Javascript Exception Handling Lab

Exception and Error handling will be covered along with debugging code in a subsequent lecture. Before covering this topic in a lecture, this lab will be used to introduce an example of an exception and how to handle it.

When executing code, an exception maybe encountered. Often the exception is not visible to the user. The only way they know something is wrong is that what they expect to happen does not happen. Sometimes they may not know anything wrong has occurred. For example, a web page loads, everything seems ok, but an exception could have occurred which the user is not aware of which will have an impact on other functionality on the web page.

Typically, exceptions are of interest to developers. Developers need to use a development tool, such as those available with browsers like Chrome or IE. The developer tools will highlight any exceptions.

In some circumstances exceptions may not be obvious, even when using a development tool. Even when they are highlighted in the development tool, the information on the exception can be vague. Though most modern versions of browsers have development tools which provide ample information pointing out the type of error and where exactly in the code the error is encountered.

In some circumstances an exception may not occur and can be missed in testing or may be known about, but only occurs in exceptional circumstances (e.g. if 10 million users visit a web page in one day). Typically, when an exception occurs if affects all other functionality and the web page becomes unusable.

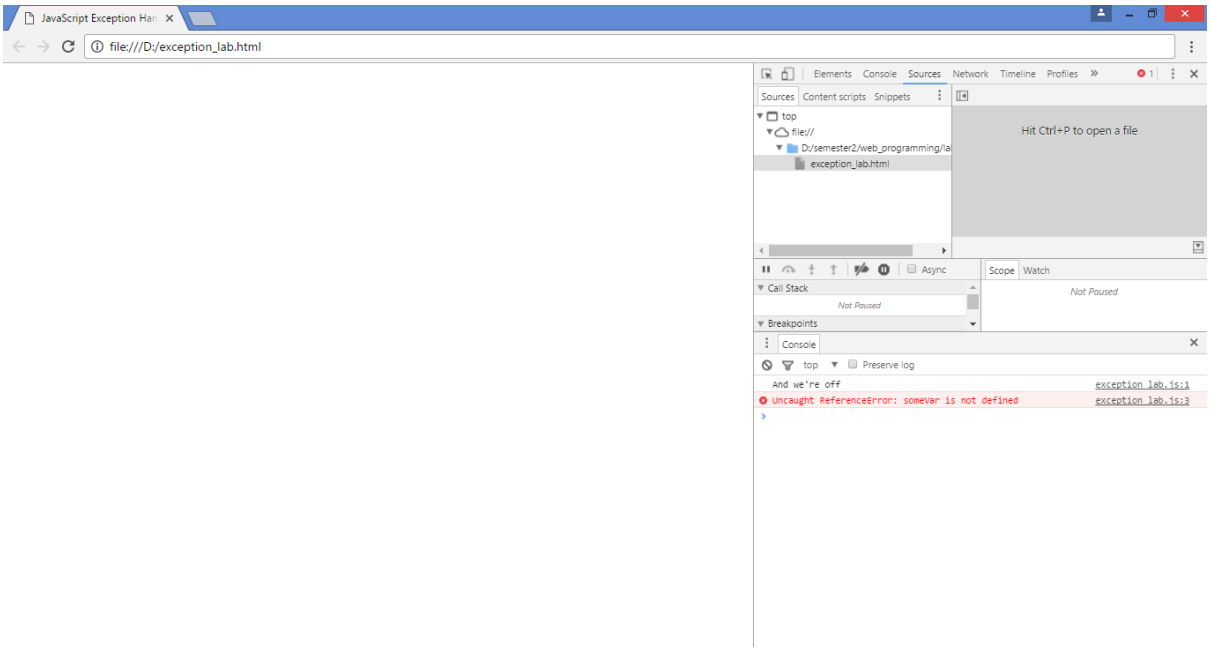
There is a way to handle exceptions, so that if they do occur, the developer tool does not report them in their console, and the web page application is still usable, though maybe not as intended. However, it means exceptions are handled gracefully and a workaround is in place so that the application can still be used.

In JavaScript, try, catch and finally is used to handle exceptions:

```
try {
        // try to execute this code
} catch (exception) {
        // if there is an exception, run this code
} finally {
        // this code always gets executed
}
```

Exercise 1 – View Exception in Development Tool

1. Download the zip file from moodle which is included along with this exercise.
2. Unzip the file, and open exception_lab.html in a web browser. The page will be blank.
3. Open the browser development tool and check the console for messages. In Chrome, you would see the following:



Note: If you do not know how to access the development tool in Chrome, or wish to use another browser, just ask the lecturer.

4. The reference error reported by the console indicates that a variable is not defined. Open exception_lab.js and see line 3, or use the link from the console.

Exercise 2 – Handle the exception from exercise one

1. Wrap the line of code of code in exception_lab.js in a try block:

```
try {
  someVar;
}
```

2. Then add a catch block, to catch the error:

```
catch(e) {
  //add code here
}
```

3. Add code to the catch block to update the <p> element in exception_lab.html with text indicating there is an error and include the exception passed to the catch block as part of the text.
4. Then add a finally block which just includes an alert which contains text indicating the code in the finally block is run:

```
try {
  someVar;
} catch(e) {
  //Update paragraph on web page with exception details.
} finally {
  //alert that code in finally block executed
}
```

5. Reload exception_lab.html in the browser.
6. You should see the paragraph is updated from the code in the catch block.
7. Also, check the console on the browser developer tool to ensure no exception is recorded, as it is now handled by the code.
8. Modify the code in the try block to fix the exception.
9. Refresh the browser and see what happens (no paragraph and just an alert).