Diff between Map and forEach

1. **Purpose**:
   * **map**: Transforms each element in an array and returns a new array with the transformed elements.
   * **forEach**: Iterates over each element in an array and performs a side effect, such as logging each element or modifying an external variable. It does not return a new array.
2. **Return Value**:
   * **map**: Returns a new array with the same length as the original array, with each element transformed by the provided function.
   * **forEach**: Returns **undefined** as it only performs operations on each element without creating a new array.
3. **Mutability**:
   * **map**: Does not modify the original array; instead, it returns a new array with the transformations applied.
   * **forEach**: Can modify the original array if the callback function directly alters the elements.
4. **Chaining**:
   * **map**: Since it returns a new array, you can chain additional array methods (e.g., **filter**, **reduce**) directly on the result.
   * **forEach**: Cannot be chained because it returns **undefined**.
5. **Callback Function**:
   * **map**: The callback function should return a value, which becomes the element in the new array.
   * **forEach**: The callback function is used for performing operations and may not return anything.
6. **Use Case**:
   * **map**: Used when you want to create a new array based on the transformation of each element in an array.
   * **forEach**: Used when you want to perform an action on each element, such as logging, without returning a new array.
7. **Functional Programming**:
   * **map**: Fits well with functional programming paradigms because it transforms each element and returns a new array, maintaining immutability.
   * **forEach**: Is more aligned with imperative programming as it performs side effects on the elements of an array.
8. **Performance**:
   * **map**: Since it creates a new array, there may be slightly more overhead compared to **forEach**, which doesn't create a new array.
   * **forEach**: Can be faster for operations where you don't need to create a new array.
9. **Handling Array Length**:
   * **map**: Calls the callback function for each element in the array, even for empty slots if the array has holes.
   * **forEach**: Also calls the callback function for each element in the array, even for empty slots if the array has holes.
10. **Parallelism**:
    * **map**: In some cases, can be parallelized since the transformations of elements are independent of each other.
    * **forEach**: Generally not suitable for parallelism because the callback function might depend on external state or previous elements.