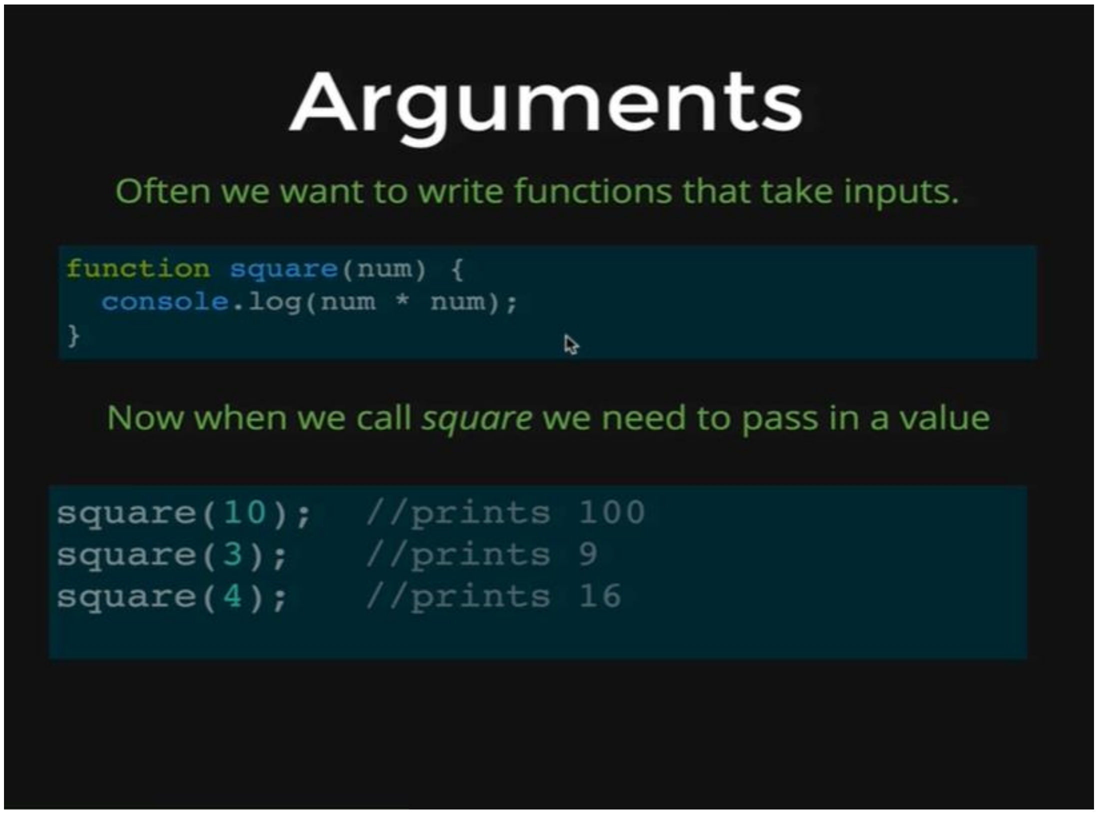
We have seen how a function does the same thing every single time it is called. Now, what if we have a function that says "Hello" and it only console.logs "Hello", but how about we want this function say hello to a specific person, what if we want to personalize it. For example, we want a function that says hello to someone called 'Jeff'. We want it to output "Hello Jeff!".

Then what should we do? To do this we need to talk about *Arguments*.

Arguments are how we can write functions that take inputs, so far, a function have not taken any inputs but below we have an example of a function called *square*.



Arguments

Often we want to write functions that take inputs.

```
function square(num) {
  console.log(num * num);
}
```

Now when we call *square* we need to pass in a value

```
square(10);   //prints 100
square(3);    //prints 9
square(4);    //prints 16
```

What this function does is that it takes a number and it console.logs the square of that number. So, the syntax to say that a function is expecting something to be passed in, then it is expecting an argument, it looks like the *square* function above, rather than empty parentheses we put the name of an argument. So, in this case we are calling the argument *num* but it can be called anything at all. Its just a placeholder.

Whenever a user calls *square* function and we pass in 10, *num* is going to hold the value of 10 temporarily, if the user call *square* with 3, num is going to hold the value of 3, and then we can use *num* inside of a function. So, in this case *num* times *num* is just going to take whatever number was passed into the parentheses and square it. So, as you can see *square(10)* prints 100, *square(3)* prints 9 and *square(4)* prints 16.

Let’s go ahead and write our own function to illustrate how we write arguments. At first, we are going to write a function that is not going to take any argument. Its going to look like below.

```
function sayHello() {
        console.log(“Hello there!”);
        }
```

Then we can just press *Enter* and JavaScript gets to view the function and now we will call the function.

```
sayHello()
```

Output:

```
Hello there!
```

And that’s it all does. Now we want to define a function that takes a person’s name and says hello to that person. So, we want our function to take a person’s name as an argument and says hello to that person.

```
function sayHello(name) {
        console.log(“Hello there” + name + “!”);
        }
```

So, after we press *Enter*, the function gets registered to JavaScript and now we will call the function by passing a name called “Rusty”.

```
sayHello(Rusty);
```

Output:

Hello there Rusty!

So, here we get “Hello there Rusty!”.

Now, we will try other names to see how much effective our function is.

sayHello(“Charlie”);

Output:

Hello there Charlie!

sayHello(“Mary”);

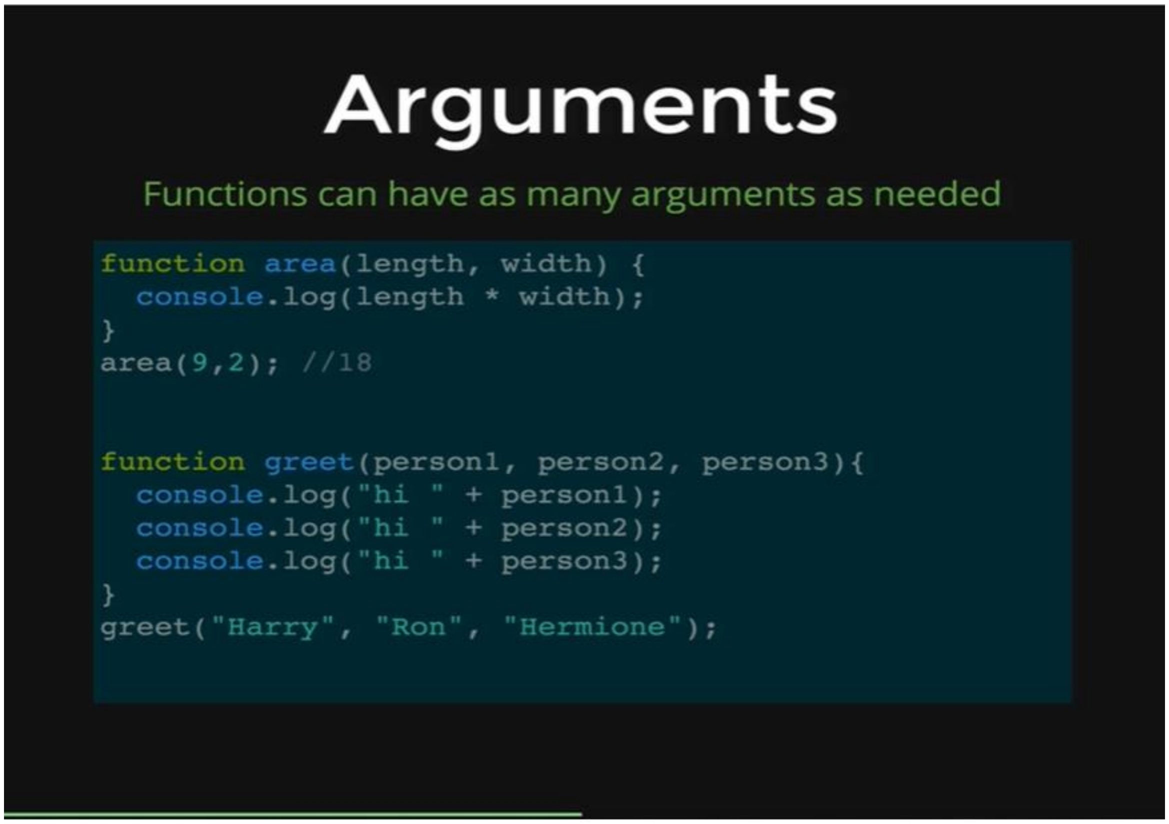
Output:

Hello there Mary!

And you can see our code changes, our function has adapted to whatever value we pass in.

The use of arguments is what makes our functions super powerful. In a website like Facebook there could be a function like *MakeHomepage* and that function might take arguments as user who is logged in. Its about making a little machine that can take in argument and then spits something else out.

Also, we are not limited to one argument. In the example below you can see our functions can take multiple arguments.



Arguments

Functions can have as many arguments as needed

```
function area(length, width) {
  console.log(length * width);
}
area(9,2); //18


function greet(person1, person2, person3){
  console.log("hi " + person1);
  console.log("hi " + person2);
  console.log("hi " + person3);
}
greet("Harry", "Ron", "Hermione");
```

Here is an example of a function that calculates an area of rectangle, it takes a length and a width and then we just multiply them together console.logs length times width and to call this function we just pass two numbers separated by a comma. The first value corresponds with length and the second value corresponds with width. It just comes down to order. If we switch the two numbers between each other, then 2 would be length and 9 would be width.

We have another example that takes three arguments as *person1*, *person2 and person3* and then it console.logs by saying "hi" to all those three persons. So, lets just try out this code.

```
function greet(person1, person2, person3) {
        console.log("hi " + person1);
        console.log("hi " + person2);
        console.log("hi " + person3);
        }
```

We press *Enter*, and the function gets registered in JavaScript. Then we call the function as below.

```
greet( “Harry”, “Ron”, “Hermione”);
```

Output:

```
hi Harry
hi Ron
hi Hermione
```

One other interesting thing to note is that if we leave an argument while calling our function then that argument will be replaced by the word *undefined* in the output.

```
greet( “Harry”, “Ron”);
```

Output:

```
hi Harry
hi Ron
hi undefined
```

In some other programming languages this might throw an error message but in JavaScript it is fine.

Arguments are one of the important pieces in our function which makes it useful because it is not about shortening our code and repeating that same chunk of code every time, it is also about changing that code depending on inputs.

Lets talk about some examples of where we might use arguments in a real web app. Imagine that we have a web game, and in that game there is a score for every player, and the player can do things that can increment the score or decrement the score. So, if a player does something crazy, we might have a function called *addToScore(),* and if we want to award them points we might pass in 100 points, *addToScore(100),* if a user dies we might subtract 100 points, *addToScore(-100),* and if we want to award few points we might just add like 5 points, *addToScore(5)*

So, here is another example we have a site that has user login, and it might have a function like *checkCredentials(),* and it might just take 2 arguments, an email and a password, it might be like *checkCredentials(“rusty@gmail.com”, “woof123”)* and then *checkCredentials* function will take the email and password and it would check them and make some decisions. There would be an *if* statement in there that would check if those two arguments correspond with each other correctly and if they do then it logs them in, *else* it gives an error message.

The ability of functions to take arguments is one of the most important part of writing functions, its not just about repeating code, its about repeating code that we can also change a little bit and we can have some variables in.