# UT1. Selección de arquitecturas y herramientas de programación

1) Repaso conceptos:
    1) Cliente y servidor web, HTML, http.
    2) Concepto de programa, proceso, demonio y variable.
2) Desarrollo web.
3) Lenguajes de programación en clientes web.
    1) Características.
    2) Compatiblidades.
    3) Seguridad.
4) Herramientas y útiles de programación.
5) Integración de código JavaScript con HTML.

# Desarrollo web

**WEB** creada por **Tim Berners-Lee**, especialista de laboratorio del CERN, para compartir información científica en 1989. El prototipo **se basa en**:

- **Hipertexto**: Es aquel texto de tal forma que, cuando pulsamos en él nos conduce a otro texto, objeto, sonido, video, sección o documento relacionado.
- **Protocolos**: utilizados para la comunicación entre los servidores web (donde residen las páginas web) y los clientes web (solicitantes de dichas páginas web). Las peticiones de páginas se realizan mediante estos protocolos.

# Desarrollo web

**NCSA** (National Center for Supercomputing Applications) **realizó muchas aportaciones** a este prototipo.

**Las evoluciones posteriores** se hicieron **bajo el consorcio W3C** (Consorcio World Wide Web), organización con base **MIT** (Massachusetts Institute of Technology) que desarrolla y mantiene los estándares web.

# Lenguajes de programación en clientes web

Una página **HTML** es **estática** por sí misma, no puede interactuar con el usuario, o su interacción es muy limitada.

Las páginas **HTML dinámicas** (DHTML) son aquellas que **interactúan con el usuario**. Por ejemplo mediante botones, gestión de eventos, etc.

La **interacción** se consigue mediante lenguajes de scripting (**JavaScript**)

# Lenguajes de programación en clientes web

En los formularios web que rellena el usuario y cuyos datos van a ser enviados al servidor para que, por ejemplo, los introduzca en una Base de Datos, **esta interacción que aporta JavaScript tiene los siguientes objetivos**:

- Ayudar a la introducción de datos por parte del usuario para mermar su frustación.
- Chequear la información introducida por el usuario antes de mandar esos datos al servidor y generar errores en el mismo.

Cuando los datos lleguen al servidor, éste también validará la integridad de los datos, en forma y en relación a los datos almacenados en la base de datos.

# Lenguajes de programación en clientes web (tipos de lenguajes)

Por tanto, hay **dos tipos de lenguajes de programación**:

- En el **lado cliente (client-side)**, para interactuar con el usuario filtrando la entrada y recepción de datos.
  - Se **ejecutan en el cliente**.
- En el **lado servidor (server-side)** para ejecutar las peticiones del usuario en una base de datos, como inserción o búsqueda de datos. Esto se hará en el servidor puesto que será ahí donde resida la base de datos.
  - Se **ejecutan en el servidor**.

**AMBOS CÓDIGOS SE ALMACENAN EN EL SERVIDOR WEB, así como las propias páginas HTML.**

# Lenguajes de programación en clientes web (ejemplos)

Ejemplos de lenguajes de programación:

- En el **lado cliente:** JavaScript (el más estandar), Jscript de Microsoft, ….
- En el **lado servidor:** php, asp, perl, java, python, ...

# Lenguajes de programación en clientes web (4 capas)

Se dan, entonces, **4 capas de desarrollo web** en el entorno cliente:

1) Comportamiento (JavaScript).
2) Presentación (CSS).
3) Estructura DOM (estructura HTML).
4) Contenidos (imágenes, videos, audios, texto,..).

# Lenguajes de programación en clientes web (DOM)

DOM es una interfaz de programación de aplicaciones (**API) para documentos HTML**. Define la **estructura lógica** de los documentos y el **modo en que se accede y manipula un documento**.

Los lenguajes de programación acceden y manipulan los elementos HTML.

Para **evitar los problemas de falta de estandarización**, un organismo internacional (el W3C) definió este estándar **DOM** que **define qué elementos se considera que conforman una página web, cómo se nombran, cómo se relacionan entre sí, cómo se puede acceder a ellos, como se modifican, etc.**

# Lenguajes de programación en clientes web (DOM)

También se utiliza DOM en el siguiente contexto. Decimos que una página web es un documento HTML. Este documento puede ser representado de diferentes maneras:

1) **Representación web**: como una página web en un navegador donde vemos imágenes, texto, colores, etc.
2) **Representación texto:** como un texto plano (código HTML) que podemos visualizar en cualquier editor de textos como el bloc de notas de Windows ó Notepad++ ó cualquier otro.
3) **Representación DOM:** como un árbol donde los elementos de la página web están organizados jerárquicamente, con nodos superiores (nodos padre o parent) y nodos que derivan de los nodos padre (nodos hijo o child).

# Lenguajes de programación en clientes web (Ejemplo de representación DOM)

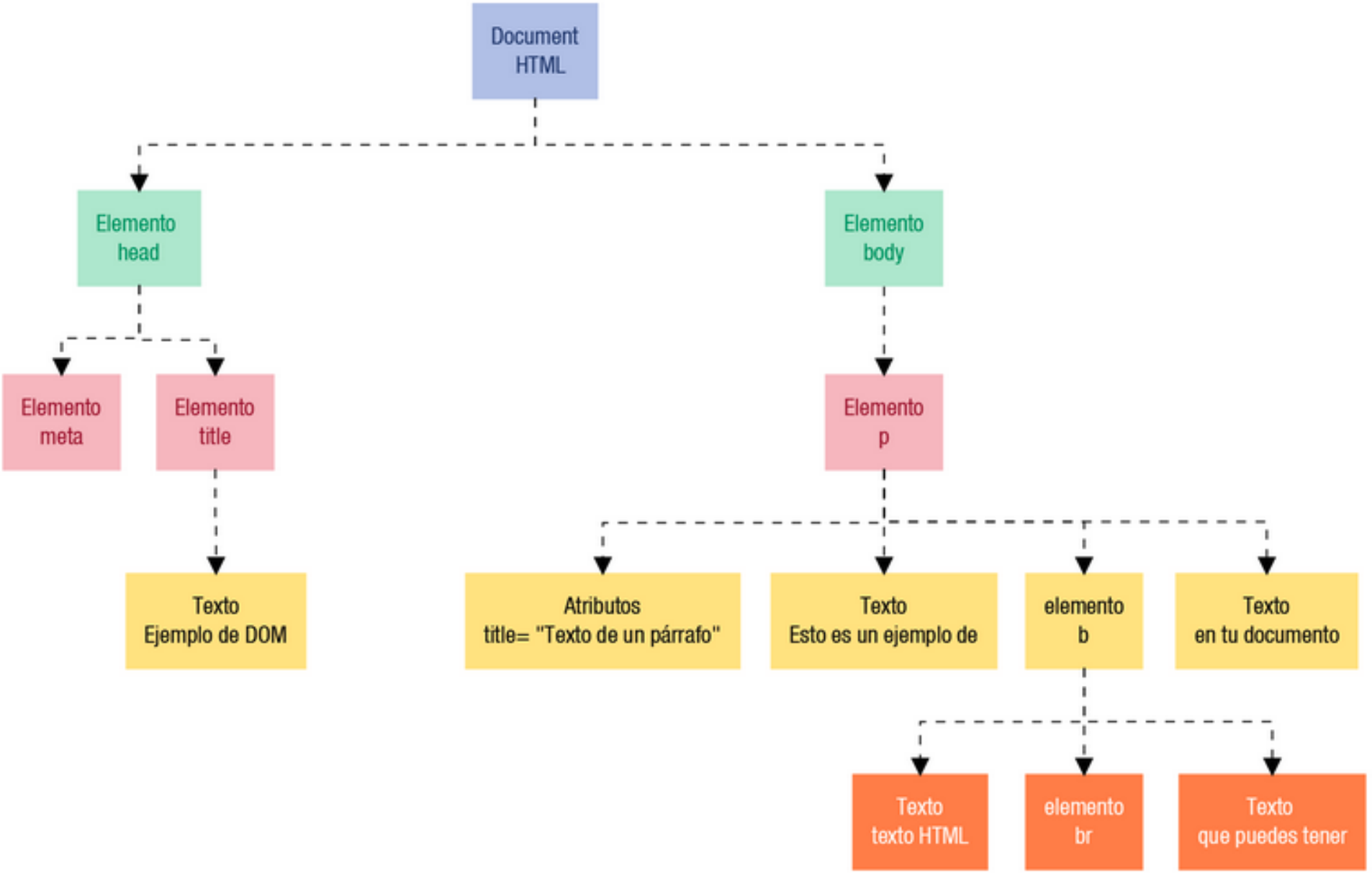
El DOM transforma **nuestro documento HTML en un conjunto de elementos, a los que llama nodos. Cada nodo es un objeto.**

Estos nodos están **conectados entre sí** en una estructura similar a un árbol, a lo que se llama árbol DOM o "**árbol de nodos**".

**Ejemplo de documento HTML:**

```
<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>

<title>Ejemplo de DOM</title>

</head>

<body>

<p title="Texto de un párrafo">Esto es un ejemplo de <b>texto
HTML<br />que puedes tener</b> en tu documento.</p>

</body>

</html>
```

# Lenguajes de programación en clientes web (Ejemplo de representación DOM)



# Lenguajes de programación en clientes web. Características

Los lenguajes de programación cliente web hacen **tareas** como:

- Interacciona con el usuario de forma rápida y directa, por ejemplo a través de los formularios. Le ayuda en el completado de los campos.
- Distribuye los datos recogidos del servidor ordenadamente y de forma amigable.
- Controla ventanas y frames, applets de Java y plugins, según lo indicado por el usuario.
- Pre-procesa datos antes de enviarlos al servidor.
- Modifica estilos y contenidos de la página HTML por interacción con el usuario.
- Solicita ficheros del servidor

# Lenguajes de programación en clientes web. Compatiblidades I

- JavaScript es **soportado por múltiples navegadores**: Firefox, Google Chrome, Safari, Opera, Internet Explorer, ….
- **Al escribir un código JavaScript tenemos que estar seguros de que es soportado por distintos navegadores** y que se puede dar la misma funcionalidad en cada uno de ellos. En los scripts de servidor el control de su ejecución es total, porque se ejecuta sobre el servidor en una plataforma que controlamos nosotros.
- Hay que **contar con los bugs de los propios navegadores, y de los sistemas operativos** sobre los que se ejecutan.

# Lenguajes de programación en clientes web. Compatiblidades II

- Existen **URLs que te indican qué aspectos de JavaScript son soportados por qué navegadores**: https://kangax.github.io/compat-table/es6/
- **A veces los errores** no **son debidos** a incompatiblidades del navegador con código JavaScript, sino por el **código HTML incluido**. **Importante**: seguir el estándar W3C. **Validador HTML W3C**: https://validator.w3.org/
- **No siempre va a ser posible la ejecución de código JavaScript en el cliente**, puesto que hay clientes que no lo soportan (algunos smartphone no lo soportan, a veces los usuarios lo desactivan, algunos navegadores de voz no lo soportan). Algunas partes del código JavaScript puede ser que no sea compatible con el navegador. Por todo ello no debemos confiar plenamente en que el código JS va a ser ejecutado. Utilizar **<noscript>**

# Lenguajes de programación en clientes web. Seguridad

**Para evitar que se distribuya scripts maliciosos** a través de la web, los navegadores web en el cliente aplican **dos tipos de restricciones de seguridad.** La mayor parte de los agujeros de seguridad son infracciones a estas dos restricciones de seguridad:

- **"espacio seguro de ejecución": solamente podrá realizar tareas relacionadas con la web**, nada de tareas genéricas de programación como creación de ficheros, lectura, etc...
- **"mismo origen": los scripts de una web no tendrán acceso a información tal como usuarios, contraseñas, o cookies enviadas desde otra web.**

**Las propias limitaciones de las tareas que se pueden realizar con código JS**.

# Lenguajes de programación en clientes web. Seguridad (motor de JavaScript)

JavaScript es un **lenguaje interpretado,** por lo que **requiere** de un **motor de ejecución javascript que interprete el código JS y lo ejecute.** Éstos son:

- Motores de JavaScript de los navegadores.
- Active Script de Microsoft, es compatible con JavaScript, pero tiene muchos aspectos que no cumplen el estandar.
- Herramientas Qt, incluye un intérprete de JavaScript.
- Java JDK 1.6 (Java Development Kit) tiene un paquete javax.script que permite la ejecución de Java Script.

**Una diferencia importante entre navegadores es la rapidez con la que sus motores de JavaScript pueden ejecutar las aplicaciones**, y la seguridad y aislamiento que ofrecen en la ejecución de las aplicaciones en diferentes ventanas o pestañas de navegación.

# Lenguajes de programación en clientes web. Seguridad (lenguaje interpretado vs compilado)

## COMPILADOS

Traducen a código máquina, creando un archivo traducido para una ejecución rápida.

Requieren que las instrucciones, sean traducidas, esto lo hace más eficiente.

El procesador sólo entiende un lenguaje que se denomina "lenguaje máquina".

## INTERPRETADOS

Las instrucciones se traducen una a una, cada vez que se ejecute el programa

Son típicamente unas 10 veces más lentos que los programas compilados.

Facilidad para lograr independencia de plataformas y menor tamaño de programa.

# Herramientas y útiles de programación (Editor I)

Para crear el código HTML y JavaScript **os recomiendo un editor de texto, en el que se vea el código tal cual es,** no una visualización del mismo como Dreamweaver o FrontPage.

Características buscadas en un editor que te pueden ayudar:

- **Resaltado de texto.** Muestra con distinto color o tipo de letra los diferentes elementos del lenguaje: sentencias, variables, comentarios, etc. También genera sangrado automático para diferenciar de forma clara los distintos bloques de un programa.
- **Completado automático.** Detecta qué estás escribiendo y cuando es posible te muestra distintas opciones para completar el texto.

# Herramientas y útiles de programación (Editor II)

- **Navegación en el código.** Permite buscar de forma sencilla elementos dentro del texto, por ejemplo, definiciones de variables.
- **Comprobación de errores al editar.** Reconoce la sintaxis del lenguaje y revisa el código en busca de errores mientras lo escribes.
- **Generación automática de código.** Ciertas estructuras, como la que se utiliza para las clases, se repiten varias veces en un programa. La generación automática de código puede encargarse de crear la estructura básica, para que sólo tengas que rellenarla.
- **Gestión de versiones.** el entorno de desarrollo te puede ayudar a guardar copias del estado del proyecto a lo largo del tiempo, para que si es necesario puedas revertir los cambios realizados.

# Herramientas y útiles de programación (Editor III)

**Editores válidos**:

- Atom.
- Sublime.
- Brackets.
- NotePad++

**Editores válidos por plataformas:**

- Para Windows tienes: Notepad++, Aptana Studio, Bluefish, Eclipse, NetBeans, etc.
- Para Macintosh tienes: Aptana Studio, Bluefish, Eclipse, KompoZer, Nvu, etc.
- Para Linux tienes: KompoZer, Amaya, Quanta Plus, Bluefish, codetch, etc.

# Herramientas y útiles de programación (Navegador)

**Para ejecutar tu código no necesitas un servidor web**, puede abrir tu fichero desde el navegador y lo ejecutará.

Si realizas cambios en el fichero, los guardas y das a recargar hasta conseguir completar el código que resuelve las especificaciones pedidas.

**Importantes comprobaciones a realizar:**

- **Validar que el código HTML sigue los estándares:** https://validator.w3.org/ Si el código HTML contiene errores es probable que tu código JavaScript no funcione.
- **Tu código  funciona en varios navegadores distintos.**

# Integración de código Javascript con HTML. Primera forma.

Directamente en el código HTML:

```
<button onclick="alert('Hola mundo!!!');">saludar</button>
```

# Integración de código Javascript con HTML. Segunda forma.

Introduciendo código JS en la página HTML entre las etiquetas **<script></script>**

```
<script type="text/javascript">
alert(“Hola mundo”);
</script>
```

**type="text/javascript"** se pone para indicar al navegador que el código que hay dentro de <script></script> es JavaScript, pero esto es antiguo, actualmente todos los navegadores reconocen JS sin indicar el tipo con esta etiqueta.

# Integración de código Javascript con HTML. Tercera forma I.

Fichero externo: (con **extensión .js**)

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

Es **la forma más recomendable**, ya que:

- Separación entre el código y la estructura de la página web.
- Compartir código entre diferentes páginas.
- Centralizar el código para la depuración de errores.
- Claridad en tus desarrollos, más modularidad, seguridad del código.
- Las páginas cargan más rápido, ya que si más de una página tiene que acceder a ese fichero lo cogerá automáticamente de la caché del navegador.

# Integración de código Javascript con HTML. Tercera forma II.

**Si usas un fichero externo (src) no se puede incluir código JScript dentro de las etiquetas script.**

**La ruta del fichero puede ser**:

- **`src="miScript.js"`**:significa que el fichero miScript.js se encuentra en el mismo directorio que el fichero html que lo está referenciando.
- **`src="./js/miScript.js":`**en un directorio js dentro del directorio en el que se encuentra el fichero html que está referenciando al script JS.
- **`Src="../js/miScript.js":`**en un directorio js dentro del directorio padre en el que se encuentra el fichero html que está referenciando al script JS.
- **`Src="/js/miScript.js":`**en un directorio js dentro del directorio raiz de nuestro servidor web.
- **`src="http://www.midominio.com/miScript.js":`** significa que el fichero JS está en el directorio raiz del dominio www.midominio.com.

# Integración de código Javascript con HTML. Visibilidad código I.

El código de un fichero JScript debe llegar al navegador cliente para ser ejecutado.

Ese **código está en abierto** porque JScript es un **lenguaje interpretado**. Cualquiera lo puede ver.

Si el código está en un fichero externo, con poner la ruta de ese fichero externo en el navegador se puede ver el código.

**Para evitar que otros utilicen tu código** tienes estas dos opciones:

- **Poner copyright** en tu código. Aunque otro desarrollador lo puede copiar sin copiar el copyright.
- **Ofuscar**:

# Integración de código Javascript con HTML. Visibilidad código II.

**Ofuscar:** lo que se sube al servidor web es el código ofuscado **(más difícil de entender)**

Para ello se utiliza las siguientes **técnicas de ofuscación:**

- Eliminación de saltos de línea.
- Eliminación de espacios en blanco innecesarios, tabuladores.
- Utilización de nombres ininteligibles en las funciones y variables.
- Utilización de variables para almacenar trozos de código.
- Uso de recursividad

**Un ofuscador:** https://www.javascriptobfuscator.com/default.aspx

# Integración de código Javascript con HTML. Visibilidad código III.

**Otra opción es compartir tu código.**

Incluir tu autoría en el fichero para publicitarte como buen programador.

Incluso podrás añadir una **licencia Creative Commons** para animar a la gente a que lo utilice, lo copie y lo mantenga público al resto del mundo.

Así **conseguirás mayor reputación** como buen programador y la gente contactará contigo para más información, posibles trabajos, etc.

**https://creativecommons.org/about/cclicenses/**

# Integración de código Javascript con HTML. `<noscript>`.

Conviene disponer de una etiqueta:

```
<noscript> Texto informativo </noscript>
```

Que te permitirá indicar un texto adicional que se mostrará indicando que ese **navegador no soporta JavaScript**.