PROGRAMACIÓN BÁSICA EN JAVASCRIPT

Módulo 2

OTEC PLATAFORMA 5 CHILE

MÓDULO 2

CLASE 1

Introducción

JavaScript

Valores

Coerción de Datos

Introducción

Arrancamos un nuevo módulo y con ello, una nueva series de desafíos de la mano de JavaScript.

Aunque en reiteradas oportunidades le hemos hablado sobre este lenguaje de programación ahora comenzaremos a aprender lo más básico y puro para luego poder desarrollar proyectos más complejos.

Por lo pronto empezará a usar JavaScript para:

- **Vincular un archivo JavaScript con un documento HTML.**
- **Acceder a la consola del navegador para ejecutar código JavaScript.**
- **Reconocer los tipos de datos: Strings, Numbers, Booleans, Null, Undefined, etc.**
- **Distinguir el uso de var, let y const para declarar una Variable.**

Como vimos al principio del curso, si HTML es el esqueleto de la página y CSS es la estética, Javascript es todo lo que le da vida y lo que te permite una interacción real.

Sin JS, podríamos ver una página web, clickear algún link o, incluso, enviar un formulario. Sin embargo, si quisiéramos interactuar con ella de un modo más complejo, tendríamos que usar un lenguaje de scripting (como JavaScript) que le indique al browser que debe pasar, cuándo y cómo.

Un Poco De Historia

JavaScript fue creado a mediados de los '90 por Brendan Eich, un developer de la primera era, que desarrolló los primeros navegadores de internet: *Netscape*.

- Previo a eso, generalmente se usaba HTML y CSS para hacer páginas web, pero eran muy básicas. Por eso, Netscape le encargó a uno de sus ingenieros de software que desarrollara otro lenguaje y, llamativamente, Brendan lo hizo en sólo 10 días. JavaScript, en la actualidad, es uno de los lenguajes más usados en todo el mundo porque es el único que se usa tanto en el front-end como en el back-end (es decir, del lado del servidor, gracias a Node.js).

Front-end y Back-end

El front-end de un sitio es, básicamente, todo lo que el usuario ve y con lo que interactúa. Es decir, es el frente de una web o aplicación.

Por ejemplo, en el caso de una web, es todo lo que vemos o podemos clickear y, en una app, es lo que podemos scrollear, mover o apretar.

El back-end es todo lo que sucede por detrás y que tiene que ver con cómo fluye la información que genera la interacción del usuario. Por lo tanto, es todo lo que el usuario no ve.

Por ejemplo, cuando hacemos una búsqueda en un e-commerce, estamos enviando información a una base de datos. Para eso, hay todo un sistema que recibe ese pedido, lo procesa, hace la búsqueda y da una respuesta.

Gráficamente, nos quedaría algo así...

Si bien como developer puede especializarse solo en front o back-end, la programación full stack te permite trabajar en ambos.

Vinculación de *JavaScript* con HTML

Como vimos anteriormente, la separación de tareas es un aspecto fundamental en la programación. Cada lenguaje cumple una función diferente. Por eso, debemos guardar cada uno en un archivo distinto: todo lo que sea contenido se encontrará en un .html, lo estético estará en un .css y lo funcional en un .js.

Sintaxis

Para vincular un archivo .js con un .html:

Incorpore el tag script justo antes de cerrar el `<body>` y declare su ruta usando el atributo `src`:

```
<script src="myScript.js"></script>
```

Importante: **Cuando el navegador se encuentre con el tag script interpretará todo el contenido cargado previamente de acuerdo a las reglas de JavaScript.**

Valores en JavaScript

JavaScript manipula distintos tipos de datos o valores, con una sintaxis particular para cada uno de ellos. Es decir, deberá prestar atención a cómo escribe su código para no tener malos entendidos con su programa.

Importante: **JavaScript es case sensitive, es decir, las mayúsculas y las minúsculas tienen codificaciones distintas.**

En síntesis, los datos pueden ser primitivos o pueden ser más complejos, como son las Funciones y los Objetos.

Ahora, nos enfocaremos en los valores primitivos:

- **Números:** Funcionan igual que en la matemática. Pueden ser números enteros o racionales. Se pueden sumar, restar, multiplicar o dividir y siguen la lógica de resolución que privilegia las operaciones dentro de los paréntesis para luego resolver el resto.
- **Strings:** Son cadenas de caracteres que incluyen letras, números y espacios. Debemos encerrarlos entre comillas simples (‘’) o dobles ("") para que JS entienda que es texto y no la confunda con una Variable.
- **Booleanos:** Son datos de tipo true (verdadero) o false (falso), es decir, que activan o desactivan cierta parte del programa según el input recibido.
- **Undefined:** Es un valor que posee una Variable que está sin definir en ese momento.
- **Null:** Es un valor que posee una Variable que está explícitamente vacía (y hay una razón para ello).
- **Symbols y BigInts:** Si bien son valores primitivos, son más complejos y no los veremos en este curso.

¿Cuál es la diferencia entre *undefined* y *null*?

Si pensamos en una Variable como una caja, el valor undefined indica que la caja está vacía (hasta que le demos contenido).

En cambio, en el caso de null, el programador le asignó explícitamente el valor para que la caja permanezca vacía.

Función `typeof`

Es una Función nativa de JavaScript a la que le puede pasar un parámetro y retornará qué tipo de dato es: si se trata de un string, null, undefined, etc.

Por ejemplo:

```
typeof(5)
"number"
typeof("5")
"string"
```

Tip: **Esta sentencia sirve para comprobar que las Variables hayan sido definidas.**

Números

- **Funcionan igual que en las matemáticas.**
- **Se usan los operadores matemáticos para sumar, restar, multiplicar y dividir (+, -, *, /).**
- **Se resuelve primero todo lo que esté dentro de los paréntesis ( ). Si hay más de un par de paréntesis, se resuelve la operación desde adentro hacia afuera, por ejemplo: ((3+2)*3)+1=16 vs. 3+2*3+1=10.**
- **El operador de Módulo (%) no se considera como el valor absoluto de un número sino como el resto de una división. Por ejemplo, 3%2=1.**

¿Qué es el Operador Módulo?

En el siguiente link podrá encontrar su definición y un ejemplo para que usted lo pruebe.

Strings

Los Strings son cadenas de caracteres. Para que JS los reconozca como tal, el texto debe escribirse entre comillas dobles ("") o simples (').

Características principales:

- **Los Strings son cadenas de caracteres cuya información puede ser representada como un texto.**
- **Un carácter puede ser tanto un texto, como un espacio o un número, siempre y cuando estén entre comillas:**

```
"Tengo"+" "+"18"
```

- **Pueden concatenarse usando el operador de suma (+).**
- **Los datos pueden coercionarse. Por ejemplo, cuando se concatena un String con un número, este último será forzado a comportarse como un texto:**

```
"Tengo " + 18 + " años"
```

Si bien, el 18 es un valor numérico, JavaScript lo interpretará como un texto: `Tengo 18 años`**.**

¿Qué Significa Concatenar?

Según Mozilla, concatenar es una elegante palabra de la programación que significa "unir".

Si quiere conocer más, acceda a la documentación de Mozilla.

¿Por qué es fundamental el entrecomillado?

Números: Si un número no está entrecomillado, JS lo interpretará como un valor numérico o lo coercionará para que funcione como un texto.

Textos: Si un texto no está entrecomillado, JS lo interpretará como si fuera una Variable o instrucción que puede, o no, estar definida.

Variables

Una Variable es un contenedor que guarda información para, luego, usarla.

Para definir las Variables, hay otras formas de hacerlo con algunas diferencias entre sí:

- `var`**: Es una forma más flexible para crear una Variable ya que nos permite volver a crearla y reemplazar la anterior.**
- `let`**: Permite actualizar una Variable pero no volver a crearla.**
- `const`**: Permite crear una Variable que se mantendrá constante durante todo el programa. Es decir, no se podrá actualizar ni cambiar.**

Para definir una variable, por ejemplo, tenemos que utilizar algunas de las palabras reservadas mencionadas con anterioridad, y luego colocar es signo igual (=) y colocarle el contenido que necesitemos guardar.

Ejemplo:

```
let nombre= "Chapulín";
```

¿Qué significa DECLARAR y DEFINIR?

Cuando se habla de *declarar una Variable, se la está creando*.
Cuando se habla de *definir una Variable, se le está asignando contenido*.

Null y Undefined

Null y Undefined son dos valores que se pueden asignar a las Variables. Si bien ambos significan que no tienen contenido, su diferencia es conceptual.

Por ejemplo, si declaramos una Variable pero no le asignamos un valor, por defecto va estar undefined. Como dice el nombre, la Variable está sin definir, o sea, no tiene un valor por ahora.

En cambio, en null, la Variable está explícitamente vacía.

Built-in Methods

Los Built-in Methods son Funciones pre-armadas de JavaScript para que usemos en nuestros programas.

Veamos cada Función en más detalle.

- **Alert: el alert es una Función que muestra un mensaje al usuario. Antes de hacerlo, evalúa lo que está adentro (por ejemplo, si es una operación matemática, si es un String o si tiene que combinar una Variable con algún otro dato).**
- **Console.log: es una Función que imprime un mensaje en la consola. Si bien es una forma sencilla de evaluar código, su verdadera utilidad se encuentra al incorporarlo desde un archivo.**
- **Prompt: Es una Función que nos devuelve un valor que le pregunta previamente al usuario. Es especialmente útil para tomar un input, combinarlo con una Variable y manipular ese valor. Puede guardarlo, mostrarlo, recuperarlo o combinarlo con otras Variables.**

***Importante:* El programa siempre resolverá primero aquello que esté entre paréntesis.**

¿Qué es una función de JavaScript?

Una Función en JavaScript es un conjunto de instrucciones que realiza una tarea o calcula un valor. Debe tomar alguna entrada y devolver una salida donde haya alguna relación obvia entre ambas.

Coerción De Datos En JavaScript

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.

parseInt: Esta Función convierte un String en un number. Solo toma únicamente los números enteros.

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

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

MÓDULO 2

GRACIAS

Aquí finaliza la clase n°1 del módulo 2

OTEC PLATAFORMA 5 CHILE