Ingeniería Web

Carlos Mauricio Duque Restrepo
Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas

Javascript - Closures

Los Closures son uno de los conceptos clave de JavaScript y permiten a los desarrolladores escribir mejor código. Por lo general, están alejados de los desarrolladores que no los usaron durante años y consideran que si no los han necesitado hasta ahora, también pueden vivir sin ellos.

Los Closures no son difíciles de entender pero requieren un cambio de paradigma de los otros lenguajes principales. Tienes que mirar las funciones y el alcance de la función con una perspectiva diferente para dominar los closures.

Javascript - Closures

Un closure es una función que tiene acceso a la variable desde el alcance de otra función. Esto se logra creando una función dentro de una función. Por supuesto, la función externa no tiene acceso al ámbito interno.

- Closures son funciones anidadas que tienen acceso al ámbito externo.
- Después de devolver la función externa, al mantener una referencia a la función interna (closures), evitamos que se destruya el alcance externo.

Javascript - Closures

```
function buildName(name) {
    var greeting = "Hello, " + name + "!";
    var sayName = function() {
        var welcome = greeting + " Welcome!";
        console.log(greeting);
    };
    return sayName;
}

var sayMyName = buildName("John");
sayMyName();  // Hello, John. Welcome!
sayMyName();  // Hello, John. Welcome!
sayMyName();  // Hello, John. Welcome!
```

```
function buildContor(i) {
    var contor = i;
    var displayContor = function() {
        console.log(contor++);
        contor++;
    };
    return displayContor;
}

var myContor = buildContor(1);
myContor(); // 1
myContor(); // 2
myContor(); // 3

// new closure - new outer scope - new contor variable
var myOtherContor = buildContor(10);
myOtherContor(); // 10
myOtherContor(); // 11

// myContor was not affected
myContor(); // 4
```

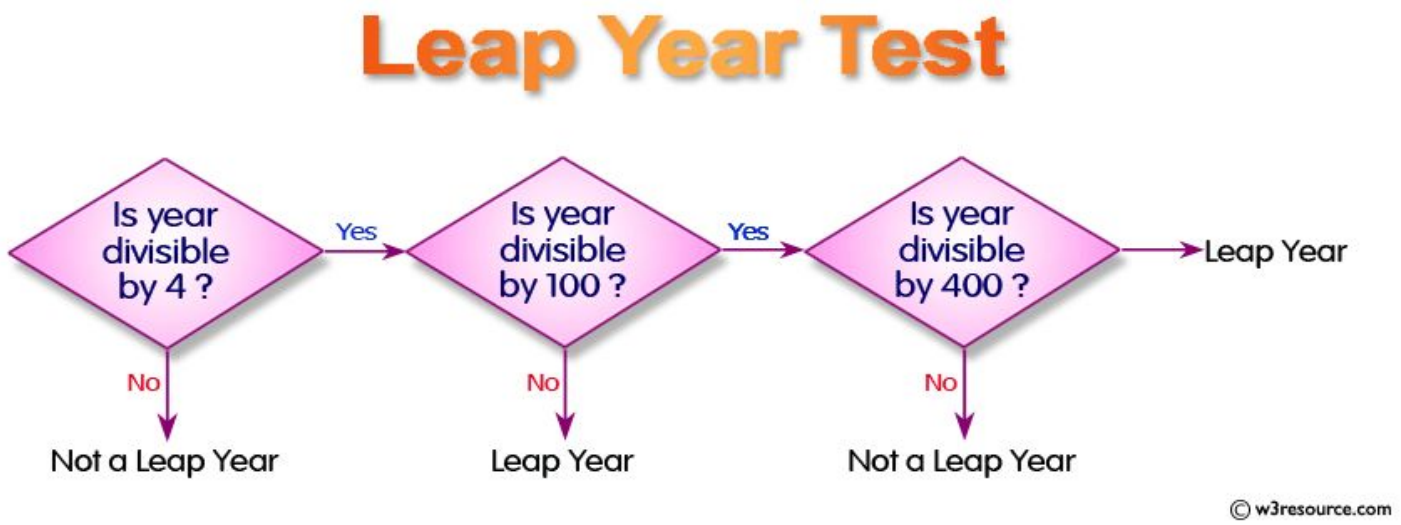
```
function initializeData() {
    var myVar = 1;
    return {
        getVar: function() {
            return myVar;
        },
        setVar: function(v) {
            myVar = v;
        }
    };
}

obj = initializeData();

console.log(obj.getVar()); // 1

obj.setVar(2);
console.log(obj.getVar()); // 2

obj.setVar("string");
console.log(obj.getVar()); // string
```

Ejercicios de Clase

1. Crear un programa javascript que permita verificar si una cadena de texto es un pangrama.
   a. Pangrama: Cadena de texto que usa todas las letras del alfabeto
   b. debe imprimir el resultado utilizado innerhtml
   c. debe imprimir en Consola
2. Resolver los Siguientes ejercicios; enviar evidencia de la respuesta(Puede ser imágenes)
   a. https://trailhead.salesforce.com/es-MX/content/learn/modules/modern-javascript-development?trailmix_creator_id=strailhead&trailmix_slug=prepare-for-your-salesforce-javascript-developer-i-credential
3. Resolver los Siguientes ejercicios; enviar evidencia de la respuesta(Puede ser imágenes, incluye únicamente las dos primeras unidades)
   a. https://trailhead.salesforce.com/es-MX/content/learn/modules/coding-for-web-accessibility?trailmix_creator_id=strailhead&trailmix_slug=prepare-for-your-salesforce-javascript-developer-i-credential

Ejercicios de Clase

4. Escriba un programa de JavaScript para determinar si un año dado es un año bisiesto en el calendario gregoriano.
   a. El año debe ingresarse mediante HTML
   b. La respuesta debe mostrarse en el HTML de forma legible para el usuario



Ejercicios de Clase

5. Se le proporciona una lista de enteros (X) no vacía. Para esta tarea, debe devolver una lista que conste solo de los elementos no únicos en esta lista. Para hacerlo, deberá eliminar todos los elementos únicos (elementos que están contenidos en una lista determinada solo una vez). Al resolver esta tarea, no cambie el orden de la lista. Ejemplo: [1, 2, 3, 1, 3] 1 y 3 elementos no únicos y el resultado será [1, 3, 1, 3].

Ejercicios de Clase

6. Los números romanos provienen del antiguo sistema de numeración romano. Se basan en letras específicas del alfabeto que se combinan para significar la suma (o, en algunos casos, la diferencia) de sus valores. Los diez primeros números romanos son:

   I, II, III, IV, V, VI, VII, VIII, IX y X.

   El sistema de números romanos se basa en decimales pero no es posicional directamente y no incluye un cero. Los números romanos se basan en combinaciones de estos siete símbolos:

Ejercicios de Clase

```
romanNumerals(6) == 'VI'
romanNumerals(76) == 'LXXVI'
romanNumerals(13) == 'XIII'
romanNumerals(44) == 'XLIV'
romanNumerals(3999) == 'MMMCMXCIX'
```

Numeral	Value
I	1 (unus)
V	5 (quinque)
X	10 (decem)
L	50 (quinquaginta)
C	100 (centum)
D	500 (quingenti)
M	1,000 (mille)