Task 1: Inventory Tracker

|  |  |
| --- | --- |
| **Variable/Command** | **Value/Output** |
| inventory | ["apple", "banana", "cherry", "date", "elderberry"] |
| inventoryCount | [10, 15, 5, 20, 7] |
| i | 0 |
| console.log(`${inventory[i]}: ${inventoryCount[i]}`) | "apple: 10" |
| i | 1 |
| console.log(`${inventory[i]}: ${inventoryCount[i]}`) | "banana: 15" |
| i | 2 |
| console.log(`${inventory[i]}: ${inventoryCount[i]}`) | "cherry: 5" |
| i | 3 |
| console.log(`${inventory[i]}: ${inventoryCount[i]}`) | "date: 20" |
| i | 4 |
| console.log(`${inventory[i]}: ${inventoryCount[i]}`) | "elderberry: 7" |

Code Overview:

* Two arrays are defined: one holds fruit names (inventory), and the other holds corresponding counts (inventoryCount).
* A loop runs through the indices of the inventory array.
* At each iteration, the program prints the fruit and its count using template literals.

Explanation:

* Initialization:  
  The inventory array contains a list of fruits, and inventoryCount holds numerical values representing the count of each fruit.  
  *Example:* inventory[0] is "apple", and inventoryCount[0] is 10.
* Loop Execution:  
  A for loop iterates over each index (from 0 to inventory.length - 1).
  + On each iteration, the current index i is used to access elements in both arrays.
  + The console.log statement prints a formatted string showing the fruit name and its corresponding count.

Final Output:

> apple: 10

> banana: 15

> cherry: 5

> date: 20

> elderberry: 7

Task 2: Alphabetical Sorting

|  |  |
| --- | --- |
| **Variable/Command** | **Value/Output** |
| word | ["zebra", "apple", "mango", "cherry", "banana"] |
| i | 0 |
| j | 1 |
| "zebra" > "apple" | true |
| temp | "zebra" |
| words[0] | "apple" |
| words[1] | "zebra" |
| j | 2 |
| "zebra" > "mango" | true |
| temp | "zebra" |
| words[1] | "mango" |
| words[2] | "zebra" |
| ...Sorting continues until fully sorted... |  |
| console.log(words) | ["apple", "banana", "cherry", "mango", "zebra"] |

Code Overview:

* An unsorted array of words is provided.
* Two nested loops are used: the outer loop selects a word, and the inner loop compares it with the remaining words.
* When a word is found that should come earlier in alphabetical order, the two words are swapped.

Explanation:

* Initialization:  
  The words array is initialized with unsorted words such as ["zebra", "apple", "mango", "cherry", "banana"].
* Sorting Process:
  + The outer loop runs with index i from 0 to words.length - 1.
  + For each i, the inner loop runs with index j from i + 1 to the end of the array.
  + Comparison:  
    The condition if (words[i] > words[j]) checks if the word at index i comes after the word at index j alphabetically.
  + Swapping:  
    If the condition is true, the words are swapped using a temporary variable (temp), ensuring the lower alphabetical word is placed before the higher one.
  + This process continues until the entire array is sorted alphabetically.

Final Output:

> ["apple", "banana", "cherry", "mango", "zebra"]

Task 3: Unique Array Builder

|  |  |
| --- | --- |
| **Variable/Command** | **Value/Output** |
| uniqueNumbers | [] |
| randomNumber | 12 (random) |
| exists | false |
| uniqueNumbers.push(12) | [12] |
| randomNumber | 5 (random) |
| exists | False |
| uniqueNumbers.push(5) | [12, 5] |
| randomNumber | 18 (random) |
| exists | false |
| uniqueNumbers.push(18) | [12, 5, 18] |
| randomNumber | 12 |
| exists | true (skip) |
| ...Loop continues until 10 unique numbers are added... |  |
| console.log(words) | [12, 5, 18, 7, 9, 3, 15, 1, 20, 11] (Example output) |

Code Overview:

* The goal is to build an array of 10 unique random numbers between 1 and 20.
* A while loop continues until the array uniqueNumbers has 10 elements.
* Within the loop, a random number is generated, and a for loop checks whether this number already exists in the array.
* If the number is unique (i.e., not already in the array), it is added to uniqueNumbers.

Explanation:

* Initialization:  
  An empty array uniqueNumbers is created to store the unique random numbers.
* Generating Random Numbers:
  + The expression Math.floor(Math.random() \* 20) + 1 generates a random integer between 1 and 20.
  + The variable randomNumber holds this generated value.
* Checking Uniqueness:
  + A flag exists is initially set to false.
  + A for loop iterates over the current elements in uniqueNumbers to check if randomNumber is already present.
  + If a match is found, exists is set to true and the loop breaks.
* Adding Unique Numbers:
  + After checking, if exists remains false, the number is added to uniqueNumbers using push().
  + This process repeats until uniqueNumbers contains 10 numbers.

Final Output:

> [12, 5, 18, 7, 9, 3, 15, 1, 20, 11]

Task 4: Triangle Checker

|  |  |
| --- | --- |
| **Variable/Command** | **Value/Output** |
| sideA | 7 |
| sideB | 10 |
| sideC | 5 |
| sideA + sideB > sideC | 7 + 10 = 17 > 5 (true) |
| sideB + sideC > sideA | 10 + 5 = 15 > 7 (true) |
| sideA + sideC > sideB | 7 + 5 = 12 > 10 (true) |
| console.log(`The sides ${sideA}, ${sideB}, and ${sideC} form a valid triangle.`); | "The sides 7, 10, and 5 form a valid triangle." |

Code Overview:

* Three variables represent the sides of a potential triangle: sideA, sideB, and sideC.
* The code uses an if statement to check if the sum of any two sides is greater than the third.
* If all three conditions are met, the sides can form a valid triangle.

Explanation:

* Triangle Inequality Theorem:  
  For any three sides to form a triangle, the sum of any two sides must be greater than the remaining side:
  + sideA + sideB > sideC
  + sideB + sideC > sideA
  + sideA + sideC > sideB
* Execution Flow:
  + sideA is set to 7, sideB to 10, and sideC to 5.
  + Each condition is checked:
    - 7 + 10 > 5 → 17 > 5 is **true**.
    - 10 + 5 > 7 → 15 > 7 is **true**.
    - 7 + 5 > 10 → 12 > 10 is **true**.
  + Since all conditions are true, the code executes the first branch of the if statement.

Final Output:

> The sides 7, 10, and 5 form a valid triangle.