Quiz on Functions and Scope

1. During execution, JS makes the arguments passed to a function available to the function as local vars with the same names as the function's parameters.

2. Statements that are true about the return value of a function.

- If the `return` statement does not include a value, function implicitly returns `undefined`
- If the function does not contain an explicit `return` statement, it implicitly returns `undefined`
- Not true: If a function has `n` parameters, you must call it with n arguments. (n is a non-negative integer.)
  - The number of args passed doesn't have to match the number of parameters, there can be more or fewer
- Not true: Function expressions and function declarations are different ways to define functions in JavaScript. It has no impact in how JavaScript processes code.
  - It has an impact since both have different hoisting rules

3. Statements about hoisting

- Not true: Hoisting is the process of moving code blocks to a different location by JavaScript.
  - It behaves as though code movement occurs since JS defines vars first, then executes the code, it doesn't alter any code
- Not true: JavaScript hoists statements that don't end with a semicolon.
- Hoisting occurs when JavaScript processes variable and function declarations within a scope, before it executes any other code in that scope.
- Not true: Hoisting lets you call a function defined with a function expression before you declare it.
  - Function expressions aren't assigned to a variable while hoisting the var. JS doens't give var a value until it executes the line that assigns the function expression.

4. Var scope statements:

- Scope describes how and where the language finds and retrieves values from declared variables.
- Functions create a new scope in JavaScript.
- Not true: Functions must contain a variable declaration to create a new scope in JavaScript.
- Not true: Variables declared inside code blocks (code enclosed by curly braces {}) have local variable scope. Variables declared inside a block aren't accessible outside the block.
  - Scope isn't defined based on the existence of var. With or without var declarations, functions create a new scope. While code blocks create an inner scope for Ruby, they don't for JS.

5. Lexical scoping statements.

- Lexical scoping determines a variable's scope based on the structure of the code.
- Lexical scoping is also called "static scoping."

10. Which of the following code snippets contains at least one function declaration?

```
var foo = function () {};

function foo() {}; // yes, contains a single function declaration

(function foo() {});

var foo = function () { // yes, contains function declaration nested inside of a function expression
  function bar() {};
};

var foo = function foo() {};

var foo = function () {
  return function bar() {};
};
```

11. Functions statements

- Not true: A function declaration can be used to define an anonymous function. (all function declarations must have a name)
- A function expression can be used to define an anonymous function.
- A function declaration that exists on line 1 of a program will always create at least one new variable.
- Not true: A function expression that exists on line 1 of a program will always create at least one new variable. (function expression can be used to define an anonymous function without creating any var)