# JAVASCRIPT - LESSON 1, TOPIC 2: DEBUGGING & CONSTANTS

INTRODUCTION
DEBUGGING
THE IMPORTANCE OF MAKING MISTAKES
CHALLENGE

## INTRODUCTION

00:01 Welcome back. This is lesson one, topic two, debugging and constants. So as ever, let's start with a bit of an overview of what we're going to cover in this video.

00:13 First off, what is debugging? Then, what are constants? Constants versus strings (the concept we looked at in the previous video) declaring constants and lastly, using constants.

00:24 So first off, let's start with what is debugging, starting with the definition. Debugging is the process of fixing errors in code.

00:38 It's something you will have to do every single day as a programmer no matter how junior or senior you are.

## DEBUGGING

00:45 And as you can see in these screenshots, we get a bit of a visual representation when something's gone wrong in JavaScript in that console window that we looked at in our previous session.

00:58 As I say,  we make mistakes all the time, but it's fine, we debug them. And as the old adage goes, if you're not making mistakes, you're not learning.

## THE IMPORTANCE OF MAKING MISTAKES

01:08 The great thing about JavaScript, and particularly CodePen, is that you get this really handy visual output when something's gone wrong.

01:16 So you can see we get this red exclamation mark and a big red error message. It can look intimidating at first, but actually, it's just a really helpful tool for pointing you in the right direction to help you work out what's gone wrong.

01:31 As a little challenge for you now, I'm wondering if, from our previous session, you can work out what's wrong with the code on screen now.

01:39 I'll give you a few seconds but don't worry, we're just about to cover it together in CodePen. Have you spotted it yet?

01:46 Don't worry if not, let's start our walk-through.

## CHALLENGE

01:51 Okay, so as you can see, I've hopped over to CodePen now and we have the contents of that screenshot that we saw previously up in our JavaScript editor here.

02:00 Have you worked out what's gone wrong? Well, don't worry if you haven't because as I say, JavaScript gives us these really useful error messages in our console window.

02:10 So first we have this exclamation mark just to notify us that something's gone wrong. And then we have this really handy error message.

02:18 So let's break it down a bit. At the start here we can see an uncaught syntax error. So just like the English language, JavaScript comes with its own set of rules that make it readable and understandable.

02:31 So in English we have things like grammar, we have punctuation, and we know that if we are representing speech, we put it in speech marks.

02:39 We also have rules just like this in JavaScript and they're called syntax. Now after this, this is the really useful part.

02:47 So we have a missing close bracket after the argument list. Okay, so maybe the argument list part's a little confusing at this stage but let's ignore that for now and focus on the missing close bracket.

03:00 So what do you think that might imply? I'm sure you're correct. So if we look back at the program that we've written up in our JavaScript editor, we can see we have the comments, which looks fine to me, then we have our console.log method, we've opened a bracket, we've put our string hello and the wave emoji.

03:19 But, what have we forgotten to add at the end here? A close bracket. So this error message is simply notifying us of the fact that we've opened a bracket and we've forgotten to give it its partner, one of our fundamental rules in JavaScript.

03:34 So let's test our theory now. Let's run this updated code and see if it works as expected. So I've clicked the run button and let's see.

03:45 There we go. We get the message that we expected. So as we can see, the error here was simply forgetting to add that close bracket.

03:53 To the end of our console dot log method just as pointed out in that handy error in the console.

04:00 Now that you've debugged your very first piece of JavaScript, let's look at constants, and specifically the idea of giving reusable data a name.

04:11 Let's start with a definition. Constance is a kind of variable, a way to give a name to data that we want to label and then reuse.

04:21 So let's consider our hello greeting message from our previous lesson. Sometimes you can imagine that the thing we want to console.log is really long.

04:31 For example, *Hello Emily, how are you doing today? What's the weather like? Do you have any plans for this weekend? What do you think of the new Taylor Swift album?*

04:39 *Yada yada yada*. As you can imagine, it can get really long and thus it can be really easy to make errors, syntactical errors like the ones we just looked at.

04:48 Well, wouldn't it be nice if instead we could just say console.log, open up brackets, *greeting*. The meaning of your code would instantly be much clearer.

05:01 So this is where constants come in. So first, let's differentiate constants from strings. So looking at our example here, on line one, we can see we have this key phrase, const, and then message.

05:16 Message is the name that we're going to give the upcoming string. So we have const message equals we got this.

05:24 Then, instead of console. logging *we got this,* we can just log the message itself. So before the equals sign, we have the name of the constant.

05:36 And after the equals sign, we have the value that we want to give it. So we kind of declare it once and if it's really long-winded, like I was just saying before.

05:45 *Hello, Emily. How are you? How was your weekend?* We only have to type that once. It's it's really useful. Then we can reuse this message instead.

05:55 But can you spot the error in this screenshot here? Let's move over to the pen and work through this together because, as you can see, we've written console.

06:08 Log message, but instead of getting the expected, we got this. We've got just the word message in quotation marks. Why might that be?

06:20 Well, let's work this out together in our

06:24 I'm back over in CodePen with our programme. And let's have a look line by line to see what's going on.

06:30 So starting with line one, we have our keyword here, const. Now that's to tell us that the next thing is going to be a constant.

06:40 So this idea that we have some data that we want to reuse. So we're going to store it in a variable to save us typing it out every single time.

06:50 So we start with a keyword, const, and then we have the name that we're going to give our constant. So in this case, message.

06:58 Then we have the equals sign. And whatever comes after that is the data that we're going to store inside our message constant.

07:07 So in this case, we have a string. And how do we know it's a string? Remember? Because we have the two speech marks

07:14 Wrapping the text. So we declare our constant. We give it the name message. And we give it the data we've got this.

07:25 Now, on line two, we're using our old friend the console.log message. But we can see when we hit our run button up here.

07:35 Let's click that one. Something has gone awry because we're not getting what we expect. So in this case, I would expect to see.

07:44 We got this, but instead, I'm getting a string message. Now, have a think about why that might be. Are you ready for the answer?

07:54 Well, it's because when we are using our console.log method with a variable, a constant, we don't need these speech marks.

08:06 Speech marks are for text and for strings. So we become familiar with those, but this is a new data type we're looking at.

08:14 This is a constant of message and you can see where we declare it here on line one. We haven't given it any speech marks.

08:23 So we need to do the same when we want to log it. So let's run the program again and see if this time we get, we got this in the console.

08:33 So it works this time. So again, another syntax error, really, really common, really easy to make. But the thing to take away here is to remember that when we declare a constant if we want to log it, we don't need to wrap it in those quotation marks.

08:52 Those are reserved purely for strings.

08:56 So now that we've looked at the code in action, let's kind of break down what was happening there. So if you remember in our very first line, we had a constant message we got this.

09:05 So that's what we call declare entering a constant. And let's just remind you what a constant is. A constant is a kind of variable.

09:14 It's reusable. And it's a way to give a name to data that we don't want to change. We want to kind of keep using it, but that data is always going to hold the same value.

09:22 That message is always going to be, we got this in that case. And let's look at another example here. So a way to declare our constant is we use that keyword at the very start of const.

09:36 And then we give it a name, in this case, greeting, use our equals sign and give it a value.

09:45 So const greeting equals hello. Or on the second line here, we have our keywords. We give the variable a name and then we give it a name. We assign it a value after the equals sign.

09:58 So const is the keyword you use to declare a constant, but you can only give it a value once. So once we've said that greeting equals hello, later in the code, we can't suddenly say greeting equals goodbye.

10:10 It keeps its value. And equals is what we call the assignment operator. So it's an indication of what value we want our constant to hold.

10:22 Fantastic. So after we've declared our constants, we can then go about using them. So let's take a look at that in action.

10:30 So we've got our greeting, we've got our emoji, and then on lines five and six here, oh no six and seven, we're using our console.log method to output them to the console.

10:42 So after we declare a constant, we can use it as many times as we want. That's really important to note.

10:48 But the only thing we can't do is change its value, but we can keep using it, use greetings throughout our code or use emojis throughout our code.

10:55 And then in this example, we're using the console.log method. To output the value of the variable to the console. Awesome.

11:04 So let's look at that in action. Alright, so we're back in CodePen and let's start by adding a comment for future Emily so that I can remember what I was trying to achieve with this code.

11:16 If you recall, we start with two forward slashes and then I'm going to declare our constants. So next, can you remember how we declare a constant?

11:29 First, we have that handy keyword, const. And then I'm going to give it a name. So const greeting equals, and then, oh no, let's not do that.

11:43 Okay, so back in CodePen, let's put this into practice. So, if you recall, I might want to look at this code later in the future and kind of add a little, okay, so back in CodePen, and let's put what we just learned into practice.

12:01 So to start with, I'm going to add a comment. Just in case future me can't remember what I was trying to do at this point in time.

12:08 So if you recall, we need two forward slashes, and then I'm going to write what my intention is with this code.

12:13 So I'm going to put declaring a const. Now, for the actual declaration of the constant. So if you remember, first we need that keyword const and then we need to give our constant a name.

12:27 So in this case, I'm going to say message. Next, we need our really important assignment operator. In this case, a fancy way of saying equals and we need to give our constant that has the name message a value.

12:43 So I'm going to put, I'm going to make a string. So with those quotation marks, I'm going to say hello.

12:50 And then my little wave symbol. So now we've declared our constant keyword, const, and name message, and we've given it a value.

13:01 So let's add another comment. Now I want to try and output our const to the console. Can you remember the handy method we've learnt to do that?

13:15 That would be console.log and then remember we need to open a pair of brackets so it's really important that we have both, in this case, CodePen's given me two, nice and handy.

13:27 And how am I gonna get it to log just the message and not the string message? I'm gonna avoid those quotation marks, I don't need those right now and I'm gonna write the word message.

13:41 Now let's see if we've been successful, we're gonna hit this run button and let's see what happens. Hey and success!

13:52 We have hello and our little wave emoji printed on our console. Well done! You've made it to the end of topic two, constants and debugging, and now it's time to test your knowledge with a quiz.

14:03 Good luck!