PALAWAN TECHNOLOGICAL COLLEGE, INC.
245 Malvar Street, Puerto Princesa City, Palawan

Computer Programming 2
JavaScript Loops

JavaScript Loops are powerful tools for performing repetitive tasks efficiently. **Loops in JavaScript execute a block of code again and again while the condition is true.**

For example, suppose we want to print "Hello World" 5 times. This can be done using JS Loop easily.

Instead of writing:

```
text += cars[0] + "<br>";
text += cars[1] + "<br>";
text += cars[2] + "<br>";
text += cars[3] + "<br>";
text += cars[4] + "<br>";
text += cars[5] + "<br>";
```

You can write:

```
for (let i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}
```

Different Kinds of Loops
JavaScript supports different kinds of loops:

- **for** - loops through a block of code a number of times
- **for/in** - loops through the properties of an object
- **for/of** - loops through the values of an iterable object
- **while** - loops through a block of code while a specified condition is true
- **do/while** - also loops through a block of code while a specified condition is true

A. THE FOR LOOP
The for statement creates a loop with 3 optional expressions:

```
for (expression 1; expression 2; expression 3) {
  // code block to be executed
}
```

1. Expression 1 is executed (one time) before the execution of the code block.
2. Expression 2 defines the condition for executing the code block.
3. Expression 3 is executed (every time) after the code block has been executed.

Example:

```
for (let i = 0; i < 5; i++) {
  text += "The number is " + i + "<br>";
}
```

From the example above, you can read:

1. Expression 1 sets a variable before the loop starts (let i = 0).
2. Expression 2 defines the condition for the loop to run (i must be less than 5).
3. Expression 3 increases a value (i++) each time the code block in the loop has been executed.

Expression 1
Normally you will use expression 1 to initialize the variable used in the loop (let i = 0).

This is not always the case. JavaScript doesn't care. Expression 1 is optional.

You can initiate many values in expression 1 (separated by comma):

```
for (let i = 0, len = cars.length, text = ""; i < len; i++) {
  text += cars[i] + "<br>";
}
```

And you can omit expression 1 (like when your values are set before the loop starts):

```
let i = 2;
let len = cars.length;
let text = "";
for (; i < len; i++) {
  text += cars[i] + "<br>";
}
```

Expression 2
Often expression 2 is used to evaluate the condition of the initial variable.

This is not always the case. JavaScript doesn't care. Expression 2 is also optional.

If expression 2 returns true, the loop will start over again. If it returns false, the loop will end.

If you omit expression 2, you must provide a break inside the loop. Otherwise, the loop will never end. This will crash your browser. Read about breaks in a later chapter of this tutorial.

Expression 3
Often expression 3 increments the value of the initial variable.

This is not always the case. JavaScript doesn't care. Expression 3 is optional.

Expression 3 can do anything like negative increment (i--), positive increment (i = i + 15), or anything else.

Expression 3 can also be omitted (like when you increment your values inside the loop):

```
let i = 0;
let len = cars.length;
let text = "";
for (; i < len; ) {
  text += cars[i] + "<br>";
  i++;
}
```

Loop Scope
Using var in a loop:

```
var i = 5;

for (var i = 0; i < 10; i++) {
  // some code
}

// Here i is 10
```

For this example, using var, the variable declared in the loop redeclares the variable outside the loop.

Using let in a loop:

```
let i = 5;

for (let i = 0; i < 10; i++) {
  // some code
}

// Here i is 5
```

In the second example, using let, the variable declared in the loop does not redeclare the variable outside the loop.

When let is used to declare the i variable in a loop, the i variable will only be visible within the loop.

B. THE FOR IN LOOP

The JavaScript for in statement loops through the properties of an Object:

```
for (key in object) {
  // code block to be executed
}
```

Example:

```
const person = {fname:"John", lname:"Doe", age:25};

let text = "";
for (let x in person) {
  text += person[x];
}
```

Example Explained

- The for in loop iterates over a person object
- Each iteration returns a key (x)
- The key is used to access the value of the key
- The value of the key is person[x]

For In Over Arrays
The JavaScript for in statement can also loop over the properties of an Array:

Syntax:

```
for (variable in array) {
  code
}
```

Example:

```
const numbers = [45, 4, 9, 16, 25];

let txt = "";
for (let x in numbers) {
  txt += numbers[x];
}
```

Do not use for in over an Array if the index order is important.

The index order is implementation-dependent, and array values may not be accessed in the order you expect.

It is better to use a for loop, a for of loop, or Array.forEach() when the order is important.

Array.forEach()
The forEach() method calls a function (a callback function) once for each array element.

Example:

```
const numbers = [45, 4, 9, 16, 25];

let txt = "";
numbers.forEach(myFunction);

function myFunction(value, index, array) {
  txt += value;
}
```

Note that the function takes 3 arguments:

- The item value
- The item index
- The array itself

The example above uses only the value parameter. It can be rewritten to:

```
const numbers = [45, 4, 9, 16, 25];

let txt = "";
numbers.forEach(myFunction);

function myFunction(value) {
  txt += value;
}
```

C. THE FOR OF LOOP

The JavaScript for of statement loops through the values of an iterable object.

It lets you loop over iterable data structures such as Arrays, Strings, Maps, NodeLists, and more:

Syntax:

```
for (variable of iterable) {
  // code block to be executed
}
```

- variable - For every iteration the value of the next property is assigned to the variable. Variable can be declared with const, let, or var.

- iterable - An object that has iterable properties.

Browser Support

For/of was added to JavaScript in 2015 (ES6). Safari 7 was the first browser to support for of:

Chrome 38	Edge 12	Firefox 51	Safari 7	Opera 25
Oct 2014	Jul 2015	Oct 2016	Oct 2013	Oct 2014

For/of is not supported in Internet Explorer.

Looping over an Array
Example:

```
const cars = ["BMW", "Volvo", "Mini"];

let text = "";
for (let x of cars) {
  text += x;
}
```

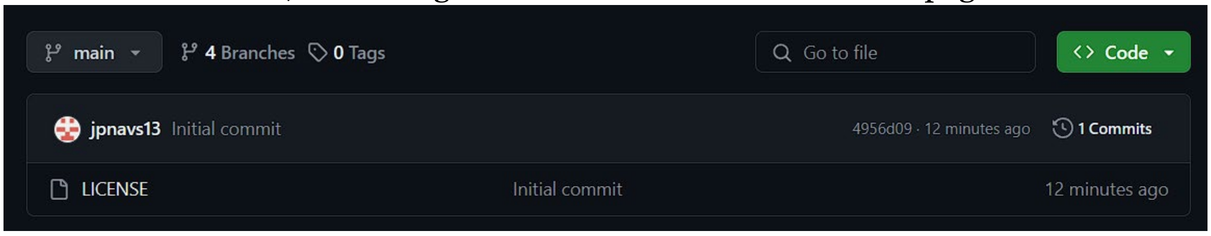
Looping over a String
Example:

```
let language = "JavaScript";

let text = "";
for (let x of language) {
text += x;
}
```

Access the files here:
https://github.com/jpnavs13/js-lesson4-loops

Go to the link above, then navigate to "branches" in the branches page.

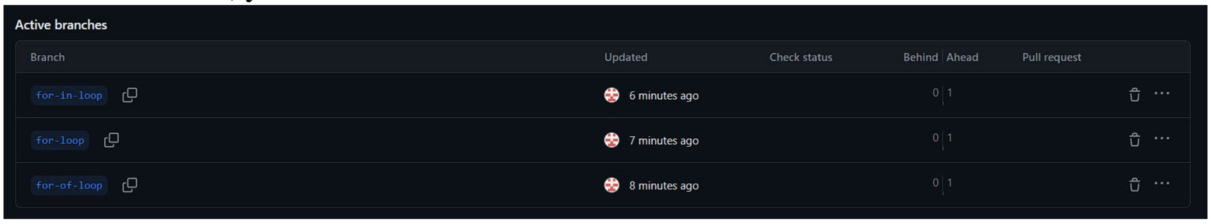


main · 4 Branches · 0 Tags · Go to file · Code

jpnavs13 Initial commit 4956d09 · 12 minutes ago 1 Commits

LICENSE Initial commit 12 minutes ago

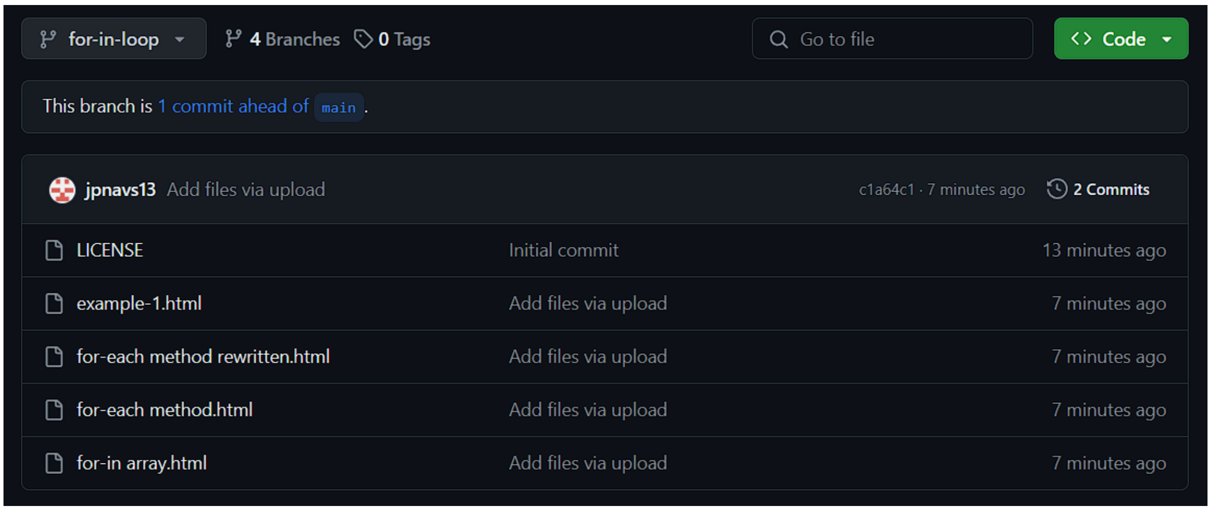
Under branches, you can see active branches



Active branches

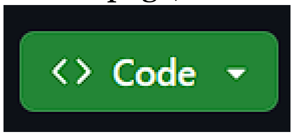
Branch	Updated	Check status	Behind	Ahead	Pull request
for-in-loop	6 minutes ago		0	1	
for-loop	7 minutes ago		0	1	
for-of-loop	8 minutes ago		0	1	

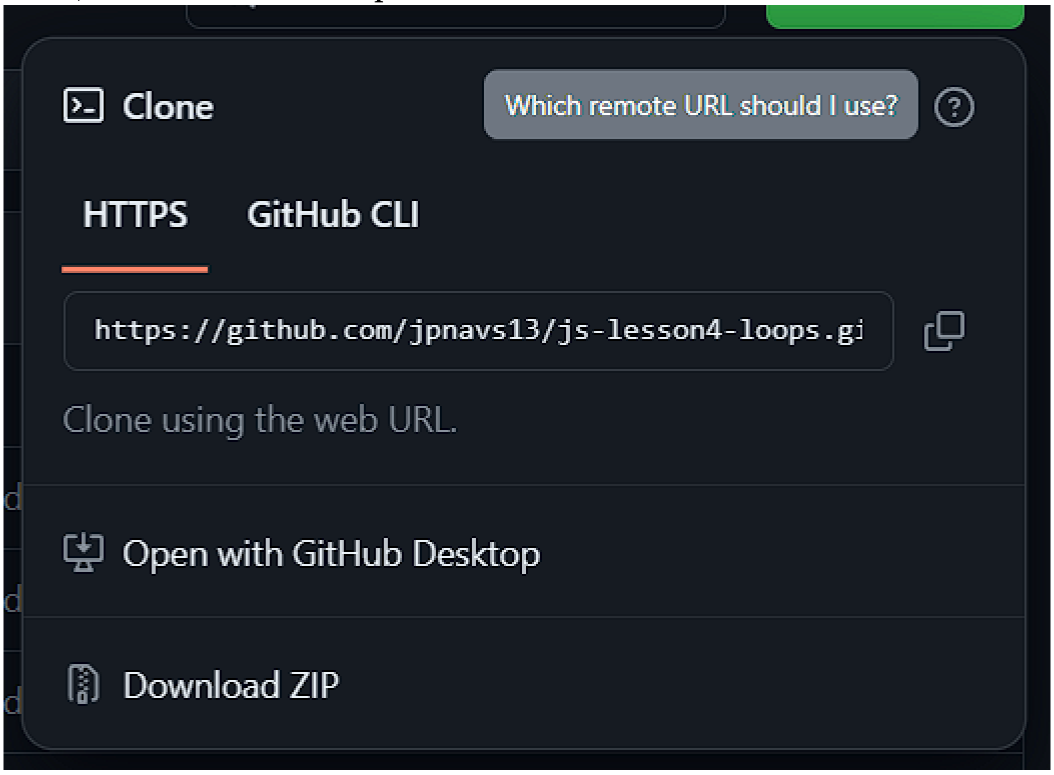
Select a branch. For example, I selected **for in loop** branch. This will redirect me to its dedicated page.


for-in-loop · 4 Branches · 0 Tags · Go to file · Code

This branch is 1 commit ahead of `main`.

jpnavs13 Add files via upload	c1a64c1 · 7 minutes ago	2 Commits
LICENSE	Initial commit	13 minutes ago
example-1.html	Add files via upload	7 minutes ago
for-each method rewritten.html	Add files via upload	7 minutes ago
for-each method.html	Add files via upload	7 minutes ago
for-in array.html	Add files via upload	7 minutes ago

On this page, click on


Then, select download a zip.

Clone

Which remote URL should I use?

HTTPS · GitHub CLI

```
https://github.com/jpnavs13/js-lesson4-loops.gi
```

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

Once downloaded, you can run the samples using your preferred text editor (VS Code/Sublime Text 3).