JavaScript Variables and Constants

JavaScript Variables

A JavaScript variable is a container for storing data. For example,

```
let num = 5;
```

Here, `num` is a variable that stores the number 5.

Declare Variables in JavaScript

In JavaScript, we use the `var` or `let` keywords to declare variables. For example,

```
var age;
let name;
```

Here, `age` and `name` are variables.

What is the difference between var and let?

Both `var` and `let` are used to declare variables. However, there are some differences between them.

var	let
`var` is used in older versions of JavaScript.	`let` is the new way of declaring variables, starting with ES6 (ES2015).
Variables created with `var` are function-scoped, meaning they can be accessed anywhere within the function they were defined in.	Variables declared with `let` are block-scoped, meaning they can only be accessed within the block where they were declared.
For example, `var x;`	For example, `let y;`

To learn more, visit JavaScript let Vs var.

> Note: It is recommended we use `let` instead of `var`. However, there are a few browsers that do not support `let`. To learn more, visit JavaScript let browser support.

Initialize Variables in JavaScript

We use the assignment operator `=` to assign a value to a variable.

```
// declare variable num
let num;

// assign 5 to num
num = 5;
```

Here, 5 is assigned to the variable `num`.

You can also initialize variables during its declaration.

```
// declare variable num1 and assign 5 to it
let num1 = 5;

// declare variable num2 and assign 6 to it
let num2 = 6;
```

Declare multiple variables in a single statement.

In JavaScript, it's possible to declare multiple variables in a single statement.

```
// declare variables num1, num2, and num3
// assign values 5, 6, and 7 respectively
```

```
let num1 = 5, num2 = 6, num3 = 7;
```

Here, we have declared and assigned values to three variables in a single line:

- The value assigned to `num1` is 5.
- The value assigned to `num2` is 6.
- The value assigned to `num3` is 7.

Use a variable without initializing it.

Change the Value of Variables

The value of a variable may vary. Hence, the name variable.

Let's look at the example below to learn how to change the value of a variable:

```
// assign 5 to variable score
let score = 5;
console.log(score); // 5

// change the value of score to 3
score = 3;
console.log(score); // 3
```

Run Code

Here, the value of the `score` variable is changed from 5 to 3 when we assign a new value to it.

Rules for Naming JavaScript Variables

- Variable names must start with a letter, an underscore _, or the dollar sign `$`. For example,

```
// valid
let message = "hello";
let _message = "hello";
let $message = "hello";
```

- Variables cannot start with numbers. For example,

```
// invalid
let 1message = "hello"; // this gives an error
```

- Variable names are case-sensitive. So `age` and `Age` are different variables. For example,

```
let age = 23;
let Age = 20;

console.log(age); // 23
console.log(Age); // 20
```

Run Code

- Variable names cannot be keywords (special words reserved for specific purposes in JavaScript such as `if`, `else`, `let`, `var`, etc.). For example,

```
//invalid
let new = 5; // Error! new is a keyword
```

Recommended ways to name a variable in JavaScript.

You can name the variables any way you want. However, we recommend you use the following naming conventions:

- In JavaScript, variables are generally named in camelCase format if they have multiple words.

  For example, `firstName`, `annualSalary`, `numberOfBooks`, etc.
- It's a good practice to give a descriptive name to a variable.

  For example, if you are using a variable to store the number of apples, it is better to name that variable `apples` or `numberOfApples` rather than `x` or `n`.

JavaScript Constants

A constant is a type of variable whose value cannot be changed.

In JavaScript, we use the `const` keyword to create constants. For example,

```
// assign 5 to num
const num = 5;
```

Once a constant is initialized, we cannot change its value.

```
// assign 5 to num
const num = 5;

// assign 10 to num
num = 10;
console.log(num) // Error! constant cannot be changed
```

Run Code

Always Initialize a Constant During Declaration

If you do not initialize a constant at the time of declaration, it throws an error.

For example,

```
// Error! Missing initializer in const declaration
const x;

// attempt to initialize constant after declaration
x = 5;

console.log(x)
```

Run Code

Note: If you are sure that the value of a variable won't change throughout the program, we recommend you use `const`.

However, there are a few browsers that do not support `const`. Visit JavaScript const browser support to learn more.

Also Read

- JavaScript Data Types

Before we wrap up, let’s put your knowledge of JavaScript Variables and Constants to the test! Can you solve the following challenge?

Challenge:

Write a function to calculate the product of two numbers.

- Return the product of two numbers `num1` and `num2`.
- For example, if `num1 = 5` and `num2 = 3`, the expected output is 15.

```
function calculateProduct(num1, num2) {



}
```

Check Code

JavaScript console.log()

In JavaScript, the `console.log()` method displays messages or variables in the browser's console.

Here's a quick example of `console.log()`. You can read the rest of the tutorial for more details.

Example

```
let message = "Hello, JavaScript!";
console.log(message);
```

```
// Output: Hello, JavaScript!
```

Run Code

When we run the above code, `Hello, JavaScript!` is printed on the console.

Syntax of JavaScript console.log()

```
console.log(message);
```

Here, `message` is a value or a variable whose value is to be printed to the console.

Example 1: JavaScript console.log() Method

```
console.log("Good Morning!");
console.log(2000);
```

Run Code

Output

```
Good Morning!
2000
```

Here,

- `console.log("Good Morning!")` prints the string `"Good Morning!"` to the console.
- `console.log(2000)` prints the number 2000 to the console.

Example 2: Print Values Stored in Variables

We can also use `console.log()` to display the values stored in variables. For example,

```
// store value in greet variable
const greet = "Hello";

// print the value of greet variable
console.log(greet);
```

Run Code

Output

```
Hello
```

In this example, we have used `console.log()` to print the value of the `greet` variable, which is set to the string `"Hello"`.