Arrays y Objetos

Objetos

{desafío} latam_

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

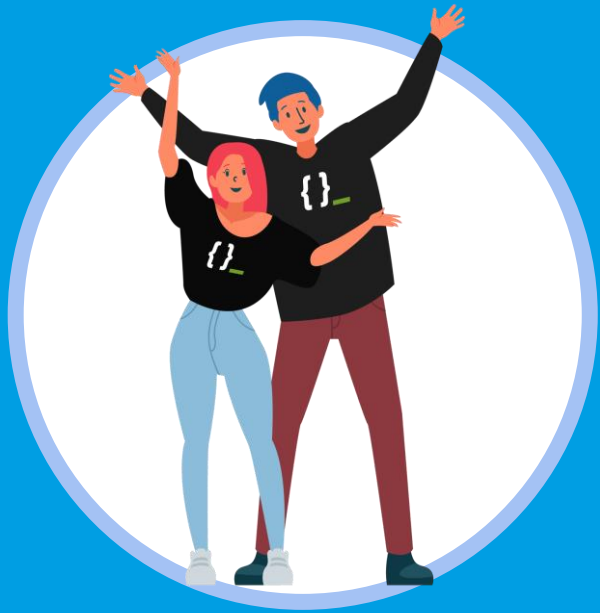
- Unidad 1:
  Introducción al lenguaje JavaScript
- Unidad 2:
  Funciones y Ciclos
- Unidad 3:
  Arrays y Objetos
- Unidad 4:
  APIs

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Identifica las características principales de los objetos en el lenguaje Javascript así como sus objetos preconstruidos más utilizados.*
- *Utiliza el objeto Math para la construcción de un algoritmo que resuelve un problema.*
- *Utiliza el objeto String para la construcción de un algoritmo que resuelve un problema.*

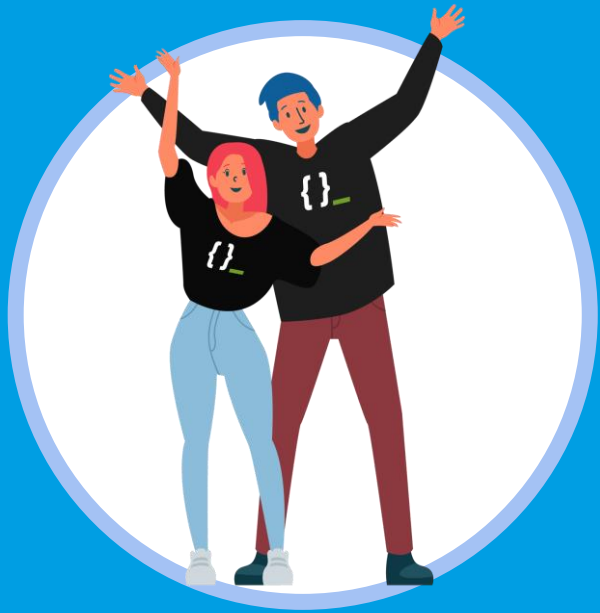
De acuerdo a lo aprendido, ¿cuál es el principal objetivo de los array?



{desafío} latam_

¿Cuál es el resultado del siguiente código?

```
var array = [1,2,3,4,5];
console.log(array[3])
```



{desafío} latam_

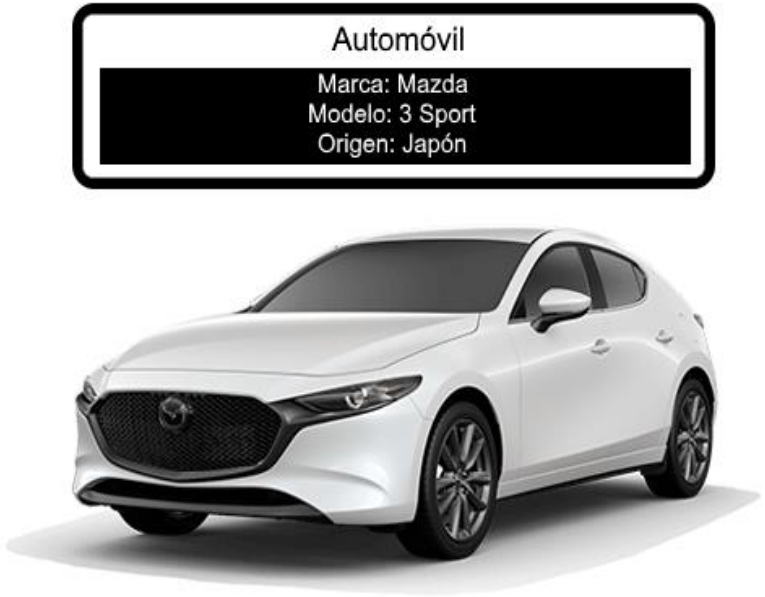
/* Estructura y utilidad de un objeto */

Estructura y utilidad de un objeto

¿Qué es un objeto?

- Los objetos son uno o varios conjuntos de propiedades y/o métodos que están representados mediante la definición de una variable.

- Cada una de estas propiedades o métodos, obedece a una relación **nombre**, **valor** o **clave**, donde el nombre o clave es un puntero o referencia al valor asignado.



Declaración e Inicialización de un objeto

- Un objeto puede ser instanciado definiendo una variable y asignándole los valores requeridos entre llaves, los cuales pueden estar presentes o no en un comienzo

```
var automovil = {};
```

- Si tratamos de imprimir el objeto **automóvil** que acabamos de crear mediante la función **alert**, obtendremos la siguiente salida:

```
[objeto Objeto]
```

Esto quiere decir que acabamos de instanciar nuestro primer objeto. Esta forma de declaración e inicialización de objetos da como resultado un **objeto literal**.

Acceso a las propiedades de un Objeto

- Para acceder a las propiedades de un objeto, podemos realizarlo de 2 maneras distintas: mediante la notación de punto o la notación de corchetes.

- Por ejemplo, volviendo al objeto automóvil que creamos anteriormente y tratar de acceder a la propiedad patente. Esto se puede hacer de la siguiente forma:

    - Notación de puntos: `automovil.patente`

    - Notación de corchetes: `automovil['patente']`

Demostración - “Accediendo a las propiedades de un objeto”

Ejercicio guiado

Accediendo a las propiedades de un objeto

Se solicita mostrar los valores de raza, peso y amigable existentes en el objeto, implementando las notaciones de acceso:

```
let perro = {
   raza: 'Pastor Alemán',
   origen: 'Alemania',
   pelaje: 'Lanudo',
   peso: '33kg',
   edad: 12,
   amigable: true,
   sonidos: function(){
       console.log('El perro ladra');
   }
};
```

{desafío} latam_

Ejercicio guiado

Accediendo a las propiedades de un objeto

- **Paso 1:** Debemos implementar cualquiera de las dos notaciones mostradas al inicio.

```
console.log(perro.raza);
console.log(perro.peso);
console.log(perro.amigable);
```

```
console.log(perro['raza']);
console.log(perro['peso']);
console.log(perro['amigable']);
```

- Ejecuta el código anterior y obtendrás el mismo resultado en ambos casos, por lo que es igual utilizar una notación u otra.

Acceso a las funciones de un objeto

- El acceso a las funciones es similar al cómo se realizan las propiedades de un objeto. Salvo que **es necesario agregar como sufijo paréntesis** para que se ejecute la función.
  - Notación de puntos:

```
automovil.encender();
```

  - Notación de corchetes:

```
automovil['encender']();
```

- Si no se agregan los paréntesis, la función no se ejecuta, sino que solo accede a su contenido, es decir, a su código fuente.

Demostración - “Accediendo a las funciones de un objeto”

{desafío} latam_

Ejercicio guiado

Accediendo a las funciones de un objeto

Acceder a la función que se encuentra en el objeto con el nombre de “perro”, que se muestra en el siguiente bloque de código:

```
let perro = {
   raza: 'Pastor Alemán',
   origen: 'Alemania',
   pelaje: 'Lanudo',
   peso: '33kg',
   edad: 12,
   amigable: true,
   sonidos: function(){
       console.log('El perro ladra');
   }
};
```

{desafío} latam_

Ejercicio guiado

Accediendo a las funciones de un objeto

- **Paso 1:** Primero revisemos la función que se encuentra dentro del objeto, la cual es:

```
sonidos: function(){
   console.log('El perro ladra');
}
```

- **Paso 2:** Si se quiere acceder a esa llave del objeto que lleva por nombre “sonidos”, se puede implementar cualquiera de las notaciones anteriores

```
console.log(perro.sonidos());
console.log(perro['sonidos']());
```

{desafío} latam_

Ejecuta el código anterior obtendrás el mismo resultado para ambos métodos

/* Modificación de propiedades o funciones dentro de un Objeto */

Modificación de propiedades o funciones dentro de un Objeto

- La modificación de elementos dentro de un objeto se hace de manera similar a como lo haríamos con un array.

- Por ejemplo, si deseamos modificar el valor de la propiedad **patente** dentro del objeto **automóvil,** podemos hacerlo de la siguiente forma:

```
automovil.patente = 'JJKX12';
```

- Esta acción sobreescribe el valor de la propiedad sin importar su naturaleza

Demostración - “Modificando el objeto original”

Ejercicio guiado

Modificando el objeto original

Realizar algunas modificaciones en el objeto original, como el peso, la edad y el elemento amigable mediante la notación punto, sobreescribiendo los valores originales por unos valores nuevos. Utilizamos el siguiente objeto:

```
let perro = {
   raza: 'Pastor Alemán',
   origen: 'Alemania',
   pelaje: 'Lanudo',
   peso: '33kg',
   edad: 12,
   amigable: true,
   sonidos: function(){
       console.log('El perro ladra');
   }
};
```

Ejercicio guiado

Modificando el objeto original

- **Paso 1:** Para modificar el peso, la edad y el elemento amigable con nuevos valores, se realizaría de la siguiente manera:

```
perro.peso = '30kg';
perro.edad = 11;
perro.amigable = false;

console.log(perro);
```

Ejercicio guiado

Modificando el objeto original

- **Paso 2:** Utilizamos un console.log para visualizar el objeto nuevamente, obteniendo como resultado:

```
let perro = {
   raza: 'Pastor Alemán',
   origen: 'Alemania',
   pelaje: 'Lanudo',
   peso: '30kg',
   edad: 11,
   amigable: true,
   sonidos: function(){
       console.log('El perro ladra');
   }
};
```

/* Espacio de nombres dentro de un Objeto */

Espacio de nombres dentro de un Objeto

- Las propiedades dentro de un objeto pueden contener a su vez objetos que pueden establecer cierta jerarquía definida por nosotros.

- Esta estructura, formalmente llamada espacio de nombres, puede ser accedida de la siguiente manera. Por ejemplo, si quisiéramos agregar más detalles a la propiedad marca del objeto automóvil:

```
var automovil = {
   marca: {
        nombre: 'Mazda',
        origen: 'Japón'
  },
   modelo: '3 sport',
   patente: 'LJKH63',
   color: 'azul',
   kilometraje: 15000,
   usado: false,
   encender: function(){
        alert('automóvil
encendido.');
  }
};
```

Demostración - “Agregando objetos a un objeto”

{desafío}
latam_

Ejercicio guiado

Agregando objetos a un objeto

Utilizar el siguiente objeto perro, al cual hemos agregado algunas propiedades como propietario y lugar.

```
let perro = {
   propietario: {
    nombre: 'Juan',
    edad: 34,
    lugar: {
      pais: 'Chile',
      ciudad: 'Santiago de Chile'
    },
   },
   raza: 'Pastor Alemán',
   origen: 'Alemania',
   pelaje: 'Lanudo',
   peso: '30kg',
   edad: 11,
   amigable: true,
   sonidos: function(){
       console.log('El perro
ladra');
   }
};
```

Ejercicio guiado

Agregando objetos a un objeto

- **Paso 1:** Para poder acceder a los valores indicados, se utilizará la notación mediante punto y se mostrará el resultado con un console.log, por lo tanto:

```
console.log(perro.propietario.nombre);
console.log(perro.propietario.lugar.pais);
```

- **Paso 2:** Ahora implementamos la notación mediante el uso de corchetes, mostrando el resultado a través de un console.log, eso se logra de la siguiente manera:

```
console.log(perro['propietario']['nombre']);
console.log(perro['propietario']['lugar']['pais']);
```

/* Uso del operador this dentro de un Objeto */

Uso del operador this dentro de un Objeto

- Un error muy común es escribir variables con el mismo nombre una y otra vez dentro de contextos distintos, lo que induce a confusiones. Por esto, existe un operador llamado **this.**

- Esta palabra reservada, **nos permite acceder a elementos que existen sólo dentro del contexto en el cual estamos trabajando.**

```
this.elemento.elemento;
```

Uso del operador this dentro de un Objeto

- Para comprender este concepto, modificaremos el objeto automóvil que creamos y utilizamos anteriormente con los siguientes valores:

```
let automovil = {
   marca: {
       nombre: 'Mazda',
       origen: 'Japón'
  },
   modelo: '3 sport',
   patente: 'LJKH63',
   color: 'azul',
   kilometraje: 15000,
   usado: false,
   encender: function(){
       alert('automóvil ' +
this.marca.nombre + ' ' + this.modelo + '
encendido.');
  }
};
```

Demostración - “Utilizando el operador this”

{desafío}
latam_

Ejercicio guiado

Utilizando el operador this

Crear una función dentro del objeto que muestre los valores de raza, edad y el nombre del propietario. La función debe ser parte de un elemento denominado “datos” y se debe mostrar el siguiente mensaje: “La raza del perro es pastor Alemán, tiene una edad de 11 años y el propietario es Juan”. El objeto a trabajar es:

```
let perro = {
   propietario: {
    nombre: 'Juan',
    edad: 34,
    lugar: {
      pais: 'Chile',
      ciudad: 'Santiago de
Chile'
    },
   },
   raza: 'Pastor Alemán',
   origen: 'Alemania',
   pelaje: 'Lanudo',
   peso: '30kg',
   edad: 11,
   amigable: true,
};
```

Ejercicio guiado

Utilizando el operador this

- **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, utilizar el objeto facilitado en el enunciado y realizar las modificaciones correspondientes para agregar la función dentro del objeto. En este caso, el nombre del elemento que contendrá la función será “datos”, donde pasaremos una función y dentro de ella el mensaje en un console.log para mostrar en el navegador web el resultado:

```
datos: function(){ console.log(``) } };
```

{desafío} latam_

Ejercicio guiado

Utilizando el operador this

- **Paso 3:** Agregar el elemento creado dentro del objeto con el mensaje en el console.log, implementando el operador this para hacer la referencia a los elementos del objeto que deseamos mostrar dentro de esa función:

```
let perro = {
   propietario: {
    nombre: 'Juan',
    edad: 34,
    lugar: {
      pais: 'Chile',
      ciudad: 'Santiago de Chile'
    },
   },
   raza: 'Pastor Alemán',
   origen: 'Alemania',
   pelaje: 'Lanudo',
   peso: '30kg',
   edad: 11,
   amigable: true,
   datos: function(){
       console.log(`La raza del perro es
${this.raza}, tiene una edad de
${this.edad} años y el propietario es
${this.propietario.nombre}`);
   }
};
```

Ejercicio guiado

Utilizando el operador this

- **Paso 4**: Utilizar la notación de punto, llamemos al elemento que contiene la función para poder ejecutarla y se muestra el resultado:

```
perro.datos();
```

- **Paso 5:** Ejecutar el código anterior en el navegador web y el resultado debería ser:

```
La raza del perro es Pastor Alemán, tiene una edad de 11
años y el propietario es Juan.
```

Demostración - “Integrando HTML”

Ejercicio guiado

Integrando HTML

Para profundizar en las bondades de utilizar e iterar sobre colecciones de datos. Se solicita recorrer un objeto y mostrar en una tabla un documento HTML. Lo que debemos mostrar, será la raza, origen, pelaje, peso y edad del arreglo perros:

```
let perros = [{
   raza: 'Pastor Alemán',
   origen: 'Alemania',
   pelaje: 'Lanudo',
   peso: '33kg',
   edad: 12
},
{
   raza: 'Poodle',
   origen: 'Francia',
   pelaje: 'Lanudo',
   peso: '20kg',
   edad: 14
},
];
```

Ejercicio guiado

Integrando HTML

- **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. En el archivo index.html utilizaremos la estructura básica del documento más una tabla, que nos permitirá agregar dentro la información del objeto:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-
width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible"
content="ie=edge">
<title>Integrando HTML</title>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
</style>
</head>
<body>
<table id="cuerpo-tabla">
</table>

<script src="script.js"></script>
</body>
</html>
</html>
```

Ejercicio guiado

Integrando HTML

- **Paso 2:** Dentro del archivo script.js, declarar el objeto y una variable con las cabeceras de la tabla:

```
let perros = [{
   raza: 'Pastor Alemán',
   origen: 'Alemania',
   pelaje: 'Lanudo',
   peso: '33kg',
   edad: 12
},
{
   raza: 'Poodle',
   origen: 'Francia',
   pelaje: 'Lanudo',
   peso: '20kg',
   edad: 14
},
];

  var texto =
"<tr><th>Raza</th><th>Origen</
th><th>Pelaje</th><th>Peso</th
><th>Edad</th></tr>";
```

Ejercicio guiado

Integrando HTML

- **Paso 3**: Dentro del archivo script.js, declarar el objeto y generar la lógica para recorrer los elementos raza, origen, pelaje, peso y edad:

```
  var texto =
"<tr><th>Raza</th><th>Origen</th><th>Pelaje</th><th>Peso</th><th>Edad</th></t
r>";

    for (var i = 0; i < perros.length; i++) {
    }
```

Ejercicio guiado

Integrando HTML

- **Paso 4:** Generar una estructura de tablas que almacene la fila del elemento en la posición [i] en la variable texto:

```
  var texto =
"<tr><th>Raza</th><th>Origen</th><th>Pelaje</th><th>Peso</th><th>Edad</th
></tr>";

    for (var i = 0; i < perros.length; i++) {
        texto += `<tr>
                <td>${perros[i].raza}</td>
                <td>${perros[i].origen}</td>
                <td>${perros[i].pelaje}</td>
                <td>${perros[i].peso}</td>
                <td>${perros[i].edad}</td>
            </tr>`;
    }
```

Ejercicio guiado

Integrando HTML

- **Paso 5:** Imprimir la información en el HTML, agregando un id cuerpo-tabla donde se cargará el texto:

```
  var texto =
"<tr><th>Raza</th><th>Origen</th><th>Pelaje</th><th>Peso</th><th>Edad</th
></tr>";

    for (var i = 0; i < perros.length; i++) {
        texto += `<tr>
                <td>${perros[i].raza}</td>
                <td>${perros[i].origen}</td>
                <td>${perros[i].pelaje}</td>
                <td>${perros[i].peso}</td>
                <td>${perros[i].edad}</td>
            </tr>`;
    }
    document.getElementById("cuerpo-tabla").innerHTML = texto;
```

/* Iterando sobre objetos */

Métodos para recorrer un Objeto

Object.keys

- Crea un array con las propiedades del objeto, es decir, el método “keys” devuelve un objeto del arreglo con las claves de ese arreglo.

- Por ejemplo, si tenemos un objeto con los elementos “manzanas, naranjas y peras” y cada uno de ellos con valores asignados, podemos implementar el Object.keys del objeto para retornar un arreglo con los elementos o propiedades de esos objeto, más no los valores que tiene cada propiedad.

```
var compras = {
 manzana: 2,
 naranjas: 5,
 peras: 10,
};

var keys = Object.keys(compras);
console.log(keys);
```

```
(3) [ "manzana", "naranjas",
"peras" ]
```

Métodos para recorrer un Objeto

Object.values

- El método “values” retorna un arreglo con los valores correspondientes a las propiedades dispuestas en un objeto.
- Las propiedades retornan en el mismo orden a como lo haría un ciclo repetitivo for...in.
- Este objeto construirá un nuevo arreglo pero con los valores de la propiedades

- Si continuamos con el objeto utilizado anteriormente denominado compras, y aplicamos el Object.values a ese objeto:

```
var compras = {
 manzana: 2,
 naranjas: 5,
 peras: 10,
};

var values =
Object.values(compras);
console.log(values);
```

```
(3) [ 2, 5, 10 ]
```

Métodos para recorrer un Objeto

Object.entries

- Devuelve una matriz de pares, en donde el primer elemento del arreglo es la propiedad del objeto y el otro el valor, es decir, creará un arreglo por cada elemento-valor del objeto.

- Agregando el método al objeto, guardando en una variable el resultado y finalmente mostrando esa variable mediante un console.log en el ejemplo anterior, quedaría

```
var compras = {
 manzana: 2,
 naranjas: 5,
 peras: 10,
};

var entries = Object.entries(compras);
console.log(entries);
```

```
[[ "manzana", 2 ],[ "naranjas", 5 ],[
"peras", 10 ]]
```

Demostración - “Métodos para recorrer un objeto”

Ejercicio guiado

Métodos para recorrer un objeto

A continuación vamos a experimentar los métodos vistos para obtener las claves, valores y matriz de pares del objeto perro con el que hemos estado trabajando:

```
let perro = {
   propietario: {
    nombre: 'Juan',
    edad: 34,
    lugar: {
      pais: 'Chile',
      ciudad: 'Santiago de
Chile'
    },
   },
   raza: 'Pastor Alemán',
   origen: 'Alemania',
   pelaje: 'Lanudo',
   peso: '30kg',
   edad: 11,
   amigable: true,
};
```

Ejercicio guiado

Métodos para recorrer un objeto

- Paso 1: En nuestra consola, creamos el arreglo y para obtener las claves, utilizamos el método keys() sobre el objeto 'perro', de la siguiente manera:

```
var keys = Object.keys(perro);
console.log(keys);
```

- Esto retorna un arreglo con las claves que posee el objeto:

```
(7) ["propietario", "raza", "origen", "pelaje", "peso",
"edad", "amigable"]
```

Ejercicio guiado

Métodos para recorrer un objeto

- **Paso 2:** Para obtener los valores, utilizamos el método values() sobre el objeto 'perro', de la siguiente manera:

```
var values = Object.values(perro);
console.log(values);
```

- Esto retorna un arreglo con los valores que posee el objeto:

```
(7) [{…}, "Pastor Alemán", "Alemania", "Lanudo", "30kg", 11, true]
0: {nombre: "Juan", edad: 34, lugar: {…}}
1: "Pastor Alemán"
2: "Alemania"
3: "Lanudo"
4: "30kg"
5: 11
6: true
```

Ejercicio guiado

Métodos para recorrer un objeto

- **Paso 3**: Finalmente, para obtener la matriz de pares: clave, valor, utilizamos el método entries():

```
var entries =
Object.entries(perro);
console.log(entries);
```

- Que retorna la siguiente información:

```
(7) [Array(2), Array(2), Array(2),
Array(2), Array(2), Array(2), Array(2)]
0: (2) ["propietario", {…}]
1: (2) ["raza", "Pastor Alemán"]
2: (2) ["origen", "Alemania"]
3: (2) ["pelaje", "Lanudo"]
4: (2) ["peso", "30kg"]
5: (2) ["edad", 11]
6: (2) ["amigable", true]
```

/* Sets */

Sets

- En ES6 se introdujo el objeto de la clase Set, que es una colección de elementos únicos en un listado, veamos un par de ejemplos de cómo inicializar un objeto de la clase set.

- Existen diferentes maneras de declarar un set:

    - **Declaración de un set vacío:**

```
// set vacio
var set1 = new Set();
console.log(set1);
```

```
Set []
```

Sets

- **Declaración pasando un string con caracteres repetidos:**

```
var set2 = new Set("hooola");
console.log(set2);
```

```
Set(4) [ "h", "o", "l", "a" ]
```

- **Otra manera es pasarle un array al set:**

```
var set3 = new Set([1,2,3,4,5]);
console.log(set3);
```

```
Set(5) [ 1, 2, 3, 4, 5 ]
```

Métodos por defecto que heredan los sets

Método add

- Agrega un elemento a la lista.

```
var set1 = new
Set();

set1.add(1);
set1.add(2);
set1.add(3);
console.log(set1);
```

```
Set(5) [ 1, 2, 3, 4, 5 ]
```

Método delete

- Elimina un elemento a la lista.

```
var set1 = new
Set([1,2,3,4,5]);
set1.delete(5)
console.log(set1);
```

```
Set(4) [ 1, 2, 3, 4 ]
```

Métodos por defecto que heredan los sets

Método forEach

- Recorre la lista

```
var set1 = new Set([1,2,3,4,5]);
set1.forEach(function(item) {
          console.log(item);
});
```

```
1
2
3
4
5
```

Demostración - “Crear un arreglo desde un mensaje”

Ejercicio guiado

Crear un arreglo desde un mensaje

Se solicita crear un nuevo arreglo partiendo del siguiente mensaje: “Arreglo de string”. Luego, eliminar los seis (6) últimos elementos del arreglo formado. Posteriormente, agregar al final del arreglo “simple”. Finalmente, mostrar todo el contenido del arreglo implementando el objeto Set y sus métodos.

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

Ejercicio guiado

Crear un arreglo desde un mensaje

- **Paso 2:** En el archivo script.js, inicializar el arreglo utilizando el objeto set, lo cual sería:

```
let arreglo = new Set("Arreglo de string");
console.log(arreglo);
```

- **Paso 3:** Al ejecutar el código anterior y ver el resultado en la consola del navegador web, el resultado sería:

```
0: "A"
1: "r"
2: "e"
3: "g"
4: "l"
5: "o"
6: " "
7: "d"
8: "s"
9: "t"
10: "i"
11: "n"
```

Ejercicio guiado

Crear un arreglo desde un mensaje

- **Paso 4:** En el resultado anterior, se puede observar como el objeto set elimina todos los caracteres que se encuentren repetidos. Ahora, se procede a eliminar los últimos seis (6) elementos, siendo estos: “ ,d,s,t,i,n”. Aquí se debe implementar el método delete del objeto set.

```
arreglo.delete(" ");
arreglo.delete("d");
arreglo.delete("s");
arreglo.delete("t");
arreglo.delete("i");
arreglo.delete("n");
console.log(arreglo);
```

Ejercicio guiado

Crear un arreglo desde un mensaje

- **Paso 5:** Ejecutar en el navegador web la instrucción anterior, el resultado quedaría:

```
Set(6) [ "A", "r", "e", "g", "l", "o" ]
```

- **Paso 6:** Implementar el método add y agregar la palabra “simple” al arreglo resultante después de eliminar los elementos indicados. Quedando el código de la siguiente manera:

```
arreglo.add("s");
arreglo.add("i");
arreglo.add("m");
arreglo.add("p");
arreglo.add("l");
arreglo.add("e");
console.log(arreglo);
```

Ejercicio guiado

Crear un arreglo desde un mensaje

- **Paso 7:** Obteniendo como resultado al ejecutar el código anterior:

```
Set(10) [ "A", "r", "e", "g", "l", "o", "s", "i", "m", "p" ]
```

- **Paso 8:** Mostrar todo el arreglo con el método forEach:

```
arreglo.forEach(function(elementos) {
        console.log(elementos);
});
```

/* ¿Objeto o Arreglos? */

¿Objetos o Arreglos?

Diferencia	Objeto	Array
Uso	Sirven para definir una "cosa" ej: autos, animales, personas, etc..	Sirven para listar elementos
Declaración vacía	`var obj= {}, var obj = new Obj()`	`var array = []`
Declaración con elementos	`var obj= {a:1,b:2}, var obj = new Obj(1,2)`	`var array = ["1","2",3,4 ]`
Acceso a valores	`obj.a, obj["a"]`	`array[0]` donde 0 es la posición del elemento
Agregar valor	`obj.c = "nuevo valor"`	`array.push("nuevo valor")` o `array.unshift("nuevo valor")`
Eliminar valor	`delete obj.c`	`array.pop()` o `array.shift()`

¿Existe algún concepto que no hayas comprendido?

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

Próxima sesión...

- *Realizaremos en conjunto el material sincrónico que corresponde a un **Desafío guiado**, con el que pondrás en práctica lo aprendido en las sesiones.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales