Functions Are Just Variables

Functions Are Flexible

We'll cover the following

- Exercise

Functions in Javascript are considered to be **first class**, meaning they are treated like any other variable we can make use of. This allows us to store functions as *items* in an array or as *properties* in an object.

Let's see how this plays out in practice. Here we have four variables, each of them functions, that complete some operation on two numbers:

```
var sum = function(x, y){ return x + y; };
var subtract = function(x, y){ return x - y; };
var multiply = function(x, y){ return x * y; };
var divide = function(x, y){ return x / y; };
```

We can use these variables like any other variable type. This means we could store all of these functions in a single array, named `operations`, to clearly indicate the functions' intent.

```
//functions can be stored in an array
var operations = [sum, subtract, multiply, divide];

//functions can be called from an array by accessing them and using the () operator
for(var i = 0; i < operations.length; i++){
  console.log(operations[i](5,10));
}
```

Functions in Javascript are treated like any other variable

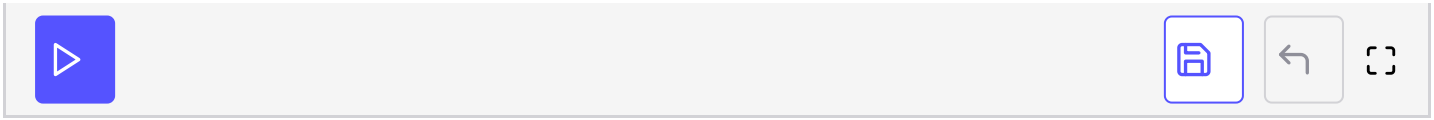
Since Javascript treats functions as just another variable, it is possible to use the `operations` array's functions by accessing them using *bracket notation* (`[]`).

We can then use the `()` operator to **invoke** the function.

This same logic applies to Javascript objects:

```
//functions can be stored in an object as property values
var operations = {
  sum: function(x, y){ return x + y; },
  subtract: function(x, y){ return x - y; },
  multiply: function(x, y){ return x * y; },
  divide: function(x, y){ return x / y; }
};

//functions can be called from an object by accessing a property (dot or bracket)
//and using then the () operator
console.log(operations.multiply(5, 10));
console.log(operations["multiply"](5, 10));
```

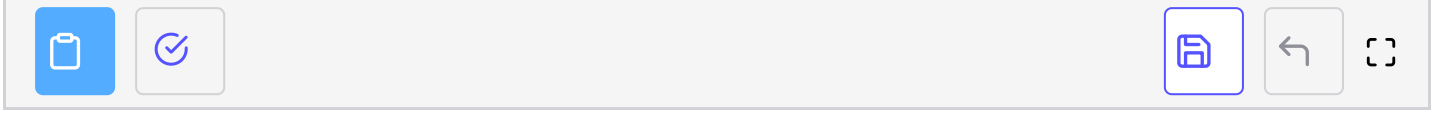


In subsequent lessons, we will encounter something similar to the above example. Often times, functions are stored as *object properties* as the function directly relates to that object in some fashion. We will go over this concept in more detail in the next lesson.

Exercise

Create an additional property (named `modulo`) in the `operations` object that computes the *remainder* of dividing a number by another number.

```
//write your code here
```



Now that you have got hands-on using functions, let's learn about the `this` keyword in the next lesson.