# 6.8 Arrays

## Array introduction

An ***array*** is an ordered collection of values called ***elements***. Each array element is stored in a numeric location called an ***index***. An array is initialized by assigning an array variable with brackets `[ ]` containing comma-separated values.

Array elements may be of the same type or different types. Arrays increase in size as elements are added and decrease as elements are removed.

**PARTICIPATION ACTIVITY** 6.8.1: Initializing and displaying array elements.

1 2 3 4 5 2x speed

```
let scores = [];

scores[0] = 6;
scores[1] = 15;
scores[2] = 8;

console.log(scores[0]);
console.log(scores[1]);
console.log(scores[2]);

let teams = ["Tigers", "Bisons",
             "Eagles", "Cobras"];

console.log(teams);
```

| | scores |
|---|---|
| 0 | 6 |
| 1 | 15 |
| 2 | 8 |

| | teams |
|---|---|
| 0 | Tigers |
| 1 | Bisons |
| 2 | Eagles |
| 3 | Cobras |

```
6
15
8
Tigers,Bisons,Eagles,Cobras
```

All four elements are output to the console separated by commas.

Captions

1. An empty array called "scores" is declared with [ ].
2. Three elements are added to the scores array at indexes 0, 1, and 2.
3. The three elements are output to the console.
4. The teams array is initialized with four elements.
5. All four elements are output to the console separated by commas.

Feedback?

**PARTICIPATION ACTIVITY** 6.8.2: Initializing and displaying array elements.

1) Initialize `names` to an empty array.

**Correct**

```
let names = [];
```

Check Show answer

[]

Values may be placed inside the square brackets when initializing an array.

2) How many elements does `names` have?

```
let names = [];
names[0] = "Sue";
names[1] = "Bob";
names[2] = "Jeff";
```

3

Check Show answer

**Correct**

3

`names` is initially empty, but three strings are assigned to indexes 0, 1, and 2. Assigning values to previously undefined elements increases the array size.

3) What is output to the console?

```
let names = [];
names[0] = "Sue";
names[1] = "Bob";
names[2] = "Jeff";
console.log(names[0] +
names[1]);
```

SueBob

Check Show answer

**Correct**

SueBob

names[0] and names[1] are concatenated, so "Sue" + "Bob" = "SueBob".

4) What is the largest number `nameIndex` can be to output a name?

```
let names = ["Maria",
"Braden", "Jaden", "Aditya"];
console.log(names[nameIndex]);
```

3

Check Show answer

**Correct**

3

The array holds four elements, so the last element is index 3. Using an index > 3 results in "undefined" being output because array elements that are not assigned a value are undefined.

5) What is output to the console?

```
let nums = [5, 2, 9, 3];
nums[0] = nums[1] + nums[2];
console.log(nums[0]);
```

11

Check Show answer

**Correct**

11

nums[0] = nums[1] + nums[2] = 2 + 9 = 11.

6) Write the simplest code to output all elements in `pets`.

```
let pets = ["cats", "dogs", "fish"];
console.log( pets );
```

Check Show answer

**Correct**

pets

Specifying the array's name with no brackets outputs all elements separated by commas.

Feedback?

## Adding and removing array elements

An array is an ***Array object*** that defines numerous methods for manipulating arrays. A ***method*** is a function that is attached to an object and operates on data stored in the object. Methods are called by prefacing the method with the object. Ex: `myArray.method();`.

Table 6.8.1: Array methods for adding and removing array elements.

| Method | Description | Example |
|---|---|---|
| ***push(value)*** | Adds a value to the end of the array | `let nums = [2, 4, 6];`<br>`nums.push(8);  // nums = [2, 4, 6, 8]` |
| ***pop()*** | Removes the last array element and returns | `let nums = [2, 4, 6];`<br>`let x = nums.pop();  // returns 6, nums = [2, 4]` |

| Method | Description | Example |
|---|---|---|
| | the element | |
| ***unshift(value)*** | Adds a value to the beginning of the array | `let nums = [2, 4, 6];`<br>`nums.unshift(0);  // nums = [0, 2, 4, 6]` |
| ***shift()*** | Removes the first array element and returns the element | `let nums = [2, 4, 6];`<br>`let x = nums.shift();  // returns 2, nums = [4, 6]` |
| ***splice(startingIndex, numElemToDelete, valuesToAdd)*** | Adds or removes elements from anywhere in the array and returns the deleted elements (if any) | `let nums = [2, 4, 6, 8, 10];`<br><br>`// Deletes all elements from index 3 to the end`<br>`nums.splice(3);   // nums = [2, 4, 6]`<br><br>`// Deletes 2 elements starting at index 0`<br>`nums.splice(0, 2);   // nums = [6]`<br><br>`// Adds 3, 5 starting at index 0`<br>`nums.splice(0, 0, 3, 5);   // nums = [3, 5, 6]`<br><br>`// Adds 7, 9, 11 starting at index 2`<br>`nums.splice(2, 0, 7, 9, 11);   // nums = [3, 5, 7, 9, 11, 6]` |

Feedback?

**PARTICIPATION ACTIVITY** 6.8.3: Adding and removing array elements.

The six individuals in the `line` array are waiting in line. Write the JavaScript code to add or remove elements to/from the array to simulate the following events:

1. The person at the front of the line (index 0) leaves the line (`shift`).

2. The person at the end of the line cuts in front of the person at the front of the line (`pop` and `unshift`).
3. Two new people named "Poe" and "Snoke" cut into line behind the second person in line (`splice`).
4. The fifth person in line leaves the line (`splice`).
5. A new person named "Han" enters the back of the line (`push`).

Finally, display the contents of the `line` array to view the new line occupants. A correct solution will show: Leia, Finn, Poe, Snoke, Maz, Han.

```
1 // People waiting in line (Kylo is in front, Leia at the end)
2 let line = ["Kylo", "Finn", "Rey", "Maz", "Leia"];
3
4 // Show entire line
5 console.log(line);
6
```

Run JavaScript Reset code

**Your console output**

```
Kylo,Finn,Rey,Maz,Leia
```

▶ View solution

Feedback?

## Looping through an array

The array property ***length*** contains the number of elements in the array. The `length` property is helpful for looping through an array using a for loop.

### Figure 6.8.1: Looping through an array with a for loop.

```
let groceries = ["bread", "milk", "peanut butter"];

// Display all elements in groceries array
for (i = 0; i < groceries.length; i++) {
   console.log(i + " - " + groceries[i]);
}
```

```
0 - bread
1 - milk
2 - peanut butter
```

Feedback?

The ***for-of loop*** is a simplified for loop that loops through an entire array. The array name is placed after the `of` keyword in a for-of loop. Each time through the loop, the next array element is assigned to the variable in front of the `of` keyword.

### Figure 6.8.2: Looping through an array with a for-of loop.

```
let groceries = ["bread", "milk", "peanut butter"];

// Display all elements in groceries array
for (let item of groceries) {
   console.log(item);
}
```

```
bread
milk
peanut butter
```

Feedback?

The `Array` method ***forEach()*** also loops through an array. The `forEach()` method takes a function as an argument. The function is called for each array element in order, passing the element and the element index to the function.

### Figure 6.8.3: Looping through an array with the forEach() method.

```
let groceries = ["bread", "milk", "peanut butter"];

// Display all elements in groceries array
groceries.forEach(function(item, index) {
   console.log(index + " - " + item);
});
```

```
0 - bread
1 - milk
2 - peanut butter
```

Feedback?

**PARTICIPATION ACTIVITY** 6.8.4: Looping through an array.

1) What is `autos.length`?

```
let autos =
["Chevrolet",
"Dodge", "Ford",
"Ram"];
```

- 0
- 3
- 4 ✓

**Correct**

`autos` contains 4 elements.

2) What is output to the console?

```
let autos =
["Chevrolet", "Dodge",
"Ford", "Ram"];
for (i = 0; i < 2;
i++) {

console.log(autos[i]);
}
```

- Chevrolet, Dodge ✓
- Chevrolet, Dodge, Ford
- Chevrolet, Dodge, Ford, Ram

**Correct**

The for loop terminates when `i` = 2, so only `autos[0]` and `autos[1]` are output.

3) What is output to the console?

```
let autos =
["Chevrolet", "Dodge",
"Ford", "Ram"];
for (i = 0; i <
autos.length; i++) {
   if (i % 2 == 0) {

console.log(autos[i]);
   }
}
```

- ○ Chevrolet, Dodge, Ford, Ram
- ◉ Chevrolet, Ford
- ○ Dodge, Ram

**Correct**

Only elements with even indexes are output.

4) What is output to the console?

```
let autos = ["Chevrolet",
"Dodge", "Ford", "Ram"];
autos.forEach(function(item,
index) {
   if (index % 3 == 0) {
      console.log(item);
   }
});
```

- ○ Chevrolet, Dodge, Ford, Ram
- ○ Chevrolet, Ford
- ◉ Chevrolet, Ram

**Correct**

Only elements with indexes that are evenly divisible by 3 are output.

5) What is missing in the for-of loop to display all the elements in the `autos` array?

```
let autos =
["Chevrolet",
"Dodge", "Ford",
"Ram"];
for (____) {

console.log(auto);
}
```

- `let auto of autos`
- `let item of autos`
- `let autos of auto`

**Correct**

Each array element in `autos` is assigned to the `auto` variable, which is logged to the console.

6) Which loop is best for looping through an array in reverse order (from last element to first element)?

- for loop
- for-of loop
- forEach() loop

**Correct**

The for-of and forEach() loops always loop through an array starting with the first element and ending at the last element. The for loop provides more control than the other loops over the order in which array elements are accessed. The code below outputs all the autos in reverse order:

```
for (let i = autos.length - 1; i >= 0;
i--) {
   console.log(autos[i]);
}
```

Feedback?

**PARTICIPATION ACTIVITY** 6.8.5: Practice looping.

Duke and North Carolina have a famous basketball rivalry dating back to 1920. The number of points each team has scored in head-to-head competition over five years is provided in the `dukeScores` and `ncScores` arrays. Ex: North Carolina won the first game 76-72 since `dukeScores[0]` is 72 and `ncScores[0]` is 76.

Write a for loop that examines the `dukeScores` and `ncScores` arrays and places "D" in the `winningTeam` array if Duke won or "N" if North Carolina won, for every game. Ex: `winningTeam[0]` should be "N" because North Carolina won 76-72, and `winningTeam[1]` should be "D" because Duke won 74-73.

Then display the contents of the `winningTeam` array using a for-of or forEach() loop.

```
let dukeScores  = [72, 74, 84, 92, 93, 66, 69, 73, 70, 85, 75, 
let ncScores    = [76, 73, 77, 90, 81, 74, 53, 68, 88, 84, 58, 
let winningTeam = [];

// Who won the first game?
if (dukeScores[0] > ncScores[0]) {
   console.log("Duke won " + dukeScores[0] + "-" + ncScores[0] 
}
else {
   console.log("North Carolina won " + ncScores[0] + "-" + duke
}

```

Run JavaScript Reset code

**Your console output**

```
North Carolina won 76-72.
```

▶ View solution

Feedback?

## Passing arrays to functions

An array can be passed to a function as an argument. A function may modify an array argument's elements.

**PARTICIPATION ACTIVITY** 6.8.6: Passing arrays to functions.

1 2 3 4 5 6 2x speed

```
function findAverage(numbers) {
   let sum = 0;
   for (let i = 0; i < numbers.length; i++) {
      sum += numbers[i];
   }
   return sum / numbers.length;
}

function giveBonus(scores, bonus) {
   for (let i = 0; i < scores.length; i++) {
      scores[i] += bonus;
   }
}

let examScores = [79, 85, 60, 93];
console.log("Average is " + findAverage(examScores));

giveBonus(examScores, 5);
console.log("New average is " + findAverage(examScores));
```

examScores

| | |
|---|---|
| 0 | 84 |
| 1 | 90 |
| 2 | 65 |
| 3 | 98 |

```
Average is 79.25
New average is 84.25
```

Since examScores is changed, the new average is 5 points higher than before.

Captions

1. The examScores array is passed to the findAverage() function.
2. findAverage's numbers parameter refers to the same array as examScores.
3. findAverage() sums the scores in the numbers array, divides the sum by 4, and returns the result.
4. examScores is passed to the giveBonus() function. The scores parameter and examScores argument refer to the same array.
5. The for loop adds 5 to each element in scores, which also changes examScores.
6. Since examScores is changed, the new average is 5 points higher than before.

Feedback?

**PARTICIPATION ACTIVITY** 6.8.7: Passing arrays to functions.

Match the value to the corresponding array element after the code below executes.

```
function makeOdd(numbers) {
   for (let i = 0; i < numbers.length; i++) {
      if (numbers[i] % 2 == 0) {
         numbers[i]++;
      }
   }
}

let ages = [10, 3, 5, 8];
makeOdd(ages);
```

If unable to drag and drop, refresh the page.

| | | |
|---|---|---|
| 11 | ages[0]<br>`makeOdd()` adds one to even elements. 10 is even, so 10 changes to 11. | Correct |
| 3 | ages[1]<br>`makeOdd()` does not change odd elements, and 3 is odd. | Correct |
| 5 | ages[2]<br>`makeOdd()` does not change odd elements, and 5 is odd. | Correct |
| 9 | ages[3]<br>`makeOdd()` adds one to even elements. 8 is even, so 8 changes to 9. | Correct |

Reset

Feedback?

## Searching an array

The array methods ***indexOf()*** and ***lastIndexOf()*** search an array and return the index of the first found value or -1 if the value is not found. `indexOf()` searches from the beginning of the array to the end. `lastIndexOf()` searches from the end of the array to the beginning. Both functions take two arguments:

1. `searchValue` - The value to search for
2. `startingPosition` - Optional argument that indicates the index at which the search should begin (default is 0 for `indexOf()` and `array.length - 1` for `lastIndexOf()`)

Figure 6.8.4: Searching for array elements.

```
let scores = [80, 92, 75, 64, 88,
92];

s = scores.indexOf(92);          // 1
s = scores.indexOf(92, 2);       // 5
s = scores.indexOf(100);         //
-1
s = scores.lastIndexOf(92);      // 5
s = scores.lastIndexOf(92, 4);   // 1
s = scores.lastIndexOf(50);      //
-1
```

Feedback?

**PARTICIPATION ACTIVITY** 6.8.8: Searching an array.

Refer to the `artists` array.

```
let artists = ["Raphael", "Titian", "Masaccio", "Botticelli",
"Titian"];
```

1) `artists.indexOf("Botticelli")` returns 3.
   - (•) True
   - ( ) False

**Correct**

"Botticelli" is located at index 3.

2) `artists.lastIndexOf("Titian")` returns 1.
   - ( ) True
   - (•) False

**Correct**

"Titian" is located at index 1 and 4, and lastIndexOf() returns the last index, which is 4.

3) `artists.indexOf("Michaelangelo")` returns NaN.
   - ( ) True
   - (•) False

**Correct**

"Michaelangelo" is not in the array, and indexOf() returns -1 when the search value is not found.

Feedback?

**PARTICIPATION ACTIVITY** 6.8.9: Practice searching an array.

The `validCredentials()` function contains two parallel arrays of usernames and passwords. Modify `validCredentials()` to use the `indexOf()` method to search the `usernames` array for the given `enteredUsername`. If the username is

found, the same location in the `passwords` array should contain the `enteredPassword`. Return `true` if the passwords are equal, `false` otherwise. `validCredentials()` should also return `false` if the given username was not found.

```
5     // Database of usernames and passwords
6     let usernames = ["smith",  "tron",      "ace",      "ladyj"
7     let passwords = ["qwerty", "EndOfLine", "year1942", "ladyj1
8
9     // Search the usernames array for enteredUsername
10
11    // Only return true if the enteredUsername is in username,
12    // same location in passwords is enteredPassword
13    return true;
14 }
15
16
17 console.log("Login for ladyj: " + validCredentials("ladyj", "l
18 console.log("Login for ace: " + validCredentials("ace", "wrong
19 console.log("Login for jake: " + validCredentials("jake", "???
20
```

Run JavaScript Reset code

**Your console output**

```
Login for ladyj: true
Login for ace: true
Login for jake: true
```

▶ View solution

Feedback?

## Sorting an array

The array method ***sort()*** sorts an array in ascending (increasing) order. `sort()`'s default behavior is to sort each element as a string using the string's Unicode values. Sorting by

Unicode values may yield unsatisfactory results for arrays that store numbers. Ex: 10 is sorted before 2 because "10" is < "2" when comparing the Unicode values of "1" to "2".

The `sort()` method can sort elements in other ways by passing a comparison function to `sort()`. The comparison function returns a number that helps `sort()` determine the sorting order of the array's elements:

- Returns a value < 0 if the first argument should appear before the second argument.
- Returns a value > 0 if the first argument should appear after the second argument.
- Returns 0 if the order of the first and second arguments does not matter.

### Figure 6.8.5: Sorting an array of numbers.

```
let numbers = [200, 30, 1000, 4];

// Sort based on Unicode values: [1000, 200, 30, 4]
numbers.sort();

// Sort numbers in ascending order: [4, 30, 200,
1000]
numbers.sort(function(a, b) {
   return a - b;
});
```

Feedback?

**PARTICIPATION ACTIVITY** 6.8.10: Sorting arrays.

1) What is output to the console?

```
let names = ["Sue",
"Bob", "Jeff"];
names.sort();
console.log(names[0]);
```

Bob

Check Show answer

**Correct**

Bob

By default, `sort()` sorts by Unicode values. "Bob" is placed first in the array because "B" has a lower Unicode value than "S" and "J".

2) What is output to the console?

**Answer**

Sue

The comparison function sorts strings in descending order. "Sue" is > "Bob" and > "Jeff", so the comparison function returns -1 when comparing "Sue" and "Bob"

and "Sue" and "Jeff", placing "Sue" before "Bob" and "Jeff".

```
let names = ["Sue",
"Bob", "Jeff"];
names.sort(function(a,
b) {
   if (a > b) {
      return -1;
   }
   else if (a < b) {
      return 1;
   }
   return 0;
});
console.log(names[0]);
```

Sue

Check Show answer

3) What is output to the console?

```
let totals = [90, 4,
21, 34];
totals.sort();
console.log(totals[0]);
```

21

Check Show answer

**Correct**

21

Numbers are converted into strings before comparing, so the numbers are sorted by Unicode values in ascending order. The "2" in 21 is < the "9", "4", and "3" in 90, 4, and 34. `totals[0]` is the value with the lowest Unicode values.

4) What is output to the console?

```
let totals = [99, 4,
250, 38];
totals.sort(function(a,
b) {
   return b - a;
});
console.log(totals[0]);
```

250

Check Show answer

**Correct**

250

When comparing 99 and 4 with arguments (99, 4), the comparison function returns 4 - 99 = -95, so the first argument 99 is placed before 4. When comparing 4 and 250 with arguments (4, 250), the comparison function returns 250 - 4 = 246, which is positive, so 4 is placed after 250. Larger numbers move to the front of the array, and smaller numbers move to the end. Therefore, the comparison function sorts the array in descending order.

Feedback?

**CHALLENGE ACTIVITY** 6.8.1: Arrays.

530096.4000608.qx3zqy7

Jump to level 1

Add each element in origArray with the corresponding value in offsetAmount. Store each value in a new array named finalArray. Do not output to the console.

```
let origArray = [ 40, 50, 60, 70 ]; // Tests will use different d
let offsetAmount = [ 2, 5, 8, 4 ]; // Tests will use different ar
let finalArray = [];

/* Your solution goes here */
for(let i = 0;i<origArray.length;i++){
   finalArray.push(origArray[i]+offsetAmount[i]);
}
```

1 2 3 4

Check | Try again

**Done**. Click any level to practice more. Completion is preserved.

Testing that finalArray has the expected values with origArray = [ 40, 50, 60, 70 ] and offsetAmount = [ 2, 5, 8, 4 ]

Yours `42,55,68,74`

Testing that finalArray has the expected values with origArray = [ 3, 30, 300 ] and offsetAmount = [ -5, -10, 15 ]

Yours `-2,20,315`

Feedback?

Exploring further:

- Array object (MDN)

How was this section?

Provide section feedback