PLP-3: functions in javascript

For your write-up, address the following items:

1. What is the syntax for declaring a function in your language?
    a. In javascript, the fucntion is defined with the function keyword, followed by a name, followed by parentheses (). Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
2. Are there any rules about where the function has to be placed in your code file so that it can run?
    a. Functions must be in scope when they are called, but the function declaration can be hoisted (appear below the call in the code). The scope of a function declaration is the function in which it is declared (or the entire program, if it is declared at the top level).
3. Does your language support recursive functions?
    a. Yes! It allows for creating a function that keeps calling itself but with a smaller input every consecutive time until the code's desired result from the start is achieved.
4. Can functions in your language accept multiple parameters? Can they be of different data types?
    a. JavaScript functions can accept any type as parameters.
5. Can functions in your language return multiple values at the same time? How is that implemented? If not, are there ways around that problem? What are they?
    a. NO, multiple values cannot be returned from a function in JavaScript but multiple values can be stored in an array or in an object and then the array or the object can be returned.
6. Is your language pass-by reference or value? Check your code against outside sources in case there is anything tricky going on (like in Perl).
    a. Javascript is pass-by-value; when a variable is assigned to another variable, the value stored in the variable is copied into the new variable. They are independent of each other, each occupying its own memory space.
7. Where are the arguments, parameters and local variables stored (value-on-stack, ref-to-heap-on stack) during execution?
    a. V**ariables** are typically **stored** in the **stack** or the **heap**.
8. What are the scoping rules in your language (visibility and lifetime of variables before, during and after code blocks)?
    a. **Global Scope**: Accessible anywhere; lasts for the application's lifetime.

    b. **Function Scope**: Accessible only within the function; lasts while the function is executing.
    c. **Block Scope**: Accessible only within the block; lasts while the block is executed.
    d. **Hoisting**: Variable declarations are hoisted, but initializations are not.
    e. **Lexical Scoping**: Inner functions have access to outer function variables.
9. Are side-effects possible? Are there guard rails against side-effects?
    a. Side effects in JavaScript occur when a function modifies state outside its scope, such as changing global variables, altering DOM elements, or logging to the console. To minimize side effects, developers can employ practices like writing pure functions (which always return the same output for the same inputs without modifying external state), using encapsulation to manage state, adopting functional programming principles, and utilizing immutable data structures. Additionally, implementing const and let, conducting code reviews, and writing tests can further safeguard against unintended side effects, promoting more predictable and maintainable code.
10. Where are local variable values stored? (on the stack? On the heap?)
    a. They are stored in the stack. Object types (Array, Function, and Object) are stored in the heap. The variable in the stack will contain the address (reference) to the location in heap memory where the data is stored.
11. Are there any other aspects of functions in your language that aren't specifically asked about here, but that are important to know in order to write one? What are they? (e.g. dynamic vs static scope)

Sources:

- https://www.w3schools.com/js/js_functions.asp
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions#:~:text=Functions%20must%20be%20in%20scope,declared%20at%20the%20top%20level).