Loops

Content

For Loop
While Loop
For Each
For ... in

What are loops?

In computer science, a loop is a sequence of instructions that is continually repeated until a certain condition is reached.

For Loop

```
var sum = 0;
for (var i = 1; i <= 50; i++) {
   sum = sum + i;
}
alert("Sum = " + sum);
```

For Loop: Explained

A for loop consists of three parts, separated by semicolons.
The first is the *initializer* (var i = 1) which initializes the loop and is executed only once at the start.
The second is a *test condition* (i <= 50). When a conditional expression evaluates to true, the body of the loop is executed. When false, the loop terminates.
The third part is an *updater* (i++) which is invoked after each iteration. The updater typically increments or decrements the loop counter.

While Loop

```
var sum = 0;
var number = 1;
while (number <= 50) {  // -- condition
 sum += number;         // -- body
 number++;              // -- updater
}
console.log("Sum = " + sum);  // => Sum = 1275
```

While Loop

The condition is first evaluated. If true, the block of statements following the while statement is executed.

This is repeated until the condition becomes false. This is known as a *pre-test loop* because the condition is evaluated before the block is executed.

For Loop Vs. While Loop

In a for-loop, all three parts i.e. initializer, test condition, and updater are written together in a single line (called an iteration statement), whereas in a while, they're scattered and lie at different places. This makes a for-loop more readable than a while-loop and as a result, more easily maintainable.
So when do we use for and when while? If the number of iterations is known, use the for-loop. If you want to loop until a certain condition is met, use the while-loop.

Do-While Loop

```
var sum = 0;
var number = 1;
do {
  sum += number;              // -- body
  number++;                   // -- updater
} while (number <= 50);     // -- condition
console.log("Sum = " + sum);     // => Sum = 1275
```

Do-While Loop

The block following do is executed first and then the condition is evaluated. If the while condition is true, the block is executed again and repeats until the condition becomes false.

This is known as a *post-test loop* as the condition is evaluated after the block has executed.

While Vs. Do-While

The do-while loop is executed at least once whereas the while loop may not execute at all.

The do-while is typically used in a situation where the body of a loop contains a statement that generates a value that you want to use in your conditional expression, like this:

```
do {
   // read a character from keyboard in the body
} while (if ch === '0');      // => terminate loop if '0' is entered
```

forEach()

The forEach() method executes a function once for each element in an **array**.

```
var array1 = ['a', 'b', 'c'];

array1.forEach(function(element) {
  console.log(element);
});

// expected output: "a"
// expected output: "b"
// expected output: "c"
```

MDN Source

for ... in

The for ... in statement loops over properties of an **object**.

```
var string1 = "";
var object1 = {a: 1, b: 2, c: 3};

for (var property1 in object1) {
  string1 += object1[property1];
}

console.log(string1);
// expected output: "123"
```

MDN Source

Why use for...in?

`for...in` is built for looping over object properties. It is not recommended to use with arrays, so what might be the use of `for...in`?

It may be most practically used for **debugging** purposes since it is an easy way to check the properties of an object (by outputting to the console or otherwise). Although arrays are often more practical for storing data, in situations where a key-value pair is preferred for working with data, there may be instances where you want to check if any of those keys hold a particular value.

MDN Source