Desarrollo web en entorno cliente

Unidad 1: Selección de arquitecturas y herramientas de programación

ÍNDICE

INTRODUCCIÓN

Lo primero que tenemos que tener claro al empezar este módulo es **cómo funcionan las distintas arquitecturas** que vamos a utilizar. Para ello es muy importante ver las diferentes **alternativas** que existen y cuáles son las más utilizadas.

Haremos especial hincapié en el **modelo cliente/servidor**, el más utilizado cuando hablamos de **programación web**, y, en el caso de este módulo, centrándonos en el apartado del cliente.

Pese a ello no podemos olvidarnos de la parte del **servidor**, al menos para entender cómo funciona el proceso y poder adaptar así la programación del cliente.

Además, durante la unidad veremos los diferentes **lenguajes** que podemos utilizar en el entorno cliente, de entre los cuales nos quedaremos sobre todo con JavaScript para el desarrollo del módulo.

OBJETIVOS / CAPACIDADES

En esta unidad de aprendizaje, las capacidades que más se van a trabajar son:

- ✔ **Seleccionar lenguajes, objetos y herramientas, interpretando las especificaciones para desarrollar aplicaciones Web con acceso a bases de datos.**

PROYECTO DE LA UNIDAD

Antes de empezar a trabajar el contenido, te presentamos la **actividad** que está relacionada con esta unidad de aprendizaje. Se trata de un **caso práctico** basado en una **situación real** con la que te puedes encontrar en tu puesto de trabajo. Con esta actividad se evaluará la puesta en práctica de los **criterios de evaluación** vinculados al resultado de aprendizaje que se trabaja en esta unidad. Para realizarla deberás hacer lo siguiente: lee el enunciado que te presentamos a continuación, dirígete al área general del módulo profesional, concretamente a la actividad de evaluación que se encuentra dentro de esta unidad, allí encontrarás todos los detalles sobre fecha y forma de entrega, objetivos... A lo largo de la unidad irás adquiriendo los conocimientos necesarios para ir elaborando este proyecto.

Enunciado:

Elección de tecnologías y primeras pruebas

Una vez superada la primera unidad nos vemos con las suficientes fuerzas para empezar a crear nuestros propios proyectos de programación web en entorno cliente. Charlando con unos compañeros del módulo nos han surgido varias dudas sobre las tecnologías a utilizar. En concreto sobre:

- El modelo de ejecución.
- El entorno de desarrollo.
- El lenguaje de programación.

Es por ello que hemos decidido crear un documento con los distintos puntos a discutir, para tenerlo todo mucho más claro y organizado.

Durante la unidad hemos visto diferentes modelos de ejecución de código vía web, siendo los más comunes, el modelo cliente/servidor y el modelo P2P.

En el documento añadiremos un apartado comparando estos dos modelos, con ventajas e inconvenientes y cuál creemos que vamos a usar más.

En la charla han aparecido tres entornos de desarrollo diferentes que podemos utilizar: Sublime, VSCode y Notepad++.

Otro apartado del documento tendrá que ver con estos IDEs, diferenciando sus características, para finalmente realizar la instalación del escogido.

Una vez escogido e instalado el IDE correspondiente nos vamos a centrar en la programación de scripts. De entre todos los lenguajes nos han recomendado JavaScript, pero para cerciorarnos de que la decisión es la correcta, vamos a crear otro apartado con sus ventajas y desventajas frente a otros lenguajes de script.

Además, es importante también destacar las diferencias con los lenguajes tradicionales.

Una vez lo tenemos claro, haremos nuestra primera web con JavaScript, pero como aún no hemos empezado a ver la sintaxis, lo primero que se nos ocurre es buscar el código del famoso "Hello World" por internet e intentar visualizarlo en una página web.

Se pide insertar el código JavaScript en un fichero HTML sencillo y ejecutarlo (en al menos 3 navegadores), para mostrar el resultado. La página debe mostrar "Hello World" en el navegador.

Por último, abriremos la consola de nuestro navegador, para poder ver nuestro código fuente, en al menos 3 navegadores diferentes.

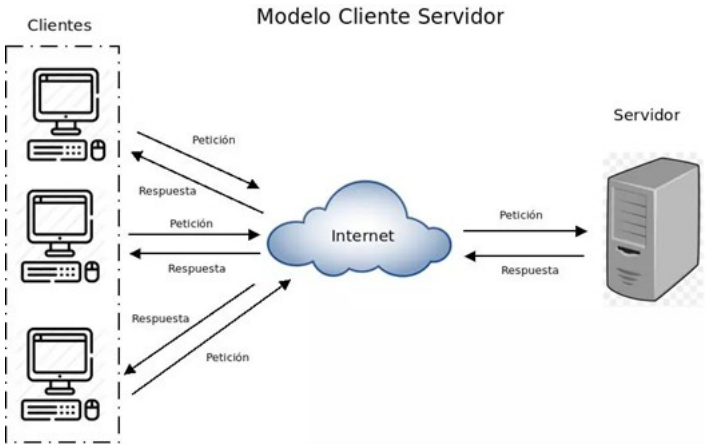
1. MODELOS DE PROGRAMACIÓN EN ENTORNOS CLIENTE/SERVIDOR

El modelo **cliente/servidor** es un modelo de **diseño** de software, donde los procesos se reparten entre dos partes diferenciadas:

- **Servidor**: son los **proveedores** de servicios o recursos.
- **Clientes**: son los encargados de usar los servicios, o **demandantes** de ellos.

En este modelo, un cliente realiza peticiones a un programa alojado en el servidor, el cual le da la respuesta que necesita.

Cuando hablamos de clientes o servidores podemos referirnos a procesos que se ejecutan en máquinas distintas, o incluso a procesos que se ejecutan en la misma máquina, aunque lo común suele ser la primera opción.



➔ **Capacidad de proceso**: en este modelo la capacidad de proceso está **repartida entre los clientes y los servidores**, de tal manera que cada uno se encarga de una parte del trabajo. Dependiendo del sistema, podemos tener la carga más balanceada hacia el lado del cliente o del servidor.

➔ **Ventaja**: la gran ventaja (pese a que a veces pueda convertirse en desventaja) está en la **implementación de la organización**, donde la centralización de la

información y la separación de responsabilidades facilita y esclarece el diseño del sistema.

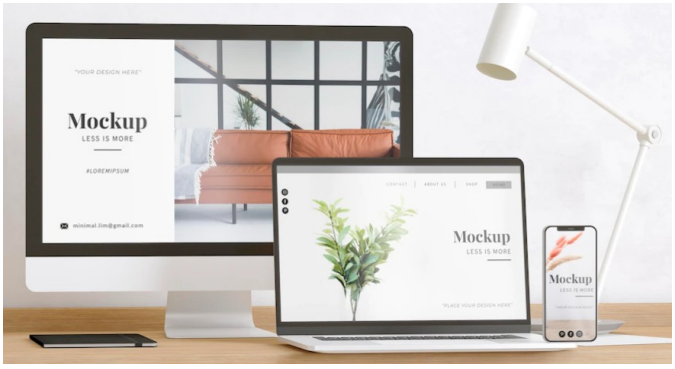
- **Aplicaciones informáticas**: algunos ejemplos de aplicaciones informáticas que utilizan el modelo cliente/servidor son, el **correo electrónico**, un **servidor de impresión** o el más usado de todos, la **WWW** (World Wide Web).
- **Otros modelos**: existen otros modelos que también hace uso de clientes y servidores, pero que se organizan de manera diferente:
  - **Modelos P2P**: donde todos los nodos son servidores y clientes a la vez.
  - **Modelo cliente/cola/cliente**: donde un servidor es cliente de otros servidores y agrupa las peticiones en una cola, para, finalmente, dar respuesta al cliente final.

Vídeo: modelo cliente servidor

Visualiza este vídeo en el que podrás ver una explicación simple sobre el modelo cliente servidor:
https://www.youtube.com/embed/49zdlyLSwhQ

2. MECANISMOS DE EJECUCIÓN DE CÓDIGO EN UN NAVEGADOR WEB

Para entender los diferentes **mecanismos de ejecución de código** en un navegador web, primero hemos de tener claro qué tipo de código se puede ejecutar en estos navegadores, y para ello es crucial entender la diferencia entre dos tipos de páginas web:



- **Páginas web estáticas**: son páginas, generalmente escritas en **código HTML, cuyo contenido no cambia dependiendo de ninguna petición externa, si no, que permanece inaltera**ble en cada visita (a menos que se modifique expresamente el código, claro está).

TOMA NOTA

Aquí estaríamos hablando sobre todo de las primeras páginas web que existieron, así como páginas por ejemplo de blogs o de simple información.

- **Páginas web dinámicas**: son páginas web que nos proporcionará, normalmente, el servidor web, con un **contenido diferente en función de la petición que le hagamos**.

TOMA NOTA

En este caso hablamos de páginas, por ejemplo, de una web de una tienda, de reservas de vuelos, o de cualquier otra página que no tenga un contenido fijo.

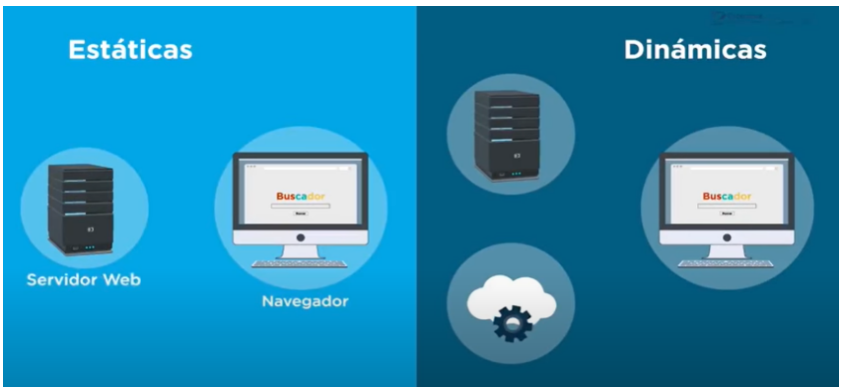


Imagen cogida del vídeo de youtube: Diferencia entre páginas web estáticas y dinámicas.

Enlace: https://www.youtube.com/watch?v=jBuAK1P9GwU.

Autor: Curso laboral

Las páginas web dinámicas en el cliente son pues, aquellas que están diseñadas para que se **interprete o ejecute su código en el lado del cliente**.

Esto permite ciertas ventajas, como liberar al servidor de carga, reducir el ancho de banda utilizado u ofrecer respuestas inmediatas al usuario.

TOMA NOTA

En los capítulos siguientes veremos más detalles acerca de la ejecución de estas páginas en el navegador web del cliente.

3. CAPACIDADES Y LIMITACIONES DE EJECUCIÓN

DESTACADO

Cuando hablamos de programación web desde el entorno cliente, nos referimos a scripts (independientemente del lenguaje) que se ejecutan en la máquina del cliente, y, por tanto, generalmente en el navegador web del usuario.

A diferencia de la programación tradicional, en entorno escritorio, por ejemplo, estos scripts tienen una serie de **limitaciones** en su ejecución. Estas **limitaciones** están ligadas al hecho de que el usuario pueda confiar en el código que se va a ejecutar dentro de su máquina.

A continuación, trataremos de enumerar las **principales limitaciones de los scripts**:

- Un script no puede **comunicarse con recursos** que no pertenezcan a su propio dominio.
- Un script no puede **cerrar ventanas** que no hayan sido abiertas por él mismo.
- Las ventanas abiertas no pueden colocarse **fuera de la vista** del usuario.
- Un script no puede acceder a **archivos** del ordenador del usuario, a menos que éste le dé permisos explícitamente.
- Si el navegador web detecta que la ejecución de un script está durando demasiado, o consume **demasiados recursos**, avisará al usuario para poder detener su ejecución si así lo desea.

TOMA NOTA

Pese a todas estas limitaciones, como siempre, existen bastantes alternativas que permiten saltárselas, siendo la principal, la firma digital del script, o pedir permisos al usuario para poder realizar estas acciones.

4. COMPATIBILIDAD CON NAVEGADORES WEB. CONFIGURACIÓN DE UN NAVEGADOR

Hemos visto que el principal entorno de ejecución de código web es el **navegador web** del cliente. Esto hace que no todos los navegadores web se comporten de la misma manera en cuanto a la ejecución de ese código, dependiendo del propio navegador, o incluso de las diferentes versiones.

➔ **Concepto**:

DESTACADO

El navegador, pues, se puede considerar como una interfaz de usuario universal, con las siguientes funciones:

- *Petición de páginas web al servidor.*
- *Representación de manera adecuada de su contenido.*
- *Gestión de los posibles errores que se puedan producir.*

➔ **Contenido de los scripts**: en función del contenido de los **scripts**, nos podemos encontrar con que los diferentes navegadores no sean capaces de ejecutarlos de manera correcta.

Por ejemplo, si en nuestro código hacemos uso de los **acelerómetros** de un dispositivo móvil, no todos los navegadores serán capaces de controlar esas funciones.

➔ **Compatibilidad del script**: lo ideal sería que un script se pudiera ejecutar de la misma manera en todos los navegadores, es decir que fuera **compatible** con todos. Por desgracia, si intentamos que siempre sea así, podemos estar perdiendo posibles **capacidades**, por el hecho de que no se puedan ejecutar en algún navegador.

Por este motivo, siempre habría que intentar tener un término medio entre la **compatibilidad** del script y las **tecnologías** que utiliza. Y en caso de no ser posible, al menos, crear diferentes **versiones** de éstos para adaptarse a los distintos navegadores.

Existen diferentes **recursos** para comprobar la compatibilidad de un navegador web. Uno interesante es Can i use, que permite si podemos usar ciertas tecnologías en los distintos navegadores.

- **Configuración de un navegador**: la ejecución de scripts en el navegador no siempre está habilitada por defecto, por lo que a menudo tenemos que entrar a la **configuración** de éstos, para permitir su ejecución.

  Esto se debe, como hemos dicho antes, a las **limitaciones** de ejecución de los scripts y, sobre todo, a temas de seguridad que hacen que el usuario tenga que permitir explícitamente la ejecución para evitar males no deseados.

En los recursos para ampliar puedes encontrar información acerca de cómo habilitar JavaScript en distintos navegadores.

Vídeo: can I use

Visualiza este vídeo en el que hablamos sobre la página Can I use.

Cuestionario

Lee el enunciado e indica la opción correcta:

¿Cómo se denomina el modelo de conexión entre ordenadores por red más utilizado?

a. Cliente / Servidor.

b. P2P.

c. HTML.

Lee el enunciado e indica la opción correcta:

¿Cuál de los siguientes es un lenguaje de creación de páginas web estáticas?

a. Java.

b. HTML.

c. JavaScript.

Lee el enunciado e indica la opción correcta:

¿Cómo se denomina al tipo de lenguajes más común en programación web en entorno cliente?

a. Lenguajes anidados.

b. Lenguaje JavaScript.

c. Lenguajes de Script.

5. CARACTERÍSTICAS DE LOS LENGUAJES DE SCRIPT. VENTAJAS Y DESVENTAJAS SOBRE LA PROGRAMACIÓN TRADICIONAL

En este apartado veremos algunas **características** de los **lenguajes de script**:

- La utilización de lenguajes de script es la forma más habitual de dotar de **dinamismo** a las páginas web desde el punto de vista del cliente.
- Los scripts nos permiten la ejecución de código que normalmente irá **asociado a eventos** que, en general, se generan a partir de acciones que realiza el usuario en la página. Por ejemplo, pulsar un botón, entrar en una caja de texto, pasar el cursor sobre una imagen…
- Los lenguajes de script también pueden actuar sobre el navegador mediante los **objetos integrados**, que representan al documento, la ventana activa, o cada uno de los elementos de un formulario.
- Hay que destacar que algunos lenguajes de script, como JavaScript, también pueden implementarse en el **lado del servidor**, aunque no es lo más común.
- Veamos ahora algunos **usos habituales** de los lenguajes de script:
  - **Validar** datos de un formulario en el cliente y comprobar su **consistencia** antes de enviar el formulario.
  - Actualizar los **campos relacionados** en un formulario.
  - Realizar procesos que no requieran de la utilización de información centralizada den el servidor.
  - Servir de **base** para la utilización de otras tecnologías (HTML dinámico, XML, ActiveX…).

En general podemos decir que los lenguajes de script son bastante **seguros**. Existen fuertes **restricciones** de acceso a los recursos de la máquina, no por imposibilidades tecnológicas, si no por las propias restricciones de los diseñadores. Esto hace que navegar por una web de este tipo, en general, no suponga **ningún riesgo** para el usuario.

A diferencia de la programación tradicional, los lenguajes de script son lenguajes que están diseñados para integrarse con código de otros lenguajes de programación, como por ejemplo HTML.

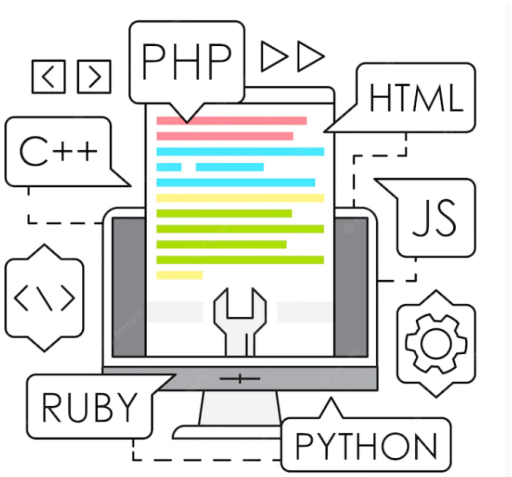
Además, los lenguajes de script son **interpretados**, por el contrario, los lenguajes tradicionales suelen compilarse, previamente a su ejecución.

Figure: Compiler

Figure: Interpreter

6. LENGUAJES DE PROGRAMACIÓN EN ENTORNO CLIENTE. CARACTERÍSTICAS

Los lenguajes de programación en entorno cliente nos permiten generar el código que, usualmente, se ejecutará en el **navegador** web del usuario.



- Entre estos lenguajes de programación podemos distinguir dos **tipos** principales:
  - ✔ Lenguajes que permiten diseñar la **estructura**, dar **formato** y **estilo** a las páginas web. Aquí entrarían lenguajes como HTML o CSS, por ejemplo.
  - ✔ Lenguajes que permiten que una página web sea **dinámica**, es decir, que su contenido varíe en función de las peticiones del usuario. Serían lo que se conoce como lenguajes de **script**.
- **Generar contenido en el cliente**: hay que tener en cuenta que utilizar cualquier lenguaje para generar **contenido dinámico** en el **cliente** no exime de tener que utilizar lenguajes para lo propio en el **servidor**. Una página web dinámica, generalmente requiere del dinamismo tanto en la parte del cliente como del servidor.
- **Aplicaciones**: otra característica importante de los lenguajes de script es que no sólo están presentes en la programación de páginas web, si no que **intervienen en muchas otras aplicaciones**. Como ejemplos podemos citar a Google Docs, Adobe Acrobat u OpenOffice.
- **Limitaciones**: por último, remarcar, como ya hemos dicho anteriormente, que las propias **limitaciones** de los lenguajes de script hacen que se consideren lenguajes **seguros**, en cuanto, por ejemplo, al tratamiento de información del usuario.

A lo largo del módulo veremos principalmente los lenguajes que permiten que una página web sea **dinámica**, basándonos sobre todo en el lenguaje más utilizado a día de hoy, **JavaScript**.

Pese a eso, para un correcto aprovechamiento del módulo es más que recomendable tener conocimientos sobre lenguajes de diseño (que se ven en otros módulos), para poder entender la interacción entre ellos.

7. TECNOLOGÍAS Y LENGUAJES ASOCIADOS

A continuación, veremos algunas de las **tecnologías** más utilizadas en la programación web en entorno cliente, remarcando los **lenguajes** principales:

- **Lenguajes de script**: como ya hemos visto, es la forma más habitual de dotar de **dinamismo** al cliente.

  La utilización del lenguaje **JavaScript** está muy extendida en las aplicaciones web.

- **VBScript**: es un lenguaje de programación de script que únicamente es **compatible con Internet Explorer**, cosa que no lo hace demasiado útil.

  Está basado en el lenguaje de escritorio **Visual Basic**, aunque tienen sus claras diferencias.

- **ActiveX**: esta tecnología de **Microsoft** nos permite ejecutar **código máquina** en los equipos de lo usuarios.

  Funciona tanto en el Explorer de Microsoft como en otros navegadores.

  Tiene algunas **ventajas** como la potencia, eficiencia o sencillez, pero, por el contrario, **desventajas** como la poca seguridad, ya que es posibles que estas aplicaciones puedan obtener el control total de la máquina.

- **Applets de Java**: su filosofía es **similar al ActiveX**, pero la forma en cómo los utilizan los navegadores es muy diferente.

  Un Applet no puede ejecutarse de forma independiente, ya que necesita de un **contexto**, como por ejemplo un navegador web.

  Son aplicaciones creadas en Java, que presentan unas ciertas restricciones y que pueden ser enviadas por internet desde un servidor a un cliente.

- **Adobe Flash**: pese a ser una tecnología prácticamente **obsoleta**, su uso llegó a ser muy **extendido** y aún sigue funcionando en múltiples páginas web.

Es una tecnología de **animación** con licencia de Adobe y que utiliza **ActionScript** como lenguaje principal.

Se consideraba muy útil para complementar las páginas web en el cliente, debido, sobre todo, a la **sencillez** a la hora de crear gráficos y animaciones vectoriales.

Sin embargo, su elevado consumo de CPU, **incompatibilidades** o la aparición de nuevas tecnologías como el **HTML5**, han sido las razones de su ocaso.

Actividad de aprendizaje 1: entornos de desarrollo integrados

Para complementar tus estudios en programación web, estás valorando la idea de empezar a realizar algunos trabajos por tu cuenta, a modo de aprendizaje y portfolio.

Lo primero que tienes que hacer es tener claro las tecnologías y entornos que usarás, así que le pides ayuda a un compañero, y entre los dos realizáis un pequeño dossier con la comparativa de los diferentes entornos de desarrollo integrados (tanto gratuitos como de pago) disponibles.

Enumera sus características, licencias, ventajas e inconvenientes.

Finalmente, tenéis que escoger uno de entre todos y argumentarlo.

Entrega un archivo PDF en el espacio habilitado para ello.

Cuestionario

Lee el enunciado e indica la opción correcta:

Los lenguajes de script suelen ser…

a. Interpretados.

b. Compilados.

c. Ninguna de las dos.

Lee el enunciado e indica la opción correcta:

¿Dónde se ejecuta normalmente el código web en entorno cliente?

a. Navegador web.

b. Aplicación de escritorio.

c. Aplicación móvil.

Lee el enunciado e indica la opción correcta:

¿Qué es CSS?

a. Un lenguaje dinámico.

b. Un lenguaje para estructurar páginas.

c. Un lenguaje para dar estilo.

8. HERRAMIENTAS DE PROGRAMACIÓN. ENTORNOS DE DESARROLLO INTEGRADOS. PERSONALIZACIÓN. DOCUMENTACIÓN DE CÓDIGO

DESTACADO

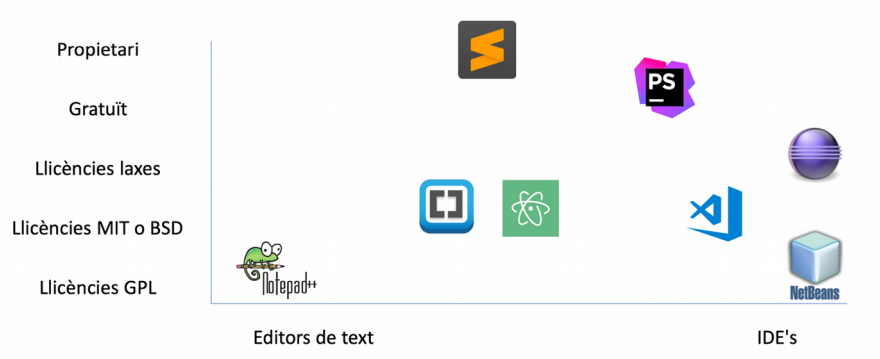
Un entorno de desarrollo integrado, a partir de ahora IDE, es un conjunto de herramientas que nos facilitan todos los procesos que conlleva la programación. Desde la creación de código, la compilación o ejecución hasta la depuración de errores.

TOMA NOTA

Actualmente, en el mercado podemos encontrar IDEs para todos los gustos, tanto de programación web como de programación de escritorio. Incluso muchos de ellos son compatibles con ambos mundos.

Existen **entornos de desarrollo gratuitos**, de pago, open source, propietarios, multiplataforma… Algunos incluso aportan características **WYSIWYG**, lo cual hace mucho más visual el proceso de programación.

Lo más importante para su elección es sobre todo **sentirse cómodo con él**, y que tenga **soporte para los lenguajes y herramientas** que necesitarás como desarrollador: HTML, XML, JSON, PHP, JavaScript, JSP…



En la imagen anterior podemos ver una serie de **IDE's** organizados en relación a su complejidad y licencia. Siendo los más utilizados para VSCode, Sublime o Eclipse.

La mayoría de IDE's, o al menos los más complejos permiten una total personalización de su interfaz, de manera que el desarrollador lo pueda hacer suyo, y se sienta lo más cómodo posible trabajando.

Esto es esencial, porque ciertas cosas que son importantes para un desarrollador, no lo serán tanto para otro, por lo que poder mostrar u ocultar ciertas opciones hace mucho más **rápido** el trabajo.

Vídeo: ¿qué es un IDE de programación?

Visualiza el siguiente vídeo sobre ¿qué es un IDE de Programación? Significado de entorno de desarrollo integrado: https://www.youtube.com/embed/_WKWpJEv9UY

9. DEPURACIÓN DEL CÓDIGO. ANÁLISIS Y TRATAMIENTO DE ERRORES. HERRAMIENTAS DEL NAVEGADOR

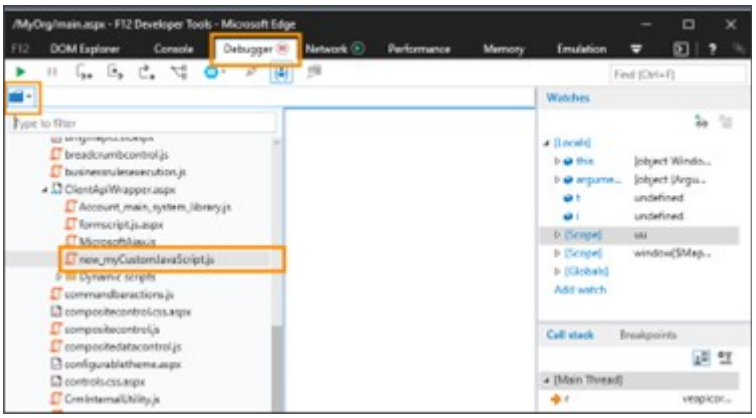
En programación, ya sea web o de escritorio una de las partes principales en el proceso de desarrollo, es la **prueba** del código. Probar nuestro código en varios **entornos** es esencial para evitar errores cuando éste pase a producción.

- **Depuración**: una de las herramientas más importantes para la prueba de aplicaciones es la **depuración** (**debug** en inglés), lo cual nos permite ejecutar nuestro código, línea a línea y en un entorno controlado, para ver lo que va sucediendo.

  Prácticamente la totalidad de IDEs actuales poseen algún tipo de herramienta de depuración, más o menos compleja.

- **Depuración de JavaScript**: en el caso de los lenguajes de script, y en concreto de **JavaScript**, que es el lenguaje que utilizaremos durante el curso, la depuración, además, se puede hacer a través del propio **navegador web**.

  Esto es realmente útil, ya que nos permite probar nuestro código directamente en **diferentes navegadores**, viendo cómo reaccionan nuestras aplicaciones en cada uno de ellos.



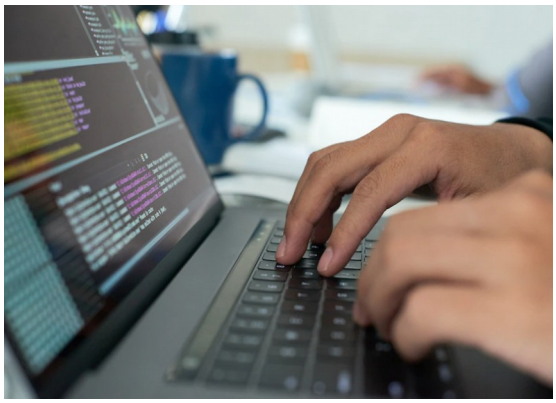
Los **navegadores** más habituales poseen alguna opción en el menú que permite abrir una **consola**, o **herramienta de depuración**.

Cuando hablemos de JavaScript en posteriores unidades ahondaremos más en este tema. De momento, puedes echarle un vistazo a los recursos para ampliar, donde verás como depurar código en distintos navegadores.

Visualiza este vídeo en el que hablamos sobre la depuración en el navegador.

10. INTEGRACIÓN DEL CÓDIGO CON LAS ETIQUETAS HTML, XML Y OTROS. CÓDIGO EN LÍNEA Y EN FICHEROS EXTERNOS. CUMPLIMIENTO DE LOS ESTÁNDARES DE W3C

Los lenguajes de script permiten la creación de código para hacer dinámicas las páginas web. No obstante, en la mayoría de ocasiones necesitamos combinar este código con el creado para definir el diseño y estilo de la página.



Es por ello, que necesitamos mecanismos para poder **integrar** código de script dentro de código **HTML**, XML o similar.

Existen varias opciones de **insertar código** de script dentro de una página HTML, siendo la más popular la etiqueta "**script**". Además, mediante un atributo "**type**", podemos especificar el tipo del lenguaje de script que estamos insertando.

- **Código en línea**: al estilo de las **etiquetas de HTML**, tendremos que abrir y cerrar el bloque, quedando de la siguiente manera:

```
<script  type="text/javascript">

   //  Código JavaScript

</script>
```

  Esto es lo que se llama **código en línea**. Dicho de otra manera, incrustado dentro del código HTML.

- **Ficheros externos**: hay otra manera de utilizar código de script dentro de HTML, mediante **ficheros externos**. Si usamos, por ejemplo, **JavaScript**, la

idea es escribir el código en un fichero con extensión .js, y referenciarlos desde la etiqueta "script" de HTML.

Teniendo el código JavaScript en un fichero llamado "código.js", el código HTML quedaría así:

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

En este caso estamos referenciando únicamente el **nombre** del fichero, pero podríamos usar una **ruta relativa**, indicando la carpeta donde está contenido, o incluso, una **ruta absoluta** o una ruta dentro de un **dominio**.

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

Esta segunda manera de trabajar, mediante ficheros externos, hace que el código HTML quede mucho más **limpio**, y el código de script más **organizado**.

Otro punto a tener en cuenta a la hora de insertar código de script, es que deberíamos respetar, en lo posible, los **estándares W3C**. Esto hace que nuestro código pueda ser compartido y comprendido por cualquier desarrollador, por ejemplo, si trabajamos en equipo, al seguir unas normas de escritura comunes.

Puedes encontrar más información sobre W3C en la bibliografía.

11. NAVEGADORES. TIPOS Y CARACTERÍSTICAS

En el mundo del **desarrollo web** es de vital importancia el poder llegar al máximo de gente posible a través de los **navegadores**. Para eso es imprescindible conocer los distintos tipos de navegadores del mercado e intentar que nuestras **aplicaciones** tengan el máximo de compatibilidad con ellos.

A día de hoy existen **dos grandes tipos de navegadores**, los que están basados en **Chromium** y los que no. Pero veamos antes qué es Chromium:

DESTACADO

Chromium es un proyecto de código abierto que busca hacer un navegador ligero y sencillo. Podemos decir que es la base para diseñar otros navegadores a partir de él.

Los principales **navegadores basados en Chromium** son:

- **Google Chrome**: propiedad de Google, es uno de los navegadores más utilizados del mundo. Desde su creación en 2008, ha ido aumentado su adopción. En algunos momentos ha pecado de exceso en el uso de recursos del sistema, provocando un aumento en el gasto de memoria.
- **Microsoft Edge**: propiedad de Microsoft, es la evolución natural del veterano Internet Explorer. Durante años fue la principal alternativa, al ser el navegador por defecto de Windows, lo cual le llevó incluso a problemas legales relacionados con el monopolio.
- **Chromium**: propiedad de Google, y como hemos dicho, la base del resto de estos navegadores. Destaca por ser de código abierto.
- **Opera**: otro de los veteranos en el mundo de los navegadores, aunque, desde luego, no de los más populares para la mayoría del público.

De entre los que **no están basados en Chromium** hay que destacar:

- **Mozilla Firefox**: propiedad de Mozilla, es otro de los veteranos y más utilizados durante mucho tiempo. Es la evolución natural, desde 2004 del obsoleto Netscape. A día de hoy ha dejado de ser de los más populares, en favor de Chrome.
- **Safari**: propiedad de Apple, es el navegador por defecto de los dispositivos iOS y ordenadores Mac, por lo que lo hace ser de los más populares dentro de su ecosistema. Eso sí, fuera del mundo Apple su uso es bastante escaso.

Actividad de aprendizaje 2: insertar JavaScript

Ahora que ya tienes más claras las tecnologías a utilizar y el entorno de desarrollo escogido llega el momento de hacer las primeras pruebas. Por desgracia aún no has empezado a estudiar el lenguaje JavaScript, pero sí como insertar éste en una página HTML.

Dado el siguiente fichero con un código simple en JavaScript: fichero fecha.

Para mostrar la hora:

- Insértalo en una nueva página HTML (lo más simple posible), como código en línea.
- Haz lo mismo, pero en forma de fichero externo.

Entrega los dos archivos HTML creados en el espacio habilitado para ello.

Cuestionario

Lee el enunciado e indica la opción correcta:

¿Cómo se denomina al conjunto de herramientas que facilitan el trabajo al desarrollador?

a. Editor de textos.

b. Navegador web.

c. IDE.

Lee el enunciado e indica la opción correcta:

¿Qué significa depurar un código?

a. Convertir a código máquina.

b. Ejecutar paso a paso en busca de errores.

c. Comentar un código.

Lee el enunciado e indica la opción correcta:

¿Qué etiqueta HTML se usa para insertar código de script?

a. Script.

b. Type.

c. JavaScript.

RESUMEN

Esta unidad ha servido a modo introductorio de los conocimientos que se irán adquiriendo a lo largo del módulo. Es quizás la unidad más teórica que veremos, ya que requerimos de esta base para seguir trabajando.

Hemos conocido las diferentes **arquitecturas de programación web**, centrándonos en el **modelo cliente/servidor**, que es el más utilizado actualmente.

Hemos hecho un repaso por las distintas **tecnologías y lenguajes para la programación web en entorno cliente**. También repasado las diferentes **herramientas y entornos de desarrollo** que nos ayudarán en todas las tareas de programación.

Finalmente hemos podido ir viendo conceptos sobre los **navegadores**, que es el entorno dónde se ejecutará nuestro código en el cliente. Conociendo sus **compatibilidades**, **tipos** y **características**.

Ahora ya estamos preparados para el siguiente paso. Conocer en profundidad el lenguaje **JavaScript**.

RECURSOS PARA AMPLIAR

PÁGINAS WEB

- Depuración del código JavaScript para aplicaciones basadas en modelo - Power Apps: *https://docs.microsoft.com/es-es/power-apps/developer/model-driven-apps/clientapi/debug-javascript-code* [Consulta mayo 2022].
- La guía definitiva para depurar JavaScript: *https://tech-wiki.online/es/javascript-debugging.html* [Consulta mayo 2022].
- Depurar javascript con Chrome: *http://chuwiki.chuidiang.org/index.php?title=Depurar_javascript_con_Chrome* [Consulta mayo 2022].
- Cómo habilitar JavaScript en tu navegador y por qué hacerlo: *https://www.enable-javascript.com/es/* [Consulta mayo 2022].

BIBLIOGRAFÍA

PÁGINAS WEB

- W3C: *https://www.w3.org/* [Consulta mayo 2022].
- Can I use... Support tables for HTML5, CSS3, etc: *https://caniuse.com/* [Consulta mayo 2022].

GLOSARIO

- **HTML**: lenguaje de marcas de hipertexto. Es el principal lenguaje para la creación de páginas web estáticas.
- **Java**: lenguaje de programación de escritorio, de los más utilizados, que permite además la creación de Applets.
- **Script**: programa relativamente sencillo, que consiste en una secuencia de comandos.
- **WWW** (World Wide Web): es la mayor red mundial, que funciona a través de Internet y en cual se alojan la inmensa mayoría de páginas web.
- **WYSIWYG** (What You See Is What You Get): funcionalidad que permite ver el resultado final del código a medida que se va creando.