Specifying Function Expressions

In this lesson we will learn how to specify a function expression.
Let's begin!

JavaScript has a number of great things that are built on function expressions. Just like other objects, functions can be assigned to variables:

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

This assignment ensures that you can invoke the function through the variable, just like if it were a statically declared function:

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

console.log(add(12, 23));
```

Function expressions provide great flexibility.

For example, you can define different functions for a certain operation depending on the run time context:

```
var kind = "subtract";
// ...
var op;
if (kind == "add") {
  op = function (a, b) {
    return a + b;
  };
}
else {
  op = function (a, b) {
    return a - b;
  };
}
console.log(op(35, 23)); // 12
```

Here, depending on the value of the kind variable, the `op` variable holds a function that adds or subtracts two numbers. When the operation is called through the `op` variable, the previously set up function is invoked. As you can see, functions defined with function expressions do not need a name.

Because functions are objects, function expressions provide you a way to return a function from a function.

You can wrap the previous logic into a function:

```
var kind = "subtract";
var op = createOperation(kind);
console.log(op(35, 23)); // 12

function createOperation(kind) {
  if (kind == "add") {
    return function (a, b) {
      return a + b;
    };
  }
  else {
    return function (a, b) {

      return a - b;
    };
  }
}
```

Returning function from another function sounds odd only the first time.

If you think it over, all other imperative programming languages provide a way to do that. C and C++ can return function pointers and C# has delegates that can be returned from functions, too.

This mechanism in JavaScript is very powerful, as you will learn soon.

In the *next lesson*, we’ll learn how to use recursive functions!

See you there!