***ARREGLOS EN JAVASCRIPT***

**Los arreglos tipados en JavaScript** son objetos similares a arreglos que proporcionan un mecanismo para leer y escribir datos binarios sin procesar en búferes de memoria. Como ya sabrás, los objetos [Arreglo](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array) crecen y se encogen dinámicamente y pueden tener cualquier valor de JavaScript. Los motores de JavaScript realizan optimizaciones para que estos arreglos sean rápidos.

Los **arrays** son objetos similares a una lista cuyo prototipo proporciona métodos para efectuar operaciones de recorrido y de mutación. Tanto la longitud como el tipo de los elementos de un **array** son variables

En JavaScript, los arreglos pueden ser una colección de elementos de cualquier tipo. Esto significa que tú puedes crear un arreglo  con elementos de tipo Cadena , Boolean, Número, Objetos, e incluso otros Arreglos.

Aquí hay un ejemplo de un arreglo con otros cuatro elementos: tipo Número, Boolean, Cadena  y Objecto.

const mixedTypedArray = [100, true, 'freeCodeCamp', {}];

La posición de un elemento en el arreglo es conocido como indice. En JavaScript, el indice del arreglo empieza con 0, e incrementa uno a uno con cada elemento.

Entonces, por ejemplo, en el arreglo de arriba, el elemento 100 es en indice 0, cierto si está en indice 1, 'freeCodeCamp' está en indice 2, y así.

El número de elementos en el arreglo determina su longitud. Por ejemplo, la longitud del arreglo de arriba es cuatro.

Curiosamente, los arreglos de JavaScript no tienen longitud fija. Tú puedes cambiar la longitud en cualquier momento asignando un valor numérico positivo.

crear un arreglo de diferentes formas en JavaScript. La forma más sencilla es asignar un valor de arreglo a una variable.

const salad = ['🍅', '🍄', '🥦', '🥒', '🌽', '🥕', '🥑'];

También puedes usar el constructor de Arreglo para crear un arreglo.

const salad = new Array('🍅', '🍄', '🥦', '🥒', '🌽', '🥕', '🥑');

Tenga en cuenta: new Array(2) creará un arreglo de longitud 2 y ninguno de los elementos son definidos en él. Sin embargo, new Array(1,2) creará un arreglo de longitud dos con elementos 1 y 2 en él.

Hay otros métodos como Array.of() y Array.from(), y el spread operador(...) que te ayuda a crear arreglos, también.

acceder y traer elementos de un arreglo usando su indice. Tú necesitas usar la sintaxis square bracket para acceder a los elementos del arreglo.

const element = array[index];

Según sus casos de uso, tú debes escoger acceder a los elementos del arreglo uno por uno o en un bucle.

Cuando accedes a elementos usando un indice como este:

const salad = ['🍅', '🍄', '🥦', '🥒', '🌽', '🥕', '🥑'];

salad[0]; // '🍅'

salad[2]; // '🥦'

salad[5]; // '🥕'

Puedes usar la longitud de un arreglo para retroceder y acceder elementos.

const salad = ['🍅', '🍄', '🥦', '🥒', '🌽', '🥕', '🥑'];

const len = salad.length;

salad[len - 1]; // '🥑'

salad[len - 3]; // '🌽'

También puedes iterar a través del arreglo usando el común bucle for o forEach, o cualquier otro bucle.

const salad = ['🍅', '🍄', '🥦', '🥒', '🌽', '🥕', '🥑'];

for(let i=0; i<salad.length; i++) {

console.log(`Element at index ${i} is ${salad[i]}`);

}

Y aquí esta el resultado:

image-30

Como añadir elementos al arreglo en JS

Usa el método push() para añadir un elemento en el arreglo. El método push() añade un elemento al final del arreglo. Ve como añadimos algunos cacahuetes a la ensalada, como esto:

const salad = ['🍅', '🍄', '🥦', '🥒', '🌽', '🥕', '🥑'];

salad.push('🥜');

Ahora el arreglo salad es:

["🍅", "🍄", "🥦", "🥒", "🌽", "🥕", "🥑", "🥜"]

Nota que el método push() añade un elemento al final del arreglo. Si tu quieres añadir un elemento al inicio del arreglo, vas a necesitar usar el método unshift().

const salad = ['🍅', '🍄', '🥦', '🥒', '🌽', '🥕', '🥑'];

salad.unshift('🥜');

Ahora el arreglo salad es:

["🥜", "🍅", "🍄", "🥦", "🥒", "🌽", "🥕", "🥑"]

La manera más sencilla de eliminar un solo elemento de un arreglo usando el método pop() . Cada vez que llamas el método pop(), este elimina un elemento del final de un arreglo. Entonces este regresa con el elemento eliminado y cambia el arreglo original.

const salad = ['🍅', '🍄', '🥦', '🥒', '🌽', '🥕', '🥑'];

salad.pop(); // 🥑

console.log(salad); // ['🍅', '🍄', '🥦', '🥒', '🌽', '🥕']

Usa el método shift() para eliminar un elemento desde el principio del arreglo. Como el método pop(), shift() regresa el elemento eliminado y cambia el arreglo original.

const ensalada = ['🍅', '🍄', '🥦', '🥒', '🌽', '🥕', '🥑'];

salad.shift(); // 🍅

console.log(ensalada); // ['🍄', '🥦', '🥒', '🌽', '🥕', '🥑'];