Introduction to JavaScript

Scope

Print cheatsheet

Scope

Scope is a concept that refers to where values and functions can be accessed.

Various scopes include:

- *Global* scope (a value/function in the global scope can be used anywhere in the entire program)
- *File* or *module* scope (the value/function can only be accessed from within the file)
- *Function* scope (only visible within the function),
- *Code block* scope (only visible within a `{ ... }` codeblock)

```
function myFunction() {

  var pizzaName = "Volvo";
  // Code here can use pizzaName

}

// Code here can't use pizzaName
```

Block Scoped Variables

`const` and `let` are *block scoped* variables, meaning they are only accessible in their block or nested blocks. In the given code block, trying to print the `statusMessage` using the `console.log()` method will result in a `ReferenceError`. It is accessible only inside that `if` block.

```
const isLoggedIn = true;

if (isLoggedIn == true) {
  const statusMessage = 'User is logged in.';
}

console.log(statusMessage);

// Uncaught ReferenceError: statusMessage is not defined
```

Global Variables

JavaScript variables that are declared outside of blocks or functions can exist in the *global scope*, which means they are accessible throughout a program. Variables declared outside of smaller block or function scopes are accessible inside those smaller scopes.

Note: It is best practice to keep global variables to a minimum.

```
// Variable declared globally
const color = 'blue';

function printColor() {
  console.log(color);
}

printColor(); // Prints: blue
```