JavaScript 03 – Objetos en Javascript

Introducción

En Javascript podemos definir cualquier variable como un objeto, existen dos formas para hecerlo:

- Declarándola con **new** (NO se recomienda)
- Forma recomendada: creando un *literal object* usando notación **JSON**.

Ejemplo con *new*:

```
let alumno = new Object();
alumno.nombre = 'Carlos';     // se crea la propiedad 'nombre' y se le asigna un valor
alumno['apellidos'] = 'Pérez Ortiz';    // se crea la propiedad 'apellidos'
alumno.edad = 19;
```

Creando un *literal object* según la forma recomendada, el ejemplo anterior sería:

```
let alumno = {
    nombre: 'Carlos',
    apellidos: 'Pérez Ortiz',
    edad: 19,
};
```

Propiedades de un objeto

Podemos acceder a las propiedades con `.` (punto) o `[ ]`:

```
console.log(alumno.nombre);       // imprime 'Carlos'
console.log(alumno['nombre']);    // imprime 'Carlos'

let prop = 'nombre';
console.log(alumno[prop]);        // imprime 'Carlos'
```

Si intentamos acceder a propiedades que no existen no se produce un error, se devuelve *undefined*:

```
console.log(alumno.ciclo);      // muestra undefined
```

Sin embargo, se genera un error si intentamos acceder a propiedades de algo que no es un objeto:

```
console.log(alumno.ciclo);           // muestra undefined
console.log(alumno.ciclo.descrip);      // se genera un ERROR
```

En versiones anteriores de JavaScript, para evitar este error se comprobaba que existían las propiedades previamente. Veamos un ejemplo:

```
console.log(alumno.ciclo && alumno.ciclo.descrip);
// si alumno.ciclo es un objeto muestra el valor de
// alumno.ciclo.descrip y si no muestra undefined
```

Con ES2020 (ES11) se ha incluido el operador **?.** para evitar tener que comprobar esto nosotros:

```
console.log(alumno.ciclo?.descrip);
// si alumno.ciclo es un objeto muestra el valor de
// alumno.ciclo.descrip y si no muestra undefined
```

Este nuevo operador también puede aplicarse a **arrays**:

```
let alumnos = ['Juan', 'Ana'];
console.log(alumnos?.[0]);
// si alumnos es un array y existe el primer elemento muestra el valor
// si ese elemento no existe muestra undefined
// si no existe el objeto con el nombre alumnos da ERROR
```

Podremos recorrer las propiedades de un objecto con `for..in`:

```
for (let prop in alumno) {
    console.log(prop + ': ' + alumno[prop])
}
```

Resultado:

```
for (let prop in alumno) {
    console.log(prop + ': ' + alumno[prop])
}
```

```
nombre: Carlos
apellidos: Pérez Ortiz
edad: 19
>
```

Una propiedad de un objeto puede ser una función:

```
alumno.getInfo = function() {
    return 'El alumno ' + this.nombre + ' ' + this.apellidos + ' tiene ' + this.edad +
'años'
}

console.log(alumno.getInfo()); //El alumno Carlos Pérez Ortiz tiene 19años
```

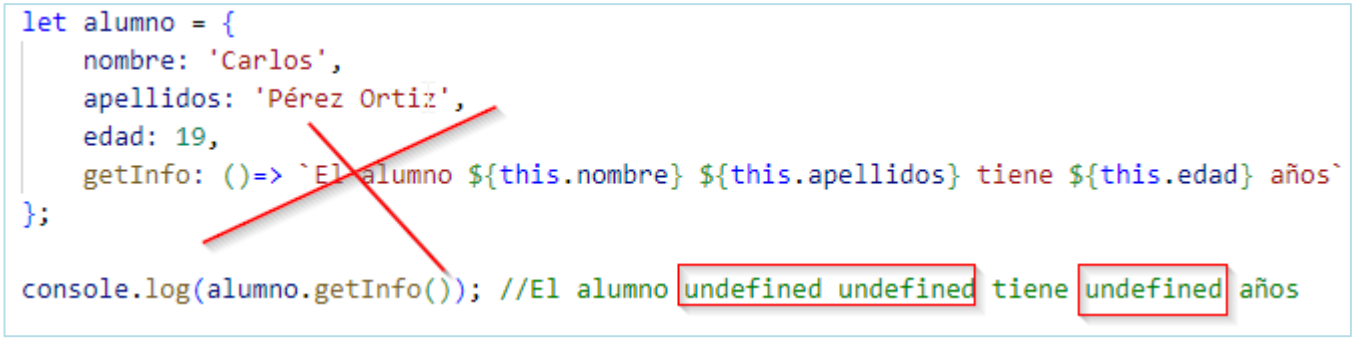
También se puede incluir la declaración del método en la declaración del objeto:

```
let alumno = {
    nombre: 'Carlos',
    apellidos: 'Pérez Ortiz',
    edad: 19,
```

```
    getInfo: function(){
        return `El alumno ${this.nombre} ${this.apellidos} tiene ${this.edad} años`;
    }
};

console.log(alumno.getInfo()); //El alumno Carlos Pérez Ortiz tiene 19 años
```

OJO: deberíamos poder ponerlo con sintaxis *arrow function*, pero no funciona.



No funciona bien porque this tiene distinto valor dependiendo del contexto, y no se puede usar en estos casos con función flecha. Tienes un documento titulado "JavaScript - Anexo - Uso de this en contexto" que lo explica.

Si el valor de una propiedad es el valor de una variable que se llama como ella, desde ES2015 no es necesario ponerlo:

```
let nombre = 'Carlos'

let alumno = {
    nombre,           // es equivalente a nombre: nombre
    apellidos: 'Pérez Ortiz',
    ...
```

EJERCICIO: Crea un objeto llamado tvSamsung con las propiedades **nombre** (TV Samsung 42"), **categoria** (Televisores), **unidades** (4), **precio** (345.95) y con un método llamado **importe** que devuelve el valor total de las unidades (nº de unidades * precio).

Prueba el uso del método con un ejemplo.

Solución:

```
let tvSamsung = {
    nombre: 'TV Samsung 42"',
    categoria: 'televisores',
    unidades: 4,
    precio: 345.95,
    importe: function(){
         return this.unidades * this.precio;
    }
}

console.log(`${tvSamsung.nombre}: ${tvSamsung.importe()} €`);
// TV Samsung 42": 1383.8 €
```