JavaScript: Looping

The Complete Web Developer in 2018

The Complete Web Developer in 2018
Zero to Mastery
Andrei Neagoie
Lecture Notes by Stephanie

Javascript Looping

for
while
do
forEach (new in ECMAScript 5)

textbook:
https://eloquentjavascript.net/02_program_structure.html

```
> var toDoList = [
      "clean room",
      "brush teeth",
      "exercise",
      "study js",
      "eat healthy"
  ];
```

```
  toDoList.length
<- 5
```

For Loop (for loop is a type of while loop including counter)

```
  for (var i=0; i < toDoList.length; i++) {
      toDoList[i] = toDoList[i] + "!";
  }
```

```
> toDoList
<- ▶ (5) ["clean room!", "brush teeth!", "exercise!", "study
    js!", "eat healthy!"]
```

For Loop example 2

```
> var toDoList = [
      "clean room",
      "brush teeth",
      "exercise",
      "study js",
      "eat healthy"
  ];

  for (var i=0; i < toDoList.length; i++) {
      toDoList.pop();
  }
```

```
> toDoList
<- ▶ (2) ["clean room", "brush teeth"]
```

For Loop example 3

```
> var toDoList = [
      "clean room",
      "brush teeth",
      "exercise",
      "study js",
      "eat healthy"
  ];

  var toDoListLength = toDoList.length;
  for (var i=0; i < toDoListLength; i++) {
      toDoList.pop();
  }
```

```
> toDoList
<- ▶ []
```

While Loop

```
> var counterOne = 0;
  while (counterOne < 10) {
      console.log(counterOne);
      counterOne++
  }
```

```
0                                   pathturbo.js:1
1                                   pathturbo.js:1
2                                   pathturbo.js:1
3                                   pathturbo.js:1
4                                   pathturbo.js:1
5                                   pathturbo.js:1
6                                   pathturbo.js:1
7                                   pathturbo.js:1
8                                   pathturbo.js:1
9                                   pathturbo.js:1
```

While example 2

```
> var counterOne = 10;
  while (counterOne > 0) {
      console.log(counterOne);
      counterOne--;
  }
```

```
10                                  pathturbo.js:1
9                                   pathturbo.js:1
8                                   pathturbo.js:1
7                                   pathturbo.js:1
6                                   pathturbo.js:1
5                                   pathturbo.js:1
4                                   pathturbo.js:1
3                                   pathturbo.js:1
2                                   pathturbo.js:1
1                                   pathturbo.js:1
```

Do-While Loop

```
> var counterTwo = 10;
  do {
      console.log(counterTwo);
      counterTwo--;
  } while (counterTwo > 0);
```

```
10                                                    pathturbo.js:1
9                                                     pathturbo.js:1
8                                                     pathturbo.js:1
7                                                     pathturbo.js:1
6                                                     pathturbo.js:1
5                                                     pathturbo.js:1
4                                                     pathturbo.js:1
3                                                     pathturbo.js:1
2                                                     pathturbo.js:1
1                                                     pathturbo.js:1
```

While vs Do-While Loop

While loop checks condition, then does stuff
Do-while does stuff, then checks condition

```
> var counterOne = 10;
  while (counterOne > 10) {
      console.log(counterOne);
      counterOne--;
  }
```

```
<· undefined
```

```
> var counterTwo = 10;
  do {
      console.log(counterTwo);
      counterTwo--;
  } while (counterTwo > 10);
```

```
10
```

Most of the time >> use For Loop

```
> var toDoList = [
      "clean room",
      "brush teeth",
      "exercise",
      "study js",
      "eat healthy"
  ];
```

forEach method

A simpler way to do for loop. Can also be broken up and use same function with multiple arrays

Instead of doing this:

```
> var toDoListLength = toDoList.length;
  for (var i=0; i < toDoListLength; i++) {
      console.log(toDoList[i], i);
  }
```

Use forEach:

```
> toDoList.forEach(function(toDo, i) {
      console.log(toDo, i);
  })
```

Result:

```
clean room 0
brush teeth 1
exercise 2
study js 3
eat healthy 4
```

Now lets do the same thing, but break up the forEach into a reusable function + forEach…

```
function logToDoList(toDo, i) {
    console.log(toDo, i);
}

toDoList.forEach(logToDoList);
```

```
clean room 0                                    pathturbo.js:1
brush teeth 1                                   pathturbo.js:1
exercise 2                                      pathturbo.js:1
study js 3                                      pathturbo.js:1
eat healthy 4                                   pathturbo.js:1
```

Let's reuse same function for a different array…

Here's the array:

Use forEach method with previous function…

```
> var toDoListOfFool = [
    "messy room!!!",
    "dirty teeth!",
    "sit around!!!!",
    "watch netflix!!",
    "eat junk!!!"
];
```

```
> toDoListOfFool.forEach(logToDoList);
```

```
messy room!!! 0                                 pathturbo.js:1
dirty teeth! 1                                  pathturbo.js:1
sit around!!!! 2                                pathturbo.js:1
watch netflix!! 3                               pathturbo.js:1
eat junk!!! 4                                   pathturbo.js:1
```

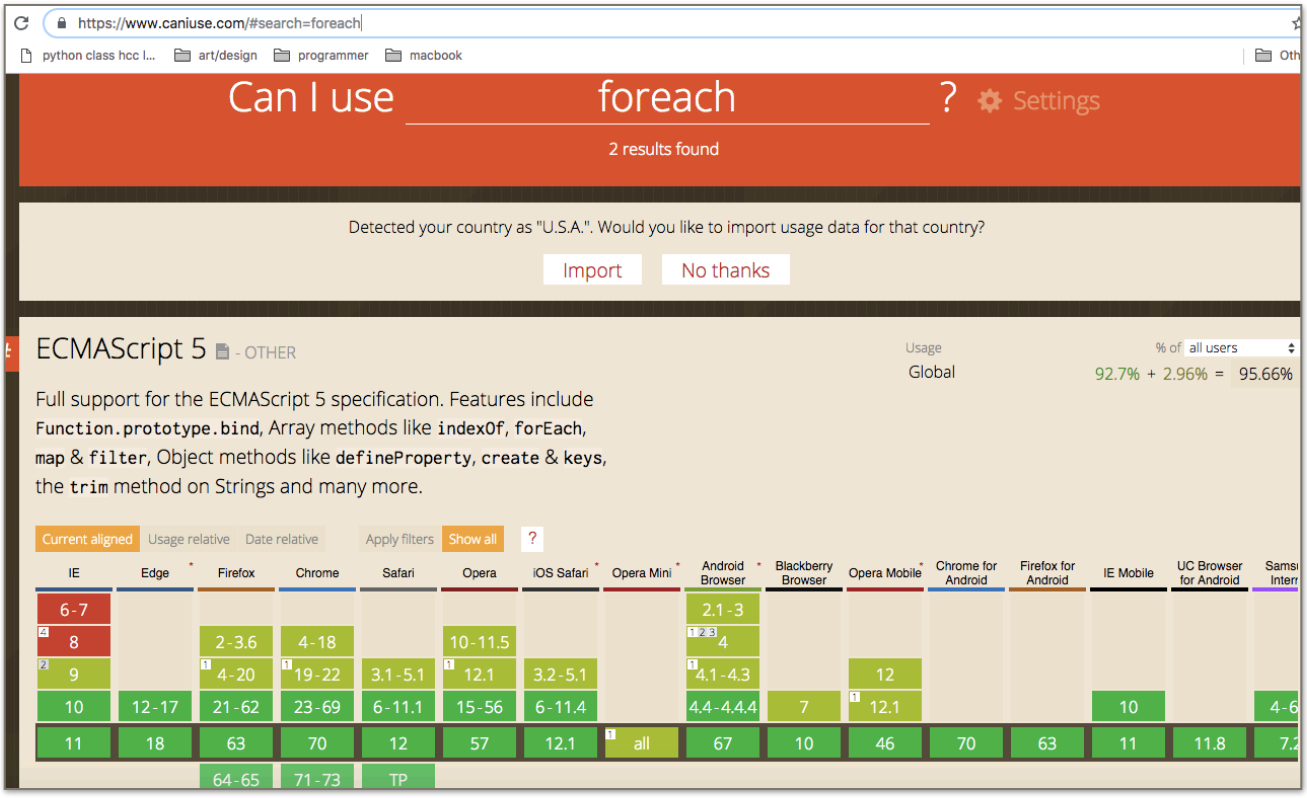
We only have to write the function once and can use over and over (extensible). Had we used a For Loop, we would have to re-write the function for each array.

Final version of forEach function that stores the result:

```
function logToDoList(array) {              // takes array as input
      logArray=[];
      array.forEach(function(toDo, i){
            console.log(toDo, i);
            logArray.push(toDo + " " + i);  // adds each to logArray
      })
      return logArray;
}

var resultArray = logToDoList(toDoListOfFool);   // store return array
```

```
> resultArray
<- ▼(5) ["messy room!!! 0", "dirty teeth! 1", "sit around!!!!
    2", "watch netflix!! 3", "eat junk!!! 4"]
     0: "messy room!!! 0"
     1: "dirty teeth! 1"
     2: "sit around!!!! 2"
     3: "watch netflix!! 3"
     4: "eat junk!!! 4"
     length: 5
   ▶ __proto__: Array(0)
```

Use www.caniUse.com to see whether forEach is supported across all browsers:



Supported everywhere except partial support in Opera Mini >> good to go!

Exercise: Build Facebook 2
Incorporate loops to check username/password against multiple username/passwords

```
> // OUR FACEBOOK BUILD

  var database = [          // array (list)
      {                                 // element 0
          username: "andrei",
          password: "supersecret"
      },
      {                                 // element 0
          username: "sally",
          password: "123"
      },
      {                                 // element 0
          username: "ingrid",
          password: "777"
      }
  ];

  var newsFeed = [         // array (list)
      {                                 // element 0
          username: "Bobby",
          timeline: "So tired from school!"
      },
      {                                 // element 1
          username: "Sally",
          timeline: "Javascript is bad ass"
      }
  ];

  var userNamePrompt = prompt("Enter username");
  var passwordPrompt = prompt("Enter password");
```

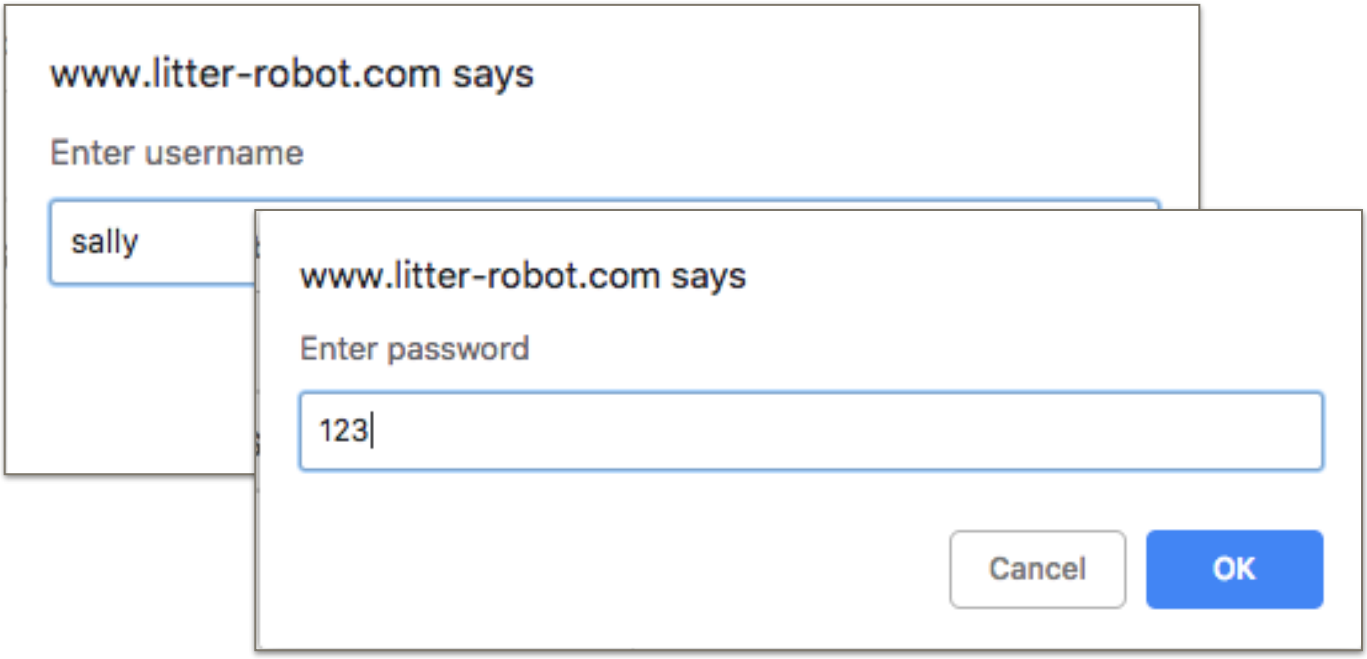
]

```
function isUserValid(user, pass) {
    for (var i=0; i < database.length; i++) {
        if(database[i].username === user &&
            database[i].password === pass) {
            return true; // returns true if un/pw in db
        }
    }
    return false; // returns false if un/pw dont match db
}

function signIn(user, pass) {   // funxn declaration
    // console.log(isUserValid(user,pass)); // logs true
if un/pw correct

    if (isUserValid(user,pass)) {
        console.log(newsFeed); // isUserValid = true
    } else {
        alert("Sorry, wrong UN or PW") // isUserValid =
false
    }

}

signIn(userNamePrompt, passwordPrompt);
```

When we call signIn( ) function in last line of code, the UN and PW entries are passed to the signIn( ) function. The if statement then passes them to the isUserValid( ) function. The isUserValid( ) uses a for loop to compare the parameters to the database, returning true only if they match.

If isUserValid is true, the newsFeed is logged
If isUserValid is false, we get a wrong password alert.

If username and password match a set in database:



Timeline is logged…

If no match to database…