Coerción de Tipos 1

También conocida como coerción de datos. La coerción de tipos es un comportamiento muy importante que posee Javascript y que hace referencia al proceso mediante el cual JavaScript convierte el valor de una variable de un tipo a otro, normalmente de forma automática (sin que el programador lo indique).

Este proceso se realiza al momento de ejecutar una operación donde uno de los operandos contiene un valor de un cierto tipo de datos, pero el otro operando posee un valor de otro tipo.

Por medio de la coerción de tipos, JavaScript permite operar con valores sin importar si son de distintos tipos de datos. Los casos más comunes son:

- Sumar el valor de una variable de tipo `number` con un valor `string`.
- Comparar un valor de tipo `boolean` con otro de tipo `number`.
- Concatenar un `number` con un `string` y formar una nueva cadena de texto.

Existen dos tipos de coerción: Implícita y Explícita

1.1 Coerción Implícita

Es la coerción que se aplica de forma automática cuando se intenta ejecutar una operación con dos valores de distintos tipos. En este caso, JavaScript intenta interpretar los valores y convertir uno de ellos al tipo de dato del otro valor, para que la operación se pueda llevar a cabo.

Este método de coerción de datos es directo e imperceptible, porque que no hay que agregar ningún código extra o llamar a ninguna función.

Destacar que este tipo puede provocar posibles comportamientos no deseados en el código si no se hace con cuidado.

Ejemplo:

```
console.log(4 / "2"); // 2
console.log(10 + "5"); // 105
console.log(10 * ""); // 0
console.log(10 + 4 + "dos"); // 14dos
console.log(false + true); // 1
console.log(true && "0"); // 0
console.log("hola" || 2); // hola
console.log(false || null); // null
console.log("2"+"2"-"2") // 20

0 == "0" // true
0 = [] // true
"0" == [] // false
```

En este tipo de coerción son los aperadores los que aplican el comportamiento descrito. Para ello cada operador tiene una serie de reglas de comportamiento en caso de encontrarse con operandos de diferente tipo

1.1.1 Operador suma (+)

Si uno de los valores es de tipo texto y el otro no, se aplicará coerción implícita sobre el tipo que no es cadena para devolver una cadena como resultado.

```
"1" + true // "1true"
10 + "texto" + 5 // "10texto5"
10 + "" // "10"
```

Por otro lado, si ninguno de los operandos es una cadena, el operador va a intentar realizar la suma numérica (transformando en número todo aquel operando que no lo sea)

```
true + false // 1
true + true // 2
```

1.1.2 Operadores lógicos (`||`, `&&`, `!`)

Si se usa uno de estos operadores, Javascript va a intentar aplicar coerción implícita de forma que todos los operandos sean booleanos.

```
true && false // false
false || !false // true
!2 // false
"texto" || 0 // "texto"
2 || "prueba" // 2
```

Importante destacar lo siguiente:

- Cualquier número que no sea el 0, su equivalente booleano es `true`.
- Cualquier cadena que no sea la cadena vacía (`""`), será `true`
- `||` y `&&` realizan la conversión booleana internamente pero devuelven el valor original de los operandos.

1.1.3 Operadores de igualdad

Cuando se usa el operador de igualdad simple ==, JavaScipt aplica coerción de datos si es necesario y luego, cuando ambos valores son del mismo tipo, efectúa la comparación entre ambos. Lo mismo pasa con el operador de diferencia simple `!=`

En cambio, el operador de igualdad estricta === no aplica coerción de datos y directamente compara los valores. Obviamente, en este caso, si los operandos son de distinto tipo, la comparación fallará.

```
"1" == 1 // true
100 != "100" // false
"1" === 1 // false
```

1.1.4 Operadores Aritméticos

Serían: `+,-,*,/,%,<,>,<=,>=`

Cuando se usan estos operadores la coerción que se aplica trata de convertir los operandos a valores numéricos.

```
"3. " * 2 // 6
"5;" - 1 // NaN
10 / null // Infinity
"10" > 5 // true
true <= 10 // true
```

1.2 Coerción Explícita

En este tipo de coerción el programador indica explícitamente, usando ciertas funciones provistas por JavaScript, a qué tipo de dato se desea convertir un valor.

Esto es similar al `casting` o *"casteo"* explicito que existe en otros lenguajes como JAVA.

```
Boolean(-0) // false
Number("10.   ") // 10
Number("") // 0
String(10) // "10"
```

En este tipo de coerción es necesario usar funciones que conviertan de un tipo a otro:

- `Number(op)`
- `String(op)`
- `Boolean(op)`