Arrays - Arreglos o Matrices

¿Qué es una matriz?

Las matrices se describen como "objetos tipo lista"; básicamente son objetos individuales que contienen múltiples valores almacenados en una lista. Los objetos de matriz pueden almacenarse en variables y tratarse de la misma manera que cualquier otro tipo de valor, la diferencia es que podemos acceder individualmente a cada valor dentro de la lista y hacer cosas útiles y eficientes con la lista, como recorrerlo con un bucle y hacer una misma cosa a cada valor.

Tal vez tenemos una serie de productos y sus precios almacenados en una matriz, y queremos recorrerlos e imprimirlos en una factura, sumando todos los precios e imprimiendo el total en la parte inferior.

Si no tuviéramos matrices, tendríamos que almacenar cada elemento en una variable separada, luego llamar al código que hace la impresión y agregarlo por separado para cada artículo. Esto sería mucho más largo de escribir, menos eficiente y más propenso a errores. Si tuviéramos 10 elementos para agregar a la factura, ya sería suficientemente malo, pero ¿ qué pasa con 100 o 1000 artículos? Volveremos a este ejemplo más adelante.

Creando una matriz

Las matrices se construyen con corchetes, que contiene una lista de elementos separados por comas.

1. Digamos que queríamos almacenar una lista de compras en una matriz — haríamos algo como lo siguiente. Ingresa las siguientes líneas en la consola:

```
let shopping = ['bread', 'milk', 'cheese', 'hummus', 'noodles'];
shopping;
```

   En este caso, cada elemento de la matriz es una cadena, pero ten en cuenta que puedes almacenar cualquier elemento en una matriz — cadena, número, objeto, otra variable, incluso otra matriz. También puedes mezclar y combinar tipos de elementos — no todos tienen que ser números, cadenas, etc.
   Prueba estos:

```
let sequence = [1, 1, 2, 3, 5, 8, 13];
let random = ['tree', 795, [0, 1, 2]];
```

2. Intenta crear un par de matrices por tu cuenta, antes de continuar.

Accediendo y modificando elementos de la matriz

Puedes entonces acceder a elementos individuales en la matriz mediante la notación de corchetes, del mismo modo que accediste a las letras de una cadena.

1. Ingresa lo siguiente en tu consola:

```
shopping[0];
// returns "bread"
```

También puedes modificar un elemento en una matriz simplemente dando a un ítem de la matriz un nuevo valor. Prueba esto:

```
shopping[0] = 'tahini';
shopping;
// shopping will now return [ "tahini", "milk", "cheese", "hummus", "noodles" ]
```

Ten en cuenta que una matriz dentro de otra matriz se llama matriz multidimensional. Puedes acceder a los elementos de una matriz que estén dentro de otra, encadenando dos pares de corchetes.
Por ejemplo, para acceder a uno de los elementos dentro de la matriz, que a su vez, es el tercer elemento dentro de la matriz random, podríamos hacer algo como esto:

```
random[2][2];
```

3. Intenta seguir jugando y haciendo algunas modificaciones más a tus ejemplos de matriz antes de continuar.

Encontrar la longitud de una matriz

Puedes averiguar la longitud de una matriz (cuántos elementos contiene) exactamente de la misma manera que determinar la longitud (en caracteres) de una cadena— utilizando la propiedad length. Prueba lo siguiente:

```
sequence.length;
// should return 7
```

Esto tiene otros usos, pero se usa más comúnmente para indicarle a un ciclo que continúe hasta que haya recorrido todos los elementos de la matriz. Así por ejemplo:

```
let sequence = [1, 1, 2, 3, 5, 8, 13];
for (var i = 0; i < sequence.length; i++) {
```

```
  console.log(sequence[i]);
}
```

Aprenderás acerca de bucles correctamente en un artículo futuro, pero brevemente, éste código dice:

1. Comienza el bucle en el elemento de la posición 0 en la matriz.
2. Detén el bucle en el número de ítem igual a la longitud de la matriz. Esto funcionará para una matriz de cualquier longitud, pero en este caso el ciclo se detendrá en el elemento número 7 (esto es bueno, ya que el último elemento — que queremos que recorra el bucle — es 6).
3. Para cada elemento, imprime en la consola del navegador con `console.log()`.

Alguno métodos de matriz útiles

En esta sección veremos algunos métodos bastante útiles relacionados con matrices que nos permiten dividir cadenas en elementos de matriz y viceversa, y agregar nuevos elementos en matrices.

Conversión entre matrices y cadenas

A menudo se te presentarán algunos datos brutos contenidos en una cadena larga y grande, y es posible que desees separar los elementos útiles de una forma más conveniente y luego hacerle cosas, como mostrarlos en una tabla de datos.

Para hacer esto, podemos usar el método `split()`. En su forma más simple, esto toma un único parámetro, el caracter que quieres separar de la cadena, y devuelve las subcadenas entre el separador como elementos en una matriz.

1. Vamos a jugar con esto, para ver cómo funciona. Primero, crea una cadena en tu consola:

```
let myData = 'Manchester,London,Liverpool,Birmingham,Leeds,Carlisle';
```

Ahora dividámoslo en cada coma:

```
let myArray = myData.split(',');
myArray;
```

Finalmente, intenta encontrar la longitud de tu nueva matriz y recuperar algunos elementos de ella:

```
myArray.length;
myArray[0]; // the first item in the array
```

```
myArray[1]; // the second item in the array
myArray[myArray.length-1]; // the last item in the array
```

También puedes ir en la dirección opuesta usando el método join(). Prueba lo siguiente:

```
let myNewString = myArray.join(',');
myNewString;
```

Otra forma de convertir una matriz en cadena es usar el método toString(). Es posiblemente más simple que join() ya que no toma un parámetro, pero es más limitado. Con join() puedes especificar diferentes separadores (intenta ejecutar con un caracter diferente a la coma).

```
let dogNames = ['Rocket','Flash','Bella','Slugger'];
dogNames.toString(); //Rocket,Flash,Bella,Slugger
```

Agregar y eliminar elementos de la matriz

Todavía no hemos cubierto la posibilidad de agregar y eliminar elementos de la matriz — echemos un vistazo a esto ahora. Usaremos la matriz myArray:

```
let myArray = ['Manchester', 'London', 'Liverpool', 'Birmingham', 'Leeds', 'Carlisle'];
```

Antes que nada, para añadir o eliminar un elemento al final de una matriz podemos usar push() y pop() respectivamente.

1. primero usemos push() — nota que necesitas incluir uno o más elementos que desees agregar al final de tu matriz. Prueba esto:

   ```
   myArray.push('Cardiff');
   myArray;
   myArray.push('Bradford', 'Brighton');
   myArray;
   ```

   La nueva longitud de la matriz se devuelve cuando finaliza la llamada al método. Si quisieras almacenar la nueva longitud de matriz en una variable, podrías hacer algo como esto:

   ```
   let newLength = myArray.push('Bristol');
   myArray;
   newLength;
   ```

Eliminar el último elemento de una matriz es tan simple como ejecutar `pop()` en ella. Prueba esto:

```
myArray.pop();
```

El elemento que sé eliminó se devuelve cuando se completa la llamada al método. Para guardar este elemento en una variable, puedes hacer lo siguiente:

```
let removedItem = myArray.pop();
myArray;
removedItem;
```

`unshift()` y `shift()` funcionan exactamente igual de `push()` y `pop()`, respectivamente, excepto que funcionan al principio de la matriz, no al final.

Primero `unshift()` — prueba el siguiente comando:

```
myArray.unshift('Edinburgh');
myArray;
```

Ahora `shift()`; ¡prueba estos!

```
let removedItem = myArray.shift();
myArray;
removedItem;
```

Aprendizaje activo: ¡Imprimiendo esos productos!

Volvamos al ejemplo que describimos anteriormente — imprima los nombres de los productos y los precios en una factura, luego, sume los precios e imprímelos en la parte inferior. En el ejemplo editable a continuación, hay comentarios que contienen números — cada uno de estos marca un lugar donde debe agregar algo al código. Ellos son los siguientes:

1. Debajo de `// number 1` hay un número de cadena, cada una de las cuales contiene un nombre de producto y un precio separados por dos puntos. Nos

gustaría que convirtieras esto en una matriz y lo almacenamos en una matriz llamada `products`.

2. En la misma línea que el comentario `// number 2` es el comienzo de un ciclo `for`. En esta línea, actualmente tenemos `i <= 0`, que es una prueba condicional que hace que el bucle for se detenga inmediatamente, porque dice "detener cuando `i` es menor o igual 0", y `i` comienza en 0. Nos gustaría que reemplazaras esto con una prueba condicional que detenga el ciclo cuando `i` no sea inferior a la longitud la matriz `products` .
3. justo debajo del comentario `// number 3` queremos que escriba una línea de código que divide el elemento actual de la matriz `(nombre:precio)` en dos elementos separados, uno que contiene solo el nombre y otros que contienen solo el precio.
4. Como parte de la línea de código anterior, también querrás convertir el precio de una cadena a un número.
5. Hay una variable llamada `total` que se crea y se le da un valor de 0 en la parte superior del código. Dentro del ciclo (debajo de `// number 4`) queremos que agregues una línea que añade el precio actual del artículo a ese total en cada iteración del ciclo, de modo que al final del código el total correcto se imprima en la factura. Es posible que necesites un **operador de asignación** para hacer esto.
6. Queremos que cambies la línea justo debajo `// number 5` para que la variable `itemText` se iguale a "nombre de elemento actual — $precio de elemento actual", por ejemplo "Zapatos — $23.99" en cada caso, por lo que la información correcta del artículo está impreso en la factura. Esto es simplemente una concatenación de cadenas, lo que debería ser familiar para ti.

Aprendizaje Activo: Top 5 búsquedas

Un buen uso para los métodos de matriz como `push()` y `pop()` es cuando estás manteniendo un registro de elementos actualmente activos en una aplicación web.

En una escena animada por ejemplo, es posible que tengas una matriz de objetos que representan los gráficos de fondo que se muestran actualmente, y es posible que sólo desees que se muestren 50 a la vez, por razones de rendimiento o desorden. A medida que se crean y agregan nuevos objetos a la matriz, se puede eliminar los más antiguos de la matriz para mantener el número deseado.

En este ejemplo vamos a mostrar un uso mucho más simple — aquí te daremos un sitio de búsqueda falso, con un cuadro de búsqueda. La idea es que cuando los términos se ingresan en un cuadro de búsqueda, se muestran el top 5 de términos de búsqueda previos en la lista. Cuando el número de términos supera el 5, el último término comienza a borrarse cada vez que agregas un nuevo término a la parte superior, por lo que siempre se muestran los 5 términos anteriores.

Nota: En una aplicación de búsqueda real, probablemente puedas hacer clic en los términos de búsqueda anteriores para volver a los términos de búsqueda anteriores y

¡se mostrarán los resultados de búsqueda reales! Solamente lo mantendremos simple por ahora.

Para completar la aplicación necesitamos:

1. Agregar una línea debajo del comentario `// number 1` que agrega el valor actual ingresado en la entrada de la búsqueda al inicio de la matriz. Esto se puede recuperar usando `searchInput.value`.
2. Agrega una línea debajo del comentario `// number 2` que elimina el valor actualmente al final de la matriz.

Conclusión

Después de leer este artículo, estamos seguros de que estarás de acuerdo en que las matrices parecen bastante útiles; las verás aparecer en todas partes en JavaScript, a menudo en asociación con bucles para hacer una misma cosa con cada elemento de la matriz.