

Return Statements

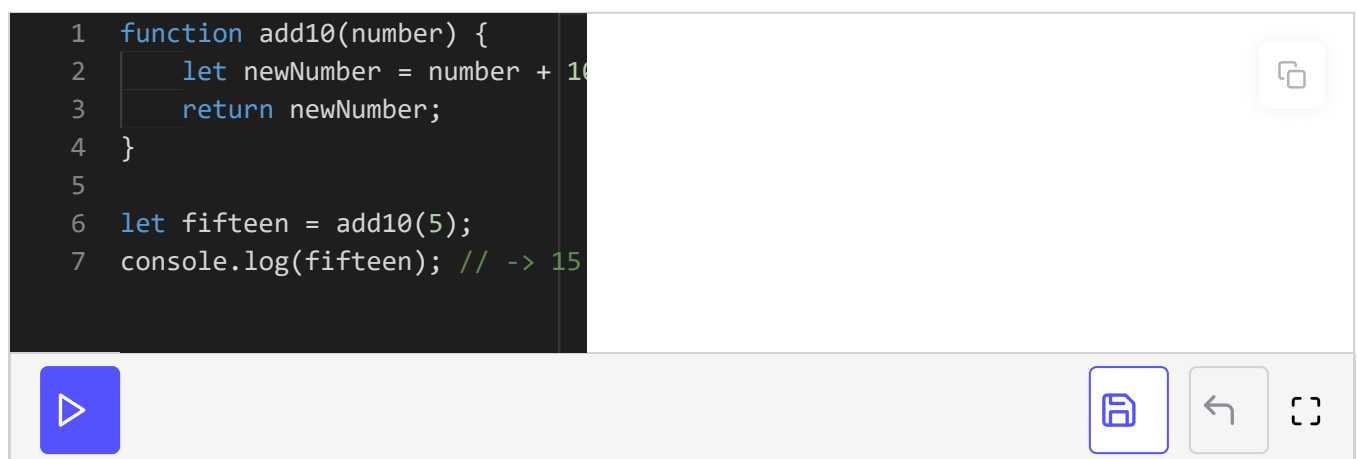
Learn the last powerful part of functions. The return statement allows a function to send information back out. With this final tool, we can master the full power of functions.

Introduction

There's one more powerful feature of functions. The return statement. In addition to accepting values in through arguments, a function can send something back out.

Here's how that works.

```
1 function add10(number) {  
2   let newNumber = number + 10;  
3   return newNumber;  
4 }  
5  
6 let fifteen = add10(5);  
7 console.log(fifteen); // -> 15
```



Breakdown

We give `add10` a value of 5 when we call it. Inside the function, a new variable `newNumber` is created, which is equal to what was passed in plus 10.

We then *return* `newNumber` from the function. Pay attention to line 6. We create a new variable `fifteen` and set it equal to the *function call*. When we do this, **the variable receives the value that the function returns.**

Essentially, on line 6, we're telling JavaScript to pause and jump up to line 1 and call `add10` with out parameter. This function will run to completion. When it's done, the value that is returned will be given to the variable `fifteen` and the engine will continue running where it left off.

While a function can accept multiple arguments, it can only return one value

While a function can accept multiple arguments, it can only return one value.

```
1 function add(num1, num2) {  
2   return num1 + num2;  
3 }  
4  
5 let twenty = add(5, 15);  
6 console.log(twenty); // -> 20
```



Returning **undefined**

If we use a return statement without returning a value, the function will automatically return **undefined**.

```
function add(num1, num2) {  
  let newNumber = num1 + num2;  
  return;  
}  
  
let twenty = add(5, 15);  
console.log(twenty); // -> undefined
```



Stopping a Function

Often, we use return statements even though we don't want anything back from the function.

If a function sees a return statement, it will stop executing. It'll just quit.

```
function print() {  
  console.log('This will print!');  
  return;  
  console.log('This will not print :(');  
}  
  
print(); // -> This will print!
```



This is very useful in conditionals. For example, say we don't want a function

to run if we forget to pass in an argument. We can stop it using an if-statement.

```
function print(item) {  
  if (item === undefined) {  
    console.log('No item was passed in!');  
    return;  
  }  
  
  console.log('The item I was given is:', item);  
}  
  
print(10); // -> The item I was given is: 10  
print(); // -> No item was passed in!
```



As we can see, everything we've learned so far is coming together.

Code Challenges

Feel free to test your understanding.

INSTRUCTIONS

Modify this function so that it takes in two arguments and returns their product.

```
function multiply() {}
```



INSTRUCTIONS

Modify this function so that it takes in a string argument and returns a new string. The new string should be equal to the string passed in, prepended with 'Hello, '.

Example:

```
greet('John'); -> 'Hello, John'
```

```
function greet(name) {}
```



INSTRUCTIONS

Modify this function so that if it's called with 1 or 0 arguments, it returns `null`.

```
function divide(num1, num2) {  
  return num1 / num2;  
}
```

