11-9-2019

Arredondo Benítez Emiliano - Víctor Enrique Lino Hernández – Nieves Diaz Carlos Jaid – Pérez Romero Jorge Andrés

Ingeniería del conocomiento – 3NM71

Proyecto Banco

Equipo 9

# CODIFICACIÓN Y DECODIFICACIÓN EN LA FUNCIÓN DE TRANSACCIÓN

## Codificación



5

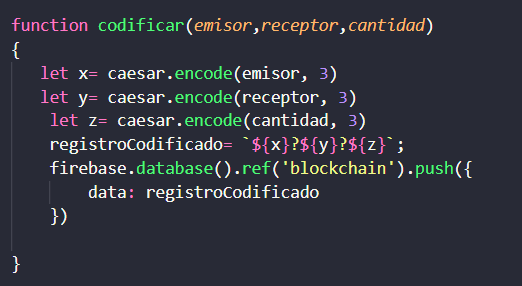
4

3

2

1

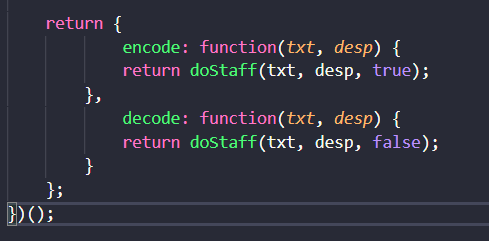
1. Para la codificación y decodificación primero se obtienen los datos a encriptar, por lo cual, se agrega en primera instancia un event handler, y se envían a variables los valores deseados de acuerdo a las denominaciones de los billetes.
2. La función suma, simplemente suma las cantidades de billetes multiplicadas por su denominación para formar un valor “total”, que no se guardará en ninguna variable y será usado como argumento para la función codificar, de forma directa.
3. Emisor y receptor se declaran obteniendo sus valores de acuerdo a su ID de elemento.
4. Por último, se envían los elementos “Emisor”, “Receptor” y “Cantidad” (la cual está siendo multiplicada por 132 a manera de ser usado el valor como llave de encriptación) como argumentos para la función codificar.
5. Se imprime la suma total de la cantidad de la transacción y se imprime.



2

1

1. Se declaran los valores que serán enviados a la base de datos, y directamente se les asigna el valor de una variable objeto, con el método “encode”, enviando como argumentos los datos que recibe la función codificar, y un numero de desplazamientos que se usarán para la codificación.
2. Se guardan los datos codificados en un registro que después se envía a la base de datos.



2

1

1. “encode” es una etiqueta que hace referencia a otra función, que únicamente recibe 2 parámetros (los enviados desde el objeto caesar en la función codificar), y como retorno manda a llamar a otra función de nombre “doStaff”, la cual tiene tres argumentos, el texto recibido, el numero de desplazamientos recibido, y un valor booleano por defecto “true”.
2. La etiqueta “decode” como veremos más adelante, hace referencia a la función de decodificación.



8

7

6

5

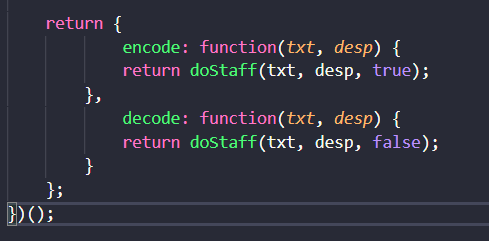
4

3

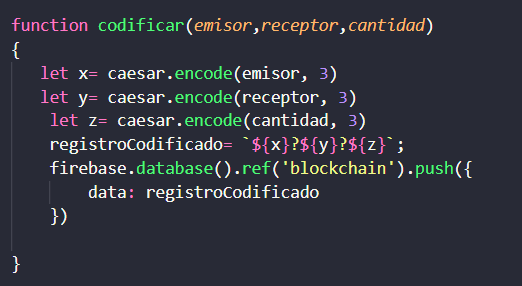
2

1

1. El objeto “caesar” utilizado en la función “codificar” es declarado previamente como una función de funciones anidadas, que se utilizaran como métodos referenciados a través de las etiquetas “encode” y “decode”.
2. La primera función, la cual es la que se manda a llamar desde la función “encode” (la cual por cierto también es un método del tipo de objeto “caesar”), recibe como parámetros los 3 argumentos enviados; texto, desplazamiento, y el valor booleano que determina si codificara o decodificara.
3. La función anidada “replace” (que es un objeto declarado con métodos) de la función “doStaff”, primero declara un arreglo con los caracteres del abecedario, y una variable con la longitud de este arreglo (los cuales se pueden editar para añadir más caracteres), y su valor de retorno es asimismo otra función.
4. La función de retorno de “replace” declara una variable “i” que será utilizada para ver si el carácter siendo evaluado es valido para la codificación, ya que, si su valor de index no se encuentra dentro de las posiciones del arreglo declarado como “abc”, entonces tomará el valor de -1 y se saltara el proceso completo debido a que es invalido, siendo retornado con su valor tal cual ingreso a la función de encriptación.
5. En caso de ser un caracter valido, se declara una variable de posición de y se evalúa en un IF-ELSE la variable booleana que entro desde la función “doStaff”, para saber si se quiere hacer una codificación o decodificación.
6. Al ser “true” el valor que estamos enviando para la codificación, se entra en la primera parte de la condición IF-ELSE. La cual toma el valor de la posición actual del carácter respecto al arreglo declarado “abc” y le agrega las posiciones declaradas en el argumento “desp” que determina la cantidad de posiciones a desplazarse hacia adelante. Y en la línea siguiente se evalúa una condición: si el valor de la variable posición es mayor o igual a 1, entonces se le resta al valor posición el valor “l” que hace referencia a la longitud del arreglo de los caracteres del abecedario y en caso contrario, se le resta un 0, todo esto a su valor actual después del desplazamiento.
7. El return dentro de está función anidada, asigna el valor actual después del algoritmo de encriptación a la variable c, y la retorna, la cual a su vez es retornada por la función “replace” hacia la función “doStaff”.
8. Esta parte de la función es la que utiliza la cadena enviada a la función “doStaff” para invocar a todo el proceso de codificación o decodificación usando la cadena “txt” recibida desde un inicio, por lo cual, la función o método “doStaff” primero declara sus métodos anidados y posteriormente usando los argumentos que recibió, invoca a dichos métodos. En esta parte del código, se toma cada nuevo carácter para formar una cadena que después será retornada por su método.



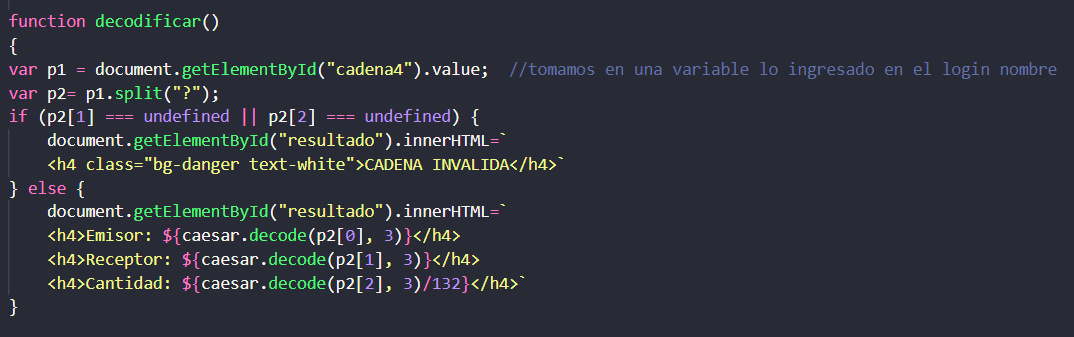
Después de todo el proceso anterior, la función de la etiqueta “encode” retorna el valor de la cadena formada por el método “doStaff”.



Y a su ves la declaración de las variables, asigna dicha cadena a su respectiva variable codificada, y el proceso se repite para la declaración de x, y & z, que representan respectivamente al Emisor, Receptor y la Cantidad, de forma codificada, para después ser enviadas de dicha manera a la base de datos.

## Decodificación

Para explicar la decodificación se toma como base todo el código anterior.



4

3

2

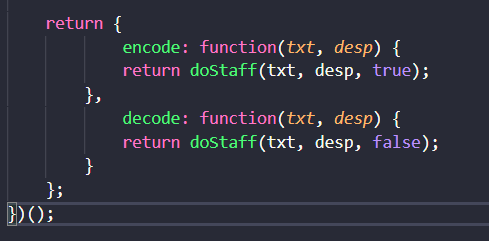
1

1.- La función utiliza las propiedades del DOM (Document Object Model), para poder obtener los elementos que se utilizaran para poder hacer referencia a los valores codificados que se quieren decodificar. En este caso obteniéndolos primero en una variable llamada “p1” por medio de su id, para después separarlos por medio de un “?” en todos sus elementos, ya que, en el algoritmo de encriptación, estos datos se guardaban en un solo registro de manera conjunta, por lo que, para decodificarlos de manera individual, se tienen que separar.

2.- La condición IF-ELSE se encarga de determinar que los valores en los campos 1 y 2 de nuestro objeto p2 sean validos para continuar con el proceso, y en caso de no serlo, se envía la advertencia de que la cadena es INVALIDA.

3.- Después de separar los elementos, se obtienen uno por uno y se envían al método “decode” que como sabemos del caso anterior, es una etiqueta que hace referencia a una función que se encargará del proceso de decodificación.

4.- En el caso del campo para “cantidad”, se tiene que tomar en cuenta que el valor se debe dividir entre 132, ya que para la codificación se usó este numero como llave de codificación.



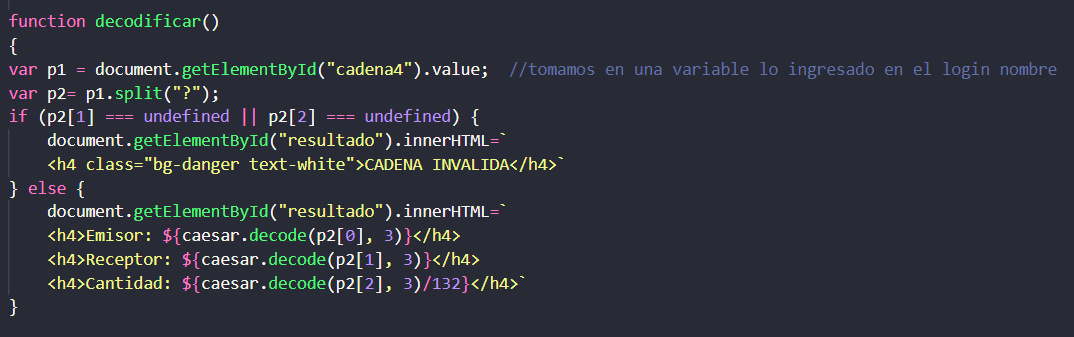
1

1.- La etiqueta “decode” llama a la función pasando los parámetros recibidos. Es importante que el valor de la variable “desp” sea mismo en la codificación y decodificiación, ya que de lo contrario, no funcionará el algoritmo.



1

1.- El procedimiento es igual que el del caso anterior, pero está vez el algoritmo es inverso. Se le resta la cantidad desplazada a la variable “pos”, y en la condicional, se le suma el valor de la variable “l” o 0 en lugar de ser restado, para que así el proceso de codificación sea totalmente inverso al de decodificación y se obtenga el mismo resultado de vuelta.

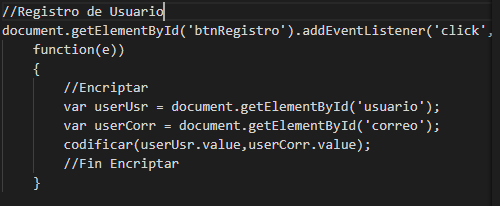


1

1.- Se asigna a los elementos <h4> en el DOM, las cadenas que se obtuvieron después de la decodificación para así mostrar los resultados. Es por eso que se encuentran referenciados hacia dicha etiqueta con el carácter “$”. Todo este proceso fue simultaneo, así que el valor asignado a las etiquetas se obtiene después de todo el proceso del método “decode”.

Para realizar la transformación de campo usuario y correo en una cuenta se debe modificar las funciones siguientes.

Registro.



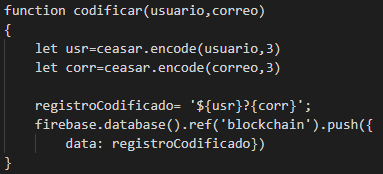
3

2

1

1. Se utiliza un “EventListener” para captar cuando se hace el uso del botón “btnRegistro”, una vez hecho esto se procede a obtener la información.
2. Se crean variables para poder almacenar los valores para la creación de la cuenta.
3. Se utiliza la función “codificar” enviando como parámetros los valores necesarios para la cuenta.

Codificar.



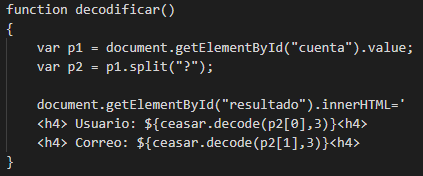
3

2

1

1. Utilizando el método encode para encriptar los parámetros recibidos en esta función se asignan a dos nuevas variables que posteriormente se enviarán.
2. Se asigna a un objeto un arreglo con dos parámetros, los cuales son las variables anteriormente creadas para los parámetros recibidos.
3. Se “llama” a la base de datos en firebase, utilizando la referencia “Blockchain” para especificar el modo de envío de información, y el método “push” para agregar el valor almacenado en el objeto.

Decodificar.



2

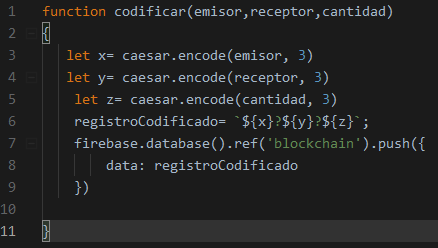
1

1. Se obtienen los elementos de la cuenta y se asignan a la variable “p1”, posteriormente en la variable, que es un arreglo, “p2” se asignan los elementos de la variable “p1”, utilizando el método Split para “separar” sus elementos.
2. Después de “separar” los elementos en el arreglo se utiliza el método decode para poder desencriptarlos.

¿Algoritmo para el numero de transacción?

La transacción nos permite encontrar tanto al emisor, receptor y la cantidad de la misma. La cadena que tenemos que insertar está formada por los datos codificados de los mismos, al entrar en la sección de “decodificador” deberemos ingresar el blockchain.

Cuando realizamos una operación, debemos indicar al emisor, receptor y por supuesto la cantidad, siguiendo con la lógica de arriba, tenemos que la codificación de los 3 elementos forma una nueva variable “registroCodificado” (líneas 3 a 5), dicha variable es la concatenación de el emisor codificado, el receptor y la cantidad. Mismo que es enviado a la base de datos y va a guardarse como ‘blockchain’ (esto se realiza en las líneas 7 y 8)

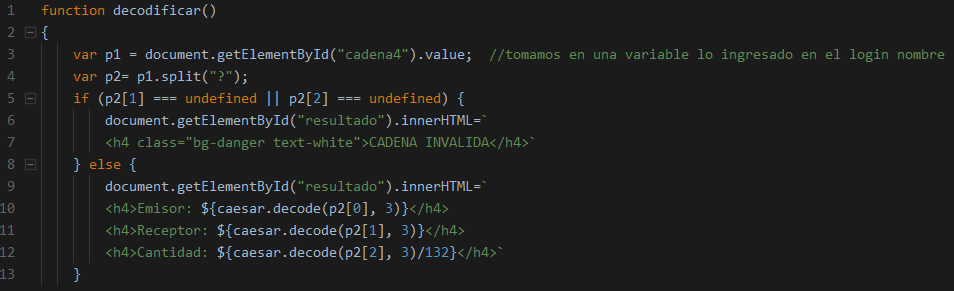


Ahora bien, tenemos una referencia a ‘blockchain’, básicamente se llena una tabla con todos los ‘blockchain’ que genera cada transacción, y podemos confirmar que cada cadena, tiene un “?” a modo de separador, mismo que servirá más tarde.



Este fragmento de código es de utilidad por que el decodificador requiere de una cadena con las características del ‘blockchain’ las cuales permiten hacer el rastreo de la transacción.

La página “Decodificador” pide una cadena, esta cadena no es más ni menos que alguna de las que podemos observar en la página “Registro de operaciones”



La “cadena4” hace referencia a la única caja de texto de la página, de aquí tomamos la cadena que previamente copiamos de “Registro de operaciones”, esta cadena, se asigna a la variable “p2” misma que se iguala a “p1.split(“?”)” lo que le indica que va a partir la cadena cada que se encuentre el carácter “?”, mismo que le dará a la variable 3 posiciones, la posición 0 corresponde al emisor, la posición 1 corresponde al receptor y la posición 2 a la cantidad.

La cláusula if evalúa que la cadena ingresada sea válida y pueda continuar, de lo contrario nos dará un mensaje que dice “CADENA INVALIDA”.

De tal manera que en la asignación de etiquetas se accede a “caesar.decode(p[n],3)” donde “n” es la “parte” (posición) que deseamos manipular, para el emisor y receptor (líneas 10 y 11) se hace lo mismo, pero para el caso de la cantidad se divide entre 132 para obtener la cantidad original.

Así pues, se puede realizar el rastreo de cada transacción y obtener los correos de usuarios pues ya sabemos cómo se encripta/des encriptan los datos en el momento que queremos leerlos.

¿Se puede saber toda la historia?

A este punto si, lo que explico anterior a esta pregunta ya nos da el emisor y receptor, al menos nos da sus ligas de transacción. Solo hace falta que las busquemos en la página “Registro” para saber a qué usuario corresponde.



Haciendo una búsqueda podemos saber quién es el emisor de manera sencilla, finalmente todo el trabajo de des encriptación se realiza cuando ingresamos el blockchain en el decodificador. La página registro, muestra que el usuario “Edward” con el correo “Master@gmail.com” fue el emisor de esa operación.

