HOISTING - FUNCTIONS

Interviewquestion-144

Follow on

@Duvvuru Kishore

HOISTING

When variables and functions are declared, javascript allocates space to variables and functions in the memory even before its executed

HOISTING WITH REGULAR FUNCTIONS

Regular functions defined using the function keyword are hoisted entirely. This means you can call a function before its declaration in the code.

```
console.log(add(2, 3)); // Outputs: 5

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

HOISTING WITH ARROW FUNCTIONS

Arrow functions are often assigned to variables. When an arrow function is assigned to a variable declared with **let or const**, the variable is hoisted, but it remains uninitialized until the line where it is defined.

This leads to a "**temporal dead zone**," during which the arrow function cannot be accessed or called until the code execution reaches its declaration. Therefore, attempting to use the function before its definition will result in a **ReferenceError**

```
console.log(add(2, 3));
// ReferenceError: Cannot access 'add' before initialization

const add = (a, b) => a + b;
```

Understanding these differences is crucial for writing predictable JavaScript code, as it helps avoid common pitfalls related to function and variable usage before their declarations.