JavaScript

(Video Concepto de Programación: https://www.youtube.com/watch?v=UzYgNZlKA4k)

JavaScript es el único lenguaje de programación que funciona en los navegadores de forma nativa (lenguaje interpretado sin necesidad de compilación). Por tanto se utiliza como complemento de HTML y CSS para crear páginas webs.

¿Qué es JavaScript?

JavaScript es el lenguaje de programación encargado de dotar de mayor interactividad y dinamismo a las páginas web.

Se lo reconoce como uno de los tres lenguajes nativos de la web junto a HTML (contenido y su estructura) y a CSS (diseño del contenido y su estructura).

No conviene confundir JavaScript con Java, que es un lenguaje de programación muy diferente. La confusión proviene del nombre, registrado por la misma empresa creadora de Java (Sun Microsystems). JavaScript (JS) se creó posteriormente, y la empresa norteamericana lo que hizo simplemente fue cambiar el nombre que le habían puesto sus creadores al comprar el proyecto (LiveScript). El lenguaje de programación Java está orientado a muchas más cosas que la web desde sus inicios.

¿Para qué sirve JavaScript?

Los lenguajes de programación suelen clasificarse por el lugar donde trabajan, por eso es que JS se lo conoce como que trabaja del lado del cliente.

El código de programación de JavaScript se ejecuta en los navegadores, ya sean de escritorio o móviles, ya sean Android o Iphone.

JavaScript es capaz de detectar errores en formularios, de crear bonitos sliders que se adapten a cualquier pantalla, de hacer cálculos matemáticos de forma eficiente, de modificar elementos de una página web de forma sencilla.

Pero también JS es el encargado de que existan herramientas como Google Analytics, Google Tag Manager, Facebook Pixel y tantas otras, que son claros ejemplos de JavaScript.

JavaScript ahora mismo es el lenguaje más popular.

De hecho, **desde hace años se ha creado una versión que es capaz de ser ejecutada también en el lado del servidor (Node JS)**.

Por tanto, ahora mismo se ejecuta JavaScript en los navegadores y en los servidores, creando a su alrededor una amplísima comunidad de desarrolladores casi full-stack.

JavaScript del lado del servidor compite en igualdad de condiciones con PHP, por ejemplo.

Como casi todo lenguaje de programación, podemos hacer programación orientada a objetos en JavaScript. Sin duda alguna, la forma en la que se crean, modifican y se muestran los objetos en el navegador ha sido uno de los grandes causantes de su auge.

La librería de JavaScript más utilizada en la historia, y que todavía se sigue utilizando es jQuery**.** Con jQuery podíamos hacer más cosas, escribiendo menos. Con una sintaxis mucho más sencilla, podíamos modificar nuestro sitio web, crear plugins, animar videojuegos y muchas cosas más. En la actualidad, jQueey ha perdido espacio en favor otras tecnologías más modelas como React o Angular.

JavaScript es un lenguaje de programación dinámico que tiene la capacidad de ser utilizado en muchos dispositivos diferentes. Puede usarse en ordenadores personales, servidores web y teléfonos inteligentes.

Es un lenguaje interpretado, orientado a objetos, débilmente tipado y dinámico.

¿Por qué decimos que Javascript es un lenguaje dinámico?

Corre directamente en la etapa de Runtime, sin una etapa de compilación previa.

Esto permite probar nuestro código inmediatamente; pero también es lo que hace que los errores no se muestren sino hasta que se ejecuta el programa.

Lo que se ve a primera vista, cuando se analiza el código, es muy probable que no sea lo que se va a obtener cuando el programa se ejecute.

JavaScript permite declarar (por ejemplo) variables cuyo valor (y tipo) solo se conocerá al momento de su ejecución en función de las condiciones dadas al momento de correrlo en un entorno real.

En cambio, los lenguajes estáticos no compilarán en código ejecutable a menos que todos los valores (o tipos de valores) se conozcan por adelantado.

¿Por qué es débilmente tipado?

Porque los tipos de datos no están bien definidos en el lenguaje y permite, por ejemplo, operaciones entre números y letras.

Esto sucede porque el lenguaje asume tipos de datos que no necesariamente fueron los que se querían representar.

Se pueden hacer operaciones entre tipos distintos de datos (enteros con strings, booleanos con enteros, etc.). Ejemplo:

```
4 + "7"; // 47
4 * "7"; // 28
2 + true; // 3
false - 3; // -3
```

¿Realmente es Javascript un lenguaje interpretado?

Sí, y la razón es que el navegador lee línea por línea nuestro código, el cual le indica lo que tiene que ir haciendo, sin la necesidad de compilar. Todo esto es controlado por el motor de Javascript V8 del navegador.

Potencia de JavaScript

Muchas de las tareas que realizamos con HTML y CSS se podrían realizar con Javascript.

De hecho, es muy probable que al principio nos parezca que es mucho más complicado hacerlo con Javascript, y que por lo tanto no merece la pena.

Sin embargo, con el tiempo veremos que Javascript nos ofrece una mayor flexibilidad y un abanico de posibilidades más grande, y que bien usadas, pueden ahorrarnos bastante tiempo.

Para comprenderlo, un ejemplo muy sencillo sería el siguiente:

```
<div class="item">
```

```
  <p>Número: <span class="numero">1</span></p>
  <p>Número: <span class="numero">2</span></p>
  <p>Número: <span class="numero">3</span></p>
  <p>Número: <span class="numero">4</span></p>
  <p>Número: <span class="numero">5</span></p>
</div>
```

Imaginemos que tenemos que crear una lista de números desde el 1 hasta el 500. Hacerlo solamente con HTML sería muy tedioso, ya que tendríamos que copiar y pegar esas filas varias veces hasta llegar a 500.

Sin embargo, mediante Javascript, podemos decirle al navegador que escriba el primer párrafo <p>, que luego escriba el mismo pero sumándole uno al número. Y que esto lo repita hasta llegar a 500.

De esta forma y con este sencillo ejemplo, con HTML habría que escribir 500 líneas mientras que con Javascript no serían más de 10 líneas.

Variables

Tipos de Datos

Estructuras de Control

Las estructuras de control son el conjunto de reglas que permiten controlar el flujo de ejecución de las instrucciones de un algoritmo o de un programa.

¿Para qué sirven las estructuras de control?

Las estructuras de control nos dan el poder de alterar, controlar o modificar el orden o el flujo en el que se ejecutan las instrucciones de un software a voluntad.

Tipos de Estructuras de Control

- Secuenciales
- Selectivas
- Iterativas

1) **Secuenciales o de secuencia**

Esta es la estructura básica, ya que nos permite asegurar que una instrucción se ejecuta después de la otra siguiendo el orden en que fueron escritas.

```
INSTRUCCIÓN 2
.
.
.
INSTRUCCIÓN N
```

2) Selectivas, de selección o condicionales

Este tipo de estructuras de control nos sirven cuando necesitamos que se evalúe el valor de alguna variable o de alguna condición para decidir qué instrucciones ejecutar a continuación.

Selectivas simples
Evalúan un valor o una condición y determinan las instrucciones a ejecutar en caso de cumplirse la condición.

```
SI <CONDICIÓN> ENTONCES
   INSTRUCCIÓN/ES
FIN SI
```

Selectivas dobles
Evalúan un valor o una condición. Determinan las instrucciones a ejecutar en caso de cumplirse la condición y también las instrucciones a ejecutar en caso de no cumplirse.

```
SI <CONDICIÓN> ENTONCES
  INSTRUCCIÓN/ES
SI NO
  INSTRUCCIÓN/ES
FIN SI
```

Selectivas múltiples o anidadas
Permiten combinar selectivas simples y dobles para crear estructuras y condiciones más complejas cuando el algoritmo en cuestión lo necesite.

```
SI <CONDICIÓN 1> ENTONCES
  INSTRUCCIÓN 1
SI NO
  SI <CONDICIÓN 2> ENTONCES
    INSTRUCCIÓN 2
```

```
  SI NO
    INSTRUCCIÓN 3
  FIN SI
FIN SI
```

3) Iterativas, de iteración, de repetición o repetitivas

Este tipo de estructuras de control nos sirven cuando necesitamos que se ejecute un conjunto específico de instrucciones en diversas ocasiones.
La cantidad de veces que se repite dicho bloque de acciones puede ser estático o puede depender del valor de alguna variable o de alguna condición.

Iterativas con cantidad fija de iteraciones

Se utilizan cuando a priori se conoce la cantidad de ocasiones que debe repetirse un bloque de instrucciones. Normalmente, usan una variable de iteración o índice para contar la cantidad de repeticiones que se han realizado.

```
PARA <VARIABLE> DESDE <VALOR1> HASTA <VALOR2> CON PASO <VALOR3> HACER
    ACCIÓN/ES
FIN PARA
```

El valor inicial <valor1> de la <variable> irá aumentando o disminuyendo según el paso <valor3> hasta llegar al valor <valor2>. Si no se especifica el valor de paso, se sobrentiende que el aumento es de uno en uno.

Iterativas con cantidad variable de iteraciones

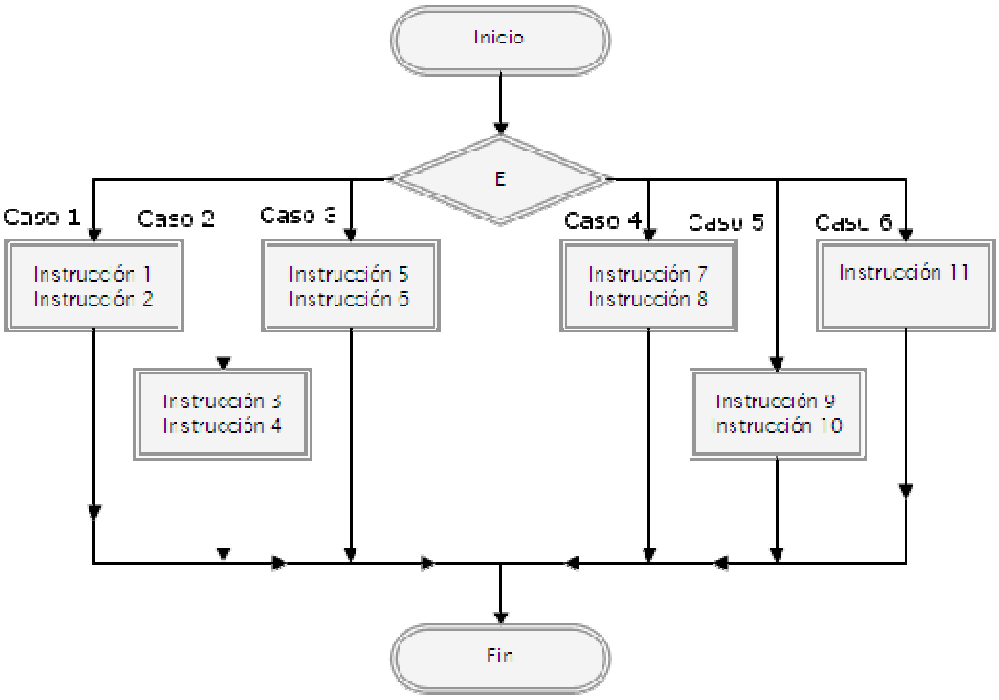
Se utilizan cuando la cantidad de ocasiones que debe repetirse un bloque de instrucciones está determinado por una condición. Por lo regular, existen dos variantes: repetir un bloque de instrucciones mientras se cumpla una condición o repartirlo hasta que se cumpla una condición.

```
MIENTRAS QUE <CONDICIÓN> HACER
  ACCIÓN/ES A REPETIR
FIN MIENTRAS
```

```
REPETIR
    ACCIÓN/ES
HASTA QUE <CONDICIÓN>
```

Estructuras de Control – SWITH

La estructura selectiva switch selecciona una de entre múltiples alternativas. Esta estructura es especialmente útil cuando la selección se basa en el valor de una variable simple o de una expresión simple denominada expresión de control o selector.



Operadores Lógicos

Funciones

Las funciones, también llamados métodos, nos permiten dividir el trabajo que hace un programa, en tareas más pequeñas separadas de la parte principal.

Video Funciones: https://www.youtube.com/watch?v=MiH3pbP4EFc

Procedimientos

Un procedimiento o subrutina es un subalgoritmo que recibiendo o no datos permite devolver varios resultados, un resultado o ninguno.

Un procedimiento está compuesto por un grupo de sentencias a las que asigna un nombre (identificador o simplemente nombre del procedimiento) y constituye una unidad de programa.

Video de Javascript:

https://www.youtube.com/watch?v=Zwcqq-7IDI0&list=PLPl81lqbj-4I11QPam9ApoT7tGbmyBg9P