wantsome
The friendly IT Academy

.NET Development

Data Structures

wantsome
The friendly IT Academy

Topics

- Iterations (Loops)
- Casting
- Arrays
- Lists

Iterations in .NET

- For
- Foreach
- While
- Do...while

For loop

- Number of iterations is known a priori
- Example

```
for (int i = 1; i <= 5; i++)
{
    Console.WriteLine(i);
}
```

- Executes instructions until condition is false

Foreach loop

- Execute a block of instructions for each element of an array or another collection
- Doesn't allow changing the collection (adding or removing items)
- Current index not available
- Example

```
int[] array = new int[] { 0, 1, 1, 2, 3, 5, 8, 13 };
foreach (int element in array)
{
    Console.WriteLine(element);
}
```

While

- Executes a block of instructions until the condition is false
- Example

```
int n = 1;
while (n < 6)
{
    Console.WriteLine("Current value of n is {0}", n);
    n++;
}
```

Do while

- Executes a block of instructions until the condition evaluates to false
- Loop executed at least once
- Example

```
int x = 0;
do
{
    Console.WriteLine(x);
    x++;
} while (x < 5);
```

Breaking out of loops

- **Continue** – go to next iteration in loop

```
for (int i = 0; i < 100; i++)
{
    if (i == 5)
    {
        continue;
    }
    Console.WriteLine(i);
}
```

Breaking out of loops

- **Break** – exit entire loop

```
for (int i = 0; i < 100; i++)
{
    if (i == 5)
    {
        break;
    }
    Console.WriteLine(i);
}
```

Breaking out of loops

- **Return** – Exit method
- **Throw** – Exit method with exception

Casting

- Converting an object from one data type to another
- Can be achieved in 2 ways:
  - Using the cast operator ()
  - Example

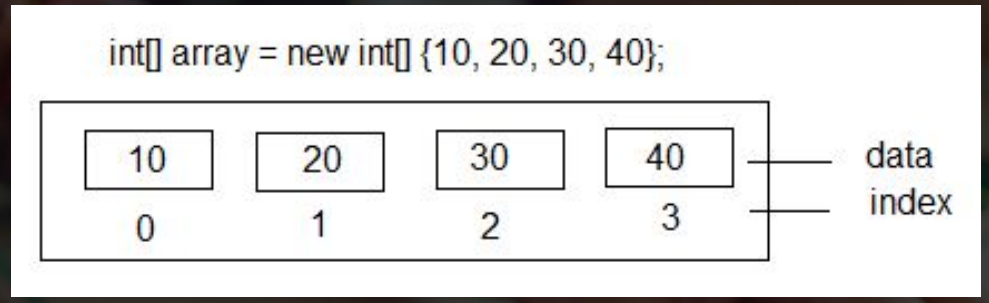
```
int x = (int)4.5;
```

  - Using **as** keyword – only for reference types
  - Example

```
int x = 5;
object y = x as object;
```

Arrays

- A collection of elements of the same type
- Stored in a contiguous memory location
- Fixed size



- Example

```
int[] numbers;
```

Arrays

- Initializing an array

```
int[] numbers = new int[6];
```

- Assigning values to an array

```
numbers[3] = 42;
```

- Assigning values on declaration

```
int[] numbers = { 3, 5, 2, 13, 7, 10 };
```

- Initialize and assign values

```
int[] numbers = new int[5] { 5, 2, 6, 8, 1 };
```

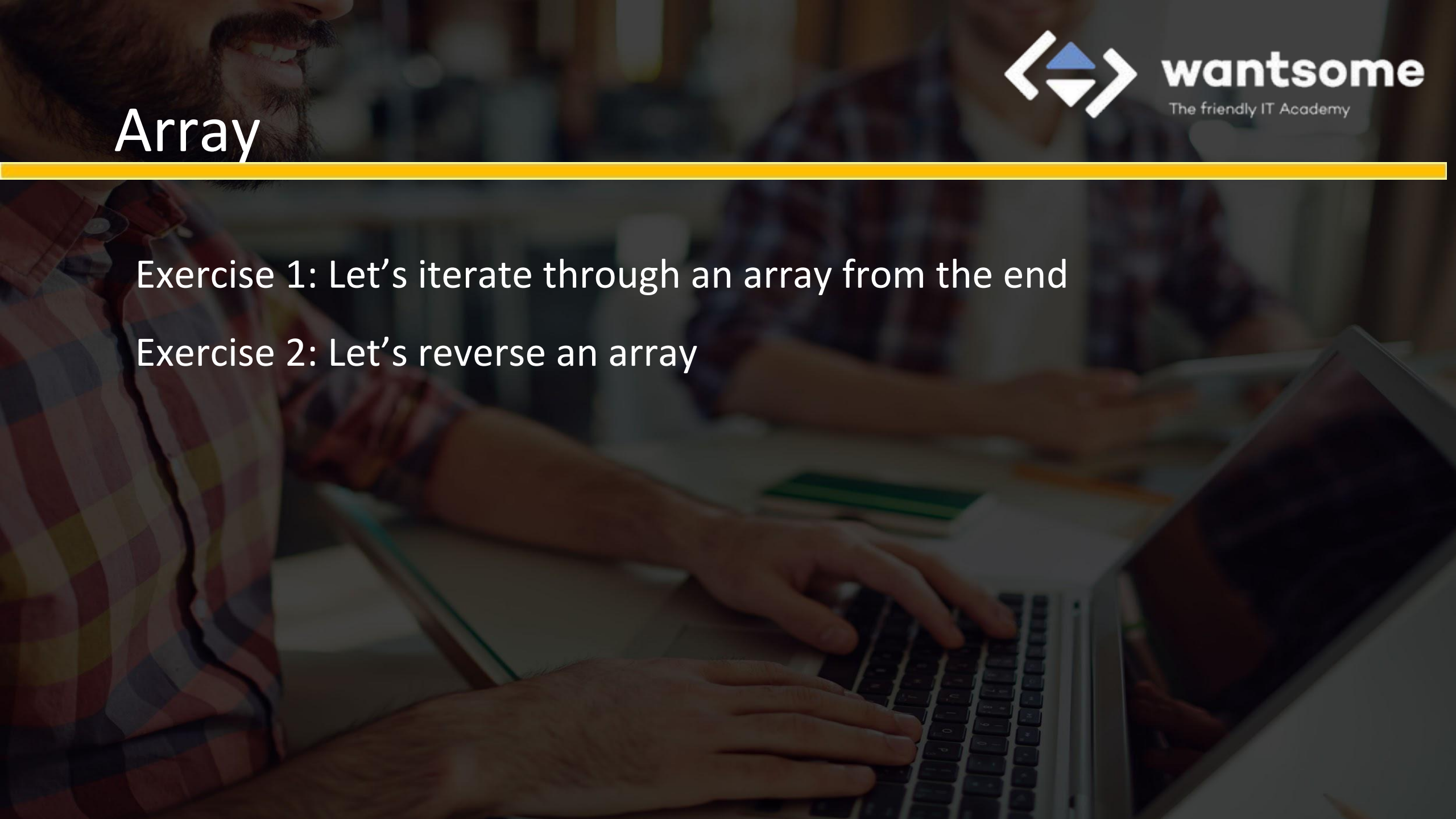
Array

- Accessing elements from arrays

```
int x = numbers[2];
```

Example – create an array with numbers from 10 to 20

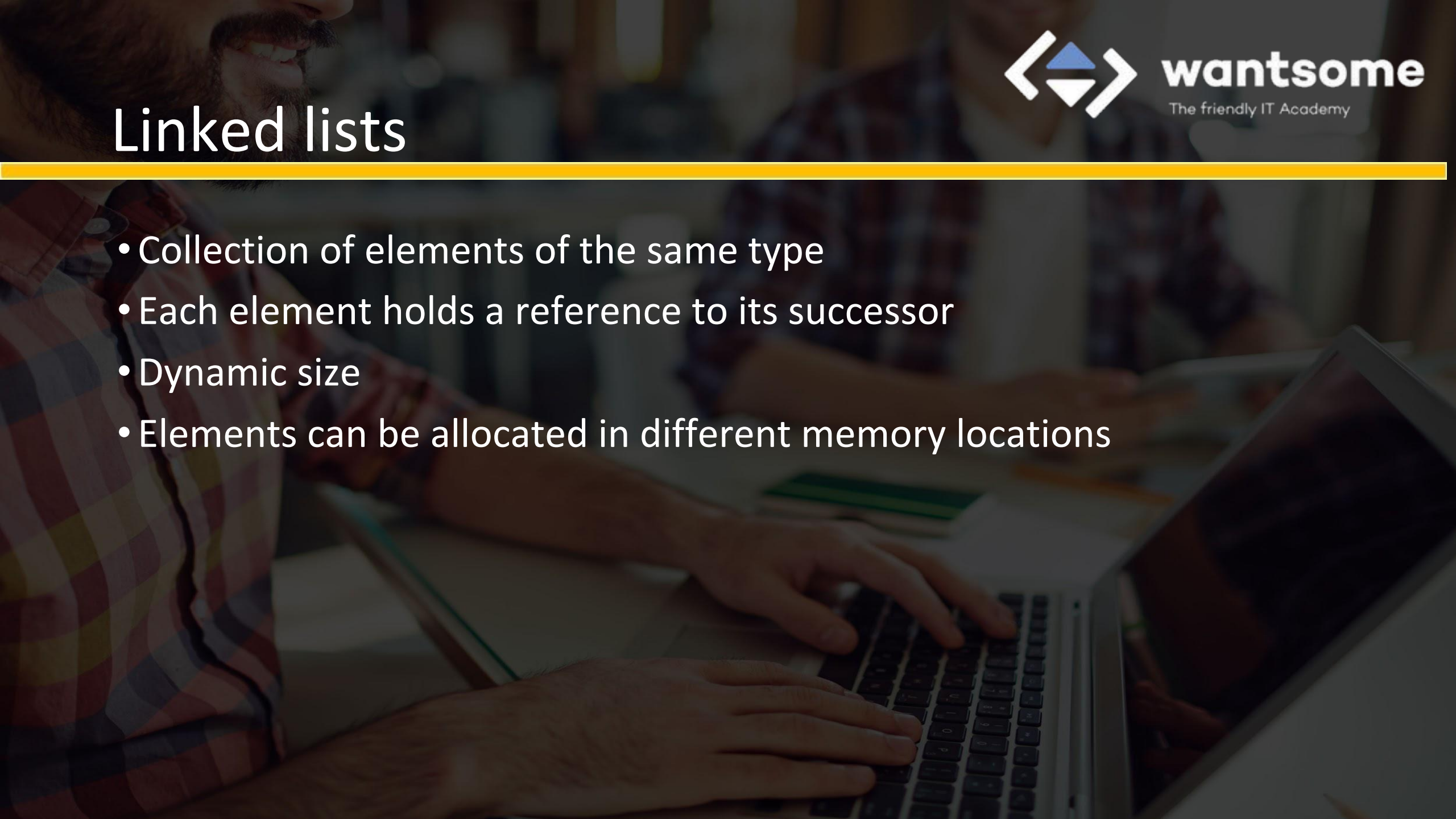
```
int[] numbers = new int[10];
for (int i = 0; i < 10; i++)
{
    numbers[i] = i + 10;
}
```



Array

Exercise 1: Let’s iterate through an array from the end

Exercise 2: Let’s reverse an array



Linked lists

- Collection of elements of the same type
- Each element holds a reference to its successor
- Dynamic size
- Elements can be allocated in different memory locations

Linked lists

Linked lists

- Accessing a specific element requires traversing the list
- Adding an element is easier at the beginning of the list
- Types:
  - Single linked
  - Double linked
- .NET implementation: LinkedList<T>

List<T>

- An array implementation that can grow in size
- Most commonly used
- Strongly typed

Exercises

1. Print to console all even numbers between 15 and 97.
2. Print all numbers divisible by 3 between 20 and 65.
3. Count all numbers divisible by 7 and multiple of 5, from 1400 to 2300 and print the result to console.
4. Write a program to guess a number between 1 and 10. To generate a random number, use Random class from .NET Framework.
5. Read a text from console and print it reversed.
6. Print numbers from 1 to 10 except 4 and 7.
7. Print the Fibonacci sequence from 0 to 50. (i.e. Exery next number is found by adding up those two before it: 0, 1, 1, 2, 3, 5, 8, 13, ...)

Exercises

1. Sum the values of an array and display it to console.
2. Calculate the average value of array elements.
3. Find the index of an element in an array.
4. Remove a specific element from an array.
5. Insert an element into an array at a specified position.
6. Find the maximum and minimum value of an array.
7. Find common elements between two arrays of integers.
8. Copy elements from an array into another.

Homework 1

FizzBuzz

For all numbers from 1 to 1000, print the following:

- If number is divisible by 3, print Fizz
- If number is divisible by 5, print Buzz
- If number is divisible by 3 and 5, print FizzBuzz
- Otherwise, print the number

Homework 2

Write a program to count the frequency of each element in an array

e. g. [1, 4, 5, 2, 1, 4, 3, 1, 2] should output:

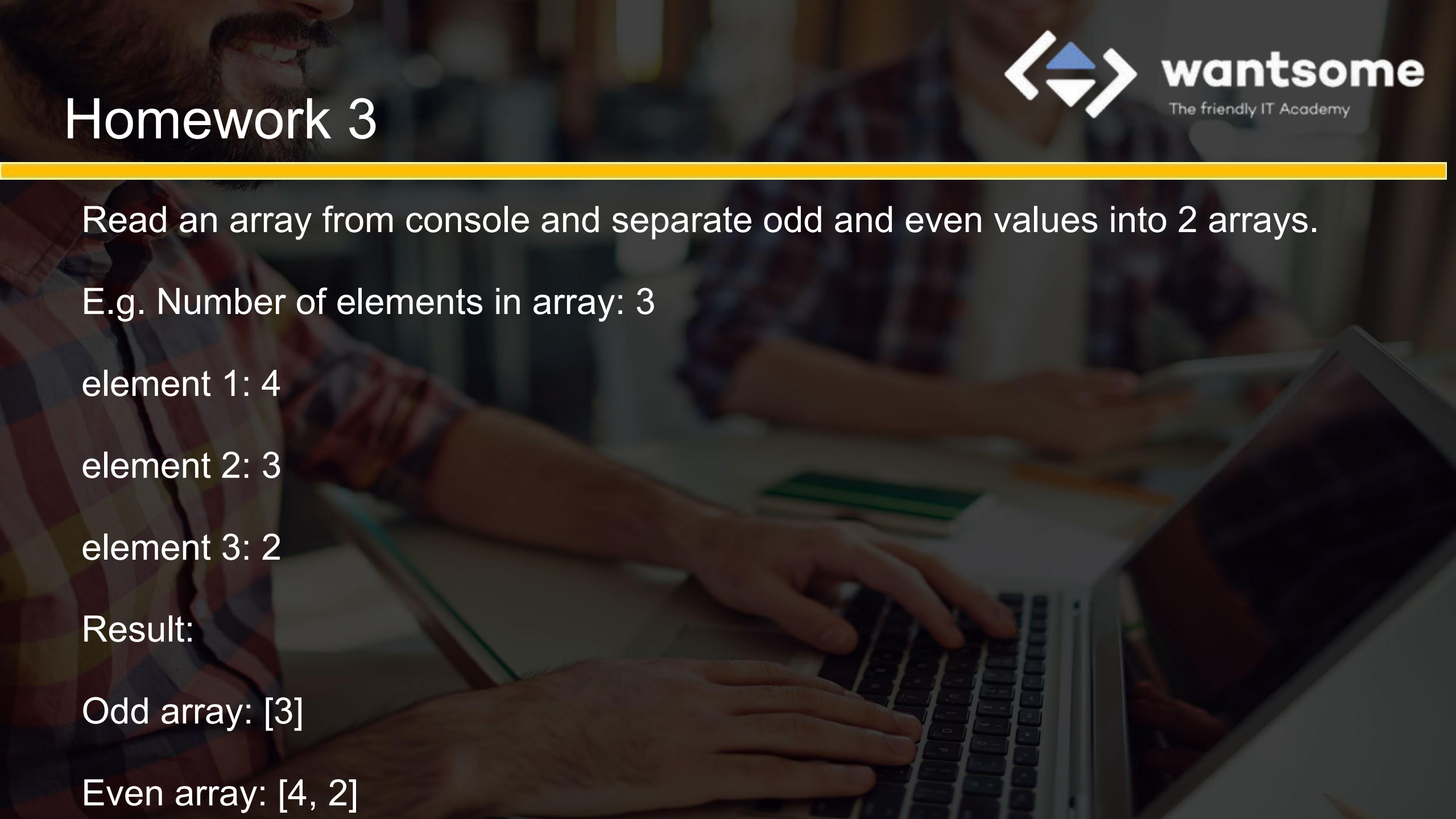
1 - 3 times

4 - 2 times

5 - one time

2 - 2 times

3 - one time



Homework 3

Read an array from console and separate odd and even values into 2 arrays.

E.g. Number of elements in array: 3

element 1: 4

element 2: 3

element 3: 2

Result:

Odd array: [3]

Even array: [4, 2]