

# Arquitecturas de aplicaciones web

# Introducción

## ○ Aplicaciones web

- A través de un navegador web (cliente liviano) se accede a una dirección de internet desde la que se obtendrán los servicios necesarios.

## ○ Lados de la arquitectura de aplicaciones web:

- Lado cliente. Donde se encuentra el usuario ejecutando su navegador web
- Lado del servidor. Donde se encuentra la aplicación y los datos que la misma utilice.

## ○ Tipos de aplicaciones según su acceso:

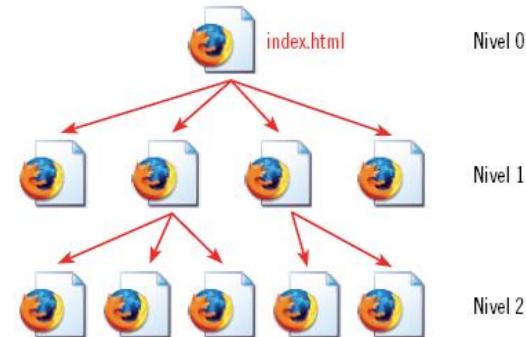
- De acceso público: portales de internet, tiendas virtuales, etc.
- De acceso restringido
  - Intranets
  - Extranets

# Estructura de un sitio

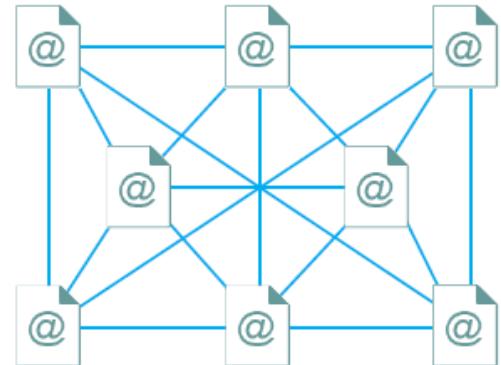
- Un sitio web está formato por un conjunto de páginas interrelacionadas entre sí.
- Cada página puede contener múltiples elementos de tipos muy diferentes:
  - texto
  - imágenes
  - videos
  - tablas
  - formularios
  - etc.
- Antes de comenzar a desarrollar un sitio web es muy recomendable elaborar la estructura de las secciones que contendrá el mismo
  - Existe multitud de software que nos permitirá elaborar esta estructura (como los ya vistos Balsamiq Mockups o JustindMind Prototyper)

# Tipos de estructuras

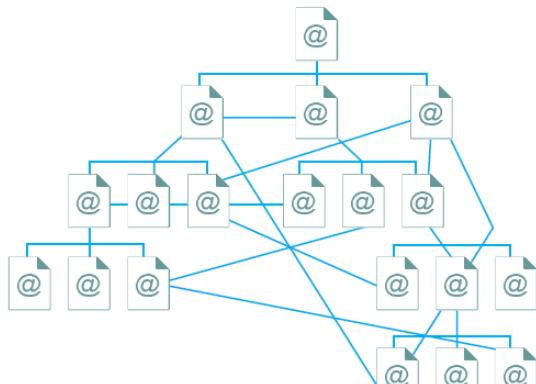
- **Jerárquica.** Estructura en árbol, existe una página principal o de bienvenida. Esta página nos dará acceso al resto de secciones. Cada vez que accedemos a una sección, ésta a su vez puede darnos acceso a otras subsecciones y así sucesivamente.



- **Red.** El usuario puede navegar entre las diferentes páginas sin un orden aparente.



- **Mixta.** Este tipo de estructuras es una mezcla de las anteriores



Etapas para  
desarrollar el  
diseño  
conceptual



- O Delimitación del tema.** Qué se pretenden alcanzar con el desarrollo de la plataforma y a quién irá dirigida (qué usuarios se quieren atraer al sitio web). Se define qué contenido llevará y cuál no.
- O Recolección de la información.** Recoger toda la información que va a tener la web (vídeos, imágenes, texto, etc.)
- O Agregación y descripción.** Clasificar la información de manera apropiada.
- O Diseño y estilo gráfico.** Hacer el mejor uso posible de las tecnologías apropiadas y asegurar que la imagen de marca del sitio se establezca y se mantenga.
- O Ensamble final.** Incorporar el estilo gráfico especificado, así como la totalidad del contenido de las distintas páginas que tendrá el sitio web.
- O Testeo.**
  - O** La comprobación de una página debe hacerse en cada fase, en lugar de esperar a que esté totalmente finalizada.
  - O** Debemos comprobar el funcionamiento de nuestra web en el mayor número de navegadores posible.

# Arquitectura en capas

# Introducción

- **Arquitectura en dos capas:** Cuentan con dos tipos de nodos en la red:
  - Clientes
  - Servidores
- **Arquitectura en tres capas.** Disponen de tres tipos de nodos
  - Clientes
  - Servidores
  - Servidores de base de datos

# Sistema de dos capas

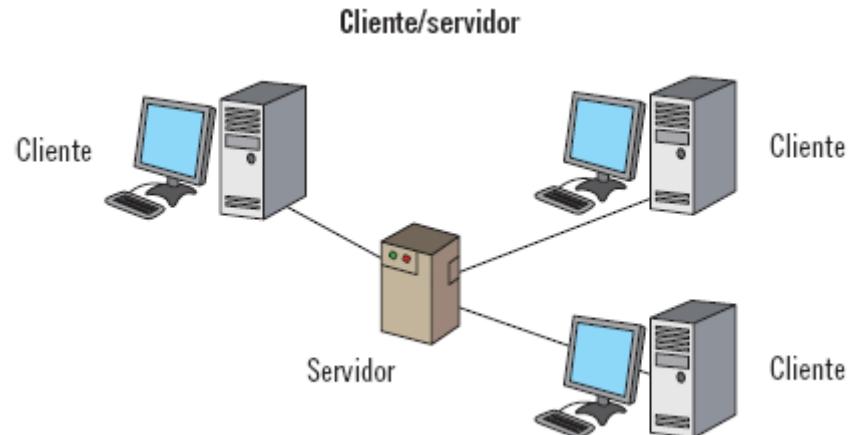
○ Cuentan con dos tipos de nodos: Cliente y Servidor

## ○ Ventajas

- El desarrollo de aplicaciones es mucho más rápido.
- El cliente se conecta al servidor cuando necesita realizar algún tipo de petición de información. Una vez recibida esta, la conexión se cierra y deja la red libre para que este u otro usuario pueda realizar otra conexión. De esta manera, se reduce de una forma muy importante el tráfico de red.
- Gran parte del proceso se ejecuta en el navegador (ordenador del cliente). Así se consigue mejorar el balance de carga en los diversos servidores.

## ○ Desventajas

- En los sistemas de dos capas, al encontrarse la mayoría de la aplicación en el cliente, a la hora de realizar cambios hay que volver a enviar la nueva versión a todos los clientes que la utilicen.
- Continuamente van apareciendo nuevas versiones, tanto del entorno cliente como de base de datos. El depender de una determinada herramienta puede comprometer las futuras ampliaciones y desarrollos.



### **Capa cliente/servidor**

El navegador solicita una página web

Navegador

La página web es devuelta al navegador

El *software* intermedio determina dónde está la página y la solicita al servidor

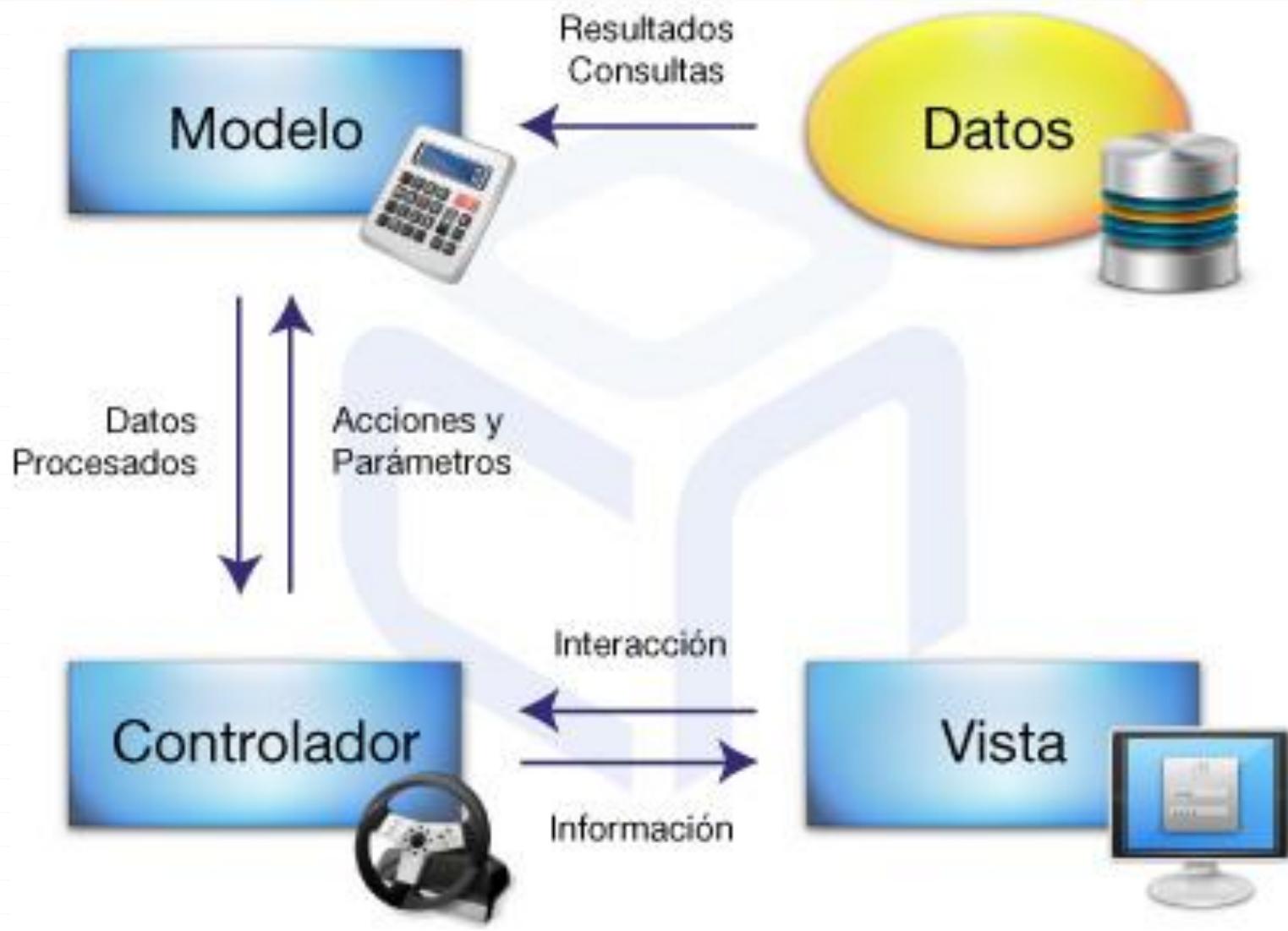
Servidor web

Documentos HTML

La página es enviada al *software* intermedio por el servidor

# Sistema de tres capas: MVC

- Disponen de tres tipos de nodos:
  - Clientes (normalmente navegadores web), que se encargan de las peticiones realizadas por el usuario y de devolver el resultado de estas peticiones
  - Servidores, que procesan los datos (son los encargados de devolver la consecución de las peticiones realizadas por el usuario)
  - Servidores de base de datos, en este caso son los encargados de almacenar la información
- El procedimiento es el siguiente:
  - El usuario interactúa con el cliente.
  - Dicho cliente hace una solicitud al servidor de aplicaciones.
  - Este servidor requiere información de la base de datos de otro servidor.
  - Este, a su vez, le devuelve la petición con el resultado al servidor de aplicaciones y, finalmente, se visualiza en el cliente.
- El sistema de arquitectura en tres capas es conocido en el mundo del desarrollo de software como modelo MVC (Modelo Vista Controlador).
  - Vista. La Capa cliente
  - Controlador. El Servidor de aplicaciones
  - Modelo. El Servidor de base de datos



## ○ Servicios implicados:

- Capa 1: cliente de aplicación. Navegador web.
- Capa 2: servidor de aplicaciones: El principal servidor de aplicaciones usado hoy en día es el servidor Apache
- Capa 3: servidor de datos. Base de datos, servidor SMTP, etc.

## ○ Ventajas :

- Las llamadas realizadas al servidor por parte de la estación de trabajo del cliente son más flexibles, ya que solamente se necesita enviar la petición del cliente a la capa intermedia.
- Con esta arquitectura es posible modificar los parámetros, tanto del servidor de aplicación como del servidor de base de datos, sin que por ello haya que realizar cambios en la capa cliente.
- El código del cliente se mantiene separado de las demás capas. De esta forma, es más fácil realizar el mantenimiento.

## ○ Desventajas:

- Pueden incrementar el tráfico de la red y requieren más balance de carga.
- Los navegadores web no son todos iguales.
- El desarrollar aplicaciones para este tipo de arquitecturas es más difícil, ya que hay que ir probando entre los distintos dispositivos.
- Se puede usar un software específico y de esta forma tener en la misma máquina física funcionando diversos sistemas operativos distintos, incluyendo software de servidor.

# Navegadores web

- Las páginas web están escritas en código HTML. Para que un usuario pueda ver el contenido de estas páginas de manera correcta, necesita de una aplicación que traduzca este código y lo transforme en texto e imágenes.
  - El navegador web
- Como no sabemos con qué navegador va a visualizar un usuario la página web, lo normal es que a la hora de diseñar na página lo hagamos para que el usuario pueda ver la página de manera correcta indistintamente del navegador que esté utilizando.

# Interfaz del navegador

- Existen dos tipos de navegadores:
  - Basados en texto
  - Basados en interfaz de usuarios. Estos últimos son los más habituales hoy en día.
- La interfaz de usuario es el medio a través del cual un usuario va a interactuar con el navegador.
- Algunos de los navegadores más populares hoy en día basados en interfaz de usuario son:
  - Internet Explorer
  - Google Chrome
  - Mozilla Firefox
  - Safari
  - Opera

# Motor de exploración

- Internet está formado por miles de millones de páginas web, de forma que, para encontrar la información que se busca, sería imposible ir a cada una de ellas.
- Es aquí donde entra en juego el **motor de exploración**, que ayuda a encontrar las páginas que sean lo más relevantes posibles a las frases o palabras especificadas.

Motor de exploración = motor de búsqueda

- Entre los más importantes se incluyen
  - Google
  - Yahoo!
  - MSN
  - Bing

# Motor de presentación

- La función básica de los navegadores web es la de ejecutar documentos HTML y mostrarlos por pantalla.
- También permiten guardar información en disco o crear marcadores (bookmarks).
- Cuando se accede a una página web, parte del contenido de esta es almacenado de forma temporal en el ordenador que se está utilizando, sobre todo el contenido multimedia e imágenes.
  - Así, la próxima vez que se accede a esa misma página, en lugar de descargar nuevamente todas las imágenes y el contenido multimedia, este se recupera de forma local, haciendo así que la web cargue de forma mucho más rápida

# Navegadores más populares

- Internet Explorer. Microsoft
- Google Chrome. Google
- Firefox. Mozilla Firefox
- Safari. Apple
- Opera. Opera Software

# Seguridad en los navegadores

- **Navegación privada:** mediante esta opción no se va a dejar ningún rastro de la web que se visita en el ordenador. Es idóneo si se desea acceder a la banca electrónica, o incluso si el usuario se encuentra en un equipo consultando el correo electrónico en un cibercafé.
- **Integración con antivirus:** la tendencia actual es que exista una simbiosis entre el navegador y el antivirus, de tal forma que este último tenga la capacidad de detectar posibles amenazas que podrían atacar el equipo.
- **Actualizaciones:** Deberemos mantener actualizado el navegador con la última versión del mismo para evitar posibles vulnerabilidades
- **Identificación del sitio web:** al acceder a sitios web seguros, por regla general el navegador mostrará información de la web a la que se está accediendo, confirmando que es un sitio con contenido legítimo.
- **Limpiar el historial reciente:**
- **Plugins:** Mantener actualizados todos los plugins con los que cuente el navegador

# Plugins

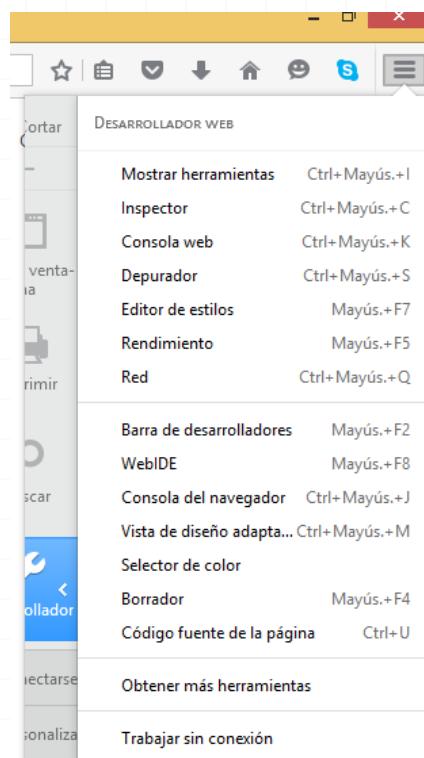
- Los plugins son pequeñas aplicaciones que, una vez añadidos, complementan la funcionalidad de los navegadores.
  - Por ejemplo, los plugins son los encargados de mostrar el contenido multimedia, como pueden ser las animaciones en flash, o vídeos en cualquier formato.
- La mayoría de los plugins son desarrollados de forma externa al propio navegador. Muchos de ellos son gratuitos, mientras que otros son de pago.

WebDeveloper

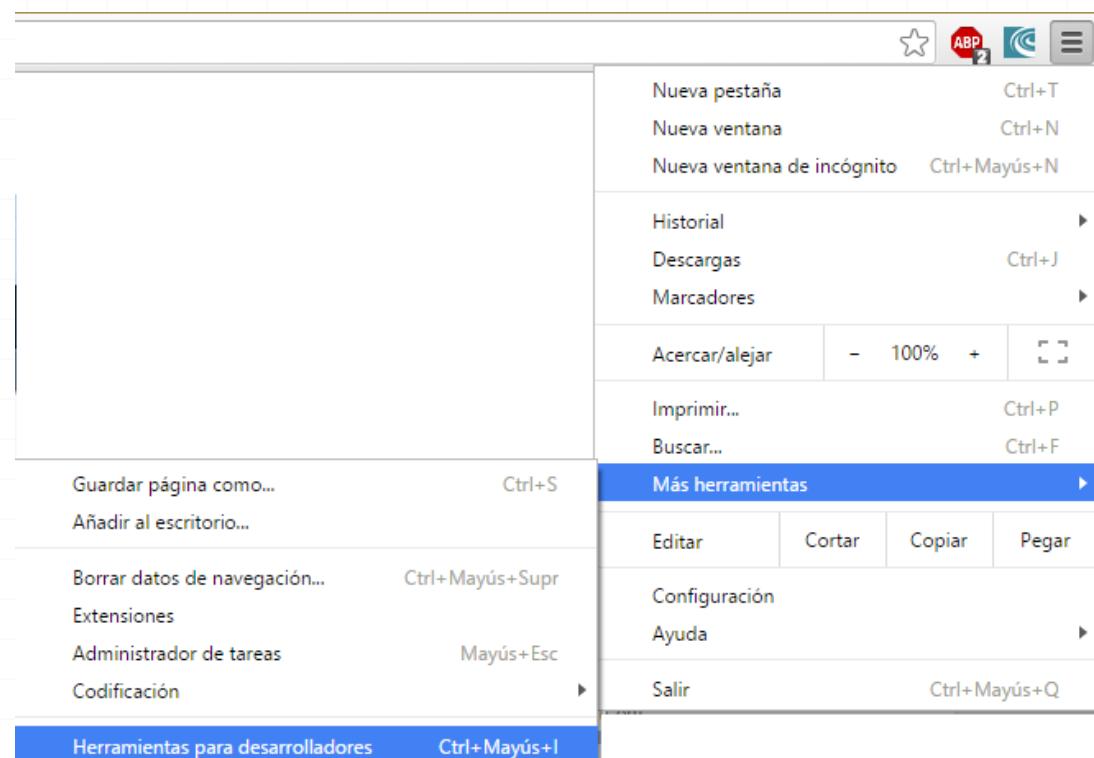
# Herramientas para desarrollador

- Casi todos los navegadores hoy en día tienen una serie de complementos y herramientas que les ayudan en la comprobación de páginas web.
- Son las herramientas para desarrollador.

Firefox



Google Chrome



## ○ Pestaña Elements

- Desde esta pestaña del inspector de propiedades podemos ver el código HTML, los estilos CSS que se están aplicando.
- También podemos hacer modificaciones para ver en tiempo real cómo afectarían determinados cambios a la página

## ○ Pestaña Consola

- Desde esta pestaña se pueden ejecutar instrucciones de JavaScript y ver cómo afectarán a determinados parámetros

## ○ Pestaña Sources

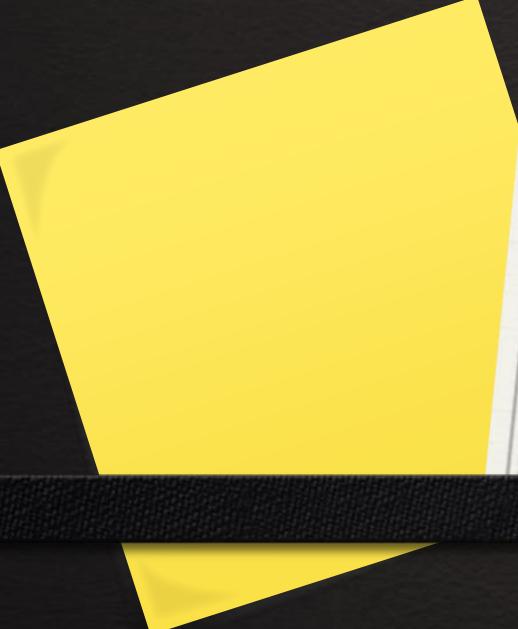
- Se pueden ver las fuentes de la página

## ○ Pestaña red

- Desde esta pestaña se puede ver el tráfico de red que genera la página

## ○ Pestaña Resources

- Desde esta pestaña se puede ver diferente tipo de información, como por ejemplo las cookies que tenga el sitio



Conformidad con  
los estándares

- El que una web cumpla con los estándares puede significar que facilite a los motores de búsqueda la evaluación y acceso a la información de los documentos que contiene, siendo estos incluidos en los buscadores y dotándolos de una mayor visibilidad.
- Si se desea que el sitio cumpla con los estándares web, existen herramientas gratuitas que brindan la posibilidad de verificar la página mediante un servicio de validación.
  - Estos servicios analizan los documentos, y presentan a modo de lista los errores que estos puedan contener, facilitando la tarea de encontrar y corregir dichos errores, y ahorrando de esta forma mucho tiempo.
- El cumplimiento de los estándares es importante, no solo para que el sitio sea mejor indexado por los motores de búsqueda, sino también para permitir que las páginas sean entendidas por usuarios con navegadores distintos a los usuales, ya se encuentren accediendo desde un ordenador, un smartphone o una pantalla de televisión, sin olvidar los navegadores de voz, que leen páginas web en voz alta a personas con dificultades visuales, los navegadores braille, etc.

- Para esta tarea, es posible utilizar una herramienta gratuita proporcionada por la W3C.

<https://validator.w3.org/>

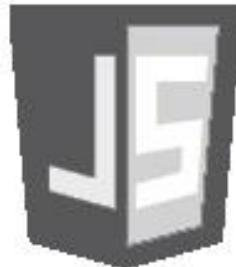
- Esta herramienta consta de tres pestañas distintas:
  - **Validate by URI:** aquí se añade el URL de la web que se deseé analizar, muy útil para aquellas páginas que ya estén en Internet.
  - **Validate by File Upload:** es posible subir un archivo HTML, lo que será útil para aquellas webs que todavía no hayan sido publicadas en Internet.
  - **Validate by Direct Input:** permite añadir directamente el código a validar, copiando y pegando.
- En caso de que la página analizada contenga errores, la herramienta mostrará un listado y sugerencias de cómo resolverlos

# Introducción a JavaScript



JAVA *is to*  
JAVASCRIPT *as*

HAM *is to*  
HAMSTER



# ¿Qué podemos hacer con JS?

- Cambiar el contenido HTML ([Ejemplo1.html](#))
- Cambiar los atributos HTML ([Ejemplo2.html](#))
- Cambiar los estilos CSS ([Ejemplo3.html](#))
- Validar el valor introducido en un campo ([Ejemplo4.html](#))
- También podemos crear códigos para insertar diferentes elementos como:
  - Reloj
  - Contadores de visitas
  - Fechas
  - Calculadoras
  - Validadores de formularios
  - Detectores de navegadores e idiomas
  - Etc.

Cómo incluir  
JavaScript en  
documentos  
HTML

# En los elementos HTML

- Consiste en introducir trozos de JS dentro del código HTML de la página

```
<body>
  <h1>Mi primer script</h1>
  <p onclick="alert('Hola Mundo')">Párrafo de texto.</p>
</body>
```

- Es el método menos utilizado.
- Su uso se suele limitar a definir algunos eventos



# Manos a la obra

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>JS en línea</title>
  </head>
  <body>
    <p onclick="alert('Un mensaje de prueba')">
      Haz click aquí.
    </p>
  </body>
</html>
```

# En el mismo documento

- En este caso, el código puede ir en el head o en el body

En el body	En el head
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt;...&lt;/head&gt; &lt;body&gt;   &lt;script&gt;     alert("Mensaje de prueba");   &lt;/script&gt;   &lt;h1&gt;Mi página web&lt;/h1&gt;   &lt;p&gt;Un párrafo&lt;/p&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt;   &lt;script&gt;     alert("Mensaje de prueba");   &lt;/script&gt; &lt;/head&gt; &lt;body&gt;   &lt;h1&gt;Mi página web&lt;/h1&gt;   &lt;p&gt;Un párrafo&lt;/p&gt; &lt;/body&gt; &lt;/html&gt;</pre>

- Anteriormente, para definir el script se utilizaba la sentencia:

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

- Realmente, no se requiere el atributo type. JavaScript es el lenguaje de scripts por defecto en HTML.

# En un archivo externo

- Crear un archivo con la extensión .js (en este ejemplo, lo llamaremos código):

```
alert("Un mensaje de prueba2");
```

- Archivo html

```
<!DOCTYPE html>
<html>
  <head>
    ...
    <script src="codigo.js"></script>
  </head>
  <body>
    <h1>Mi primer Script</h1>
    <p>Un párrafo de texto</h1>
  </body>
</html>
```

- Se puede llamar a un script externo desde la cabecera o desde el body

# Sintaxis JavaScript

# Definiciones

○ **Programa.** Lista de "instrucciones" a ser "ejecutadas" por el ordenador.

○ En JS los programas se ejecutan en el navegador web.

○ **Script.** Cada uno de los programas, aplicaciones o trozos de código creados con el lenguaje de programación JavaScript.

○ Es decir, un programa en JavaScript

○ **Sentencia.** Cada una de las instrucciones que componen el programa.

○ Las sentencias de JavaScript se escriben separadas por ; (punto y coma)

Sentencia 1 → `var z = 4;`

`document.write(z);` ← Sentencia 2

# Tipos de datos

- Un dato es una representación simbólica (numérica, alfabética, etc.) de un atributo o variable cuantitativa o cualitativa.
- A la hora de programar, vamos a necesitar utilizar datos y esos datos van a tener un tipo.
  - Vida real:
    - Nombre de una persona: texto
    - Edad de la persona: valor numérico
  - En JavaScript los datos se dividen en tres tipos básicos:
    - Números. Son números.
    - Strings. Son valores de texto.
    - Booleans. Solamente contempla dos alternativas: true o false.

# Valores JavaScript

- La sintaxis de JavaScript define dos tipos de valores:
  - Valores fijos. Denominados **literales**
    - Números. Se escriben con o sin decimales (10 ó 10.05)
    - Strings. Son cadenas de texto. Se pueden escribir entre comillas simples (' ') o comillas dobles (" ")
  - Valores de variables. Denominados **variables**
    - En programación, las variables se utilizan para almacenar un valor que puede ir cambiando a lo largo de la ejecución del programa
    - Para definir una variable en JavaScript se utiliza la variable var y para asignarle un valor se utiliza el signo =

```
var x;
```

```
x = 6;
```

# Variables

# Variable

- Una variable guarda valores para su uso posterior.
- Definir una variable. Las variables se crean con la sentencia de definición de variables
  - La sentencia comienza con la palabra reservada var
  - A continuación vienen una o más definiciones de variables separadas por comas
- Inicializar una variable. A una variable se le puede asignar un valor inicial usando el operador de asignación =

```
var x = 5; //Define la variable x y la inicializa con el valor 5  
var y; //Define la variable pero no le asigna ningún valor  
var z=1, t="Hola"; //Define las variables z y t con  
//los valores 1 y "Hola" respectivamente
```

- Las variables sin ningún valor asignado contienen el valor **undefined**
  - Undefined. Valor (y tipo) especial de JavaScript que significa indefinido (en el ejemplo anterior la variable y).

# Nombres de variables

- El nombre (o identificador) de una variable debe comenzar por una letra, guion bajo (\_) o signo de dólar (\$). Además puede contener números. No puede contener espacios en blanco
  - Nombres bien construidos: x, ya\_vás, \$A1, \$, \_43dias
    - Aunque ya\_vás es correcto no conviene utilizar acentos en los nombres de variables.
  - Nombres mal construidos: 1A, 123, %3, v=7, a?b, ..
- Los nombres de variables son Case Sensitive: mi\_var y Mi\_var son variables diferentes
- Un nombre de variable no debe ser una palabra reservada de JavaScript (palabras utilizadas por JS para construir las sentencias)
  - break, case, catch, continue, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with
- Se recomienda que sean descriptivos

# Expresiones con variables

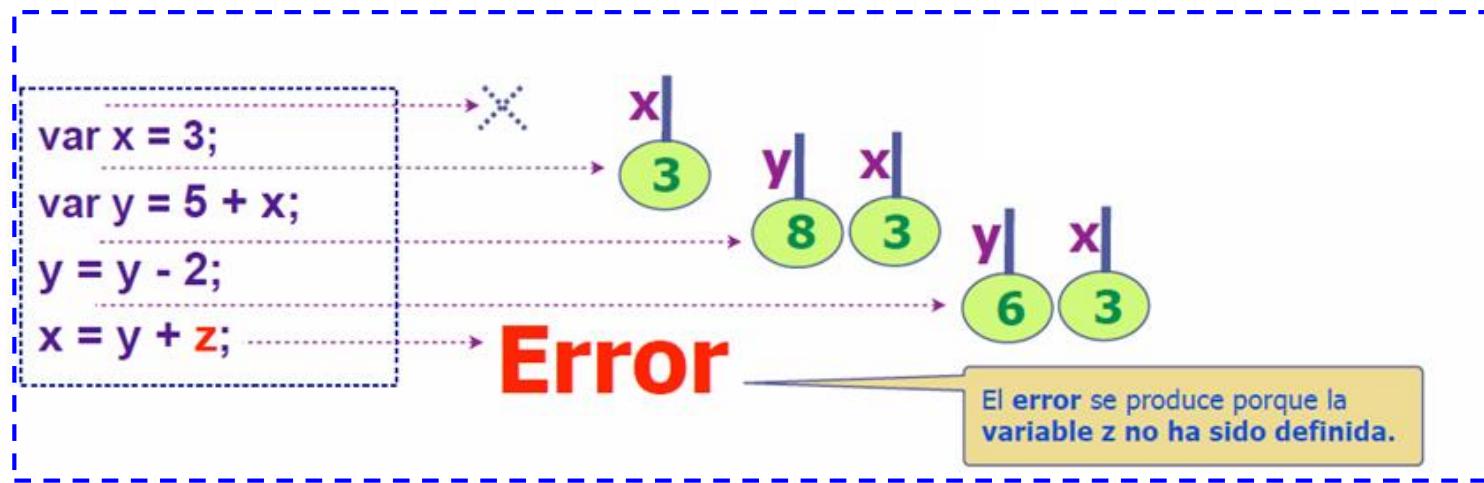
- Una variable se puede utilizar en expresiones como cualquier otro valor

$$y=5+x$$

- Una variable puede utilizarse en la expresión que se asigna a ella misma:

$$y=y+2$$

- Usar una variable no definida en una expresión provoca un error y la ejecución del programa se interrumpe



Texto en JS

# Texto: strings

- El texto escrito se representa en JavaScript con strings
- Un string delimita el texto con **comillas** o **apóstrofes**, por ej.
  - Frases: "**hola, que tal**" o '**hola, que tal**'
  - String vacío: "" o "
- Ejemplo de "**texto 'entrecomillado'** "
  - comillas y apóstrofes se pueden anidar
  - '**entrecomillado**' forma parte del texto
- Operador + concatena strings, por ejemplo
  - "H~~o~~**la**" + " " + "Pepe" => "**H~~o~~la Pepe**"
- La **propiedad length** de un string indica su longitud (Número de caracteres)
  - "Pepe".length => 4
  - "H~~o~~**la Pepe**".length => 9

# Texto complejo

- Ya hemos visto que si el propio texto contiene comillas simples o dobles, la estrategia que se sigue es la de encerrar el texto con las comillas (simples o dobles) que no utilice el texto:

```
var texto1 = "Una frase con 'comillas simples' dentro";  
var texto2 = 'Una frase con "comillas dobles" dentro';
```

- Sin embargo a veces esto no es tan sencillo. Por ejemplo una cadena de caracteres que tenga que incluir comillas dobles (" ") y comillas simples (' '). Además, existen otros caracteres que son difíciles de incluir en una variable de texto (tabulador, ENTER, etc.)
- Para resolver estos problemas, JavaScript define un mecanismo para incluir de forma sencilla caracteres especiales y problemáticos dentro de una cadena de texto.

Si se quiere incluir...	Se debe usar...
Una nueva línea	\n
Un tabulador	\t
Una comilla simple	\'
Una comilla doble	\"
Una barra inclinada	\\"

# Comentarios

- Los comentarios sólo tienen valor informativo.
  - Explican partes de código para ayudar a entender un programa.
- JavaScript permite utilizar:
  - Comentarios multilínea: delimitado por /\* ... \*/
  - Comentario de una línea: Empieza con // y acaba al final de la línea

```
/*Comentario en  
Varias Líneas*/  
var z = 4; //Definir de la variable z  
document.write(z); //Mostrar la variable
```

# Operadores

# Expresiones numéricas y operadores

- Las variables por sí solas son de poca utilidad.
- Para poder construir expresiones que permitan realizar cálculos complejos, necesitaremos de los operadores.
  - Los operadores permiten manipular el valor de las variables, realizar operaciones matemáticas con sus valores y comparar diferentes variables
- JavaScript incluye diferentes tipos de operadores

# Operadores aritméticos

- Permiten realizar manipulaciones matemáticas sobre el valor de las variables numéricas
- Los operadores aritméticos son los siguientes:
  - + (suma)
  - - (resta)
  - \* (multiplicación)
  - / (división)
  - % (módulo). Este operador devuelve el resto de la división entera de dos números (Ej.:  $9 \% 5 = 4$ )
- Cuando en una misma expresión, tenemos varios operadores, el orden de las operaciones se llevará a cabo teniendo en cuenta la prioridad de los operadores
  - $6 + 3 * 4 = 18$
  - $(6 + 3) * 4 = 36$

# Operadores aritméticos

0 Expresiones mal construidas dan error o llevan el interprete a un estado inconsistente

**13 + 7**      =>    20      // Suma de números

**13 - 1.5**      =>    11.5      // Resta de números

// OJO! la coma española es un punto en la sintaxis inglesa

**(8\*2 - 4)/3**      =>    4      // Expresión con paréntesis

**8 / \* 3**      =>    ??      // Expresión incorrecta

**8 3**      =>    ??      // Expresión incorrecta

# Operadores de asignación

- Este operador se utiliza para guardar un valor específico en una variable.
- El símbolo utilizado es =
- A la izquierda del operador, siempre debe indicarse el nombre de una variable. A la derecha del operador, se pueden indicar variables, valores, condiciones lógicas, etc.

```
var nombre = "Marta"
```

```
var precio = 5
```

```
x = y
```

# Otros operadores de asignación

Operador	Ejemplo	Significado
<code>+=</code>	<code>x += y</code>	$x = x + y$
<code>-=</code>	<code>x -= y</code>	$x = x - y$
<code>*=</code>	<code>x *= y</code>	$x = x * y$
<code>/=</code>	<code>x /= y</code>	$x = x / y$
<code>%=</code>	<code>x %= y</code>	$x = x \% y$

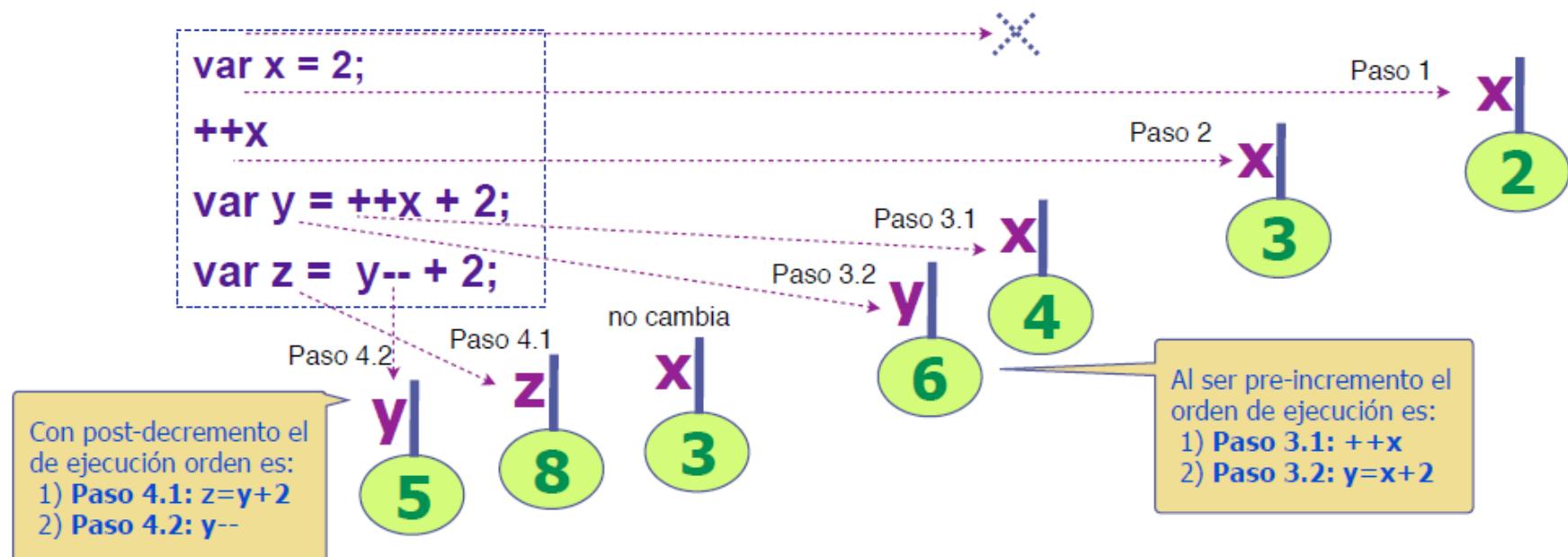
# Incremento (++) y decremento (--)

- Estos dos operadores solamente son válidos para las variables numéricas
- Se utilizan para incrementar o decrementar en una unidad el valor de una variable
- Ambos operadores se pueden utilizar como prefijo o sufijo

++	--
<pre>var numero = 5; numero++; //numero=6</pre>	<pre>var numero = 5; numero--; //numero=4</pre>
<pre>var numero = 5; ++numero; //numero=6</pre>	<pre>var numero = 5; --numero; //numero=4</pre>

# Pre y post autoincremento o decremento

- Tanto el operador autoincremento, como el operador autodecremento, se pueden aplicar por la derecha o por la izquierda a las variables de una expresión
  - Si se aplica por la izquierda a la variable (pre), el incremento/decremento se realiza antes de evaluar la expresión
  - Si se aplica por la derecha (post) se incrementa/decrementa después de evaluarla.



# Operadores de comparación

- Comparan sus operandos y devuelve true o false según la comparación sea verdadera o falsa.
- Los operandos pueden ser numéricos, cadena de caracteres, valores lógicos, etc.
- Si los operandos que se quieren comparar no son del mismo tipo, JavaScript intenta convertirlos en el tipo apropiado para permitir la comparación.
  - Las únicas excepciones que tiene esta conversión son los operadores === y !== que no intentarán convertir los operandos a tipo compatible antes de comprobar su igualdad.

# Operadores de comparación

`var var1 = 3, var2 = 4;`

Operador	Descripción	True
<code>==</code> Igualdad	Devuelve true si ambos operandos son iguales.	<code>3 == var1</code> <code>"3" == var1</code> <code>3 == '3'</code>
<code>!=</code> Desigualdad	Devuelve true si ambos operandos no son iguales.	<code>var1 != 4</code> <code>var2 != "3"</code>
<code>====</code> Estrictamente igual	Devuelve true si los operandos son igual y tienen el mismo tipo.	<code>3 === var1</code>
<code>!==</code> Estrictamente desigual	Devuelve true si los operandos no son igual y/o no tienen el mismo tipo.	<code>var1 !== "3"</code> <code>3 !== '3'</code>
<code>&gt;</code> Mayor que	Devuelve true si el operando de la izquierda es mas grande que el operando de la derecha.	<code>var2 &gt; var1</code> <code>"12" &gt; 2</code>
<code>&gt;=</code> Mayor o igual que ( <code>&gt;=</code> )	Devuelve true si el operando de la izquierda es mas grande que el operando de la derecha.	<code>var2 &gt;= var1</code> <code>var1 &gt;= 3</code>
<code>&lt;</code> Menor que	Devuelve true si el operando de la izquierda es mas pequeño que el operando de la derecha.	<code>var1 &lt; var2</code> <code>"2" &lt; "12"</code>
<code>&lt;=</code> Menor o igual que	Devuelve true si el operando de la izquierda es mas pequeño o igual que el operando de la derecha.	<code>var1 &lt;= var</code>

# Consideraciones sobre operadores

# Sobrecarga de operadores

- Algunos operadores tienen varias semánticas (“*varios usos*”) dependiendo del contexto en el que se usan
- Por ejemplo, el operador + tiene tres semánticas diferentes:
  - Suma de enteros (operador binario)
  - Signo de un número (operador unitario)
  - Concatenación de strings (operador binario)

13 + 7

=> 20

// Suma de números



+13

=> 13

// Signo de un número

"Hola " + "Pepe" => "Hola Pepe" // Concatenación de strings



- Estos operadores se dice que están sobrecargados

# Conversión de tipos en expresiones

- JavaScript realiza conversión automática de tipos
  - cuando hay ambigüedad en una expresión utiliza las reglas de prioridad para resolverla
- La expresión "13" + 7 es ambigua
  - porque combina un string con un number
  - Con ambigüedad JavaScript da prioridad al operador + de strings, convirtiendo 7 a string
- La expresión +"13" también necesita conversión automática de tipos
  - El operador + solo esta definido para number (no hay ambigüedad)
  - JavaScript debe convertir el string "13" a number antes de aplicar operador +

The screenshot shows a browser's developer tools console with the title '<top frame>'. It displays the execution stack for the expression '13' + 7. The stack consists of several frames, each containing a line starting with either a right-pointing arrow (>) or a left-pointing arrow (<). The stack shows the progression from the original expression to its evaluated result:

```
> 13 + 7  
< 20  
> "13" + "7"  
< "137"  
> "13" + 7  
< "137"  
> +"13" + 7  
< 20  
>
```

# Prioridad de los operadores

- Cuando una expresión contiene varios operadores, ¿en qué orden se ejecutan las operaciones?
  - Por el orden de prioridad de los operadores
- Algunas prioridades:
  - ()
  - ++
  - --
  - \*
  - /
  - +
  - -
- Puedes ver la lista completa en  
[http://www.w3schools.com/js/js\\_arithmetic.asp](http://www.w3schools.com/js/js_arithmetic.asp)



La consola del  
navegador

# ¿En qué consiste la consola?

- Casi todos los navegadores modernos traen una herramienta denominada Consola.
- Desde la consola, se nos informará si se producen errores durante la ejecución de nuestros scripts,incluso nos suele indicar en qué linea se produce el error.
- Para activar la consola en Firefox o Chrome pulsaremos F12.

# Expresiones con variables

- Indicar el resultado de evaluar la siguiente expresión JS: 33 - 10
- 2
- 4
- 5
- "23"
- 33
-   23
- 1023
- ErrorDeEjecución

Indicar el resultado de evaluar la siguiente expresión JS:  $4*4+7$

2

4

5

"23"

33



23

1023

ErrorDeEjecución

Indicar el resultado de evaluar la siguientes expresión JavaScript:  
`2+"3"`

2

4

5

23

33



"23"

"1023"

ErrorDeEjecución

0 Indicar el resultado de evaluar la siguiente expresión JS:  
+"2"+3

0 2

0 4

 5

0 "23"

0 33

0 23

0 1023

0 ErrorDeEjecución

0 Al ejecutar un programa cuyas 2 primeras sentencias son var x = 2; var y = "2"; Si la tercera sentencia es la indicada al final, indicar para cada una el contenido de la variable x después de ejecutarla, o si ocurrirá ErrorDeEjecución: var %t = 3;

0 1

0 2

0 3

0 4

0 23

0 "2"

0 "23"



ErrorDeEjecución

○ Al ejecutar un programa cuyas 2 primeras sentencias son var x = 2; var y = "2"; Si la tercera sentencia es la indicada al final, indicar para cada una el contenido de la variable x después de ejecutarla, o si ocurrirá ErrorDeEjecución: x = "2";

○ 1

○ 2

○ 3

○ 4

○ 23

○ "2" 

○ "23"

○ ErrorDeEjecución

○ Al ejecutar un programa cuyas 2 primeras sentencias son var x = 2; var y = "2"; Si la tercera sentencia es la indicada al final, indicar para cada una el contenido de la variable x después de ejecutarla, o si ocurrirá ErrorDeEjecución: x = y;

○ 1

○ 2

○ 3

○ 4

○ 23

○ "2" 

○ "23"

○ ErrorDeEjecución

○ Al ejecutar un programa cuyas 2 primeras sentencias son: var x = 2; var y = "2"; Si la tercera sentencia es la indicada al final, indicar para cada una el contenido de la variable x después de ejecutarla, o si ocurrirá ErrorDeEjecución: y = 8;

○ 1



○ 3

○ 4

○ 23

○ "2"

○ "23"

○ ErrorDeEjecución

# Salidas de JavaScript

# Salida JS

- Son aquellas funciones que nos permiten mostrar información que tenemos almacenada en nuestro programa al usuario.
- JavaScript nos ofrece varias formas para poder mostrar los datos:
  - Escribiendo dentro de un cuadro de alerta, utilizando `window.alert()`
  - Escribiendo en la salida HTML usando `document.write()`
  - Escribiendo dentro de un elemento HTML, usando `innerHTML`



# Manos a la obra

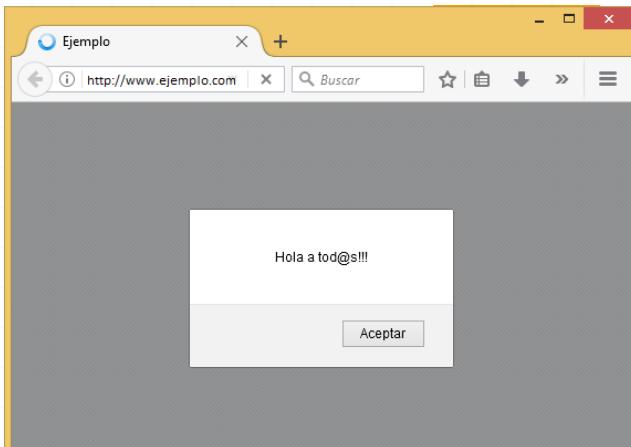
Ejemplo 1

window.alert(): mostrará un cuadro de alerta para mostrar el dato

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <script>
    window.alert("Hola a tod@s !!!");
  </script>
</head>
<body>
  <h1>Mi primera página con JS</h1>
  <p>Bienvenidos al curso de JavaScript</p>
  
</body>
</html>
```

## Ejemplo 1

- Aunque no lo parezca, existen diferencias entre colocar el script en el head o en el body.
  - El navegador interpreta el código HTML de arriba abajo.
  - Si ponemos el script en el head, lo ejecutará antes de cargar el contenido del body, por lo que en este ejemplo no mostrará el contenido de la página hasta que se acepte el mensaje del alert.



Una vez aceptado el mensaje, se cargará el contenido de la página

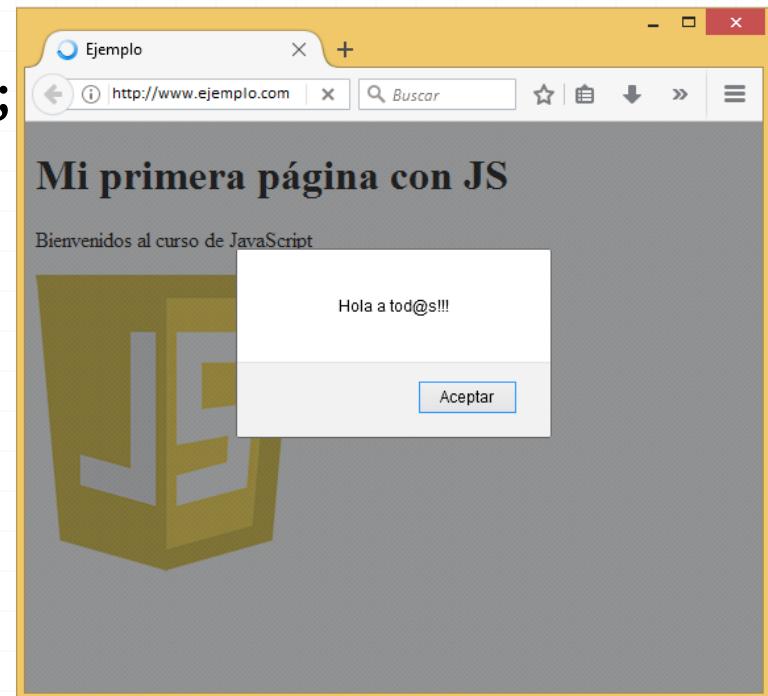


La ventana alert es una ventana modal, es decir, se abre en primer plano y no deja manipular nada en segundo plano y tampoco se pueden minimizar. Además detiene el flujo de ejecución del programa hasta que se cierra.

## Ejemplo 1

Cambiar el script al body, después de la etiqueta img

```
<!DOCTYPE html>
<html lang="es">
<head> ... </head>
<body>
    ...
<script>
    window.alert(Hola a tod@s!!!);
</script>
</body>
</html>
```





# Manos a la obra

document.write()

Ejemplo 2

```
<!DOCTYPE html>
<html lang="es">
<head>
...
<script>
    document.write("Hola a tod@s");
</script>
</head>

<body>
    <h1>Mi primera página con JS</h1>
    <p>Bienvenidos al curso de JS</p>
    
</body>
</html>
```



*Entrada de datos*

# Función prompt

- En el ejemplo anterior, hemos introducido el valor en pesetas que queríamos convertir a euros directamente en el script.
- Cada vez que queramos convertir otro importe, tendremos que modificar el script para poner el nuevo importe.
- Lo ideal sería que cuando abrimos la página, nos solicite el importe a convertir.
- Esto lo podremos conseguir mediante la función de entrada de datos prompt.

0 Crear una página web que pida el nombre y apellidos del alumno y luego muestre un saludos personalizado.



ARRAYS

# Concepto de Array (Arreglos o Matrices)

Un **array** es un tipo de variable especial en el que podemos introducir varios valores



Artículo 1: Camiseta azul  
Artículo 2: Pantalón vaquero  
Artículo 3: Camisa manga larga  
Artículo 4: Playeros reebok  
Artículo 5: Calcetines Adidas



```
var camiseta;  
var pantalon;  
var camisa;  
var playeros;  
var calcetines;
```

Camiseta azul    Pantalón vaquero    Camisa m/l    Playeros reebok    Calcetín Adidas

```
var articulos = ["Camiseta azul", "Pantalón vaquero", "Camisa m/l", "Playeros Reebok", "Calcetines Adidas"];
```

# Concepto de Array (Arreglos o Matrices)

```
BMW var coche1 = "BMW";
```

```
SEAT var coche2 = "SEAT";
```

```
RENAULT var coche3 = "RENAULT";
```



```
var coches=["BMW", "SEAT", "RENAULT"];
```

```
Luis García
```

```
C/ Uría 14, 2º C
```

```
24
```

```
24/02/2017
```

```
var persona = ["Luis García", "C/ Uría 14, 2º C", 24, 24/02/2017];
```

```
Array vacío
```

```
var frutas = [];
```

# Definir un array

○ Hay dos formas de declarar un array en JS

○ Definir una variable con un literal de tipo matriz:

```
var array-name = [item1, item2, ...];
```

○ Utilizar la sentencia new Array:

```
var array-name = new Array(item1, item2, ...);
```

```
var persona=["Luis García", "C/ Uría 27", 24, 24/05/2017];
```

```
var persona=new Array("Luis García","C/ Uría 27",24,24/05/2017);
```



*No poner coma después del último elemento del array, puede causar fallos en los navegadores*

*Las dos sentencias vistas anteriormente hacen lo mismo por lo que por simplicidad se aconseja utilizar el primero de los métodos.*

# ACCEDER A LOS ELEMENTOS DE UN ARRAY

## 0 Índice o posición



[0] [1] [2]



*El primer elemento de un array es el [0]*

*Un array de 3 elementos, el índice que identifica al último elemento es el [2]*

## 0 Acceder a un elemento del array

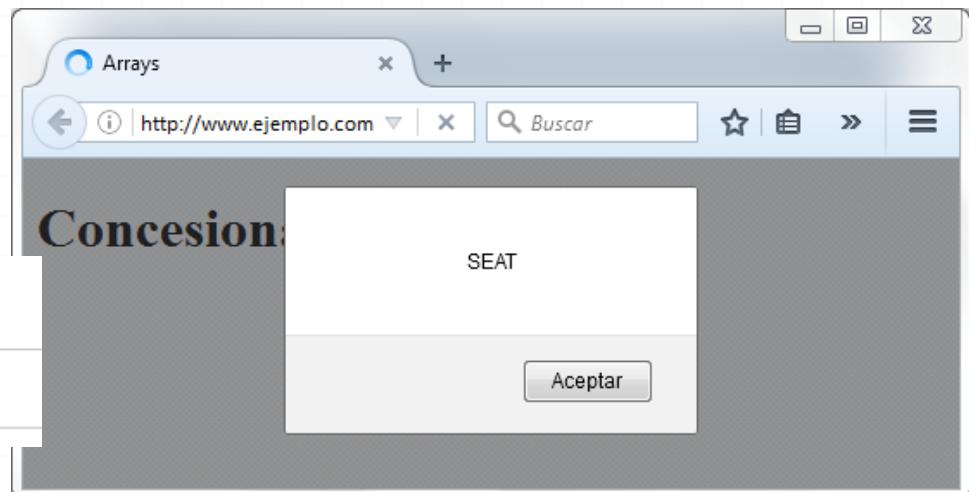
`coches[0] => "BMW"`

`coches[1] => "SEAT"`

## 0 `alert(coches[1]);` => Mostraría la palabra SEAT en una ventana emergente.

This statement modifies the first element in cars:

```
cars[0] = "Opel";
```



# Propiedades y métodos de los arrays

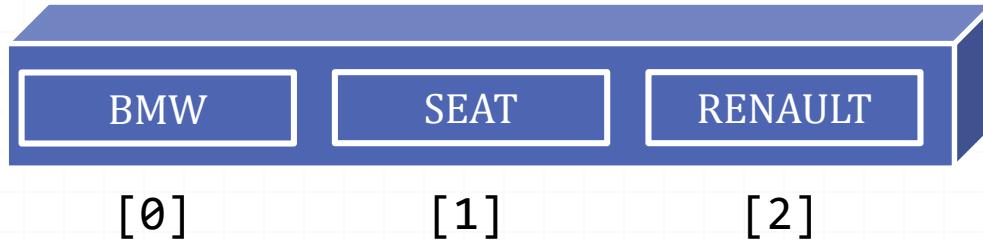
Propiedades	Métodos
length	push
	unshift
	pop
	shift
	delete
	splice
	slice
	sort

# Propiedad *Length*

*array-name.length*

- Devuelve el número de elementos de un array.

coches



`coches.length = 3`

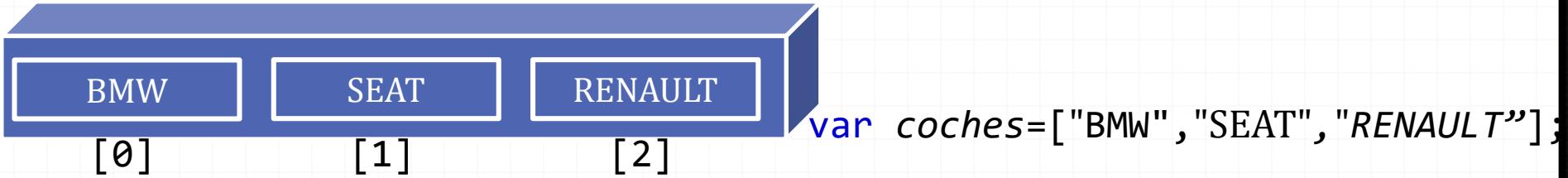
# Agregar elementos a un array

- o **push**. Agrega uno o varios elemento al final del array

*array-name.push(item1, item2, ..., item\_n);*

- o **unshift**. Agrega un elemento al principio del array

*array-name.unshift(item1, item2, ..., item\_n);*



```
coches.push("Citroen"); //coches=["BMW", "SEAT", "RENAULT", "Citroen"];
coches.unshift("Citroen"); //coches=["Citroen", "BMW", "SEAT", "RENAULT"];

coches.push("Citroen", "Mercedes");
```

# Borrar elementos de un array

o **pop**. Elimina el último elemento de un array

*array-name.pop();*

o **shift**. Elimina el primer elemento del array y "desplaza" (shift) todos los demás elementos a un índice inferior.



```
coches.pop(); //coches=["BMW","SEAT"];
```

```
coches.shift(); //coches=["SEAT","RENAULT"];
```

# *delete*

○ Elimina un elemento de un array.

○ Sintaxis:

```
delete nombrearray[num_elemento]
```

```
var coches = ["BMW", "Seat", "Renault"];
delete coches[1]; //Cambia el elemento que se
                  encuentra en la posición 1 de
                  coches por el valor undefined
```

*El uso de delete puede dejar agujeros indefinidos en la matriz. Utilice pop () o shift () en su lugar.*



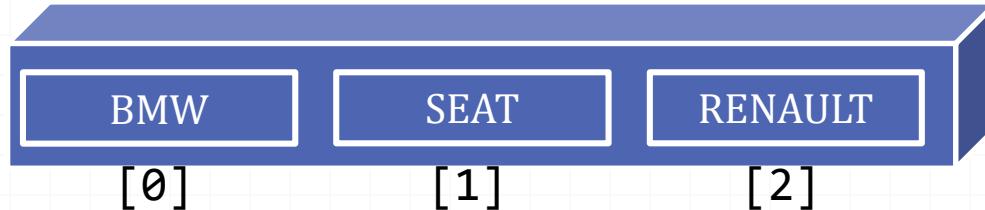


# Manos a la obra

Arrays\_01

- Definir el array coches con tres valores: BMW, Seat y Renault.
- Crear un script que muestre:
  - El primer elemento del array.
  - El número total de elementos que tiene el array
  - Insertar al final del array coches el valor citroen y mostrar el nuevo tamaño con el que queda el array.
  - Añadir el coche Mercedes y mostrar el total de elementos del array en una sola sentencia y sin utilizar la propiedad length
  - Repetir los dos puntos anteriores para añadir la marca Peugeot y Audi pero al principio del documento.
  - Mostrar el primer elemento del array, borrarlo y repetir la sentencia que muestra el primer elemento del array
  - Mostrar el último elemento del array, borrarlo y repetir la sentencia que muestra el último elemento del array

# Acceder a ultimo elemento de un array



```
document.write("el ultimo elemento es " + coches[2]);
```



*EL array tiene 3 elementos, pero recordemos que el índice comienza a contar en el 0 por lo que la ultima posición es la 2. si utilizáramos coches[3] como la posición 3 en el array no existe, JS nos emitiría un mensaje de error.*

```
Document.write("el último elemento es" + coches[coches.length]);
```

```
Document.write("el último elemento es" + coches[coches.length-1]);
```



## Manos a la obra

Modificar el html del ejercicio anterior para que todos los puntos que muestran el último elemento lo hagan utilizando la propiedad `length`.

# Más operaciones con arrays

- Si queremos mostrar el contenido del array completo, usaremos el nombre del array.

`document.write(coches)` imprime todo el array coches

- Podemos utilizar, por ejemplo, una ventana prompt para solicitar los valores del array al usuario.

`coches.push(prompt("Introduce el coche que quieras añadir"));`



## Manos a la obra

Inicializar el array coches con los valores: BMW, Seat y Renault.

Pedir al usuario un nuevo coche a añadir

Una vez que el usuario haya añadido un nuevo coche, se muestre por pantalla el contenido íntegro del array.

# *splice ()*

- Permite insertar o eliminar elementos dentro de un array incluso de forma simultánea.
- Sintaxis:

```
array_name.splice(pos_inicial, numelemaborrar,  
[item1[, item2[, . . . [, itemN]]]])
```

```
var coches = ["BMW", "Seat", "Renault"];  
coches.splice(2,0,"Citroen", "Peugeot");  
//coches=["BMW", "Seat", "Citroen", "Peugeot", "Renault"]  
coches.splice(1,2); //coches=["BMW", "Peugeot", "Renault"]
```

# *concat()*

0 Devuelve un array que contiene la concatenación de dos o más arrays.

0 Sintaxis:

nombrearray1. concat([array2[, array2[ , . . . [, arrayN]]]])

```
var coches1 = ["BMW", "Seat", "Renault"];
var coches2 = ["Mercedes", "Peugeot"];
var coches = coches1.concat(coches2);
//coches=["BMW", "Seat", "Renault", "Mercedes", "Peugeot"]
```

# *slice()*

- Crea un nuevo array con los elementos seleccionados en un array existente.
- Sintaxis:

nombrearray.slice(numelemínicial [,numelemfinal]);

```
var coches = ["BMW", "Seat", "Renault", "Peugeot",
"Mercedes"];
var coches2= coches.slice(1);
//Crea un array llamado coches2 con el elemento Seat
var coches3= coches.slice(1,3);
//Crea un array llamado coches3 con los elementos que se
encuentran en la posición desde la 1 hasta la 3 (pero no
incluye la 3) coches3=["Seat", "Renault"]
```

# Ordenar un array

- o `sort()`. Ordena un array alfabéticamente

```
nombrearray.sort();
```

- o `reverse()`. Invierte los elementos de un array

```
var coches = ["BMW", "Seat", "Renault", "Peugeot"];
coches.sort(); //coches=["BMW", "Peugeot", "Renault", "Seat"]
coches.reverse(); //invierte el orden de los elementos del array
```



*La función `sort()` trata los valores del array como textos. Debido a esto, `sort()` producirá resultados incorrectos al ordenar números*



# Manos a la obra

Arrays\_04

- 0 Definir tres arrays:
  - 0 coches1 = ["BMW", "Seat", "Renault"]
  - 0 coches2 = ["Mercedes", "Peugeot"]
  - 0 coches3 = ["Citroen"]
- 0 Concatenar los tres arrays anteriores en un array al que llamaremos coches.
- 0 Insertar dos coches nuevos (Audi y Fiat) en el array coches en la posición 3 y 4 (cuidado con los números de índice)
- 0 Mostrar el contenido del array coches después de realizar la operación anterior.
- 0 Ordenar el array coches alfabéticamente y luego mostrar el array ordenado.

# PROGRAMACIÓN ORIENTADA A OBJETOS CON JS

# POO (OOP, en inglés)

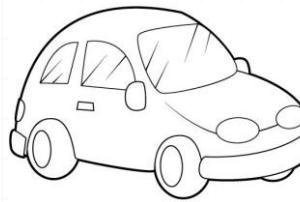
- POO = Programación Orientada a Objetos
- Hay dos grandes grupos de lenguajes de programación:
  - Lenguajes de programación orientados a procedimientos
  - Lenguajes de programación orientados a objetos
- La POO intenta asimilarse a la vida real ¿Qué tenemos en la vida real? **Objetos**
  - Coche
  - Mesa
  - Lavadora
- En la vida real esos objetos tienen
  - Características (alto, ancho, color, peso, temperatura, etc.)
  - Capacidades (lavar ropa, soportar objetos, etc.)

## CARACTERÍSTICAS



**Propiedades** (cómo es el objeto):

- Tiene puertas
- Tiene ruedas
- Tiene ventanas
- Tiene motor
- Tiene color
- Etc.



Objeto = coche

## CAPACIDADES



**Métodos** (¿Qué puede hacer?)

- Arrancar
- Acelerar
- Frenar
- Girar
- Etc.

También sabemos que en la vida real hay muchos tipos de coches



A los diferentes tipos de coches en programación es lo que denominamos **instancias**



Instancia de coche. Coche1

color=verde  
puertas = 2  
Ruedas=pequeñas  
Motor=pequeño

Ambos

Arrancan  
Aceleran  
Frenan  
Giran

Instancia de coche. Coche2

Color= rojo  
puertas = 4  
Ruedas=grandes  
Motor=grande

# Objetos en JavaScript

- ¿Cómo nos llevamos el concepto de objeto en la vida real a la programación?
- Ejemplo, supongamos un botón que insertamos en nuestro formulario de contacto:
  - Ese botón es un objeto
  - Como objeto que es ese botón tendrá
    - Propiedades
      - Ancho
      - Alto
      - Color
    - Métodos
      - Capturar el foco
      - Hacer clic
      - Hacer doble clic
      - Etc.

# Nomenclatura del punto

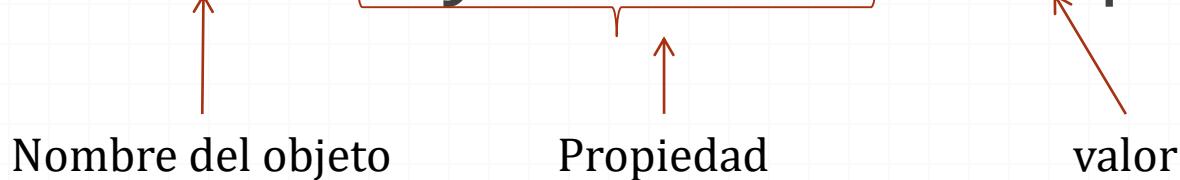
- ¿Cómo indicamos que queremos cambiar una propiedad?
- En JavaScript se utiliza la jerarquía y el operador punto.
  - Por ejemplo, supongamos que en nuestro coche anterior queremos modificar la propiedad color:

```
seat600.color="verde";
```



- Ejemplos en JavaScript:

```
boton.style.width="500px";
```



○ `boton.style.backgroundColor="red";`

○ En cuanto a los métodos ¿cómo indicamos a nuestro objeto que realice algo que es capaz de hacer?

○ Por ejemplo, vamos a decirle a nuestro coche que acelerar:

`seat600.acelera();`

Nombre del objeto

método

○ Ejemplos en JavaScript:

`document.write();`

Nombre del objeto

método

○ `window.alert();`

○ `boton.focus();`

# Ejemplo. Manipular propiedades

- Vamos a crear una página que inserte un botón en una página web

```
<body>
    <input type="button" id="boton1">
</body>
```

- Crear un script que modifique el ancho y el alto de ese botón

- Crear una variable que almacene el objeto con id="boton1"

```
var miboton=document.getElementById("boton1");
```

- Cambiar las propiedades ancho y alto del objeto botón

```
miboton.style.width="200px";
```

```
miboton.style.height="30px";
```

- Cada tipo de objeto tiene sus propiedades

- Añadir al script anterior un cuadro de texto de 300px de ancho y con un valor predeterminado que ponga "Escribe aquí tu mail"



Creando nuevos  
objetos

# Definir las propiedades

- JS tiene una serie de objetos predefinidos.
- Pero puedes crear tus propios objetos.
- Podemos crear un objeto haciendo uso del inicializador de objeto cuya sintaxis es la siguiente:

```
var objeto = { propiedad_1: valor_1,  
               propiedad_2: valor_2,  
               ...  
               "propiedad n": valor_n };
```

- Donde, objeto es el nombre del nuevo objeto
- Cada propiedad\_i es un identificador (ya sea un nombre, un número o una cadena literal)
- Cada valor\_i es una expresión cuyo valor se asigna a la propiedad\_i

```
var coche = {tipo:"Seat", modelo:"600", color:"verde"};
```

# Definir los métodos

- Los métodos son las acciones que pueden realizar los objetos
- Los métodos se definen a través de funciones