**Greedy Algorithms in JavaScript:**

Greedy algorithms are a class of algorithms that make locally optimal choices at each stage with the hope of finding a global optimum solution. In other words, they select the best choice available at each step without considering the overall structure of the problem. While greedy algorithms are simple and easy to implement, they do not always guarantee an optimal solution.

**Characteristics:**

* Local Optimality: Greedy algorithms make decisions based solely on the current state of the problem, without considering the consequences of those decisions on future steps.
* No Backtracking: Once a decision is made, it cannot be undone. Greedy algorithms do not backtrack or reconsider previous decisions.
* Efficiency: Greedy algorithms are often efficient in terms of time complexity since they make quick decisions based on locally optimal choices.

**Example:**

Consider the problem of making change using the fewest possible coins. A greedy algorithm for this problem would repeatedly choose the largest denomination coin that is less than or equal to the remaining amount until the remaining amount becomes zero. This strategy may not always yield the optimal solution, but it often produces a reasonably good solution in practice.

**Applications:**

* Optimization Problems: Greedy algorithms are commonly used for optimization problems where finding an optimal solution is not feasible due to the problem's complexity.
* Scheduling and Routing: Greedy algorithms are used in scheduling tasks or routing paths where making locally optimal decisions leads to reasonably good solutions.

**Limitations:**

* Suboptimal Solutions: Greedy algorithms do not guarantee an optimal solution for all problems. In some cases, they may produce suboptimal solutions that are not globally optimal.
* Need for Proof: It is essential to prove the correctness of a greedy algorithm and assess whether it consistently produces acceptable solutions for a given problem.