J2

Loops - datos - funciones

SENPAI academy

Javascript

Bucles con **For** y **While**

```
// For valorInicial; condiciónDeFrenado; siguienteValor
const precios = [10, 1500, 24.5, 320];
let total = 0;

for (let i = 0; i < precios.length; i++) {
    total = total + precios[i];
}

// While
let balance = 120;
let total = 0;
let items = [10, 20, 30, 24.4, 100,1];
let itemsQuePuedeComprar = [];
let i = 0;

while (total <= balance) {
    total = total + items[i];
    itemsQuePuedeComprar.push(items[i]);
    i++;
    if (i >= items.length) {
        break;
    }
}
```

A veces es necesario ejecutar un fragmento de código varias veces. En esos casos podemos utilizar un bucle.

Usamos For cuando sabemos con anticipación la cantidad de iteraciones a realizar.

Usamos While cuando no sabemos la cantidad necesaria de iteraciones

Usamos Do/While (no está en la foto) cuando, además de no saber la cantidad de iteraciones, queremos que se corra al menos una vez.

Nota: también existen métodos de arreglos como map(), forEach() o reduce() que muchas veces pueden ser una opción más elegante.

SENPAI ACADEMY

LOOPS
for & while

A veces es necesario ejecutar un fragmento de código varias veces. En esos casos podemos utilizar un bucle.

Usamos For cuando sabemos con anticipación la cantidad de iteraciones a realizar.

Usamos While cuando no sabemos la cantidad necesaria de iteraciones

Usamos Do/While (no está en la foto) cuando, además de no saber la cantidad de iteraciones, queremos que se corra al menos una vez.

Nota: también existen métodos de arreglos como map(), forEach() o reduce() que muchas veces pueden ser una opción más elegante.

For

- Valor inicial
- Condición
- Incremento

```
// For valorInicial; condiciónDeFrenado; siguienteValor
const precios = [10, 1500, 24.5, 320];
let total = 0;

for (let i = 0; i < precios.length; i++) {
    total = total + precios[i];
}

// While
let balance = 120;
let total = 0;
let items = [10, 20, 30, 24.4, 100,1];
let itemsQuePuedeComprar = [];
let i = 0;

while (total <= balance) {
    total = total + items[i];
    itemsQuePuedeComprar.push(items[i]);
    i++;
    if (i >= items.length) {
        break;
    }
}
```

ARREGLOS
ESTRUCTURAS DE DATOS

¿Qué pasa si queremos guardar muchos datos, pero no sabemos cuántos?

¿Cómo hará youtube para saber todos los videos que voy a poner en mi lista si ni yo se cuantos me van a gustar?

La clave es la palabra **LISTA**

Arreglos (Arrays)

Los arrays agrupan datos de cualquier tipo en una lista ordenada.

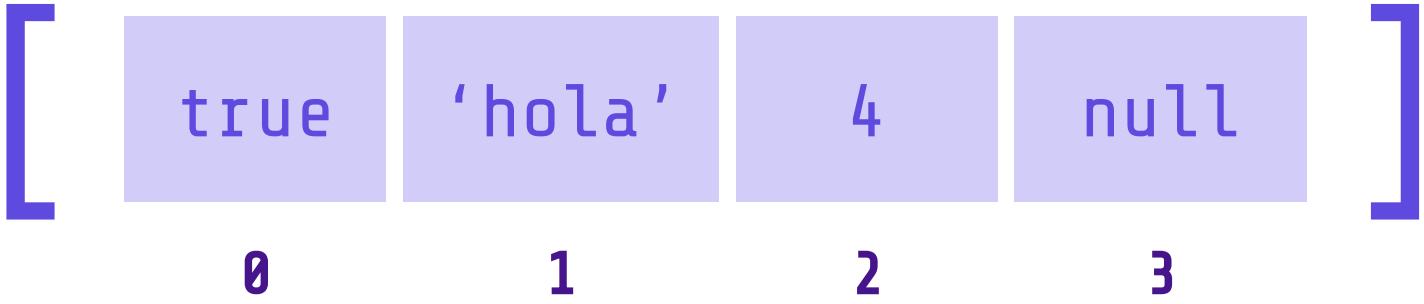
No tienen una longitud fija y se pueden agregar o remover elementos mediante dos operaciones: **`push`** y **`pop`**.

```
[1, 3, 10, 25, 25]
[true, 3, 'HOLA']
[]
```

Estas son listas o arreglos válidos de elementos.

Accediendo un array

Los arrays tienen índices que permiten acceder a cada elemento almacenado en cada espacio del array. La primera posición siempre es cero (**0**)



Arrays: función map()

```
// Creamos función que recibe un número y lo duplica
function duplicarValor(valor) {
    return valor * 2;
}

// Creamos un array de números
const valores = [1,2,3,4];

// Creamos un array de valores duplicados
// utilizamos la función .map() y pasamos 'duplicar' como parámetro
const valoresDuplicados = valores.map(duplicar);
```

- Los arreglos tienen un buen número de funciones predefinidas, y map() es una de ellas.
- Toma como parámetro una función y la ejecuta para cada elemento del array.
- Al terminar, devuelve un array con los resultados de cada ejecución.

En grupo investigar los siguientes métodos de array:

- Filter()
- FindIndex()
- Find()
- Push() y pop()
- Includes()
- Sort()

Documentación: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

SENPAI ACADEMY

OBJETOS
ESTRUCTURAS DE DATOS

Cuando programamos, buscamos representar una abstracción de la realidad.

Si pensamos una página web que muestre los datos de una persona, quisiéramos trabajar con código que ‘represente’ una persona

Por ejemplo, podríamos crear un objeto que represente a una persona.

Objetos (Object)

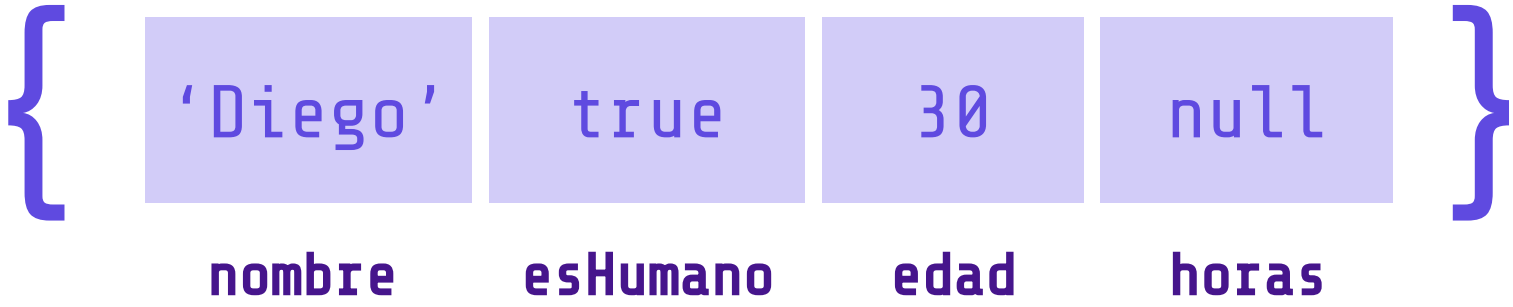
Los objetos agrupan datos de cualquier tipo en una lista identificada por llaves (keys) y valores.

Las llaves le ponen nombre a los valores que ese objeto tiene.

```
let persona = {
  nombre: 'Diego',
  apellido: 'Rompecódigos',
  edad: 30,
  esProfe: true
}
```

Accediendo un objeto

Los objetos identifican sus valores usando las llaves y para acceder a esos valores usamos la key.



Modificando valores

Los valores de los objetos se pueden modificar usando el operador de asignación a una de las keys del mismo:

```
let persona = {
  nombre: 'Diego'
  apellido: 'Rompecódigos',
  edad: 30,
  esProfe: true
}
```

```
console.log(persona.nombre)
 ➜ 'Diego'
persona.nombre = 'Gabo'
console.log(persona.nombre)
 ➜ 'Gabo'
```

¡GRACIAS!

SENPAI
academy