Functions

A function is basically a piece of code which can be reused without having to write it again and again. Let's discover it and learn how to create, refer to and call a function.

The concept of function

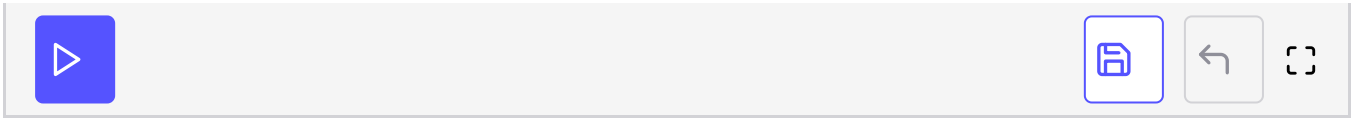
Think of a function like a mathematical function giving you a relationship between input and output variables. If you don't like maths, think of a function like a vending machine. You give it some coins and a number, and it spits out some cold coke.

```
function add( a, b ) {
    return a + b;
}
```

This function definition describes the relationship between its input variables `a` and `b`, and the return value of the function.

The return statement returns the value of the function. When calling the `add` function with arguments `a` and `b`, it computes the value `a+b` and returns it. Example:

```
function add( a, b ) {
  return a + b;
}
console.log(add( 5,2));
```



Try to modify the input variables while making the function call `add(5,2)`. The return value also changes. Try to call the add function with one variable, e.g. `add( 5 )`, and see what happens.

Functions are useful to create reusable chunks of code that you can call with different arguments. We will write more useful functions once you learn the

basics of control structures.

Making functions using a variable as a reference

You can also define functions without placing the name between the `function` keyword and the argument list. This structure is great if you want to create a reference to it using a variable. Remember, the variable `subtract` is a handle to a drawer. This time, your drawer contains a function.

```
let subtract = function( a, b ) {
    return a - b;
}
```

Calling functions

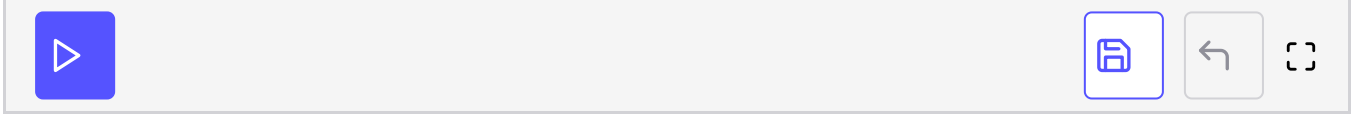
All functions can be called using their references:

```
let add=function( a, b )
{
  return a+b;
}

let subtract=function( a, b ) {
  return a-b;
}

let multiply=function( a, b ) {
  return a*b;
}

let square=function( a ) {
  return a**2;
}
console.log(add( 2,3));
console.log(subtract( 2, 3 ));
console.log(multiply( 2, 3 ));
console.log(square( 2 ));
```

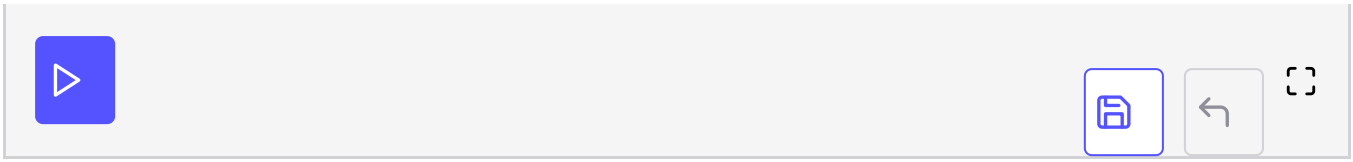


What if the function does not return anything?

When a function does not return anything, its return value becomes `undefined`:

```
let empty = function() {}
```

```
console.log(empty())
```

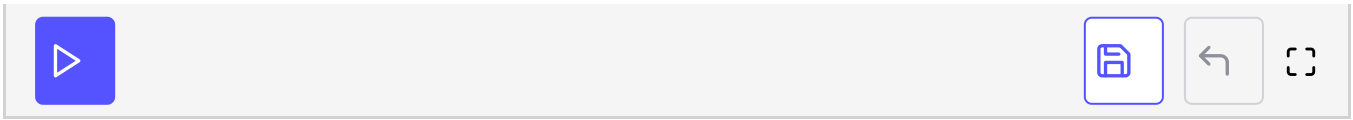


Function Scope

Variables inside the function are not the same as variables outside the function:

```
let a = 5;
function addOne( a ) {
    a = a + 1;
    return a;
}

console.log( addOne( a ))
console.log(a)
```



The `a` variable inside the function is valid inside the *scope* of the function. It *shadows* the variable `a` outside the function. Therefore, adding one to the internal `a` variable does not have any effect on the external value. Therefore, changes in the value of `a` inside the function do not change the value of the variable `a` outside the function.