Conceptos básicos de lógica

En tu proceso de aprendizaje, ten muy presente los siguientes conceptos para aplicarlos en los ejercicios de lógica de programación o desarrollo.

CONCEPTO	EJEMPLO
Creación de objetos: Podemos crear un objeto usando las llaves {…} con una lista opcional de *propiedades*. Una propiedad es un par “key:value”, donde key es un string (también llamado “nombre clave”), y value puede ser cualquier cosa. (P.D. Para fines prácticos de la lección, nos referiremos a este par de conceptos como “clave:valor”).	`let user = new Object(); // sintaxis de "constructor de objetos"` `let user = {};  // sintaxis de "objeto literal"`
Propiedades de los objetos: Una propiedad tiene una clave (también conocida como “nombre” o “identificador”) antes de los dos puntos ":" y un valor a la derecha. En el objeto user hay dos propiedades: La primera propiedad tiene la clave "name" y el valor "John". La segunda tiene la clave "age" y el valor 30.	`let user = {     // un objeto` `  name: "John",  // En la clave "name" se almacena el valor "John"` `  age: 30        // En la clave "age" se almacena el valor 30` `};`
Métodos de acceso a propiedades de objetos: Se puede acceder a los valores de las propiedades utilizando la notación de punto:	`// Obteniendo los valores de las propiedades del objeto:` `console.log( user.name ); // John` `console.log( user.age ); // 30`
Métodos de acceso a propiedades de objetos: Para acceder a un valor, basta con usar el identificador seguido de corchetes[], dentro de los cuales debe ir la clave que corresponde con el valor que queremos obtener. Por ejemplo, teniendo en cuenta el código anterior, objeto[1] , devolverá el valor uno; objeto[“dos”], devolverá el valor dos; objeto[“tres”] , devolverá el valor tres. Notar que cuando la clave es una referencia o una cadena, siempre se pone dentro de los **corchetes en**	`let objeto = {1: "uno", "dos": "dos", tres: "tres"};` `console.log(objeto[1]);//imprimirá uno` `console.log(objeto["dos"]);//imprimirá dos` `console.log(objeto["tres"]);//imprimirá tres`

comillas, cuando la clave es un entero, no. Esto es:	
Agregado de propiedades a objetos ya creados: Mientras estamos manipulando objetos podemos crear nuevas propiedades para dicho objeto, este agregado de propiedades se veria asi:	`user.isAdmin = true;` `//Al objeto user crearemos un campo llamado isAdmin con el valor de true También lo podríamos hacer con //user["isAdmin"] = true;`
Eliminación de propiedades a objetos ya creados: Para eliminar una propiedad podemos usar el operador delete:	`delete user.age;` `//Eliminamos la propiedad age, del objeto user. También lo podríamos hacer con [] delete user["age"]`
Propiedades de objetos con más de una palabra: También podemos nombrar propiedades con más de una palabra. Pero, de ser así, debemos colocar la clave entre comillas "..." :	`let user = {` `  name: "John",` `  age: 30,` `  "VankVersity student": true  // Las claves con más de una palabra deben ir entre comillas` `};`
Más propiedades de los objetos: El contenido de un objeto puede consistir en variables, funciones o ambos. Las variables que se encuentran en los objetos son propiedades, mientras que las funciones son métodos. Los métodos permiten que los objetos usen las propiedades dentro de ellos para realizar algún tipo de acción.	`const objeto1 = {` `  usuario: "Alex",` `  nacionalidad: "Colombia",` `  profesion: "Ingeniero de Software",` `  miBiografia() {` `  console.log(` `  `Mi nombre es ${this.usuario}. Soy un ${this.profesion} de ${this.nacionalidad}` `);` ` },` `};` `console.log(objeto1.usuario); // Alex`

	`` console.log(objeto1.nacionalidad); // Colombia console.log(objeto1.profesion); // Ingeniero de Software console.log(objeto1.miBiografia()); // Mi nombre es Alex. Soy un Ingeniero de Software de Nigeria ```
método .keys(): El método .keys() de un objeto nos retorna un array con todas las claves de un objeto.	``` let persona = { nombre: "Juan", edad: 30 }; console.log(Object.keys(persona)); // ["nombre", "edad"] ```
método .values(): El método .values() de un objeto nos retorna una array con todos los valores del objeto.	``` let persona = { nombre: "Juan", edad: 30 }; console.log(Object.values(persona)); // ["Juan", 30] ```
método .entries(): El método .entries() de un objeto nos retorna un array de pares `[clave, valor]` del objeto.	``` let persona = { nombre: "Juan", edad: 30 }; console.log(Object.entries(persona)); // [["nombre", "Juan"], ["edad", 30]] ```

Recorrido de objetos con ciclos

FOR-IN

Para recorrer cada clave en un objeto, podemos usar el ciclo for - in. Por ejemplo, para recorrer cada clave en nuestro objeto de ejemplo, se procede así:

```
let objeto = {1: "uno", "dos": "dos", tres: "tres"};
for ( clave in objeto) {
console.log("clave", clave);
console.log("valor", objeto[clave]);
console.log("--------");
}
```

FOREACH

Para usar forEach en un objeto, primero debemos convertir las propiedades o los valores del objeto en un array. Esto se puede hacer con Object.keys(), Object.values(), o Object.entries().

Aquí te muestro cómo hacerlo con Object.entries(), que devuelve un array de pares [clave, valor]:

```
let persona = {
    nombre: "Juan",
    edad: 30,
    profesion: "Ingeniero"
};

Object.entries(persona).forEach(([clave, valor]) => {
    console.log(clave + ": " + valor);
});
```