JavaScript Algorithms and Data Structures Masterclass

# Section 6: Recursion

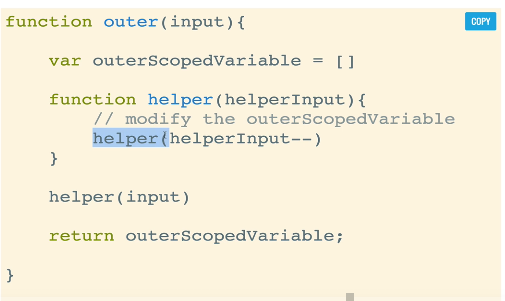
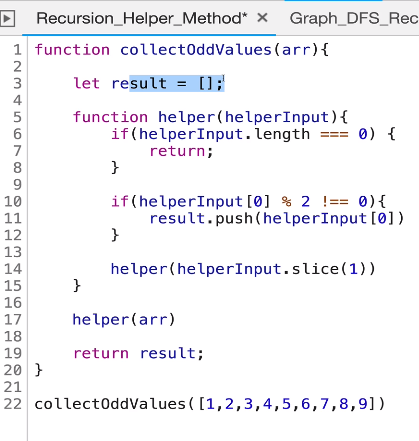
## What is Recursion

* A *function* that calls itself
  + **ex**. JSON.parse or JSON.stringify;
  + **ex**. doc.getElementById or DOM traversal
  + **ex**. Object traversal

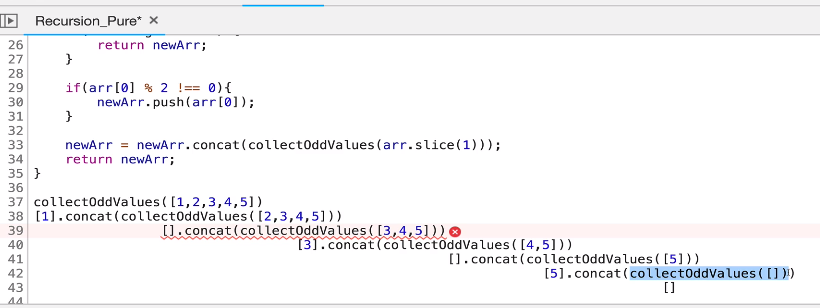
## The Call Stack

* A stack data structure
  + Anytime a function is invoked, it gets pushed onto top of call stack
  + Return means the function is popped off the stack
* Invoke the same function w/ diff input until you hit your base case
  + **Base Case**:
    - Condition to end recursion
* Recursion = Base Case + Diff Input
* **Pitfalls**:
  + Base Case may be wrong == infinite stack calls!
  + Forgetting to return or return wrong value
  + Stack Overflow == infinite stack calls

## Helper Method Pattern for Recursion

* 
  + Recursive function is outside == the “outer” function
    - Helper Recursive function is inside (contains logic)
      * It is used so that you can compile a result for an array since you can save it
      * 

## Pure Recursion Pattern

* 
* Utilizes .concat to add to a single value which you can then fully concat
* **Tips for Pure Recursion**:
  + For Arrays:
    - use .slice, spread operator, .concat
      * makes copies of the arrays so they don’t mutate
  + For Strings
    - they are immutable so you need to use:
      * .slice, .substr, or .substring to make copies of the strings
  + For Objects
    - use:
      * Object.assign or Spread Operator