JavaScript break

Summary: in this tutorial, you'll learn how to use the JavaScript `break` statement to terminate a loop prematurely.

The label statement

In JavaScript, you can label a statement for later use. Here's the syntax of the `label` statement:

```
label: statement;
```

In this syntax, the label can be any valid identifier. For example, the following shows how to label a `for` loop using the `outer` label:

```
outer: for (let i = 0; i < 5; i++) {
    console.log(i);
}
```

Once you define a label, you can reference it in the `break` or `continue` statement.

Introduction to JavaScript break statement

The `break` statement prematurely terminates a loop such as `for`, `do...while`, and `while` loop, a `switch`, or a `label` statement. Here's the syntax of the `break` statement:

```
break [label];
```

In this syntax, the `label` is optional if you use the `break` statement in a loop or `switch`. However, if you use the `break` statement with a label statement, you need to specify it.

This tutorial focuses on how to use the `break` statement to terminate the loop prematurely.

Using JavaScript break statement in a for loop

The following `for` loop statement outputs five numbers from `0` to `4`:

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

Output:

```
0
1
2
3
4
```

To terminate the `for` loop prematurely, you can use a `break` statement. For example, the following illustrates how to use a `break` statement inside a `for` loop:

```
for (let i = 0; i < 5; i++) {
  console.log(i);
  if (i == 2) {
    break;
  }
}
```

Output:

```
0
1
2
```

In this example, we use an `if` statement inside the loop. If the current value of `i` is `2`, the `if` statement executes the `break` statement that terminates the loop.

This flowchart illustrates how the `break` statement works in a `for` loop:

Using the break statement to terminate a nested loop

A nested loop has one loop inside another. For example, the following uses a nested `for` loop to output a pair of numbers from `1` to `3`:

```
for (let i = 1; i <= 3; i++) {
  for (let j = 1; j <= 3; j++) {
    console.log(i, j);
  }
}
```

Output:

```
1 1
1 2
1 3
```

```
2 1
2 2
2 3
3 1
3 2
3 3
```

If you use a `break` statement inside an inner loop, it only terminates the enclosing loop. For example:

```
for (let i = 1; i <= 3; i++) {
  for (let j = 1; j <= 3; j++) {
    if (i + j == 4) {
      break;
    }
    console.log(i, j);
  }
}
```

Output:

```
1 1
1 2
2 1
```

In this example, if the sum of `i` and `j` is `4`, the `break` statement terminates the inner loop. To terminate the nested loop, you use a label statement. For example:

```
outer: for (let i = 1; i <= 3; i++) {
  for (let j = 1; j <= 3; j++) {
    if (i + j == 4) {
      break outer;
    }
    console.log(i, j);
  }
}
```

Output:

```
1 1
1 2
```

In this example, we label the outer loop with the label `outer`. Inside the inner loop, we specify the `outer` label in the `break` statement. The `break` statement to terminate the nested loop if the sum of `i` and `j` is `4`.

Using JavaScript break statement in a while loop

The following output five numbers from 1 to 5 to the console using a `while` loop:

```
let i = 0;

while (i < 5) {
  i++;
  console.log(i);
}
```

Output:

```
1
2
3
4
5
```

Like a `for` loop, the `break` statement terminates a `while` loop prematurely. For example:

```
let i = 0;

while (i < 5) {
  i++;
  console.log(i);
  if (i == 3) {
    break;
  }
}
```

Output:

```
1
2
3
```

In this example, when the current value of `i` is `3`, the `break` statement terminates the loop. Therefore, you see only three numbers in the output.

The following flowchart illustrates how the break statement works in a `while` loop:

Using JavaScript break statement in a do...while loop

The following example uses a `do...while` statement to output five numbers from 0 to 5 to the console:

```
let i = 0;

do {
  i++;
  console.log(i);
} while (i < 5);
```

Output:

```
1
2
3
4
5
```

Like a `while` loop, you can use a `break` statement to terminate a `do...while` loop. For example:

```
let i = 0;

do {
  i++;
  console.log(i);
  if (i == 3) {
    break;
  }
} while (i < 5);
```

Output:

```
1
2
3
```

The following flowchart shows how the break statement works in a `do while` loop:

Summary

- Use the `break` statement to terminate a loop including `for`, `while`, and `do...while` prematurely.
- The `break` statement terminates the enclosing loop in a nested loop. To terminate the nested loop, you use a label statement.

Quiz