

arrays

En programación, un **arreglo** es una colección de elementos o cosas. Los arreglos **guardan** data como elementos y **los regresan** cuando los necesitas.

La estructura de datos de arreglo es ampliamente usada en todos los lenguajes de programación que la soportan.

Que es un arreglo en JavaScript?

Un par de square brackets [] representa un arreglo en JavaScript. Todos los elementos en un arreglo están separados por una comma(,).

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 Objeto.

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

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, 'Anshula' 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. Aprenderemos más acerca de esto dentro de poco.

Como crear un arreglo en JavaScript

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

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

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

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

Segun sus casos de uso, tu 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]; // '🌽'
```

Tambien puedes iterar a traves del arreglo usando el comun 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 el resultado seria

```
Element at index 0 is 🍅
Element at index 1 is 🍄
Element at index 2 is 🥦
Element at index 3 is 🥒
Element at index 4 is 🌽
Element at index 5 is 🥕
Element at index 6 is 🥑
```

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:

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

Como eliminar elementos de un arreglo en JS

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(); // 🥑
```

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(); // 🥑
```

Ahora el arreglo **salad** es:

```
["🍅", "🍄", "🥦", "🥒", "🌽", "🥕", ""]
```