DWC

(Desarrollo Web en entorno cliente)

Índice

1.- ¿Qué es JavaScript?

JavaScript se creó inicialmente para "dar vida a las páginas web".

Los programas en este lenguaje se llaman *scripts*. Se pueden escribir directamente en el HTML de una página web y ejecutarse automáticamente a medida que se carga la página.

Los scripts se proporcionan y ejecutan como texto sin formato. No necesitan preparación especial o compilación para correr.

En este aspecto, JavaScript es muy diferente de otro lenguaje llamado Java .

¿Por qué se llama JavaScript?

> Cuando se creó JavaScript, inicialmente tenía otro nombre: "LiveScript".
>
> Pero Java era muy popular en ese momento, por lo que se decidió que sería útil posicionar un nuevo lenguaje como un "hermano menor" de Java.
>
> Pero a medida que evolucionó, JavaScript se convirtió en un lenguaje totalmente independiente con su propia especificación llamada **ECMAScript**, y ahora no tiene ninguna relación con Java.

Hoy, JavaScript puede ejecutarse no solo en el navegador, sino también en el servidor, o en cualquier dispositivo que tenga un programa especial llamado **motor de JavaScript**.

El navegador tiene un motor incrustado a veces llamado "máquina virtual de JavaScript".

Diferentes motores tienen diferentes "nombres en clave". Por ejemplo:

- V8 - en Chrome y Opera.
- SpiderMonkey - en Firefox.
- Hay otros nombres en clave como "Trident" y "Chakra" para diferentes versiones de IE, "ChakraCore" para Microsoft Edge, "Nitro" y "SquirrelFish" para Safari, etc.

Los términos anteriores son buenos para recordar porque se usan en artículos para desarrolladores en Internet. Los usaremos también. Por ejemplo, si "una

característica X es compatible con V8", entonces probablemente funcione en Chrome y Opera.

¿Cómo funcionan los motores?

Los motores son complicados. Pero lo básico es fácil.

1. El motor (incrustado si es un navegador) lee ("analiza") el script.
2. Luego convierte ("compila") el script al lenguaje de máquina.
3. Y luego se ejecuta el código de la máquina, bastante rápido.

El motor aplica optimizaciones en cada paso del proceso. Incluso observa el script compilado mientras se ejecuta, analiza los datos que fluyen a través de él y optimiza aún más el código de la máquina en función de ese conocimiento.

2.-¿Qué puede hacer JavaScript en el navegador?

JavaScript moderno es un lenguaje de programación "seguro". No proporciona acceso de bajo nivel a la memoria o la CPU, porque se creó inicialmente para navegadores que no lo requieren.

Las capacidades de JavaScript dependen en gran medida del entorno en el que se está ejecutando. Por ejemplo, Node.js admite funciones que permiten a JavaScript leer / escribir archivos arbitrarios, realizar solicitudes de red, etc.

JavaScript en el navegador puede hacer todo lo relacionado con la manipulación de la página web, la interacción con el usuario y el servidor web.

Por ejemplo, JavaScript en el navegador puede:

- Agregar nuevo HTML a la página, cambiar el contenido existente, modificar estilos a través del **DOM**.
- Reaccionar a las acciones del usuario a través de los **eventos**, (clics del mouse, movimientos del puntero, pulsaciones de teclas…)
- Enviar solicitudes a través de la red a servidores remotos, las llamadas tecnologías **AJAX**.
- Obtener y establecer **cookies**, Almacenar los datos en el lado del cliente (**Local Storage** "almacenamiento local").

3.- ¿Qué NO PUEDE hacer JavaScript en el navegador?

Las capacidades de JavaScript en el navegador están limitadas por la seguridad del usuario. El objetivo es evitar que una página web malvada acceda a información privada o dañe los datos del usuario.

Los ejemplos de tales restricciones incluyen:

- JavaScript en una página web no puede leer / escribir archivos arbitrarios en el disco duro, copiarlos o ejecutar programas. No tiene acceso directo a las funciones del sistema operativo. Los navegadores modernos le permiten trabajar con archivos, pero el acceso es limitado y solo se proporciona si el usuario realiza ciertas acciones, como "colocar" un archivo en una ventana del navegador o seleccionarlo mediante un `<input>` etiqueta de HTML. Hay formas de interactuar con la cámara / micrófono y otros dispositivos, pero requieren el permiso explícito del usuario. Por lo tanto, una página habilitada para JavaScript puede no habilitar furtivamente una cámara web, observar los alrededores y enviar la información.

- Las diferentes pestañas / ventanas generalmente no se conocen entre sí. A veces lo hacen, por ejemplo, cuando una ventana usa JavaScript para abrir la otra. Pero incluso en este caso, JavaScript de una página puede no acceder a la otra si provienen de diferentes sitios (de un dominio, protocolo o puerto diferente). Esto se llama la **"Política del mismo origen"**. Para evitarlo, *ambas páginas* deben aceptar el intercambio de datos y contener un código JavaScript especial que lo maneje. Esta limitación es, nuevamente, para la seguridad del usuario. Una página desde la `http://anysite.com`que un usuario ha abierto no debe poder acceder a otra pestaña del navegador con la URL `http://gmail.com`y robar información desde allí.

- JavaScript puede comunicarse fácilmente a través de la red con el servidor de donde proviene la página actual. Pero su capacidad para recibir datos de otros sitios / dominios está paralizada. Aunque es posible,

requiere un acuerdo explícito (expresado en encabezados HTTP) desde el lado remoto. Una vez más, esto es una limitación de seguridad.

Dichos límites no existen si JavaScript se usa fuera del navegador, por ejemplo, en un servidor. Los navegadores modernos también permiten complementos / extensiones que pueden solicitar permisos extendidos.

4.- ¿Qué hace que JavaScript sea único?

Hay al menos *tres* grandes cosas sobre JavaScript:

- Integración completa con HTML / CSS.
- Las cosas simples se hacen simplemente.
- Soporte por todos los principales navegadores y habilitado por defecto.

JavaScript es la única tecnología de navegador que combina estas tres cosas.
Eso es lo que hace que JavaScript sea único. Es por eso que es la herramienta más extendida para crear interfaces de navegador.
Dicho esto, JavaScript también permite crear servidores, aplicaciones móviles, etc.

5.- Lenguajes "sobre" JavaScript

La sintaxis de JavaScript no se adapta a las necesidades de todos. Diferentes personas quieren diferentes características.
Eso es de esperar, porque los proyectos y requisitos son diferentes para todos.
Recientemente, apareció una gran cantidad de nuevos idiomas, que se ***transpilan*** (convierten) a JavaScript antes de que se ejecuten en el navegador.
Las herramientas modernas hacen que la transpilación sea muy rápida y transparente, lo que permite a los desarrolladores codificar en otro idioma y convertirla automáticamente.

Ejemplos de tales lenguajes:

- **CoffeeScript** es un "azúcar sintáctico" para JavaScript. Introduce una sintaxis más corta, lo que nos permite escribir código más claro y más preciso. Por lo general, a los desarrolladores de Ruby les gusta.
- **TypeScript** se concentra en agregar "mecanografía estricta de datos" para simplificar el desarrollo y el soporte de sistemas complejos. Está desarrollado por Microsoft.
- **Flow** también agrega mecanografía de datos, pero de una manera diferente. Desarrollado por Facebook.
- **Dart** es un lenguaje independiente que tiene su propio motor que se ejecuta en entornos que no son del navegador (como aplicaciones móviles), pero que también se puede transpilar a JavaScript. Desarrollado por Google.

Hay más. Por supuesto, incluso si usamos uno de los idiomas transpilados, también deberíamos saber JavaScript para comprender realmente lo que estamos haciendo.

6.- Resumen características JavaScript

- JavaScript se creó inicialmente como un lenguaje solo para el navegador, pero ahora también se usa en muchos otros entornos.
- Hoy, JavaScript tiene una posición única como el lenguaje de navegador más ampliamente adoptado con plena integración con HTML / CSS.
- Hay muchos idiomas que se "transpilan" a JavaScript y proporcionan ciertas funciones. Se recomienda echarles un vistazo, al menos brevemente, después de dominar JavaScript.

7.- Especificación del lenguaje

La especificación ECMA-262 contiene la información más detallada, detallada y formal sobre JavaScript. Define el lenguaje.

Pero al ser tan formalizado, es difícil de entender al principio. Entonces, si necesita la fuente de información más confiable sobre los detalles del idioma, la especificación es el lugar correcto. Pero no es para uso diario.

Se lanza una nueva versión de especificación cada año. Entre estos lanzamientos, el último borrador de la especificación se encuentra en **https://tc39.es/ecma262/**.

8.- Manuales de referencia

Existen en Internet muchos manuales y tutoriales sobre JavaScript pero todos ellos hacen referencia de alguna forma a los siguientes.

- **MDN (Mozilla) JavaScript Reference** es un manual con ejemplos y otra información. Es genial obtener información detallada sobre las funciones, métodos, etc. del lenguaje individual.Se puede encontrar en https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia.
  Sin embargo, a menudo es mejor usar una búsqueda en Internet. Simplemente use "MDN [term]" en la consulta, por ejemplo, https://google.com/search?q=MDN+parseInt para buscar la función `parseInt`.

- **MSDN** : manual de Microsoft con mucha información, incluido JavaScript (a menudo denominado JScript).
  Si necesita algo específico para Internet Explorer, mejor vaya allí: http://msdn.microsoft.com/ .
  Además, podemos utilizar una búsqueda en Internet con frases como "RegExp MSDN" o "RegExp MSDN jscript".

9.- Editores de código

Un editor de código es el lugar donde los programadores pasan la mayor parte de su tiempo.

Hay dos tipos principales de editores de código: IDEs y editores ligeros. Muchas personas usan una herramienta de cada tipo.

9.1.- IDE

El término IDE (Entorno de desarrollo integrado) se refiere a un editor potente con muchas características que generalmente opera en un "proyecto completo". Como su nombre indica, no es solo un editor, sino un "entorno de desarrollo" a gran escala.

Un IDE carga el proyecto (que puede tener muchos archivos), permite la navegación entre archivos, proporciona autocompletado basado en todo el proyecto (no solo el archivo abierto) y se integra con un sistema de administración de versiones (como git), un entorno de prueba y otras cosas de "nivel de proyecto".

Si aún no ha seleccionado un IDE, considere las siguientes opciones:

- Visual Studio Code (multiplataforma, gratis).
- WebStorm (multiplataforma, de pago).

Para Windows, también hay "Visual Studio", que no debe confundirse con "Visual Studio Code". "Visual Studio" es un editor pago y poderoso solo para Windows, muy adecuado para la plataforma .NET. También es bueno en JavaScript. También hay una versión gratuita de Visual Studio Community .

Muchos IDEs son pagados, pero tienen un período de prueba. Su costo generalmente es insignificante en comparación con el salario de un desarrollador calificado, así que solo elija el mejor para usted.

9.2.- Editores ligeros

Los "editores ligeros" no son tan potentes como los IDE, pero son rápidos, elegantes y simples.

Se utilizan principalmente para abrir y editar un archivo al instante.

La principal diferencia entre un "editor ligero" y un "IDE" es que un IDE funciona a nivel de proyecto, por lo que carga muchos más datos al inicio, analiza la estructura del proyecto si es necesario y así sucesivamente. Un editor ligero es mucho más rápido si solo necesitamos un archivo.
En la práctica, los editores ligeros pueden tener muchos complementos, incluidos analizadores de sintaxis de nivel de directorio y autocompletadores, por lo que no hay un límite estricto entre un editor ligero y un IDE.

Las siguientes opciones merecen su atención:

- **Atom** (multiplataforma, gratis).
- **Sublime Text** (multiplataforma, shareware).
- **Notepad ++** (Windows, gratis).
- **Vim** y **Emacs** también son geniales si sabes cómo usarlos.

10.- Herramientas de desarrollo (developer console)

El código es propenso a errores. Seguro que nuestros scripts van a tener errores, pero en el navegador, los usuarios no ven errores por defecto. Entonces, si algo sale mal en el script, no veremos lo que está mal y no podemos corregirlo.

Para ver los errores y obtener mucha otra información útil sobre los scripts, se han incorporado "herramientas de desarrollo" en los navegadores.

La mayoría de los desarrolladores se inclinan por Chrome o Firefox para el desarrollo porque esos navegadores tienen las mejores herramientas para desarrolladores. Otros navegadores también proporcionan herramientas de desarrollo, a veces con características especiales, pero generalmente están jugando a ponerse al día con Chrome o Firefox. Por lo tanto, la mayoría de los desarrolladores tienen un navegador "favorito" y cambian a otros si un problema es específico del navegador.

Las herramientas de desarrollo son potentes; Tienen muchas características. Para comenzar, aprenderemos cómo abrirlos, observar errores y ejecutar comandos JavaScript.

10.1.- Google Chrome

Veamos la página siguiente bug.html.

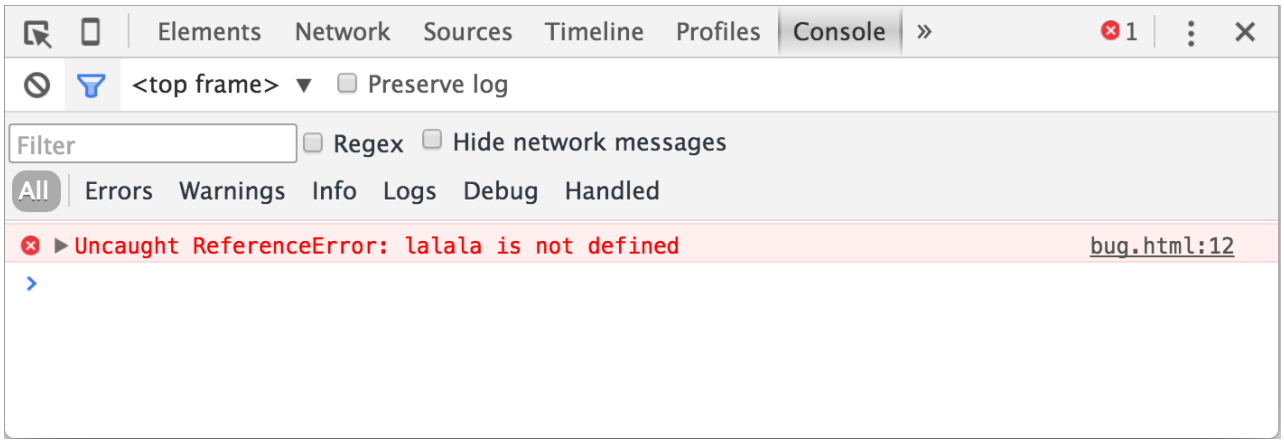
```
<!DOCTYPE HTML>
<html>

<head>
  <meta charset="utf-8">
</head>

<body>

  There is an error in the script on this page.
  <script>
    lalala
  </script>

</body>

</html>
```

Hay un error en el código JavaScript en él. Está oculto a los ojos de un visitante, así que abramos las herramientas de desarrollador para verlo.

Presione F12 o, si está en Mac Cmd+Opt+J

Las herramientas de desarrollador se abrirán en la pestaña Consola de forma predeterminada.

Se ve algo así:



El aspecto exacto de las herramientas de desarrollador depende de su versión de Chrome. Cambia de vez en cuando, pero debería ser similar.

- Aquí podemos ver el mensaje de error de color rojo. En este caso, el script contiene un comando desconocido "lalala".
- A la derecha, hay un enlace en el que se puede hacer clic a la fuente `bug.html:12` con el número de línea donde se produjo el error.

Debajo del mensaje de error, hay un `>` símbolo azul. Marca una "línea de comando" donde podemos escribir comandos JavaScript. Presione Enter para ejecutarlos.

Entrada multilínea

Por lo general, cuando ponemos una línea de código en la consola y luego presionamos Enter, se ejecuta.

Para insertar varias líneas, presione Shift+Enter . De esta manera, se pueden introducir fragmentos largos de código JavaScript.

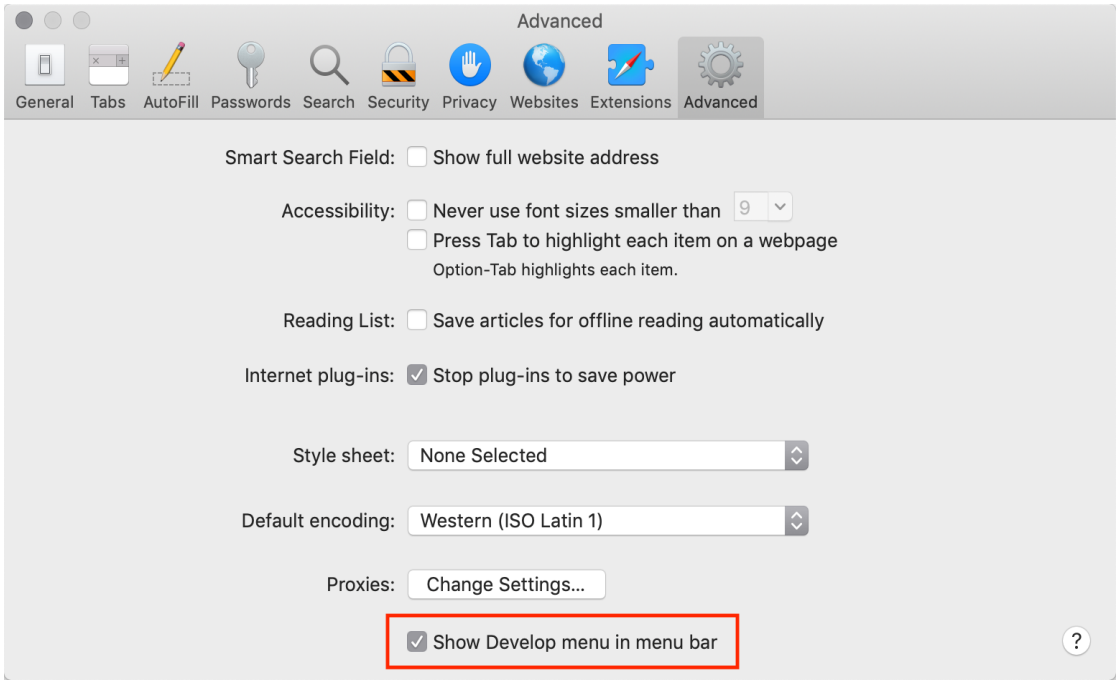
10.2.- Firefox, Edge y otros

La mayoría de los navegadores usan F12 para abrir herramientas de desarrollador.

La apariencia de ellos es bastante similar. Una vez que sepa cómo usar una de estas herramientas (puede comenzar con Chrome), puede cambiar fácilmente a otra.

10.3.- Safari

Safari (navegador Mac, no compatible con Windows / Linux) es un poco especial aquí. Necesitamos habilitar primero el "Menú de desarrollo".
Abra Preferencias y vaya al panel "Avanzado". Hay una casilla de verificación en la parte inferior:



Ahora puede alternar la consola. Además, tenga en cuenta que ha aparecido el nuevo elemento del menú superior denominado "Desarrollar". Tiene muchos comandos y opciones.