4.1. Sintaxis y conceptos básicos de JavaScript

LENGUAJES DE PROGRAMACIÓN NIVEL 3. UNIDAD 4

JavaScript - LENGUAJE DE PROGRAMACIÓN PARA LA INTERACCIÓN EN EL FRONTEND

Financiado por la Unión Europea

Plan de Recuperación, Transformación y Resiliencia

Contenido

Introducción

Te doy la bienvenida al mundo de JavaScript, el lenguaje de programación que ha transformado la web desde su creación. Este módulo se sumerge en los fundamentos de JavaScript, explorando su sintaxis, conceptos clave y cómo interactúa con HTML y CSS para crear experiencias web dinámicas y ricas.

Desde su humilde comienzo hasta convertirse en una de las tecnologías más influyentes en el desarrollo web, JavaScript ha evolucionado para permitir todo, desde simples animaciones hasta complejas aplicaciones web. Acompáñanos en este viaje para descubrir cómo puedes utilizar JavaScript para dar vida a tus páginas web.

Origen del JavaScript

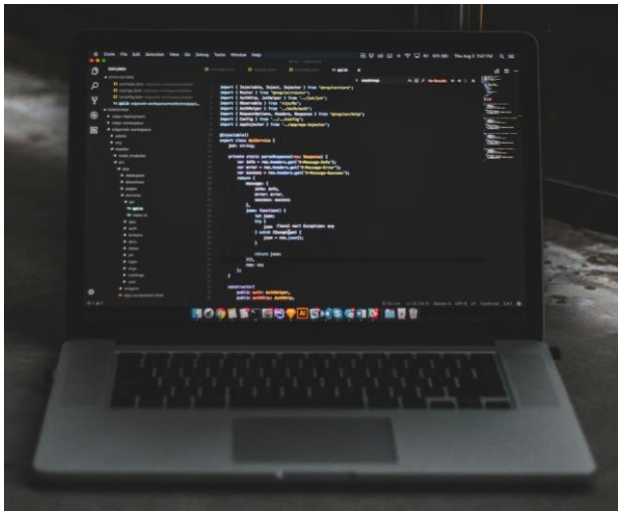
JavaScript fue creado en 1995 por Brendan Eich, quien trabajaba en Netscape Communications Corporation. Originalmente fue desarrollado bajo el nombre de Mocha, luego renombrado a LiveScript, y finalmente a JavaScript. Este cambio de nombre coincidió con una época en la que Netscape estaba formando una alianza con Sun Microsystems, la compañía detrás de Java, lo que llevó a cierta confusión en cuanto a la relación entre Java y JavaScript; aunque ambos lenguajes comparten algunas similitudes en la sintaxis, son fundamentalmente diferentes en diseño y propósito.

JavaScript empezó siendo un lenguaje de scripting del lado del cliente, es decir, un tipo de lenguaje de programación que se ejecuta en el navegador web del usuario en lugar de en el servidor web y que permite agregar interactividad simple a las páginas web. Sobre esta base, JavaScript ha evolucionado enormemente:

- Con la introducción de AJAX (Asynchronous JavaScript and XML) a principios de los años 2000, JavaScript comenzó a permitir la creación de aplicaciones web más complejas y dinámicas.
- El lanzamiento de bibliotecas y frameworks como jQuery, Angular, React, y Vue ha simplificado aún más el desarrollo de aplicaciones ricas en el frontend.

En la actualidad, JavaScript no se limita solo al navegador. Node.js, introducido en 2009, ha permitido que JavaScript se ejecute en el servidor, lo que abre un mundo de posibilidades para el desarrollo de aplicaciones de página completa y en tiempo real, así como herramientas y aplicaciones de línea de comandos.

El papel de JavaScript en el desarrollo web



JavaScript juega un papel crucial en el trío de tecnologías fundamentales del desarrollo web, junto con HTML y CSS. Mientras que HTML estructura el contenido y CSS se encarga del estilo y la presentación, JavaScript introduce la interactividad, permitiendo a los desarrolladores web crear una experiencia de usuario rica y dinámica.

¿Cómo introduce a grandes rasgos esa interactividad?

- **Interacción con HTML:** JavaScript puede manipular el DOM (Document Object Model)[1], lo que permite modificar dinámicamente el contenido, la estructura y el estilo de una página web. Esto posibilita la creación de características como formularios interactivos, galerías de imágenes dinámicas, y contenido que se actualiza en tiempo real sin necesidad de recargar la página.
- **Colaboración con CSS:** JavaScript también puede manipular estilos CSS directamente, permitiendo cambios dinámicos en la presentación de elementos de la página basados en la interacción del usuario, condiciones ambientales, o datos externos. Esto incluye animaciones, transiciones y la adaptación del layout para diferentes dispositivos y tamaños de pantalla, complementando las capacidades de diseño responsivo de CSS.

[1] El Document Object Model (DOM) es una interfaz de programación para documentos web. Proporciona una representación estructurada del documento HTML o XML, permitiendo que los lenguajes de programación, como JavaScript, accedan y modifiquen el contenido, la estructura y el estilo de la página web de manera dinámica. El DOM representa el documento como un árbol de nodos, donde cada nodo representa una parte del documento, como elementos, atributos y texto.

Configuración del entorno de desarrollo

El entorno de desarrollo es el conjunto de herramientas y procesos que utilizas para crear, probar y depurar tu código. Una configuración adecuada del entorno de desarrollo es esencial para cualquier programador, ya que proporciona el marco necesario para escribir código de manera eficiente, probarlo en condiciones similares a las de producción y resolver problemas que puedan surgir durante el desarrollo.

En el contexto de JavaScript, especialmente cuando se trabaja en el desarrollo web, esta configuración incluye no solo el editor de código y las herramientas de desarrollo del navegador, sino también la forma en que incorporas y ejecutas tu código JavaScript dentro de las páginas web.

Un entorno bien configurado te ayuda a centrarte en la lógica y la funcionalidad de tu código, minimizando las distracciones y maximizando tu productividad al proporcionarte retroalimentación instantánea y herramientas de depuración accesibles.

Este módulo te guiará a través de los pasos básicos para configurar tu entorno de desarrollo y te introducirá a herramientas esenciales que todo desarrollador de JavaScript debería conocer.

Incluir JavaScript en páginas web

Una de las primeras habilidades que debes dominar es cómo incluir JavaScript en tus páginas web. Esto se logra mediante el uso de la etiqueta <script>, que tiene dos usos principales

- Incorporar JavaScript directamente en HTML:
  Puedes escribir código JavaScript directamente dentro de un documento HTML utilizando la etiqueta <script>:

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

[2]

- Referenciar un archivo JavaScript Externo:
  Para mantener tu código organizado y promover la reutilización, es una buena práctica mantener tu JavaScript en archivos separados. Puedes enlazar estos archivos a tu HTML mediante el atributo src de la etiqueta <script>:

```
<script src="ruta/a/tu/archivo.js"></script>
```

[3]

  Esta técnica mantiene tu código limpio, facilita el mantenimiento y mejora la carga de la página al permitir la caché del navegador.

[2] Accesibilidad:
<script>
alert(“Hola mundo!”);
</script>

[3] Accesibilidad:
<script src=”ruta/a/tu/archivo.js”></script>

Uso de la consola del navegador

La consola del navegador es una herramienta invaluable para cualquier desarrollador de JavaScript. Te permite probar fragmentos de código en tiempo real, depurar errores y ver mensajes de log que tu código o el navegador generan.

- Acceso a la consola:
  La mayoría de los navegadores modernos permiten acceder a la consola de desarrollador presionando F12 o haciendo clic derecho en la página y seleccionando "Inspeccionar" o "Inspeccionar elemento", y luego navegando a la pestaña "Consola".
- Depuración y pruebas:
  Puedes usar console.log(), console.error(), y otros métodos de console para imprimir mensajes de depuración. Esto es útil para entender el flujo de tu programa o para identificar y corregir errores.

```
console.log("Mensaje de log para depuración");
```

[4]

- Ejecución de Código JavaScript:
  La consola también te permite ejecutar código JavaScript directamente, lo que es excelente para experimentar con nuevos conceptos o probar pequeños fragmentos de código.

[4] Accesibilidad;
console.log(“Mensaje de log para depuración”);

Editores de código

Además de entender cómo incluir JavaScript en tus páginas web y cómo utilizar la consola del navegador para pruebas y depuración, seleccionar un editor de código adecuado es crucial en la configuración de tu entorno de desarrollo. Un editor de código eficiente no solo te brinda un espacio de trabajo organizado y personalizable para escribir tu código, sino que también te equipa con funcionalidades avanzadas que pueden acelerar tu proceso de desarrollo y mejorar la calidad de tu código.

Elección de un editor de código

- **Editores populares:** Algunos de los editores de código más populares y ampliamente utilizados por los desarrolladores de JavaScript incluyen Visual Studio Code (VS Code), Sublime Text y Atom. Estos editores son apreciados por su interfaz de usuario intuitiva, extensa biblioteca de extensiones y plugins, y su capacidad para manejar proyectos grandes y complejos.
- **Características importantes:**
    - Resaltado de sintaxis: Facilita la lectura y escritura de código al diferenciar visualmente los elementos del lenguaje, como variables, funciones y keywords.
    - Autocompletado: Proporciona sugerencias de código basadas en el contexto actual, lo que puede acelerar significativamente la escritura de código.
    - Depuración integrada: Algunos editores ofrecen herramientas de depuración integradas que permiten inspeccionar y solucionar problemas en tu código directamente desde el editor.
    - Soporte para extensiones: La capacidad de instalar extensiones o plugins adicionales significa que puedes personalizar el editor según tus necesidades específicas, agregando soporte para control de versiones, formateo de código automático, soporte para diferentes frameworks y bibliotecas, y mucho más.

La elección del editor de código correcto depende en gran medida de tus preferencias personales, las necesidades de tu proyecto y tu flujo de trabajo. Muchos desarrolladores prueban varios editores antes de seleccionar el que mejor se adapta a sus necesidades.

Para empezar....

Para un alumno que comienza a explorar JavaScript, Visual Studio Code (VS Code) es a menudo la opción más recomendable debido a su balance entre facilidad de uso, potentes funcionalidades y una vasta comunidad de soporte. Aquí te explico por qué VS Code puede ser una excelente elección para principiantes en JavaScript:

Características

- **Interfaz amigable:** VS Code tiene una interfaz de usuario clara y bien organizada que es fácil de navegar, incluso para aquellos que son nuevos en los entornos de desarrollo integrados (IDEs) o editores de código.
- **Extensibilidad:** Una de las mayores fortalezas de VS Code es su sistema de extensiones. Hay miles de extensiones disponibles que pueden ayudarte a personalizar el editor según tus necesidades, incluyendo extensiones específicas de JavaScript que ofrecen autocompletado de código, snippets y detección de errores en tiempo real.
- **Soporte integrado para JavaScript:** VS Code ofrece un excelente soporte integrado para JavaScript, incluyendo resaltado de sintaxis, completado de código inteligente (IntelliSense), refactoring de código y más, lo cual es ideal para acelerar el proceso de aprendizaje y desarrollo.
- **Herramientas de depuración:** VS Code incluye poderosas herramientas de depuración integradas que te permiten inspeccionar variables, ver llamadas de función y navegar por tu código paso a paso para identificar y resolver problemas.
- **Gratuito y de código abierto:** VS Code es completamente gratuito y de código abierto, lo que significa que tienes acceso a una herramienta profesional sin costo alguno, y la comunidad contribuye constantemente con mejoras y nuevas funcionalidades.
- **Documentación y tutoriales:** Dado que VS Code es ampliamente utilizado, encontrarás una abundancia de tutoriales, guías y foros de la comunidad para ayudarte a comenzar y resolver cualquier problema que encuentres.
- **Plataforma cruzada:** VS Code funciona en Windows, MacOS y Linux, lo que te permite trabajar en el sistema operativo con el que te sientas más cómodo.

Consideraciones para principiantes

- **Curva de aprendizaje:** Aunque VS Code es amigable para los principiantes, tiene muchas características y configuraciones avanzadas. No te sientas abrumado; comienza con las funcionalidades básicas y explora más características a medida que te familiarices con el editor.
- **Comunidad y recursos:** Aprovecha la amplia comunidad de usuarios de VS Code. Hay numerosos recursos en línea, desde la documentación oficial hasta tutoriales en YouTube y cursos en plataformas de aprendizaje en línea, que pueden proporcionarte una guía detallada y soporte.

Al comenzar con JavaScript, la elección del editor correcto puede tener un impacto significativo en tu experiencia de aprendizaje. VS Code, con su combinación de facilidad de uso, potentes características y extenso soporte comunitario, ofrece un entorno de desarrollo sólido y accesible para los principiantes, facilitando el camino hacia convertirse en un desarrollador de JavaScript competente.

Sintaxis básica de JavaScript

La sintaxis, en el contexto de la programación, se refiere al conjunto de reglas y principios que definen la estructura de un lenguaje de programación. Es la gramática que guía cómo escribimos instrucciones que las computadoras pueden interpretar y ejecutar.

Al igual que en los idiomas humanos, donde la sintaxis determina el orden correcto de las palabras y la puntuación para formar oraciones comprensibles, en la programación, la sintaxis nos indica cómo combinar símbolos, palabras clave y estructuras para crear programas funcionales. Una sintaxis correcta es esencial para que el código se ejecute correctamente; un error sintáctico, por pequeño que sea, puede hacer que un programa falle o se comporte de manera inesperada. Por lo tanto, comprender y aplicar correctamente la sintaxis de JavaScript es fundamental para desarrollar aplicaciones web eficaces y eficientes.

Los elementos básicos que forman la sintaxis de JavaScript son los siguientes:

Variables y declaraciones

Las variables se declaran con let, const, o var (menos recomendado). Se utilizan para almacenar datos que pueden ser modificados y utilizados a lo largo del programa.

Hay de dos tipos:

- Variables: Se utilizan para almacenar datos que pueden cambiar durante la ejecución del programa. En JavaScript, las variables se declaran con las palabras clave let o var.
- Constantes: Se utilizan para almacenar datos que no deben cambiar una vez que se han asignado. Las constantes se declaran con la palabra clave const.

```
let contador = 0; // Variable que puede cambiar
const PI = 3.14159; // Constante cuyo valor no cambia
```

5

Tipos de datos

JavaScript es un lenguaje de tipado dinámico, lo que significa que no es necesario declarar el tipo de dato de una variable.

Los tipos de datos básicos incluyen String, Number, Boolean, null, undefined, Object, y Array.

```
let nombre = "Alice"; // String
let edad = 30; // Number
let estudiante = true; // Boolean
let indefinido; // Undefined
let hobbies = null; // Null
let objeto = { nombre: "Alice", edad: 30 }; // Object
let numeros = [1, 2, 3, 4, 5]; // Array
```

6

[5] Accesibilidad:
let contador=0; //Variable que puede cambiar
const PI=3.141516; //Constante cuyo valor no cambia

[6] Accesibilidad:
let nombre = "Alice"; //String
let edad = 30; //Number
let estudiante = true; //Boolean
let indefinido; // Undefined
let hobbies = null; // Null
let objeto = {nombre: "Alice", edad: 30 }; // Object
let numeros = [1, 2, 3, 4, 5]; // Array

Operadores

Se utilizan para realizar operaciones matemáticas, lógicas y de otro tipo entre variables y valores.

Los operadores incluyen aritméticos (+, -, *, /), de asignación (=, +=, -=), de comparación (==, !=, ===, !==, >, <), y lógicos (&&, ||, !).

```
let suma = 5 + 3; // Aritmético
let esMayor = edad > 18; // Comparación
let esAdulto = esMayor && estudiante; // Lógico
```

7

Expresiones y sentencias

Expresiones: Son combinaciones de valores, variables y operadores que se evalúan para producir otro valor. Por ejemplo, 5 * 10 o nombre + ' ' + apellido.

Sentencias: Son instrucciones que realizan una acción, como declarar una variable, ejecutar un bucle o evaluar una condición.

```
5 * 10; // Expresión que se evalúa en 50
if (edad > 18) { console.log("Adulto"); } // Sentencia condicional
```

8

[7] Accesibilidad:
let suma = 3+5; // Aritmético
let esMayor = edad > 18; // Comparación
let esAdulto = esMayor && estudiante; // Lógico

[8] Accesibilidad:
5 * 10; // Expresión que se evalúa en 50
If (edad >18) {console.log("Adulto"); } // Sentencia condicional

Funciones

Las funciones son bloques de código reutilizables que se definen una vez y se pueden ejecutar o "invocar" en cualquier momento. Las funciones pueden tomar parámetros, realizar una tarea y opcionalmente devolver un valor.

```
function saludar(nombre) {
    console.log("Hola, " + nombre + "!");
}
saludar("Alice"); // Invocar la función
```

9

[9] Accesibilidad:

```
function saludar(nombre) {
    console.log("Hola, "+ nombre + "!");
}
saludar("Alice"); // Invocar la función
```

Estructuras de control

Permiten dirigir el flujo del programa. Incluyen sentencias condicionales (if, else, switch) y bucles (for, while, do...while).

```
// Sentencia condicional
if (edad > 18) {
    console.log("Adulto");
} else {
    console.log("Menor de edad");
}

// Bucle
for (let i = 0; i < numeros.length; i++) {
    console.log(numeros[i]);
}
```

10

[10] Accesibilidad:

```
// Sentencia incondicional
if (edad > 18) {
    console.log("Adulto");
} else {
    console.log("Menor de edad");
}

// Bucle
for (let i = 0; i < números.length; i++) {
    console.log(numeros[i]);
}
```

Comentarios

Los comentarios son textos que se incluyen en el código para explicar lo que hace o para dejar notas para los desarrolladores.

Los comentarios pueden ser de una sola línea, iniciados con //, o de múltiples líneas, encerrados entre /* y */.

```
// Esto es un comentario de una sola línea

/*
  Esto es un comentario
  de múltiples líneas
*/
```

11

Puntos y comas

Se utilizan para separar las instrucciones. Aunque JavaScript permite omitirlos en muchos casos debido a la Inserción Automática de Punto y Coma (ASI), es una buena práctica incluirlos para evitar posibles errores.

```
let a = 5; let b = 10; a = a + b; // Uso de punto y coma
```

12

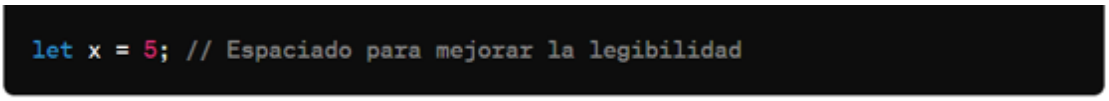
[11] Accesibilidad:
// Esto es un comentario de una sola línea

/*
  Esto es un comentario
  de múltiples líneas
*/

[12] Accesibilidad:
  let a = 5; let b =10; a = a + b; // Uso de punto y coma

Espacios en blanco

JavaScript ignora los espacios en blanco excesivos, pero usarlos adecuadamente mejora la legibilidad del código.



13

La combinación de estos elementos forma la estructura básica de un programa en JavaScript, permitiendo la creación de aplicaciones web interactivas y dinámicas. Entender estos fundamentos es esencial para cualquier desarrollador que desee trabajar con JavaScript.

[13] Accesibilidad:
let x = 5; // Espaciado para mejorar la legibilidad

Buenas prácticas iniciales

A medida que te adentras en el mundo de la programación con JavaScript, adoptar buenas prácticas desde el principio es esencial para desarrollar código claro, eficiente y mantenible. Este módulo se basa en los fundamentos de la sintaxis de JavaScript que has aprendido anteriormente, enfocándose en cómo aplicar esos conocimientos de manera que mejoren la calidad de tu código y tu eficacia como programador.

Empecemos:

Importancia de nombrar adecuadamente las variables

Los nombres de las variables deben ser descriptivos y claros, lo que facilita la comprensión del propósito de la variable simplemente mirando su nombre. Un buen nombre de variable puede hacer que tu código sea mucho más legible y autoexplicativo, reduciendo la necesidad de comentarios adicionales.

Algunos consejos sobre nombrar variables:

- Usa nombres que describan claramente el valor almacenado en la variable.
- Prefiere el uso de camelCase para nombres compuestos (ej., miVariableImportante).

- Evita nombres genéricos como x o data a menos que sean en contextos muy específicos y de corta duración.

Sensible a mayúsculas y minúsculas

Recuerda que JavaScript es sensible a mayúsculas y minúsculas. Esto significa que identificadores como variable, Variable y VARIABLE serían considerados completamente diferentes por el intérprete de JavaScript. Mantener una consistencia en el uso de mayúsculas y minúsculas te ayudará a evitar errores difíciles de detectar.

Conceptos iniciales de depuración

La depuración es una parte crucial del desarrollo de software. Aprender a identificar y corregir errores en tu código te permitirá desarrollar aplicaciones más robustas y fiables.

En este sentido, no olvides tener en cuenta los siguientes aspectos:

- **Herramientas de depuración:** Familiarízate con las herramientas de depuración disponibles en los navegadores, como la consola y el depurador en las herramientas de desarrollo de Chrome o Firefox. Estas herramientas te permiten inspeccionar el estado de tu programa, establecer puntos de interrupción y ver cómo se ejecuta tu código en tiempo real.
- **Mensajes de error:** Aprende a leer y entender los mensajes de error que JavaScript proporciona. A menudo, estos mensajes te darán pistas sobre la línea de código que está causando el problema y el tipo de error que se ha encontrado.
- **Pruebas y validaciones:** Realiza pruebas con diferentes entradas y en diferentes escenarios para asegurarte de que tu código se comporta como esperas. La validación de datos de entrada también es crucial para prevenir errores inesperados.

Conclusión

Este módulo de sintaxis y conceptos básicos es solo el comienzo de tu viaje en el aprendizaje de JavaScript. Cada concepto que has aprendido y cada buena práctica que has adoptado te servirán como herramientas valiosas a medida que profundices en aspectos más avanzados del lenguaje. Te animo a que continúes explorando los módulos siguientes con un espíritu de curiosidad y la disposición para enfrentarte a retos más complejos.

Gracias por dedicar tu tiempo y esfuerzo para completar este módulo.

Esperamos que este módulo te haya proporcionado una comprensión clara de las buenas prácticas iniciales en JavaScript y te haya motivado a seguir aprendiendo y descubriendo todo lo que este poderoso lenguaje de programación tiene para ofrecer.

Nos vemos en el siguiente.

Recursos adicionales

Para complementar tu aprendizaje y ofrecerte más herramientas y recursos, aquí tienes una selección de enlaces a documentación básica y tutoriales para principiantes en JavaScript. Estos recursos son ideales para reforzar los conceptos que has aprendido y explorar nuevos temas.

Documentación y referencias

- MDN Web Docs (Mozilla Developer Network)
  Una de las mejores fuentes de documentación para desarrolladores web, incluyendo una completa sección sobre JavaScript.
  https://developer.mozilla.org/es/docs/Web/JavaScript
- W3Schools JavaScript Tutorial
  Un recurso amigable para principiantes que cubre los fundamentos de JavaScript con ejemplos interactivos.
  https://www.w3schools.com/js/

Tutoriales y cursos

- JavaScript.info
  Un tutorial moderno que cubre desde los aspectos más básicos hasta temas avanzados de JavaScript.
  https://javascript.info/
- Codecademy
  Proporciona un curso interactivo de JavaScript que es excelente para principiantes absolutos.
  https://www.codecademy.com/catalog/language/javascript

Herramientas interactivas

- CodePen y JSFiddle
  Plataformas que te permiten escribir, compartir y descubrir fragmentos de código HTML, CSS y JavaScript.
  https://codepen.io/
  https://jsfiddle.net/

Estos recursos te proporcionarán una base sólida y oportunidades amplias para explorar y mejorar tus habilidades en JavaScript. La práctica constante y la curiosidad por aprender conceptos nuevos son clave para tu crecimiento como desarrollador. ***¡Disfruta el viaje!***