**Arrays**

**Array introduction**

An ***array*** is an ordered collection of values called ***elements***. Each array element is stored in a numeric location called an ***index***. An array is initialized by assigning an array variable with brackets [] containing comma-separated values.

Array elements may be of the same type or different types. Arrays increase in size as elements are added and decrease as elements are removed.

 Initializing and displaying array elements.

Graphical user interface, application

Description automatically generated

1. An empty array called "scores" is declared with [ ].
2. Three elements are added to the scores array at indexes 0, 1, and 2.
3. The three elements are output to the console.
4. The teams array is initialized with four elements.
5. All four elements are output to the console separated by commas.
6. **Adding and removing array elements**
7. An array is an ***Array object*** that defines numerous methods for manipulating arrays. A ***method*** is a function that is attached to an object and operates on data stored in the object. Methods are called by prefacing the method with the object. Ex: myArray.method();.

Table Array methods for adding and removing array elements.

| Method | Description | Example |
| --- | --- | --- |
| ***push(value)*** | Adds a value to the end of the array | let nums = [2, 4, 6];  nums.push(8); *// nums = [2, 4, 6, 8]* |
| ***pop()*** | Removes the last array element and returns the element | let nums = [2, 4, 6];  let x = nums.pop(); *// returns 6, nums = [2, 4]* |
| ***unshift(value)*** | Adds a value to the beginning of the array | let nums = [2, 4, 6];  nums.unshift(0); *// nums = [0, 2, 4, 6]* |
| ***shift()*** | Removes the first array element and returns the element | let nums = [2, 4, 6];  let x = nums.shift(); *// returns 2, nums = [4, 6]* |
| ***splice(startingIndex, numElemToDelete, valuesToAdd)*** | Adds or removes elements from anywhere in the array and returns the deleted elements (if any) | let nums = [2, 4, 6, 8, 10];  *// Deletes all elements from index 3 to the end*  nums.splice(3); *// nums = [2, 4, 6]*  *// Deletes 2 elements starting at index 0*  nums.splice(0, 2); *// nums = [6]*  *// Adds 3, 5 starting at index 0*  nums.splice(0, 0, 3, 5); *// nums = [3, 5, 6]*  *// Adds 7, 9, 11 starting at index 2*  nums.splice(2, 0, 7, 9, 11); *// nums = [3, 5, 7, 9, 11, 6]* |

**Looping through an array**

The array property ***length*** contains the number of elements in the array. The length property is helpful for looping through an array using a for loop.

Figure 6.8.1: Looping through an array with a for loop.

let groceries = ["bread", "milk", "peanut butter"];

*// Display all elements in groceries array*

for (i = 0; i < groceries.length; i++) {

console.log(i + " - " + groceries[i]);

}

0 - bread

1 - milk

2 - peanut butter

The ***for-of loop*** is a simplified for loop that loops through an entire array. The array name is placed after the of keyword in a for-of loop. Each time through the loop, the next array element is assigned to the variable in front of the of keyword.

Figure 6.8.2: Looping through an array with a for-of loop.

let groceries = ["bread", "milk", "peanut butter"];

*// Display all elements in groceries array*

for (let item of groceries) {

console.log(item);

}

bread

milk

peanut butter

The Array method ***forEach()*** also loops through an array. The forEach() method takes a function as an argument. The function is called for each array element in order, passing the element and the element index to the function.

Figure 6.8.3: Looping through an array with the forEach() method.

let groceries = ["bread", "milk", "peanut butter"];

*// Display all elements in groceries array*

groceries.forEach(function(item, index) {

console.log(index + " - " + item);

});

0 - bread

1 - milk

2 - peanut butter