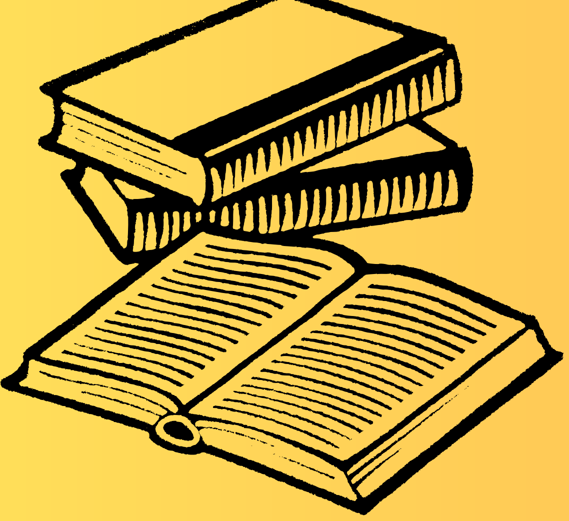
HOISTING IN JAVASCRIPT

@AYUSHAGARWAL15



Have you ever wondered why you can sometimes use variables or functions before declaring them in your JavaScript code? It's all thanks to hoisting!

1. Variable Hoisting

Consider this example:

```
console.log(apple);
 // Output: undefined
var apple = "Delicious fruit!";
console.log(apple);
 // Output: Delicious fruit!
```

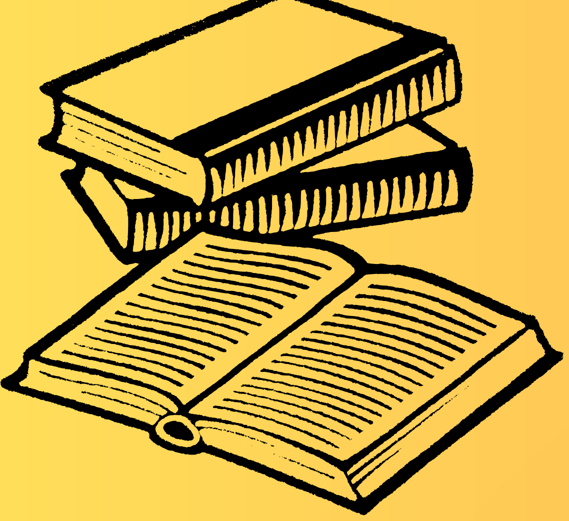
The variable declaration (var apple) is hoisted to the top, allowing us to use apple before it's officially declared. The first console.log outputs undefined because only the declaration is hoisted, not the initialization.

2. Function Hoisting

Now, let's explore function hoisting:

```
greet();
// Output: Hello, hoisting!

function greet() {
console.log("Hello, hoisting!");
}
```



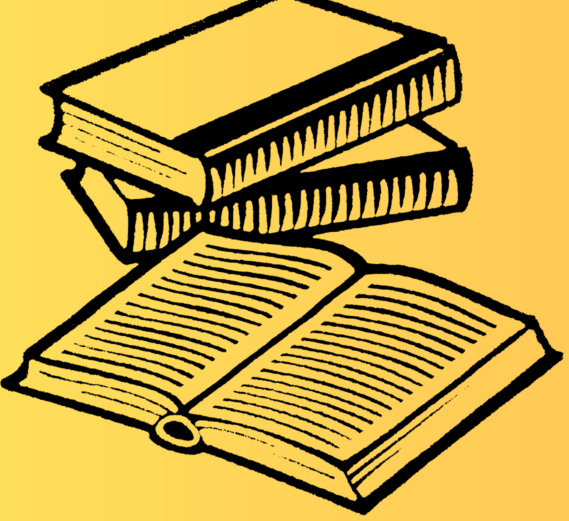
Similar to variables, function declarations are hoisted, allowing us to call the function before its actual declaration.

@AYUSHAGARWAL15

3. let & const Hoisting

However, it's crucial to note that let and const behave a bit differently:

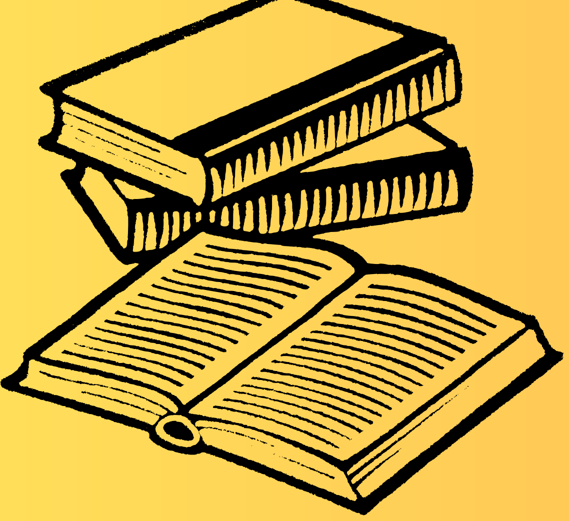
```
console.log(orange);
// Output: ReferenceError: Cannot access 'orange' before initialization
let orange = "Vibrant citrus!;
```



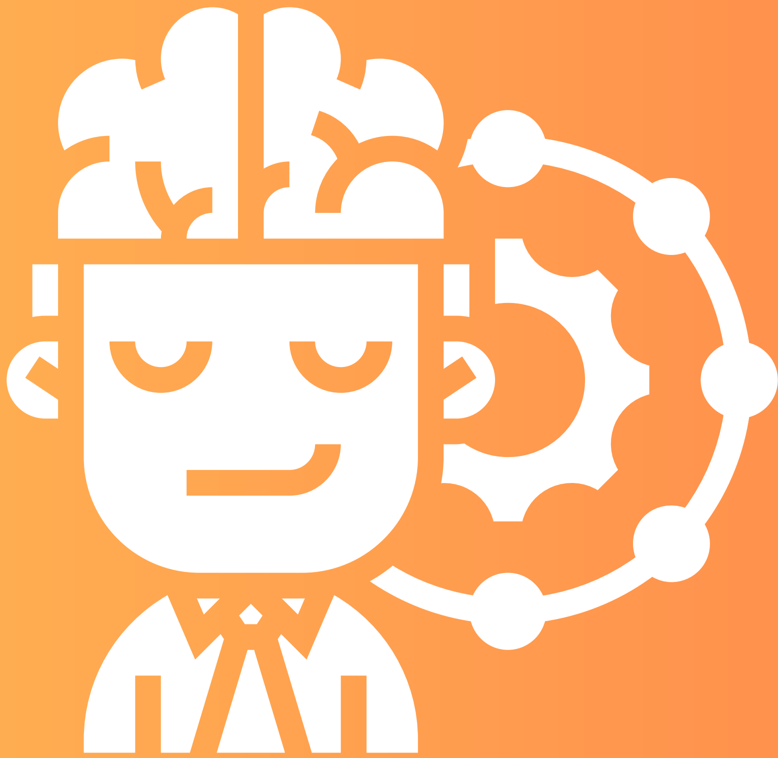
Accessing a let variable before its declaration results in a ReferenceError, indicating that the variable has not been initialized.

```
console.log(grape);
// Output: SyntaxError: Missing initializer
in const declaration
const grape;
```

For const variables, attempting to access them before declaration leads to a SyntaxError due to the missing initializer.



Variables declared with let and const are also hoisted, but not initialized (unlike var).



In summary, while hoisting allows us to use variables and functions before their declarations, the behavior differs between var, let, and const. Understanding hoisting can help you write cleaner and more predictable JavaScript code. Happy coding!

FOLLOW

AYUSH
AGARWAL

@AYUSHAGARWAL15