Arrays y Objetos

Métodos para acceder a 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.

{desafío} latam_

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

¿Qué aprenderás en esta sesión?

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

A partir de lo aprendido, ¿cuál es la principal diferencia entre los métodos pop y shift?

¿Qué método permite fusionar los elementos de dos o más arrays dentro de un solo resultado?

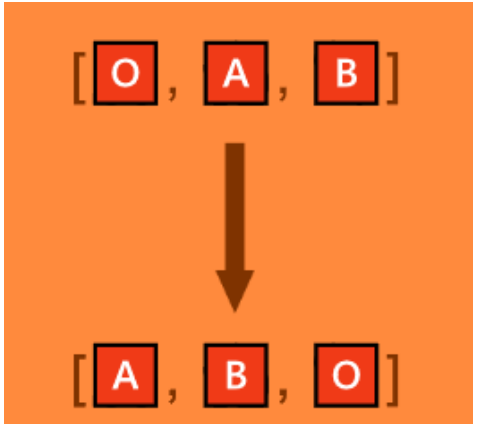
{desafío} latam_

/* Métodos para acceder a elementos */

Métodos para acceder a elementos

Método sort

- Este método nos permite ordenar de manera alfabética los tipos de datos string. Para otros tipos de datos, sort ordena de acuerdo a su valor en el estándar Unicode:



- Ordenemos el siguiente array usando el método sort:

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

```
(4) ["Claudia", "Cristian", "Erick", "Max"]
```

Demostración - “Método sort”

Ejercicio guiado

Método sort

Se entrega un arreglo de números o con tipos de datos numéricos (number) y solicitan ordenar los números pero en base al primer dígito de menor a mayor. Siendo los elementos del arreglo: “1,5,20,23”.

- **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 sort

- **Paso 2:** En el archivo de JavaScript, plantear el arreglo y crear una variable denominada num para guardar la información. Luego, mediante el uso de la consola mostrar el resultado al aplicar el método sort al arreglo. Quedando de la siguiente manera:

```
var numeros = [1, 5, 20, 23];
console.log(numeros.sort());
```

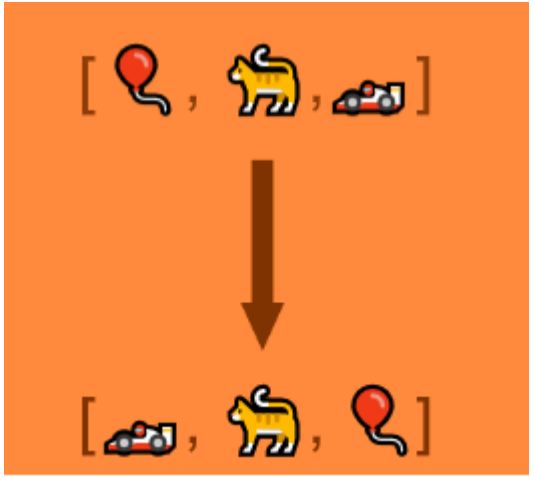
- **Paso 3:** Ejecutar el código anterior, obtendremos la siguiente salida:

```
(4) [1, 20, 23, 5]
```

Métodos para acceder a elementos

Método reverse

- Con reverse, se puede invertir el orden de los elementos dentro de un array



- Veamos cómo usar el método reverse:

```
var muchachos = ["Juan", "Lucas", "Pedro",
"Marcos"];
console.log(muchachos.reverse());
```

```
(4) ["Marcos", "Pedro", "Lucas",
"Juan"]
```

Demostración - “Método reverse”

Ejercicio guiado

Método reverse

Aplicar el método reverse al siguiente arreglo con los elementos: "Charizard, Charmeleon, Charmander", logrando ordenar de forma inversa los datos que contiene el arreglo. Para realizar esto, sigamos los siguientes pasos:

- **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 reverse

- **Paso 2:** En el archivo de JavaScript, plantear el arreglo y crear una variable denominada pokémon para guardar la información sobre los pokemones. Ahora implementar el método reverse para darle un orden inverso a como están presentados actualmente los datos en el arreglo.

```
var pokemon = ["Charizard","Charmeleon","Charmander"];
console.log(pokemon.reverse());
```

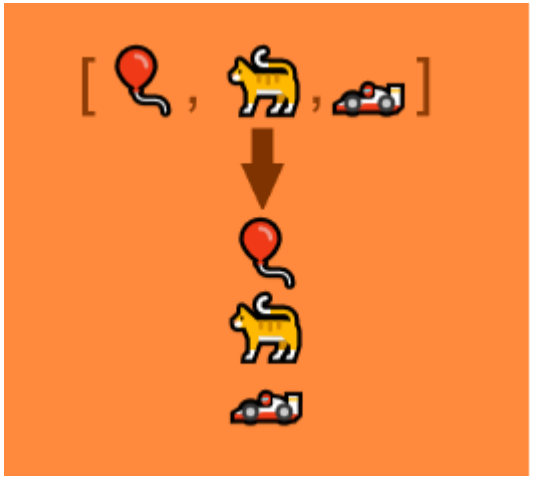
- **Paso 3**: El resultado de lo anterior sería lo siguiente:

```
(3) [ "Charmander", "Charmeleon", "Charizard" ]
```

Métodos para acceder a elementos

Método forEach

- Podemos recorrer cada elemento de un array y realizar alguna acción con respecto a cada uno de éstos



- La estructura básica de un forEach es:

```
arreglo.forEach(function callback(elemento,
index, arreglo) {
          // cuerpo de la función dentro del
iterado forEach
};
```

Demostración - “Método forEach”

Ejercicio guiado

Método forEach

Recorrer un arreglo conformado por varios objetos y mostrar solamente el valor del elemento “nombre” que tiene cada uno de los objetos usando el método forEach. Siendo el arreglo:

```
let clientes = [
  {nombre: 'Juan', edad: 28},
  {nombre: 'Carlos', edad: 22},
  {nombre: 'Karla', edad: 27},
];
```

Ejercicio guiado

Método forEach

- **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:** Accede al método forEach en el array clientes, el cual recibe como parámetro una función que a su vez tiene como parámetro una variable llamada cliente.

```
var clientes = [
  {nombre: 'Juan', edad: 28},
  {nombre: 'Carlos', edad: 22},
  {nombre: 'Karla', edad: 27},
];

clientes.forEach(function(cliente) {
   console.log(cliente.nombre);
});
```

Métodos para acceder a elementos

Método find

- Nos permite obtener un objeto que cumpla alguna condición que especifiquemos. Funciona de forma muy similar al método filter

- La sintaxis básica de este método es:

```
arreglo.find(callback(element[,
index[, array]])[, thisArg])
```

Demostración - “Método find”

Ejercicio guiado

Método find

Para trabajar y practicar la implementación de este método, obtengamos de una lista de productos algún objeto que tenga el nombre kapo implementando el método find. El arreglo con productos es el siguiente:

```
var productos = [
  {nombre: 'coca-cola', precio: 990},
  {nombre: 'papas fritas', precio: 590},
  {nombre: 'ramitas', precio: 290},
  {nombre: 'kapo', precio: 190},
];
```

Ejercicio guiado

Método find

- **Paso 2:** En el archivo script.js, agregar el arreglo indicado en el enunciado del ejercicio, el cual contiene cuatro (4) elementos con una información en específico de cada producto, como el nombre y el precio.

```
var productos = [
  {nombre: 'coca-cola', precio: 990},
  {nombre: 'papas fritas', precio: 590},
  {nombre: 'ramitas', precio: 290},
  {nombre: 'kapo', precio: 190},
];

var kapo = productos.find(function(producto){
   return producto.nombre == 'kapo'
});

console.log(kapo);
```

{desafío} latam_

Ejercicio guiado

Método find

- **Paso 3:** Posteriormente, imprimir el objeto por consola y obtendremos la siguiente salida:

- **Paso 4:** Obtener el producto que cuesta 290 pesos utilizando el método find del mismo arreglo de productos con el que venimos trabajando hasta el momento.

```
{nombre: "kapo", precio: 190}
```

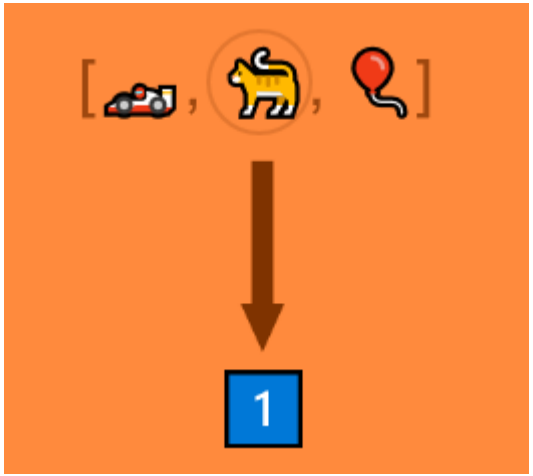
```
var productos = [
  {nombre: 'coca-cola', precio: 990},
  {nombre: 'papas fritas', precio: 590},
  {nombre: 'ramitas', precio: 290},
  {nombre: 'kapo', precio: 190},
];

var kapo = productos.find(function(producto){
   return producto.precio == 290
});
console.log(kapo);
```

```
{nombre: "ramitas", precio: 290}
```

Métodos para acceder a elementos

Método findIndex

- El método findIndex nos permite obtener el índice del objeto que cumpla en primera instancia con alguna condición que declaremos.



- La sintaxis básica del método findIndex es:

```
arreglo.findIndex(callback( element[, index[, array]] )[, thisArg])
```

{desafío} latam_

Demostración - “Método findIndex”

Ejercicio guiado

Método findIndex

El objetivo es recuperar el índice del objeto ramitas, en el arreglo utilizado anteriormente, realizaremos lo siguiente:

```
var productos = [
  {nombre: 'coca-cola', precio: 990},
  {nombre: 'papas fritas', precio: 590},
  {nombre: 'ramitas', precio: 290},
  {nombre: 'kapo', precio: 190},
];
```

- **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 findIndex

- **Paso 2**: En el archivo script.js, agregar el arreglo, el cual contiene cuatro (4) elementos con información en específico de cada producto, como el nombre y el precio. El método findIndex actuará como una rutina de iteración sobre cada elemento presente en el arreglo productos. Si encuentra una coincidencia que cumpla con la condición definida en su interior, retorna el índice donde se encuentre el producto en cuestión y termina su ejecución.

```
var ramitasIndice =
productos.findIndex(function(product
o){
   return producto.nombre ==
'ramitas'
});

console.log(ramitasIndice);
```

```
2
```

Demostración - “Utilizando findIndex para buscar la posición de un elemento”

{desafío} latam_

Ejercicio guiado

Utilizando findIndex para buscar la posición de un elemento

Buscar la posición en que se encuentra Juan dentro del arreglo denominado personas, con los elementos: "pedro, juan, diego".

- **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 el arreglo correspondiente, el cual, se debe iterar para buscar el elemento llamado “juan” y retornar a la posición donde se encuentre.

```
var personas = ["pedro","juan","diego"];

var juanIndice = personas.findIndex(function(persona){
   return persona == 'juan'
});

console.log(persona);
```

{desafío} latam_

Métodos para acceder a elementos

Método some

- Nos permite verificar si algún objeto o elemento dentro de un array cumple con alguna condición que nosotros queramos verificar, es decir, revisar si un arreglo contiene o no un valor.

- La sintaxis básica del método some es:

```
arreglo.some(callback(element[, index[, array]])[, thisArg])
```

Demostración - “Método some”

Ejercicio guiado

Método some

Implementar el método some para saber si existe un objeto del arreglo autos, que contenga algún automóvil que utilice como combustible el Diesel. El arreglo auto será el siguiente:

```
var autos = [
  {marca: 'Toyota', modelo: 'Corolla', combustible: 'Gasolina'},
  {marca: 'Mazda', modelo: '3', combustible: 'Gasolina'},
  {marca: 'Honda', modelo: 'Civic', combustible: 'Gasolina'},
  {marca: 'Bmw', modelo: '116d', combustible: 'Diesel'},
];
```

- **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 some

- **Paso 2:** En el archivo script.js, agregar el arreglo correspondiente de autos indicado en el enunciado del ejercicio, el cual, se debe iterar con el método some para buscar e indicar si existe algún elemento que contenga que utilice como combustible el Diesel, retornando true o false si lo encuentra o no respectivamente.

```
var autos = [
  {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 algunDiesel =
autos.some(function(auto){
   return auto.combustible == 'Diesel'
});

console.log(algunDiesel);
```

```
true
```

Demostración - “Utilizando some para encontrar un número”

{desafío} latam_

Ejercicio guiado

Utilizando some para encontrar un número

Verificar si existe algún número menor de tres en el arreglo denominado números, con los elementos: “1,2,3,4,5,6,7”. Resolvamos el ejercicio:

- **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 el arreglo correspondiente con los números indicados, luego, mediante el método some, apliquemos la función con un retorno solo para números menores que 3.

```
var numeros = [1,2,3,4,5,6,7];

var algunMenorTres =
numeros.some(function(num){
   return num < 3
});
console.log(algunMenorTres);
```

```
true
```

/* Destructuring y los operadores Rest y Spread */

{desafío}
latam_

Destructuring y los operadores Rest y Spread

- Además de los métodos disponibles en los prototipos de los arreglos y objetos, también disponemos del destructuring, el cual es un instrumento disponible en JavaScript para la recolección de elementos contenidos en un objeto o un arreglo.

- La sintaxis consiste en crear las variables ocupando la misma simbología del tipo de dato del cual queremos obtener elementos.

Demostración - “Utilizando el destructuring para obtener un valor”

Ejercicio guiado

Utilizando el destructuring para obtener un valor

Según el siguiente arreglo de objetos:

```
var cursos = [ { nombre: "JavaScript"}, { nombre: "Python"} ]
```

Imprimir por consola el nombre del segundo curso.

Ejercicio guiado

Utilizando el destructuring para obtener un valor

- **Paso 1:** Usar el destructuring para crear 2 variables que representen el primer y segundo curso.

```
var [ primerCurso, segundoCurso ] = cursos
```

- **Paso 2**: Usar nuevamente el destructuring para crear una variable con la propiedad nombre del segundoCurso y posteriormente imprimirlo por consola.

```
var { nombre } = segundoCurso
console.log(nombre)
```

Resumen

- Hemos podido comprender y experimentar los distintos métodos predefinidos que provee JavaScript para el manejo de arrays.

- Vemos que existen métodos que nos permiten manipular el contenido de los arreglos: agregar, eliminar y modificar datos, además de crear arreglos en base a criterios definidos e información de otras fuentes.

- También aprendimos que existen métodos que nos permiten acceder fácilmente a la información contenida en las colecciones de datos: consultar, ordenar y conocer si un valor está presente.

¿Existe algún concepto que no hayas comprendido?

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

Próxima sesión...

- *Se revisará y desarrollará el material sincrónico que corresponde a un **Desafío evaluado,** con el cual pondrás a prueba tus conocimientos adquiridos.*

{desafío}
latam_

Academia de
talentos digitales