Mongo

Express

Angular

Node

PHP

JavaScript

HTML5

CSS3

Sublime Text

Bootstrap

Grunt

Vagrant

Docker

PATROCINADORES

infranetworking

Web Hosting Profesional

В



Esta publicación tiene 14 meses de antigüedad, te invitamos a buscar publicaciones más

Sales CRM for small teams

COMPARTIR EN FACEBOOK

COMPARTIR EN TWITTER

recientes relacionados a este tema aquí.

S+ COMPARTIR EN GOOGLE+

Registro

Login

Algoritmos en Javascript, en esta primera entrega hablaremos sobre las pilas. pipedrive

Hola chicos después de un tiempo sin escribir regreso para publicar una serie de tutoriales sobre Estructura de Datos y

with big ambitions SIGN UP NOW

Una pila es una colección de datos que sigue el principio de LIFO (Last In First Out - Último en entrar primero en salir). La eliminación o adición de nuevos elementos siempre se agregarán al inicio, quiere decir que los elementos más viejos estarán siempre en la base, ¿te parece confuso? Imaginate una pila de libros:

lo colocas en la parte superior de la pila de libros y de igual manera los que borras los quitas desde arriba también. Creando una pila en JavaScript

Quiere decir que si quisieras sacar el libro que esta en la base primero tendrías que quitar uno a uno los libros hasta llegar a él (ya se que puedes tirarlos de un golpe y tomarlo, pero imaginemos que es 1 a 1). Entonces cualquier libro que agregues

• add(element(s)): Con este método agregaremos uno o más elementos a nuestra pila (en la parte superior).

• pop(): Este método quitará de la parte superior el último elemento y también lo regresará. • getTopElement(): Este método solamente nos regresará el elemento que se encuentre en la parte superior sin

Antes de crear nuestra pila en Javascript primero definiremos que métodos vamos a implementar:

- modificarlo ni removerlo. • hasElements(): Retornará true si nuestra pila tiene al menos un elemento, o en su defecto false si esta vacía.
- removeAll(): Limpia la pila quitando todos los elementos. size(): Regresará el tamaño de la pila (cuantos elementos hay en ella).
- De manera que nuestro código será el siguiente (puedes guardarlo como stack.js):
- function Stack() {
- var elements = [];

```
this.add = add;
     this.pop = pop;
     this.getTopElement = getTopElement;
     this.hasElements = hasElements;
     this.removeAll = removeAll;
     this.size = size;
     function add(element) {
         elements.push(element);
     function pop() {
         return elements.pop();
     function getTopElement() {
         return elements[elements.length - 1];
     function hasElements() {
         return elements.length > 0;
     function removeAll() {
         elements = [];
     function size() {
         return elements.length;
Para probar que funciona podemos escribir lo siguiente (en el mismo archivo):
```

fruitsStack.add('Apple'); fruitsStack.add('Mango'); console.log(fruitsStack.size()); // Mostrará 3

var fruitsStack = new Stack(); if (!fruitsStack.hasElements()) { fruitsStack.add('Banana');

var res; // Resto o residuo.

var str = ''; // Será la cadena que regresaremos

navegador.

```
var mango = fruitsStack.pop(); //Obtiene mango y lo saca de la pila, ahora solo quedan 2 elementos.
     console.log(mango);
     console.log(fruitsStack.getTopElement()); // Imprimirá Apple, puesto que es el elemento que quedo hasta arr
     console.log(fruitsStack.size()); // Mostrará 2
     fruitsStack.removeAll(); // Limpia la pila
     console.log(fruitsStack.hasElements()); // Retornará falso puesto que la pila esta vacía
Para correrlo simplemente escribe en tu consola (si tienes Node.js instalado):
 node stack.js
O bien agregalo a un HTML dentro de una etiqueta script o si quieres algo más rápido puedes correrlo en la consola del
```

Ejemplo práctico de una pila Si buscas algún otro ejemplo práctico para pilas, podemos realizar un convertidor de decimal a base X (binario, octal &

o en otro archivo siempre y cuando incluyamos primero el stack.js. function convert(number, base) { var resStack = new Stack(); // Creamos nuestro objecto de pila para guardar los REStos (o residuos).

hexadecimal). Tomando de base la función Stack que creamos anteriormente, podemos escribir el siguiente código debajo

var characters = '0123456789ABCDEF'; // Lista de caracteres que usaremos para la conversión a binario, octa

```
// Mientras que el numero dado sea mayor a 0...
    while (number > 0) {
        res = Math.floor(number % base); // Sacamos el módulo del número (su resto o residuo).
        resStack.add(res); // Agregamos el resto a la pila
        number = Math.floor(number / base); //Dividimos entre la base el numero y así continuamos hasta llegar
    // Mientras la pila tenga elementos...
    while (resStack.hasElements()) {
         //Concatenamos cada numero o letra que salga del array de caracteres y en cada vuelta se saca un elemen
        str += characters[resStack.pop()];
    // Retornamos la cadena final.
 console.log(convert(16777215, 16)); // Regresará FFFFFF
 console.log(convert(100, 8)); // Regresará 144
 console.log(convert(4, 2)); // Regresará 100
Espero les haya sido de utilidad, en la próxima publicación hablaré sobre las colas, saludos!
                  COMPARTIR EN FACEBOOK
                                           Y COMPARTIR EN TWITTER
                                                                    S+ COMPARTIR EN GOOGLE+
                   ¿Te gustó esta publicación? Márcala como favorita 🖈
```

Carlos Santana Roldán Codejobs Estados Unidos



https://www.codejobs.biz Publicaciones del autor

Estructura de Datos: ¿Qué son los...

Sales CRM for small teams

with big ambitions

Instalar el tema Seti_Ul para Sublime Text 3 - Aprende a

Agrega Disqus a tu sitio

1 Comment

Recomendar

TAMBIÉN EN CODEJOBS

Subscribe





Únete a la discusión... mauricio - hace 10 meses Muchas gracias!:) ∧ V - Reply - Share >

Handlebars, Stylus, Gulp y más: Gulp #3 - Aprende a ... Programar 2 comments - hace un año-1 comment • hace un año-Jose Gelimer Gomez — Porque no pones TODO el codigo de webmedia digital — thanks nice

¿Que es un objeto en JavaScript? - Aprende a Programar Recursion en JavaScript - Aprende a Programar 1 comment • hace 9 meses• 2 comments • hace 5 meses • Ori Yanelli - Muchas gracias por el aporte! Nico — Lo que pasa con la recursión es que va dejando tareas pendientes en memoria, en el ejemplo que dieron power(2,3), el

Tecnologías Codejobs es una comunidad de amantes a la

Privacidad



tecnología que creemos que el conocimiento es la única forma de ser verdaderamente libres y autónomos.



DISQUS

🚺 Login 🤻

Estructura de Datos y Algoritmos co...

Curso: Algoritmos y Estructura de D...

pipedrive

SIGN UP NOW

Desarrollando un CMS desde cero con Node.js Express,

resultado parcial sería base*power(2,2), pero habría que hallar el ...

CA IS IT

5 5 B 🛱 🕡

© 2017 Codejobs