Problem 2754: Bind Function to Context

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Enhance all functions to have the

bindPolyfill

method. When

bindPolyfill

is called with a passed object

obj

, that object becomes the

this

context for the function.

For example, if you had the code:

function f() { console.log('My context is ' + this.ctx); } f();

The output would be

"My context is undefined"

. However, if you bound the function:

function f() { console.log('My context is ' + this.ctx); } const boundFunc = f.boundPolyfill({ "ctx": "My Object" }) boundFunc();

The output should be

"My context is My Object"

.

You may assume that a single non-null object will be passed to the

bindPolyfill

method.

Please solve it without the built-in

Function.bind

method.

Example 1:

Input:

fn = function f(multiplier) {   return this.x * multiplier; } obj = {"x": 10} inputs = [5]

Output:

50

Explanation:

const boundFunc = f.bindPolyfill({"x": 10}); boundFunc(5); // 50 A multiplier of 5 is passed as a parameter. The context is set to {"x": 10}. Multiplying those two numbers yields 50.

Example 2:

Input:

fn = function speak() {   return "My name is " + this.name; } obj = {"name": "Kathy"} inputs = []

Output:

"My name is Kathy"

Explanation:

const boundFunc = f.bindPolyfill({"name": "Kathy"}); boundFunc(); // "My name is Kathy"

Constraints:

obj

is a non-null object

0 <= inputs.length <= 100

Can you solve it without using any built-in methods?

Code Snippets

JavaScript:

```
/**
* @param {Object} obj
* @return {Function}
*/
Function.prototype.bindPolyfill = function(obj) {

}
```

TypeScript:

```
type Fn = (...args) => any

interface Function {
```

```
bindPolyfill(obj: Record<any, any>): Fn;
}

Function.prototype.bindPolyfill = function(obj): Fn {

}
```

Solutions

JavaScript Solution:

```
/**
* Problem: Bind Function to Context
* Difficulty: Medium
* Tags: dp
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {Object} obj
* @return {Function}
*/
Function.prototype.bindPolyfill = function(obj) {

}
```

TypeScript Solution:

```
/**
* Problem: Bind Function to Context
* Difficulty: Medium
* Tags: dp
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

```
type Fn = (...args) => any

interface Function {
bindPolyfill(obj: Record<any, any>): Fn;
}

Function.prototype.bindPolyfill = function(obj): Fn {

}
```