });
```

The execution flows reaches the breakpoint

Step 3:

Move the mouse over the mark variable name in the highlighted line. Chrome shows in a tooltip that its value is undefined (Figure below) because the breakpoint stopped right before the value of the variable has been set.

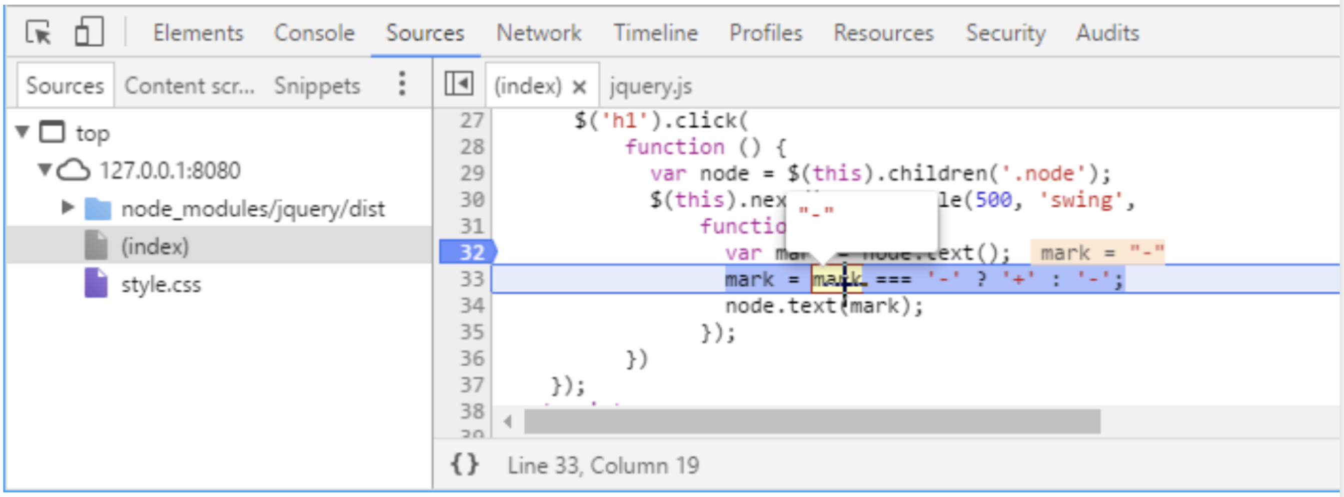


```
        $('h1').click(
            function () {
                var node          ildren('.node');
                $(this).          ggle(500, 'swing',
                    func
                    var mark = node.text();
                    mark = mark === '-' ? '+' : '-';
                    node.text(mark);
                });
            })
        });
```

The value of mark is undefined.

Step 4:

Press `F10` (press `fn` key on Mac) to execute the current line. Now the debugger stops at the next code line. Move the mouse over the mark variable name, and you can see, its initial value is `–` (Figure below), indicating the expanded state of the first-level heading.



```
        $('h1').click(
            function () {
                var node = $(this).children('.node');
                $(this).nex          le(500, 'swing',
                    functio
                    var mar          ext();  mark = "-"
                    mark = mark === '-' ? '+' : '-';
                    node.text(mark);
                });
            })
        });
```

The value of mark is set to "–"

Step 5:

Press `F10` again and check the value of mark again. Now, it is toggled from `–` to `+`.

Amazing work! Now that we have had our short tour of the Holy Trinity that is HTML, CSS, and JavaScript, let's summarize what we've learned in the *next lesson*.

lesson.

See you there! :)