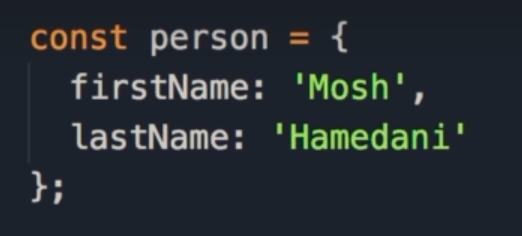
JavaScript Basics - Functions

# Section 6 - Getters & Setters

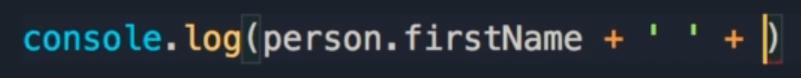
In this section, we are going to look at 2 special methods in objects, called getters and setters.



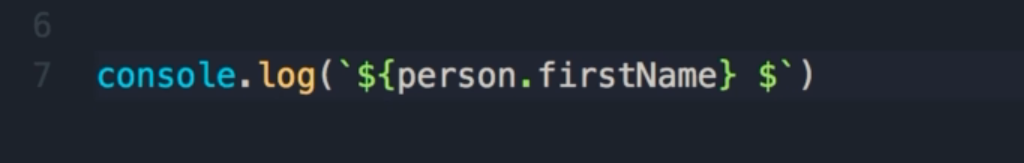
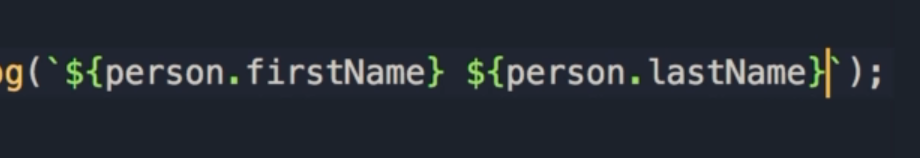
We start by initializing a person object with 2 properties: firstName & lastName.

## Displaying Full Name

Now we want to display a person’s full name.



We can either log the person.firstName, concatenate it with a space, then add person.lastName, or we could use template literals.



In between backticks, we have 2 arguments, the first is person.firstName, and the second is person.lastName

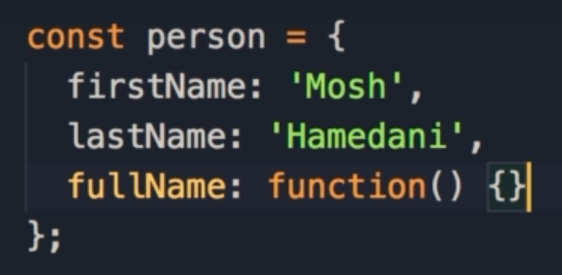
The problem with this approach is maybe there are multiple places in our application where we want to display someone's full name.

With the current implementation, we'll have to repeat this template literal in multiple places.

## Creating the method

A better approach is to define a method in the object, call it fullName, and move this expression in there.

And then whenever we want to display a person's full name we simply call that person's method.

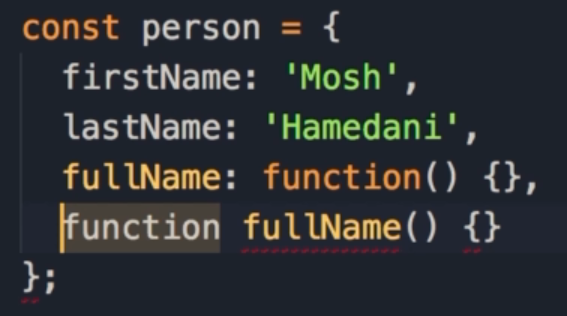


Here we start with a key-value pair: fullName which we set to a function.

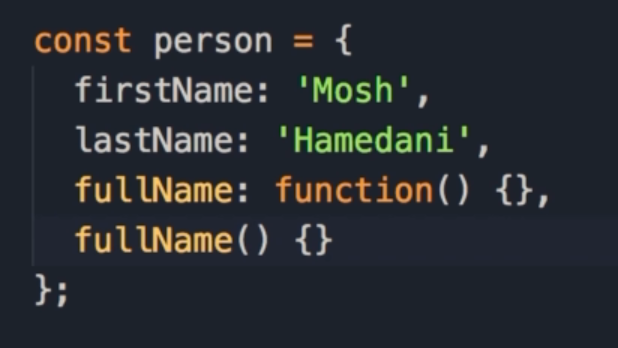
## Shortening the Syntax

As we have learnt before, we have a shorter syntax to add a method to a function.

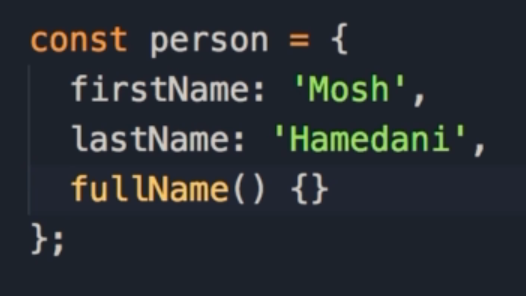
To define a function outside of an object, we use the function keyword, give the function a name followed by parentheses and curly braces.



When we need to add this inside of an object, we just drop the function keyword.



So you can see, this new syntax is shorter and cleaner than the older syntax.

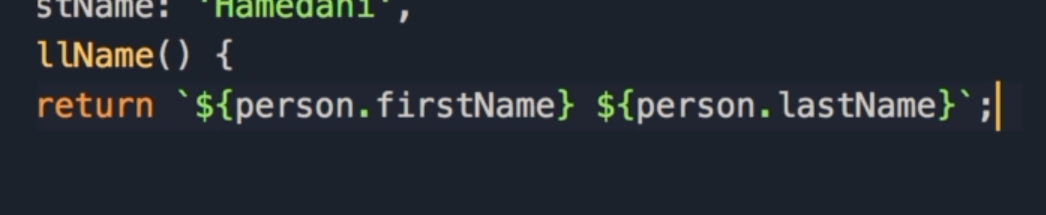
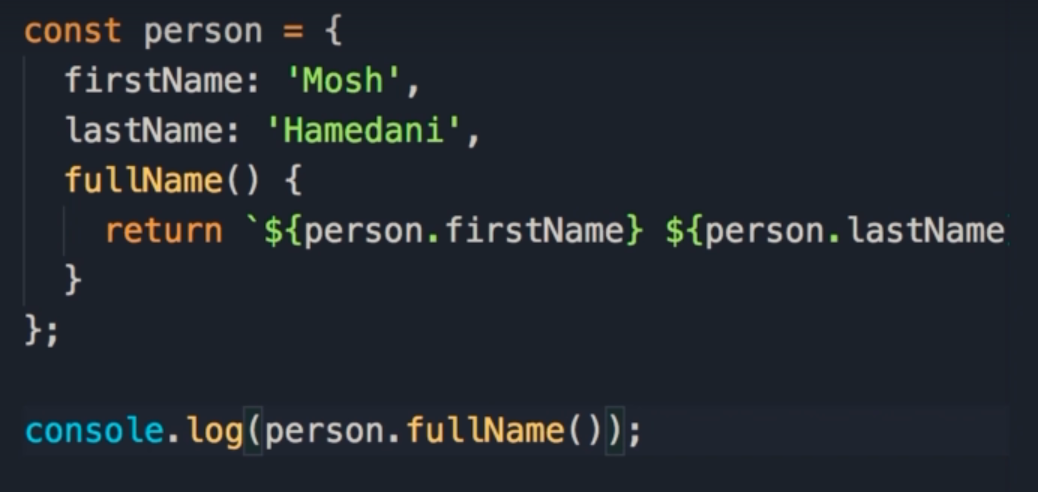


Here is the fullName method, which returns firstName and lastName.

Any time we want to access the full name, we simply call person.fullName()

In our console, we see the full name printed out.

There are a few problems with this approach however.

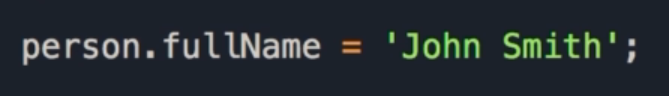




## Read Only

The first problem is that this is read only.

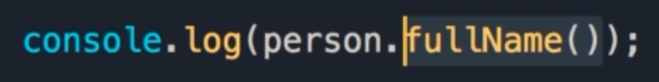
In other words, we cannot change the full name by assigning it using dot notation.



It will be nice if we could do this, and then the first name and last name properties would be automatically set based on what we passed here.

## Calling a method

The other issue, is that we would prefer to call it like a property, not a method.



Basically, to remove the parentheses.

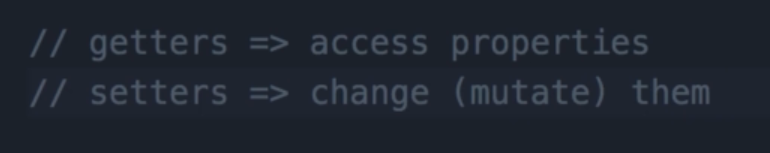


How do we deal with this?

This is where getters and setters come into play.

## Getters & Setters

We use getters to access properties in an object, and setters to change or mutate them.In this object we should add a getter, and with that getter we can read this person's full name, like a property, and with a setter we can set it from the outside.



## Getters

This is how it works:

We prefix our method with the get keyword.



Now this method is a getter.

This means we can access fullName like a property.



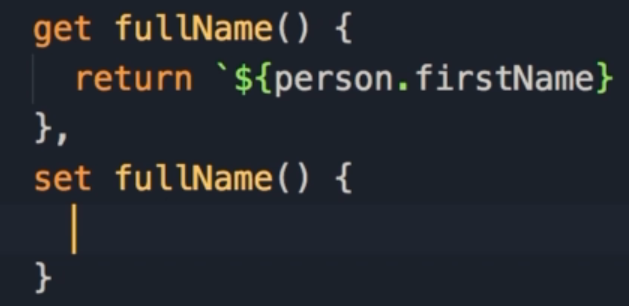


In order to set the fullName from outside the object, we need a setter.

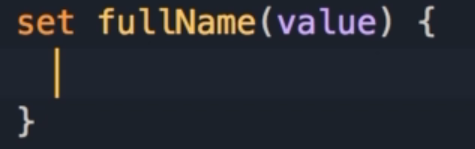
## Setters

A setter is very similar.

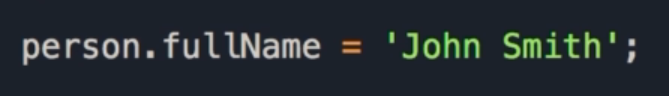
We use the set keyword, followed by the name of the property or method, followed by a code block.



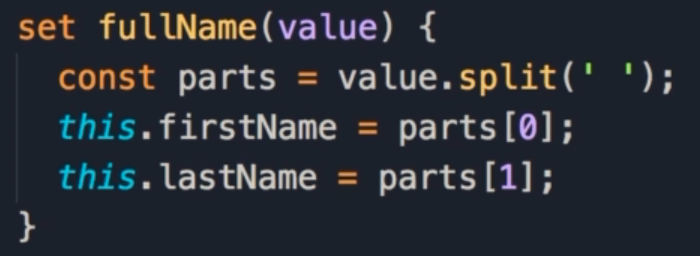
However, this method needs a parameter, which we will call value.



The value of this parameter is what we have on the right side of the assignment operator when calling the setter.



Let’s assume that value is a valid string: we need to split that string by a blank space, take the parts and set firstName and lastName properties.



We start by calling the split method of strings on the value we pass in, and separate these by blank spaces.

The split method returns an array, so we store that in a variable called parts.

Now we set firstName to the first item in the array, and lastName to the second item in the array.

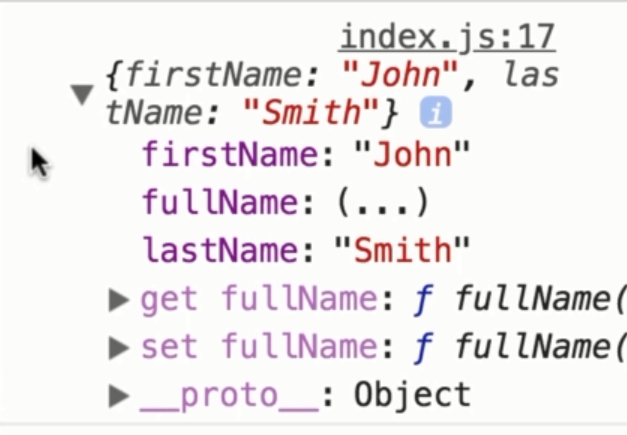
Full Example

Here is the full object, with firstName and lastName properties, as well as fullName getter and setters.



Underneath we use the setter method to change the full name to John Smith.

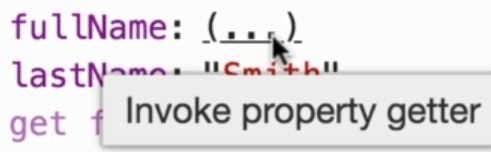
Let’s take a look at what happens when we log the entire person object to the console.



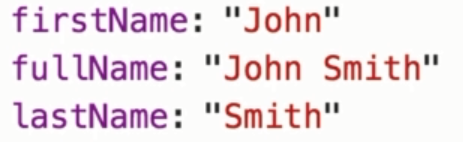
Here we can see that firstName and lastName properties have been updated.

Also note that the value for fullName is (...)

Sometimes when you log objects to the console, you may see properties with this kind of value.



As you can see from the tooltip, this is a getter.

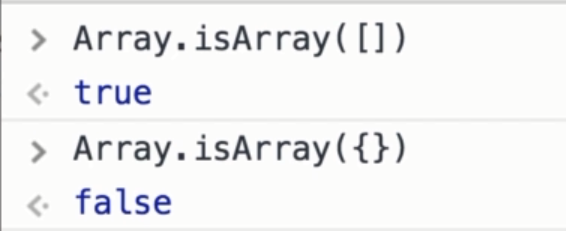


In order to read the value you need to click here, and now this method is executed and the result is returned in the console.

## Recap

We use getters to access properties in an object, and setters to change or mutate them.

# Tasks

1. Sum of Arguments:  
   Create a function called sum() that takes a varying amount of arguments, and returns their sum. You can pass in 1 or more arguments
   1. To challenge yourself, modify the sum() function to accept an array of arguments as well and return the same result
   2. To do this, you will need to use the Array.isArray() method
      1.   
         If you pass an array into the Array.isArray() method it returns true, if you pass anything else, like an object, it will return false
2. Area of a Circle  
   Create a circle object using the object literal syntax
   1. This object should have a radius property that we can read an write to
      1. Eg. circle.radius = 2;
   2. An area property that is read only. We should not be able to set the area from the outside, but we should be able to read it
      1. Eg. console.log(circle.area);
      2. The formula to calculate area of a circle: area = Pi \* radius \* radius