BadOPCode Contains (#28) ... History

4 contributors

73 lines (60 sloc) | 2.28 KB

```typescript
/**
 * @namespace NoDash
 * @author Shawn Rapp
 * @license MIT
 */
"use strict";

export interface IMergeBehavior {
    [key: string]: (originalObject: any, newObject: any) => any;
}

const getObjTypeName = (obj:any) => {
    if (obj === null) return "Null";
    if (obj === undefined) return "Undefined";
    return obj.constructor?.name || "Unknown";
};

const handleDefinedBehavior = (originalObject: any, newObject: any, behavior?: IMergeBehavior) => {
    const originalTypeName = getObjTypeName(originalObject);
    const newTypeName = getObjTypeName(newObject);
    if (!behavior) {
        return;
    }

    if (behavior[`${originalTypeName}To${newTypeName}`]) {
        return behavior[`${originalTypeName}To${newTypeName}`](originalObject, newObject);
    }

    if (behavior[originalTypeName] !== undefined) {
        return behavior[originalTypeName](originalObject, newObject);
    }
};

const handleDefaultBehavior = (originalObject: any, newObject: any, behavior?: IMergeBehavior) => {
    const originalTypeName = getObjTypeName(originalObject);
    const newTypeName = getObjTypeName(newObject);
    if (originalTypeName === "Object" && newTypeName === "Object") { // built-in behavior
        for (const p in newObject) {
            if (isPrototypePolluted(p)) continue;
            originalObject[p] = Merge(originalObject[p], newObject[p], behavior);
        }

        return originalObject;
    }

    return newObject;
};

const isPrototypePolluted = (key: any) => {
    return ["__proto__", "constructor", "prototype"].includes(key);
};

/**
 * Recursively merge two objects together.
 * @param originalObject The base object. Properties here will be overwritten
 * by properties that also exist in newObject.
 * @param newObject Properties in this object that are also in the original will
 * be overwritten by the values in this object.
 */
export const Merge = (originalObject: any, newObject: any, behavior?: IMergeBehavior) => {
    const definedBehaviorResults = handleDefinedBehavior(originalObject, newObject, behavior);
    if (definedBehaviorResults !== undefined) {
        return definedBehaviorResults;
    }

    if (typeof (newObject) === "undefined") {
        return originalObject;
    }

    return handleDefaultBehavior(originalObject, newObject, behavior);
};

export default Merge;
```