JavaScript do...while Loop

Summary: in this tutorial, you will learn how to use the JavaScript `do...while` statement to create a loop that executes a block until a condition is `false`.

Introduction to the JavaScript do...while statement

The `do...while` loop statement creates a loop that executes a block until a condition evaluates to `false`.

Here's the syntax of the `do...while` loop:

```
do {
  statement;
} while(expression);
```

Unlike the `while` loop, the `do-while` loop always executes the `statement` at least *once* before evaluating the `expression`.

Since the `do...while` loop evaluates expression after each iteration, it's often referred to as a post-test loop.

Inside the loop body, you need to make changes to some variables to ensure that the `expression` is `false` after some iterations. Otherwise, you'll have an indefinite loop.

Note that starting with ES6+, the trailing semicolon (`;`) after the `while(expression)` is optional. So you can use the following syntax:

```
do {
  statement;
} while(expression)
```

The following flowchart illustrates the `do-while` loop statement:

In practice, you often use the `do...while` statement when you want to perform at least one iteration before checking the condition.

JavaScript do while statement examples

Let's take some examples of using the `do...while` statement.

Basic JavaScript do while statement example

The following example uses the `do...while` statement to output five numbers from 0 to 4 to the console:

```
let count = 0;
do {
  console.log(count);
  count++;
} while (count < 5)
```

Output:

```
0
1
2
3
4
```

In this example:

- First, declare and initialize the `count` variable to zero.
- Second, show the `count` and increase its value by one in each iteration until its value is greater or equal to 5.

Using the do while statement to make a number-guessing game

The following example uses the `do...while` statement to create a number guessing game.

The script generates a random integer between 1 and 10. You have to make a number of guesses until your number matches the random number.

```
// generate a secret number between 1 and 10
const MIN = 1;
const MAX = 10;

let secretNumber = Math.floor(Math.random() * (MAX - MIN + 1)) + MIN;

let guesses = 0; // for storing the number of guesses
let hint = ''; // for storing hint
let number = 0;
do {
  // get input from user
  let input = prompt(`Please enter a number between ${MIN} and ${MAX}` + hint);

  // get the integer
  number = parseInt(input);

  // increase the number of guesses
  guesses++;

  // check input number with the secret number provide hint if needed
  if (number > secretNumber) {
```

```
    hint = ', and less than ' + number;
  } else if (number < secretNumber) {
    hint = ', and greater than ' + number;
  } else if (number == secretNumber) {
    alert(`Bravo! you're correct after ${guesses} guess(es).`);
  }
} while (number != secretNumber);
```

How it works.

First, declare the `MIN` and `MAX` constants and initialize their values to 1 and 10:

```
const MIN = 1;
const MAX = 10;
```

Second, use `Math.random()` function to generate a random floating-point number with the value in the range of 0 and 1 (inclusive of zero but not one).

To generate a random number between `MIN` and `MAX` (exclusive), you use the following expression:

```
Math.random() * (MAX - MIN + 1)
```

However, the result is a floating-point number. Therefore, you need to use the `Math.floor()` function to convert it to an integer:

```
Math.floor(Math.random() * (MAX - MIN + 1))
```

To generate a random number between min and max, you use the following expression:

```
let secretNumber = Math.floor(Math.random() * (MAX - MIN + 1)) + MIN;
```

Third, define three variables for storing the number of guesses, hints, and user's input number:

```
let guesses = 0; // for storing the number of guesses
let hint = ''; // for storing hint
let number = 0;
```

Fourth, use the `prompt()` function to get the input from the user:

```
let input = prompt(`Please enter a number between ${MIN} and ${MAX}` + hint);
```

Note that the `prompt()` function only works on web browsers. If you run the code in another environment such as Node.js, please check the corresponding function.

The `prompt()` function returns a string, therefore, you need to use the `parseInt()` function to convert it to an integer:

```
number = parseInt(input);
```

Fifth, increment the number of guesses in each iteration:

```
guesses++;
```

Sixth, check the input number with the secret number (random) number and give a hint. If the numbers are matched, show the message using the `alert()` function:

```
if (number > secretNumber) {
  hint = ', and less than ' + number;
} else if (number < secretNumber) {
  hint = ', and greater than ' + number;
} else if (number == secretNumber) {
  alert(`Bravo! you're correct after ${guesses} guess(es).`);
}
```

Seventh, perform the next iteration until the number matches the secret number.

```
while (number != secretNumber);
```

Summary

- Use the `do while` statement to create a loop that executes a code block until a condition is `false` .

Quiz