JavaScript Essentials

Debugging JavaScript

Table of Contents

Lesson Objectives

- Gain the ability and confidence to start fixing problems in your own code
- Get familiar with Browser DevTools and able to use it to start troubleshooting problem
- Inspect and manipulate runtime flows with Console

Section 1

Types of Error

Type of Error

- Generally speaking, when you do something wrong in code, there are two main types of error that you'll come across:
  - **Syntax errors**: These are spelling errors in your code that actually cause the program not to run at all, or stop working part way through — you will usually be provided with some error messages too. These are usually okay to fix, as long as you are familiar with the right tools and know what the error messages mean!
  - **Logic errors**: These are errors where the syntax is actually correct but the code is not what you intended it to be, meaning that program runs successfully but gives incorrect results. These are often harder to fix than syntax errors, as there usually isn't an error message to direct you to the source of the error.

Type of Error – Summary

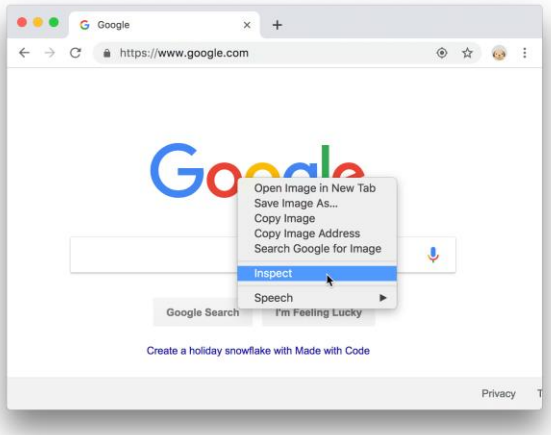
- There are two main types of error that you'll come across: syntax error and logic error
- **Syntax error**: spelling errors in your code that actually cause the program not to run at all
- **Logic error**: syntax is actually correct but the code is not what you intended it to be

Section 2

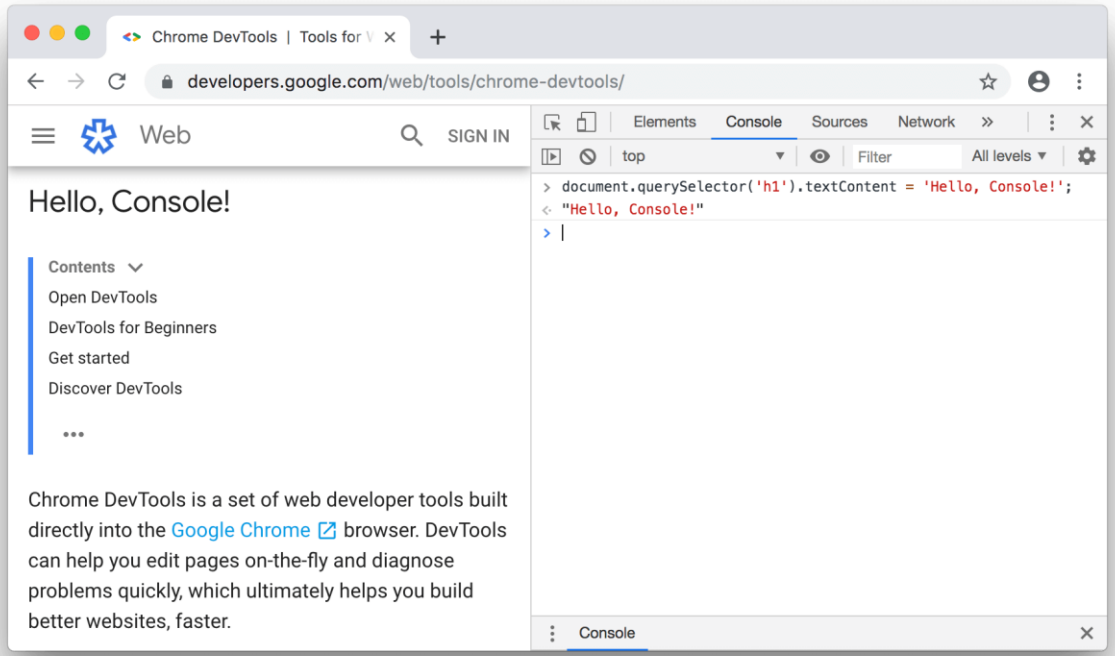
Browser DevTools

Browser DevTools

- Chrome DevTools is a set of web developer tools built directly into the Google Chrome browser. DevTools can help you edit pages on-the-fly and diagnose problems quickly, which ultimately helps you build better websites, faster.



Browser DevTools



Browser DevTools - Overview

- DevTools is most important tools for every Front-end Developer
- DevTools can help you edit pages on-the-fly and diagnose problems quickly, which ultimately helps you build better websites, faster.
- Use DevTools wisely will improve you productively and save you hundred of hour

Section 3

An erroneous example

An erroneous example

- To get started, let's return to our number guessing game — except this time we'll be exploring a version that has some deliberate errors introduced. Go to Github and make yourself a local copy of number-game-errors.html (see it running live here).
  - To get started, open the local copy inside your favorite text editor, and your browser.
  - Try playing the game — you'll notice that when you press the "Submit guess" button, it doesn't work

Events Object - Overview

- **Event object** it is automatically passed to event handlers to provide extra features and information
- The target property of the event object is always a reference to the element that the event has just occurred upon
- e.target is incredibly useful when you want to set the same event handler on multiple elements and do something to all of them when an event occurs on them
- Use event.preventDefault() method to prevent an event from doing what it does by default
- Modern Browser supports Event capturing and Event bubbling mode (default)
- Take advantage of **Event Delegation** to write less code but do more task

Section 4

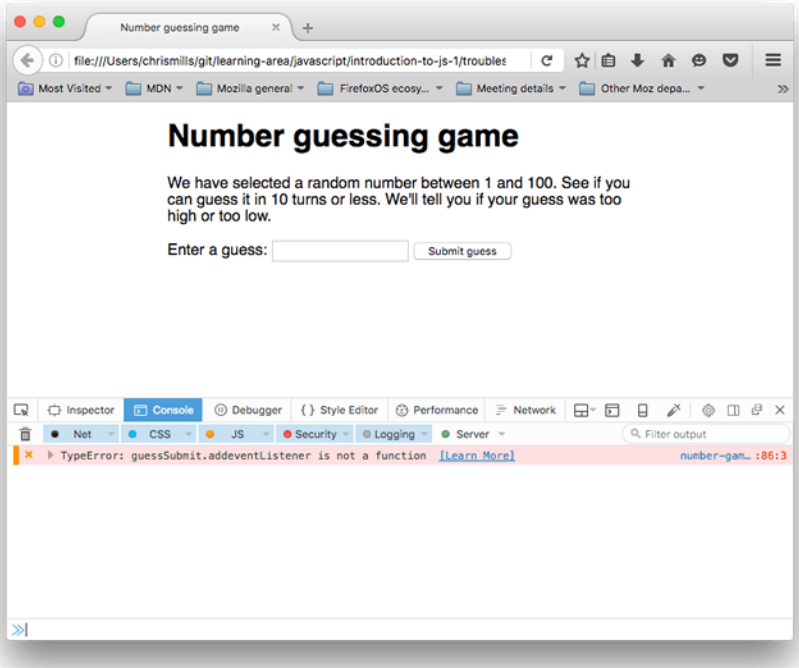
Fixing syntax errors

Fixing syntax errors – Round 1

- Earlier on in the course we got you to type some simple JavaScript commands into the developer tools JavaScript console (if you can't remember how to open this in your browser, follow the previous link to find out how).
- What's even more useful is that the console gives you error messages whenever a syntax error exists inside the JavaScript being fed into the browser's JavaScript engine.
- Now let's go hunting.

Fixing syntax errors – Round 1

- Go to the tab that you've got number-game-errors.html open in, and open your JavaScript console. You should see an error message along the following lines:



Fixing syntax errors – Round 1

- This is a pretty easy error to track down, and the browser gives you several useful bits of information to help you out (the screenshot above is from Firefox, but other browsers provide similar information). From left to right, we've got:
  - A red "x" to indicate that this is an error.
  - An error message to indicate what's gone wrong: "TypeError: guessSubmit.addeventListener is not a function"
  - A "Learn More" link that links through to an MDN page that explains what this error means in greater detail.
  - The name of the JavaScript file, which links through to the Debugger tab of the developer tools. If you follow this link, you'll see the exact line where the error is highlighted.
  - The line number where the error is, and the character number in that line where the error is first seen. In this case, we've got line 86, character number 3.

Fixing syntax errors – Round 1

- If we look at line 86 in our code editor, we'll find this line:

```
1 | guessSubmit.addeventListener('click', checkGuess);
```

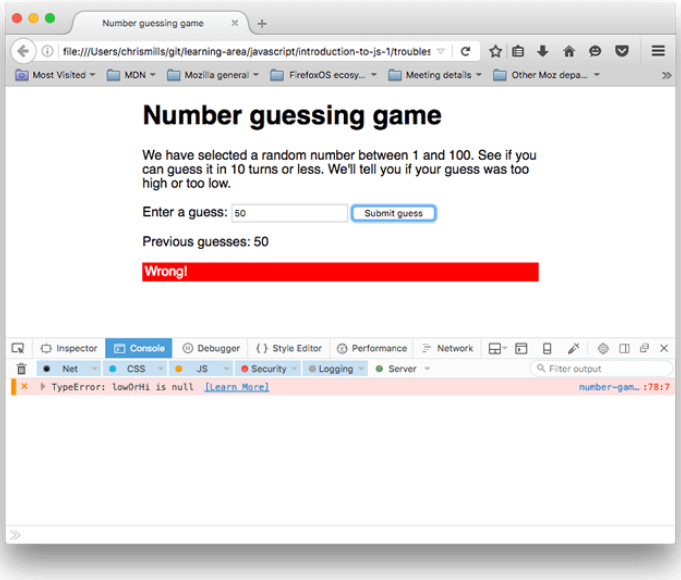
- The error message says "guessSubmit.addeventListener is not a function", which means that the function we're calling is not recognized by the JavaScript interpreter.
- Often, this error message actually means that we've spelled something wrong.
- If you are not sure of the correct spelling of a piece of syntax, it is often good to look up the feature on MDN.
- The best way to do this currently is to search for "mdn *name-of-feature*" with your favorite search engine.
- Here's a shortcut to save you some time in this instance: addEventListener().

Fixing syntax errors – Round 1

- So, looking at this page, the error appears to be that we've spelled the function name wrong! Remember that JavaScript is case sensitive, so any slight difference in spelling or casing will cause an error.
  Changing addeventListener to addEventListener should fix this. Do this now.

Fixing syntax errors – Round 2

- Save your page and refresh, and you should see the error has gone.
- Now if you try to enter a guess and press the Submit guess button, you'll see ... another error!



Fixing syntax errors – Round 2

- This time the error being reported is "TypeError: lowOrHi is null", on line 78.
- Have a look at line 78, and you'll see the following code:

```
1 | lowOrHi.textContent = 'Last guess was too high!';
```

- This line is trying to set the textContent property of the lowOrHi constant to a text string, but it's not working because lowOrHi does not contain what it's supposed to. Let's see why this is — try searching for other instances of lowOrHi in the code. The earliest instance you'll find in the JavaScript is on line 48:

```
1 | const lowOrHi = document.querySelector('lowOrHi');
```

Fixing syntax errors – Round 2

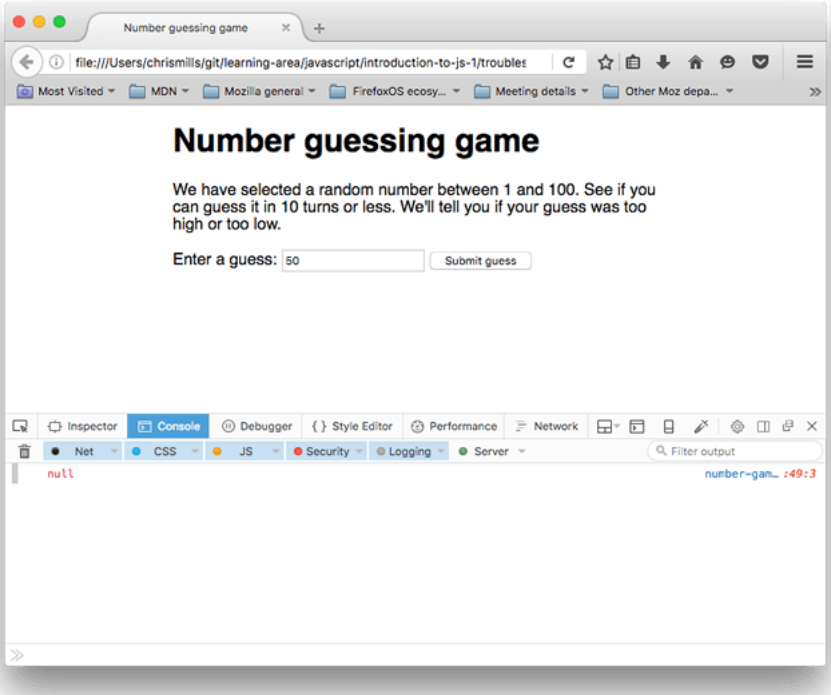
- At this point we are trying to make the variable contain a reference to an element in the document's HTML. Let's check whether the value is null after this line has been run. Add the following code on line 49:

```
1 | console.log(lowOrHi);
```

- **Note**: console.log() is a really useful debugging function that prints a value to the console. So it will print the value of lowOrHi to the console as soon as we have tried to set it in line 48.

Fixing syntax errors – Round 2

- Save and refesh, and you should now see the console.log() result in your console



Fixing syntax errors – Round 2

- Sure enough, lowOrHi's value is null at this point, so there is definitely a problem with line 48.
- Let's think about what the problem could be. Line 48 is using a document.querySelector() method to get a reference to an element by selecting it with a CSS selector. Looking further up our file, we can find the paragraph in question:

```
1 | <p class="lowOrHi"></p>
```

Fixing syntax errors – Round 2

- So we need a class selector here, which begins with a dot (.), but the selector being passed into the querySelector() method in line 48 has no dot. This could be the problem! Try changing lowOrHi to .lowOrHi in line 48.
- Try saving and refreshing again, and your console.log() statement should return the <p> element we want. Phew! Another error fixed! You can delete your console.log() line now, or keep it to reference later on — your choice.

Fixing syntax errors – Practice Time

- Now if you try playing the game through again, you should get more success — the game should play through absolutely fine, until you end the game, either by guessing the right number, or by running out of guesses.
- At that point, the game fails again, and the same error is spat out that we got at the beginning — "TypeError: resetButton.addeventListener is not a function"! However, this time it's listed as coming from line 94.
- Looking at line number 94, it is easy to see that we've made the same mistake here. We again just need to change addeventListener to .addEventListener. Do this now.

Fixing syntax errors - Summary

- Syntax errors: are easy to find since JavaScript console will display an error if it encounter any syntax error at compile time or runtime
- Use Console DevTools along with console.log() to investigate the root cause.
- Once you find the cause, try to think about why that happened, what is going wrong. From that you can come up with a fix

Section 5

Fixing logic errors

Fixing logic errors

- At this point, the game should play through fine, however after playing through a few times you'll undoubtedly notice that the "random" number you've got to guess is always 1. Definitely not quite how we want the game to play out!
- There's definitely a problem in the game logic somewhere — the game is not returning an error; it just isn't playing right.

Fixing logic errors

- Search for the randomNumber variable, and the lines where the random number is first set. The instance that stores the random number that we want to guess at the start of the game should be around line number 44:

```
1 | let randomNumber = Math.floor(Math.random()) + 1;
```

Fixing logic errors

- And the one that generates the random number before each subsequent game is around line 113:

```
1 | randomNumber = Math.floor(Math.random()) + 1;
```

- To check whether these lines are indeed the problem, let's turn to our friend console.log() again — insert the following line directly below each of the above two lines:

```
1 | console.log(randomNumber);
```

Fixing logic errors – Work through logic

- To fix this, let's consider how this line is working. First, we invoke Math.random(), which generates a random decimal number between 0 and 1, e.g. 0.5675493843.
- Next, we pass the result of invoking Math.random() through Math.floor(), which rounds the number passed to it down to the nearest whole number. We then add 1 to that result:
- Rounding a random decimal number between 0 and 1 down will always return 0, so adding 1 to it will always return 1. We need to multiply the random number by 100 before we round it down. The following would give us a random number between 0 and 99: Math.floor(Math.random()*100);
- Hence us wanting to add 1, to give us a random number between 1 and 100: Math.floor(Math.random()*100) + 1;
- Try updating both lines like this, then save and refresh — the game should now play like we are intending it to!

Fixing logic errors – Summary

- Logic errors are hard to catch as you need to understand the logic of your program in order to fix it
- It's very important to understand the requirement before you code
- A misunderstanding of requirement will easily cause logic errors in your program
- And you can't fix those errors until you fully understand the requirement

Section 6

Other common errors

Other common errors

- There are other common errors you'll come across in your code. This section highlights most of them.
- SyntaxError: missing ; before statement

This error generally means that you have missed a semicolon at the end of one of your lines of code, but it can sometimes be more cryptic. For example, if we change this line inside the checkGuess() function:

```
1 var userGuess = Number(guessField.value);
```

to

```
1 var userGuess === Number(guessField.value);
```

Other common errors

- The program always says you've won, regardless of the guess you enter
- This could be another symptom of mixing up the assignment and strict equality operators. For example, if we were to change this line inside checkGuess():

```
1 | if (userGuess === randomNumber) {
```

to

```
1 | if (userGuess = randomNumber) {
```

Other common errors

- SyntaxError: missing) after argument list
- This one is pretty simple — it generally means that you've missed the closing parenthesis at the end of a function/method call.

Other common errors

- SyntaxError: missing : after property id
- This error usually relates to an incorrectly formed JavaScript object, but in this case we managed to get it by changing

Other common errors

- SyntaxError: missing } after function body
- This is easy — it generally means that you've missed one of your curly braces from a function or conditional structure. We got this error by deleting one of the closing curly braces near the bottom of the checkGuess() function.

Other common errors

- SyntaxError: expected expression, got 'string' or SyntaxError: unterminated string literal
- These errors generally mean that you've left off a string value's opening or closing quote mark. In the first error above, *string* would be replaced with the unexpected character(s) that the browser found instead of a quote mark at the start of a string. The second error means that the string has not been ended with a quote mark.
- For all of these errors, think about how we tackled the examples we looked at in the walkthrough. When an error arises, look at the line number you are given, go to that line and see if you can spot what's wrong. Bear in mind that the error is not necessarily going to be on that line, and also that the error might not be caused by the exact same problem we cited above!

Other common errors – Summary

- Real program won’t always be that simple to work out what's wrong in your code
- This will save you a few hours of sleep and allow you to progress a bit faster when things don't turn out right, especially in the earlier stages of your learning journey

Thank you

Q&A