Arrays

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR

@deborahkurata | blogs.msmvps.com/deborahk/

0	"Red"
1	"Espresso"
2	"White"
3	"Navy"

0	1 "Saw" 9.99
1	2 "Wrench" 8.98
2	3 "Steel Hammer" 15.95

Array

A fixed-size list of elements that can be accessed using a positional index number

Module Overview

Declaring and Populating an Array

Using Collection Initializers

Retrieving an Element from an Array

Iterating Through an Array

Using Array Methods

FAQ

Declaring an Array

```
string[] colorOptions;
```

- Collection of what?
- Array element type
- Array variable

"Red"
"Espresso"
"White"
"Navy"

Initializing an Array

```
string[] colorOptions;
colorOptions = new string[4];
```

- Reference type
- new keyword
- Type and size of the array

0	null
1	null
2	null
3	null

Declaring and Initializing an Array

```
string[] colorOptions;
colorOptions = new string[4];
```

```
string[] colorOptions = new string[4];
```

```
var colorOptions = new string[4];
```

Populating an Array

```
colorOptions[0] = "Red";
colorOptions[1] = "Espresso";
colorOptions[2] = "White";
colorOptions[3] = "Navy";
```

0	Red
1	Espresso
2	White
3	Navy

Array Best Practices

Do:

Consider using an array when the required size of a list can be determined at design time

Use a plural variable name for the array

Avoid:

Using an array when the size of the list is not known

Declaring and Populating an Array

```
var colorOptions = new string[4];
```

```
colorOptions[0] = "Red";
colorOptions[1] = "Espresso";
colorOptions[2] = "White";
colorOptions[3] = "Navy";
```

Collection Initializers

```
var colorOptions = new string[4];
```

```
colorOptions[0] = "Red";
colorOptions[1] = "Espresso";
colorOptions[2] = "White";
colorOptions[3] = "Navy";
```

```
string[] colorOptions = new string[4] {"Red", "Espresso", "White", "Navy"};
```

```
string[] colorOptions = {"Red", "Espresso", "White", "Navy"};
```

```
string[] colorOptions = {"Red", "Espresso", "White", GetMyFavoriteColor()};
```

Array Initialization Best Practices

Do:

Use collection initializers

Avoid:

Manually populating an array

Retrieving an Array Element

```
var colorOptions = new string[4];
```

```
colorOptions[0] = "Red";
colorOptions[1] = "Espresso";
colorOptions[2] = "White";
colorOptions[3] = "Navy";
```

```
Console.WriteLine(colorOptions[1]);
```

Retrieving Array Element Best Practices

Do:

Take care when referencing elements by index

Will generate a runtime exception

Avoid:

Retrieving elements by index when you need all elements

Iterate through instead

Iterating Through an Array

```
1 "Saw" 9.99
2 "Wrench" 8.98
3 "Steel Hammer" 15.95
```

```
"Red"
"Espresso"
"White"
"Navy"
```

`foreach`

`for`

Iterating an Array

`foreach`	`for`
Quick and easy	Complex but flexible
Iterate all elements	Iterate all or a subset of elements
Array element is read-only	Array element is editable

Using Array Methods

- Arrays derive from System.Array class

```
var brownIndex = Array.IndexOf(colorOptions, "Espresso");
```

```
colorOptions.SetValue("Blue", 3);
```

Frequently Asked Questions

- When is it appropriate to use an array?
  - When working with a list whose length is defined at design time.
  - When multiple dimensions are needed.
  - To squeeze out a bit more performance with large sets.
- What is the difference between `foreach` and `for` when iterating through an array?
  - `foreach` provides simple syntax for iterating all elements in an array.
  - `for` provides more complex but flexible syntax for iterating all or any subset of elements in an array.
    - Plus the iterated items are updateable.

Module Summary

- Declaring and Populating an Array
- Using Collection Initializers
- Retrieving an Element from an Array
- Iterating Through an Array
- Using Array Methods