JS | Arrays & Objects

Objetivos:

- ✓ Comprender el término estructura de datos
- ✓ Comprender cómo y por qué las estructuras de datos a menudo están anidadas
- ✓ Acceder a valores desde estructuras profundamente anidadas

¿Qué es una estructura de datos?

Es una forma particular de organizar los datos.

Cuanto mejor podamos organizarlos, mejor podremos representar personas, lugares, objetos, elementos.

Array:

Para referirnos a un elemento en particular, necesitamos hacer referencia al índice del array:

```
const students = [
    "Bob",
    "Susy",
    "Ted",
    "Sarah",
    "Bill"
];

console.log(students[0]);
// Bob
```

Objetos:

Los objetos son otra forma de estructurar nuestros datos. Son buenos para etiquetar datos y construir estructuras más complejas.

```
const bob   = { name: "Bob", age: 17 };
const susy  = { name: "Susy", age: 18 };
const sarah = { name: "Sarah", age: 20 };
const bill  = { name: "Bill", age: 19 };

console.log(bob.name); // <== Bob
console.log(susy.age); // <== 18
```

Podemos acceder a valores específicos haciendo referencia a las claves

Estructuras de datos anidadas

Objects in arrays:

En realidad, una mejor solución para una lista de estudiantes sería un conjunto de objetos. Cada alumno es un objeto y una colección de ellos forma el conjunto de alumnos.

```
const students = [
    { name: "Bob", age: 17 },
    { name: "Susy", age: 18 },
    { name: "Ted", age: 18 },
    { name: "Sarah", age: 20 },
    { name: "Bill", age: 19 }
];

console.log(students[3].name); // <== Sarah
```

Adding To Arrays

Podemos usar el método **.push()** para agregar cosas a los arrays. Esto se aplica también a los objetos en arrays.

PRÁCTICA 1

En el array de estudiantes:

1. Obtenga el valor del nombre del primer alumno
2. Obtenga la edad de la estudiante llamada Sarah

```
const students = [
    { name: "Bob", age: 17 },
    { name: "Susy", age: 18 },
    { name: "Ted", age: 18 },
    { name: "Sarah", age: 20 },
    { name: "Bill", age: 19 }
];
```

Arrays in Arrays

A veces necesitamos tener una estructura de datos para anidar un array dentro de otro array. Esto se llama **two-dimensional array**.

Veamos un ejemplo.

Un ejemplo más complejo: array of arrays containing objects.

Veamos un ejemplo

PRÁCTICA 2

array de clases anterior:

- Recuperar la segunda "classroom" de estudiantes
- Recupere el “firstName” Antonette
- Recuperar la edad 18
- Recupere el apellido "Beatty"

Objetos dentro de objetos

Los objetos dentro de los objetos pueden ser difíciles de manejar. Vamos a crear un objeto classRoom, que tendrá un profesor.

```
const classRoom = {
    teacher:  {
        firstName: 'Greg',
        lastName: 'Dach',
        age: 38
    }
};

console.log(classRoom.teacher.firstName); // <== 'Greg'

console.log(classRoom.teacher.age); // <== 38
```

PRÁCTICA 3

Recupere la edad del profesor del objeto classRoom.

PRÁCTICA 4

- Agregue un estudiante con el nombre de Lucille D. Lozano a Fake School 2, en el primer salón de clases.
- Recupere el objeto "Fake School 3"
- Recupere al profesor con el nombre de "Nathanael"
- Recupere al alumno con el nombre de "Saúl"

Aplicaciones del mundo real

- Bases de datos

Cuando lleguemos a las bases de datos, la mayoría de los objetos de nuestra base de datos estarán en forma de objetos anidados y arrays.

Aplicaciones del mundo real

- API web

Cuando intentemos obtener información en el futuro de las API (servicios web que nos devuelven información), se mostrará de manera similar a nuestros objetos JavaScript.

Por ejemplo:

https://api.punkapi.com/v2/beers/random

Resumiendo: Aprendimos...

- sobre todas las diferentes variedades de estructuras de JavaScript y cómo acceder a los datos dentro de ellas.
- cómo se vería todo eso en las aplicaciones del mundo real.
- que las estructuras de datos nos permiten representar el mundo que nos rodea de manera más efectiva, y serás más eficiente si desarrollas la habilidad para estructurar la información necesaria para desarrollar nuestras aplicaciones con éxito.