JavaScript Variables

Summary: in this tutorial, you'll learn about JavaScript variables and how to use variables to store values in the application.

A variable is a label that references a value like a number or string. Before using a variable, you need to declare it.

Declaring a variable

To declare a variable, you use the `var` keyword followed by the variable name as follows:

```
var variableName;
```

A variable name can be any valid identifier. For example:

```
var message;
```

By default, a variable has a special value `undefined` if you don't assign it a value.

In JavaScript, variable names follow these rules:

- Variable names are case-sensitive. This means that the `message` and `Message` are different variables.
- Variable names can only contain letters, numbers, underscores, or dollar signs and cannot contain spaces. Also, variable names must begin with a letter, an underscore (`_`) or a dollar sign (`$)` .
- Variable names cannot use the reserved words.

By convention, variable names use `camelCase` like `message` , `yourAge` , and `myName` .

JavaScript is a dynamically typed language. This means that you don't need to explitly specify the variable's type in the declaration like other static-typed languages such as Java or C#.

Starting in ES6, you can use the `let` keyword to declare a variable like this:

```
let message;
```

It's a good practice to use the `let` keyword to declare a variable.

Later, you'll learn the differences between `var` and `let` keywords. For now, you should not worry about it.

Initializing a variable

After declaring a variable, you can initialize it with a value. To initialize a variable, you specify the variable name, followed by an equals sign (`=`) and a value:

```
variableName = initialValue;
```

For example, The following declares the `message` variable and initializes it with a literal string `"Hello"` :

```
let message;
message = "Hello";
```

To declare and initialize a variable at the same time, you use the following syntax:

```
let variableName = value;
```

For example, the following statement declares the `message` variable and initializes it with the literal string `"Hello"` :

```
let message = "Hello";
```

JavaScript allows you to declare two or more variables using a single statement. To separate two variable declarations, you use a comma (`,`) like this:

```
let message = "Hello",
    counter = 100;
```

Since JavaScript is a dynamically typed language, you can assign a value of a different type to a variable. Although, it is not recommended. For example:

```
let message = "Hello";
message = 100;
```

In this example, we initialize the `message` variable with a literal string `"Hello"` to the messaage variable. Then, we assign a number `100` to it.

Changing a variable

After declaring or initializing a variable, you can change its value by setting a different value. For example:

```
let message = "Hello";
message = 'Bye';
```

Undefined vs. undeclared variables

It's important to distinguish between undefined and undeclared variables.

An undefined variable is a variable that has been declared but has not been initialized with a value. For example:

```
let message;
console.log(message); // undefined
```

In this example, the `message` variable is declared but not initialized. Therefore, the `message` variable is undefined.

In contrast, an undeclared variable is a variable that has not been declared. For example:

```
console.log(counter);
```

Output:

```
console.log(counter);
            ^
ReferenceError: counter is not defined
```

In this example, the `counter` variable has not been declared. Hence, accessing it causes a `ReferenceError` .

Constants

A constant holds a value that doesn't change. To declare a constant, you use the `const` keyword. When defining a constant, you need to initialize it with a value immediately. For example:

```
const workday = 5;
```

Once you define a constant, you cannot change its value.

The following example attempts to change the value of the workday constant to 4 and causes an error:

```
workday = 2;
```

Error:

```
Uncaught TypeError: Assignment to constant variable.
```

Later, you'll learn that the `const` keyword actually defines a read-only reference to a value in the constants tutorial.

Sometimes, you'll see the constant names in uppercase like this:

```
const PORT = 3000;
```

This is a community convention, not a rule.

Summary

- A variable is a label that references a value.
- Use the `let` keyword to declare a variable.
- An undefined variable is a variable that has been declared but not initialized while an undeclared variable is a variable that has not been declared.
- Use the `const` keyword to define a readonly reference to a value.

Quiz