[MUSIC] So in this lecture, we're going to

talk about common language constructs. And these are the things that you

pretty much do in any language, but we're going to look at how you

get them done in JavaScript. So here I am in Sublime editor,

and I'm looking at script.js, which is located in the Lecture43 folder. And again, the set up is very similar. I have the Chrome Developer Tools

open here in the Console tab. And therefore, whenever we save this

script.js, the Browsersync tool that is running in the background will

sense we've updated this file, and it will go ahead and refresh the

browser, executing our JavaScript code. So I have a bunch of topics that

I'd like to go through, and basically, the code for those topics is

sitting right in the same file, but I, at this point commented them out,

so we could go one by one. The first thing I want to take

a look at is string concatenation. Well, just like a lot of other languages,

JavaScript allows you to concatenate strings on the fly as if they are numbers,

as if they are arithmetic numbers. So for example, here we have declared

a string with the value, Hello. And the very next line,

we're concatenating the string with another string

with a space and the word World. So now this operator right here, +=, is really equivalent to the operator,

let me comment that string out. Comment that line out and

if I said that, string = string + World, this would be an identical

line to this one. So let's go ahead and

keep both those lines for now, and we'll comment out the second line. And the way I comment out the thing

without actually having to type it is to press Ctrl+/ on Windows and

Cmd+/ on a Mac. And basically what I'm doing is I'm

concatenating a string with the word World, and then I am going to output

it to the console using string + and an !, and other string. So the only difference between these

two statements, this one and this one, is that this one is actually going to

store a new string in the string variable as opposed to this one is just

going to concatenate it and throw it right into the log function. So let's go ahead and save this. As you can see, it refreshed and

it said Hello World! So let me test this out as I said before,

if I comment this line out and uncomment this one,

these lines are exactly equal in length. This is just a shortcut to the say

the exact same thing as we're saying right here. So let's save that, and again,

it's going to refresh and saying exactly the same thing. So let's go ahead and

comment this whole thing out. So we could go to the next example

that I want to talk about. So actually,

let me close the file browser on the side, so we could see it a little bit better. Okay, so the next one we're going to

talk about is math operators. And the math operators

are straightforward, and if you've done any kind of math

operation in other languages, there's really no difference

here pretty much. Actually, let me comment out

the second line for now. So the first thing is there's

a plus operator, the minus, the multiply operator,

and divide operator. And the same time of rules that

apply in other languages and really in math in general

apply here as well. So, for example, the precedence

of parentheses also applies here. So as you can see here I'm doing 5 + 4, and that will get executed

before the division by 3. So 5 + 4, that's 9 / 3, that's 3. So if I say that, I should see that it

is going to cancel that log output 3. Now let me uncomment the second line here,

and what we're doing here is kind of strange. We're doing undefined / 5. Now when we execute that, let's go ahead

and save that, we're going to get an NaN. Now what is NaN? NaN is a special symbol that

indicates not a number. And the biggest reason I'm showing it

to you is because it's very common to define a function, let's say we define

a function test1 that takes an a argument. And in the function,

we'll do console.log, and console.log will say a / 5. So now, if we call this test1 function,

we never really passing the a in here. So what's going to happen? Go ahead and let's go ahead and save that. And you'll see we're going to

get an NaN again, not a number. So when you debug your code and

you call a function, and somewhere inside the function,

something starts throwing not a number. Chances are that there's something that

was supposed to be passed in into that function, and

either not neglected to be passed in or something happened to it along the way

right before it was supposed to be passed in, and then you have an undefined

variable inside your function. Okay, so let's go ahead and

comment this out. And we'll save that again, so

we'll clear the screen for us. And let's go to this

topic called Equality. Let's uncomment this, so here what we

have is, we're declaring a variable, defining variable x = 4 and

defining the variable y = 4, as well. Now notice, by the way, that we're

defining them with only one keyword, var. This is a shortcut in JavaScript that if

you wanted to, you could just put a comma in between the variable definitions,

and that is a perfectly legal syntax. And in fact, this is very common

in other languages as well. In the next line, we have an if statement. And the if statement in JavaScript is

nothing special compared to any other language. It's pretty straightforward. And what we're doing here

is we're comparing x and y. And we're comparing it with a == operator. Now one of the most common mistakes is,

instead of ==, is to say x = y. Well, x = y will take the value of y,

place it in x, and use the result of that as the value to

evaluate whether or not it's true or not. So we don't want that. We actually want to compare x and

y, and therefore, it's a ==. So what we're doing here

is saying if x = y, then we're going to execute the next line,

which is this console x == y. So let's go ahead and comment

the second part out here and save that. And you can see it says x is equal to y,

because x is 4, and y is 4 as well. Now let's make this a little

bit more interesting. Let's uncomment this line and

see what we're doing here. Here first of all,

we're changing the type of x. The x now is no longer 4 as a number,

x = 4 as a string. So what happens if we

actually compare them now? Well, let's go ahead and try that. Let's save that. And you could see it says x = 4,

4 as a string is equal to y = 4. Now what is going on here? Obviously, the string can't

possibly be equal to a number. Well, what's going on here

is called type coercion. And type coercion just means that

the language converts something for you from one type to another

type automatically on the fly. So when the JavaScript tries

to evaluate this equality, it converts one of the types to be

the same type as the other variable and only then compares the values. Now what if you didn't want that? What if you didn't want JavaScript

to automatically convert things for you when trying to evaluate its equality? Well, that's where strict

equality comes in. Let's go ahead and comment that out and go

to the Strict equality section here, and we'll uncomment that. So strict equality is

signified by using ===. So now if I'm comparing x === y,

it will first see whether or not these things, this x and

y, are the same type. And if they're not, it's not going to try

to compare them anymore at that point. And in fact, I just realized I commented

out x and y that I'm defining here, so let's go ahead and

actually uncomment this whole thing. That way,

we'll have the variables available. So now if we execute this, x at this

point, is a string 4, and y is a number 4. So this will never actually

execute right here. What will execute,

however, is the else part. And here you get the chance to see what

elses looks like, in JavaScript, and it's really again, nothing special. Other languages have

basically the same construct. And in the else clause,

we're having console.log, that says a Strict x as a string

4 is NOT equal to y = 4, and that is what we actually should

see in the second printout. So let's go ahead and save that and

see what it says, it says Strict x is NOT equal to y = 4, because one

is a string and one is a number. In part two of this lecture, we're

going to pick up where we left off and speak about what JavaScript considers

false and what it considers true.