Arrays y Objetos

Métodos transformadores para añadir, eliminar y modificar elementos

{desafío} latam_

Utilizar estructuras de tipo arreglo y sentencias iterativas para el control del flujo de un algoritmo que resuelve un problema simple acorde al lenguaje Javascript.

Utilizar objetos preconstruidos para la codificación de un algoritmo que resuelve un problema acorde al lenguaje Javascript.

- Unidad 1:
  Introducción al lenguaje JavaScript
- Unidad 2:
  Funciones y Ciclos
- Unidad 3:
  Arrays y Objetos
- Unidad 4:
  APIs

{desafío}
latam_

¿Qué aprenderás en esta sesión?

- *Utiliza operaciones de creación, agregación, modificación y eliminación de elementos de un arreglo acorde al lenguaje Javascript para resolver un problema.*

{desafío} latam_

A partir de lo aprendido, ¿cuál es la principal ventaja de trabajar con objetos?

¿A qué método corresponde la siguiente definición? “Retorna un arreglo con los valores correspondientes a las propiedades dispuestas en un objeto”

{desafío} latam_

Métodos Integrados con Arreglos

- Como ya mencionamos anteriormente, los arreglos son objetos en JavaScript. Esto quiere decir que cada arreglo que creamos hereda por defecto, propiedades y métodos:

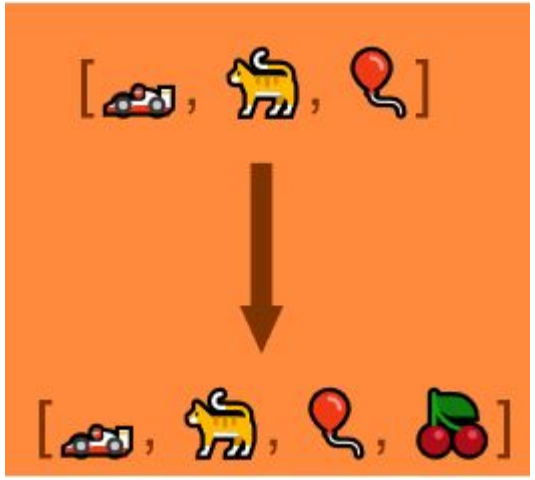
- Los métodos que muestra la imagen nos pueden ayudar a cumplir ciertas tareas, que por otras vías, serían mucho más difíciles de alcanzar.

/*Métodos que modifican los arreglos*/

Métodos que modifican los arreglos

Método push

- Nos permite agregar uno o más elementos al final de nuestro array



- Por ejemplo, agregando tres elementos al arrays amigos

```
var amigos = ["Erick", "Cristian", "Max",
"Claudia"];
amigos.push("Gary", "Arturo", "Alexis");
console.log(amigos);
```

Demostración - “Método push”

Ejercicio guiado

Método push

Incorporar dos objetos usando el método push. En este caso, el arreglo principal contiene los objetos: {auto: "Peugeot", color: "rojo" },{auto: "Mazda", color: "azul" },{auto: "BMW", color: "negro" }.

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

Ejercicio guiado

Método push

- **Paso 2:** Ahora para poder pasar objetos a un arreglo, simplemente se debe colocar el objeto completo dentro de los paréntesis del push y separar por coma si se va a agregar otro objeto.

```
var autos = [{auto: "Peugeot", color: "rojo" },{auto: "Mazda", color: "azul" },{auto:
"BMW", color: "negro" }];
autos.push({auto: "Suzuki", color: "verde" },{auto: "Chevrolet", color: "amarillo" });
console.log(autos);
```

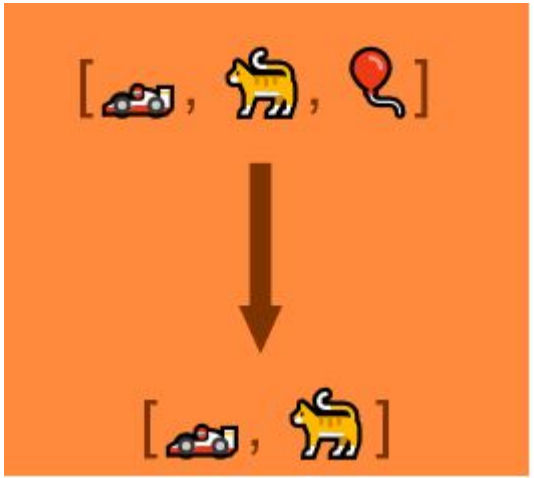
- **Paso 3:** Ejecutar el código anterior en el navegador, se generará lo siguiente:

```
(5) [{auto: "Peugeot", color: "rojo" },{auto: "Mazda", color: "azul"
},{auto: "BMW", color: "negro" },{auto: "Suzuki", color: "verde" },{auto:
"Chevrolet", color: "amarillo" }]
```

Métodos que modifican los arreglos

Método pop

- Se utiliza para eliminar el último dato que tenemos en el arreglo



- Por ejemplo;

```
var amigos = ["Erick", "Cristian", "Max",
"Claudia"];
amigos.pop();
console.log(amigos);
```

```
(3) ["Erick", "Cristian", "Max"]
```

Demostración - “Método pop”

Ejercicio guiado

Método pop

Eliminar el último objeto del listado mostrado a continuación: [{titulo: "Harry Potter", pag: "300" },{titulo: "El principito", pag: "100" },{titulo: "De animales a dioses", pag: "350" }].

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

Ejercicio guiado

Método pop

- **Paso 2:** En el archivo script.js, agregar el siguiente código. En este caso, como estamos en presencia de un objeto dentro de un arreglo, utilizamos el método pop() sin pasar ningún tipo de valor entre los paréntesis:

```
let libros = [{titulo: "Harry Potter", pag: "300" },{titulo: "El principito", pag: "100" },{titulo: "De animales a dioses", pag: "350" }];
libros.pop();
console.log(libros);
```

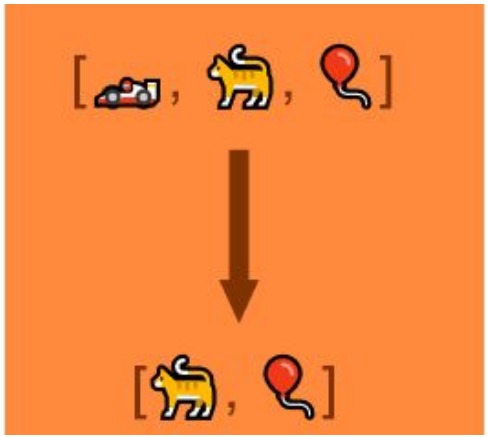
- **Paso 3**: Al ejecutar el código anterior y revisar la consola del navegador web, nos generará una salida como la siguiente:

```
(2) [{titulo: "Harry Potter", paginas: "300" },{titulo: "El principito", paginas: "100" }]
```

Métodos que modifican los arreglos

Método shift

- Nos permite eliminar un elemento de un arreglo, pero en este caso el elemento eliminado no es el último sino el primero.



- Por ejemplo:

```
var amigos = ["Erick", "Cristian", "Max",
"Claudia"];
amigos.shift();
console.log(amigos);
```

```
(3) ["Cristian", "Max", "Claudia"]
```

Demostración - “El método shift”

Ejercicio guiado

Método shift

Emplear el método shift y eliminar el primer elemento del siguiente arreglo: {titulo: "Harry Potter", pag: "300" },{titulo: "El principito", pag: "100" },{titulo: "De animales a dioses", color: "350" }, por consiguiente:

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

Ejercicio guiado

Método shift

- **Paso 2**: Si queremos eliminar el primer objeto de un array implementado shift(), debemos aplicar ese método al arreglo sin pasar ningún valor dentro de los paréntesis, quedando el siguiente código que agregamos al archivo script.js:

```
var libros = [{titulo: "Harry Potter", pag: "300" },{titulo: "El
principito", pag: "100" },{titulo: "De animales a dioses", pag: "350" }];
libros.shift();
console.log(libros);
```

- **Paso 3**: Ejecutar en el navegador web lo anterior, se generará a siguiente salida:

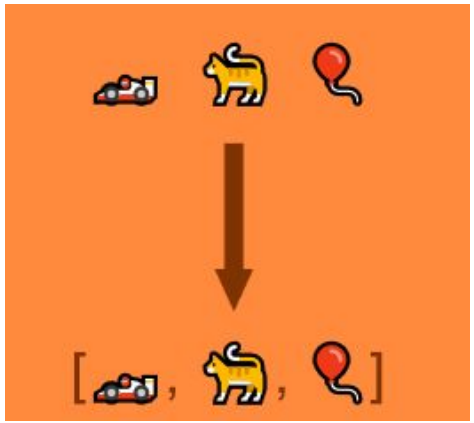
```
(2) [{titulo: "El principito", paginas: "100" },{titulo: "De animales a
dioses", paginas: "350" }]
```

/* Métodos que conservan el arreglo inicial y generan uno nuevo */

Métodos que conservan el arreglo inicial y generan uno nuevo

Método split

- El método split nos permite dividir una cadena de texto string en un arreglo. Aunque este método no se aplica en arreglos y objetos directamente, sí nos entrega como resultado un arreglo



- Por ejemplo,

```
var cliente = 'Juan Carlos;29;jcarlos@gmail.com';
var arregloCliente = cliente.split(';');
console.log(arregloCliente);
```

```
(3) ["Juan Carlos", "29", "jcarlos@gmail.com"]
```

Demostración - “Método split”

Ejercicio guiado

Método split

Tenemos el siguiente texto en una variable denominada filtros: “Comida china RM nunoa” y queremos construir un arreglo con esa cadena de caracteres mediante los espacios que tiene.

- **Paso 1:** Crea una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

Ejercicio guiado

Método split

- **Paso 2:** Ahora, como queremos construir un nuevo arreglo con la variable filtros, utilizamos el método split. Dentro de los paréntesis pasamos un espacio como parámetro, que será el separador común que utilizaremos para construir el nuevo arreglo, quedando el siguiente código que incluiremos en el archivo script.js:

```
var filtros = 'Comida china RM nunoa';
var filtros = filtros.split(' ');
console.log(filtros);
```

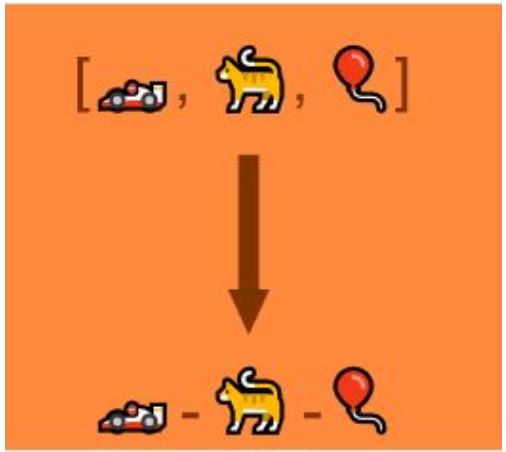
- **Paso 3:** El resultado de lo anterior sería un arreglo como el siguiente:

```
(3) ["Comida", "china", "RM", "nunoa"]
```

Métodos que conservan el arreglo inicial y generan uno nuevo

Método join

- El método join une todos los elementos de un arreglo especificado en una sola cadena de texto



- Demos formato a este array transformándolo a una cadena de texto separada por "-”

```
var amigos = ["Erick", "Cristian", "Max",
"Claudia"];
console.log(amigos.join(" - "));
```

```
Erick - Cristian - Max - Claudia
```

Demostración - “Método join”

Ejercicio guiado

Método join

Se solicita dar formato al siguiente arreglo ["Rebeca Matte 18","Santiago"] que contiene dos elementos como string e implementado el método join, donde el separador debe ser una coma (,).

- **Paso 1**: Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

Ejercicio guiado

Método join

- **Paso 2**: Teniendo el arreglo definido, aplicar el método join y dentro del paréntesis agregamos una coma con un solo espacio, para poder separar los dos elementos del arreglo en uno solo pero separado por la coma. Quedando el siguiente código que agregamos al archivo script.js:

```
var dirección = ["Rebeca Matte 18","Santiago"];
console.log(dirección.join(", "));
```

- **Paso 3**: Ejecutar el código anterior en el navegador web, nos entregará una salida como la siguiente:

```
"Rebeca Matte 18, Santiago"
```

Métodos que conservan el arreglo inicial y generan uno nuevo

Método map

- El método map nos permite obtener un array procesado en base a otro array, como muestra la siguiente imagen:

- Por consiguiente, la estructura básica es:

```
var nuevo_array = arreglo.map(function
(elemento, index, array) {
      // Elemento devuelto de nuevo_array
};
```

Demostración - “Método map”

Ejercicio guiado

Método map

Partiendo de un arreglo que posee distintos objetos, se requiere crear un nuevo arreglo que contenga toda la información de los clientes originales y que precise un nuevo elemento indicando si el cliente es mayor de edad. El arreglo original es:

```
var clientes = [
  {nombre: 'Juan', edad: 28},
  {nombre: 'Carlos', edad: 17},
  {nombre: 'Karla', edad: 27},
];
```

Ejercicio guiado

Método map

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- **Paso 2:** En el archivo script.js escribir el arreglo facilitado al inicio del ejercicio así como la estructura básica del método map, el cual lleva una función interna encargada de ejecutarse por cada elemento que contenga el arreglo al cual le estamos aplicando el método:

```
clientes_modificado = clientes.map(function(cliente) {
});
```

Ejercicio guiado

Método map

- **Paso 3**: Ahora por cada cliente en el array realizamos una validación (utilizando if) para revisar si la edad es mayor o igual a 18 (para verificar que sea mayor de edad).

```
clientes_modificado =
clientes.map(function(cliente) {
   if (cliente.edad >= 18) {
        cliente.adulto = true;
  } else {
        cliente.adulto = false;
  }
   return cliente;
});

console.log(clientes_modificado);
```

- **Paso 4**: Ejecutar el código anterior el resultado será:

```
0: {nombre: "Juan", edad: 28, adulto: true}
1: {nombre: "Carlos", edad: 17, adulto:
false}
2: {nombre: "Karla", edad: 27, adulto: true}
```

Demostración - “Multiplicando elementos con map”

{desafío} latam_

Ejercicio guiado

Multiplicando elementos con map

Aplicar el método map para multiplicar por 2 los elementos del siguiente arreglo [10,15,20,25,30] y guardar el nuevo arreglo en una variable.

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

Ejercicio guiado

Multiplicando elementos con map

- **Paso 2:** En el archivo script.js escribir el arreglo facilitado al inicio del ejercicio así como la estructura básica del método map, que lleva una función interna encargada de ejecutarse por cada elemento que contenga el arreglo, luego, dentro de la función, lo que se quiere es retornar el valor del elemento multiplicado por dos.

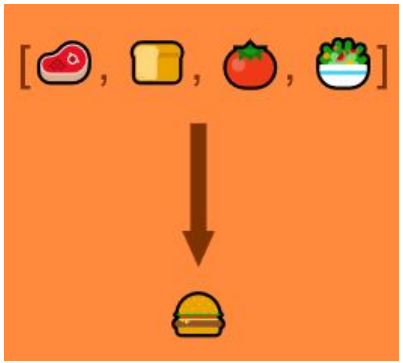
```
var numeros = [10,15,20,25,30];

numeros_nuevo =
numeros.map(function(num) {
   return num * 2;
});
console.log(numeros_nuevo);
```

```
(5) [ 20, 30, 40, 50, 60 ]
```

Métodos que conservan el arreglo inicial y generan uno nuevo

Método reduce

- Con reduce podemos transformar los elementos de un array en un valor único. Es de gran utilidad si por ejemplo, deseamos sumar dichos valores.



- La sintaxis básica de este método, es la siguiente:

```
arreglo.reduce(callback(acumulador,
valorActual[, índice[, array]])[,
valorInicial])
```

Demostración - “Método reduce”

Ejercicio guiado

Método reduce

Utilizar el método reduce para sumar las deudas de una persona, las cuales están almacenadas en un array, por ejemplo: [10000, 5000, 50000, 35000].

- **Paso 1**: Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

Ejercicio guiado

Método reduce

- **Paso 2:** En el archivo script.js escribir el arreglo facilitado al inicio del ejercicio con una variable “deudas”, así como la estructura básica del método reduce. En este caso, el método reduce recibirá como parámetro una función, que a su vez recibirá dos parámetros: un contador y la deuda. El contador representa el resultado de lo ejecutado dentro de la función, mientras que la deuda representa el elemento actual en el cual se está iterando.

```
var deudas = [10000, 5000, 50000, 35000];

var sumatoriaDeudas =
deudas.reduce(function(contador, deuda){
   return contador + deuda;
});

console.log(sumatoriaDeudas);
```

```
100000
```

Demostración - “Utilizando reduce sobre objetos”

Ejercicio guiado

Utilizando reduce sobre objetos

Es posible utilizar reduce sobre objetos, pero la ejecución cambia un poco. Se solicita imprimir por consola una lista de nombres de un arreglo sobre un único string, siendo los objetos del arreglo: {nombre: 'Juan', edad: 28}, {nombre: 'Carlos', edad: 17}, {nombre: 'Karla', edad: 27}.

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

Ejercicio guiado

Utilizando reduce sobre objetos

- **Paso 2:** En el archivo script.js, escribir el arreglo facilitado al inicio del ejercicio con una variable “clientes”, así como la estructura básica del método reduce. En este caso, el método reduce recibirá un parámetro adicional que representa el objeto o variable inicial sobre la cual se acumularán las operaciones realizadas dentro de reduce, la cual al finalizar la ejecución será retornada a la variable nombres y tiene como valor inicial un string vacío. Dentro de la función entregada a reduce, acumulamos los nombres de cada cliente en el arreglo de clientes.

```
var clientes = [{nombre:
'Juan', edad: 28},{nombre:
'Carlos', edad: 17},{nombre:
'Karla', edad: 27}];

var nombres =
clientes.reduce(function(acumul
ador, cliente){
       return acumulador + ' |
' + cliente.nombre;
}, '');

console.log(nombres);
```

```
| Juan | Carlos | Karla
```

Demostración - “Utilizando reduce para obtener totales”

{desafío} latam_

Ejercicio guiado

Utilizando reduce para obtener totales

Del siguiente arreglo de objetos, se debe obtener la cantidad total de años de experiencia:

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

```
var experiencias = [
        {
           titulo: "Practica",
           anios: 1,
        },
        {
           titulo: "Programador Junior",
           anios: 2,
        },
        {
           titulo: "Programador Senior",
           anios: 4,
        },
        {
           titulo: "Jefe de proyecto",
           anios: 5,
        }
];
```

Ejercicio guiado

Utilizando reduce para obtener totales

- **Paso 2:** En el archivo script.js, trabajar con la estructura básica del método reduce, en este caso, recibirá un valor adicional. Este parámetro representa el objeto o variable inicial que se le cargará al acumular como valor de inicio para la primera iteración, luego sumará lo que tenga en el acumulador con el valor actual que sería la experiencia en años de cada objeto, retornando un solo valor numérico final.

```
var aniosDeExperiencia = experiencias.reduce(function(acumulador,
experiencia){
   return acumulador + experiencia.anios;
},0);

console.log(aniosDeExperiencia);
```

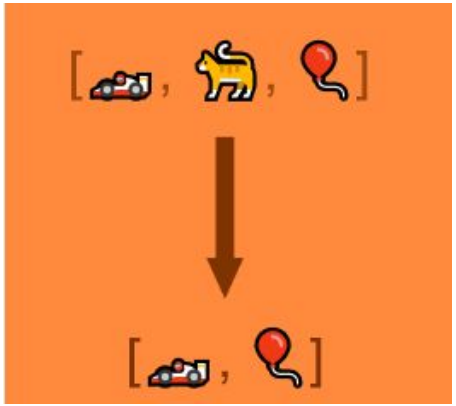
```
12
```

{desafío} latam_

Métodos que conservan el arreglo inicial y generan uno nuevo

Método filter

- El método filter permite realizar un filtrado de objetos o elementos que se encuentran al interior de un arreglo y devolver el resultado en forma de arreglo mediante una iteración.



- La sintaxis básica del método filter es la siguiente:

```
var nuevoArreglo =
arreglo.filter(callback(currentValue
[, index[, array]])[, thisArg])
```

Demostración - “Método filter”

Ejercicio guiado

Método filter

Se desea filtrar y obtener solo los clientes que sean mayores de edad del siguiente array con datos de clientes:

```
var clientes = [
  {nombre: 'Juan', edad: 28},
  {nombre: 'Carlos', edad: 17},
  {nombre: 'Karla', edad: 27},
];
```

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

Ejercicio guiado

Método filter

- **Paso 2:** En el archivo script.js, agregar el arreglo llamado clientes. Luego, creamos una variable llamada adultos donde almacenaremos el resultado del filtro. En el interior del método filter, retornamos un valor booleano para saber si el objeto se filtra o no. En este caso, el filtro es la edad del cliente: si la edad es mayor o igual a 18, el objeto no se filtrará, en cambio, si la edad del cliente es menor a eso, se filtrará y no se almacenará.

```
var clientes = [
  {nombre: 'Juan', edad: 28},
  {nombre: 'Carlos', edad: 17},
  {nombre: 'Karla', edad: 27},
];

var adultos =
clientes.filter(function(cliente
){
       return cliente.edad >= 18
});

console.log(adultos);
```

```
{nombre: "Juan", edad: 28}
{nombre: "Karla", edad: 27}
```

Demostración - “Utilizando filter con una condición”

{desafío} latam_

Ejercicio guiado

Utilizando filter con una condición

Del siguiente arreglo, obtener sólo los trabajos donde se mantuvo más de 2 años continuos.

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

```
var experiencias = [
        {
           titulo: "Practica",
           anios: 1,
        },
        {
           titulo: "Programador Junior",
           anios: 2,
        },
        {
           titulo: "Programador Senior",
           anios: 4,
        },
        {
           titulo: "Jefe de proyecto",
           anios: 5,
        }
];
```

Ejercicio guiado

Utilizando filter con una condición

- **Paso 2:** En el archivo script.js, agregar el arreglo indicado en el enunciado del ejercicio, solo nos interesa por el momento los años, porque por medio de esa información se aplicará el filtro correspondiente. Luego, creamos una variable llamada anosDeExperiencia donde almacenaremos el resultado del filtro, es decir, solo los que tengan más de dos años de experiencia serán guardados en la nueva variable.

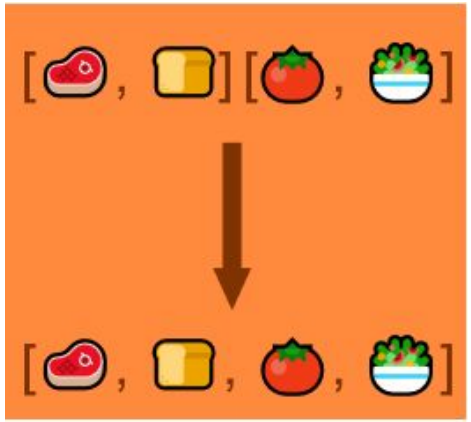
```
var aniosDeExperiencia = experiencias.filter(function(experiencia){
   return experiencia.anios > 2;
});
console.log(aniosDeExperiencia);
```

```
(2) [{ titulo: "Programador Senior", anios: 4 },{ titulo: "Jefe de proyecto", anios: 5 }]
```

Métodos que conservan el arreglo inicial y generan uno nuevo

Método concat

- Podemos fusionar los elementos de dos o más arrays dentro de un solo resultado. Puede ser muy útil para unir elementos que pertenezcan a una misma clasificación



- La sintaxis básica del método concat es la siguiente:

```
var nuevo_arreglo =
arreglo_1.concat(arreglo_2);
```

Demostración - “Método concat”

{desafío} latam_

Ejercicio guiado

Método concat

Se necesita fusionar los elementos de dos arreglos denominados `autosCompactos` y `autosDeportivos`, de acuerdo a lo siguiente:

```
var autosCompactos = [
  {marca: 'Toyota', modelo: 'Corolla', combustible: 'Gasolina'},
  {marca: 'Mazda', modelo: '3', combustible: 'Gasolina'},
  {marca: 'Honda', modelo: 'Civic', combustible: 'Gasolina'},
  {marca: 'Bmw', modelo: '116d', combustible: 'Diesel'},
];
var autosDeportivos = [
  {marca: 'Opel', modelo: 'Astra OPC', combustible: 'Gasolina'},
  {marca: 'Renault', modelo: 'Megane RS', combustible: 'Gasolina'},
  {marca: 'Mitsubishi', modelo: 'Lancer Evo', combustible: 'Gasolina'},
];
```

Ejercicio guiado

Método concat

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- **Paso 2:** En el archivo script.js, agregar los dos arreglos indicados, luego mediante el método concat al arreglo de `autosCompactos` le unimos con el arreglo de `autosDeportivos`.

```
var autos = autosCompactos.concat(autosDeportivos);

console.log(autos);
```

Ejercicio guiado

Método concat

- **Paso 3**: El resultado de la operación anterior sería un arreglo con el siguiente contenido:

```
0: {marca: "Toyota", modelo: "Corolla", combustible: "Gasolina"}
1: {marca: "Mazda", modelo: "3", combustible: "Gasolina"}
2: {marca: "Honda", modelo: "Civic", combustible: "Gasolina"}
3: {marca: "Bmw", modelo: "116d", combustible: "Diesel"}
4: {marca: "Opel", modelo: "Astra OPC", combustible: "Gasolina"}
5: {marca: "Renault", modelo: "Megane RS", combustible: "Gasolina"}
6: {marca: "Mitsubishi", modelo: "Lancer Evo", combustible: "Gasolina"}
```

Demostración - “Utilizando concat para unir arreglos”

{desafío} latam_

Ejercicio guiado

Utilizando concat para unir arreglos

Se requiere unir los siguientes arreglos de letras, letrasA = ["a","b","c","d"] y letrasB = ["e","f","g","h"], en un solo arreglo.

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- **Paso 2:** En el archivo script.js, agregar los dos arreglos indicados, luego mediante el método concat al arreglo letrasA le unimos el arreglo letrasB para obtener un solo arreglo.

```
var letrasA = ["a","b","c","d"];
var letrasB = ["e","f","g","h"];

var letras = letrasA.concat(letrasA);
console.log(letras);
```

```
(8) ["a","b","c","d","e","f","g","h"];
```

¿Existe algún concepto que no hayas comprendido?

Volvamos a revisar los conceptos que más te hayan costado antes de seguir adelante.

Próxima sesión...

- *Reconocer los métodos integrados para acceder, recorrer y ordenar elementos de un objeto o arreglo, para utilizar adecuadamente las características del lenguaje JavaScript.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales