

DESARROLLO WEB EN ENTORNO CLIENTE

U.D. 1:

**Selección de arquitecturas y herramientas de
programación**

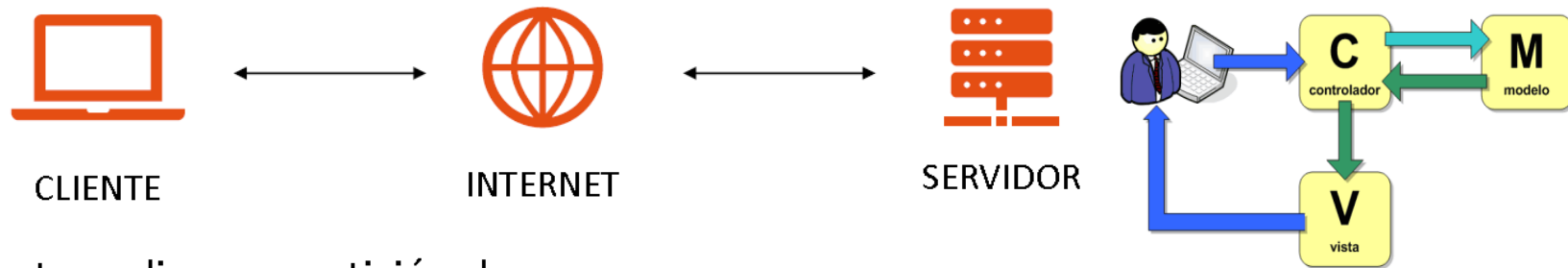


¿Qué es una aplicación Web?

Las aplicaciones Web son aquellas herramientas donde los usuarios pueden acceder a un servidor Web a través de la red mediante un navegador determinado. Por lo tanto, se define como una aplicación que se accede mediante la Web por una red ya sea intranet o Internet. Por lo general se menciona aplicación Web a aquellos programas informáticos que son ejecutados a través del navegador.

Luján Mora, 2002

Arquitectura de una aplicación Web



1. El cliente realiza una petición de un recurso
 - Introduce una dirección en un navegador web (e.g. www.google.com)
 - A través de un **servicio DNS** (*Domain Name Server*) traduce los nombres de dominio a **direcciones IP** (e.g. 142.250.176.196)
 - La dirección **IP** le permite contactar con el servidor y enviarle la **petición HTTP** (*HyperText Transfer Protocol*) tipo GET
2. Se establece una **conexión TCP** (*Transmission Control Protocol*)
3. El servidor proporciona el recurso solicitado:
 - Envía por HTTP los ficheros asociados (.HTML, .CSS, .JS, contenido multimedia, otros...)
4. Se cierra la conexión

Arquitectura de una aplicación Web

- Funcionamiento del protocolo HTTP:
 - Usuario accede a la URL mediante enlace de un documento HTML o introduciéndola directamente en el campo *Location* del cliente Web.
 - El cliente Web decodifica la URL separando partes: protocolo acceso, DNS o IP servidor, puerto y objeto.
 - Se abre una conexión TCP/IP con el servidor llamando al puerto TCP correspondiente. Se realiza la petición, enviando:
 - Comando (GET, POST, HEAD,...).
 - Contenido URL que sigue a la dirección del servidor.
 - Versión protocolo HTTP.
 - Información: capacidades navegador, datos opcionales,...
 - Servidor devuelve respuesta al cliente: Código estado y tipo de dato MIME más la información.
 - Se cierra la conexión TCP.

Arquitectura de una aplicación Web

- Funcionamiento del protocolo HTTP:
 - Proceso se repite en cada acceso al servidor HTTP.
 - Ej. si se recoge un documento HTML donde están insertadas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML y cuatro para las imágenes.
 - Tipos de mensajes que utiliza HTTP:
 - GET: para recoger cualquier información del servidor. Se usa al pulsar sobre un enlace o al teclear directamente una URL.
 - POST: para enviar información al servidor. Ej. datos de un formulario.
 - HEAD: solicita información sobre un objeto (fichero) como tipo, tamaño, fecha modificación. Usado por los gestores de cachés o servidores proxy para saber cuando actualizar copia de un fichero.

Front-end vs. Back-end



Front-end

- Parte visual de una página web
- Muestra el diseño, los contenidos y permite a los visitantes navegar por la página

Back-end

- Gestión de datos de un servicio
- Incluye la conexión con las bases de datos, la gestión de usuarios, la distribución de la información dentro de la nube, la gestión de permisos de usuarios...

Front-end vs. Back-end

me working
in backend



**I completed the
entire backend
in just one day**

me working
in frontend



**how the f*ck
I can align
this button
at center?**



developers_team

Cliente Web



¿Qué es un cliente?

Un navegador web con el que interactúa el usuario



Chrome

(64,67 %)



Safari

(19,06 %)



Edge

(3,99 %)



Firefox

(3,66 %)



Opera

(2,36 %)

Cuota de mercado octubre 2021
(fuente: [gstatcounter.com](https://www.gstatcounter.com))

Evolución y características de los navegadores Web

- World Wide Web (WWW).
 - Conjunto de recursos interconectados que conforman el conocimiento humano actual.
 - *Hubs*, repetidores, puentes, pasarelas, encaminadores.
 - Protocolos de comunicaciones: TCP, IP, HTTP, FTP, SMTP.
 - Sistema de nombres de dominio (DNS).
- Configuración arquitectónica más habitual: *Cliente/Servidor*.
 - *Cliente* es un componente consumidor de servicios.
 - *Servidor* es un proceso proveedor de servicios.

Evolución y características de los navegadores Web

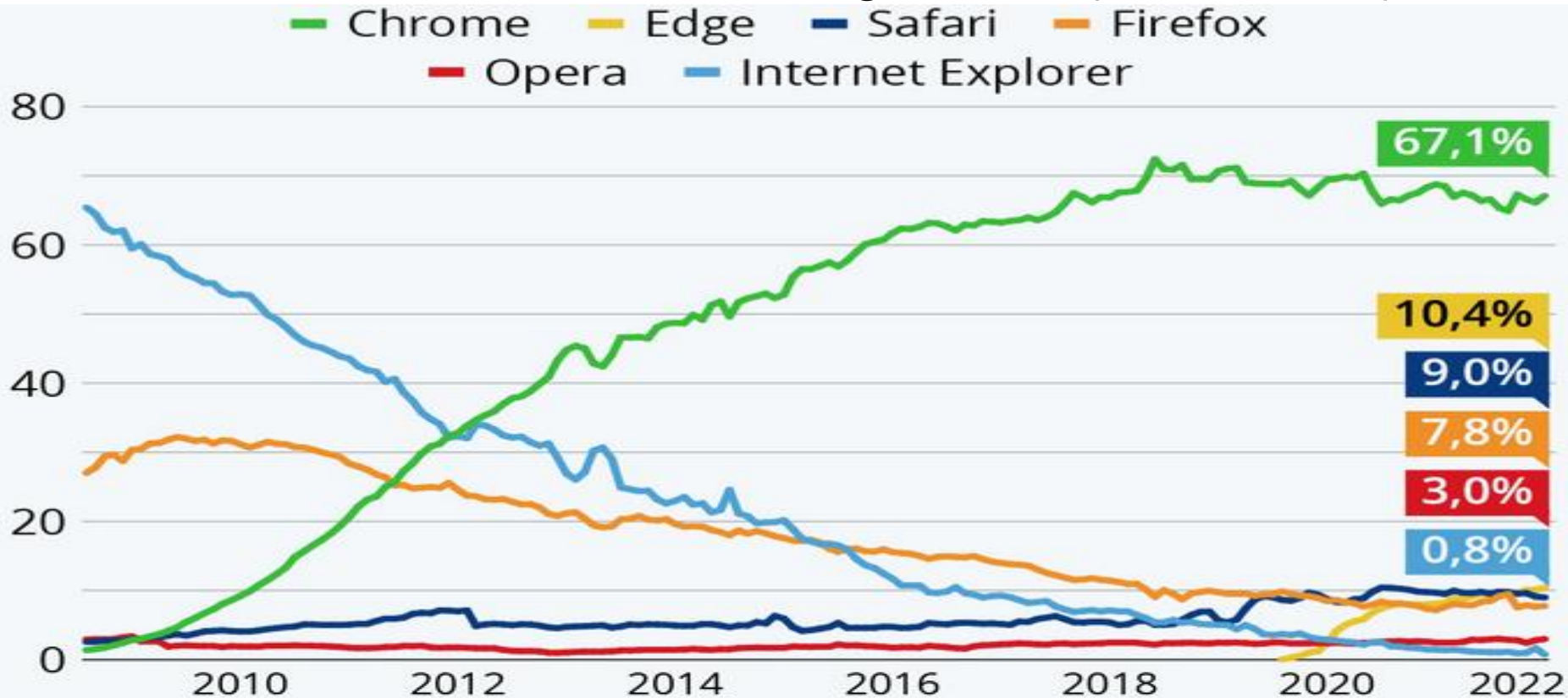
- Navegador Web:
 - Componente software que se utiliza en el cliente y que permite acceder al contenido ofrecido por los servidores de Internet sin la necesidad de que el usuario instale un nuevo programa.
 - Aplicación, distribuida habitualmente como software libre, que permite a un usuario acceder (y normalmente visualizar) a un recurso publicado por un servidor Web a través de Internet y descrito mediante una dirección URL (*Universal Resource Locator*).

Evolución y características de los navegadores Web

- Navegador Web. Ejemplos:
 - Mosaic. Uno de los primeros navegadores Web y el primero con capacidades gráficas.
 - Netscape Navigator (después Communicator). Fue el primer navegador en incluir un módulo para la ejecución de código *script* (JavaScript).
 - Edge. Es el navegador de Microsoft.
 - **Mozilla Firefox**. Se trata de un navegador de código abierto multiplataforma de gran aceptación.
 - **Google Chrome**. Es el navegador de Google compilado a partir de componentes de código abierto.
 - Safari. Es el navegador por defecto de los sistemas de Apple.
 - Dolphin Browser. Específico para el sistema operativo Android, fue uno de los primeros en incluir soporte para navegación multitáctil.

Evolución y características de los navegadores Web

- Estadísticas de uso de navegadores (2009-2022):



Datos mensuales de enero de 2009 a junio de 2022 extraídos el 13 de junio de 2022.
Fuente: StatCounter

Evolución y características de los navegadores Web

- Navegador Web. Criterios de clasificación:
 - **Plataforma de ejecución.** Sistema operativo.
 - **Características del navegador.** Funcionalidades adicionales.
 - **Personalización de la interfaz.** Funciones de accesibilidad.
 - **Soporte de tecnologías Web.** Grado de soporte de los estándares de la Web.
 - **Licencia de software.** Código libre y navegadores propietarios.

Arquitectura de ejecución

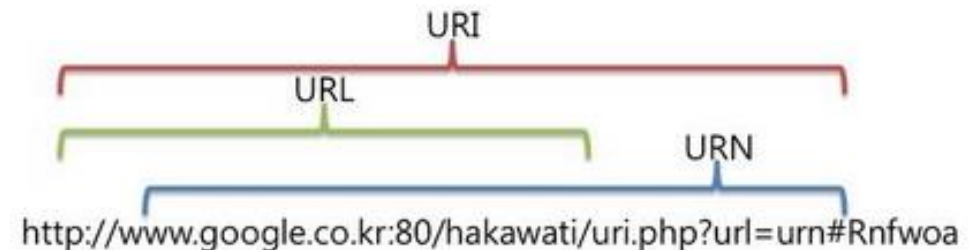
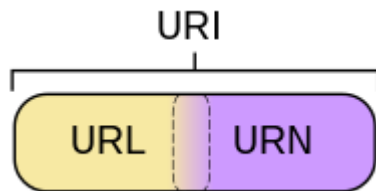
- Funcionamiento del navegador Web:
 - Solicitar al servidor los recursos Web que elija el usuario y mostrarlos en una ventana.
 - El recurso suele ser un documento codificado en HTML aunque también pueden ser archivos (pdf, Word, audio, imagen,...).
 - El usuario especifica la ubicación del recurso mediante el uso de una dirección URI (Uniform Resource Identifier) o Identificador.

Arquitectura de ejecución

■ Estructura de una URI:

- Sistema global que condensa la dirección (URL) y el nombre (URN) del recurso para identificarlo dentro de la red.
- Cadena corta de caracteres que identifica de manera única un recurso.
- Esquema : Parte jerárquica ? Solicitud # Fragmento

esquema://máquina/directorio/archivo?solicitud#fragmento



Arquitectura de ejecución

- URI -> Esquema:
 - Identifica el protocolo a utilizar a la hora de solicitar el recurso:
 - http: es el más habitual.
 - ftp: transferencia de archivos.
 - Otros: https, file, telnet, gopher, ldap, mailto, etc.

Arquitectura de ejecución

- URI -> Parte jerárquica:
 - Información del dominio o dirección IP para acceder al servidor y la ruta en el servidor para acceder al recurso.
 - Ej. `//www.servidor.com/ruta/recurso.html`
 - Las 2 barras inclinadas al principio `//` indican que la dirección debe ser pasada al recurso para que éste la interprete.
 - El servidor puede ser una dirección IP o un nombre de dominio y puede llevar parámetros como el puerto e información de control de acceso.
 - Ej. `http://usuario:clave@miembros.sitio.com:80/`

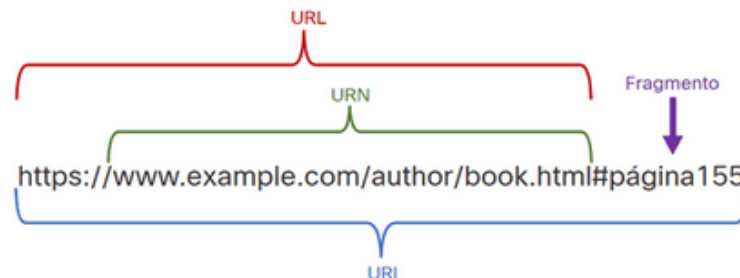
Arquitectura de ejecución

- URI -> Solicitud:
 - Variables que se pasan al recurso (ej. página Web).
 - Está separada de la ruta mediante el signo ? y termina donde empieza el fragmento delimitado por # si lo hubiere.
 - Ej. /miruta.html?variable=valor&variable2=valor2

Arquitectura de ejecución

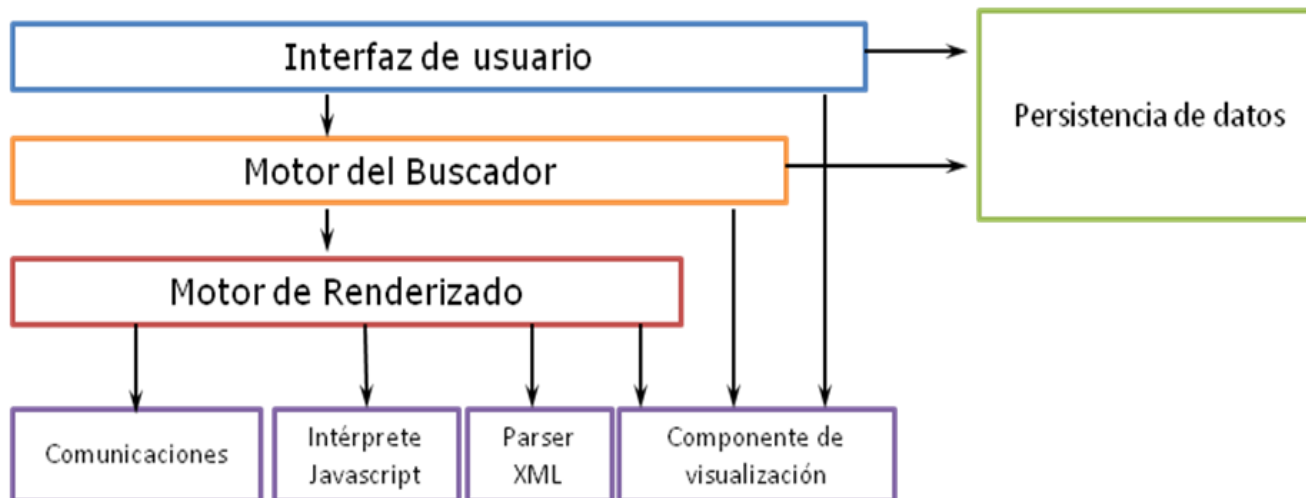
■ URI -> Fragmento:

- Permite indicar una subdirección dentro del recurso al que apunta la dirección.
- Está delimitado por el símbolo # y se extiende hasta donde se termina la URI.
 - Ej. /miruta.html#subdireccion
- Este fragmento es la diferencia entre URL y URI:
 - Las URL no identifican fragmentos, son un subconjunto de URI.
 - Se utiliza URI cuando se habla de direcciones completas.
 - Las URL son identificadores que permiten acceder a recursos Web, normalmente páginas, localizan, mientras que URI identifican.



Arquitectura de ejecución

- Proceso de ejecución:
 - Se inicia con el usuario indicando la dirección del recurso al que quiere acceder y termina con la visualización del recurso por parte del navegador en la pantalla del usuario.
- Arquitectura de referencia de un navegador Web:



Arquitectura de ejecución

- Arquitectura de referencia de un navegador Web (I):
 - **Subsistema de interfaz de usuario.** Es la capa que actúa de interfaz entre el usuario y el motor del buscador (o de navegación).
 - **Subsistema del motor del buscador o motor de navegación.** Este subsistema es un componente que ofrece una interfaz de alto nivel para el motor de renderizado.
 - **Subsistema de renderizado.** Este componente es el encargado de producir una representación visual del recurso obtenido a partir del acceso a una dirección Web.
 - **Subsistema de comunicaciones.** Es el subsistema encargado de implementar los protocolos de transferencia de ficheros y documentos utilizados en Internet (HTTP, FTP, etc.).

Arquitectura de ejecución

- Arquitectura de referencia de un navegador Web (II):
 - **Intérprete de JavaScript.** Será el encargado de analizar y ejecutar código JavaScript.
 - **Parser XML.** Módulo que permite cargar en memoria una representación en árbol de la página Web.
 - **Componente de visualización.** Este subsistema ofrece funcionalidades relacionadas con la visualización de los contenidos de un documento HTML en una página Web.
 - **Subsistema de persistencia de datos.** Funciona como almacén de diferentes tipos de datos para los principales subsistemas del navegador.

Arquitectura de ejecución

- Proceso de carga en un navegador Web:
 - Conforme el servidor recibe código, se muestra en el navegador en el área destinada a ello.
 - Se comienza a interpretar la estructura del documento y la búsqueda de recursos externos, scripts para su descarga.
 - Las imágenes, scripts y demás archivos de una página se guardan en una carpeta temporal.
 - Si se dispone de antivirus analiza estos archivos.
 - La velocidad de carga es mayor si se repiten las peticiones.

Arquitectura de ejecución

- ¿Hay diferencias entre navegadores para el desarrollo front-end?
 - Aunque en general el comportamiento es parecido, puede haber diferencias puntuales. Ej. soporte de vídeo HTML5 por códec:

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari
H.264	 4	 12	 35	 9	 25	 3.2
HEVC (H.265)	 No	 18	 No	 11	 No	 11
AV1	 70	 75	 67	 No	 57	 No
VP8 (WebM)	 25	 14	 4	 9	 16	 12.1
VP9 (WebM)	 29	 14	 28	 No	 10.6	 No

Tipos de aplicaciones Web

- Existen diferentes clasificaciones de aplicaciones web, por lo que no hay un número determinado de tipos de aplicaciones web
- Una primera clasificación básica es:

Páginas web estáticas

- Muestran contenido fijo
- Ofrecen poca o nula interactividad
- Son simples y se cargan rápido
- La actualización es más compleja
- Ejemplo: páginas con contenido que no suele variar (portfolios, páginas de presentación de empresas, currículums digitales...)

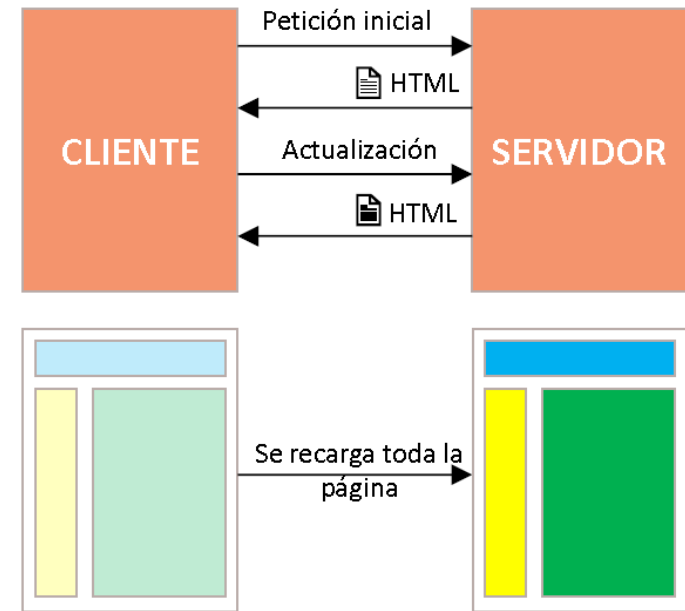
Páginas web dinámicas

- Generan datos en tiempo real en función de las peticiones del cliente (interactividad)
- Utilización de bases de datos
- Más complejas y el tiempo de carga es mayor
- Su actualización es más sencilla
- Ejemplo: la mayoría de webs comerciales

Tipos de aplicaciones Web

Multiple Page Applications (MPA)

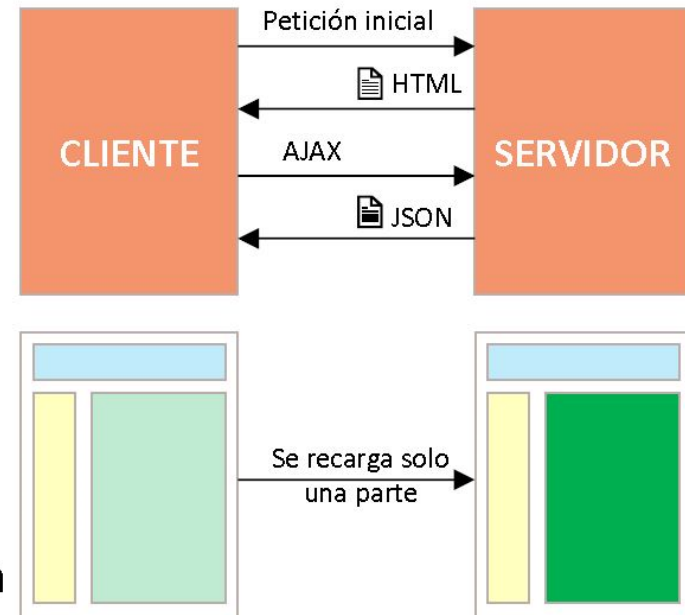
- Enfoque clásico de la programación de páginas web: una web está compuesta por varias páginas, que se cargan según el usuario va navegando
- Tienen una carga inicial más rápida
- La experiencia de navegación para el usuario puede ser negativa por el tiempo de navegación entre páginas



Tipos de aplicaciones Web

Single Page Applications (SPA)

- Enfoque más actual de programación
- Tienen una única página web
- El navegador solo recarga ciertas secciones de la página en función de las peticiones realizadas por el cliente
- Carga inicial más lenta comparada con MPA
- La página se actualiza localmente en el cliente, no en el servidor
- Mejora la experiencia del usuario, ya que la navegación es más fluida



NETFLIX

Tipos de aplicaciones Web

SPA vs MPA

	Single Page Application	Multi Page Application
Velocidad	<ul style="list-style-type: none">▪ La carga inicial puede ser alta▪ Navegación fluida y velocidades rápidas después de la carga inicial	<ul style="list-style-type: none">▪ La carga inicial es menor▪ Requiere de buena conexión a Internet, sobre todo si las páginas web contienen muchos elementos gráficos
Offline	<ul style="list-style-type: none">▪ Trabaja de forma offline una vez se ha cargado	<ul style="list-style-type: none">▪ Requiere conectividad a Internet
Problemas de memoria	<ul style="list-style-type: none">▪ Puede tener problemas de memoria en el navegador	<ul style="list-style-type: none">▪ Menos probabilidad de tener problemas de memoria
Seguridad	<ul style="list-style-type: none">▪ Más fácil de sufrir ciber ataques	<ul style="list-style-type: none">▪ Puede ser protegida frente a vulnerabilidades
Posicionamiento	<ul style="list-style-type: none">▪ Más difícil posicionamiento SEO	<ul style="list-style-type: none">▪ El posicionamiento en buscadores es más fácil
Usos	<ul style="list-style-type: none">▪ Webs planificadas para apps▪ No requiere SEO▪ Soluciones SaaS▪ Redes sociales	<ul style="list-style-type: none">▪ Empresas que ofrecen un amplio catálogo de servicios y/o productos▪ Requieren SEO▪ Tiendas eCommerce▪ Blogs

Capas de una aplicación Web

- **Capa de estructura de la web.** Indica qué elementos tiene una web y cómo se relacionan semánticamente entre ellos: secciones, cabeceras, pie de página, menú de navegación, listas, tablas... Se escribe usualmente en lenguaje HTML o alguno de sus derivados
- **Capa de presentación de los elementos de la web.** Indica cómo se tienen que mostrar los elementos descritos en la estructura de la web: colores de los textos, fondo de las secciones, fuentes tipográficas, tamaño de los iconos... Se describe en CSS
- **Capa de comportamiento de la web.** Gestiona los cambios de una web producidos por interacciones del usuario o porque llegan nuevas secciones desde el servidor web: qué sucede cuando el usuario hace clic sobre un botón, dónde colocar las noticias que llegan del servidor o cómo enviar un texto que ha introducido el usuario a un servidor de correo electrónico. Se programa en JavaScript y utiliza lenguajes como XML y JSON para comunicarse con el servidor

Lenguajes y tecnologías de programación en entorno cliente

- Los lenguajes de programación del entorno de cliente son aquellos que se ejecutan en el navegador Web.
 - Lenguajes principales:
 - HTML.
 - DHTML.
 - XML.
 - XHTML.
 - Lenguajes de scripting:
 - JavaScript.
 - VBScript.
 - Otros lenguajes:
 - ActionScript.
 - AJAX.

Lenguajes y tecnologías de programación en entorno cliente

- Lenguajes básicos del desarrollo front-end:

HTML



Estructura

CSS



Presentación

JavaScript



Comportamiento

- El consorcio W3C (World Wide Web Consortium) determina las especificaciones y estándares CSS y HTML.

Lenguajes y tecnologías de programación en entorno cliente

- HTML y derivados (I):
 - *HTML: Hyper Text Markup Language* (lenguaje de marcado de hipertexto) es el lenguaje de marcas de texto más utilizado en la *World Wide Web*.
 - Se basa en la utilización de un sistema de etiquetas cerrado aplicado a un documento de texto.
 - No necesita ser compilado, sino que es interpretado (ejecutado a medida que se avanza por el documento HTML).
 - Hipervínculo: enlace de una página Web o un archivo a otra página Web u otro archivo.

Lenguajes y tecnologías de programación en entorno cliente

- HTML y derivados (II):
 - **XML**: lenguaje de etiquetado extensible cuyo objetivo principal es describir datos para su transferencia eficiente y no mostrarlos, como es el caso de HTML.
 - **XHTML**: adaptación de HTML al lenguaje XML.
 - HTML Dinámico (**DHTML**): integración de HTML con lenguajes de *scripting* (JavaScript), hojas de estilo personalizadas (CSS) y la identificación de los contenidos de una página Web en formato de árbol (DOM).

Lenguajes y tecnologías de programación en entorno cliente

- **CSS** (*Cascade Style Sheets*): sirve para separar el formato que se quiere dar a la página Web de la estructura de la página Web y las demás instrucciones.
- **JavaScript**: lenguaje de programación de *scripting* (interpretado) y, normalmente, embebido en un documento HTML.
- **AJAX** (*Asynchronous JavaScript And XML*): conjunto de técnicas y métodos de desarrollo Web para la creación aplicaciones Web interactivas y asíncronas.

Lenguajes y tecnologías de programación en entorno cliente

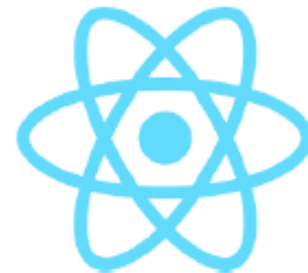
- **Frameworks:** permiten la programación front-end de una forma más sencilla y estructurada



Angular



Vue.js



React

Ejercicios

- Realiza todos los ejercicios del bloque 'Ejercicios 1.1'.

Herramientas de edición

- Resulta esencial utilizar un **editor de texto plano** (aquellos que no tienen formato) para la programación web
- Cualquier herramienta básica (como el “Bloc de notas”) sirve para programar en HTML, CSS y JavaScript
- Sin embargo, existen herramientas que facilitan la programación web: “formateando el código”, autocompletando texto, detectando errores...

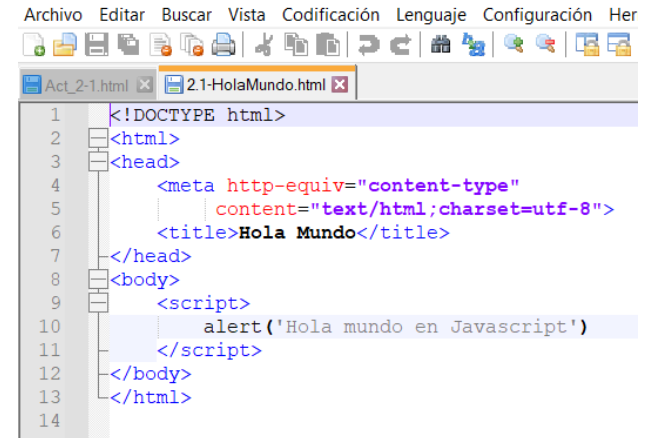
```
<!DOCTYPE html>
<html lang="es-ES">
<head>
<meta charset="UTF-8">
<title> Test </title>
<link rel="icon" href="https://www.universidadvlu.com/media/layout/favicon.PNG">
</head>
<body>
<header>
<h1> Integración de servicios telemáticos </h1>
<h2> Práctica número 1 </h2>
</header>
<nav class="principal">
<ul>
<li><a href="seccion1.html" target="_self">Sección 1</a></li>
<li><a href="seccion2.html" target="_self">Sección 2</a></li>
</ul>
</nav>
<hr>
</body>
</html>
```

```
1 <!DOCTYPE html>
2 <html lang="es-ES">
3 <head>
4   <meta charset="UTF-8">
5   <title> Test </title>
6   <link rel="icon" href="https://www.universidadvlu.com/media/layout/favicon.PNG">
7 </head>
8 <body>
9   <header>
10    <h1> Máster Universitario en Desarrollo de Aplicaciones y Servicios Web </h1>
11    <h2> Herramientas para el desarrollo web </h2>
12  </header>
13  <nav class="principal">
14    <ul>
15      <li><a href="seccion1.html" target="_self">Sección 1</a></li>
16      <li><a href="seccion2.html" target="_self">Sección 2</a></li>
17    </ul>
18  </nav>
19  <hr>
20 </body>
21 </html>
```

Herramientas de edición

- **Notepad++:** Editor de texto plano y de código fuente libre. Coloreado y envoltura de sintaxis: si se escribe en un lenguaje de programación o marcado, es capaz de resaltar las expresiones propias de la sintaxis de ese lenguaje para facilitar su lectura.

- <https://notepad-plus-plus.org/downloads/>



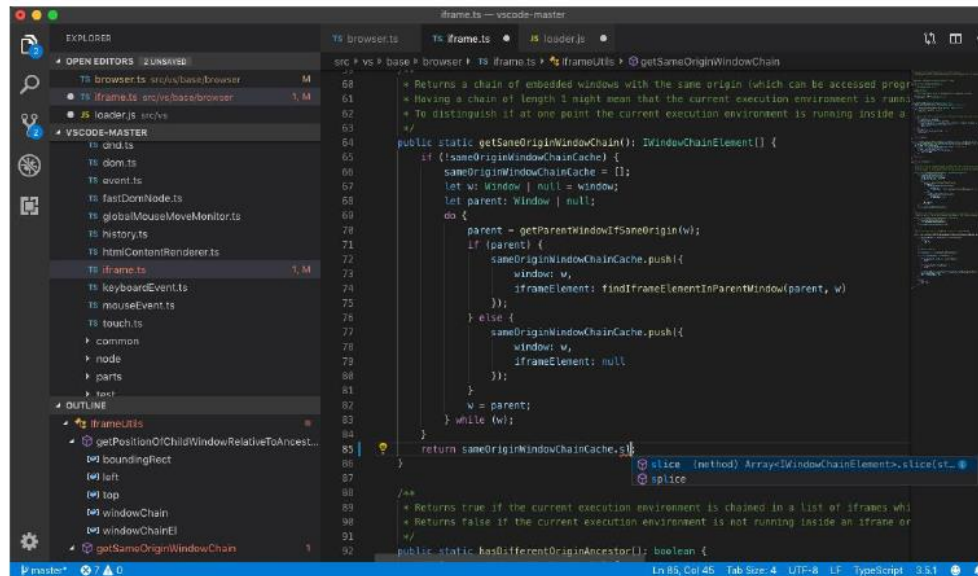
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta http-equiv="content-type"
5           content="text/html; charset=utf-8">
6     <title>Hola Mundo</title>
7 </head>
8 <body>
9
10    <script>
11        alert('Hola mundo en Javascript')
12    </script>
13 </body>
14 </html>
```

- **UltraEdit:** Requiere licencia.
- <https://www.ultraedit.com/downloads/>

Herramientas de edición

Visual Studio Code

- Seguramente, el editor de texto más utilizado para la programación web
- Disponible en: <https://code.visualstudio.com/>



Herramientas de edición

Sublime Text

- <https://www.sublimetext.com/>

```
1 <!doctype html>
2 <html>
3 <head>
4   <!--Title-->
5   <title></title>
6
7   <!--Meta-->
8   <meta http-equiv="X-UA-Compatible" content="IE=edge">
9   <meta name="revisit-after" content="30 days">
10  <meta name="distribution" content="web">
11  <meta charset="UTF-8">
12
13  <meta name="description" content="">
14  <meta name="keywords" content="">
15  <meta name="author" content="Twiply">
16
17  <!--Stylesheets-->
18  <link href="/css/styleset.css" type="text/css" rel="stylesheet">
19  <link href="https://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css" rel="stylesheet">
20  <link href="https://netdna.bootstrapcdn.com/bootstrap/2.3.1/css/bootstrap.min.css" rel="stylesheet">
21
22  <!--Icons-->
23  <link rel="shortcut icon" href="">
24
25  <!--JavaScript-->
26  <script src="https://code.jquery.com/jquery-1.10.2.min.js"></script>
27  <script src="https://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
28  <script src="https://netdna.bootstrapcdn.com/bootstrap/3.1.0/js/bootstrap.min.js"></script>
29
30 </head>
31 <body>
32   <div class="container">
33
34 </div>
35 </body>
36 </html>
```



Atom

- <https://atom.io/>

```
Project
├── real-time
│   ├── .git
│   └── lib
│       ├── buffer-binding.js
│       ├── editor-binding.js
│       ├── ghost-portal-binding.js
│       ├── join-portal-dialog.js
│       ├── normalize-url.js
│       └── real-time-package.js
├── node_modules
├── script
├── styles
├── test
├── .gitignore
├── .travis.yml
├── index.js
├── package-lock.json
├── package.json
└── README.md

real-time-package.js
1  const {CompositeDisposable} = require('atom')
2  const {allowUnsafeNewFunction} = require('loophole')
3
4  let Client
5  allowUnsafeNewFunction(() => { Client =
6
7  const BufferBinding = require('./buffer-binding')
8  const EditorBinding = require('./editor-binding')
9
10 module.exports =
11 class RealTimePackage {
12   constructor (options) {
13     cons
14
```

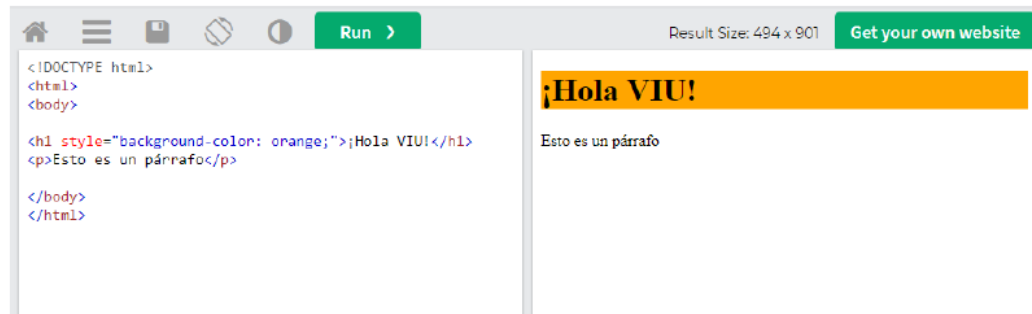


Herramientas online de edición

- Permite editar código HTML, CSS y JavaScript (entre otros) desde un navegador

W3Schools

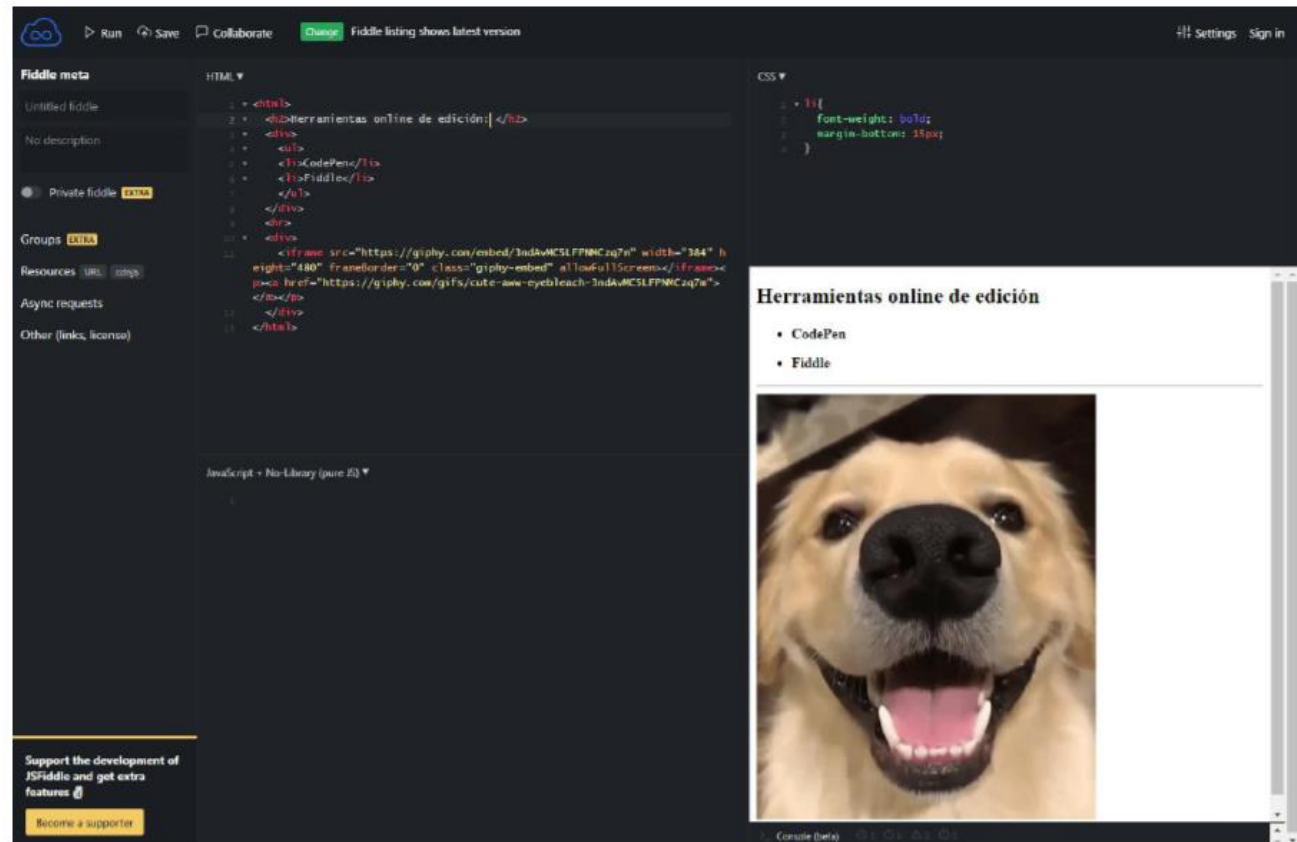
- <https://www.w3schools.com>
- Página de referencia para la programación web
- Ofrece múltiples ejemplos y una aplicación para probar código



Herramientas online de edición

Fiddle

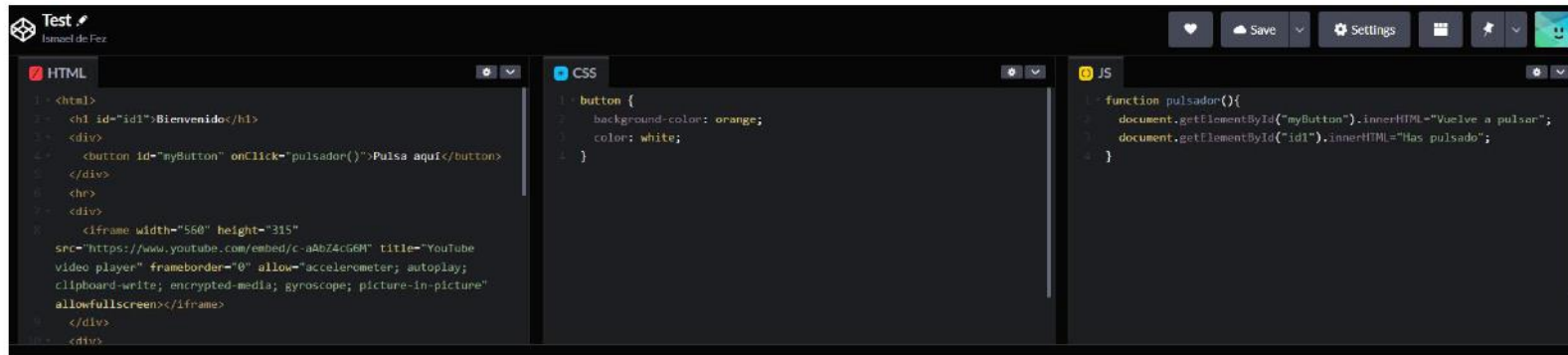
- <https://jsfiddle.net/>



Herramientas online de edición

CodePen

- <https://codepen.io/>



Herramientas de depuración en navegadores

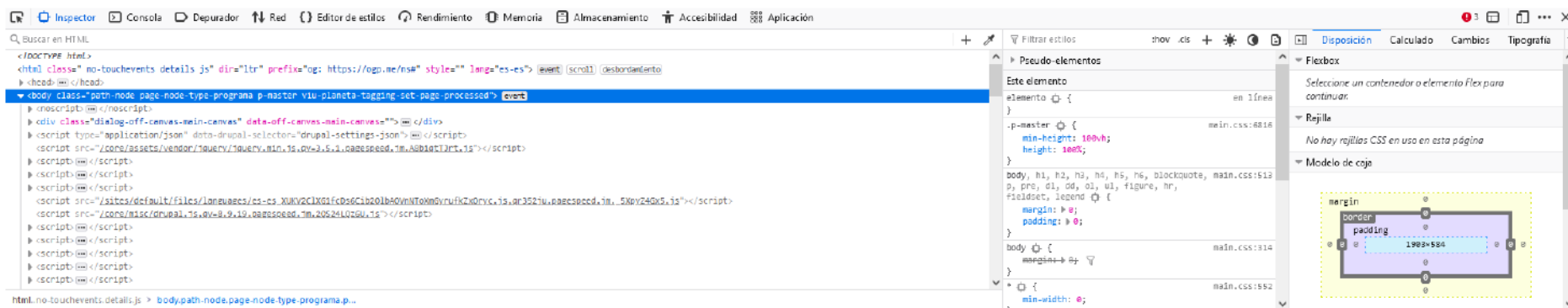
- Es posible ver el código de una página web a través de los diferentes navegadores (e.g. *view-source:https://www.amazon.es/*)
 - Sin embargo, gran parte del código está “enmascarado” por la utilización de librerías y de elementos minimizados y ofuscados

```
(function(d,h,N){function H(a){return a&&a.replace&&a.replace(/\$+|\$+$/,"")}function u(a){return"undefined"===typeof a}function c){return c+b}}),d.ue_sid=b,&&a.tag("page-source":"+c),d.ue_fpf=}function P(){var a={};return function(b){b&&a[a[b]]=b[1];for(var function(a){??(e(a),k(a)):(k(a),e(a))}[b[c]&&b[c][u]eH=1]}function S(m,b,c,a){function p(b,c){var d=[b],g=0,f={},h,k;c[d.push("m a.iel.push(e),e.src=b,a.count&&a.count("postbackImageSize",b.length));if(w){var m=h.encodeURIComponent;m&&b&&(e=new Image,b="+d.ue (d=a.ssw(a oid),d,e[|u.val])|{a.isNRBF=1&d.val?0:1)},[a.isNRBF]||[b+&+&nrnf="+a.isNRBF]),a.isSBFT&&a.isNRBF&&(b+&+&bf+a.isSBFT if(!("ld"=m&&"ul"=m[|&&b[1]=s)}(if(!("ld"=m&&"val"?[h[K]&&h[K].isUeH&&h[K]=null)]catch(i){if(h.chrome)for(n=0;t<L.length;t++){G(L,t co_.f.connectStart,_co:f.connectEnd,sc_.f.secureConnectionStart,rq_.f.requestStart,rs_.f.responseStart,_rs:f.responseEnd,dl_.f.domL b==s[|ca(b){c=d.ue_mbl}&c.cnt&1&&(g+c.cnt())};?e="wb",b,2):"ld"=m&&a. lid=H(s)};for(r in a.sci){if(1==e("wb",r))break;if(1){if "ld"!=m[|b|a.markers[|](a.markers=|),c(a.markers,e("t",b)),e("t",b,{));a.tag&&a.tag(|.length&(g+=&cmstags="+a.tag(|.join(")", else if(c[W])c[W]("on"+a,b))}function T(a,b,c){c=c[|h;if(c[X])c[X](a,b,1);else if(c[Y])c[Y]("on"+a,b)}function Z(){function a(){d.o {requestId:e,transitionType:"soft"},p("mark","transitionStart",b));a.tag("ajax-transition")}d.ueinit=d.ueinit[|0]+1;var a=d.ue.d fcp:"firstContentfulPaint",bb:"bodyBegin",be:"bodyEnd",ld:"loaded"},E=h.Date,B=h.performance||h.webkitPerformance,f(B[|]);.timing
```

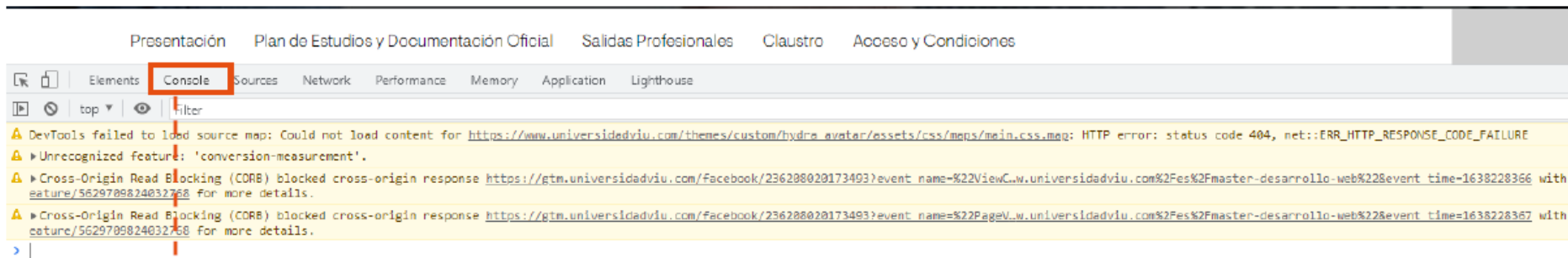
- Los navegadores disponen de “Herramientas para desarrolladores” que ayudan a realizar el desarrollo front-end

Herramientas de depuración en navegadores

- Las funcionalidades básicas que ofrecen los distintos navegadores son prácticamente las mismas, y difieren en alguna funcionalidad adicional



Herramientas de depuración en navegadores



- ▶ La opción de consola es muy importante cuando desarrollamos porque podemos ver errores que surgen y los mensajes que mostramos de prueba

Integración del código con las etiquetas HTML

- JavaScript en el mismo documento HTML (estilo interno o embebido):
 - Uso de unas etiquetas predefinidas para marcar el texto (`<script>` y `</script>`).
 - Puede incluirse en cualquier parte del documento, aunque se recomienda que se defina dentro de la cabecera del documento HTML.
 - 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.

Integración del código con las etiquetas HTML

- JavaScript en el mismo documento HTML - Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8">
    <title>Hola Mundo</title>
    <script>
      alert("Prueba de JavaScript");
    </script>
  </head>
  <body>
    <h1>Ejemplo 1: código embebido</h1>
  </body>
</html>
```


Integración del código con las etiquetas HTML

- JavaScript en un archivo externo (estilo externo):
 - Las mismas instrucciones de JavaScript que se incluyen entre un bloque `<script></script>` pueden almacenarse en un fichero externo con extensión `.js`.
 - La forma de acceder y enlazar esos ficheros `.js` con el documento HTML/XHTML es a través de la propia etiqueta `<script>`.
 - No existe un límite en el número de ficheros `.js` que pueden enlazarse en un mismo documento HTML/XHTML.

Integración del código con las etiquetas HTML

- JavaScript en un archivo externo - Ejemplo:

Archivo mensaje.js:

```
alert("Prueba de JavaScript");
```

Archivo ejemplo2.html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8">
    <title>Hola Mundo</title>
    <script type="text/javascript"
      src="mensaje.js"></script>
  </head>
  <body>
    <h1>Ejemplo 2: fichero externo</h1>
  </body>
</html>
```

Integración del código con las etiquetas HTML

- JavaScript en atributos de elementos HTML (inline):
 - Consiste en insertar fragmentos de JavaScript dentro de atributos de etiquetas HTML de la página.
 - 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.

Integración del código con las etiquetas HTML

- JavaScript en atributos de elementos HTML - Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8">
    <title>Hola Mundo</title>
  </head>
  <body>
    <p onclick="alert('Prueba de JavaScript');">
      Ejemplo 3: código en atributos
    </p>
  </body>
</html>
```