## Tres maneras de invertir una cadena en JavaScript

Este artículo está basado en el scripting de algoritmo básico de freeCodeCamp "[Invertir una cadena](https://www.freecodecamp.org/espanol/learn/javascript-algorithms-and-data-structures/basic-algorithm-scripting/reverse-a-string)"

La inversión de una cadena es una de las preguntas de JavaScript más frecuentes en la ronda técnica de una entrevista. Los entrevistadores pueden pedirte que escribas diferentes maneras de invertir una cadena, o quizás que inviertas una cadena sin usar los métodos incorporados, o incluso que lo hagas por medio de la recursión.

Hay potencialmente decenas de maneras diferentes de hacerlo, excluyendo el método incorporado reverse, ya que JavaScript no tiene una.

A continuación están mis 3 maneras más interesantes de resolver el problema de invertir una cadena en JavaScript.

Reto del algoritmo

Invierte la cadena dada.  
Es posible que necesites convertir la cadena a un arreglo antes de que puedas invertirla.  
Tu resultado debe ser una cadena.

function invertirCadena(cad) {

return cad;

}

invertirCadena("hola");

Casos de prueba proporcionados

*invertirCadena(“hola”)* debe convertirse en “aloh”

*invertirCadena(“Hey”)* debe convertirse en “yeH”

*invertirCadena(“Saludos desde la Tierra”)* debe devolver como ”arreiT al edsed sodulaS”

1. Invertir una cadena con las funciones incorporadas

Para esta solución, usaremos tres métodos: String.prototype.split(),  Array.prototype.reverse() y Array.prototype.join().

El método split() separa un objeto de cadena en un arreglo de cadena al separar la cadena en subcadenas.

El método reverse() invierte un arreglo. El primer elemento del arreglo se convierte en el último y el último se convierte en el primero.

El método join() une todos los elementos de un arreglo en una cadena.

function invertirCadena(cad) {

// Paso 1. Usa el método split() para devolver un nuevo arreglo

var separarCadena = cad.split(""); // var separarCadena = "hola".split("");

// ["h", "o", "l", "a"]

// Paso 2. Usa el método reverse() para invertir el nuevo arreglo creado

var invertirArreglo = separarCadena.reverse(); // var invertirArreglo = ["h", "o", "l", "a"].reverse();

// ["a", "l", "o", "h"]

// Paso 3. Usa el método join() para unir todos los elementos del arreglo en una cadena

var unirArreglo = invertirArreglo.join(""); // var unirArreglo = ["a", "l", "o", "h"].join("");

// "aloh"

//Paso 4. Devolver la cadena invertida

return unirArreglo; // "aloh"

}

invertirCadena("hola");

Encadenando los tres métodos juntos:

function invertirCadena(cad) {

return cad.split("").reverse().join("");

}

invertirCadena("hola");

2. Invertir una cadena con un bucle for decremento

function invertirCadena(cad) {

// Paso 1. Crear una cadena vacía que almacenará la nueva cadena creada

var nuevaCadena = "";

// Paso 2. Crear el bucle FOR

/\* El punto de partida del bucle será (cad.length - 1) que corresponde al último caracter de la cadena, "a"

Mientras i sea mayor que o igual a 0, el bucle se ejecutará

Decrementamos i después de cada iteración \*/

for (var i = cad.length - 1; i >= 0; i--) {

nuevaCadena += cad[i]; // o nuevaCadena = nuevaCadena + cad[i];

}

/\* Aquí la longitud de hola es igual a 4

Por cada iteración: i = cad.length - 1 y nuevaCadena = nuevaCadena + cad[i]

Primera iteración: i = 4 - 1 = 3, nuevaCadena = "" + "a" = "a"

Segunda iteración: i = 3 - 1 = 2, nuevaCadena = "a" + "l" = "al"

Tercera iteración: i = 2 - 1 = 1, nuevaCadena = "al" + "o" = "alo"

Cuarta iteración: i = 1 - 1 = 0, nuevaCadena = "alo" + "h" = "aloh"

Fin del bucle FOR\*/

// Paso 3. Devolver la cadena invertida

return nuevaCadena; // "aloh"

}

invertirCadena('hola');

Sin los comentarios:

function invertirCadena(cad) {

var nuevaCadena = "";

for (var i = cad.length - 1; i >= 0; i--) {

nuevaCadena += cad[i];

}

return nuevaCadena;

}

invertirCadena('hola');

3. Invertir una Cadena con recursión

Para esta solución, usaremos dos métodos:  String.prototype.substr() y  String.prototype.charAt().

El método substr() devuelve los caracteres de una cadena que comienza en la ubicación especificada a través del número especificado de caracteres.

"hola".substr(1); // "ola"

El método charAt() devuelve el carácter especificado de una cadena.

"hola".charAt(0); // "h"

La profundidad de la recursión es igual a la longitud de la cadena. Esta solución no es la mejor y será realmente lenta si la cadena es muy larga y el tamaño de la pila es de gran preocupación.

function invertirCadena(cad) {

if (cad === "") // Este es el caso terminal que finalizará la recursión

return "";

else

return invertirCadena(cad.substr(1)) + cad.charAt(0);

/\*

Primera parte del método de recursión

Necesitas recordar que no tendrás solo una llamada a la recursión; tendrás varias llamadas anidadas

Cada llamada: cad === "?" invertirCadena(cad.subst(1)) + cad.charAt(0)

1ra. llamada – invertirCadena("Hola") devolverá invertirCadena("ola") + "h"

2da. llamada – invertirCadena("ola") devolverá invertirCadena("la") + "o"

3ra. llamada – invertirCadena("la") devolverá invertirCadena("a") + "l"

4ta. llamada – invertirCadena("a") devolverá invertirCadena("") + "a"

Segunda parte del método de recursión

El método cumple la condición del if y la llamada más anidada es devuelta inmediatamente

4ta. llamada devolverá invertirCadena("") + "a" = "a"

3ra. llamada devolverá invertirCadena("a") + "l" = "a" + "l"

2da. llamada devolverá invertirCadena("la") + "o" = "a" + "l" + "o"

1ra. llamada devolverá invertirCadena("ola") + "h" = "a" + "l" + "o" + "h"

\*/

}

invertirCadena("hola");

Sin los comentarios:

function invertirCadena(cad) {

if (cad === "")

return "";

else

return invertirCadena(cad.substr(1)) + cad.charAt(0);

}

invertirCadena("hola");

Operador Condicional (ternario):

function invertirCadena(cad) {

return (cad === '') ? '' : invertirCadena(cad.substr(1)) + cad.charAt(0);

}

invertirCadena("hola");

Invertir una cadena en JavaScript es un algoritmo pequeño y simple que puede ser preguntado en una selección técnica telefónica o una entrevista técnica. Puedes tomar la ruta corta para resolver este problema, o tomar el enfoque de resolverlo con recursión o incluso con soluciones más complejas.

Espero que hayas encontrado esto útil.