NODE JS

Colecciones en JavaScript

adn POTENCIÁ TU FUTURO

Nación Servicios

IT Patagonia

Codeki™

Clase 11

01

Introducción a Colecciones

02

Clasificación

Nación Servicios


Introducción a
Colecciones

01

Colecciones

A nivel general, en informática las **Colecciones** son estructuras de datos. Las mismas permiten almacenar y organizar múltiples elementos, generalmente con condiciones en común.

01

Colecciones

Muchas de las **Estructuras** (con este nombre también se conocen a las colecciones, ya que son estructuras de organización de datos) son estándares a lo largo de la inmensa mayoría de tecnologías de programación. Difieren entre sí en cuanto a criterios y formatos, dado así utilidad a distintos casos de uso.

Clase 7

01

Introducción a Colecciones

02

Clasificación

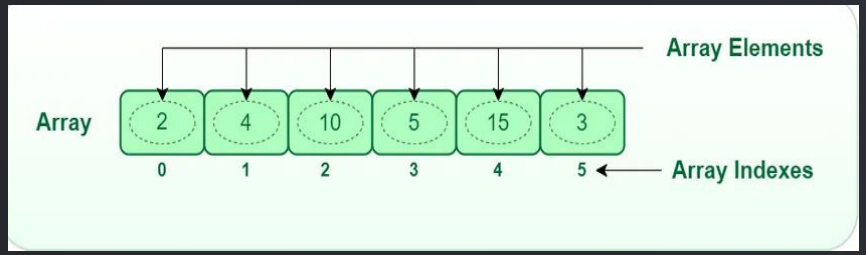
Nación Servicios


Clasificación

02

Arrays (Arreglos)

Un array es una colección ordenada de elementos que se pueden acceder mediante un **índice numérico** (index), comenzando desde cero. Los arrays en JavaScript pueden contener diferentes tipos de datos, como números, cadenas, objetos, entre otros.



02

Arrays (Arreglos)

Para acceder a un determinado elemento del Array, se puede hacer mediante el index. Por otro lado, se encuentra estandarizado, que el agregado de elementos es en orden creciente.

02

Arrays (Arreglos)

Para acceder a un determinado elemento del Array, se puede hacer mediante el index. Por otro lado, se encuentra estandarizado, que el agregado de elementos es en orden creciente.

```
const miArray = [1, 2, 3, "hola", { nombre: "Juan" }];
```

02

Set (Conjunto)

Bajo la definición matemáticamente estricta, de **Conjunto**, dentro del mismo, no pueden existir valores reiterados. Por lo que se elimina la posibilidad de que existan elementos duplicados y el orden está ajustado al orden de inserción, ya que no existe herramienta como el índice.

```
const miSet = new Set([1, 2, 3, 1, 2]);
```

02

Set (Conjunto)

Bajo la definición matemáticamente estricta, de **Conjunto**, dentro del mismo, no pueden existir valores reiterados. Por lo que se elimina la posibilidad de que existan elementos duplicados y el orden está ajustado al orden de inserción, ya que no existe herramienta como el índice.

```
const miSet = new Set([1, 2, 3, 1, 2]);
```

02

Map (Mapa)

Un Map es una colección de **pares clave-valor**. Permite asignar valores a claves y recuperar esos valores más tarde mediante la **clave**. Las claves en un Map pueden ser de cualquier tipo, incluso objetos o funciones.

```
const miMapa = new Map();
miMapa.set("nombre", "Juan");
miMapa.set("edad", 30);
```

02

Colecciones Weak

Para las dos versiones previamente mencionadas, aparecen las colecciones **Weak** (WeakSet, WeakMap,), caracterizadas por solo poder contener Objetos (no permiten valores primitivos) y dichos objetos son **"weakly held"**.
lo que significa que si no hay otras referencias a un objeto, este puede ser recolectado por el recolector de basura.

02

Garbage Collector

Su principal función es administrar la memoria de manera automática para liberar recursos que ya no están siendo utilizados por el programa, evitando así el mal uso de memoria.

Cuando un programa crea objetos o reserva memoria para otras estructuras de datos, es necesario que esa memoria se libere cuando los objetos ya no son necesarios.

Garbage Collector

Su principal función es administrar la memoria de manera automática para liberar recursos que ya no están siendo utilizados por el programa, evitando así el mal uso de memoria.

Cuando un programa crea objetos o reserva memoria para otras estructuras de datos, es necesario que esa memoria se libere cuando los objetos ya no son necesarios.

Garbage Collector

Un objeto se considera "alcanzable" si hay una referencia directa o indirecta desde el código del programa que aún puede acceder a él. Si un objeto ya no es alcanzable desde el código, el Garbage Collector lo marca como "basura" y se encarga de liberar la memoria ocupada por ese objeto para que pueda ser reutilizada en futuras asignaciones.

¡Muchas gracias!