

Arrays

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Arreglos o Matrices (Arrays)

Los Arreglos “**son una manera ordenada**” de almacenar una lista de elementos de datos bajo un solo nombre de variable, pudiendo acceder a cada elemento individual de la lista.



Creación de un Arreglo

Un arreglo se representa con corchetes [], dentro se coloca el contenido. Cada elemento es separado por coma.

```
> var verduras = []; // Arreglo Vacio
```

```
> var frutas = ['Pera', 'Manzana', 'Platano', 'Naranja'];
```

Los elementos incluso pueden ser de diferente tipo:

```
> var miArreglo = ['Soy un String', 3, true, 'Hola', 5.66, false];
```

Acceder a los valores de un Arreglo

Podemos acceder a cada contenido individual indicando la posición numérica del elemento que queremos acceder entre corchetes `[]` (esto se llama **índice** o **index**). Importante: La primera posición es 0.

```
> var frutas = ['Pera', 'Manzana', 'Platano', 'Naranja'];
```

```
> frutas
```

```
< ▶ (4) ["Pera", "Manzana", "Platano", "Naranja"]
```

Posición	0	1	2	3
----------	---	---	---	---

```
> frutas[2]
```

```
< "Platano"
```

Modificar un valor de un Arreglo

Podemos modificar el valor de un elemento individual asignando un nuevo valor a una posición determinada del arreglo, indicada entre corchetes [].

```
> var frutas = ['Pera', 'Manzana', 'Platano', 'Naranja'];
```

Posición	0	1	2	3
----------	---	---	---	---

```
> frutas[3] = "Uvas";
```

```
< "Uvas"
```

```
> frutas
```

```
< ▶ (4) ["Pera", "Manzana", "Platano", "Uvas"]
```

Actividad

Ejercicios Arreglos

Challenge:

Sigamos creando nuestra calculadora de propinas. Donde se da el 15% de propina siempre y cuando la cuenta esté entre los \$100 y \$800. Si la cuenta es diferente a ese rango la propina será del 20%.

1. Escribe una función `calculateTip` que tome como argumento el valor de la cuenta y regrese la correspondiente propina.
2. Crea un arreglo `bills` que contenga las siguientes cuentas: \$75, \$280 y \$1350 (esta es del brunch en el Hilton)
3. Crea un arreglo `tips` que contenga las propinas para cada una de las cuentas anteriores usando la función que creaste en (1.)
4. **Bonus:** Crea un arreglo `total` que contenga los totales a pagar (cuenta + propina).
5. Imprime en consola bills, tips y total.

Métodos de Arrays

DEV.F
DESARROLLAMOS(PERSONAS);

dev

length

La propiedad **length** nos devuelve el número total de elementos en el arreglo.

Este método es indispensable para poder iterar (recorrer) el arreglo y hacer operaciones con dichos elementos (*se verá más adelante*).

```
> var frutas = ['Pera', 'Manzana', 'Platano', 'Uvas'];
```

```
> frutas.length;
```

```
< 4
```


indexOf

indexOf regresa el índice del elemento que estamos buscando, si no lo encuentra retorna -1



```
1  const friends = ['Rachel', 'Monica', 'Joey', 'Chandler', 'Phoebe', 'Ross'];
2  indexOfJoey = friends.indexOf('Joey');
3  console.log(indexOfJoey);
4  // → 2
5
6  indexOfRichard = friends.indexOf('Richard');
7  console.log(indexOfRichard);
8  // → -1
```

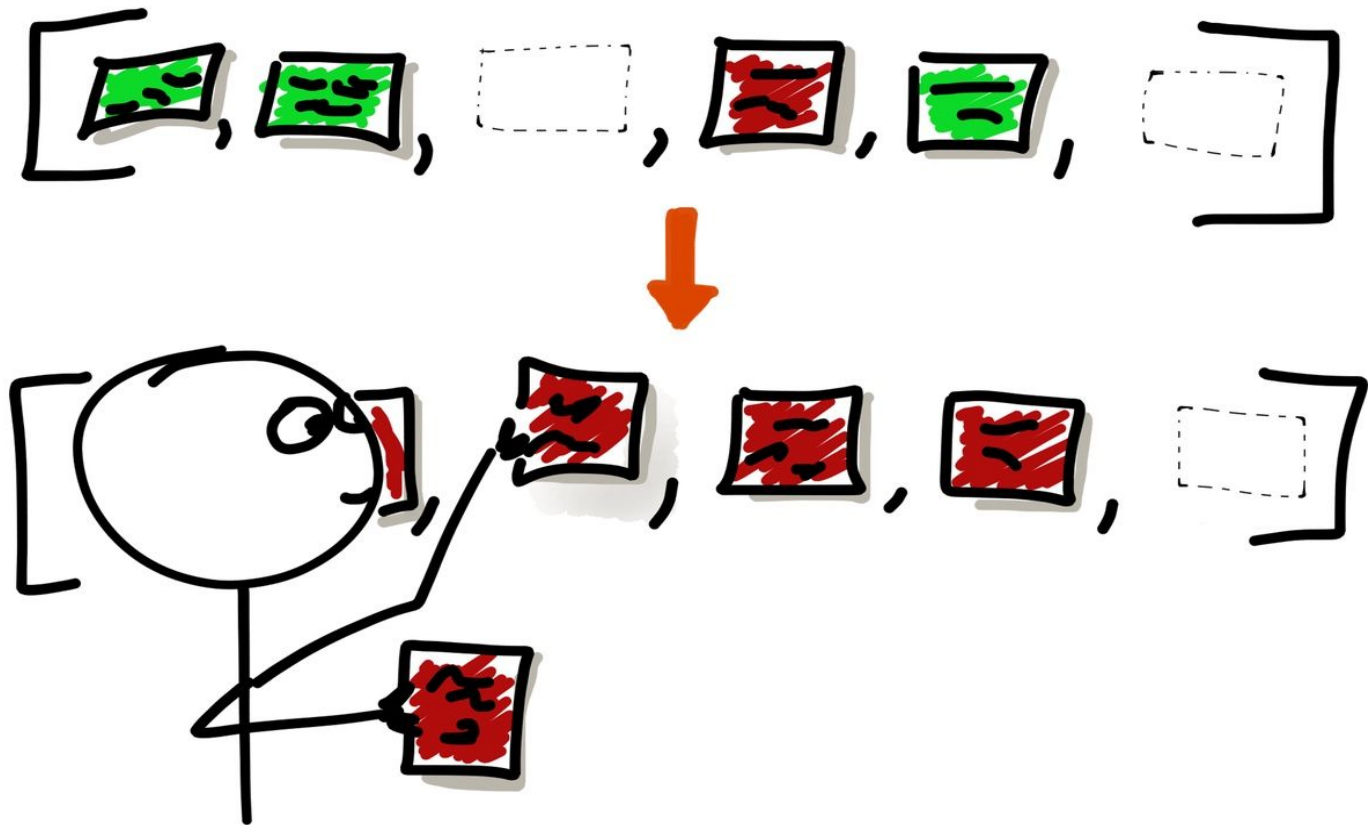
includes

includes regresa **true** si el arreglo contiene el valor que se le pasa como argumento en una de sus posiciones y si no regresa **false**



```
1  const friends = ['Rachel', 'Monica', 'Joey', 'Chandler', 'Phoebe', 'Ross'];
2  includesJoey = friends.includes('Joey');
3  console.log(includesJoey);
4  // → true
5
6  includesRichard = friends.includes('Richard');
7  console.log(includesRichard);
8  // → false
```

Métodos de Arreglos (Arrays)



push y unshift

El método **push** agrega un ítem al final de la lista.

```
const friends = ['Rachel', 'Monica', 'Joey', 'Chandler', 'Phoebe', 'Ross'];  
friends.push('Gunther');  
console.log(friends);  
// → ['Rachel', 'Monica', 'Joey', 'Chandler', 'Phoebe', 'Ross', 'Gunther']
```

El método **unshift** agrega un ítem al principio de la lista.

```
const friends = ['Rachel', 'Monica', 'Joey', 'Chandler', 'Phoebe', 'Ross'];  
friends.unshift('Gunther');  
console.log(friends);  
// → ['Gunther', 'Rachel', 'Monica', 'Joey', 'Chandler', 'Phoebe', 'Ross']
```

Ambos son funciones y regresan la longitud del nuevo arreglo

pop y shift

El método **pop** elimina el ítem que está **al final de la lista**.

```
const friends = ['Rachel', 'Monica', 'Joey', 'Chandler', 'Phoebe', 'Ross'];  
friends.pop();  
console.log(friends);  
// → ['Rachel', 'Monica', 'Joey', 'Chandler', 'Phoebe']
```

El método **shift** elimina el ítem que está **al principio de la lista**.

```
const friends = ['Rachel', 'Monica', 'Joey', 'Chandler', 'Phoebe', 'Ross'];  
friends.shift();  
console.log(friends);  
// → ['Monica', 'Joey', 'Chandler', 'Phoebe', 'Ross']
```

Ambos son funciones y regresan el valor del elemento eliminado

split

Divide una cadena (string) en una matriz de subcadenas, tomando como referencia donde encuentre un carácter indicado.

```
> var verduras = "Cebolla,Perejil,Tomate,Calabaza";
```

```
> var arregloVerduras = verduras.split(',');
```

```
> arregloVerduras;
```

```
◀ ▶ (4) ["Cebolla", "Perejil", "Tomate", "Calabaza"]
```

Slice (porción)

Quita una parte de una cadena y **devuelve una NUEVA cadena**.

El nuevo array empieza en el índice indicado por el parámetro start.

Al usar el parámetro end, el nuevo array tendrá como último elemento el elemento con índice (end-1).

Retorna el valor de los elementos eliminados.



```
1 array.slice(start, end)
```

Slice (porción)

```
1  var frutas = ['pera', 'manzana', 'plátano', 'uvas', 'mandarina'];
2  // Index      0      1      "2"      3      4
3
4  frutas.slice(2);
5  //regresa en consola [ "plátano", "uvas", "mandarina" ]
6  //es decir elimina los elementos hasta antes del indice 2
7
8  console.log(frutas);
9  //imprime [ "pera", "manzana", "plátano", "uvas", "mandarina" ]
10
11 newFrutas = frutas.slice(2);
12 console.log(newFrutas);
13 //Output: [ "plátano", "uvas", "mandarina" ]
14
15 newFrutas2 = frutas.slice(1,3);
16 console.log(newFrutas2);
17 //Output: [ "manzana", "plátano" ]
18 //Índice      1      2
19 //      (start = 1) (end-1 = 3-1 = 2)
```


splice

Sirve para agregar o borrar elementos de un arreglo. Pide como parámetros el **index** y un **número** de elementos a borrar. Splice modifica el arreglo original.



```
1 array.splice(index, noDeElementosABorrar, 'elementos', 'por', 'agregar')
```

Agrega los elementos en el lugar del índice indicado.

splice



```
1  var frutas = ['pera', 'manzana', 'plátano', 'uvas', 'mandarina'];
2  // Index      0      1      2      3      4
3
4  frutas.splice(2, 0, 'limón', 'sandía');
5
6  console.log(frutas);
7  //Output: ['pera', 'manzana', 'limón', 'sandía', 'plátano', 'uvas', 'mandarina']
8  // Index      0      1      2      3      4      5      6
9  // nuevos valores agregados      ^      ^
```

No se borraron elementos ya que el segundo parámetro es 0 (cero)

splice



```
1  var frutas = ['pera', 'manzana', 'plátano', 'uvas', 'mandarina'];
2  // Index      0      1      2      3      4
3
4  frutas.splice(2, 1, 'limón', 'sandía');
5  // regresa en consola los valores eliminados, elimina 1 elemento a partir del índice 2
6  // Output: ['plátano']
7
8  console.log(frutas);
9  //Output: ['pera', 'manzana', 'limón', 'sandía', 'uvas', 'mandarina']
10 // Index      0      1      2      3      4      5
```

Se borró 'plátano' y en su lugar se agregó 'limón' y 'sandía'

sort

Método que ordena los arreglos. Por default se ordenan de manera ascendente convirtiendo los elementos a strings y comparando los valores UTF-16 de las secuencias.

No modifica el arreglo original



```
1  const letters = ['b', 'p', 'd', 'a'];
2  const sortedLetters = letters.sort();
3
4  console.log(sortedLetters);
5  // → [ 'a', 'b', 'd', 'p' ]
```



```
1  const letters = ['b', 'P', 'd', 'a'];
2  const sortedLetters = letters.sort();
3
4  console.log(sortedLetters);
5  // → [ 'P', 'a', 'b', 'd' ]
```

reverse

Coloca los elementos del arreglo al revés. Este método altera el arreglo original.



```
1  const letters = ['b', 'P', 'd', 'a'];  
2  letters.reverse();  
3  
4  console.log(letters);  
5  // → [ 'a', 'd', 'P', 'b' ]
```

concat

Este método une (concatena) el contenido de 2 arreglos existentes. **No modifica dichos arreglos**, si no que devuelve uno nuevo.

Se puede concatenar más de un arreglo.

```
1  const letters1 = ['a', 'e', 'i'];  
2  const letters2 = ['o', 'u'];  
3  const letters3 = [1, 2];  
4  
5  const lettersConcat = letters1.concat(letters2, letters3);  
6  
7  console.log(lettersConcat);  
8  // → ['a', 'e', 'i', 'o', 'u', 1, 2]
```

Actividad

Ejercicios Arreglos

Challenge:

Crea el siguiente arreglo de personajes de Harry Potter y la Cámara Secreta:

Dobby, Harry, Vernon, Fred, George, Ron, Molly, Arthur, Lockhart, Lucius

Usando los métodos de arreglos llega a al siguiente arreglo, que contiene los personajes de Harry Potter y el prisionero de Azkaban:

Marge, Harry, Vernon, Petunia, Stan, Ron, Hermione, Lupin, Sirius, Buckbeak