Exercise on Default Arguments

It's time to play with function arguments. These exercises build further upon what we learned in the previous lesson.

Exercise 1:

Write a function that executes a callback function after a given delay in milliseconds. The default value of delay is one second.

The `setTimeout()` method can be used to specify the time delay before a function is executed.

Explanation:

The main objective of this exercise was to define a default argument for `delay`.

Using ES6 conventions, we can simply state the default value in the function arguments:

```
delay = 1000
```

The statement above sets the delay at 1000. You can define your own delay by passing it into the function. Otherwise, the `executeCallback()` function will execute after 1000 milliseconds.

If we want to use the ES5 syntax, we'd have to write something like this:

```
delay = delay || 1000;
```

We use the built in `setTimeout` method to start the timer and execute our function.

Exercise 2:

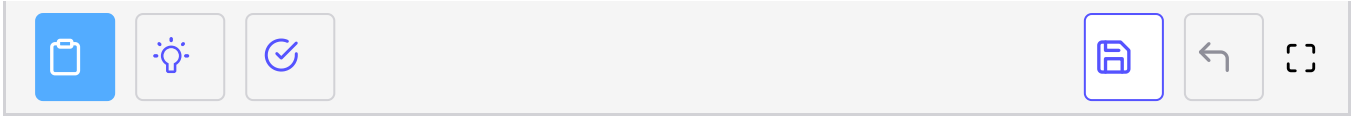
Change the below code such that the second argument of `printComment` has a default value that's initially `1`, and is incremented by `1` after each call.

Select the show console button in the widget below to see your output.

```
function printComment( comment, line ) {
    console.log( line, comment );
}


//Edit above this line

for (var i = 1; i <= 5; i++)
  printComment('I should be lineNumber ' + i);
```



Explanation:

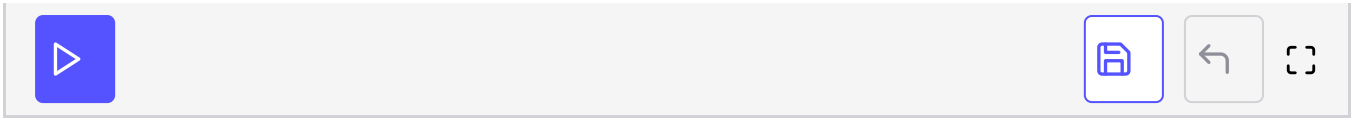
We create a new variable `lineNumber` which is initialized to 1. This way, we are updating a variable rather than changing the default argument of the `printComment` function each time.

Exercise 3:

Determine the values written to the console before executing this script.

```
function argList( productName, price = 100 ) {
    console.log( arguments.length ); //(A)
    console.log( productName === arguments[0] ); //(B)
    console.log( price === arguments[1] ); //(C)
};

argList( 'Krill Oil Capsules' );
```



Explanation:

The answers are fairly simple. Let's look at them one by one.

(A): Even though we have specified that our function can have 2 arguments, we've already learned that it isn't necessary to provide both. In `argList( 'Krill Oil Capsules' );`, we specify the first argument. Hence, the length of our arguments array is `1`.

(B): Here, we are simply checking whether or not the `productName` argument exists. Since, we passed `'Krill Oil Capsules'` into the function, the console will display `true` for this statement.

(C): This is the interesting part. The `price` argument has a default value of 100. However, it is not considered to be an argument which we have passed into the function. As a result, the comparative statement `price === arguments[1]` will return `false` because `arguments[1]` is undefined.