JavaScript For Web

Week 3, Lecture 2 - JavaScript Fundamentals: Loops

Instructor: Jason Xu

Today's Overview

- What are loops useful for?
- `for` and `while` loop
- `break` and `continue` statment
- **Tutorial**: Assignment 2, Exercise 2

What are Loops

- Programming languages are very useful for rapidly completing repetitive tasks, from multiple basic calculations to just about any other situation where you've got a lot of similar items of work to complete.
- Loops are all about doing the same thing over and over again. Often, the code will be slightly different each time round the loop, or the same code will run but with different variables.
- Most of the time when you use a loop, you will have a collection of items and want to do something with every item.
- One type of collection is the `Array`, but there are other collections in JavaScript as well, including `Set` and `Map`.

`for` loop

Here is the syntax of **the standard for loop**:

```
const fruits = ["Apple", "Orange", "Banana", "Mango", "Strawberry", "Grapes", "Watermelon"]

for (let i = 0; i < fruits.length; i++) {
  console.log(fruits[i])
}
```

Here we have:

1. The keyword `for`, followed by some parentheses.
2. Inside the parentheses we have three items, separated by semicolons:

- An **initializer** — this is usually a variable set to a number, which is incremented to count the number of times the loop has run.
- A **condition** — this condition defines when the loop should stop looping.
- A **final-expression** — this is always evaluated (or run) each time the loop has gone through a full iteration.

3. Some curly braces that contain a block of code — this code will be run each time the loop iterates.

`for...of` loop

```
const fruits = ["Apple", "Orange", "Banana", "Mango", "Strawberry", "Grapes", "Watermelon"]

for (let fruit of fruits) {
  console.log(fruit)
}
```

In this example, `for (let fruit of fruits)` means:

1. Given the array `fruits`, get the first item in the array.
2. Assign it to the variable `fruit` and then run the code between the curly braces {}.
3. Get the next item, and repeat (2) until you've reached the end of the array.

`while` loop

`for` is not the only type of loop available in JavaScript. It is worth to look at the structure of other loop so that you can recognize the same features at work in a different way.

Have a look at the while loop. This loop's syntax looks like this:

```
const fruits = ["Apple", "Orange", "Banana", "Mango", "Strawberry", "Grapes", "Watermelon"]

let i = 0
while (i < fruits.length) {
  // code to run
  console.log(fruits[i])
  i++
}
```

WARNING: An **infinite loop** is a piece of code that keeps running forever as the terminating condition is never reached.

- An infinite loop can crash your program or browser and freeze your computer.
- Ensure you have at least one statement within the loop that changes the value of the comparison variable.

`break` statement

You can use the break statement if you want to **exit a loop** before all the iterations have been completed.

For example, we want find the index of "Banana" and output to the console:

```
const fruits = ["Apple", "Orange", "Banana", "Mango", "Strawberry", "Grapes", "Watermelon"]

let message

for (let i = 0; i < fruits.length; i++) {
  if(fruits[i] === "Banana") {
    message = `The position of Banana is ${i + 1} in the list.`
    break
  }
}

if(!message) {
  console.log("Banana does not found.")
} else {
  console.log(message)
}
```

`continue` statement

The `continue` statement works similarly to `break`, but instead of breaking out of the loop entirely, it skips to the next iteration of the loop.

For example, if we want to create an new array without having "Banana":

```
const fruits = ["Apple", "Orange", "Banana", "Mango", "Strawberry", "Grapes", "Watermelon"]
const new_fruits = []

for (let i = 0; i < fruits.length; i++) {
  if(fruits[i] === "Banana") {
    continue
  }

  new_fruits.push(fruits[i])
}
```

Summary

Loops:

- The standard `for` loop
- `for...of` loop
- `while` loop
- `break` statement
- `continue` statement

Thank you