Assignment #4

1. \*\*Map Transformation:\*\*  
- Q: Given an array of integers, use the `map` method to square each element and return a new array with the squared values.  
  
2. \*\*Filter and Map Combination:\*\*  
- Q: Take an array of strings, filter out the ones with a length less than 5, and then capitalize the remaining strings using the `map` method.  
  
3. \*\*Sorting Objects:\*\*  
- Q: Given an array of objects with a 'price' property, use the `sort` method to arrange them in descending order based on their prices.  
  
4. \*\*Reduce for Aggregation:\*\*  
- Q: Use the `reduce` method to find the total sum of all even numbers in an array of integers.  
  
5. \*\*Find and Modify:\*\*  
- Q: Given an array of objects with 'id' properties, use the `find` method to locate an object with a specific 'id' and update its 'status' property to 'completed'.  
  
6. \*\*Chaining Methods:\*\*  
- Q: Create a chain of array methods to find the average of all positive numbers in an array of mixed integers and return the result rounded to two decimal places.  
  
7. \*\*Conditional Filtering:\*\*  
- Q: Implement a function that takes an array of objects with 'age' properties and returns an array of those who are adults (age 18 and above) using the `filter` method.  
  
8. \*\*Advanced Sorting:\*\*  
- Q: Sort an array of strings based on their lengths in ascending order. If two strings have the same length, maintain their relative order in the sorted array.  
  
9. \*\*Nested Array Operations:\*\*  
- Q: Given an array of arrays containing numbers, use a combination of array methods to flatten the structure and then calculate the sum of all the numbers.  
  
10. \*\*Error Handling with Find:\*\*  
- Q: Modify the `find` method to handle the scenario where the desired element is not found, returning a custom default object instead.  
  
11. \*\*Map Method:\*\*  
- Q: How does the `map` method work in JavaScript, and can you provide an example of when you might use it to manipulate an array of objects?  
  
12. \*\*Filter Method:\*\*  
- Q: Explain the purpose of the `filter` method. Provide an example where you use `filter` to extract elements from an array based on a specific condition.  
  
13. \*\*Sort Method:\*\*  
- Q: Discuss the default behavior of the `sort` method for strings and numbers. How would you use a custom comparison function to sort an array of objects by a specific property?  
  
14. \*\*Reduce Method:\*\*  
- Q: Describe the purpose of the `reduce` method and provide an example where you use it to compute a single value from an array of numbers.  
  
15. \*\*Find Method:\*\*  
- Q: How does the `find` method differ from `filter`? Give an example of a scenario where using `find` is more appropriate than `filter`.  
  
16. \*\*Combining Methods:\*\*  
- Q: Create a chain of array methods (`map`, `filter`, `reduce`, etc.) to transform an array of strings into a single concatenated string with a specific condition.  
  
17. \*\*Callback Functions:\*\*  
- Q: Explain the concept of callback functions in the context of array methods. Provide an example of using a callback function with the `map` method.  
  
18. \*\*Error Handling:\*\*  
- Q: How would you handle potential errors when using array methods like `find` or `reduce`? Provide an example of error handling in such a scenario.  
  
19. \*\*Immutable Operations:\*\*  
- Q: Discuss the importance of immutability when working with array methods. Demonstrate how you would perform immutable operations using methods like `map` or `filter`.  
  
20. \*\*Performance Considerations:\*\*  
- Q: Compare the performance implications of using `map` versus `forEach`. In what scenarios would you prefer one over the other, and why?