2693. Call Function with Custom Context

Description

Enhance all functions to have the `callPolyfill` method. The method accepts an object `obj` as it's first parameter and any number of additional arguments. The `obj` becomes the `this` context for the function. The additional arguments are passed to the function (that the `callPolyfill` method belongs on).

For example if you had the function:

```
function tax(price, taxRate) {
  const totalCost = price * (1 + taxRate);
  console.log(`The cost of ${this.item} is ${totalCost}`);
}
```

Calling this function like `tax(10, 0.1)` will log `"The cost of undefined is 11"`. This is because the `this` context was not defined.

However, calling the function like `tax.callPolyfill({item: "salad"}, 10, 0.1)` will log `"The cost of salad is 11"`. The `this` context was appropriately set, and the function logged an appropriate output.

Please solve this without using the built-in `Function.call` method.

Example 1:

```
 Input:
fn = function add(b) {
  return this.a + b;
}
args = [{"a": 5}, 7]
 Output: 12
 Explanation:
fn.callPolyfill({"a": 5}, 7); // 12
callPolyfill sets the "this" context to {"a": 5}. 7 is passed as an argument.
```

Example 2:

```
 Input:
fn = function tax(price, taxRate) {
 return `The cost of the ${this.item} is ${price * taxRate}`;
}
args = [{"item": "burger"}, 10, 1.1]
 Output: "The cost of the burger is 11"
 Explanation: callPolyfill sets the "this" context to {"item": "burger"}. 10 and 1.1 are passed as additional arguments.
```

Constraints:

- `typeof args[0] == 'object' and args[0] != null`
- `1 <= args.length <= 100`
- `2 <= JSON.stringify(args[0]).length <= 10^5`