Repaso de Objetos

Los objetos, en JavaScript, son otro tipo de dato. Nos ayudan a representar valores de una forma más fácil, ordenada y agrupada. Los objetos literales se escriben utilizando **{}**.

```
{} // objeto literal
const miObjeto = {}; // objeto asignado a una variable
```

Un objeto tiene propiedades, que se definen con un nombre como las variables. A las propiedades se les puede asignar un valor utilizando dos puntos, y se separan por coma.

```
const persona = {
  nombre: 'Ada',
  edad: 27
};
```

Podemos acceder a las propiedades utilizando el nombre del objeto (variable), punto y el nombre de la propiedad. Si accedemos a una propiedad que no existe vamos a obtener undefined.

```
var persona = {
  nombre: 'Ada',
  edad: 27
};

console.log(persona.nombre); // Ada
console.log(persona.dni); // undefined
```

Para asignar un valor a una propiedad lo hacemos de la misma forma que lo hacemos con una variable. Las propiedades de un objeto terminan siendo variables de las cuales podemos obtener o asignar valores.
Un tema importante con los objetos es que son dinámicos: podemos crear propiedades que no fueron definidas en el objeto.

```
var persona = {
 nombre: 'Ada',
 edad: 27
};

console.log(persona.dni); // undefined


persona.nombre = 'Grace';
```

```
persona.edad = 30;
persona.dni = 90909090;

console.log(persona);
// { nombre: 'Grace', edad: 30, dni: 90909090 }

console.log(persona.dni); // 90909090
```

Spread Operator

El operador de propagación (spread operator) permite que un iterable (como un array o un string) se pueda expandir en un lugar donde se esperan 0 o más argumentos/elementos. También permite que objetos se expanda en un lugar donde 0 o más pares de propiedad-valor se esperan.

¿WAT?

Esta definición, en un principio, es bastante común que sea prácticamente inentendible. Por esto, vamos a dividirlo en partes y mostrar varios ejemplos para entender el funcionamiento del operador.

El **spread operator** se escribe con 3 puntos suspensivos **…**

Vamos a ver un pequeño ejemplo. Supongamos que tenemos la siguiente función **suma**:

```
function suma (a, b, c) {
 return a + b + c;
}
```

Y también tenemos un array con 3 números.

```
const valores = [1, 2, 3];
```

Si nos preguntamos a nosotros mismo, ¿cómo podríamos ejecutar la función pasando esos 3 valores como parámetros? Una posible solución, sería hacer lo siguiente:

```
const resultado = suma(valores[0], valores[1], valores[2]);
```

Funciona. Pero, ¿qué si la función tuviera un número de parámetros variable? o incluso más simple, ¿qué pasa si tenemos un escenario donde esta función espera 20 valores y yo tengo un array de 20 números?

Acá es cuando las cosas se ponen un poco ridículas, ¿vamos a acceder a todos los valores directamente a través del índice?

Consideremos el siguiente ejemplo:

```
function suma (a,b,c,d,e,f,g,h,i,j,k,ohpordio) {
 return a+b+c+d+e+f+g+h+i+j+k+ohpordio;
}


const valores = [1,2,3,4,5,6,7,8,9,11,12];
```

¡Es en estos momentos en donde empieza a brillar el **operador de propagación**!

Imaginemos que el array **valores** de números es una caja que contiene todos estos números en forma muy ordenada, cada uno con un índice.

Cuando accedemos al valor de un índice del array como por ejemplo **valores[1]**, estamos obteniendo únicamente ese valor, que es lo que hicimos en el primer ejemplo. Continuando con la misma idea, ¿qué pasaría si quisiéramos obtener TODOS los valores en el orden exacto en que están en la caja?

Tal vez podría ser muy bueno hacer algo como **valores[0-11]**, pero en lugar de eso lo escribimos así **...valores**.

Entonces en este caso de la función en particular, podemos escribir:

```
const resultado = suma(...valores);
```

Cuando sacamos los valores del array, no sólo los estamos lanzando así nomás al viento, sino que se posicionan en sus respectivos parámetros de la función. **a** será 1, b será 2 y así sucesivamente.

Casos de uso comunes

Si bien el ejemplo visto antes está muy bien, no es lo más usado.

Concatenar arrays:

```
const positivos = [1, 2, 3, 4, 5];
const negativos = [-5, -4, -3, -2, -1];


const todosLosNumeros = [...negativos, ...positivos];
```

Lo primero que hacemos es definir dos arrays **positivos** y **negativos**. Pero miremos un poco más en detalle la 3ra línea.

Vamos a guardar en la variable **todosLosNumeros** el resultado de concatenar los otros dos arrays.
Lo que estamos diciendo en este ejemplo es: JS, agarrá el contenido de lo que tenés en ambos arrays y tiralo en la nueva caja llamada **todosLosNumeros**. El orden de los elementos se va a respetar.

El resultado final es:

```
[-5, -4, -3, -2, -1, 1, 2, 3, 4, 5]
```

Copiar arrays u objetos:

Otro gran uso que se hace del **spread operator**, es para copiar arrays y/o objetos. Cuando en JS estamos asignando a una variable un array o un objeto, lo que estamos haciendo es guardar algo llamado “puntero”, que sería algo como la dirección donde este objeto/array se guardó.

Entonces, si hacemos:

```
const playlist = ['Uno Entre 1000', 'Caravana'];
const copiaPlaylist = playlist;
```

Y queremos agregar una nueva canción en la copia, pero no en el array original:

```
copiaPlaylist.push('La Excepción');
```

Se va a agregar en ambos, ya que la variable “apunta” al mismo array.

```
console.log(playlist); // ['Uno Entre 1000', 'Caravana', 'La Excepción']
console.log(copiaPlaylist); // ['Uno Entre 1000', 'Caravana', 'La Excepción']
```

Entonces, ¿cómo hacemos la copia con el **spread operator**?

```
const playlist = ['Uno Entre 1000', 'Caravana'];
const copiaPlaylist = [...playlist];

copiaPlaylist.push('La Excepción');

console.log(playlist); // ['Uno Entre 1000', 'Caravana']
console.log(copiaPlaylist); // ['Uno Entre 1000', 'Caravana', 'La Excepción']
```

Esto mismo, también funciona con objetos.

```
const persona = {
 nombre: 'Dave Grohl',
 edad: 50
};

const copiaPersona = { ...persona };

copiaPersona.altura = 183;

console.log(persona); // { nombre: 'Dave Grohl', edad: 50 }
console.log(copiaPersona); // { nombre: 'Dave Grohl', edad: 50, altura: 183 }
```

Extender/concatenar objetos:

Otro caso de uso para el **spread operator**, es la posibilidad de combinar objetos para crear uno nuevo. Esto permite que cuando estemos definiendo un nuevo objeto literal, vamos a poder agregar/copiar propiedades de otro objeto.

```
const persona = {
 nombre: 'Dave Grohl',
 edad: 50
};

const copiaPersona = {
 altura: 183,
 ...persona // copia la propiedades del objeto persona
};

console.log(copiaPersona); // { nombre: 'Dave Grohl', edad: 50, altura: 183 }
```

Funcionalidades con Math:

Math es una variable global que ya viene con JS, y nos permite realizar diferentes operaciones. Dos funciones comúnes son **Math.max()** y **Math.min()**. Con el spread operator vamos a poder propagar los diferentes valores dentro de un array en cada parámetro de las funciones (ya que ninguna de las dos acepta como parámetro de entrada un array).

```
const numbers = [9, 4, 7, 1];

console.log( Math.min(...numbers) ); // 1
```

```
console.log( Math.max(...numbers) ); // 9
```