JAVASCRIPT

CLASE 1

Material complementario

CONCEPTOS GENERALES: SINTAXIS Y VARIABLES

CODER HOUSE

JavaScript: fundamentos

¿Qué es JavaScript?

JavaScript es un lenguaje de programación que se utiliza principalmente para construir aplicaciones y sitios web dinámicos. Con esta tecnología, podemos desarrollar softwares como los que empleamos a diario: tiendas online, buscadores, gestores de contenido, entre otros.

Se dice que es un lenguaje interpretado porque el código que escribimos se puede ejecutar directamente en el navegador, lo cual significa que los archivos HTML con código JavaScript asociado se interpretan inmediatamente.

JavaScript se utiliza para crear el "cerebro" de la aplicación, lo que permite dotar a tecnologías como HTML y CSS de la interactividad y el dinamismo necesarios para construir soluciones web modernas.

Aplicación Web

La necesidad de usar JavaScript aumenta con la cantidad de funcionalidades que buscamos construir en un software. Es importante tener en cuenta que en la actualidad el desarrollo desde cero de sitios web estáticos, es decir, aquellos cuya información no cambia en respuesta a las acciones del usuario, es poco frecuente.

Lo que se busca construir en el ámbito laboral son plataformas que ofrezcan un alto nivel de interactividad, y un variado número de funcionalidades. Ya no hablamos de sitios, sino de aplicaciones web que permiten realizar tareas importantes a sus usuarios.

Por esta razón, JavaScript se vuelve cada día más importante, ya que es la tecnología más empleada para construir el funcionamiento de "la parte de la aplicación que las personas usan".

Front End y Back End

Con “la parte de la aplicación que las personas usan” nos referimos a todo lo que vemos en el navegador cuando empleamos una plataforma, y el código necesario para que la interfaz funcione. La misma está construida con HTML, CSS y JavaScript, y se la conoce como el Front End.

Por otro lado, toda aplicación web tiene otra sección con la cual el usuario no interactúa directamente, pero es la responsable de dar respuesta a acciones que se realizan en el Front End, almacenando la información que el usuario produce, o solicitando datos para que la aplicación pueda funcionar correctamente. A esta sección se la conoce como Back End, ý la misma puede presentar diversidad en las tecnologías con las que se construye, ya que esta parte de las aplicaciones puede integrar varios lenguajes de programación además de JavaScript.

No obstante, como estamos iniciando en el desarrollo de aplicaciones modernas, a lo largo del curso vamos a centrarnos en la construcción del Front End con JavaScript, ya que es necesario entender primero cómo desarrollar nuestra aplicación en respuesta a las acciones del usuario, para luego poder enfocarnos en cómo debe responder el Back End.

Herramientas

Durante el curso, emplearemos distintas herramientas para construir nuestra aplicación, y podrás escoger una de cada categoría. Utilizaremos el framework CSS, jQuery y el servidor local a partir de la mitad de la cursada; de todos modos, te compartimos el listado para que lo tengas presente:

- Navegador Web: Chrome, Firefox, Edge o Opera.
- Editor de Código Fuente: Visual Studio Code, Sublime Text 3 o Atom.
- Framework CSS: Bootstrap 4, Bootstrap 5, Bulma o Milligram.
- jQuery.
- Servidor Local:Live Server (VS Code), XAMPP , WampServer.

Evolución de JavaScript

Antes de iniciar con la manera de emplear JavaScript, vamos a repasar rápidamente su historia para resaltar el contexto que dio origen a esta tecnología . La primera versión de JavaScript se publicó en 1997, y dado que el lenguaje de programación se basa en el estándar ECMA Script, se denomina comúnmente a las versiones de JavaScript por las siglas ES, acompañadas del número de versión; así, la versión de 1997 se denominó ES1. En cada versión se incluyeron nuevas herramientas en el lenguaje, y aquí nos enfocaremos en las dos más empleadas actualmente: ES5 y ES6.

Es importante entender que los lenguajes de programación se actualizan periódicamente, siendo necesario verificar los cambios entre versión y versión. No obstante, cuando una aplicación se desarrolla con una determinada versión, sólo se actualiza si es estrictamente necesario.

Sintaxis y código

Código JavaScript

JavaScript, como todos los lenguajes de programación, tiene su propia sintaxis, que es el conjunto de reglas que empleamos para escribir el código fuente. En el Front End, el código JS está siempre asociado a un documento HTML, el cual podemos abrir con nuestro navegador.

¿Cómo escribir código JavaScript?

Existen dos formas de asociar código JavaScript a un HTML.:

1. Empleando la etiqueta <script></script> en alguna parte del documento HTML, y escribiendo dentro de ella el código JavaScript. Podemos usar dos barras para tipear una línea de comentario, la cual es una anotación que realiza el programador para orientarse respecto a lo que está codificando (esta línea no es interpretada por el navegador).

2. La segunda forma de incluir código JavaScript en nuestra aplicación es contar

con un archivo .js y asociarlo al documento .html usando la etiqueta <script>, y la ruta del archivo JavaScript. Por ejemplo: <script src="js/main.js"></script>

Sintaxis: reglas básicas

Cuando escribimos código JavaScript, hay que tener en cuenta que no importa la cantidad de espacios en blanco que ingresemos, pero tenemos que tener especial cuidado en respetar las mayúsculas y minúsculas al tipear los elementos a utilizar.Se dice que JavaScript es un lenguaje "case sensitive" porque se distingue entre mayúsculas y minúsculas, es decir que no es lo mismo escribir "UNO" (utilizando mayúsculas) que "uno" (usando sólo minúsculas).

También es posible incluir un bloque de comentarios, que es un comentario compuesto por más de una línea, como un párrafo; mientras que una línea de comentario es una única oración.

Sintaxis: palabras reservadas

Ya sabemos dónde escribimos el código JavaScript, así que ahora vamos a ver cómo lo hacemos. Usamos palabras para escribir las líneas de código, palabras que no determinan el funcionamiento de instrucción (línea de código) que estamos creando. Entonces, para aprender a programar en JavaScript debemos entender el significado de las ***palabras reservadas***, y cómo podemos emplearlas para programar.

Algunos ejemplos de estos términos son: break, case, catch, continue, default, let, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with, etcétera. En el curso aprenderemos a utilizar los más importantes para construir aplicaciones.

Variables y valores

Variables

Vamos a aprender ahora nuestras primeras palabras reservadas. Los términos a continuación se utilizan para crear elementos con valores asociados, a los cuales los conocemos por el nombre de variables.

Para crear una variable podemos usar la palabra "var", seguida por el nombre que quiero especificar para la misma, acompañada de un punto y coma (;), si no vamos a asociar ningún valor al momento de crear la variable.

También existen otras palabras para crear variables desde la versión ES6 de JavaScript, como "let" que, por así decirlo, es equivalente a "var" en la nueva versión, así como "const", la cual se usa para crear una variable cuyo valor no va a cambiar nunca.

Al proceso de "crear" una variable se le dice "declarar" una variable, mientras que cuando "le damos" un valor a la misma, como en el caso de la variable con nombre "LENGUAJE" estamos "asignando" un valor a la variable.

Tipos de valores

Podemos asociar distintos valores a una variable en JavaScript; vamos a empezar trabajando con números y texto:

- Un valor numérico puede ser entero o decimal; cuando es entero se le dice "integer", mientras que se lo denomina "float" cuando es decimal.
- Por otro lado, un texto es un valor compuesto por uno o más caracteres entre comillas simples o dobles, y se lo llama "cadena de caracteres" o "string".

Es importante aclarar que en JavaScript las variables pueden guardar cualquier tipo de valores. Esto significa que si asignar primero un número no impide luego asignar un string a la misma variable. A esta característica en los lenguajes de programación se la llama "tipado dinámico", e implica que las variables pueden guardar cualquier valor, sin que sea necesario especificar qué tipo de dato contiene la variable en la declaración de la misma.

Operaciones básicas con variables numéricas

Si tenemos variables con valores numéricos, podemos hacer operaciones matemáticas entre ellas. Es decir sumarlas, restarlas, multiplicarlas, etcétera. Ejemplo:

```
<script>
    var   numeroA = 1;
    let   numeroB = 2;
    const NUMEROC = 3;
    //Suma  de dos números (1 + 2 = 3)
    let resultadoSuma  = numeroA + numeroB;
    //Resta de dos números (2 - 1 = 1)
    let resultadoResta = numeroB - numeroA;
    //Producto de dos números (2 * 3 = 6)
    let resultadoProducto = numeroB * NUMEROC;
</script>
```

Operaciones básicas con variables string

Si tenemos variables con valores string podemos concatenarlas para crear nuevas cadenas. Es decir, un único texto compuesto por el texto de las variables. Ejemplo:

```
<script>
    var   textoA = "CODER";
    let   textoB = "HOUSE";
    const BLANCO = " ";
    //Concatenar textoA y textoB ("CODER" + "HOUSE" = "CODERHOUSE")
    let resultadoA = textoA + textoB;
    //Concatenar textoB y 1 ("HOUSE" + 1 = "HOUSE1")
    let resultadoB = textoB + 1;
    //Concatenar textoA, BLANCO y textoB ("CODER" + " " + "HOUSE" = "CODER HOUSE")
    let resultadoC = textoA + BLANCO + textoB;
</script>
```

Prompt, consola y alert

Prompt

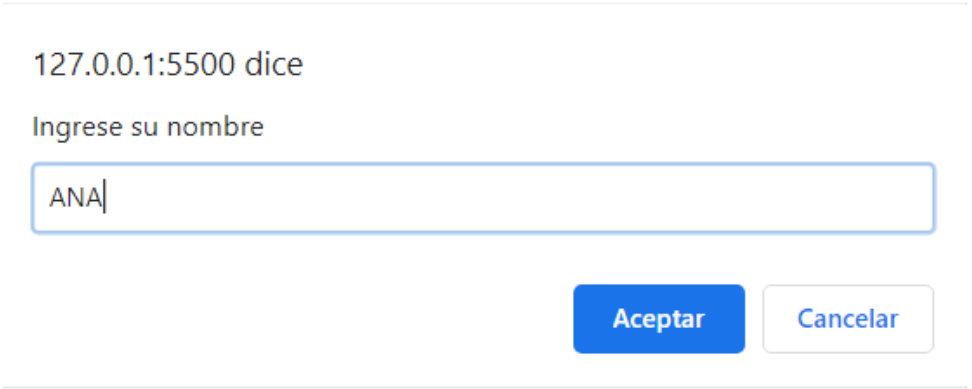
Ahora vamos aprender tres herramientas del lenguaje JavaScript que nos permiten interactuar con el usuario:

- La primera es la instrucción ***prompt()***, con la cual podemos solicitar un dato al usuario. Cuando la usamos, se visualiza en el navegador un cuadro de texto que pide un dato. Se busca asignar el valor ingresado por ***prompt()*** a una variable. Dicho valor que el usuario escribe es siempre una cadena de caracteres, por lo cual es necesario transformarlo a número si queremos hacer cálculos con él.

  Veamos entonces un ejemplo de cómo usar prompt:

```
<script>
    let nombreIngresado = prompt("Ingrese su nombre");
</script>
```

El usuario puede ingresar cualquier valor en este cuadro, y no puede limitarse el tipo de valor ingresado. A los datos que ingresa mediante prompt se los conoce con el nombre de "entradas", porque son datos que "ingresan" a la aplicación por la acción del usuario para ser procesados.



Consola

Si queremos mostrar el resultado de una variable tenemos dos formas de hacerlo. La primera, muy útil para el programador, es utilizar la instrucción ***console.log()***, la cual permite visualizar en la consola del navegador el valor asignado, una variable, o un valor cualquiera especificado entre los paréntesis. Veamos un ejemplo de cómo usar console.log:

```
<script>
    console.log("Mensaje de prueba");
</script>
```

Alert

La segunda forma de mostrar al usuario el resultado de una variable es mediante la instrucción ***alert()***, la cual hace aparecer un cuadro de texto con el valor asignado de una variable, o un valor cualquiera especificado entre los paréntesis. Veamos un ejemplo de cómo usar alert:

```
<script>
    alert("¡Hola Coder!");
</script>
```

El usuario visualizará el valor en un cuadro hasta que presione aceptar. A los datos que mostramos mediante alert se los conoce con el nombre de "salidas", porque "salen" de la aplicación por el resultado de un procesamiento.

El procesamiento implica una o más instrucciones que transforman el dato ingresado por el usuario (entrada) en información de valor, para él y la muestra (salida)

Algoritmo

Mencionamos anteriormente que necesitamos construir un procesamiento para transformar lo ingresado por el usuario en algo de valor para el mismo. Este proceso comprende una serie de pasos ordenados para transformar algo.

Pensemos en un ejemplo de la vida cotidiana: imaginemos que estamos cocinando arroz blanco, y los pasos para hacerlo podrían ser hervir el agua, arrojar el arroz, esperar 10 minutos. Si bien esta secuencia de pasos puede servir para las personas, ya que podemos interpretar cómo ejecutar cada uno de ellos, esto no es igual para las

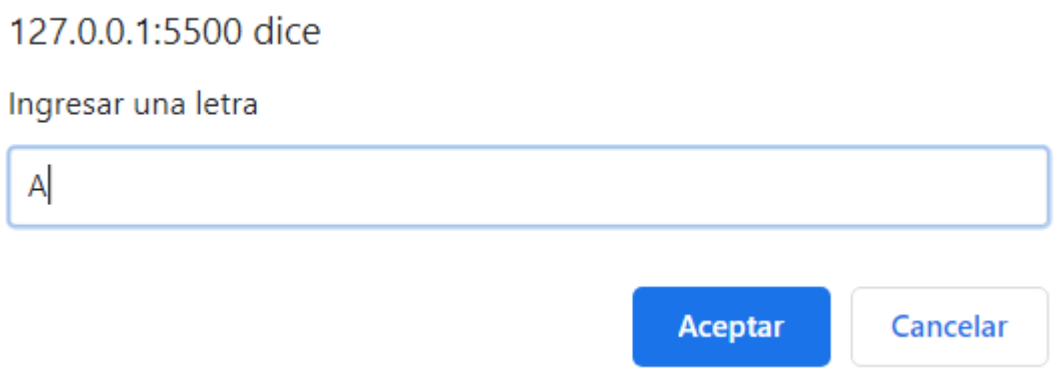
computadoras, donde tenemos que escribir específicamente qué se debe hacer para llegar a un resultado, sentencia por sentencia. Se conoce con el nombre de "algoritmo" al conjunto de instrucciones ordenadas que programamos para realizar una tarea.

Ejemplo de Script Completo

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mi primer App - CoderHouse</title>
    <script>
      let entrada = prompt("Ingresar una letra");
      let salida  = entrada + " " + "ingresada";
      alert(salida);
    </script>
  </head>
  <body>
    <h2>Esta página contiene una app</h2>
  </body>
</html>
```

Ejemplo de Entrada y Salida del Algoritmo Ejemplo

Si ingreso "A" como entrada:



Obtengo como salida: