Array in TypeScript

Array:

An array is a data structure in programming used to store and organize a collection of elements, where each element can be accessed using an index or a key. Arrays are particularly useful when you want to work with a list or a sequence of items of the same type.

In typescript, arrays can contain elements of various types, such as numbers, strings, objects, or even other arrays. Typescript allows you to define the type of elements that an array can contain, providing additional safety and clarity in your code.

PROBLEM:

If you want to store multiple items, like fruits, using individual variables for each item can become impractical, especially when dealing with a large number of items.

```
let fruit1 = "apple";
let fruit2 = "banana";
let fruit3 = "grapes";
let fruit4 = "strawberry";
let fruit5 = "orange";
// ... and so on
```

SOLUTION:

```
//Solution:
//Using an array, you can efficiently store and manage multiple items in a single variable.
let fruits = ["apple", "banana", "orange", "grapes", "strawberry"];
```

This approach is not only cleaner but also makes it easier to perform operations on the entire set of items, such as iterating through the fruits or dynamically adding/removing items. Arrays are a fundamental and powerful concept in programming for managing collections of data.

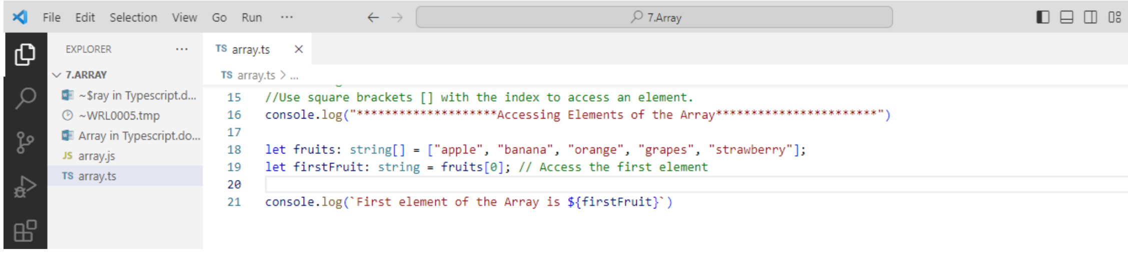
More about array:

- An array is a special variable, which can hold more than one value.
- An array can hold many values under a single name, and you can access the values by referring to an index number.
- In JavaScript, arrays always use numbered indexes.
- Array indexes start with 0.

In TypeScript, as in JavaScript, arrays support various operations and methods that allow you to manipulate, iterate over, and modify the content of arrays. Here are some common operations on arrays in TypeScript:

Accessing Elements:

Use square brackets [] with the index to access an element.



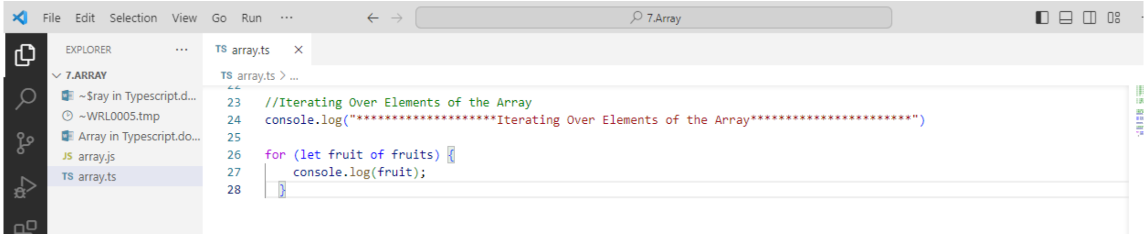
```
//Use square brackets [] with the index to access an element.
console.log("*******************Accessing Elements of the Array**********************")

let fruits: string[] = ["apple", "banana", "orange", "grapes", "strawberry"];
let firstFruit: string = fruits[0]; // Access the first element

console.log(`First element of the Array is ${firstFruit}`)
```

Iterating Over Elements:

Use for...of loop for a concise iteration over elements.

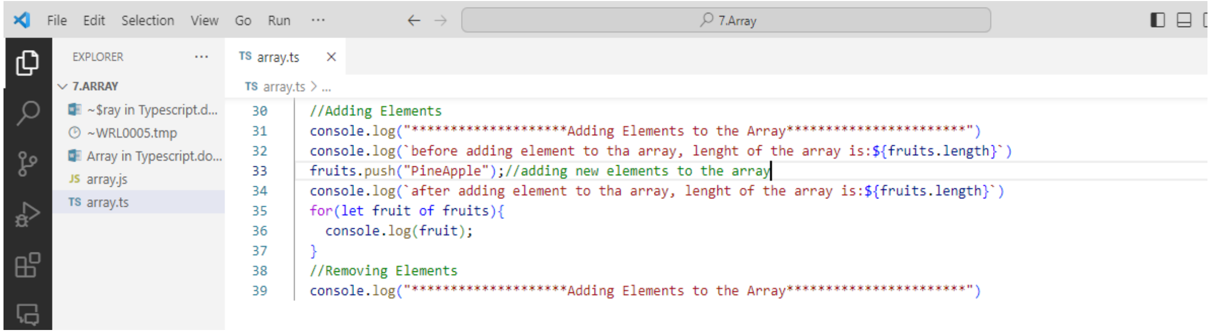


```
//Iterating Over Elements of the Array
console.log("*******************Iterating Over Elements of the Array**********************")

for (let fruit of fruits) {
    console.log(fruit);
 }
```

Adding Elements:

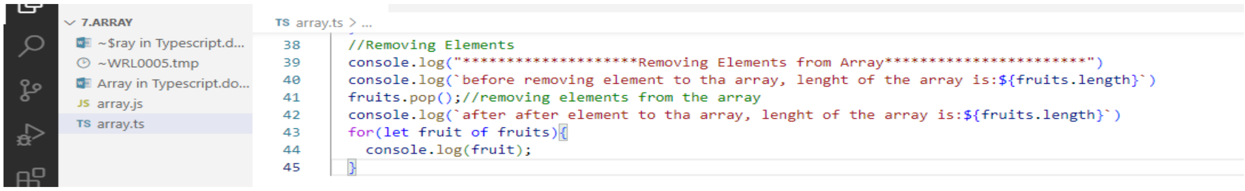
Use the push method to add elements to the end of the array.
The push() method adds a new element to an array (at the end).
The push() method returns the new array length.



```
//Adding Elements
console.log("*******************Adding Elements to the Array**********************")
console.log(`before adding element to tha array, lenght of the array is:${fruits.length}`)
fruits.push("PineApple");//adding new elements to the array
console.log(`after adding element to tha array, lenght of the array is:${fruits.length}`)
for(let fruit of fruits){
  console.log(fruit);
}
//Removing Elements
console.log("*******************Adding Elements to the Array**********************")
```

Removing Elements:

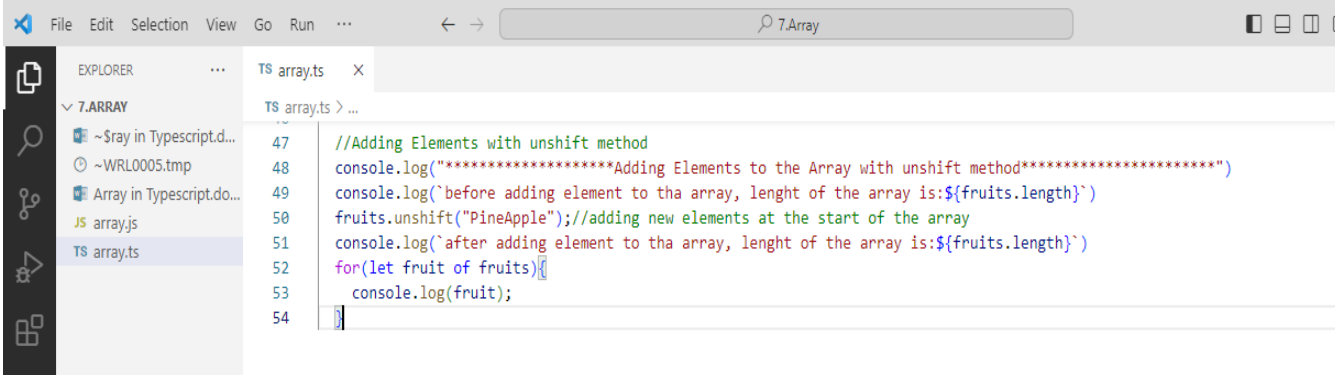
Use methods like pop (removes the last element) or splice (removes elements at a specified index).
The pop() method returns the value that was "popped out"



```
//Removing Elements
console.log("*******************Removing Elements from Array**********************")
console.log(`before removing element to tha array, lenght of the array is:${fruits.length}`)
fruits.pop();//removing elements from the array
console.log(`after after element to tha array, lenght of the array is:${fruits.length}`)
for(let fruit of fruits){
  console.log(fruit);
}
```

Unshift Method:

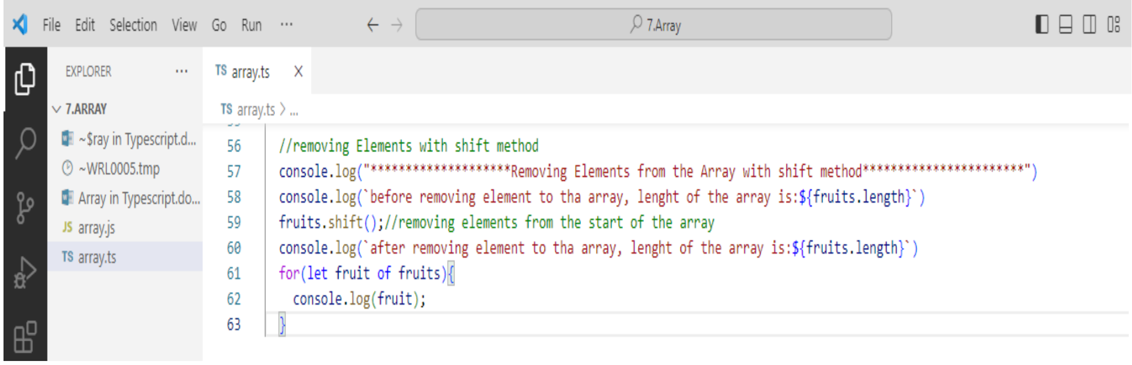
The unshift() method adds a new element to the beginning of an array.
All existing elements are "unshifted" to higher indexes.
It returns the new length of the array.



```
//Adding Elements with unshift method
console.log("*******************Adding Elements to the Array with unshift method**********************")
console.log(`before adding element to tha array, lenght of the array is:${fruits.length}`)
fruits.unshift("PineApple");//adding new elements at the start of the array
console.log(`after adding element to tha array, lenght of the array is:${fruits.length}`)
for(let fruit of fruits){
  console.log(fruit);
}
```

Shift Method:

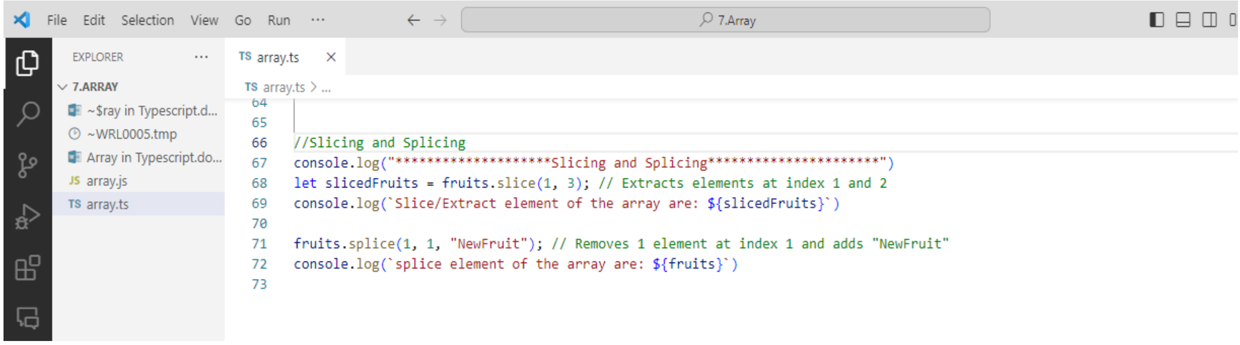
The shift() method removes the first element from an array.
All remaining elements are shifted to a lower index.
It returns the value of the element that was removed (shifted out).



```
//removing Elements with shift method
console.log("*******************Removing Elements from the Array with shift method**********************")
console.log(`before removing element to tha array, lenght of the array is:${fruits.length}`)
fruits.shift();//removing elements from the start of the array
console.log(`after removing element to tha array, lenght of the array is:${fruits.length}`)
for(let fruit of fruits){
  console.log(fruit);
}
```

Slicing and Splicing:

Use slice to extract a portion of an array and splice to add or remove elements at a specific index.



```


//Slicing and Splicing
console.log("*******************Slicing and Splicing*********************")
let slicedFruits = fruits.slice(1, 3); // Extracts elements at index 1 and 2
console.log(`Slice/Extract element of the array are: ${slicedFruits}`)

fruits.splice(1, 1, "NewFruit"); // Removes 1 element at index 1 and adds "NewFruit"
console.log(`splice element of the array are: ${fruits}`)

```