BOOTCAMP

Generación T

streambe

Clase 26

Javascript Parte 2

Coercion de datos en JS

Hay otras Funciones muy útiles en JavaScript para manipular datos.

📚 La Coerción de datos se refiere a la acción de forzar a que un dato se comporte como si fuera de otro tipo. Siempre se convertirán en los siguientes tres tipos: String, booleano o numérico.

Coercion de datos en JS

Hay otras Funciones muy útiles en JavaScript para manipular datos.

📚 La Coerción de datos se refiere a la acción de forzar a que un dato se comporte como si fuera de otro tipo. Siempre se convertirán en los siguientes tres tipos: String, booleano o numérico.

🛎️ Recordá: typeof es una Función nativa de JavaScript a la que le podés pasar un Párametro y retornará qué tipo de dato es.

Función parseInt

Una Función que sirve para manipular datos es parseInt. Esta Función convierte un String en un number.

⚠️ Importante: parseInt toma únicamente los números enteros.

Por ejemplo, si le pedimos a un usuario que ingrese su edad, para incrementarla cuando sea su cumpleaños, podríamos escribir:

```
let edad = parseInt(prompt("Ingrese su edad"))
// El usuario ingresará un String con su edad y ParseInt lo convertirá en un número.
let cumpleaños = edad + 1
// Se define una Variable con una operación matemática para saber cuántos años tendrá en un año.
alert("En un año tendrás "+ cumpleaños + " año de edad")
// Le darás un mensaje con la respuesta.
```

Al hacer un parseInt de la Variable edad se hará la operación matemática en vez de una concatenación.

¿Por Qué Usamos parseInt?

Si no usáramos parseInt en este ejemplo, en vez de sumar el valor ingresado por el usuario, se concatenaría el number "1" al String "36". Mirá este ejemplo:

```
let edad= prompt("Ingrese su edad")
undefined
edad
"36"
//String//
undefined
edad= edad+1
"361"
```

En cambio, pasándole al prompt la Función parseInt, convertirá al valor en un number y permitirá la operación matemática. 👩‍💻👨‍💻 ¡Probalo en tu consola!

⚠️ Importante: El signo de suma (+) es el único operador aritmético que tiene la doble función de sumar o concatenar.

Otros Métodos: Number Y parseFloat

Number

Number: Se usa para convertir un String en un número y admite también los decimales.

```
Number("4")
4
// Pasa el String "4" a un número 4.
Number("4.5")
4.5
```

Esta Función sólo lee caracteres numéricos. Si le pasamos otro tipo de caracter como Argumento, nos devolverá la expresión "NaN" (del inglés "Not a Number"; es decir, "No es un Número").
Por ejemplo:

```
Number($4)
NaN
```

parseFloat

parseFloat: Convierte un String en un number y admite los decimales. A diferencia de la Función Number, deja de traducir cuando encuentra un caracter no numérico.
Por ejemplo:

```
parseFloat("4.5$")
4.5
```

⚠️ Importante: Si el caracter no numérico está previo al número, no lo reconocerá porque su parseo terminará en ese momento.

```
parseFloat("$4.5")
NaN
```

Template Literals

Los Template Literals son una nueva forma de crear Strings.

Básicamente, son Strings que permiten expresiones y soportan tanto una línea de código simple como una múltiple.

Los Template Literals permiten escribir texto de varias líneas de una forma más sencilla.
⚠️ Importante: Con las comillas normales no podés poner saltos de línea.

Template Literals

De manera normal

```
var mensaje =
  '<div>\n' +
  '   <p>¡Hola!</p>\n' +
  '</div>'

console.log(mensaje);
```

Con template literals

```
let mensaje = `
  <div>
    <p>¡Hola!</p>
  </div>
`

console.log(mensaje);
```

Interpolación De Strings: Sintaxis

Los Template Literals también nos permiten escribir expresiones conocidas como interpolación de Strings. Esto permite agregar Variables u operaciones dentro de los Strings.

📚 ¿Qué Significa Interpolar?
Interpolar significa poner determinada cosa entre otras que siguen un orden, o dentro del conjunto que estas forman.

Interpolación De Strings: Sintaxis

Para crearlas, usá el signo de moneda ($), abrí una llave ({), escribí la Variable u operación que quieras incluir y cerrá la llave (}):

```
`string ${variable u operación}`
```

```
// de la forma antigüa
var nombre = 'Gabriela'
var saludo = '¡Hola, ' + nombre + '!'
console.log( saludo )
// ¡Hola, Gabriela!
```

```
// de la nueva forma
let nombre = 'Gabriela'
let saludo = `¡Hola, ${nombre}!`
console.log( saludo )
// ¡Hola, Gabriela!
```

Interpolación De Strings: Sintaxis

Quizás en el ejemplo parezca poco ahorrarse solo un par de caracteres. Sin embargo, cuando escribas miles de líneas de código, notarás cuánto más simple es hacerlo de esta manera.

Por último, en las interpolaciones, también podrás usar expresiones de JS.

Por ejemplo:

```
let a = 5
let b = 6
let suma = `La suma es igual a ${a + b}`
console.log(suma)
// La suma es igual a 11
```

GENERACIÓN T

streambe