Actividad Conversión de tipos de datos *APUNTES*

Realiza todos los ejemplos y comenta cuáles te han resultado más interesantes así como en cuáles de ellos se te han presentado mayores dificultades

---

Pequeña introducción

Nos dan un pequeño ejemplo de los tipos de datos que hay en JS;

- Hay 5 tipos de datos que pueden contener valores:

string -> cadena number -> numero boolean -> booleano object -> objeto function -> funcion

- Dentro de los objetos hay 6 tipos:

Object -> Objeto Date -> Fecha Array -> Array String -> Cadena Number -> Numero Boolean -> Booleano

- Y tenemos 2 tipos que no pueden contener valores:

null undefined

Ejemplo 1 The typeof Operator

Podemos usar el operador typeof, el cual nos devuelve el tipo de dato que le pasamos:

```
typeof "John"                 // Devuelve "string"
typeof 3.14                   // Devuelve "number"
typeof NaN                    // Devuelve "number"
typeof false                  // Devuelve "boolean"
typeof [1,2,3,4]              // Devuelve "object"
typeof {name:'John', age:34}  // Devuelve "object"
typeof new Date()             // Devuelve "object"
typeof function () {}         // Devuelve "function"
typeof myCar                  // Devuelve "undefined"
typeof null                   // Devuelve "object"
```

Curiosidades: El tipo de dato de NaN es número. El tipo de dato de array es objeto. El tipo de dato de date es objeto. El tipo de dato de null es objeto. El tipo de dato de una varibale indefinida es undefined. ***No se puede usar typeof para determinar si un objeto JS es un aray o fecha***

typeof siempre devuelve string

Ejemplo 2 The constructor Property

La propiedad del constructor devuelve la función del constructor para todas las variables de JS.

```
"John".constructor                 // Devuelve function String()  {[native
code]}
(3.14).constructor                 // Devuelve function Number()  {[native
code]}
false.constructor                  // Devuelve function Boolean() {[native
code]}
[1,2,3,4].constructor              // Devuelve function Array()   {[native
code]}
{name:'John',age:34}.constructor   // Devuelve function Object()  {[native
code]}
new Date().constructor             // Devuelve function Date()    {[native
code]}
function () {}.constructor         // Devuelve function Function(){[native
code]}
```

Se puede comprobar la propiedad del constructor para averiguar si un object es un array (contiene la palabra "Array"):

```
function isArray(myArray) {
  return myArray.constructor.toString().indexOf("Array") > -1;
}
```

O incluso más simple, puede verificar si el object es una función Array:

```
function isArray(myArray) {
  return myArray.constructor === Array;
}
```

Puede comprobar la propiedad del constructor para averiguar si un object es una fecha (contiene la palabra "Fecha"):

```
function isDate(myDate) {
  return myDate.constructor.toString().indexOf("Date") > -1;
}
```

O incluso más simple, puede verificar si el object es una funciónd e fecha:

```
function isDate(myDate) {
  return myDate.constructor === Date;
}
```

JavaScript Type Conversion

Las variables de JS se pueden convertir en una nueva variable y otro tipo de datos:

- Mediante el uso de una función de JS.
- Automáticamente por JS mismo.

Ejemplo 3 Converting Numbers to Strings

El método global `String()` puede convertir números en cadenas.

Se puede utilizar en cualquier tipo de números, literales, variables o expresiones:

```
String(x)         // Devuelve una cadena de una variable numerica X
String(123)       // Devuelve una cadena de un número literal 123
String(100 + 23)  // Devuelve una cadena de un número de una expresión
```

El método `toString()` hace lo mismo.

```
x.toString()
(123).toString()
(100 + 23).toString()
```

En el capítulo Métodos numéricos encontrarás más métodos que se pueden utilizar para convertir números en cadenas:

Método	Descripción
`toExponential()`	Devuelve una cadena, con un número redondeado y escrito en notación exponencial.
`toFixed()`	Devuelve una cadena, con un número redondeado y escrito con un número específico de decimales.
`toPrecision()`	Devuelve una cadena, con un número escrito con una longitud especificada.

Ejemplo 4 Converting Booleans to Strings

El método global `String()` puede convertir booleanos en cadenas.

```
String(false)      // Devuelve "false"
String(true)       // Devuelve "true"
```

El método booleano `toString()` hace lo mismo.

```
false.toString()   // Devuelve "false"
true.toString()    // Devuelve "true"
```

Ejemplo 5 Converting Dates to Strings

El método global `String()` puede convertir fechas en cadenas.

```
String(Date())  // Devuelve "Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe
Daylight Time)"
```

El método Date `toString()` hace lo mismo.

```
Date().toString()  // Devuelve "Thu Jul 17 2014 15:38:19 GMT+0200 (W.
Europe Daylight Time)"
```

En el capítulo Métodos de fecha encontrarás más mñetodos que se pueden usar para convertir fechas en cadenas.

Método	Descripción
`getDate()`	Obtenga el día como un número (1-31)
`getDay()`	Obtener el día de la semana un número (0-6)
`getFullYear()`	Obtenga el año de cuatro dígitos (aaaa)
`getHours()`	Obtener la hora (0-23)
`getMilliseconds()`	Obtenga los milisegundos (0-999)
`getMinutes()`	Obtenga los minutos (0-59)
`getMonth()`	Obtener el mes (0-11)
`getSeconds()`	Obtener los segundos (0-59)
`getTime()`	Obtener el tiempo (milisegundos desde el 1 de enero de 1970)

Ejemplo 6 Converting Strings to Numbers

El método global `Numer()` puede convertir cadenas en números.

Las cadenas vacías se convierten en 0.

Cualcuier cosa se convierte en NaN (no es un número).

```
Number("3.14")    // Devuelve 3.14
Number(" ")       // Devuelve 0
Number("")        // Devuelve 0
Number("99 88")   // Devuelve NaN
```

En el capítulo Métodos numéricos, encontrará más métodos que se pueden utilizar para convertir cadenas en números:

Método	Descripción
`parseFloat()`	Analiza una cadena y devuelve un número de punto flotante
`parseInt()`	Analiza una cadena y devuelve un entero

Ejemplo 7 The Unary + Operator

El operador unario + se puede utilizar para convertir una variable en un número:

```
var y = "5";      // y es una cadena
var x = + y;      // x es un número
```

Si la variable no se puede convertir, seguirá siendo un número, pero con el valor NaN (No es un número):

```
var y = "John";   // y es una cadena
var x = + y;      // x es un número (NaN)
```

Ejemplo 8 Converting Booleans to Numbers

El método global `Number()` también puede convertir booleanos en números.

```
Number(false)     // Devuelve 0
Number(true)      // Devuelve 1
```

Ejemplo 9 Converting Dates to Numbers

El método global `Number()` se puede utilizar para convertir fechas en números.

```
d = new Date();
Number(d)          // Devuelve 1404568027739
```

El método de fecha `getTime()` hace lo mismo.

```
d = new Date();
d.getTime()        // Devuelve 1404568027739
```

Ejemplo 10 Automatic Type Conversion

Cuando JS intenta operar con un tipo de dato "incorrecto", intentará conmvertir el valor en un tipo "correcto".

El resultado no siempre es el esperado:

```
5 + null    // Devuelve 5         porque null se convierte a 0
"5" + null  // Devuelve "5null"   porque null se convierte a "null"
"5" + 2     // Devuelve "52"      porque 2 se convierte a "2"
"5" - 2     // Devuelve 3         porque "5" se convierte a 5
"5" * "2"   // Devuelve 10        porque "5" y "2" se convierten en 5 y 2
```

Ejemplo 11 Automatic String Conversion

JavaScript llama automáticamente a la función `toString()` de la variable cuando intenta "generar" un objeto o una variable:

```
document.getElementById("demo").innerHTML = myVar;

// if myVar = {name:"Fjohn"}  // toString se convierte en "[object Object]"
// if myVar = [1,2,3,4]       // toString se convierte en "1,2,3,4"
// if myVar = new Date()      // toString se convierte en "Fri Jul 18 2014
09:08:55 GMT+0200"
```

Los números y los valores booleanos también se convierten, pero esto no es muy visible:

```
// Si myVar = 123             // toString se convierte en "123"
// Si myVar = true            // toString se convierte en "true"
// Si myVar = false           // toString se convierte en "false"
```

De todos los ejemplos, el más interesante y curioso:

De todos los ejemplos que he visto tomando los apuntes, el más interesante y curioso es la conversión automática *(Ejemplo 10)* ya que cuando se trata de algo sencillo funciona bien, pero al intentar sumar una cadena y un número, a veces acierta y otras concatena el número y la cadena.