JavaScript for App Development

By Mark Lassoff, Founder Framework Television

ES6 Sets

After completing this section you will:

- ☐ Creating a set
- ☐ How use the set constructor to create a set
- ☐ How to chain add() methods to add set members
- ☐ Use the has() method to test if a value is a members of a set
- ☐ Delete members of a set
- ☐ Use a `for...of` loop to iterate through set members.

Introduction

Sets are a convenient data structure for storing list-like data. In this section of the JavaScript for Application development program, you'll learn how to define and manipulate sets. Sets are one of the new data structures that are part of the ES6 standard.

Sets are somewhat similar to Arrays in Javascript, except in sets each member must be unique. You cannot have duplicate members of a set object.

Creating New Sets

Creating Sets with `New`

As with most JavaScript objects, there are several ways to create a set. Let's take a look at how to create a set with the `new` operator:

```
let songs = new Set();
songs.add("Don't Stop Believing");
songs.add("Eye of the Tiger");
songs.add("Every Breath You Take");
```

In this example `songs` is first created as an empty set. Next, each of the song titles is added to the set as a String.

Creating Sets with the set() Constructor

Additionally, sets can be created by providing the initial set values to the set constructor.

```
let gpas = new Set([4.0, 3.82, 2.225, 3.12]);
```

Note that the values here are surrounded by both parenthesis and the hard brackets. This code creates a Set object called `gpas` that contains the values listed.

> It doesn't really matter which method you use to create your set. Once you build them, they are functionally equivalent regardless of the technique used.

Chaining Add() Operators

Note that in the first example of a Set() object we used the `add()` method to add members to the set. The add() methods can be chained to create a single statement.

This can be a great convenience for developers.

```
let states = new Set()
.add("Connecticut")
.add("Texas")
.add("New York")
.add("Illinois")
.add("Florida");
```

Our `states` set created above contains the name of the five states listed as Strings. The carriage returns are added for readability and have no computational purpose.

The has() method

The `has()` method returns TRUE if an element tested is present in a set. If the element is not present, it will return FALSE.

```
if(states.has("Connecticut")) {
        alert("Present!");
  }
```

Using the `states` set created above the `if` statement would evaluate as true because the string `"Connecticut"` is part of the set. Since the statement is true the alert() function will fun.

The delete() method

Of course, developers need a way to delete members of a set when required. The creatively named `delete()` method serves this function.

```
gpas.delete(4.0);
```

The code deletes the value 4.0 from the gpas set. Since each value must be unique in a set, removing by value is convenient.

For... Of Loops

A `for...of` loop can be used to iterate through each member of the set. As the loop progresses, each member is assigned to a variable so you can address it. The format of a `for...of` loop is:

```
for(let state of states) {
        alert(state);
  }
```

This `for...of loop` loops through the set `states`. As it does, each member is assigned to the variable `state`. In this case we simply `alert()` out the value, however, we could do much more as we iterate through the loop.

Debug This:

The following code, when functional, should list all the individual grades in the first div and then print out “Average: XXXXX” in the other where XXXXX is the average of all the grades in the set.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Average</title>
  </head>
  <body>
    <div id="scores"></div>
    <div id="average"></div>
    <script>
      let sum =0;
      let grades = new set(85,67,23,91,63,88,78,71,75,82,92,80);

      for (let grades in grade)
      {
        sum += grade;
        document.getElementById('scores').innerHTML +=
grade + " ";
      }

      let average = sum/grades.size;
      document.getElementById('average').innerHTML =
"Average: " + average;


    </script>
  </body>
</html>
```

Submit This: BINGO was his Name-o

You are, perhaps, familiar with the game Bingo, where each participant is given a card of numbers broken into columns labeled B, I, N, G, and O.

You are going to create a BINGO caller program which automatically calls a game of BINGO. All

the bingo numbers will be stored in a set called `numbers`. For your convenience, you can copy the code to create the set here: https://gist.github.com/mlassoff/85c64c18a901d2c7fa2593c4dc03eee0

Your program should have two buttons. The first button displays a random member of the set to call. That number should not be called again until the game is reset. Each time the first button is pressed a new BINGO number is displayed. The second button, used when someone calls BINGO, displays all the numbers that have been called so far for verification and resets the game allowing you to play again.

There are many correct solutions here. However, correct submissions will manipulate the initial set which is provided for you. (You may want to create an additional set for the numbers that have been called.) . Good luck.

Please save your file in the following format to ensure proper credit:

`LastName_Sets.html.`