Java Arrays

❮ Previous Next ❯

Java Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with **square brackets**:

```
String[] cars;
```

We have now declared a variable that holds an array of strings. To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

Access the Elements of an Array

You access an array element by referring to the index number.

This statement accesses the value of the first element in cars:

Example

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars[0]);
// Outputs Volvo
```

Run example »

Note: Array indexes start with 0: [0] is the first element. [1] is the second element, etc.

Change an Array Element

To change the value of a specific element, refer to the index number:

Example

```
cars[0] = "Opel";
```

Example

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
System.out.println(cars[0]);
// Now outputs Opel instead of Volvo
```

Run example »

Array Length

To find out how many elements an array have, use the `length` property:

Example

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars.length);
// Outputs 4
```

Run example »

Loop Through an Array

You can loop through the array elements with the `for` loop, and use the `length` property to specify how many times the loop should run.

The following example outputs all elements in the **cars** array:

Example

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (int i = 0; i < cars.length; i++) {
  System.out.println(cars[i]);
}
```

Run example »

Loop Through an Array with For-Each

There is also a "**for-each**" loop, which is used exclusively to loop through elements in arrays:

Syntax

```
for (type variable : arrayname) {
  ...
}
```

The following example outputs all elements in the **cars** array, using a "**for-each**" loop:

Example

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
  System.out.println(i);
}
```

Run example »

If you compare the two loop examples above, you will see that the **for-each** method is easier to write, it don't need the length property, and it is more readable.

Using the Java Keyword new

You can also create an array and assign values to it with the `new` keyword:

```
// The following example (an array literal):
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

// Can also be created with the new keyword:
String[] cars = new String[] {"Volvo", "BMW", "Ford", "Mazda"};
```

Run example »

It is up to you if you want to use the `new` keyword. In general, we use an array literal when we know what values to insert into the array, while the `new` keyword is used when we know the size of the array (the number of elements), but we don't know the values in advance.

The following example will declare an array of 5 integers (without values)

Example

```
int[] myNumbers = new int[5];
```

To initialize the elements, start adding some values to them:

Example

```java
// Declare an array of 5 integers
int[] myNumbers = new int[5];

// Assign values to array elements
myNumbers[0] = 10; // First element
myNumbers[1] = 20; // Second element
myNumbers[2] = 30; // Third element
//..etc

// Output values
System.out.println("First element: " + myNumbers[0]);
System.out.println("Second element: " + myNumbers[1]);
System.out.println("Third element: " + myNumbers[2]);
//..etc
```

Run example »

Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

To create a two-dimensional array, add each array within its own set of **curly braces**:

Example

```java
int[][] myNumbers = { {1, 2, 3, 4}, {4, 5, 6, 7} };
```

myNumbers is now an array with two arrays as its elements.

To access the elements of the **myNumbers** array, specify two indexes: one for the array, and one for the element inside that array. This example accesses the third element (2) in the second array (1) of myNumbers:

Example

```
int[][] myNumbers = { {1, 2, 3, 4}, {4, 5, 6, 7} };
int x = myNumbers[1][2];
System.out.println(x); // Outputs 6
```

Run example »

We can also use a `for loop` inside another `for loop` to get the elements of a two-dimensional array (we still have to point to the two indexes):

Example

```
public class MyClass {
  public static void main(String[] args) {
    int[][] myNumbers = { {1, 2, 3, 4}, {4, 5, 6, 7} };
    for (int i = 0; i < myNumbers.length; ++i) {
      for(int j = 0; j < myNumbers[i].length; ++j) {
        System.out.println(myNumbers[i][j]);
      }
    }
  }
}
```

Run example »

❮ Previous Next ❯