Day 1: Let and Const

by AvmnuSng

Problem Submissions Leaderboard Discussions Editorial Tutorial

Variable Declaration Keywords

var

We use the *var* keyword to declare variables. The scope of a variable declared using this keyword is within the context wherever it was declared. For variables declared outside any function, this means they are globally available throughout the program. For variables declared within a function, this means they are only available within the function itself.

- EXAMPLE

```
"use strict"
process.stdin.on('data', function (data) {
    main(+(data));
});
/**** Ignore above this line. ****/

function main(input) {
    var a = input;

    // If 'a' is odd:
    if (a % 2 == 1) {
        var a = input + 1;
        console.log("if(a): " + a);
    }

    console.log("main(a): " + a);
}
```

Input

11

Run

Output

Table Of Contents

var

let

const

JavaScript `const` Keyword

Solution

Click *Run* above to execute the given code. It works in the following way:

1. Variable a is declared in the *main* function using the *var* keyword and initialized with the given value, 11.
2. $a \% 1$ evaluates to *true* because $a = 11$ is odd, so we enter the *if* block.
3. Variable a is declared a second time inside the *if* block (still using the *var* keyword) and initialized with a value of $11 + 1 = 12$. We print the value of $a = 12$.
4. We exit the *if* block and print the value of a in *main*. This value is 12 because the scope of the initial declaration of a in *main* includes the *if* block.

Go to Top

`let`

We use the *let* keyword to declare variables that are limited in scope to the block, statement, or expression in which they are used. This is unlike the *var* keyword, which defines a variable globally or locally to an entire function regardless of block scope.

\- EXAMPLE

```
"use strict"
process.stdin.on('data', function (data) {
    main(+(data));
});
/**** Ignore above this line. ****/

function main(input) {
    let a = input;

    // If 'a' is odd:
    if (a % 2 == 1) {
        // Increment 'a' by 1
        let a = input + 1;
        console.log("if(a): " + a);
    }

    console.log("main(a): " + a);
}
```

Input

```
11
```

Run

Output

Solution

Click *Run* above to execute the given code. It works in the following way:

1. Variable a is declared in the *main* function using the *let* keyword and initialized

with the given value, 11.

2. $a \% 1$ evaluates to *true* because $a = 11$ is odd, so we enter the *if* block.
3. Variable a is declared a second time inside the *if* block (again using the *let* keyword) and initialized with a value of $11 + 1 = 12$. We print the value of $a = 12$.
4. We exit the *if* block and print the value of a in *main*. Because we used the *let* keyword for both declarations and the scope of the second declaration of a was limited to the *if* block, the value of a in *main* is still 11.

It's important to note that you cannot redeclare a variable declared using the *let* keyword within the same scope as the original variable. An attempt to do this raises an *Error*, as demonstrated by the code below.

```
"use strict"
process.stdin.on('data', function (data) {
    main(+(data));
});
/**** Ignore above this line. ****/

function main(input) {
    let a = input;

    // This will throw "SyntaxError: Identifier 'a' has already be
    let a = input + 1;

    console.log(a);
}
```

Input

Run

Output

const

We use the *const* keyword to create a *read-only* reference to a value, meaning the value referenced by this variable cannot be reassigned. Because the value referenced by a constant variable cannot be reassigned, JavaScript *requires* that constant variables always be initialized.

\- EXAMPLE

Click *Run* below to see what happens when you declare a constant variable without initializing it.

```
"use strict"
process.stdin.on('data', function (data) {
    main(+(data));
});
/**** Ignore above this line. ****/

function main(input) {
    const a = input;

    // This will throw "SyntaxError: Missing initializer in const
    const b;

    console.log(a);
}
```

Input

11

Run

Output