Hands-on Lab: Troubleshooting Common Errors with Chrome DevTools

Skills Network

Estimated time needed: 45 minutes

In this lab, you will learn how to use Chrome DevTools by working directly in the Elements, Console, and Sources panels.

As you are already aware:

> Chrome DevTools is a set of web developer tools that are built directly into the Google Chrome browser. DevTools allow you to edit pages on the fly and quickly diagnose problems, allowing you to build better websites faster.

Objectives:

After completing this lab, you will be able to:

- Find, edit, and debug nodes
- Use the Console to debug JavaScript
- Use breakpoints in JavaScript code

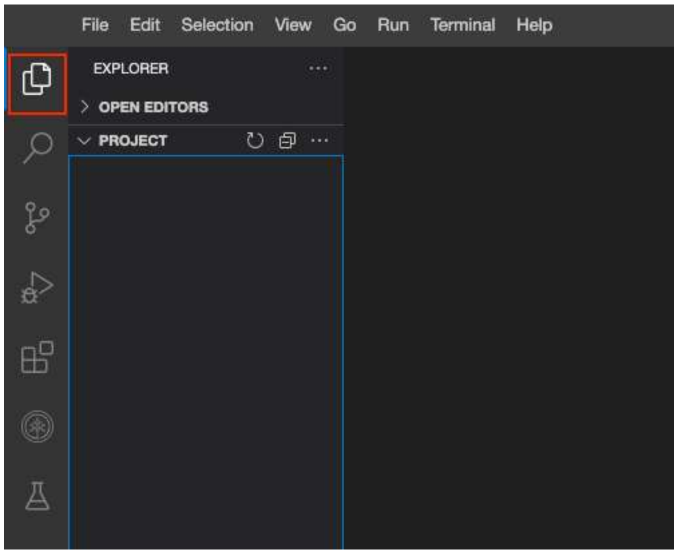
Pre-requisite:

- This lab expects you to debug the application in Chrome Web Browser. But you can use other browsers too, although you will see similar functionality but menu options and tab names will be different.

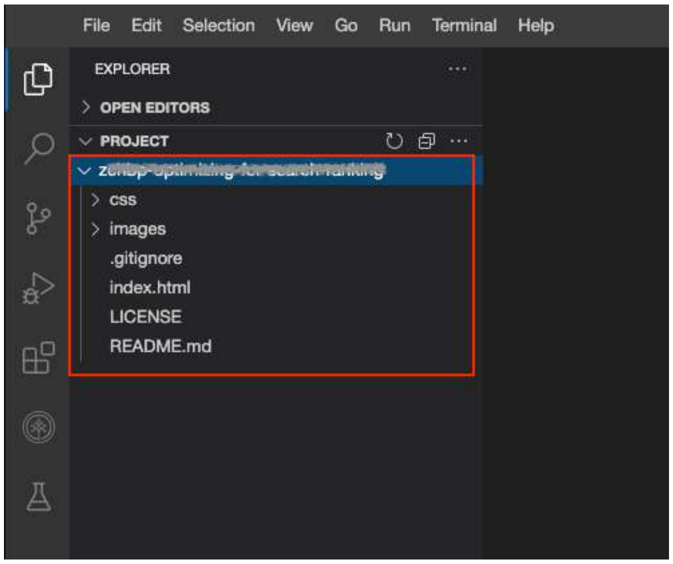
Working with files in Cloud IDE

If you are new to Cloud IDE, this section will show you how to create and edit files, which are part of your projet, in Cloud IDE.

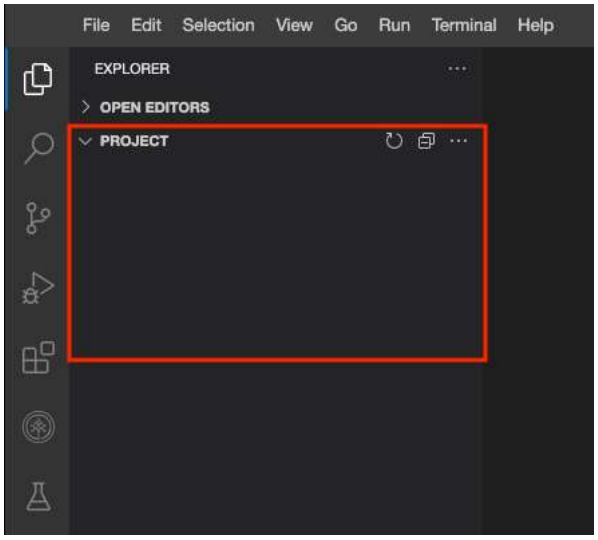
To view your files and directories inside Cloud IDE, click on this files icon to reveal it.



If you have cloned (using `git clone` command) boilerplate/starting code, then it will look like below:

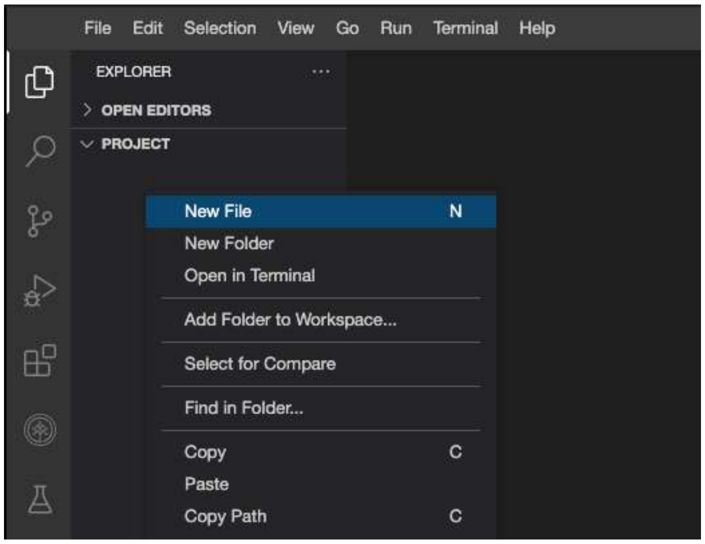


Otherwise a blank project looks like this:



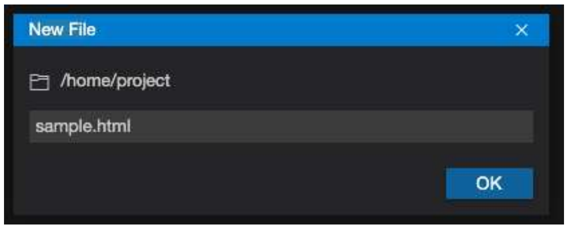
Create a new file

You can right-click and select the New File option to create a file in your project.

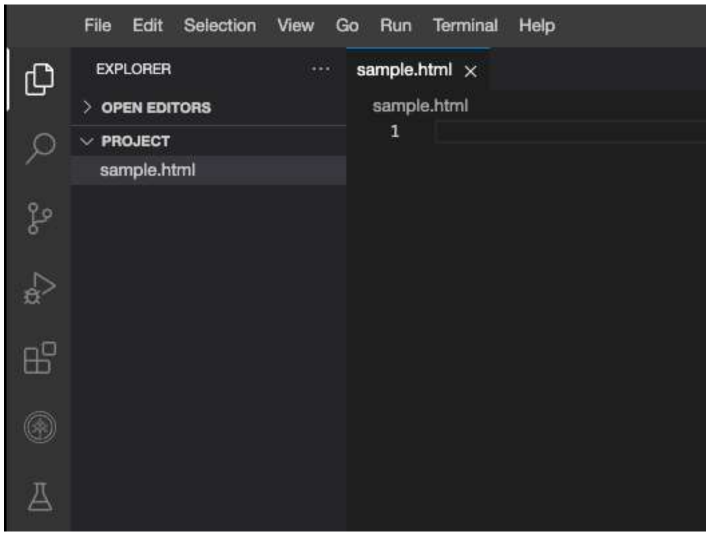


You can also choose `File -> New File` to do the same.

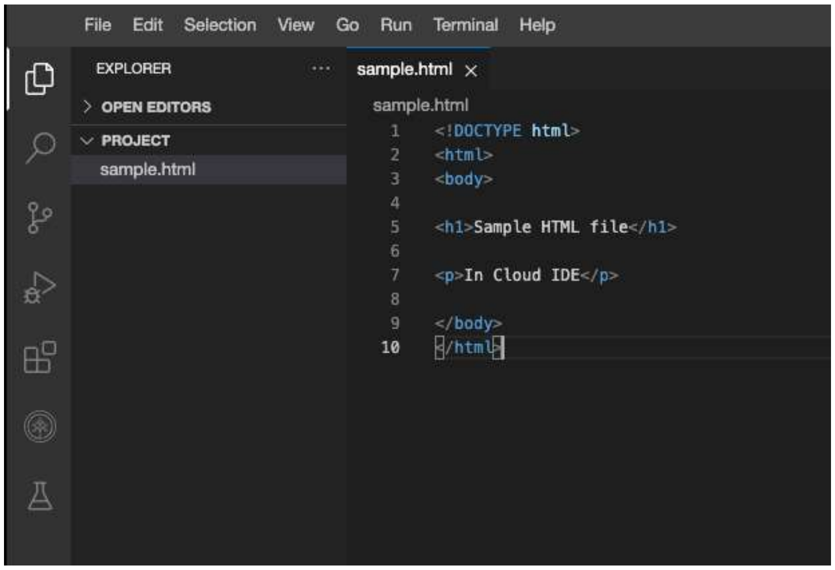
It will then prompt you to enter name of this new file. In the example below, we are creating `sample.html`.



Clicking on the file name `sample.html` in the directory structure will open the file on the right pane. You can create all different types of files; for example `FILE_NAME.js` for JavaScript file.



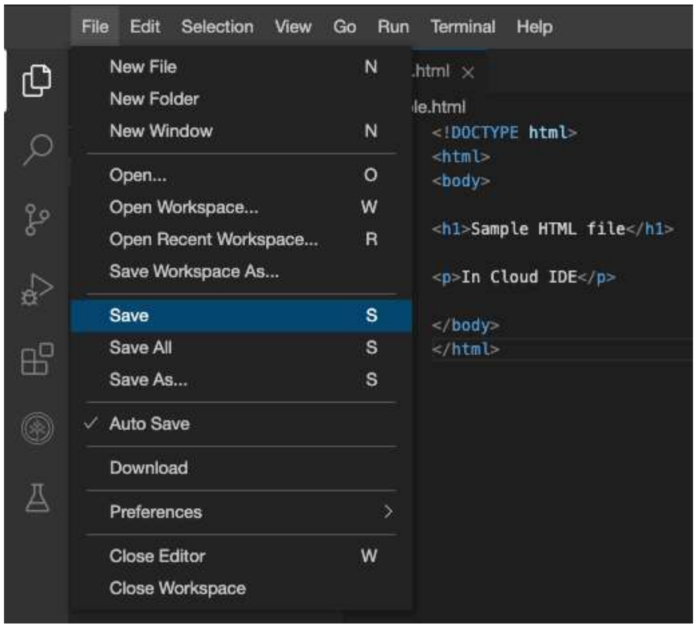
In the example, we just pasted some basic html code and then saved the file.



```html
<!DOCTYPE html>
<html>
<body>

<h1>Sample HTML file</h1>

<p>In Cloud IDE</p>

</body>
</html>
```

And saving it by:

- Going in the menu.
- Press ⌘ `Cmd + S` on Mac or `CTRL + S` on Windows.
- Or it can Autosave it for you too.



```
<!DOCTYPE html>
<html>
<body>

<h1>Sample HTML file</h1>

<p>In Cloud IDE</p>

</body>
</html>
```

Set-up: Create HTML page

Create a file called `index.html` in the Project section of Explorer and paste the following code into it. This will assist you in completing the lab.

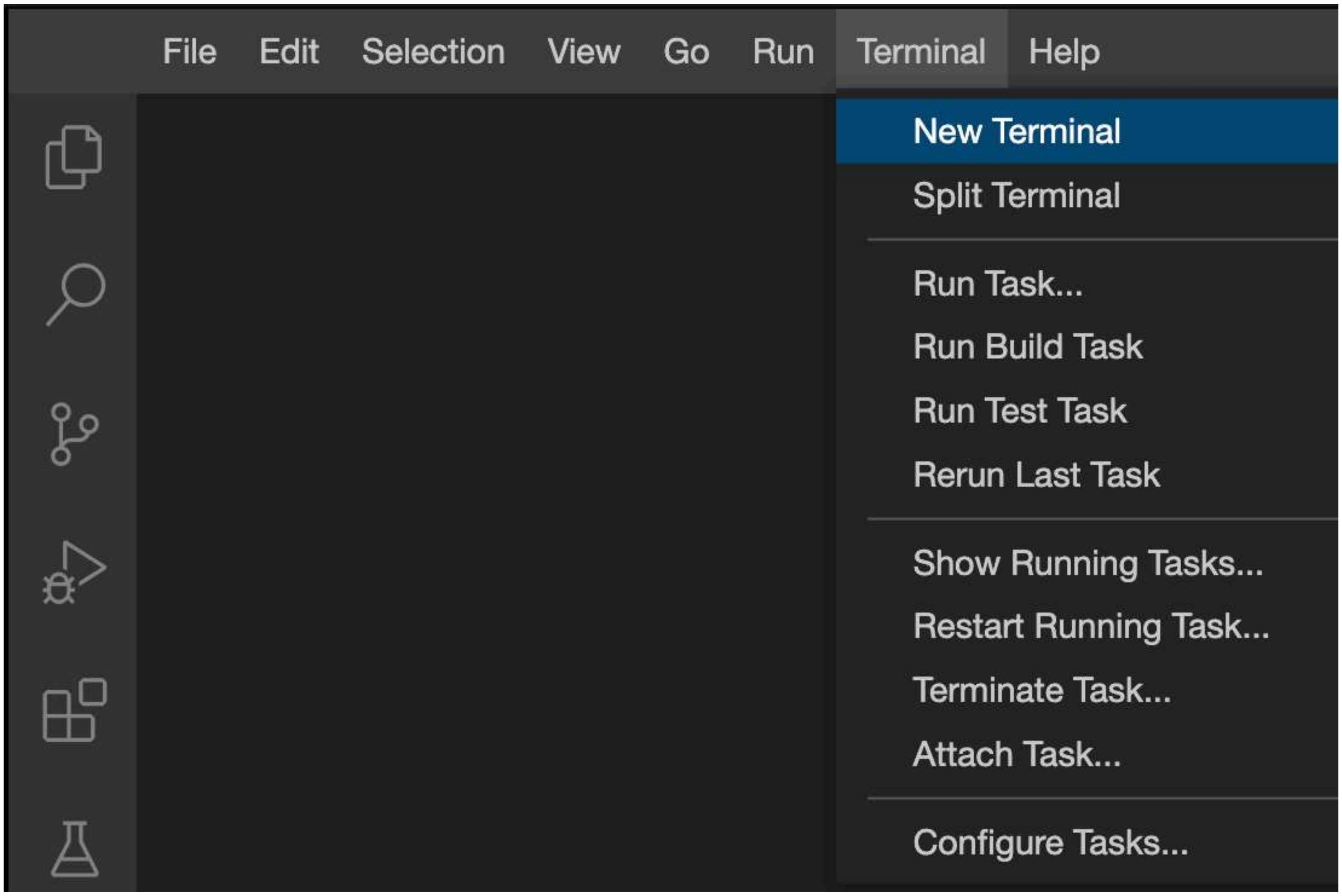
```
<!DOCTYPE html>
<html>
  <head>
    <title>Working with Chrome DevTools</title>
  </head>
  <body>
    <h1>Welcome to Chrome DevTools Lab</h1>
    <ul>
      <li>Michelangelo</li>
      <li>Raphael</li>
    </ul>
    <p style="white-space: pre">
      "To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
And by opposing end them? To die: to sleep;
No more; and by a sleep to say we end
The heart-ache and the thousand natural shocks
That flesh is heir to, 'tis a consummation
Devoutly to be wish'd. To die, to sleep;
To sleep: perchance to dream: ay, there's the rub;
For in that sleep of death what dreams may come
```

```
When we have shuffled off this mortal coil,
Must give us pause: there's the respect
That makes calamity of so long life;" - Hamlet, Act III, Scene I
    </p>
    <ul>
      <li>Eggs</li>
      <li>Milk</li>
      <li>Bread</li>
      <li>Cereal</li>
    </ul>
  </body>
</html>
```

Copied!

Test the HTML page

1. Open a terminal window by using the menu in the editor: **Terminal > New Terminal**.



2. Run the following command on the terminal to host your web page.

```
python3 -m http.server
```

Copied!

Launch Application

3. You can copy the URL and paste it into your Chrome browser.

Exercise 1: Open Chrome DevTools

In this Lab, we will look at the most common panels in Chrome DevTools.

But first, because different users require quick access to different parts of the DevTools UI, there are numerous ways to launch DevTools.

- To access the DOM or CSS, right-click an element on the page and select **Inspect** to open the Elements panel. You can also use `Command+Option+C (Mac)` or `Control+Shift+C (Windows, Linux, ChromeOS)`.
- To access the Console panel and view logged messages or run JavaScript, press `Command+Option+J (Mac)` or `Control+Shift+J (Windows, Linux, ChromeOS)`.

DevTools panels

1. Elements Panel: This panel allows developers to inspect and modify a web page's HTML and CSS. It displays the DOM tree in a hierarchical manner, allowing developers to easily locate and manipulate specific elements.
2. Console Panel: The console panel is an interactive JavaScript console that allows developers to run code and see the results in real-time. It can be used to debug and troubleshoot JavaScript errors and test and experiment with code.
3. Sources Panel: The sources panel allows developers to view and debug a web page's JavaScript code. It has features like breakpoints, step-through debugging, and profiling that can be used to find and fix bugs in the code.
4. Network Panel: The network panel displays detailed information about a web page's network requests, such as HTTP headers, request and response bodies, and timings. This data can be used to improve the performance of a web application by identifying and resolving network bottlenecks.
5. Performance Panel: The performance panel provides information about a web page's performance, including metrics such as page load time, frame rate, and memory usage. It can be used to identify performance bottlenecks and optimize web application performance.

Exercise 2: Inspect a node

Let's examine a node in our HTML page and see where it is in the DOM Tree.

Right-click "**Michelangelo**" and select **Inspect**.

The DevTools Elements panel opens with `<li>Michelangelo</li>` highlighted in the DOM Tree.

In the top-left corner of DevTools, click the **Inspect** icon.

Then select the "**Raphael**" text. In the DOM Tree, `<li>Raphael</li>` is now highlighted.

Exercise 3: Search and edit

We can now use Chrome DevTools to search for and edit nodes. But keep in mind that any changes made here are not saved and are not updated on the server. Everything takes place entirely within your browser.

Search for a node

Go to the Elements panel and press `Control+F (Windows)` or `Command+F (Mac)`. The Search bar appears at the bottom of the Elements tab. Type "**not to be**" into it. The first sentence of Hamlet will be highlighted.

Edit content

Right click "**Bread**", select the `li` in Elements, and then press the delete button on your keyboard.

Exercise 4: Work with Console

Let's now discuss how to use the Console. As you are aware:

> The console panel includes an interactive JavaScript console that allows developers to execute code and view the results in real time. It can be used to debug and troubleshoot JavaScript errors and test and experiment with the code.

Begin by adding some JavaScript to the HTML page.

Create a scripts section after the last closing `ul` as shown below:

```
<script type="text/javascript">console.log("Hello Console!");</script>
```

Copied!

The output can be seen as a message printed in Console "Hello Console!".

Exercise 5: Find a node and debug it in Console

Let's log some more in our Console.

First, we will print the text content of the `h1` element. Then we will check to see if an `h2` is present.

Logging in Console is used for two reasons:

- To make sure that code is executed in the correct order.
- To inspect the values of variables at a certain moment in time.

```
<script type="text/javascript">
```

```
  console.log("Hello Console!");
  const h1 = document.querySelector("h1");
  console.log(h1.textContent);
  console.assert(document.querySelector("h2"), "h2 not found!");
</script>
```

Copied!

The `console.assert()` method writes an error message to the console if the assertion is false. If the assertion is true, nothing happens.

Exercise 6: Sources panel (Part 1)

We will now make use of the Sources panel.

Add the following HTML before the `script` tag.

```
<div>
  <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 onClick="calculateSum()">Add Number 1 and Number 2</button>
  <p id="calculatorOutput"></p>
</div>
```

Copied!

Next, add the following Javascript at the end of the `script` tag.

```
function calculateSum() {
  var inputs = document.querySelectorAll("input");
  var addend1 = inputs[0].value;
  var addend2 = inputs[1].value;
  var sum = addend1 + addend2;
  var label = document.getElementById("calculatorOutput");
  label.textContent = addend1 + " + " + addend2 + " = " + sum;
}
```

Copied!

Enter two numbers into the box and press the button. We see the wrong answer!

Exercise 7: Sources panel Part 2

We will now use Chrome DevTools to debug and resolve this issue.

In DevTools, go to the Sources panel and set a breakpoint on line: `var sum = addend1 + addend2;` by clicking the line number on the left.

On the page, press the button again to add the numbers together (and call our function).

This time, your code will stop on that line. Hover over `addend1` and `addend2` and you will notice that they are of type `string` and not `number`.

You can also look in the Scope section of Sources and see the two variables there.

Let's fix our code by changing the line to:

```
var sum = parseInt(addend1) + parseInt(addend2);
```

Copied!

And debug again!

Congratulations! You have completed the lab for Chrome DevTools.

Summary:

In this lab, you used Chrome DevTools to inspect HTML nodes, edit them, debug JavaScript, and use the Console.

Author(s)

Muhammad Yahya

Changelog

Date	Version	Changed by	Change Description
2023-03-27	0.3	Steve Hord	QA pass minor edits
2023-03-17	0.2	Steve Hord	QA pass with edits
05-03-2023	0.1	Muhammad Yahya	Initial version created