



I.E.S. ROSA CHACEL

Desarrollo de Aplicaciones Web



Unión Europea

Fondo Social Europeo
"El FSE invierte en tu futuro"

2º DAW

DESARROLLO WEB EN ENTORNO CLIENTE

UT 1: Arquitecturas y herramientas de programación.

Profesora: Irene Rodil Jiménez

0. Contenidos



- ⇒ 1. Introducción
- ⇒ 2. Modelos de programación en entornos cliente/servidor
- ⇒ 3. Navegadores web
- ⇒ 4. Herramientas de programación
- ⇒ 5. Integración del código con las etiquetas HTML

1. Introducción



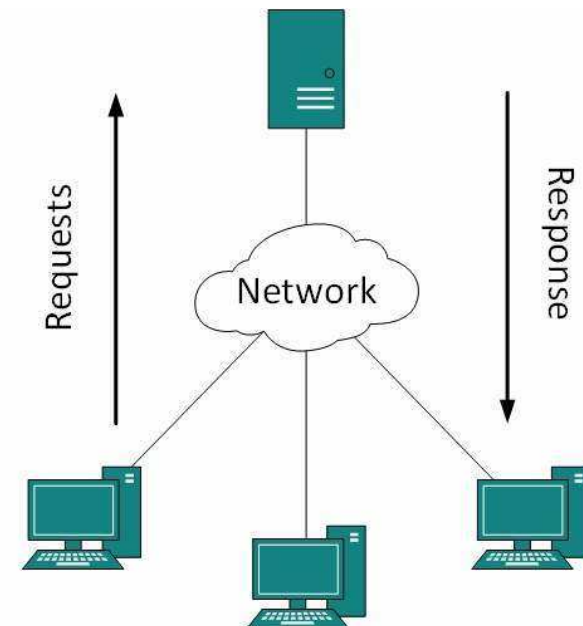
- La Web se desarrolló en 1990 por Tim Berners-Lee, fue publicada en 1992 y en 1993 el CERN anunció que sería gratuita para todos.
- En pocos años la Web ha evolucionado enormemente: se ha pasado de páginas sencillas, con pocas imágenes y contenidos estáticos a páginas complejas con contenidos dinámicos que provienen de bases de datos, lo que permite la creación de **aplicaciones web**.
- Históricamente, en el desarrollo web se trasladaba toda la interacción entre el usuario y la aplicación al servidor web → el usuario utiliza su navegador web para enviar sus peticiones a un servidor que las procesa y le devuelve el resultado que se mostraba en el navegador.
- Las mejoras hardware han hecho que esta tendencia varíe. Los usuarios finales disponen de máquinas con mayor capacidad de procesamiento, por tanto, ya no es necesario que sea el servidor quien se ocupe de la mayor carga de procesamiento.
- La tendencia actual es desarrollar aplicaciones web que trasladen la mayor carga de procesamiento al navegador web del cliente → se reduce la constante necesidad de intercambiar información con el servidor.

2. Modelos de programación en entorno cliente/servidor



- Los sitios web siguen un modelo basado en la programación **cliente/servidor** que dispone de tres elementos:
 - **Lado del servidor**: incluye el hardware y software del **servidor Web** así como diferentes elementos de programación y tecnologías incrustadas.
 - **Lado del cliente**: 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.
 - **Red**: describe los diferentes elementos de conectividad utilizados para mostrar el sitio web al usuario.

La interacción entre clientes y servidores es a través del intercambio de mensajes, utilizando el protocolo HTTP o **HTTPS**



2. Modelos de programación en entorno cliente/servidor



APLICACIONES WEB

➤ Páginas estáticas:

- En los comienzos de la web, todos los sitios web eran conjuntos de páginas web en forma de ficheros HTML, se dice que eran páginas estáticas.
- Los sitios web eran como libros pero con navegación mediante enlaces en vez de navegación secuencial.
- La edición de sitios web se realizaba con herramientas similares a la edición de documentos (por ejemplo, *Microsoft FrontPage*).

➤ Páginas dinámicas:

- Poco a poco las páginas comenzaron a ser más dinámicas, en vez de ser simples ficheros HTML almacenados en el disco duro, empezaban a ser pequeños programas que se ejecutaban cada vez que un cliente solicitaba una página.
- Inicialmente eran cambios mínimos como contadores de visitas, mostrar la fecha actual, Se realizaban con programación del lado servidor.

➤ Aplicaciones web:

- Las páginas web se han convertido en aplicaciones web, con programación tanto del lado servidor como del lado cliente.
- Las aplicaciones web tienen la ventaja de que son independientes del sistema operativo, únicamente dependen del navegador del cliente.

2. Modelos de programación en entorno cliente/servidor



DESARROLLO WEB EN EL LADO SERVIDOR

- El desarrollo web en el lado servidor también se conoce como **back-end**.
- El desarrollador debe dominar algún lenguaje del lado servidor (su código se ejecuta en un servidor web) y estar familiarizado con las base de datos.
- El usuario accede a través de páginas HTML quedando el código fuente oculto.
- Lenguajes de programación en entorno servidor:

Lenguaje	Año	Desarrollado por
Java	1995	Sun Microsystems
Phyton	1991	Guido van Rossum, Python Software Foundation
C#	2002	Microsoft
PHP	1995	Rasmus Lerdorf
Ruby	1995	Yukihiro Matsumoto et al.
Perl	1987	Larry Wall, et al.

2. Modelos de programación en entorno cliente/servidor



DESARROLLO WEB EN EL LADO CLIENTE

- El desarrollo web en el lado cliente también se conoce como **front-end**.
- Los scripts se incluyen en el documento HTML o se escriben en un archivo separado que se enlaza al documento principal.
- A diferencia de los lenguajes del lado servidor, no es el servidor el que ejecuta y procesa los scripts, sino el **cliente** solicitante.
 - Cuando un usuario solicita una página web con un script cliente, el servidor web envía el documento HTML y el script al navegador, quien lo ejecuta y presenta el resultado final.
 - Asimismo, los scripts del lado cliente contienen instrucciones concretas para el navegador web al respecto de cómo ha de reaccionar a ciertas acciones llevadas a cabo por el usuario como, por ejemplo, un clic en un botón específico. A menudo, el cliente ha de establecer para ello otro contacto con el servidor web.
- Al ejecutarse en el navegador, el usuario puede ver el código fuente, a diferencia de lo que ocurre con los scripts del lado servidor.
- Como desventaja, como los scripts del lado cliente influyen en los tiempos de carga, existen diversas extensiones para el navegador que bloquean estos scripts.

2. Modelos de programación en entorno cliente/servidor



DESARROLLO WEB EN EL LADO CLIENTE

➤ Lenguajes de programación en entorno cliente: hay dos variantes

- "Lenguajes" que nos permiten dar formato y estilo a una página web:
 - **HTML**, **CSS**, etc.
- Lenguajes que nos permite aportar dinamismo y lógica:
 - Lenguajes de scripting como **JavaScript**.



Lenguaje	Año	Descripción
HTML	1993	Es un lenguaje de marcas, que marca el texto de modo que el ordenador pueda manipularlo. Permite poner títulos, marcar texto en negrita, subrayar, crear enlaces, etc.
CSS	1996	Hace posible formatear una página web escrita en HTML. Se puede seleccionar un elemento de una página, como un párrafo o un bloque, y definir el color, el tamaño de la letra, las dimensiones, etc.
JavaScript	1995	Es un lenguaje de programación que aporta dinamismo a una página HTML.

2. Modelos de programación en entorno cliente/servidor

























LENGUAJES DE PROGRAMACIÓN WEB MÁS REQUERIDOS

- IEEE Spectrum: ranking de 2018

Language Types (click to hide)



Language Rank	Types	Jobs Ranking
1. Python	  	100.0
2. Java	  	99.2
3. C	  	98.8
4. C++	  	94.6
5. C#	  	86.2
6. JavaScript	 	85.7
7. Assembly		83.4
8. PHP		83.1
9. HTML		81.3
10. Scala	 	76.5

2. Modelos de programación en entorno cliente/servidor



LENGUAJES DEL LADO SERVIDOR O DEL LADO CLIENTE

- Al estructurar un proyecto web se deberá definir en que lugar se van a ejecutar los scripts.
- Si los scripts solo se ejecutan en el navegador:
 - Más ligera es la página para el servidor web.
 - Puede acarrear peor rendimiento para el usuario.
 - El desarrollo tiene un grado de complejidad mayor si únicamente se utilizase JavaScript.
 - Es necesario que el navegador soporte todas las funciones del lenguaje y que el usuario no utilice extensiones de bloqueo.
- Para no cargar excesivamente ni un lado ni otro → lo idóneo es optar por una buena combinación de lenguajes del lado servidor y del lado cliente.
 - Y utilizar tecnologías como **AJAX** para transferir datos de forma asíncrona entre cliente y servidor → el servidor web puede reaccionar a peticiones del lado cliente casi en tiempo real e intercambiar información con el navegador sin tener que cargar la página por completo.

2. Modelos de programación en entorno cliente/servidor



PERFILES DESARROLLADOR WEB

- Actualmente se habla mucho de 3 perfiles diferenciados en el ámbito del desarrollo web: Front-end VS Back-end

- Perfiles:



- **Front-end:** se encarga del diseño y maquetación de la aplicación web utilizando tecnologías como *HTML*, *CSS* y *Javascript* (y sus *frameworks*). Debe preocuparse también de la correcta presentación en cualquier tipo de dispositivo e incluso del posicionamiento en buscadores.
- **Back-end:** se encarga de la programación del lado servidor utilizando tecnologías como *PHP*, *ASP.NET*, *JSF*, *Python*, También se encarga de la administración del servidor de aplicaciones y la Base de Datos.
- **Full stack:** engloba los dos anteriores. El desarrollador quizás no es un experto de ninguna tecnología concreta pero tiene amplios conocimientos de todo el conjunto y es capaz de colaborar en cualquiera de las partes.

2. Modelos de programación en entorno cliente/servidor



DESARROLLADOR WEB FRONT-END

- Su trabajo no es el de diseñador Web, aunque generalmente ayuda que tenga conocimientos y/o buen gusto para el diseño.
 - Aunque algunas empresas pueden pedir *frontend developers* que sepan manejar *Illustrator*, *Photoshop* y herramientas similares.
- **Tecnologías:**
 - **CSS** y **HTML**: que aunque no son lenguajes de programación requieren aplicar mucha lógica para crear diseños adaptables a distintos dispositivos (*Responsive*), soportados por múltiples navegadores y con diseños desafiantes.
 - **JavaScript**: uno de los lenguajes de programación más utilizado.
 - **Ajax**: conjunto de técnicas y métodos de desarrollo web.
 - **DOM**: modelo de objetos del documento que consta de una API para manipular el documento HTML permitiendo la gestión de eventos, o la inserción y eliminación de elementos.
 - **Frameworks**: *ReacJS* de Facebook, *Angular* de Google, *Vue.js*
 - **Librerías**: *jQuery*, *Prototype*, *Motools*
 - **API's de HTML**: *Geolocalización*, *LocalStorage*, *Canvas*, ...
- Un ingeniero de desarrollo *frontend* se especializa en rendimiento, en performance, en frameworks frontend, ...

3. Navegadores web



CLIENTE WEB

- Al navegador web (browser) se le llama también cliente web ya que hace las tareas de solicitud y consumo de servicios.
 - El navegador se conecta con un servidor al que solicita una página web.
 - El servidor web sirve al cliente la página solicitada.
- Un navegador web es una aplicación, distribuida normalmente como software libre, que permite a un usuario acceder a un recurso publicado por un servidor web a través de Internet. Para ello se utiliza una dirección URL, luego traduce el código HTML en el que está escrita la página y muestra el contenido en pantalla (texto, imágenes, vídeo,...).
- El navegador tiene la capacidad de interpretar varios lenguajes de programación especialmente creados para la visualización y maquetación de contenido.
- Los estándares web son un conjunto de recomendaciones dadas por el **World Wide Web Consortium (W3C)** y otras organizaciones internacionales acerca de cómo crear e interpretar documentos basados en la web. Su objetivo es crear una web con sitios accesibles a más personas y que funcionen en cualquier dispositivo de acceso a Internet.

3. Navegadores web



EVOLUCIÓN DE LOS NAVEGADORES

Navegador	Año	Descripción
Nexus	1991	Creado por Tim Berners-Lee, es el primer navegador de la historia lanzado inicialmente con el nombre WoldWideWeb . Funcionaba solo en modo texto para ordenadores Next.
Netscape Navigator	1994	Primer navegador comercial basado en Mosaic (creado por un becario de 23 años). Funcionaba en entorno gráfico. En un año dominó más del 80% del mercado
Internet Explorer	1995	Microsoft lanzó el sistema operativo Windows 95 que incluía el navegador gratuitamente y se instalaba automáticamente, lo que hizo que poco a poco fuera acaparando el mercado de los navegadores (hasta el 80% de cuota de mercado). Su última versión es IE 11.
Opera	1996	Creado por un grupo de investigadores de la empresa de telecomunicaciones noruega. A partir de 2000 cambiaron la licencia de uso primero mostrando anuncios y a partir de 2005 freeware. No ha tenido nunca gran cuota de mercado, Primero en introducir en el año 2000 la navegación por pestañas.

3. Navegadores web



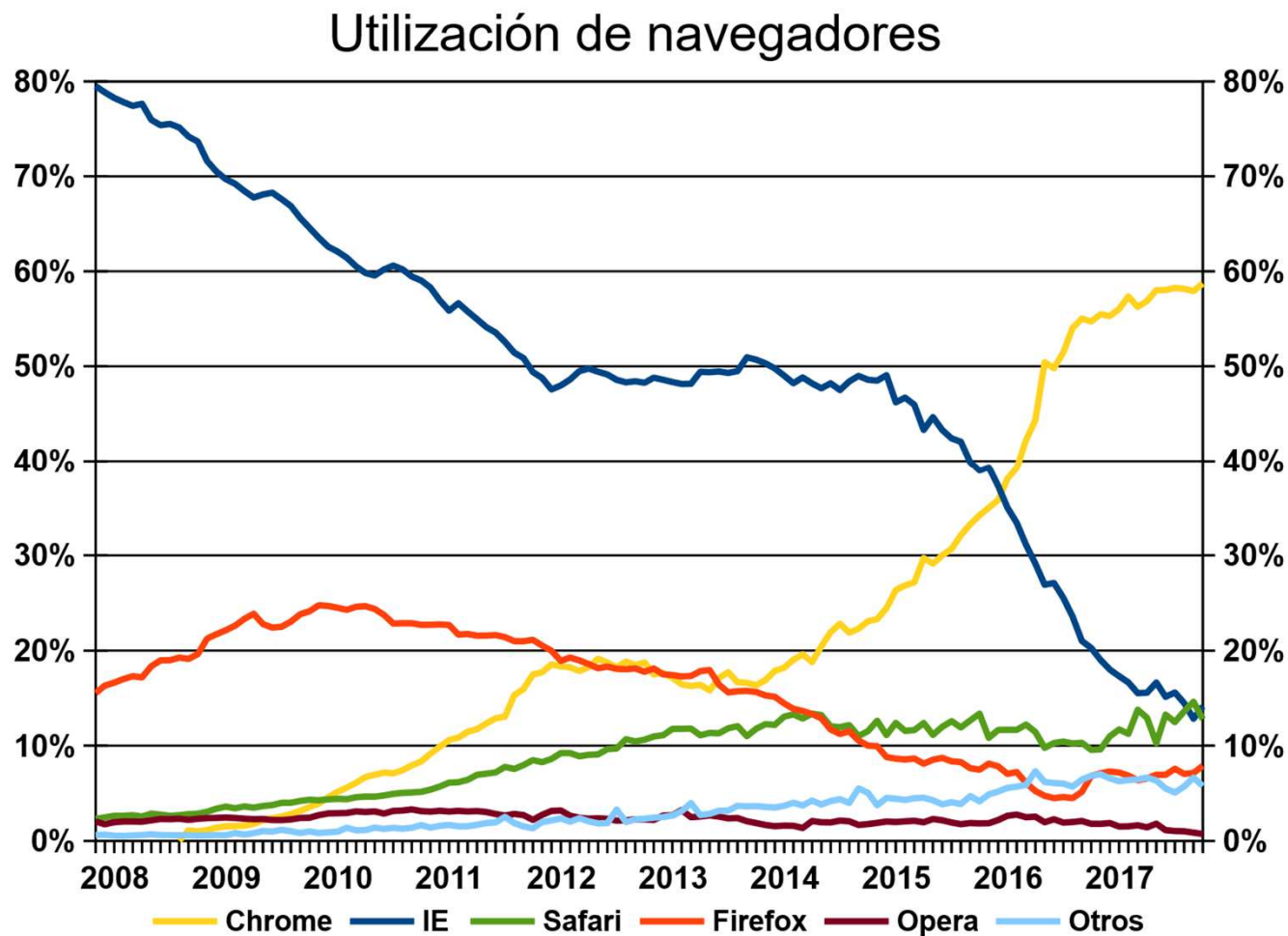
EVOLUCIÓN DE LOS NAVEGADORES

Navegador	Año	Descripción
Firefox	2002	En 1998 Netscape para frenar el éxito de IE liberó el código de su navegador y creó el proyecto Mozilla. En 2004 lanzó una versión mejorada, llamada Mozilla Firefox que ganó en poco tiempo gran popularidad gracias a sus pestañas y ser muy ligero, convirtiéndose en un gran competidor de IE. El desarrollo de Firefox está financiado principalmente por Google.
Safari	2003	Creado por Apple para sus ordenadores y dispositivos.
Google Chrome	2008	A finales de 2011 superó a Internet Explorer 8.0 como el navegador más utilizado a nivel mundial. Se destaca por su interfaz minimalista y por la velocidad de ejecución del código JavaScript.
Microsoft Edge	2015	Lanzado por Microsoft para sustituir a Internet Explorer, incluyéndolo en Windows 10 como predeterminado. Es una versión mejorada, modernizada y distinta de Internet Explorer con una línea de desarrollo independiente.

3. Navegadores web



EVOLUCIÓN DE LOS NAVEGADORES



3. Navegadores web



EVOLUCIÓN DE LOS LENGUAJES DE SCRIPT DEL LADO CLIENTE

- **Internet Explorer**: desde sus inicios utilizaba un lenguaje propio denominado **VBScript** (*VisualBasic Script*).
- **Netscape**: un empleado en Netscape (*Brendan Eich*) definió un lenguaje de programación que inicialmente se llamó *Mocha* pero fue renombrado a *LiveScript* y finalmente a **JavaScript** (por razones de marketing ya que Java estaba en auge).
 - En 1995 la compañía *Sun Microsystems* (que había adquirido Netscape y era propietaria del lenguaje de programación Java) anunció el lenguaje JavaScript y a partir de 1996 ya era operativo en el navegador Netscape.
- Desde ese momento hubo una batalla entre VBScript y JavaScript por ser el lenguaje del lado del cliente predominante, batalla que fue ganando poco a poco JavaScript.
- La compañía Netscape envió a la organización **Ecma International** la especificación para que se convirtiera en un estándar → nació ECMA-262 (conocido como **ECMAScript**). En el desarrollo del estándar trabajan hoy las principales compañías interesadas en el desarrollo del lenguaje de lado del cliente (Microsoft, Google, Apple o la fundación Mozilla).

3. Navegadores web



COMPATIBILIDAD CON NAVEGADORES

- Aunque parece simple, es bastante complicado para los desarrolladores de aplicaciones Web tener el control de todas las versiones existentes de los navegadores que tienen los usuarios. La fragmentación de posibles clientes es demasiado elevada, lo que provoca que existan muchísimos problemas en el desarrollo Web.
- JavaScript es un lenguaje de programación basado en el estándar **ECMAScript** de EMAC (otra organización diferente a W3C):
 - Pero desde que se pactan diferentes características en el estándar hasta que se implementan en los navegadores puede pasar tiempo, o incluso, algunos navegadores, incorporan características no pactadas en el estándar.
 - Por lo tanto, puede que una característica pactada en una versión pasada de ECMAScript nunca haya sido implementada en uno de los principales navegadores y por tanto el código desarrollado no se ejecutará bien.
- El desafío de crear páginas que se vean bien en todos los navegadores es real, es complicado y es para todos.

3. Navegadores web



COMPATIBILIDAD CON NAVEGADORES



➤ jQuery:

- Biblioteca creada en 2006.
- Trata de hacer multiplataforma JavaScript en todos los navegadores.
- Al ser código abierto y software libre, ha permitido que se adapte con el tiempo y hoy en día (2017) sigue siendo la biblioteca más utilizada puesto que se encuentra en aproximadamente el 96% de la Web.

➤ Frameworks del lado cliente:

- Aunque jQuery es una herramienta muy potente, no hay que olvidar que es una biblioteca construida sobre JavaScript para paliar el problema que provocan los fabricantes de navegadores en su no estandarización → la creación de aplicaciones complejas sigue siendo complejo y costoso.
- Han surgido en los últimos años frameworks en JavaScript para facilitar la creación de aplicaciones de gran envergadura.
- Ventajas: desarrollos más rápidos, optimización del rendimiento, reducción costes, ...
- Principales Frameworks: **Angular, React, Vue.js**

3. Navegadores web



CONFIGURACIÓN DEL NAVEGADOR CHROME

➤ Acceder a la configuración:

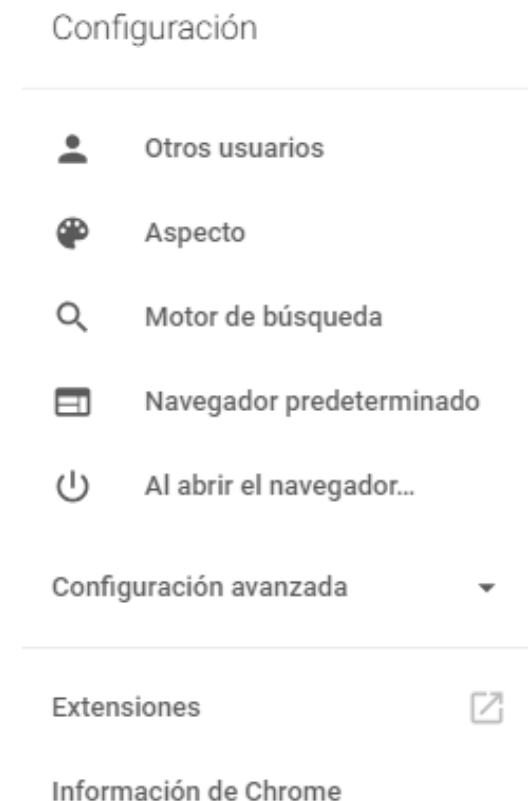
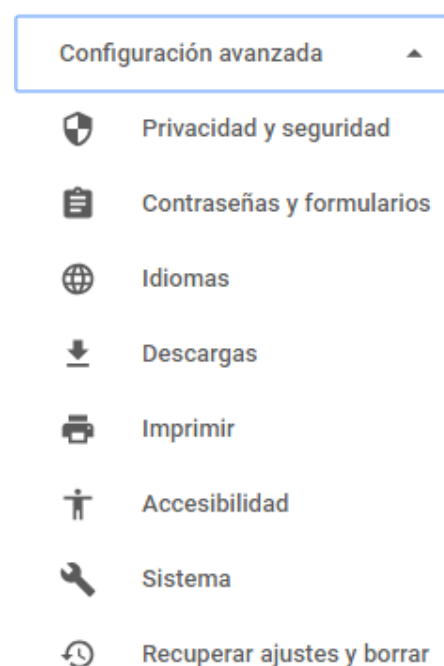
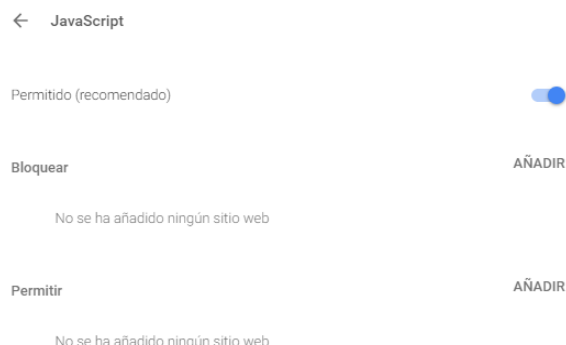
Chrome | chrome://settings

- Click en el icono **Personalizar y controlar Google Chrome** (botón de menú situado en la esquina superior derecha)
- Seleccionar la opción **Configuración**.

➤ Configuración:

- Habilitación de JavaScript:

- Configuración avanzada
- Privacidad y seguridad
- Configuración de contenido
- JavaScript



- Extensiones: instala nuevas extensiones desde *Chrome Web Store*

4. Herramientas de programación



- JavaScript no requiere que instalemos y aprendamos a usar complejos entornos de desarrollo para escribir código en nuestras páginas. Tan solo es necesario un editor de texto y un navegador Web.
- Herramientas:
 - Navegador Web: vamos a utilizar **Chrome**.
 - Los navegadores suelen incluir **Herramientas de desarrollo** para ayudarnos en la búsqueda de errores, Vamos a utilizar principalmente:
 - **Consola**: deja ver los errores y advertencias que han surgido al ejecutar el código y permite ejecutar pequeños fragmentos de código escribiendo directamente en ella.
 - **Depurador**: permite ejecutar el código línea por línea y, al mismo tiempo, ver los cambios de valor que se producen en las variables y otros elementos.
 - Editores de texto: vamos a utilizar **Visual Studio Code**.
 - Los editores avanzados de código colorean la sintaxis e incluyen una ayuda predictiva según se escribe, para no tener que recordar todos los nombres de funciones y propiedades.
 - Permite mediante extensiones realizar depuración de código.

4. Herramientas de programación



ELEGIR UN NAVEGADOR WEB

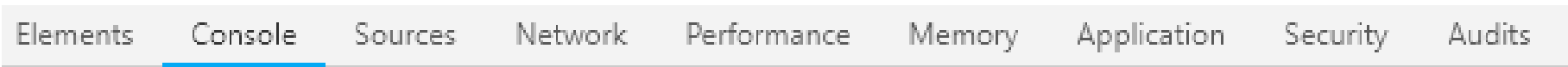
- En este curso utilizaremos **Chrome** ya que es el navegador con mayor número de usuarios y por tanto, es el primer navegador en el que tenemos que asegurarnos de que nuestro código funcione perfectamente.
- Desventajas:
 - Consumo de recursos. Es casi imposible de utilizar con ordenadores de menos de 4GB.
 - Con **Mayus + Esc** se abre el administrador de tareas de Chrome → podremos ver el consumo de las pestañas que tengamos abiertas y de las extensiones.
 - Crea un proceso por pestaña, por lo que no optimiza su velocidad. Firefox crea cuatro procesos que se reutilizan conforme abrimos pestañas. Esto ayuda a **controlar el consumo de memoria** mejora la velocidad.
 - Extensiones maliciosas en su Chrome Web Store.
 - Problemas de seguridad.
- Ventaja:
 - Permite añadir aplicaciones web y páginas al escritorio y abrirlas como ventanas independientes → permite no tener tantas pestañas abiertas.
Btn derecho → Mas herramientas → Agregar al escritorio

4. Herramientas de programación



HERRAMIENTAS PARA DESARROLLADORES

- **Código fuente:** se puede visualizar el código fuente de la página web
 - **Ctrl + U**
 - En muchas ocasiones este sistema no es práctico puesto que la página web tiene el código minificado para que ocupe menos.
- **DevTools:** herramientas de desarrollo que proporcionan los navegadores para facilitar el trabajo de los desarrolladores.



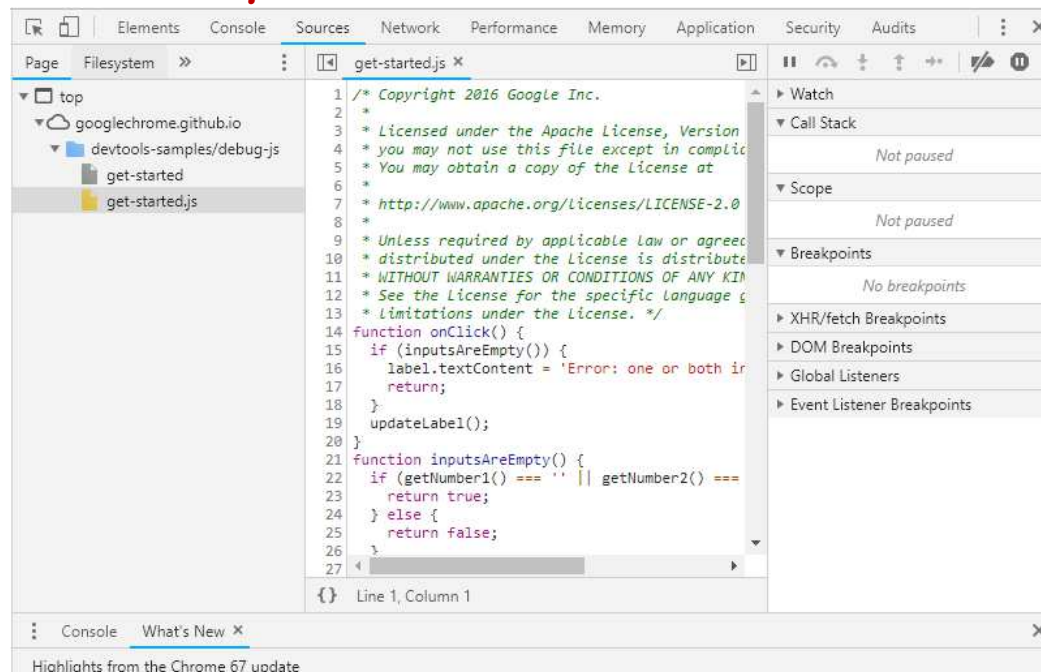
- Inspeccionar elementos (**Elements**): permite ver su código fuente sin minificar. También permite hacer cambios en el HTML y CSS y visualizarlos.
- **Console**: permite registrar información de diagnóstico durante el desarrollo o usarlo como un *shell* para interactuar con el código JavaScript en la página.
- **Sources**: permite depurar código JavaScript con puntos de interrupción.
- **Application**: permite almacenamiento local de datos, acceder a las cookies, ...
- **Network**: permite obtener información sobre recursos solicitados y descargados, y optimizar el rendimiento de carga de tu página.

4. Herramientas de programación



CHROME DEVELOPERS TOOLS

- Se puede abrir la herramienta para desarrolladores de dos formas:
 - **Ctrl + Mayusc. + I o F12**
 - **Mas herramientas → Herramientas para desarrolladores.**
 - **Btn derecho → Inspeccionar elemento**



- Vamos a aprender a **depurar** en la siguiente página:
<https://developer.chrome.com/docs/devtools/javascript/>

4. Herramientas de programación



ELEGIR UN EDITOR DE TEXTO

- Para programar en JavaScript se podría utilizar cualquier editor de texto, pero existen editores especialmente diseñados para programadores.
 - Con estos editores, las diferentes partes del código aparecen resaltadas, se autocompletan partes que se repiten y, en general, te facilita mucho la vida a la hora de programar.
- Los "mejores" editores para desarrolladores web son:
 - Visual Studio Code: es de Microsoft pero gratuita y de código abierto.
 - Brackets: desarrollado por Adobe Systems utilizando tecnologías como JavaScript, HTML y CSS. Ofrece vista previa en vivo sin instalar complementos
 - Atom: desarrollado por GitHub, algunos lo consideran un IDE por la cantidad de funciones que incorpora.
 - Sublime Text: es un software propietario. Tuvo una época de mucho éxito.
 - Eclipse con JSDT: eclipse a través del complemento JSDT (JavaScript Developer Tool) ofrece miles de funciones para escribir aplicaciones basadas en JavaScript.
- Editores online: para experimentar y compartir código
 - Permiten escribir código y observar el resultado inmediatamente.
 - Los más utilizados son **JSFiddle**, **JS Bin** y **Plunker**

4. Herramientas de programación



VISUAL STUDIO CODE

- Es un editor de código ligero pero potente.
- Desarrollado por Microsoft en 2015. Aunque tiene licencia MIT (licencia de software libre permisiva, con pocas limitaciones).
- Características:
 - Es una aplicación de escritorio multiplataforma (*Windows, macOS y Linux*).
 - Tiene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros idiomas (como C++, C#, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity).
 - Se actualiza constantemente.
 - Intellisense: capacidad de predecir la instrucción que vamos a escribir → permite autocompletar.
 - Se puede extender a través de complementos, disponibles a través de un repositorio central
 - La integración con **Git** (sw de control de versiones) viene por defecto.
 - Tecnología **Live Share** integrada: permite trabajar, depurar y hasta compartir línea de comandos en tiempo real con varios programadores, hablando y chateando al mismo tiempo.

4. Herramientas de programación



VISUAL STUDIO CODE

➤ Interfaz de usuario:

- La interfaz de usuario y distribución de los paneles es muy similar a la de otros editores como *Sublime Text*, *Atom*, *Brackets*, ...
 - A la izquierda el editor de archivos
 - A la derecha el contenido del archivo que se está editando

- 5 áreas:

- **Editor:** parte central donde se editan los archivos (máximo tres editores uno al lado del otro). Pulsa **Alt** y haz clic en un segundo o tercer archivo.
- **Panel lateral:** por defecto aparece la vista del explorador de archivos. El explorador permite trabajar fácilmente con los ficheros del proyecto.
- **Barra de estado:** situada en la parte inferior. Muestra distinta información como el número de líneas y caracteres del archivo, ...
- **Panel de actividades:** se encuentra en la parte izquierda de la aplicación. Cuenta con 5 botones que permiten cambiar entre diferentes vistas: explorador, búsqueda, control de versiones, depuración y extensiones.
- **Paneles:** en la parte inferior del editor se pueden mostrar diferentes paneles donde se recogen información de depuración, errores, avisos o un terminal integrado. Esta posición no es estática, se pueden desplazar estos paneles a la derecha para tener mas espacio vertical.



5. Integración del código con las etiquetas HTML



- Para desarrollar aplicaciones web que se ejecuten en el lado cliente necesitamos un documento base escrito en HTML.



```
1  <!DOCTYPE html>  
2  <html lang="">  
3  <head>  
4    <meta charset="utf-8">  
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
6    <title></title>  
7  </head>  
8  
9  <body>  
10  
11 </body>  
12 </html>  
13
```

- La integración de JavaScript y HTML se puede realizar de varias formas:
 - 1. JavaScript en el mismo documento HTML.
 - 2. JavaScript en un archivo externo.
 - 3. JavaScript en elementos HTML.

5. Integración del código con las etiquetas HTML



1. JAVASCRIPT EN EL MISMO DOCUMENTO HTML

- Mediante el uso de unas etiquetas predefinidas para marcar el texto: **<script> y </script>**
- Puede incluirse en cualquier parte del documento. Tradicionalmente se ponía en la cabecera, luego por motivos de optimización de carga se ponía al final del *body*. Actualmente, se pone de nuevo en la cabecera.
- Esta técnica suele utilizarse cuando se definen instrucciones que se referenciarán desde cualquier parte del documento o cuando se definen funciones con fragmentos de código genéricos.
- **Desventaja**: si lo queremos utilizar en otras páginas, debemos incluirlo en cada una de ellas → inconveniente para las modificaciones de código.

```
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Ejemplo1</title>
7      <script>
8          alert("Prueba de JavaScript");
9      </script>
10 </head>
11
12 <body>
13     <h1>Ejemplo 1: código embebido</h1>
14 </body>
15 </html>
```


5. Integración del código con las etiquetas HTML



2. JAVASCRIPT EN UN ARCHIVO EXTERNO

- Las mismas instrucciones de JavaScript que se incluyen entre un bloque `<script></script>` pueden almacenarse en un fichero externo con extensión **.js**.
 - Ejemplo: el archivo `mensaje.js` con la línea de código:
`alert("Prueba de JavaScript");`
- La forma de acceder y enlazar esos ficheros **.js** con el documento HTML es a través de la etiqueta `<script>` con el atributo **src** (ruta del archivo).
- No existe un límite en el número de ficheros **.js** que pueden enlazarse en un mismo documento HTML, pero solo se puede enlazar un archivo en cada etiqueta.

```
1  <!DOCTYPE html>
2  <html lang="es">
3
4  <head>
5      <meta charset="utf-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Ejemplo 2</title>
8      <script src="/js/mensaje.js"></script>
9  </head>
10
11 <body>
12     <h1>Ejemplo 2: fichero externo</h1>
13 </body>
14
15 </html>
```

5. Integración del código con las etiquetas HTML



2. JAVASCRIPT EN UN ARCHIVO EXTERNO

Fichero externo

- `<script type="text/javascript" src="ruta/archivo.js"></script>`
- `<script type="text/javascript" src="../js/archivo.js"></script>`
- `<script type="text/javascript" src="http://www.dominio.com/archivo.js"></script>`

Ventajas de usar un fichero externo

- Carga más rápida de páginas.
- Separación entre estructura y comportamiento.
- Compartición de código entre páginas.
- Facilidad para depuración de errores.
- Modularidad.
- Seguridad.

5. Integración del código con las etiquetas HTML



¿EN QUÉ LUGAR DEL CÓDIGO COLOCAR LA ETIQUETA SCRIPT?

➤ Antiguamente:

- Los navegadores analizaban los archivos HTML secuencialmente (línea a línea).
- Cuando encontraba una etiqueta `<script>` que referenciaba a un archivo de scripting externo → el navegador detenía el análisis del documento, descargaba el script, lo ejecutaba y luego continuaba con el análisis del resto del HTML de la página.
- Por esta razón se solían colocar los `<script>` al final del documento, antes de cerrar el `<body>`, para que el usuario no tuviese que esperar a que todos los script se descargasen.
- Problema: el navegador no puede empezar a descargar el script hasta que el documento esté analizado completamente. Poder descargar lo más rápido posible es muy importante para el rendimiento de sitios grandes con scripts y hojas de estilos pesadas,

➤ Actualmente:

- Todos los navegadores modernos a partir del 2008 incorporaron un sistema llamado **preload scanning** que les permite cargar todos los scripts paralelamente sin causar bloqueos.
- Poner los scripts dentro del `<head>` y usar los atributos *async* o *defer*.

5. Integración del código con las etiquetas HTML



¿EN QUÉ LUGAR DEL CÓDIGO COLOCAR LA ETIQUETA SCRIPT?

➤ HTML5, etiquetas async y defer:

- **1. `<script>`** (normal): el análisis HTML se detiene, se descarga el archivo (si es un script externo), se ejecuta el script y después se reanuda el análisis HTML.
- **2. `<script async>`**: el script se descarga de forma asíncrona, es decir, sin detener el análisis HTML, pero una vez descargado, si se detiene para ejecutar el script. Tras la ejecución se reanuda el análisis HTML. Sigue existiendo un bloqueo en el renderizado pero menor que con el comportamiento normal. **No se garantiza la ejecución de los scripts asíncronos en el mismo orden en el aparecen en el documento.**
- **3. `<script defer>`**: el script se descarga de forma asíncrona, en paralelo con el análisis HTML, y además su ejecución es diferida hasta que termine el análisis HTML. No hay bloqueo en el renderizado HTML. La ejecución de todos los scripts diferidos se realiza en el mismo orden en el que aparecen en el documento. Parece la mejor opción de forma general. Salvo que el script manipule o interaccione con el DOM
- **Nota:** *async* y *defer* son atributos válidos solo para elemento `<script>` con un *src* establecido.

5. Integración del código con las etiquetas HTML



¿EN QUÉ LUGAR DEL CÓDIGO COLOCAR LA ETIQUETA SCRIPT?

➤ HTML5, etiquetas async y defer:

■ Análisis HTML ■ Descarga script ■ Ejecución script ■ Análisis HTML pausado



5. Integración del código con las etiquetas HTML



3. JAVASCRIPT EN ELEMENTOS HTML

- Consiste en insertar fragmentos de JavaScript dentro de atributos de etiquetas HTML de la página.
- Suele utilizarse como forma de controlar los eventos que suceden asociados a un elemento HTML concreto.
- **Principal desventaja**: el mantenimiento y modificación del código puede resultar más complicado.

```
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Ejemplo 3</title>
7  </head>
8
9  <body>
10     <p onclick="alert('Prueba de JavaScript');">Ejemplo 3: código en atributos</p>
11 </body>
12 </html>
```

5. Integración del código con las etiquetas HTML



PROTECCIÓN DEL CODIGO

- Para que el código de JavaScript pueda ejecutarse correctamente deberá ser cargado por el navegador web → deberá estar visible al navegador.
- Por tanto, no se puede proteger el código de JavaScript que vas a programar del uso fraudulento.
- **Copyright**: puedes incluir mensajes de copyright en el código.
- Lo mejor es pensar de otra manera y en lugar de intentar proteger el código intenta promocionarlo, publicándolo en tu blog o en webs de programación.
- Añade 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.
 - **BY-NC-SA**: puedes copiar, difundir y remezclar los contenidos siempre y cuando no se obtenga un beneficio económico de ellos, y siempre que se utilice el mismo tipo de licencia: la CC-BY-NC-SA. Hay que citar y enlazar la fuente y el autor del contenido que se utilice.



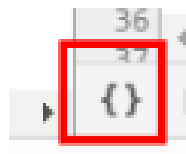
5. Integración del código con las etiquetas HTML



PROTECCIÓN DEL CODIGO

➤ Minificación y ofuscación del código:

- **Minimización** (o "minificación"): eliminar todo lo superfluo del código con el fin de hacerlo lo más compacto posible y que pese muy poco. Quita todos los espacios innecesarios, saltos de línea, tabuladores, comentarios, etc... y queda un código bastante difícil de leer, pero que ocupa mucho menos y se descarga más rápido desde el servidor.
 - Para archivos js, css y html.
 - Podemos hablar de una reducción en torno al 45-50%.
 - Generalmente se utiliza el término ".min" antes de la extensión del archivo.
 - Extensión para *Visual Studio Code*: **Minify**
 - Con las herramientas de desarrollador de los navegadores podemos volver a ver el código sin minimizar, eso si se pierden los comentarios.



5. Integración del código con las etiquetas HTML



PROTECCIÓN DEL CODIGO

➤ Minificado y ofuscación del código:

- **Ofuscación:** además de minificación, hace cambios en el código mediante procesos de parseo y sustitución, para hacerlo realmente difícil depurarlo y reutilizarlo. Para ello se llevan a cabo múltiples operaciones, como cambio de nombre a variables y funciones, recodificación de cadenas, sustitución de parte del código por otro equivalente pero más complejo...
 - Solo hay que hacerlo con los archivos que vayan a ser publicados en el entorno de producción y no en etapas de desarrollo.
 - Hay software para poder desofuscar, por lo que no sirve para ocultar información confidencial dentro del código fuente.
- Ventajas:
 - Se reduce el tiempo de carga de la página.
 - Dificulta que un atacante pueda visualizar el código de la aplicación para identificar vulnerabilidades en el mismo.
 - Protección de la propiedad intelectual.
- Software:
 - Online: <https://javascriptobfuscator.com/Javascript-Obfuscator.aspx>