Break and Continue

In this lesson, you'll get to know about the break and continue keywords.

We are almost done. Just two small constructs.

Introduction

We can use `break` or `continue` inside the loops:

- `break` exits the closest loop it is in, and continues with the next command,
- `continue` skips out from the loop body, and jumps back to the condition of the loop.

Example: sum every second element of the `numbers` array:

```
let numbers = [19, 65, 9, 17, 4, 1, 2, 6, 1, 9, 9, 2, 1];

function sumArray( values ) {
    let sum = 0;
    for ( let i in values ) {
        if ( i % 2 == 0 ) continue;
        sum += values[i];
    }
    return sum;
}

console.log("The sum is", sumArray( numbers ));
```

Continue

This example is artificial, because we could have written `i += 2` in a simple for loop to jump to every second value. So we are testing `continue` just for the sake of the example. Whenever `i` is even, continue moves execution back to the next iteration of `i in values`.

Notice that we don't have to use braces around just one statement in the code if it is followed by an `if` statement or a loop:

```
if ( i % 2 == 0 ) continue;
```

is the same as

```
if ( i % 2 == 0 ) {
    continue;
}
```

We still prefer braces, because wrong code formatting may lead to many sources of error. For example, some people might think that `statement1` and `statement2` belongs to the loop body.

```
while ( condition )
    statement1;
    statement2;
```

Wrong!

If we added the braces, we would get the following code:

```
while ( condition ) {
    statement1;
}
statement2;
```

This must be very confusing for people familiar with Python. `statement2` is outside the loop.

Break

You already know what a `break` statement looks like, because we learned it when dealing with the `switch` statement. It is doing the same thing in loops. Suppose we want to break out from the loop whenever the sum exceeds 100:

```
let numbers = [19, 65, 9, 17, 4, 1, 2, 6, 1, 9, 9, 2, 1];

function sumArray( values ) {
    let sum = 0;
    for ( let i in values ) {
        sum += values[i];
        if ( sum >= 100 ) {
            break;

        }
    }
    return sum;
}

sumArray( numbers );
```

I placed the `break` in braces on purpose to show you that `break` breaks out of `switch` statements and loops, not from if statements. When `break` is executed, control is handed over to the statement after the loop: `return sum`.

In this specific example, we don't even need a `break`, because instead of breaking out of the loop, we can return the sum:

```
let numbers = [19, 65, 9, 17, 4, 1, 2, 6, 1, 9, 9, 2, 1];

function sumArray( values ) {
    let sum = 0;
    for ( let i in values ) {
        sum += values[i];
        if ( sum >= 100 ) return sum;
    }
    return sum;
}

sumArray( numbers );
```

Technically, this is an example, where I would suggest a more flexible for loop:

```
let numbers = [19, 65, 9, 17, 4, 1, 2, 6, 1, 9, 9, 2, 1];

function sumArray( values ) {
    let sum = 0;
    for ( let i = 0; i < values.length && sum <= 100; ++i ) {
        sum += values[i];
    }
    return sum;
}

sumArray( numbers );
```