Scope

..."Scope your variables tightly..."

Definition: Scope defines where variables can be accessed or referenced.

- While some variables can be accessed from anywhere within a program, other variables may only be available in a specific context.

"You can think of scope like the view of the night sky from your window. Everyone who lives on the planet Earth is in the global scope of the stars. The stars are accessible globally. Meanwhile, if you live in a city, you may see the city skyline or the river. The skyline and river are only accessible locally in your city, but you can still see the stars that are available globally."

~Codecademy

BLOCKS and SCOPE

A "BLOCK" is the code found inside of a set of {}

```
const logSkyColor = () =>
// Block start
{
  let color = 'blue';
  console.log(color); // blue
}
```

EXAMPLE:

```
//Can be accessed globally in all functions
const city = "New York City";
function logCitySkyline(){
  //can only be accessed within the function
  let skyscraper = "Empire State Building";
  return "The stars over the " + skyscraper + " in " + city;
}
console.log(logCitySkyline());
```

GLOBAL SCOPE

- In *global scope*, variables are declared outside of blocks; called *global variables*.

- When a variable is defined inside a block, it is only accessible to the code within the curly braces {}.
- Variables that are declared with block scope are known as *local variables.*

EXAMPLE:

```
const logSkyColor = () => {
  let color = 'blue';
  console.log(color); // blue
};

logSkyColor(); // blue
console.log(color); // ReferenceError
```

SCOPE POLLUTION

- When you declare global variables, they go to the *global namespace*.
- The global namespace allows the variables to be accessible from anywhere in the program.
- Variables remain in the global namespace until the program completes, so it could get over populated easily.
- *Scope pollution* is when we have too many global variables that exist in the global namespace, or when we reuse variables across different scopes.
- **While it's important to know what global scope is, it's best practice to not define variables in the global scope.**

EXAMPLE

```
let num = 50;

const logNum = () => {
  num = 100; // Take note of this line of code
  console.log(num);
};

logNum(); // Prints 100
console.log(num); // Prints 100
```

- Defining variables mostly within blocks is good because: It will save memory in your code because it will cease to exist after the block finishes running.
- However, duplicate code can indicate that a variable may be scoped too tightly.

EXAMPLE:

TERMINOLOGY AND VOCABULARY

1. **Scope** is the idea in programming that some variables are accessible/inaccessible from other parts of the program.
2. **Blocks** are statements that exist within curly braces {}.
3. **Global scope** refers to the context within which variables are accessible to every part of the program.