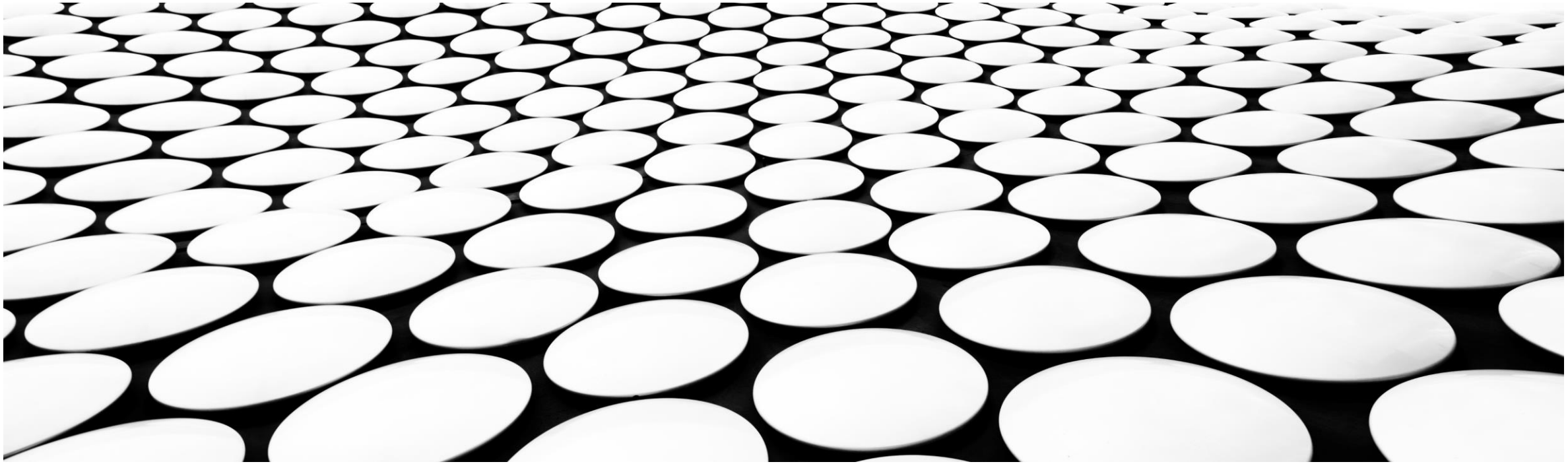
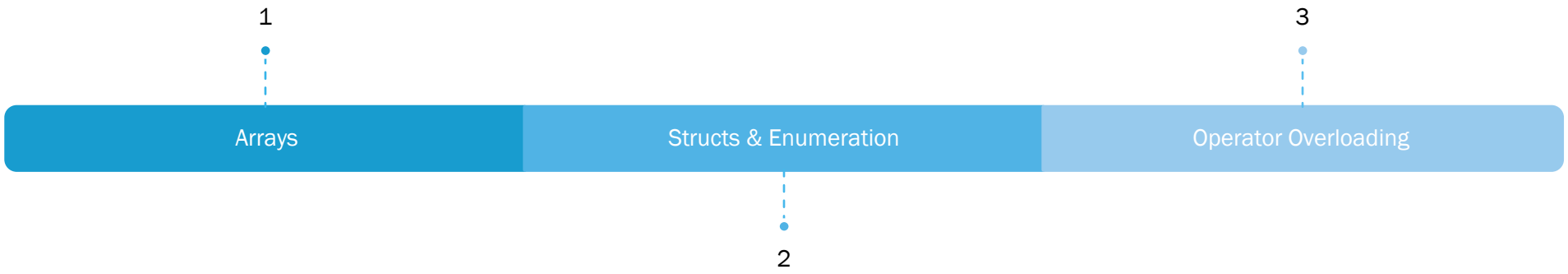
C# - FEATURES

ARCTECH INFO



C# FEATURES



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;
- To declare and initialize an array, use curly braces
  - string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
  - int[] ages = new int[3] { 10, 20, 30 };
  - int[] ages1 = new int[] { 10, 20, 30 };
  - int[] ages2 = { 10, 20, 30 };
  - int[] ages3 = new int[10];
- For late initialization, new is required.

ARRAYS ELEMENTS

- Access array elements
  - Console.WriteLine(cars[0])
- Changes array elements
  - Cars[0] = “Audi”
- Array Length
  - Console.WriteLine(cars.Length);
- Loop Through an Array
  - for and foreach

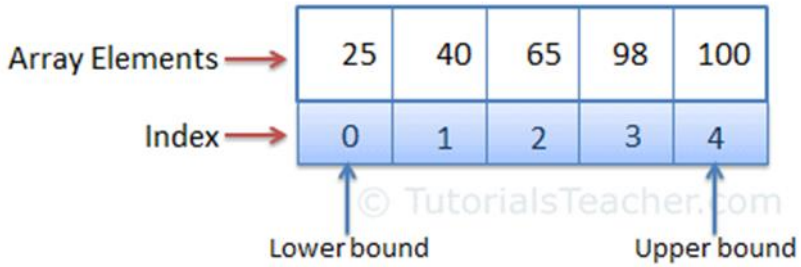
ARRAYS ACTIONS

- System namespace
  - Array.Sort(nums); // sorts array
  - Array.Reverse(nums); // sorts array in descending order
  - Array.ForEach(nums, n => Console.WriteLine(n)); // iterates array
  - Array.BinarySearch(nums, 5);// binary search
- System.Linq namespace
  - Min, Max, and Sum
  - int[] myNumbers = {5, 1, 8, 9};
  - Console.WriteLine(myNumbers.Max()); // returns the largest value

PASSING ARRAYS TO FUNCTIONS

- An array can be passed as an argument to a method parameter. Arrays are reference types, so the method can change the value of the array elements.
- {
  int[] nums = {10, 20, 30};
  UpdateArray(nums);
  }
  public static void UpdateArray(int[] arr)
  {
  …
  }

TYPES OF ARRAYS

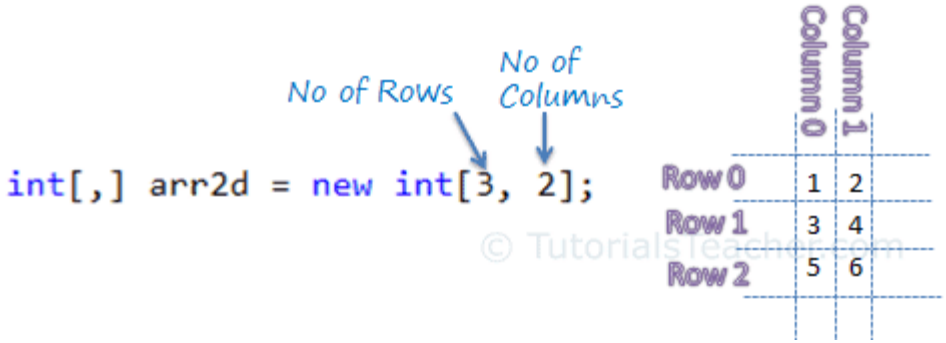


- Single Dimensional
- C# supports multidimensional arrays up to 32 dimensions.
  - The multidimensional array can be declared by adding commas in the square brackets.
  - E.g. [,] declares two-dimensional array,
  - [, ,] declares three-dimensional array, [, , ,] declares four-dimensional array, and so on.
  - So, in a multidimensional array, no of commas = No of Dimensions - 1.
  - `int[,] arr2d; // two-dimensional array`
  - `int[, ,] arr3d; // three-dimensional array`
  - `int[, , ,] arr4d ; // four-dimensional array`
  - `int[, , , ,] arr5d; // five-dimensional array`
- Jagged Arrays

TWO DIMENSIONAL ARRAYS

```
int[,] arr2d = new int[3,2]{
                                {1, 2},
                                {3, 4},
                                {5, 6}
                            };
```

```
// or
int[,] arr2d = {
                        {1, 2},
                        {3, 4},
                        {5, 6}
                    };
```




```
arr2d[0, 0]; //returns 1
arr2d[0, 1]; //returns 2
arr2d[1, 0]; //returns 3
arr2d[1, 1]; //returns 4
arr2d[2, 0]; //returns 5
arr2d[2, 1]; //returns 6
```

3D ARRAYS

```
int[, ,] arr3d1 = new int[1, 2, 2]{
                { { 1, 2}, { 3, 4} }
            };

int[, ,] arr3d2 = new int[2, 2, 2]{
                { {1, 2}, {3, 4} },
                { {5, 6}, {7, 8} }
            };

int[, ,] arr3d3 = new int[2, 2, 3]{
                { { 1, 2, 3}, {4, 5, 6} },
                { { 7, 8, 9}, {10, 11, 12} }
            };
```

```
arr3d2[0, 0, 0]; // returns 1
arr3d2[0, 0, 1]; // returns 2
arr3d2[0, 1, 0]; // returns 3
arr3d2[0, 1, 1]; // returns 4
arr3d2[1, 0, 0]; // returns 5
arr3d2[1, 0, 1]; // returns 6
arr3d2[1, 1, 0]; // returns 7
arr3d2[1, 1, 1]; // returns 8
```

JAGGED ARRAYS

- A jagged array is an array of array.
- Jagged arrays store arrays instead of literal values.
- A jagged array is initialized with two square brackets [][].
- The first bracket specifies the size of an array, and the second bracket specifies the dimensions of the array which is going to be stored.

- `int[][] jArray1 = new int[2][];`
  `// can include two single-dimensional arrays`
- `int[][,] jArray2 = new int[3][,];`
  `// can include three two-dimensional arrays`
- jArray1 can store up to two single-dimensional arrays.
- jArray2 can store up to three two-dimensional, arrays [,] specifies the two-dimensional array.