Arreglos

DEV.F:
DESARROLLAMOS(PERSONAS);

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.

Propiedades

Array Props

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
```

Manipulación de Arreglos

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

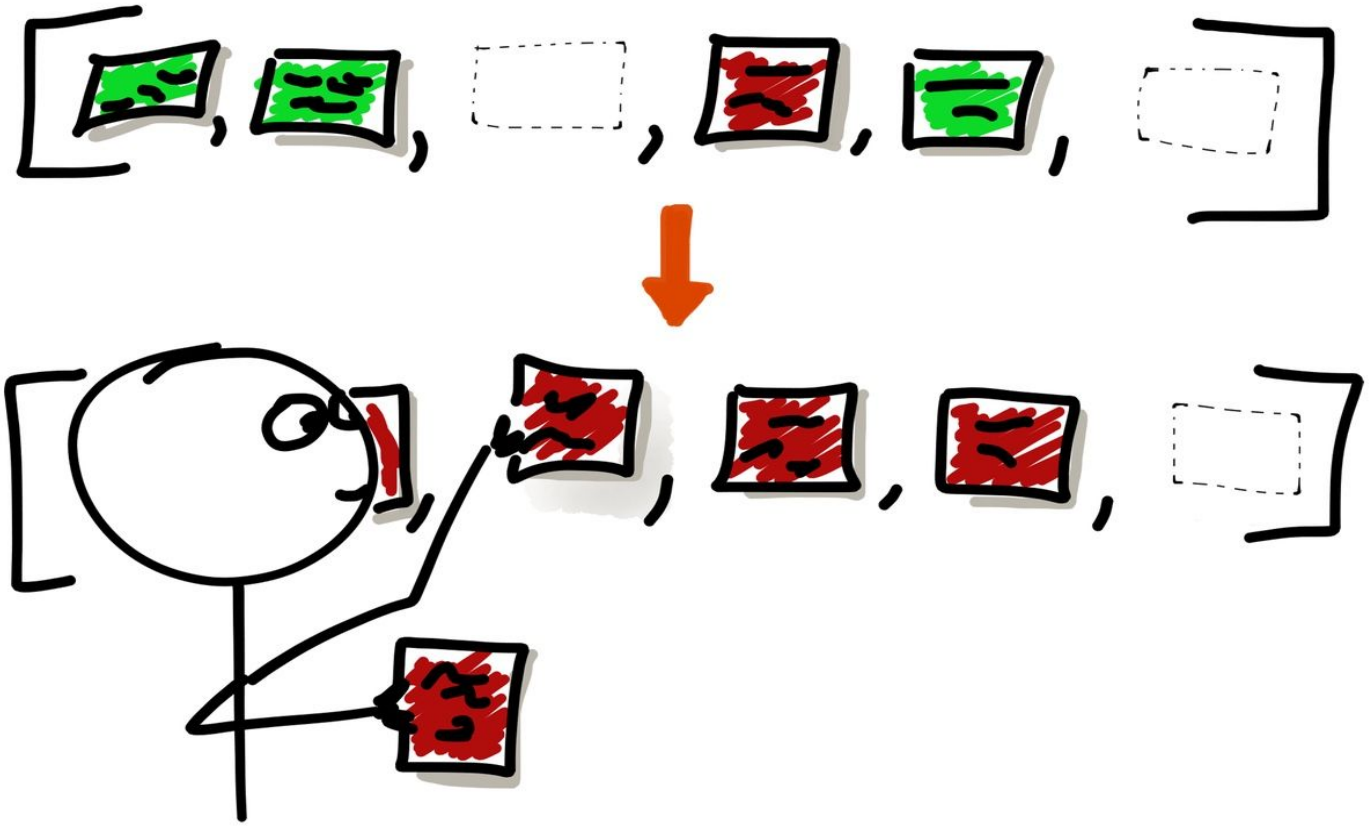
```
> frutas[3] = "Uvas";
<· "Uvas"

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

Métodos de Arrays

DEV.F:
DESARROLLAMOS(PERSONAS);

Métodos de Arreglos (Arrays)



push y pop

```
> frutas;
<· ▶ (4) ["Pera", "Manzana", "Platano", "Uvas"]
```

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

```
> frutas.push("Mandarina");
<· 5
> frutas;
<· ▶ (5) ["Pera", "Manzana", "Platano", "Uvas", "Mandarina"]
```

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

```
> frutas.pop();
<· "Mandarina"
> frutas;
<· ▶ (4) ["Pera", "Manzana", "Platano", "Uvas"]
```

unshift y shift

```
> frutas;
<· ▶ (4) ["Pera", "Manzana", "Platano", "Uvas"]
```

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

```
> frutas.unshift("Mandarina");
<· 5
> frutas;
<· ▶ (5) ["Mandarina", "Pera", "Manzana", "Platano", "Uvas"]
```

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

```
> frutas.shift();
<· "Mandarina"
> frutas;
<· ▶ (4) ["Pera", "Manzana", "Platano", "Uvas"]
```

slice

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

```
> var verduras = ['Cebolla','Perejil','Tomate','Calabaza','Brocoli'];
```

Debe indicarse al menos una posición inicial (start). La posición inicial es 0.

```
> verduras.slice(2);
◂· ▶ (3) ["Tomate", "Calabaza", "Brocoli"]
```

Opcionalmente también se puede indicar una posición final (end).

```
> verduras.slice(1,3);
◂· ▶ (2) ["Perejil", "Tomate"]
```

reverse

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

```
> var verduras = ['Cebolla','Perejil','Tomate','Calabaza','Brocoli'];
> verduras.reverse();
<· ▶ (5) ["Brocoli", "Calabaza", "Tomate", "Perejil", "Cebolla"]
```

```
> [1,2,3,4,7,8,9].reverse()
<· ▶ (7) [9, 8, 7, 4, 3, 2, 1]
```

sort

Ordena la lista de forma ascendente (A-Z) por defecto.

```
> var verduras = ['Cebolla','Perejil','Tomate','Calabaza','Brocoli'];
> verduras.sort();
<- ▶ (5) ["Brocoli", "Calabaza", "Cebolla", "Perejil", "Tomate"]
```

Es posible pasarle una función para ajustar el orden. Sobre todo para números, ya que por defecto no los ordena correctamente.

```
> [2,5,1,3,46,70,34].sort();
<- ▶ (7) [1, 2, 3, 34, 46, 5, 70]
> [2,5,1,3,46,70,34].sort(function(a, b){return a-b});
<- ▶ (7) [1, 2, 3, 5, 34, 46, 70]
```

concat

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

```
> var verduras = ['Cebolla','Perejil','Tomate'];
> var frutas = ['Manzana','Pera','Platano'];

> var listaDeCompras = verduras.concat(frutas);

> listaDeCompras;
< ▶ (6) ["Cebolla", "Perejil", "Tomate", "Manzana", "Pera", "Platano"]
```

join()

El método join() crea y retorna un nuevo string concatenando todos los elementos contenidos en el arreglo, separados por comas y por un separador específico. Si el arreglo contiene un solo elemento, retornará únicamente el ítem.

indexOf()

Este método retorna el primer index donde un elemento dado puede ser encontrado en el array. Retorna -1 cuando el elemento no está presente.

Actividad 5:

Iterando arreglos

1. Construye un código bajo las siguientes reglas:

   * Itera arreglo=[1, 4, 6, 10, 22, 55, 46, 2, 5, 0] utilizando un ciclo for

   * Imprime en consola los valores que sean mayores 3

2. Construye un código bajo las siguientes reglas

   *Declara un arreglo vacío *Con un ciclo While que se ejecute 5 veces, agrega los elementos al array

3. Por medio del ciclo **for** iterar el array arreglo=[1, 4, 6, 10, 22, 55, 46, 2, 5, 0] e imprimir cada uno de los elementos de manera ordenada.

Actividad 6:

Métodos de arreglos

1. Concatena los siguientes arreglos:

var animals = ['eagle', 'parrot', 'monkey', 'boar', 'lion']

var comingSoonAnimals = ['panter', 'dragon', 'turtle']

2. Acomoda el arreglo de menor a mayor: var arr = [4, 6, 1, 0, 8, 2, 45, 11, 5, 33, 50, 101]
3. Agrega un nuevo animal (‘cow’)
4. Retira el elemento ‘eagle’ del arreglo animals
5. Encuentra el index del animal ‘parrot’
6.