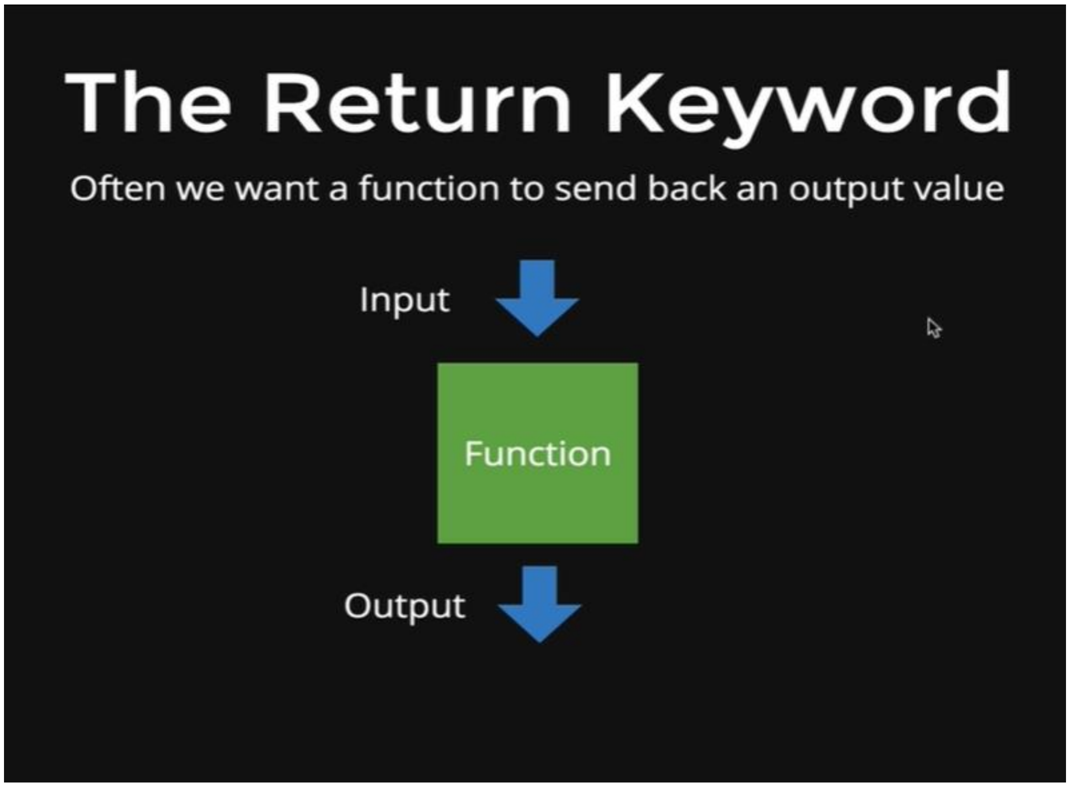
This lesson is going to focus on conceptually on what the *return* keyword is and then also how do we write functions that use it.


The Return Keyword
Often we want a function to send back an output value

We like to imagine that function is a machine, we write a function and it takes some inputs, and those will be the *arguments* and the function performs some tasks with those inputs, like it does math, it checks if a user is logged in or it adds points to a score and then it returns something at the end, there is an output that it sends back. When we use the *return* keyword in the function it means that we can capture the value that is coming back from the function, after the function had executed all the code inside it.

For example, we have a *square* function that takes input as a number and then square that number and returns the value. Now we want to use that value somewhere in our code, we might want to give an output to a user that says, “4 squared is: 16”, and when we write it in code we would write like “4 squared is: square(4)”, so that the function call square(4) is replaced by the value returned by the function. If we have a return keyword at our function then the function will return a value to the function call and our program will print “4 squared is: 16” but if we do not have a return keyword the program will output “4 squared is: *undefined”.*

This happens because nothing is being return, we are missing the output from the function. Even though the function executes the code inside it, it does not send anything out, so to do that we use the *return* keyword.

So, if we want our *square* function to return the value we got after squaring the number that was passed in the function then we will code as below,

```
function square(x) {
        return x * x;
        }
```

The *return* keyword is going to return one thing per function, we could theoretically have multiple return statements but only the first one is actually going to run unless we have some sort of *if* statement and inside of the *if* statement we returned one thing and inside of the *else* we return something else, but still only one of those is actually going to return a value or a variable.

In this case, we are returning *x* times *x* and if we run the function, we get our squared result.

```
 function square(x) {
        return x * x;
        }

square(4);
```

Output:

```
16
```

And then if we run the following code,

```
“4 squared is: ” + square(4);
```

Then the *square(4)* function is run and 16 is returned

Output:

```
4 squared is: 16
```

The other thing we can do with a *return* keyword is, save it in a variable, so we can do something like below,

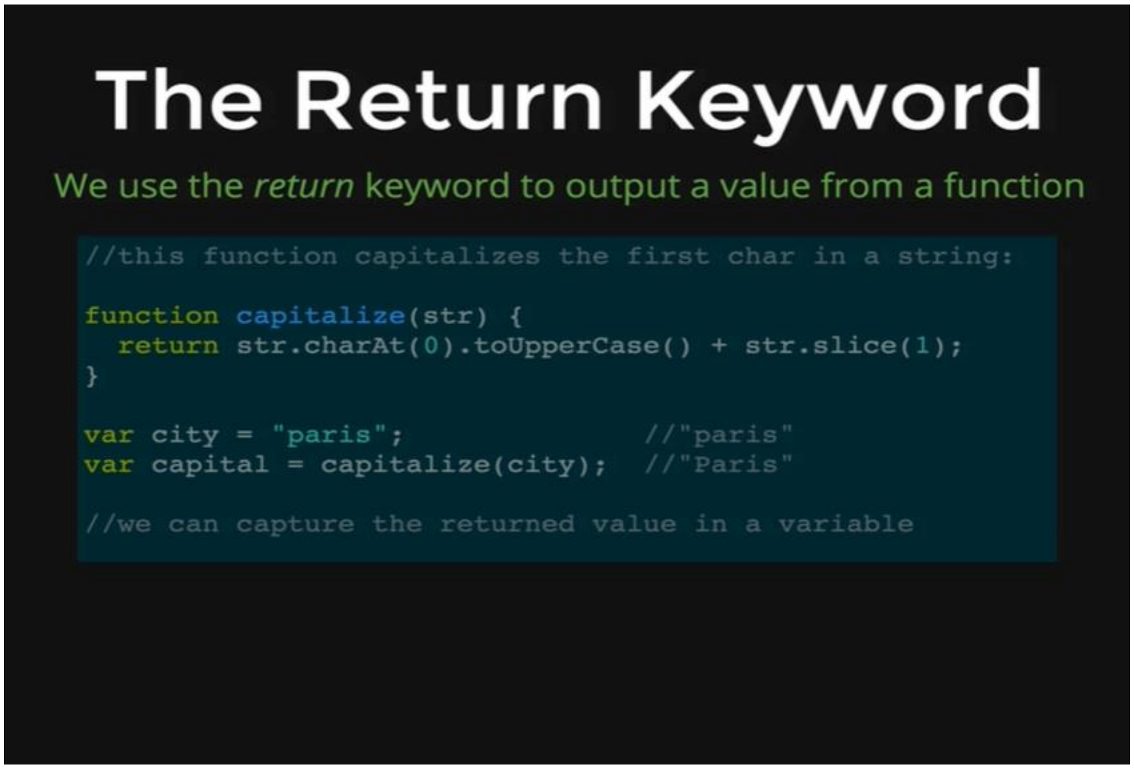
var result = square(104);

And now if we want to view the result variable, then we can view it as below

result

➔ *10816*

So, the function *square(104)* was evaluated and that returned *10816* which was then stored in the *result* variable.



The Return Keyword

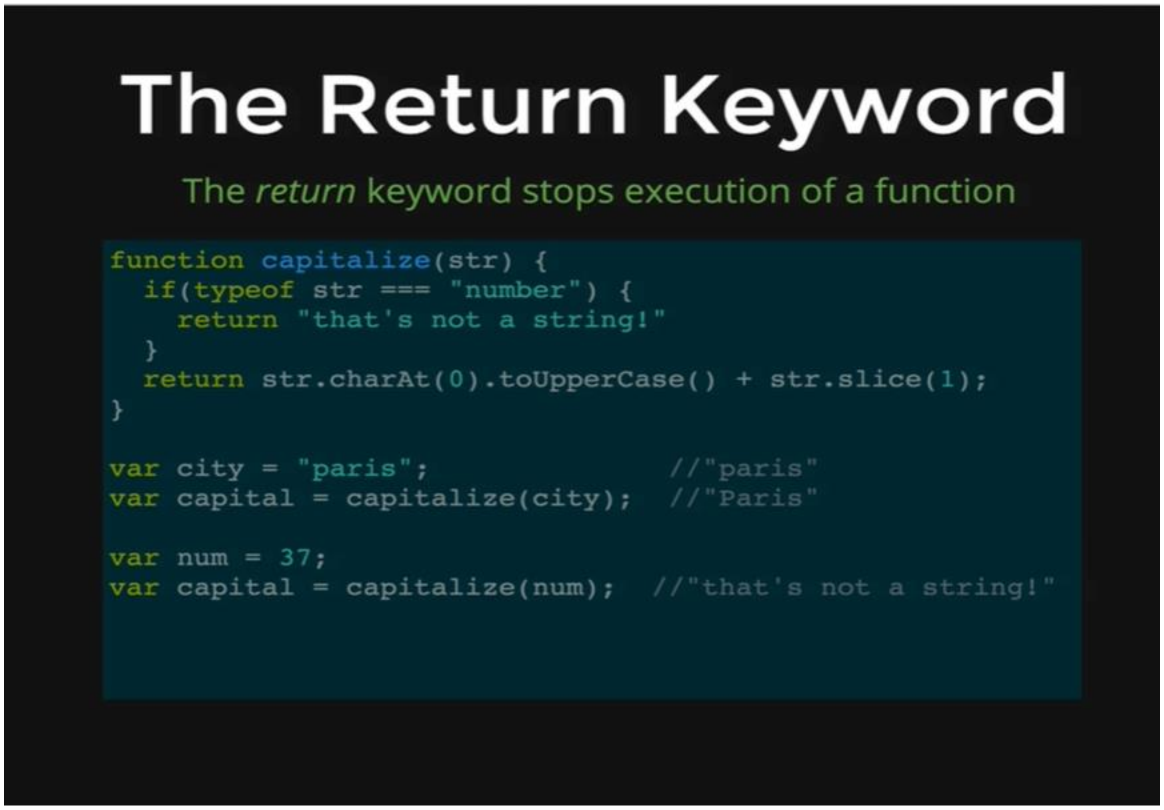
We use the *return* keyword to output a value from a function

```
//this function capitalizes the first char in a string:

function capitalize(str) {
  return str.charAt(0).toUpperCase() + str.slice(1);
}

var city = "paris";                //"paris"
var capital = capitalize(city);   //"Paris"

//we can capture the returned value in a variable
```

We have another example of a function that returns something, the function is called *capitalize* and it takes in a word like paris and what it does is, it capitalizes the first letter and returns the entire string with that first letter capitalized. So, paris turns into Paris with a capital P. The whole point of this function is that it alters our original data a little bit, so we pass in a string and we get back that string with its first letter capitalized.

What we want to emphasize is that we have a variable *city* which is set to paris with lowercase p and then we are capitalizing it by passing the variable *city* in the *capitalize* function and then we are saving the return value to a new variable called *capital*. Now we can use the return value anywhere in our code by using the *capital* variable.

Let's see how this function works. The first part takes the first letter, the character at index 0 and it uppercases it and that would give us the uppercase P and then *str.slice* takes a number in this case 1, so that takes everything from index 1 and onwards, "aris" all lowercase and it concatenates them with the capital P. So, we are capitalizing the first letter and then taking everything else after the first character and combining the two and returning that.



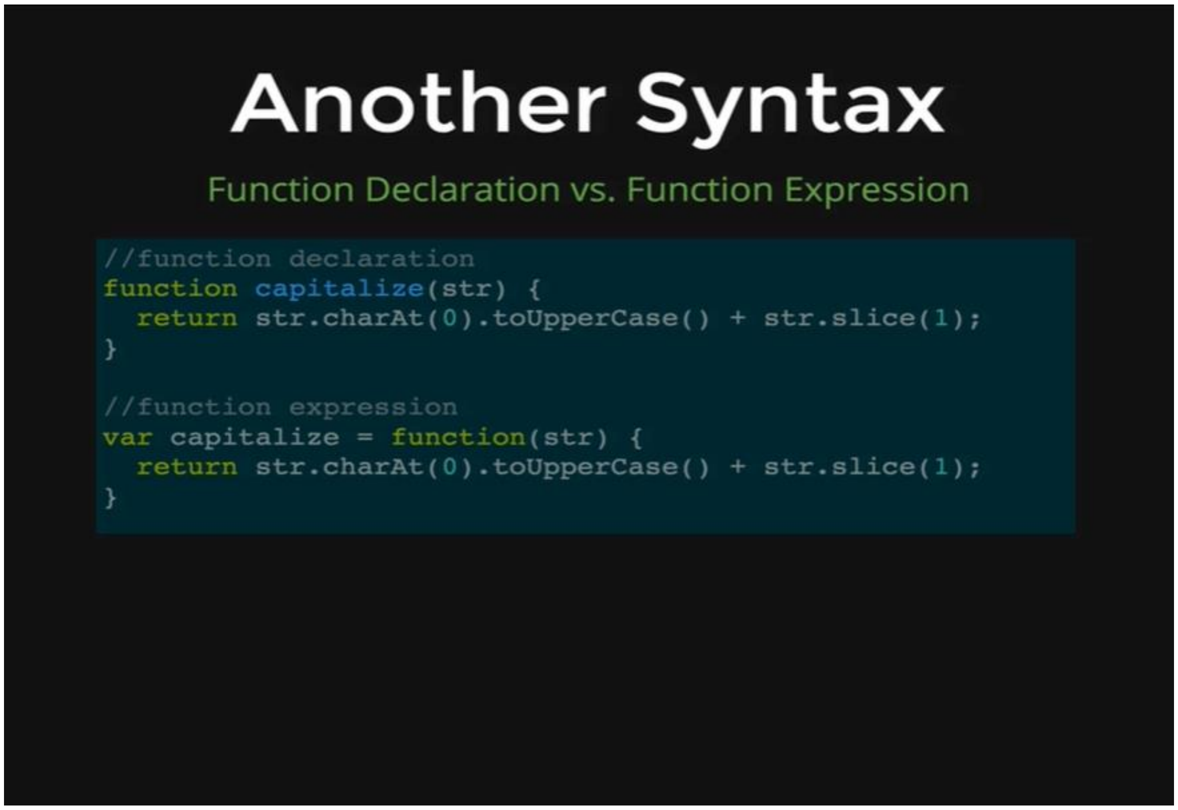
The Return Keyword

The *return* keyword stops execution of a function

```
function capitalize(str) {
  if(typeof str === "number") {
    return "that's not a string!"
  }
  return str.charAt(0).toUpperCase() + str.slice(1);
}

var city = "paris";                //"paris"
var capital = capitalize(city);    //"Paris"

var num = 37;
var capital = capitalize(num);   //"that's not a string!"
```

Another aspect of the *return* keyword is that it stops the execution of a function, so as soon as we return something, the function is done. The whole point of a function is to take an input and return something, so as soon as it returns that, it is just the end of the function execution.

Here's an example, it's the same capitalize function, except with a small difference. This function is checking if we are passing in a number instead of a string, which is what the *if* statement does, *if(typeof str == "number")* and the next line returns *"that's not a string!"* and we do not bother any of the code below, so it returns "*that's not a string*" if we pass in a number and then the function short circuits and the code below the first return statement never runs. Even though there is no *else* statement below the first return statement but that code does not run at all, because the function short circuits if we pass in a number. Otherwise, if we pass in a string like "paris" then the condition inside the *if* statement is not true thus the function continues by skipping the *if* statement and executes the second return statement.



Another Syntax

Function Declaration vs. Function Expression

```
//function declaration
function capitalize(str) {
  return str.charAt(0).toUpperCase() + str.slice(1);
}

//function expression
var capitalize = function(str) {
  return str.charAt(0).toUpperCase() + str.slice(1);
}
```

There are two different syntaxes for declaring a function, the first one is the one we have been using, its called *function declaration,* so we write function and the name of our function and then we pass arguments in and we write our function body inside of the two curly braces.

There's another way of writing a function called *function expression* and the way we do that is we write a variable; here we are declaring the variable with the name *capitalize* and then we set that equal to a function. These are two ways of defining equivalent functions, the first one is a declaration and the second one is an expression.

There is one small difference is that if we declare a function by having a variable as we have in function expression and then if we decide to change the variable *capitalize* to number 10 or number 15 or any other number, then our function will be lost. The *capitalize* variable will be set to that new number and when we use the *capitalize* variable, we will get the output of the number we set it to be instead of the function.