UD1

Arquitecturas y tecnologías para Clientes Web

Introducción

¿Cuándo se creó la web?

La web fue inicialmente concebida y creada por **Tim Berners-Lee**, un especialista del laboratorio europeo de partículas (CERN, Organización Europea para la Investigación Nuclear) en **1989**.

¿Cuál es el aspecto más importante de una web?

Algunos consideran la creación y organización de **contenido** - o más formalmente, la arquitectura de la información - como el aspecto más importante del Diseño Web. Otros factores como - la **facilidad de uso**, el **valor** y **funcionalidad** del sitio web en cuanto a su organización, su funcionalidad, accesibilidad, publicidad, etc. también forman una parte muy activa hoy en día sobre lo que se considera Diseño Web.

¿Qué influye en el desarrollo web en general?

El Desarrollo Web ha sido y sigue estando muy influenciado por múltiples campos como el de las nuevas tecnologías, los avances científicos, el **diseño gráfico**, la programación, las redes, el **diseño de interfaces de usuario**, la **usabilidad** y una variedad de múltiples recursos. Por lo tanto el Desarrollo Web es realmente un campo multidisciplinar.

Usabilidad, ¿qué es?

Cliente-Servidor

Hoy en día los sitios web siguen un modelo basado en la **programación cliente-servidor** con tres **elementos** comunes:

- El lado del **servidor (*server-side*)**: incluye el hardware y software del servidor Web así como diferentes elementos de programación y otras tecnologías, como bases de datos.
- El lado del **cliente (*client-side*)**: este elemento hace referencia a los navegadores web y está soportado por tecnologías como HTML, CSS y lenguajes como JavaScript, los cuales se utilizan para crear la presentación de la página o proporcionar características interactivas. Es justamente aquí dónde nos vamos a centrar a lo largo de todo el módulo. Las tecnologías del lado cliente son aquellas que se pueden ejecutar en el cliente, generalmente a través de un navegador web.
- La **red**: describe los diferentes elementos de conectividad utilizados para mostrar el sitio web al usuario.

¿Por qué Javascript?

Cada tipo general de programación tiene su propio lugar y la mezcla es generalmente la mejor solución. Cuando hablamos de lenguajes de programación en clientes web, podemos distinguir dos variantes:

- Lenguajes que nos permiten dar **formato** y **estilo** a una página web (HTML, CSS, etc.).
- Lenguajes que nos permite aportar **dinamismo** a páginas web (lenguajes de *scripting*).

En este módulo nos vamos a centrar principalmente en estos últimos, los lenguajes de *scripting*, y en particular en el lenguaje **JavaScript** que será el lenguaje que utilizaremos a lo largo de todo este módulo formativo.

El 95% de las aplicaciones web se crean con Javascript

Javascript

- Es muy **fácil de implementar**: tan solo es necesario colocar el código en un documento HTML y decirle al navegador que es Javascript.
- Funciona en **todos los navegadores** que usan los usuarios de la web, incluso cuando están desconectados de Internet.
- Permite crear **interfaces amigables** que mejoran la experiencia del usuario y brindan mucho **dinamismo**, sin tener que esperar a que el servidor reacciones y muestre otra página.
- Puede **cargar contenido de forma asíncrona** en el documento si el usuario lo necesita y cuando lo necesite, sin recargar toda la página.
- Puede comprobar lo que es capaz de hacer un navegador y actuar en consecuencia.
- Puede ayudar a solucionar problemas de incompatibilidades entre navegadores y corregir problemas de diseño con CSS.

Javascript y los navegadores web

Actualmente existen múltiples clientes o navegadores que soportan JavaScript, incluyendo **Firefox**, **Google Chrome**, **Safari**, **Opera**, **Edge**, etc. Por lo tanto, cuando escribimos un script en nuestra página web, tenemos que estar seguros de que será interpretado por diferentes navegadores y, por tanto, deberá aportar la misma funcionalidad y características en cada uno de ellos.

A veces las incompatibilidades entre navegadores al interpretar el código de JavaScript no vienen dadas por el propio código en sí, sino que su origen proviene del código fuente HTML. Por lo tanto es muy importante que tu código HTML siga las especificaciones del estándar W3C y, para ello, dispones de herramientas como el validador HTML W3C: http://validator.w3.org

Busca el nombre de 5 navegadores web

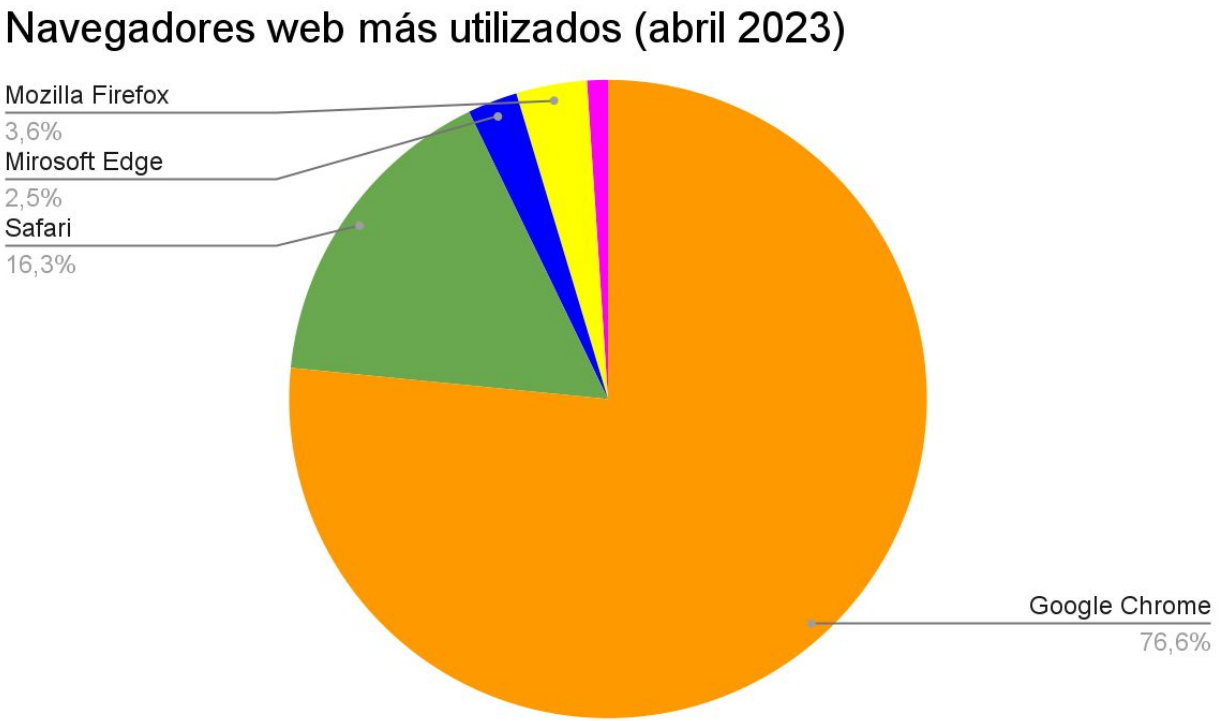
Z	G	J	M	Q	T	B	Z	A	M	I	V
F	S	A	F	A	R	I	Q	C	L	N	H
V	B	K	M	P	E	C	L	I	P	X	E
A	P	T	E	E	Y	I	Q	D	O	S	B
P	X	H	U	M	W	M	X	F	B	C	A
T	J	T	I	N	O	P	E	R	A	D	E
A	E	R	T	Y	E	R	C	S	Z	X	G
N	C	V	B	N	I	T	H	A	R	K	J
A	Z	F	G	F	K	H	B	C	T	T	O
F	T	Y	U	I	J	K	P	E	S	R	W
R	E	X	P	L	O	R	E	R	A	H	B
W	S	X	C	V	G	H	R	U	B	N	Y
F	G	H	J	K	L	R	T	F	I	V	S

HERRAMIENTAS: Navegadores web (1)

El navegador web es una de las principales herramientas, puesto que sin él no se pueden ver las aplicaciones ejecutándose en su contexto.

Atendiendo al porcentaje de uso de los diferentes navegadores, se puede ver claramente el absoluto dominio de **Google Chrome**.

¿Quiere decir esto que sólo tenga que programar para Chrome? Ni mucho menos, porque dejaría casi el 25% de usuarios poco optimizados.



HERRAMIENTAS: Navegadores web (2)

El tipo de **navegador web** que utilices es elección tuya. Eso sí, te recomiendo que uses las últimas versiones disponibles para evitar problemas de seguridad e incompatibilidades. Algunos ejemplos de navegadores gratuitos son:

- Para **Windows** tienes: Mozilla Firefox, Google Chrome, Safari, Opera, Internet Explorer, etc.
- Para **Macintosh** tienes: Mozilla Firefox, Safari, Google Chrome, Internet Explorer, etc.
- Para **Linux** tienes: Mozilla Firefox, Konqueror, Opera, etc.

Una recomendación muy interesante es el disponer de 2 o 3 tipos de navegadores diferentes, ya que así podrás comprobar la compatibilidad de tu página web y ver si tu código fuente de JavaScript se ejecuta correctamente en todos ellos.

HERRAMIENTAS: Navegadores web (3)

Para ajustar un poco más tu entorno de trabajo, lo último que necesitas es el poder ejecutar tu editor web y tu navegador de forma simultánea, ya que el **flujo típico de trabajo en JavaScript** va a ser:

1. Introducir HTML, JavaScript y CSS en el documento original en el editor web.
2. Guardarlo en disco.
3. Cambiarte al navegador web.
4. Realizar una de las siguiente tareas:
    - Si es un nuevo documento, abrirlo a través de la opción Abrir del menú Archivo > Abrir Archivo.
    - Si el documento ya está cargado en el navegador pues simplemente recargar la página.

Los pasos del 2 al 4 son acciones que se van a ejecutar muy frecuentemente. Esa secuencia grabar-cambiar-recargar la realizarás tantas veces cuando estés escribiendo y depurando tu script, que llegará a ser prácticamente un acto reflejo.

HERRAMIENTAS: Editor de código (1)

Bien, tenemos un navegador donde ver el resultado de nuestro trabajo, pero nuestro trabajo debe escribirse en algún sitio. Los editores de texto permiten a los programadores abrir múltiples archivos de código ayudándoles a acortar el tiempo de desarrollo y a realizar las tareas de manera más eficiente.

Así como un procesador de texto enriquecido es una aplicación personalizada para editar documentos, con muchas funciones avanzadas para escribir, editar, etc. un editor de código tiene muchas funciones personalizadas especialmente diseñadas para satisfacer las necesidades de un desarrollador de software.

HERRAMIENTAS: Editor de código (2)

Algunas de las características incluyen:

- **Comprobación de errores** conforme se va escribiendo, lo que proporciona un primer nivel de corrección.
- **Sugerencias de autocompletado**, que agiliza la escritura y acorta los tiempos de desarrollo.
- **Fragmentos de código previamente configurados** con aquellas porciones utilizadas con mucha frecuencia y que evitan reescribirlos constantemente.
- **Resaltado de sintaxis**, que ayuda mucho a identificar cada uno de los elementos de los programas con un simple golpe de vista.
- **Navegación entre archivos y recursos**, que ayuda a tener localizados todos los ficheros en sus ubicaciones correctas.

HERRAMIENTAS: Editor de texto (3) ¿o IDE?

A menudo se escuchan conceptos como editor de código y entorno de desarrollo integrado usados como sinónimos, cuando en realidad se trata de dos herramientas muy distintas.

Un **editor de código** sirve principalmente para crear un código más limpio y eficiente gracias a las características y funciones que incorpora. Sin embargo, un **IDE** (Entorno de Desarrollo Integrado) es un conjunto de herramientas combinadas. Los IDEs toman su nombre de la integración de otras herramientas avanzadas como depuradores, analizadores de código, optimizadores, compiladores, controladores de versiones y una gran cantidad de otras características que ayudan a un desarrollador a moverse a lo largo del ciclo de vida del desarrollo del software.

Entonces ¿es necesario un editor o un IDE? Como casi todas las preguntas en el contexto del desarrollo de aplicaciones web, todo depende del tipo y tamaño de la aplicación que vaya a construirse. Y no solo de eso, sino de la propia tarea que sea preciso completar en un momento dado. Sin ninguna duda, si estamos en un equipo de desarrollo trabajando en un proyecto grande con múltiples áreas de trabajo, vamos a necesitar un IDE. Por el contrario, si nuestro proyecto es pequeño, identificamos y corregimos con rapidez los errores y no necesitamos ninguna de las herramientas integradas de un IDE, ¿por qué gastar recursos en herramientas que no vamos a usar?

HERRAMIENTAS: Editor de texto (4)

¿Qué editor de código elijo? Debemos fijarnos en lo siguiente:

- **Funciones básicas imprescindibles**: resaltado de sintaxis, sangría automática, finalización automática, coincidencia de bloques y visualización de números de línea.
- **Experiencia del usuario**: reducir todo lo posible las dificultades para escribir código y aumentar así la eficiencia en la producción de código final.
- **Facilidad de aprendizaje**: invertir tiempo en entender el editor cuando aún no se domina el lenguaje es una auténtica pérdida de tiempo, por lo que sería deseable descartar en el inicio aquellos editores menos intuitivos.
- **Extensibilidad**: que proporcione vías para extender las capacidades del editor con más funciones y herramientas adicionales.
- **Velocidad**: la rapidez es clave a la hora de codificar y no todos los editores responden bien cuando se realicen ciertas tareas en equipos cortos de hardware.
- **Sistema operativo**: mucho mejor que el editor provee versiones para. al menos, Windows, Linux o MacOS.
- **Compatibilidad con Git**: dado que Git se ha convertido en un estándar de facto como sistema de control de versiones, sería deseable que incorporará esta funcionalidad de forma nativa o como extensión.
- **Soporte de la comunidad**: tener un lugar muy frecuentado para hacer preguntas y obtener soporte técnico rápido es crucial para la mayoría de los desarrolladores

Continúa...

HERRAMIENTAS: Editor de texto (5)

- **Compatibilidad con lenguajes de programación**: es prácticamente seguro que será necesario escribir ficheros con distintos lenguajes en un mismo proyecto. Por lo que es imprescindible elegir un editor que proporcione todas las herramientas imprescindibles para todos ellos.
- **Precio**: existen muchos editores de altísima calidad que son completamente gratuitos, otros aportan funcionalidades *premium* y un pequeño número de ellos tienen planes de pago. Siempre es recomendable analizar que ofrecen porque en ocasiones una pequeña inversión puede resultar en una sabia decisión que marque la diferencia.
- **Atajos de teclado**: los métodos abreviados de teclado ayudan mucho a los desarrolladores a crear fragmentos de código más rápido sin necesidad de utilizar el ratón constantemente.
- **Ventana de vista previa**: es otra característica interesante que puede ahorrar mucho tiempo evitando cambiar en proyector y navegador cuando se realizan pequeños cambios.
- **Gestión amigable de errores**: los editores que incorporan un sistema de coloreado e información ampliada de los errores facilitan de forma muy significativa la localización de errores y los problemas de código.

HERRAMIENTAS: Editor de texto (6)

Haz una lista de:

- **Editores de código**
- **Entornos de desarrollo integrados**
- **Editores online**

Básate en opiniones de usuarios y en todas las características mencionadas anteriormente y, finalmente, quédate con uno de cada tipo.

Te recomiendo: **Sublime Text** y **Visual Studio Code** como editores de código.

HERRAMIENTAS: Intérprete de Javascript

Evidentemente para escribir código JavaScript es necesario tener el propio intérprete del lenguaje, que permitirá probar los programas para saber si lo que se ha escrito es correcto o no.

Por la propia naturaleza de esta tecnología no es preciso realizar ninguna instalación adicional si se dispone de un navegador web, puesto que **JavaScript viene integrado en ellos**. Lo que sí debe comprobarse es que esté **habilitado**, porque en muchas ocasiones los usuarios, consciente o inconscientemente, lo tienen deshabilitado. A continuación, se explica cómo habilitarlo en los cuatro principales navegadores.

- **Chrome**: entrar en la **Configuración**, seleccionar **Seguridad y privacidad**, clicar en **Configuración de sitios** y después en **JavaScript** y finalmente seleccionar **Los sitios pueden usar JavaScript**.
- **Safari**: acceder a **Herramientas**, clicar en **Preferencias**, acceder a **Seguridad** y activar la casilla etiquetada como **Activar JavaScript**.
- **Edge**: acceder al menú de **Opciones** y desplegar **Mostrar opciones avanzadas**. En la sección de **Privacidad** clicar en **Configuración de contenido**. Navegar hasta la sección **JavaScript** y finalmente clicar en **Permitir que todos los sitios ejecuten JavaScript (recomendado)**.
- **Firefox**: hacer clic en **Herramientas**, luego en **Opciones** y en la pestaña **Contenido** hacer clic en la casilla **Activar JavaScript**.

En todos los casos, debe reiniciarse el navegador para tener un intérprete que es capaz de ejecutar los programas JavaScript.

Integración de Javascript en HTML (1)

Ahora que ya conoces las herramientas que puedes utilizar para comenzar a programar en JavaScript, vamos a ver la forma de **integrar el código de JavaScript** en tu código HTML.

Los navegadores web te permiten varias opciones de inserción de código de JavaScript. Podremos insertar código usando las etiquetas **<script> </script>** y empleando un atributo **type** indicaremos qué tipo de lenguaje de script estamos utilizando:

Por ejemplo:

```
<script  type="text/javascript">
//  El código de JavaScript vendrá aquí.
</script>
```

Integración de Javascript en HTML (2)

Otra forma de integrar el código de JavaScript es **incrustar un fichero externo que contenga el código de JavaScript**.

Ésta sería la **forma más recomendable**, ya que así se consigue una **separación entre el código y la estructura de la página web** y, como ventajas adicionales, podrás compartir código entre diferentes páginas, centralizarlo para la depuración de errores, tendrás mayor claridad en tus desarrollos, más modularidad y conseguirás que las páginas carguen más rápido. La rapidez de carga de las páginas se consigue al tener el código de JavaScript en un fichero independiente, ya que si más de una página tiene que acceder a ese fichero lo cogerá automáticamente de la caché del navegador, por lo que se acelerará la carga de la página.

Para ello tendremos que añadir a la etiqueta **script** el atributo **src**, con el nombre del fichero que contiene el código de JavaScript. Generalmente los ficheros que contienen texto de JavaScript tendrán la extensión **.js**

Por ejemplo:

```
<script  type="text/javascript"  src="tucodigo.js"></script>
```

Si necesitas cargar más de un fichero .js repite la misma instrucción cambiando el nombre del fichero. Las etiquetas de <script> y </script> son obligatorias a la hora de incluir el fichero .js.

Atención: no escribas ningún código de JavaScript entre esas etiquetas cuando uses el atributo src .

Integración de Javascript en HTML (3)

Para **referenciar el fichero origen .js** de JavaScript dependerá de la localización física de ese fichero. Por ejemplo, en la línea anterior, el fichero tucodigo.js deberá estar en el mismo directorio que el fichero .html. Podrás enlazar fácilmente a otros ficheros de JavaScript localizados en directorios diferentes de tu servidor o de tu dominio. Si quieres hacer una referencia absoluta al fichero, la ruta tendrá que comenzar por http://, en otro caso tendrás que poner la ruta relativa dónde se encuentra tu fichero .js

Ejemplos:

```
<script  type="text/javascript"  src="http://www.tudominio.com/ejemplo.js"></script>
<script  type="text/javascript"  src="../js/ejemplo.js"></script>
```

Ruta relativa: el fichero ejemplo.js se encuentra en el directorio anterior (../) al actual, dentro de la carpeta js/

Integración de Javascript en HTML (4)

Te estarás preguntando: ¿dónde pongo todos estos códigos dentro del archivo HTML?

1. **En el encabezado (head)**: puedes colocar la etiqueta <script> dentro de la sección <head> del documento HTML. Esto generalmente se hace cuando el código JavaScript **necesita cargarse antes** de que se procesen y muestren otros elementos de la página.

   Por ejemplo:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de Script en el encabezado</title>
    <script src="mi_script.js"></script>
</head>
<body>
    <!-- El contenido de la página va aquí -->
</body>
</html>
```

Integración de Javascript en HTML (5)

2. **Antes de cerrar el cuerpo (body) del documento HTML**: esto se hace a menudo cuando deseas que **la página se cargue primero** y luego el código JavaScript se ejecute. Esto puede ayudar a mejorar el rendimiento de la página web, ya que el contenido se mostrará antes de que se cargue y ejecute el código JavaScript.

   Por ejemplo:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de Script antes de cerrar el cuerpo</title>
</head>
<body>
    <!-- El contenido de la página va aquí -->
    <script src="mi_script.js"></script>
</body>
</html>
```

Integración de Javascript en HTML (6)

En ambos casos, debes especificar la ruta al archivo JavaScript que deseas incluir en el atributo src de la etiqueta <script> o bien escribir el código JavaScript directamente entre las etiquetas <script> si no estás utilizando un archivo externo.

Es importante tener en cuenta que, en general, es una **buena práctica** colocar los scripts **al final del cuerpo (</body>)** para que la página se cargue **más rápido** y sea más eficiente. Sin embargo, dependiendo de tus necesidades y requerimientos específicos, puedes optar por colocarlos en el encabezado si es necesario.

Integración de Javascript en HTML (7)

Si incluyes tu código JavaScript en un fichero en lugar de escribirlo directamente en el archivo HTML, cuando alguien examine el código fuente de tu página web verá el enlace a tu fichero .js, en lugar de ver el código de JavaScript directamente. Esto no quiere decir que tu código sea inaccesible, ya que simplemente copiando la ruta de tu fichero .js y pegándolo en el navegador, se podrá descargar el fichero .js y ver todo el código de JavaScript. En otras palabras, **nada de lo que tu navegador descargue para mostrar la página web podrá estar oculto de la vista** de cualquier programador.

Seguramente estarás pensando en cómo puedes proteger el código de JavaScript que vas a programar del uso fraudulento por otros programadores o visitantes a tu página: la respuesta rápida a esa pregunta es que **es imposible hacerlo**.

Para que el código de JavaScript pueda ejecutarse correctamente deberá ser cargado por el navegador web y, por lo tanto, su código fuente deberá estar visible al navegador. Si realmente te preocupa que otras personas usen o roben tus scripts, deberías incluir un **mensaje de copyright** en tu código fuente. Piensa que no solamente tus scripts son visibles al mundo, sino que los scripts del resto de programadores también lo son. De esta forma puedes ver fácilmente cuándo alguien está utilizando tus scripts, aunque ésto no evita que alguien copie tu código y elimine tu mensaje de copyright.

Integración de Javascript en HTML (8)

Lo más que se puede hacer es **ofuscar el código**, o **hacerlo más difícil de entender**. Las técnicas de ofuscación incluyen la eliminación de saltos de línea, espacios en blanco innecesarios, tabuladores, utilización de nombres ininteligibles en las funciones y variables, utilización de variables para almacenar trozos de código, uso de recursividad, etc. La forma más rápida de hacer todas esas tareas de ofuscación es utilizar un software que producirá una copia "comprimida" del script que has programado para facilitar su carga rápida.

Para que veas un ejemplo de ofuscador de JavaScript visita esta web.

De todos modos, lo mejor es que cambies esa forma de pensar y actúes de manera diferente. En lugar de proteger tu código, ¡promociónalo! Añade comentarios útiles en el código fuente, publícalo en tu blog o en webs de programación, etc. Incluso podrás añadir una **licencia Creative Commons** para animar a la gente a que lo utilice, lo copie y lo mantenga público al resto del mundo. Así conseguirás mayor reputación como buen programador y la gente contactará contigo para más información, posibles trabajos, etc. Aquí puedes consultar sobre este tipo de licencias, si tienes curiosidad.