Modulo

Alumno: Victor Jair Salmeron

const castPascalCaseToSnakeCase = (pascalCaseString) => {

return pascalCaseString.split(/(?=[A-Z])/).join('\_').toLowerCase();

}

const castSnakeCaseToPascalCase = (snakeCaseString) => {

const words = snakeCaseString.split('\_');

return words.map(word => word[0].toUpperCase() + word.substr(1)).join('');

}

const castPasalCaseToCamelCase = (CamelString) => {

return CamelString.replace(/\W+(.)/g, function(match, chr)

{

return chr.toUpperCase();

});

}

module.exports = {castPascalCaseToSnakeCase};

const testCastPasalCaseToCamelCase = () => {

castPasalCaseToCamelCase("esto es un ejemplo") === "estoEsUnEjemplo" ? console.log("Test 1 paso") : console.log("Test 1 fallo")

console.log(castPasalCaseToCamelCase("esto es un ejemplo"))

}

const testCastPascalCaseToSnakeCase = () => {

executeTest("EstoEsUnEjemplo","esto\_es\_un\_ejemplo",castPascalCaseToSnakeCase);

executeTest("otroEjemplo","otro\_ejemplo",castPascalCaseToSnakeCase);

executeTest("otroejemplo","otroejemplo",castPascalCaseToSnakeCase);

//castPascalCaseToSnakeCase("EstoEsUnEjemplo") === "esto\_es\_un\_ejemplo" ? console.log("Test 1 pass") : console.log("Test 1 fail")

//castPascalCaseToSnakeCase("otroEjemplo") === "otro\_ejemplo" ? console.log("Test 2 pass") : console.log("Test 2 fail")

//castPascalCaseToSnakeCase("otroejemplo") === "otroejemplo" ? console.log("Test 3 pass") : console.log("Test 3 fail")

}

const testCastSnakeCaseToPascalCase = () => {

castSnakeCaseToPascalCase("esto\_es\_un\_ejemplo") === "EstoEsUnEjemplo" ? console.log("Test 1 pass") : console.log("Test 1 fail")

castSnakeCaseToPascalCase("otro\_ejemplo") === "OtroEjemplo" ? console.log("Test 2 pass") : console.log("Test 2 fail")

castSnakeCaseToPascalCase("otroejemplo") === "Otroejemplo" ? console.log("Test 3 pass") : console.log("Test 3 fail")

}

const testCastPascalCaseToSnakeCase2 = () => {

const pascalCaseString = "EstoEsUnEjemplo";

const expectedSnakeCaseString = "Esto\_es\_un\_ejemplo";

const actualSnakeCaseString = castPascalCaseToSnakeCase(pascalCaseString);

if (actualSnakeCaseString !== expectedSnakeCaseString) {

console.log(`Test fail expected output: ${expectedSnakeCaseString} actual output: ${actualSnakeCaseString}`)

} else {

console.log("El test paso exitosamente")

}

}

const executeTest = (input, expectedOutput, functionToExecute) => {

const actualOutput = functionToExecute(input)

if (actualOutput !== expectedOutput) {

console.log(`Test fail. Expected output: ${expectedOutput} actual output: ${actualOutput}`)

} else {

console.log("Test pass")

}

}

testCastPascalCaseToSnakeCase();

//executeTest();

//testCastPascalCaseToSnakeCase2();

//testCastPasalCaseToCamelCase();

//testCastSnakeCaseToPascalCase();

* ¿Cómo elegiste los casos de prueba para validar el correcto funcionamiento del sistema?
* Probar el código con resultados esperados.
* Probar con cadenas llamando a la función
* Probar comparando cadenas con los resultados esperados
* ¿Después de concluir el reto refactorizaste el código? ¿Qué cambios realizaste?
* Modifique las funciones de prueba para que me dieran tres pruebas con datos diferentes, posteriormente hice una prueba executeTest para ahorrar tiempo en la prueba de las funciones ya que esta prueba me daba la oportunidad de probar todas las anteriores sin la necesidad de escribir una prueba para cada caso
* ¿Utilizaste un método común o múltiples métodos individuales? ¿Por qué?
* Utilice un método común ya que la función era básicamente la misma en las conversiones, una cadena de entrada, una de salida y la llamada a la función así que se simplifico con un método común que ingresando los parámetros nos diera los resultados.