Unidad N° 3

Tecnología de desarrollo web del lado del cliente

Elementos básicos de JavaScript

Introducción

- Con HTML y CSS es posible crear páginas web estáticas.
- Con JavaScript, se pueden diseñar páginas web dinámicas de cliente.
- Para esto, dentro del código de la página web se puede incluir el script que permitirá modificar la página web dinámicamente en el lado del cliente, sin sobrecargar de trabajo al servidor web.
  - La inclusión puede hacerse en línea, embebido o por archivos externos.

Elementos básicos de JavaScript

Características de JavaScript

- No es un lenguaje puro del paradigma orientado a objetos, pero lo soporta.
- Maneja eventos y tiene acceso a los elementos del navegador y los del documento web.
- Es interpretado por el navegador web, y por lo tanto débilmente tipado (menos control sobre variables).
- Hereda la sintaxis de C.
- No comparte nada con Java, excepto el nombre y parte de la sintaxis.
- Los scripts se incorporan al código de la página web o se asocian a ella y es interpretado en el navegador.

Elementos básicos de JavaScript

El primer script

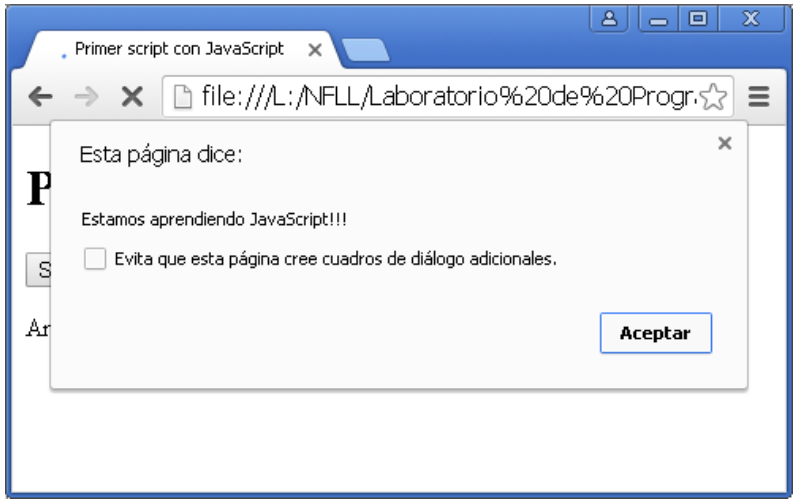
- El script puede incorporarse tanto dentro del cuerpo como del encabezado de la página web. Lo usual es definir las funciones del script en el encabezado.
- Para esto, HTML incluye la etiqueta *<script>..</script>*, la cual tiene varios atributos. Algunos son:
  - type: indica tipo del script que se incluirá dentro de la etiqueta. Para JavaScript es: "*text/javascript*".
  - src: indica el nombre del archivo "js" que contiene el código JavaScript. Este archivo puede incluir desde una lista de constantes y variables hasta una biblioteca completa de funciones.

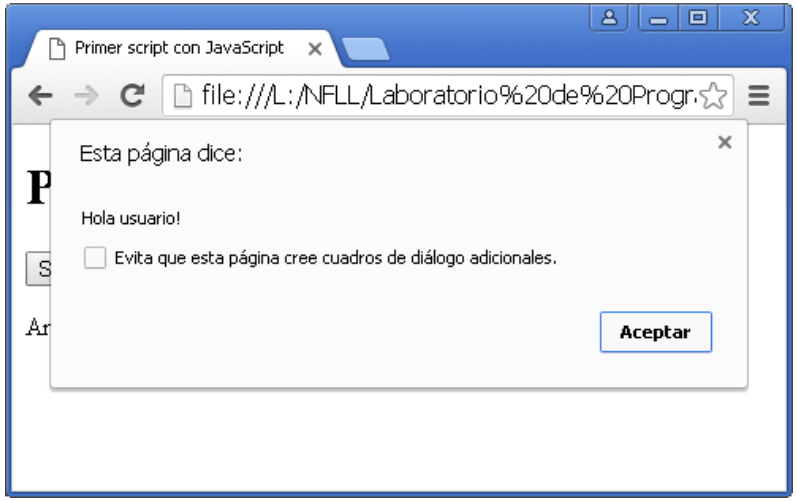
Elementos básicos de JavaScript

Ejemplo (JavaScript incorporado en línea)

```
1  <!-- Codigo_56.htm -->
2  <!DOCTYPE html>
3  <html lang="es">
4  <head>
5  <title>Primer script con JavaScript</title>
6  </head>
7  <body onLoad="alert('Estamos aprendiendo JavaScript!!!');">
8  <header>
9      <h1>Primer script con JavaScript</h1>
10 </header>
11 <section>
12     <article>
13     <button id="boton" type="button" onClick="alert('Hola usuario!');">
14     Saludame :)
15     </button>
16     </article>
17 </section>
18 <footer>
19     <p>Archivo: Codigo_56.htm</p>
20 </footer>
21 </body>
22 </html>
```

Elementos básicos de JavaScript





Elementos básicos de JavaScript

Ejemplo (JavaScript incorporado embebido)


```
1  <!-- Codigo_55.htm -->
2  <!DOCTYPE html>
3  <html lang="es">
4  <head>
5  <title>Primer script con JavaScript</title>
6  </head>
7  <body>
8  <header>
9      <h1>Primer script con JavaScript</h1>
10 </header>
11 <section>
12     <article>
13     <script type="text/javascript">
14     document.write("<h1>Estamos aprendiendo JavaScript!!!</h1>");
15     </script>
16     </article>
17 </section>
18 <footer>
19     <p>Archivo: Codigo_55.htm</p>
20 </footer>
21 </body>
```

Elementos básicos de JavaScript

Ejemplo (JavaScript incorporado embebido)


```
1  <!-- Codigo_107.htm -->
2  <!DOCTYPE html>
3  <html lang="es">
4  <head>
5  <title>Primer script con JavaScript</title>
6  <script type="text/javascript">
7  function imprimoCuerpo()
8  {
9      document.write("<header>");
10     document.write("<h1>Primer script con JavaScript</h1>");
11     document.write("</header>");
12     document.write("<section>");
13     document.write("<article>");
14     document.write("<h2>Este texto es un art&iacute;culo</h2>");
15     document.write("</article>");
16     document.write("</section>");
17     document.write("<footer>");
18     document.write("<p>Archivo: Codigo_107.htm</p>");
19     document.write("</footer>");
20 }
21 </script>
22 </head>
23 <body onLoad="imprimoCuerpo();">
24 </body>
```

Elementos básicos de JavaScript

Ejemplo (JavaScript con archivos externos)

```
1  <!-- Codigo_57.htm -->
2  <!DOCTYPE html>
3  <html lang="es">
4  <head>
5  <title>Primer script con JavaScript</title>
6  <script type="text/javascript" src="Codigo_57.js">
7  </script>
8  </head>
9  <body onLoad="imprimoCuerpo();">
10 </body>
11 </html>
```

```
1  <!-- Codigo_57.js -->
2  function imprimoCuerpo()
3  {
4      document.write("<header>");
5      document.write("<h1>Primer script con JavaScript</h1>");
6      document.write("</header>");
7      document.write("<section>");
8      document.write("<article>");
9      document.write("<h2>Este texto es un art&iacute;culo</h2>");
10     document.write("<p>El texto es impreso desde afuera</p>");
11     document.write("</article>");
12     document.write("</section>");
13     document.write("<footer>");
14     document.write("<p>Archivo: Codigo_57.htm</p>");
15     document.write("</footer>");
16 }
```

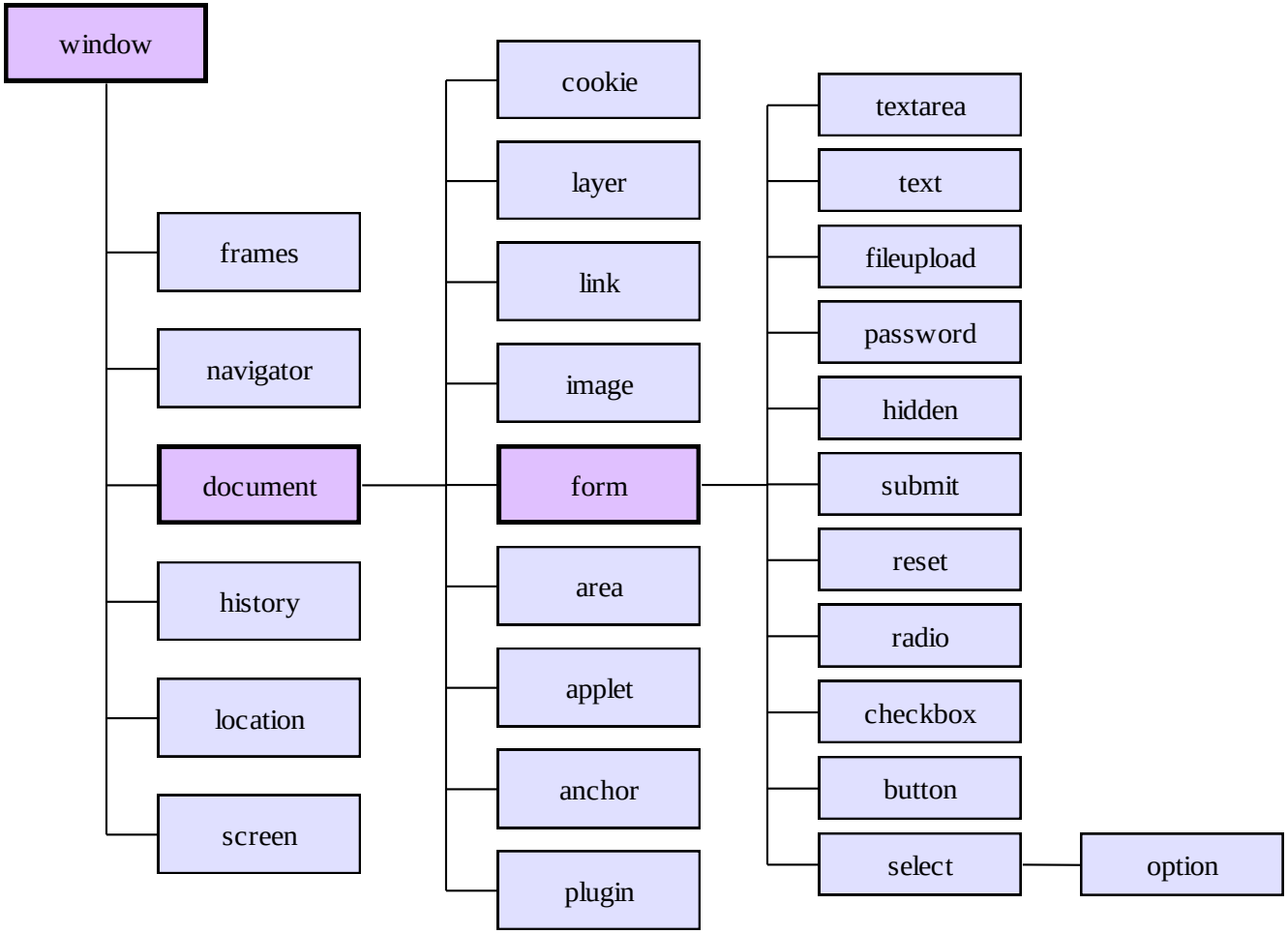
Document Object Model (DOM)

Introducción

- Como sabemos, JavaScript no es un lenguaje puro de objetos pero soporta el paradigma.
- Para esto, incorpora una jerarquía de objetos que:
  - Permiten gestionar varias cuestiones relativas al navegador que muestra un documento web:
    - Ejemplos: crear y cerrar pestañas del browser.
  - Representan a todos y cada uno de los elementos estructurales HTML que se incorporan en un documento web.
    - Y gracias a esto, permite gestionarlos mediante operaciones de creación, actualización y eliminación.

Document Object Model (DOM)

Jerarquía de clases en JavaScript



Document Object Model (DOM)

- Dentro de esta jerarquía, se destacan los objetos:
  - Window: Representa a la ventana o pestaña del navegador que muestra la página web.
  - Document: contiene el documento web actual. Se genera cuando se encuentra la etiqueta <*body*> en el documento HTML.
  - History: contiene la lista de páginas accedidas desde el navegador.
  - Location: contiene la URL de la página actual en el browser.
  - Navigator: contiene información del browser.
  - Form: contiene todos los elementos incluidos en un formulario HTML. Se genera uno por cada formulario.

Document Object Model (DOM)

Equivalencias entre objetos JavaScript y HTML

JavaScript	HTML		JavaScript	HTML
document	<body>		radio	<input type="radio">
form	<form>		checkbox	<input type="checkbox">
button	<input type="button">		select	<select>
reset	<input type="reset">		fileupload	<input type="file">
submit	<input type="submit">		hidden	<input type="hidden">
textarea	<textarea>		link	<a href="…">
text	<input type="text">		anchor	<a name="…">
password	<input type="password">		img	<img>

Document Object Model (DOM)

El objeto window

- Algunos de los atributos del objeto window son:

Propiedad	Descripción
closed	Indica si la ventana fue cerrada o no.
defaultStatus, status	Contiene el mensaje que aparece en la barra de estado del browser.
name	Nombre de la ventana.
opener	Referencia al objeto window que abrió la ventana.

Document Object Model (DOM)

- Algunos de los métodos del objeto window son:

Método	Descripción
alert(), confirm(), prompt()	Mensajes de aviso al usuario.
close()	Cierra la ventana.
focus(), blur()	Hace que la ventana esté activa o inactiva.
moveBy(), moveTo()	Mover la ventana a una posición determinada.
print()	Imprime la ventana.
resizeBy(), resizeTo()	Ajusta el tamaño de la ventana.
open()	Abre una ventana nueva, de nivel superior, con una dirección específica.

Mas información: https://developer.mozilla.org/es/docs/Web/API/Window

Document Object Model (DOM)

El objeto document

- Algunos de los atributos del objeto document son:

Propiedad	Descripción
bgColor	Cadena que mantiene el valor del atributo bgcolor.
forms	Arreglo que contiene todos los formularios definidos en el documento web.
URL	URL que esta cargada en el documento
links	Arreglo con todos los hipervínculos definidos en el documento web.
images	Arreglo con referencias a todas las imágenes incluidas en el documento web.
title	Contiene el título de la página web. Es una propiedad de lectura/escritura.

Document Object Model (DOM)

- Algunos de los métodos del objeto document son:

Método	Descripción
close()	Cierra la escritura del documento web y fuerza la visualización de su contenido.
open()	Abre la escritura sobre el documento. El primer parámetro indica el tipo de documento. El segundo parámetro tiene el valor “replace” que indica que se abre sobre el documento anterior.
write()	Escribe una o mas expresiones HTML en el documento web.
writeln()	Idem a write pero incorpora un salto de línea.

Mas información: https://developer.mozilla.org/es/docs/Web/API/Document

Document Object Model (DOM)

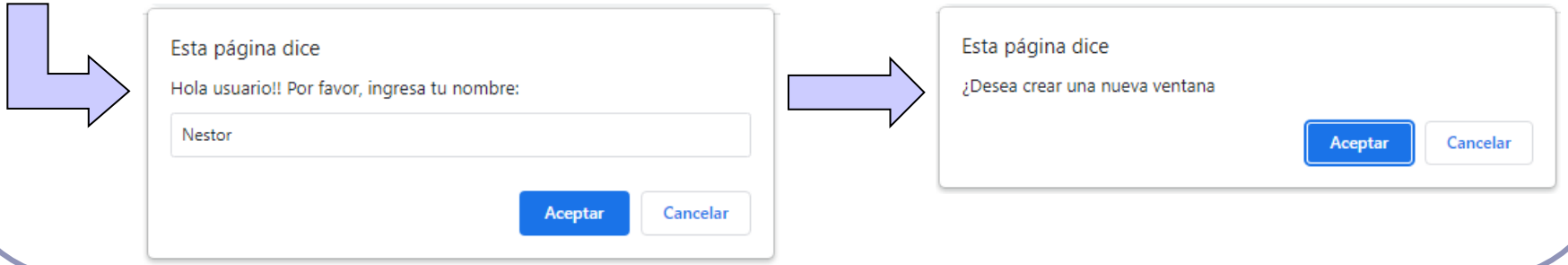
Creación de ventanas nuevas

- El método *open* tiene tres parámetros:
  - URL: indica la URL que se abrirá en la nueva ventana.
  - Name: indica el nombre de la ventana en la cual se abrirá la URL indicada como primer parámetro. Si no existe, se crea una nueva con las características del tercer parámetro.
  - Características: es una cadena que incluye las características de la nueva ventana. Algunas de ellas son: toolbar, location, status, menubar, scrollbars, resizable, width, height, left, top.
- Ejemplo: window.open(“Ejemplo.htm”, “nueva”, “width=400,height=350,status=yes,resizable=yes”)

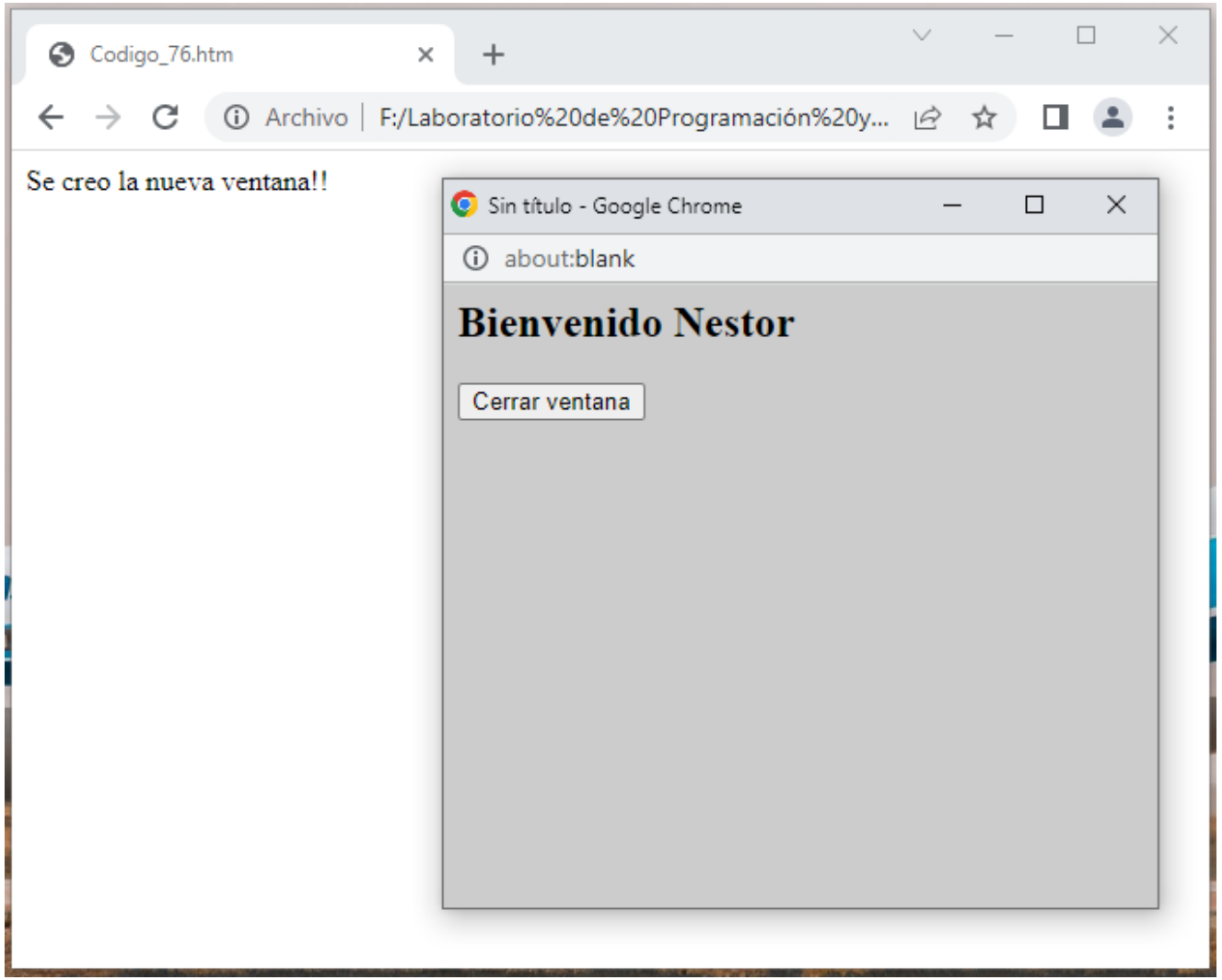
Document Object Model (DOM)

Ejemplo

```
<!-- Codigo_76.htm -->
<!DOCTYPE html>
<html lang="es">
<head>
<script type="text/javascript">
function principal()
{
var nombre, respuesta, newWindow;

window.document.title = "Estamos creando y cerrando ventanas nuevas!!";
nombre = window.prompt("Hola usuario!! Por favor, ingresa tu nombre:");
respuesta = window.confirm("¿Desea crear una nueva ventana");
if (respuesta)
{   newWindow = window.open("","","width=400,height=350,status=yes,resizable=yes,menubar=no");
    newWindow.document.write("<h2>Bienvenido " + nombre + "</h2>");
    newWindow.document.write("<input id=\"botonCerrar\" name=\"botonCerrar\" type=\"button\" onclick=\"window.close();\" value=\"Cerrar ventana\" />");
    newWindow.document.bgColor = "#CCCCCC";
    newWindow.focus();
    window.document.write("Se creo la nueva ventana!!<br>");
}
}
</script>
</head>
<body onLoad="principal();">
</body>
</html>
```



Document Object Model (DOM)



Document Object Model (DOM)

Eventos

- Los principales eventos manejados por JavaScript son:

Manejador	Se produce cuando
onBlur	El elemento de un form o una ventana pierde el foco.
onChange	El campo select, text o textarea pierde el foco y su valor fue modificado.
onClick	El usuario pulsa sobre un objeto del documento.
onDblClick	El usuario pulsa dos veces sobre un objeto del documento.
onDragDrop	El usuario libera un objeto después de arrastrarlo.
onError	La carga de una imagen o documento produce un error.

Document Object Model (DOM)

Manejador	Se produce cuando
onFocus	Un objeto recibe el foco.
onKeyDown	El usuario pulsa una tecla.
onKeyPress	El usuario pulsa una tecla o la mantiene pulsada.
onKeyUp	El usuario libera una tecla que mantenía pulsada.
onLoad	El navegador termina de cargar un documento web
onMouseDown	El usuario presiona un botón del mouse.
onMouseMove	El usuario mueve el mouse.
onMouseOut	El puntero sale del área donde estaba ubicado.

Document Object Model (DOM)

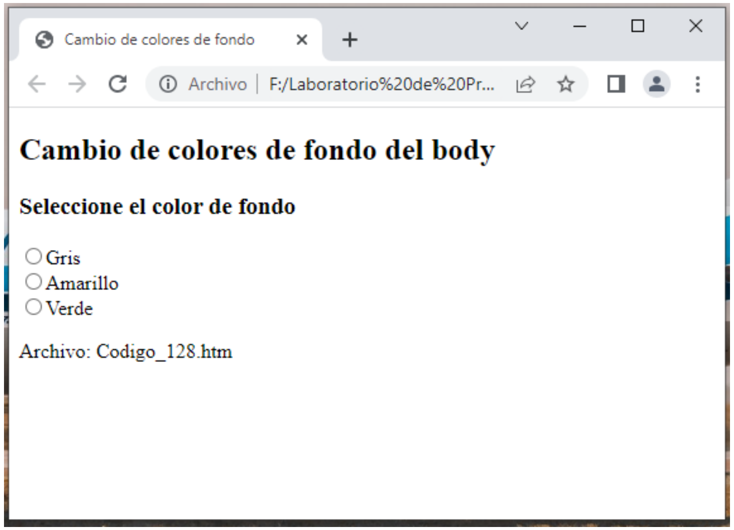
Manejador	Se produce cuando
onMouseOver	El puntero se mueve sobre un objeto o área.
onMouseUp	El usuario libera un botón del mouse que estaba presionando.
onMove	Se mueve una ventana.
onReset	Se restaura el formulario que contiene el botón.
onResize	Se cambia de tamaño una ventana.
onSelect	El usuario selecciona un texto en un cuadro o área de texto.
onSubmit	Se envía el formulario que contiene el botón.
onUnload	El usuario abandona o cierra un documento web.

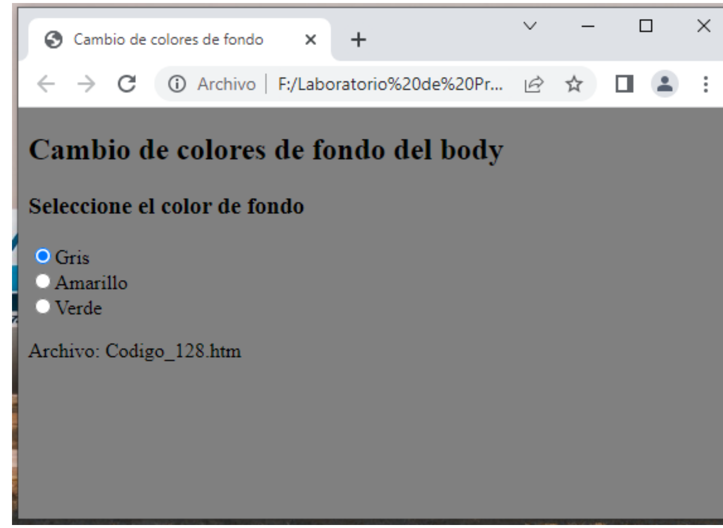
Document Object Model (DOM)

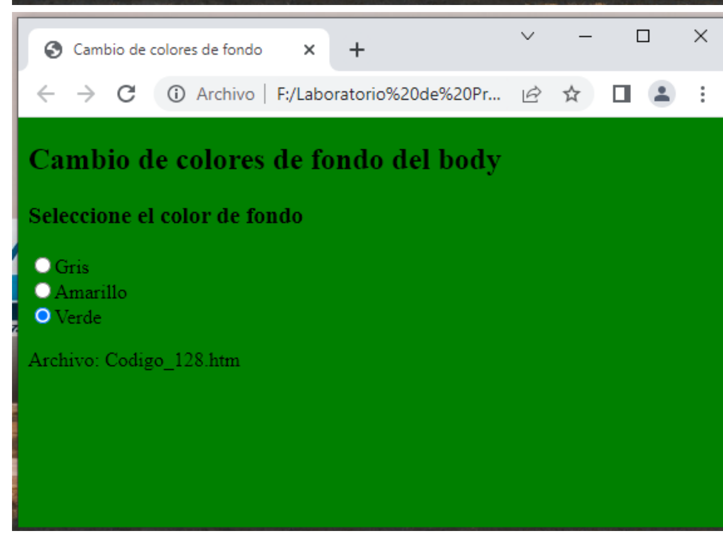
Ejemplo

```
1  <!-- Codigo_128.htm -->
2  <!DOCTYPE html>
3  <html lang="es">
4  <head>
5  <title>Cambio de colores de fondo</title>
6  <script type="text/javascript">
7  function cambioColor(colour)
8  {
9      window.document.bgColor = colour;
10 }
11 </script>
12 </head>
13 <body>
14 <header>
15     <h2>Cambio de colores de fondo del body</h2>
16 </header>
17 <section>
18     <article>
19     <p>
20     <h3>Seleccione el color de fondo</h3>
21     <input name="radio" id="radio1" type="radio" value="grey" onClick="cambioColor(this.value);"/>Gris<br>
22     <input name="radio" id="radio2" type="radio" value="yellow" onClick="cambioColor(this.value);"/>Amarillo<br>
23     <input name="radio" id="radio3" type="radio" value="green" onClick="cambioColor(this.value);"/>Verde<br>
24     </p>
25     </article>
26 </section>
27 <footer>
28     <p>Archivo: Codigo_128.htm</p>
29 </footer>
30 </body>
31 </html>
```

Document Object Model (DOM)







Document Object Model (DOM)

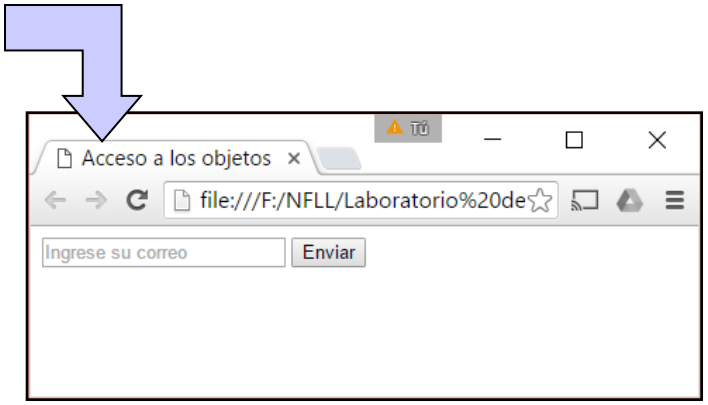
Acceso a los objetos del documento

- Existen varias formas de acceder a los objetos de un documento web:
  - La primera es utilizando la jerarquía de objetos dentro del DOM
  - La segunda es utilizando métodos del objeto *document*:
    - getElementById
    - getElementsByTagName
    - getElementsByClassName

Document Object Model (DOM)

Primera forma: usando jerarquía del DOM

```
1  <!-- Codigo_73.htm -->
2  <!DOCTYPE html>
3  <html lang="es">
4  <head>
5  <title>Acceso a los objetos</title>
6  </head>
7  <body>
8  <section>
9  <article>
10 <form id="frmMiFormulario" name="miFormulario" action="" method="post">
11   <input id="txtCorreo" name="correo" type="email" placeholder="Ingrese su correo">
12   <input id="btnEnviar" name="enviar" type="submit" value="Enviar">
13 </form>
14 </article>
15 </section>
16 </body>
17 </html>
```



URL	• window.location.href
Titulo	• window.document.title
Cantidad de elementos en el formulario	• window.document.forms[0].elements.length • window.document.forms[“miFormulario”].elements.length • window.document.miFormulario.elements.length • window.document[“miFormulario”].elements.length
Valor del cuadro de texto	• window.document.forms[“miFormulario”].elements[0].value • window.document.forms[0].elements[“Correo”].value • window.document[“miFormulario”][“Correo”].value • window.document[0][0].value

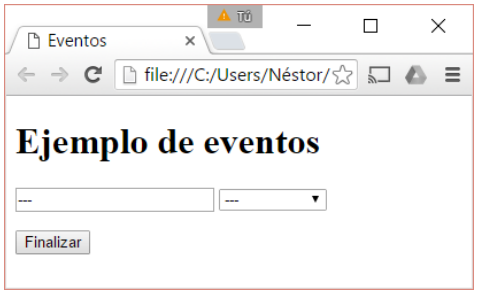
Document Object Model (DOM)

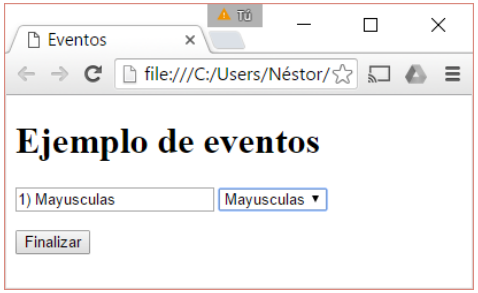
Segunda forma: métodos de document

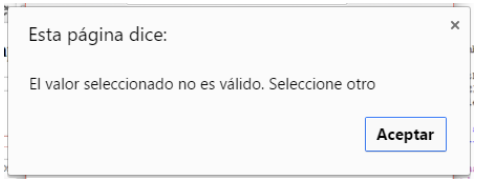
- Es la forma más utilizada. Consiste en acceder a los elementos del documento web a través de métodos del objeto *document*:
  - getElementById: Recibe como parámetro el id del elemento y a partir de él, obtiene un elemento específico del documento y crea una referencia hacia él.
  - getElementsByTagName: obtiene todos los elementos de una etiqueta HTML determinada, que se especifica como parámetro del método. Los elementos se devuelven como referencias en un arreglo.
  - getElementsByClassName: ídem al anterior, pero en el parámetro se indica el nombre de la clase.

Document Object Model (DOM)

Ejemplo







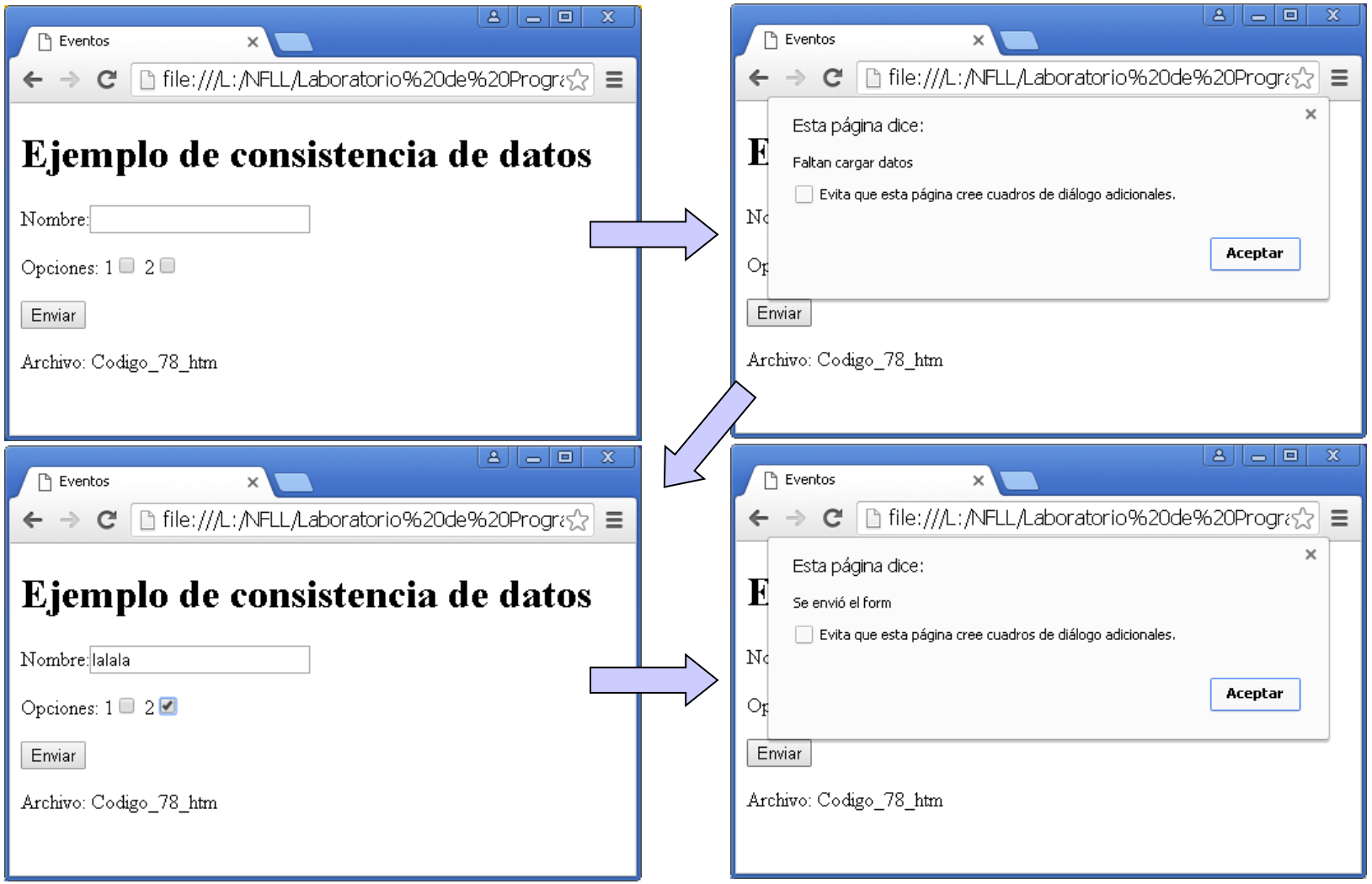

```
<!-- Codigo_77.htm -->
<!DOCTYPE html>
<html lang="es">
<head>
<title>Eventos</title></head>
<script language="javascript" >
function cerrarVentana()
{   window.close(); }

function seleccionoFormato()
{   var indSeleccionado, indCombo, valorCombo;

    indSeleccionado = window.document.forms[0].formatos.selectedIndex;
    if (indSeleccionado == 0)
    {   window.alert("El valor seleccionado no es válido. Seleccione otro");
        window.document.forms[0].nombreUsuario.value = "---";   }
    else
    {   indCombo = window.document.forms[0].formatos.options[indSeleccionado].value;
        valorCombo = window.document.forms[0].formatos.options[indSeleccionado].text;
        window.document.forms[0].nombreUsuario.value = indCombo + ") " + valorCombo;    }
}
</script>
<body>
<form id="frmFormulario" name="formulario" method="post" action="">
  <p>
    <input id="txtNombreUsuario" name="nombreUsuario" type="text" value="---" readonly>
    <select id="cmbFormatos" name="formatos" onChange="seleccionoFormato();">
      <option value="0" selected>---</option>
      <option value="1">Mayusculas</option>
      <option value="2">Minusculas</option>
      <option value="3">Negrita</option>
    </select>
  </p>
  <p><input id="btnCerrar" name="Cerrar" type="submit" onClick="cerrarVentana();" value="Finalizar"></p>
</form>
</body>
</html>
```

Document Object Model (DOM)

Ejemplo

```
<!-- Codigo_78.htm -->
<!DOCTYPE html>
<html lang="es">
<head>
<title>Eventos</title>
<script>
function enviarForm()
{   var condi1, condi2, check1, check2, cuadroTexto, formulario;
    formulario = document.getElementById("frmFormPpal");
    cuadroTexto = document.getElementById("txtNombre");
    check1 = document.getElementById("chkOpcion1");
    check2 = document.getElementById("chkOpcion2");
    condi1 = (cuadroTexto.value == "");
    condi2 = (!(check1.checked || check2.checked));
    if ((condi1) || (condi2))
    {   alert("Faltan cargar datos");
        cuadroTexto.focus();    }
    else
    {   alert("Se envió el form");
        formulario.submit();    }
}
</script>
</head>
<body>
<header>
<h1>Ejemplo de consistencia de datos</h1>
</header>
<section>
<article>
<form id="frmFormPpal" name="formulario" method="post" action="script.php">
  <p>Nombre:<input id="txtNombre" name="tNombre" type="text"></p>
  <p>Opciones:
    1<input id="chkOpcion1" name="cOpc1" type="checkbox" value="1">
    2<input id="chkOpcion2" name="cOpc2" type="checkbox" value="1">
  </p>
  <p><input id="btnEnviar" name="bEnviar" type="button" onClick="enviarForm();" value="Enviar"></p>
</form>
</article>
</section>
<footer><p>Archivo: Codigo_78_htm</p></footer>
</body>
</html>
```

Document Object Model (DOM)

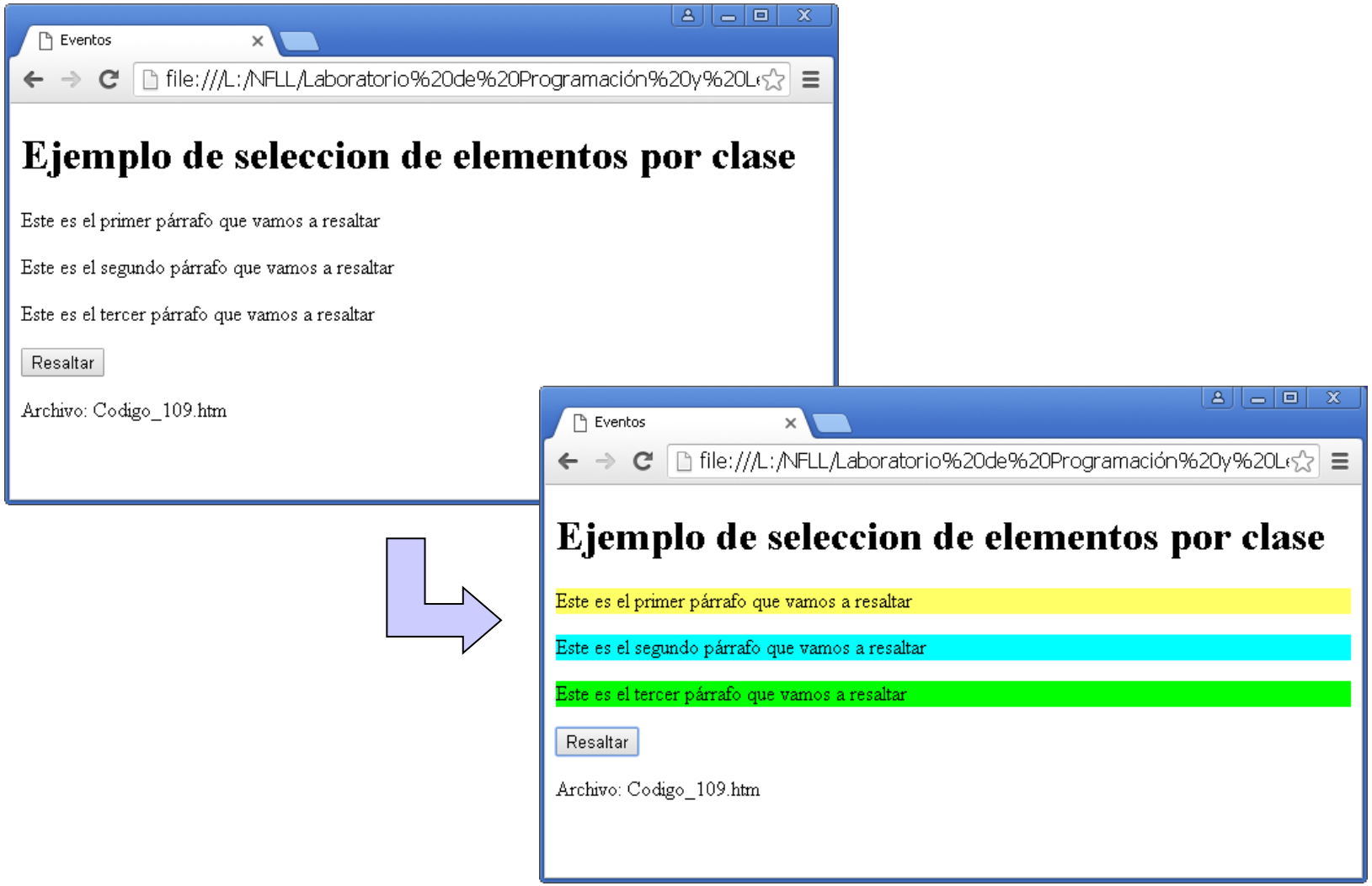


Document Object Model (DOM)

Ejemplo

```
<!-- Codigo_109.htm -->
<!DOCTYPE html>
<html lang="es">
<head>
<title>Eventos</title>
<script>
function resaltar()
{   var colores = new Array("#FFFF66","#00FFFF","#00FF00");
    var lstParrafos, indice;

    lstParrafos = document.getElementsByClassName("parrafo");
    for (indice in lstParrafos)
    {
        lstParrafos[indice].style.backgroundColor = colores[indice];
    }
}
</script>
</head>
<body>
<header>
<h1>Ejemplo de seleccion de elementos por clase</h1>
</header>
<section>
<article>
  <p class="parrafo">Este es el primer p&aacute;rrafo que vamos a resaltar</p>
  <p class="parrafo">Este es el segundo p&aacute;rrafo que vamos a resaltar</p>
  <p class="parrafo">Este es el tercer p&aacute;rrafo que vamos a resaltar</p>
  <p>
  <input id="btnResaltar" name="bResaltar" type="button" onClick="resaltar();" value="Resaltar">
  </p>
</article>
</section>
<footer><p>Archivo: Codigo_109.htm</p></footer>
</body>
</html>
```

Document Object Model (DOM)



Document Object Model (DOM)

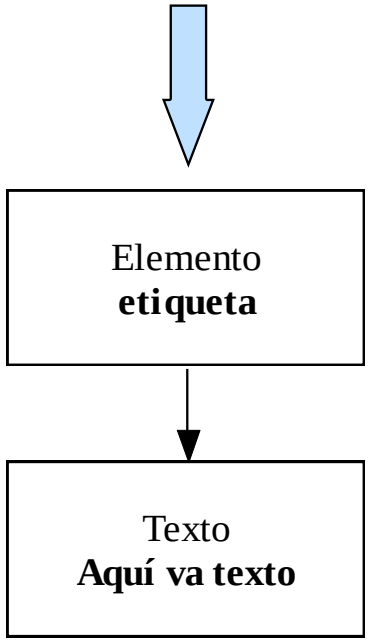
Arborescencia

- Los documentos web codificados utilizando HTML pueden esquematizarse como una arborescencia jerárquica.
- En una estructura de arborescencia, los diferentes componentes se representan con nodos.
- Así, un documento web se puede representar con una estructura de árbol, donde los nodos equivalen a los elementos estructurales de la página web.
- El árbol depende del browser: es generado cuando se carga la página web y pueden tener versiones distintas dependiendo del navegador.

Document Object Model (DOM)

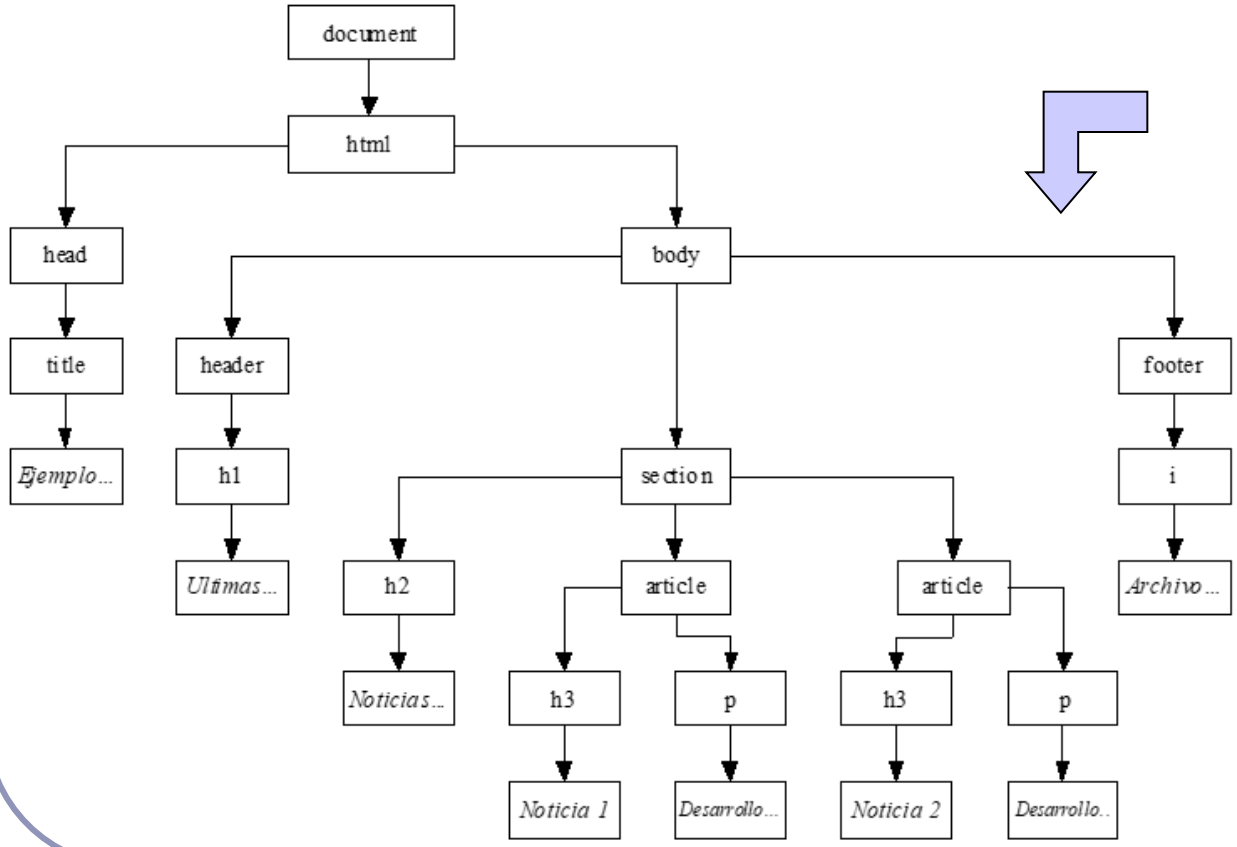
- En lo más alto de la jerarquía, se ubica el nodo “Document”.
- Los nodos pueden ser de elementos o de texto. Formalmente, se representan así:

<etiqueta>Aqui va texto</etiqueta>



Document Object Model (DOM)

Ejemplo




```
<!-- Codigo_80.htm -->
<!DOCTYPE html>
<html lang="es">
<head>
<title>Ejemplo de section y article</title>
</head>
<body>
<header>
    <h1>&Uacute;ltimas noticias</h1>
</header>
<section id="news-list">
    <h2>Noticias de hoy</h2>
    <article>
        <h3>Noticia 1</h3>
        <p>Desarrollo de la noticia 1</p>
    </article>
    <article>
        <h3>Noticia 2</h3>
        <p>Desarrollo de la noticia 2</p>
    </article>
</section>
<footer><i>Archivo: Codigo_80.htm</i></footer>
</body>
</html>
```

Document Object Model (DOM)

Propiedades de los nodos

Propiedad	Descripción
Attributes	lista de atributos del nodo actual
childNodes	lista de los hijos de un nodo
firstChild	primer hijo de un nodo
innerHTML	el contenido y el código que hay dentro del nodo
lastChild	último hijo de un nodo
nodeName	nombre del nodo
nodeType	tipo del nodo
nodeValue	valor contenido en el nodo
parentNode	obtiene el nodo padre de un nodo

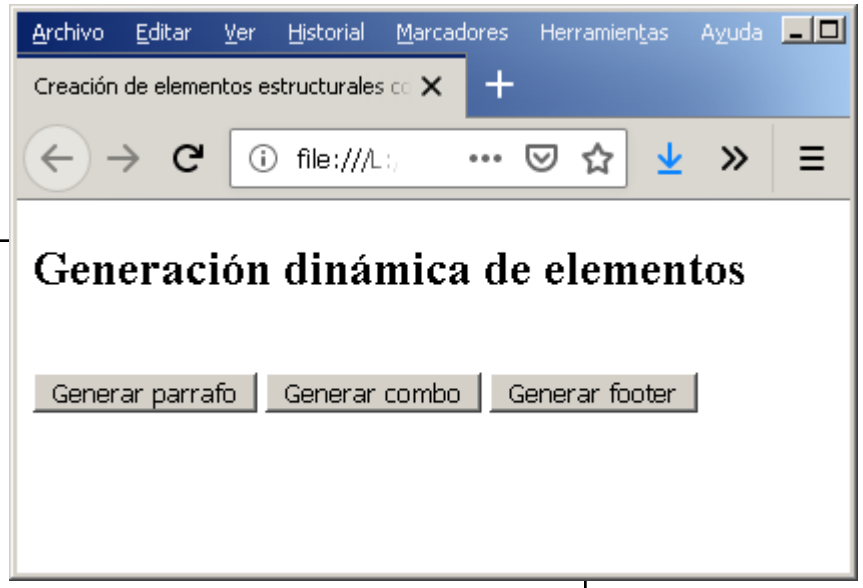
Document Object Model (DOM)

Métodos de los nodos

Método	Descripción
appendChild()	incorporación de un nodo hijo
removeChild()	eliminación de un nodo
hasChildNodes()	control de la existencia de nodos hijos
insertBefore()	inserción de un nodo
cloneNode()	duplicación de un nodo
getAtribute()	obtención del valor de un atributo de un nodo
removeAtribute()	eliminación del valor de un atributo de un nodo
setAtribute()	definición del valor de un atributo de un nodo

Document Object Model (DOM)

Ejemplo




```
<!-- Codigo_121.htm -->
<!DOCTYPE html>
<html lang="es">
<head>
<title>Creación de elementos estructurales con JavaScript</title>
<script type="text/javascript" src="Codigo_121.js"></script>
</head>

<body>
    <header>
        <h2>Generaci&oacuten din&aacutemica de elementos</h2>
    </header>
    <section>
        <article id="articulo">
        <div id="contenedor"></div><br>
        <input type="button" onClick="creoParrafo();" id="boton1" name="botonParrafo" value="Generar parrafo">
        <input type="button" onClick="creoCombo();" id="boton2" name="botonCombo" value="Generar combo">
        <input type="button" onClick="creoPie();" id="boton3" name="botonPie" value="Generar footer">
        </article>
    </section>
</body>
</html>
```

Document Object Model (DOM)

```
 6  function creoParrafo()
 7  {
 8      var enDiv = document.getElementById("contenedor");
 9      var oldParrafo = document.getElementById("parrafoNuevo");
10      if (oldParrafo)
11      {   padre = oldParrafo.parentNode;
12          padre.removeChild(oldParrafo);
13      }
14      var parrafo = document.createElement("p");
15      parrafo.setAttribute("id","parrafoNuevo");
16      parrafo.innerHTML = "Este parrafo se gener&oacute dinamicamente cuando se puls&oacute el bot&oacuten";
17      enDiv.appendChild(parrafo);
18  }
```

```
44  function creoPie()
45  {
46      var oldPie = document.getElementById("pie");
47      if (oldPie)
48      {   padre = oldPie.parentNode;
49          padre.removeChild(oldPie);
50      }
51      var newPie = document.createElement("footer");
52      newPie.setAttribute("id","pie");
53      var newParrafo = document.createElement("p");
54      newParrafo.innerHTML = "Archivo: Codigo_121.htm";
55      newPie.appendChild(newParrafo);
56      document.body.appendChild(newPie);
57  }
```

Document Object Model (DOM)

```
20  function creoCombo()
21  {
22      var cantidad = 0;
23      var enDiv = document.getElementById("contenedor");
24      var oldCombo = document.getElementById("comboNuevo");
25      if (oldCombo)
26      {   padre = oldCombo.parentNode;
27          padre.removeChild(oldCombo);
28      }
29      var combo = document.createElement("select");
30      combo.setAttribute("id","comboNuevo");
31      do
32      {
33          cantidad = parseInt(prompt("Ingrese la cantidad de opciones a generar (maximo 10)"));
34      } while (cantidad<2 || cantidad>10 || isNaN(cantidad));
35      for(i=0;i<cantidad;i++)
36      {   var opcion = document.createElement("option");
37          opcion.setAttribute("value",i);
38          opcion.innerHTML = "Opcion "+i;
39          combo.add(opcion);
40      }
41      enDiv.appendChild(combo);
42  }
```

Cookies

Introducción

- Una cookie es un recurso que permite al sitio o aplicación web guardar información en un cliente.
  - En si, es un archivo de texto almacenado en el cliente, al que sólo puede acceder y procesar el sitio o la aplicación web.
- Así, este recurso permite a los programadores guardar información de manera limitada, a fin de poder utilizarla en futuros accesos de los usuarios al sitio web.
- Un ejemplo de uso de cookies se observa en los sistemas webmail, o los contadores de visitas a los sitios web.

Cookies

- Si éstos sistemas utilizan cookies, cuando el usuario ingresa al sitio web, éstas serán procesadas y utilizadas en el script.
  - Si no hubieran cookies guardadas en el cliente, entonces el sitio web las crea y guarda para que sean utilizadas en el próximo ingreso.
- Así, el primer paso para trabajar con las cookies es crearlas, el segundo paso es leerlas y, una vez que ya no se utilicen se deben borrar.
- Para el caso de JavaScript, las cookies se guardan en el atributo *cookie* del objeto *document*, por lo que todas las acciones se realizan sobre este elemento.

Cookies

Gestión de cookies

- Para crearlas se deben tener los siguientes datos:

Dato	Descripción
Nombre (*)	Nombre de la cookie. Utilizar identificadores representativos.
Valor (*)	Valor de la cookie. Puede contener caracteres especiales. Para esto, utilizar la función escape() para codificarlos. Luego, cuando se lee la cookie, se debe decodificar con la función unescape().
Expiración	Momento de expiración de la cookie. Puede ser una fecha concreta (expires) o bien una cantidad entera (max-age) que represente a los segundos que la cookie permanecerá en la computadora del usuario. Si no se especifica, la cookie se eliminará cuando termine la sesión o se cierre el navegador.
Ruta	Indica la ruta en el servidor (path). Con el valor “/”, la cookie podrá ser utilizada en todo el sitio web. Si no se indica, el valor por defecto es la URL de la página.
Dominio	El dominio (domain) es similar a la ruta, pero aplicable al dominio del sitio
Seguridad	Este parámetro (secure) Indica si la cookie se utilizará en conexiones seguras HTTPS

(*) obligatorios

Cookies

- Ejemplo de creación de cookie:

document.cookie = "usario=nestor; path=/; expires=Mon, 08 May 2023 23:59:59 GMT; secure“;

- La lectura de una cookie no tiene una función o método específico, sino que el programador debe “ingeniárselas” para extraer el valor de la lista de cookies almacenadas que son accesibles por la página web.
  - Al tratarse de una cadena, la extracción se realiza aplicando métodos del objeto string
- Finalmente, el borrado de una cookie se realiza creándola nuevamente, pero con una fecha de expiración anterior a la actual.

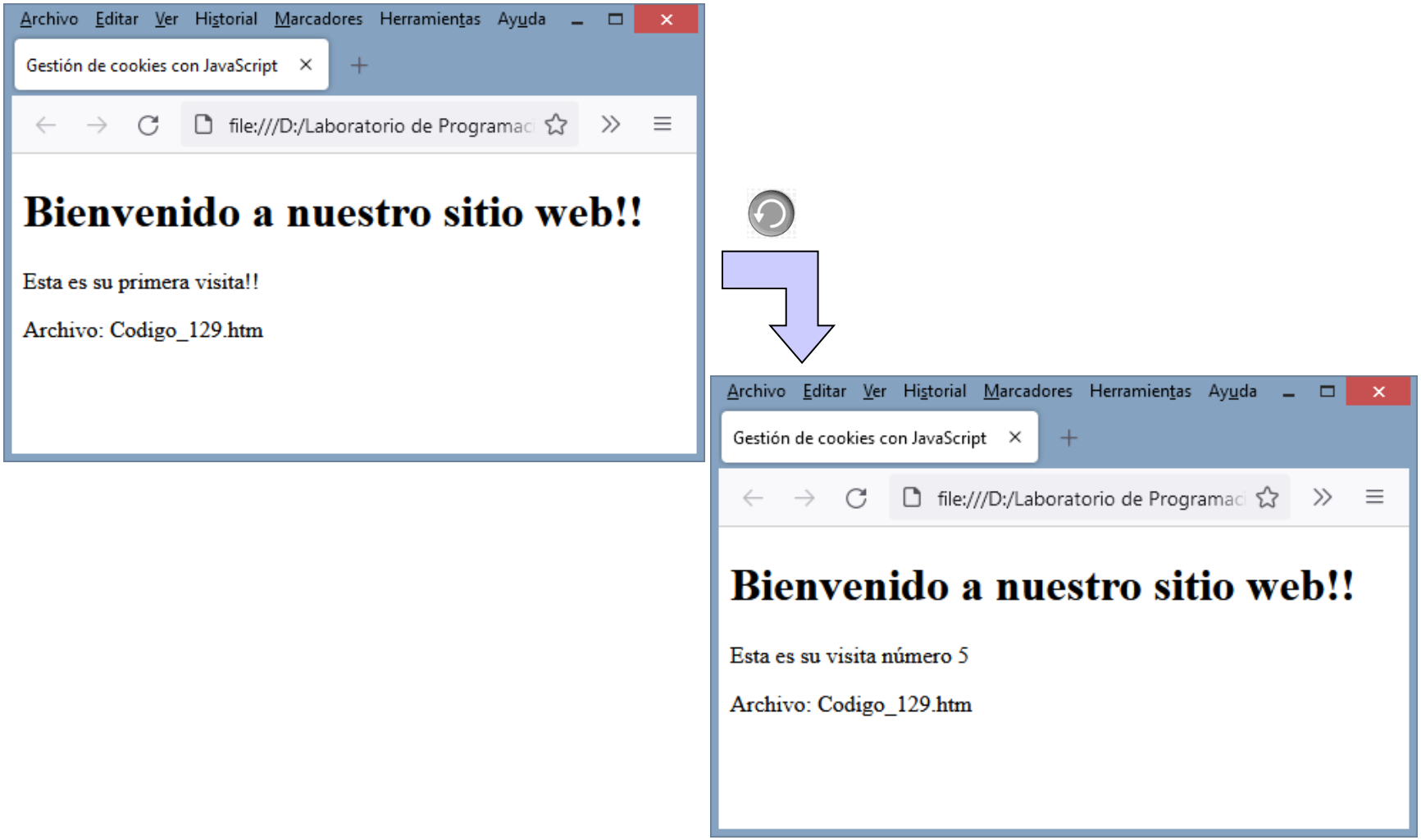
Cookies

Ejemplo

```
<!-- Codigo_129.htm -->
<!DOCTYPE html>
<html lang="es">
<head>
<title>Gestión de cookies con JavaScript</title>
<script type="text/javascript" src="Codigo_129.js">
</script>
</head>
<body onLoad="chequeoVisita();">
<header>
    <h1>Bienvenido a nuestro sitio web!!</h1>
</header>
<section>
    <article>
    <p id="visitasSite"></p>
    </article>
</section>
<footer>
    <p>Archivo: Codigo_129.htm</p>
</footer>
</body>
</html>
```

```
<!-- Codigo_129.js -->
function chequeoVisita()
{
    var nomCookie = "visitas";
    var vigencia = 60;
    //Verifico si la cookie está guardada
    var posCookie = document.cookie.search(nomCookie);
    if (posCookie == -1)
    {   //No hay cookies guardadas. Es la primera visita
        //Creo la cookie
        document.cookie = nomCookie+"=1; max-age="+vigencia;
        //Muestro la bienvenida
        document.getElementById("visitasSite").innerHTML = "Esta es su primera visita!!";
    }
    else
    {   //Hay cookies guardadas. Actualizo el contador
        //Accedo a la cookie para obtener el valor
        var posIgual = document.cookie.indexOf("=",posCookie);
        var contador = parseInt(document.cookie.substring(posIgual+1))+1;
        document.cookie = nomCookie+"="+contador+"; max-age="+vigencia;
        //Muestro el mensaje
        document.getElementById("visitasSite").innerHTML = "Esta es su visita n&uacute;mero "+contador;
    }
}
```

Cookies


Archivo Editar Ver Historial Marcadores Herramientas Ayuda
Gestión de cookies con JavaScript
file:///D:/Laboratorio de Programac
Bienvenido a nuestro sitio web!!
Esta es su primera visita!!
Archivo: Codigo_129.htm
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
Gestión de cookies con JavaScript
file:///D:/Laboratorio de Programac
Bienvenido a nuestro sitio web!!
Esta es su visita número 5
Archivo: Codigo_129.htm

Almacenamiento interno

Introducción

- Las cookies tienen la limitación de que pueden almacenar solamente 4 KB de información.
- Para guardar más información en el cliente, se puede utilizar *Web storage*(*), que permite realizar:
  - Almacenamiento local → objeto *localStorage*
  - Almacenamiento de sesión → objeto *sessionStorage*
- *Web Storage* apareció con HTML5 y es aceptado por muchos navegadores. Algunas ventajas:
  - Permite almacenar hasta 5 MB de información.
  - Se almacena localmente y no necesita transferencia HTTP

(*) también llamado objeto *Storage*

Almacenamiento interno

localStorage vs sessionStorage

- Ambas interfaces utilizan los mismos métodos.
- Los datos se guardan en el navegador, por lo que el acceso a ellos se logra solamente desde el navegador que los guardó. Sólo se guardan strings.
- La persistencia de los datos depende la interfaz:
  - localStorage → a priori no hay límite temporal y los datos se conservan aún cuando la ventana (o la pestaña) del navegador se cierre.
  - sessionStorage → los datos persisten solo mientras dure la sesión del usuario.
- Los datos no se guardan cifrados → problemas de seguridad.

Almacenamiento interno

Objeto localStorage

- La propiedad más importante de LocalStorage es *lenght* → cantidad de elementos almacenados
- Los métodos de LocalStorage son:

Método	Descripción
setItem	Se utiliza para almacenar un dato. Tiene como parámetros el nombre con el que se almacenará (clave/key) y el valor a guardar.
getItem	Se utiliza para recuperar un dato guardado. Tiene como parámetro el nombre con el que se almacenó el dato (clave/key).
removeItem	Se utiliza para eliminar un dato específico del almacenamiento local. Tiene como parámetro el nombre con el que se almacenó el dato (clave/key).
clear	Se utiliza para eliminar todos los datos almacenados localmente. No tiene parámetros.
key	Se utiliza para obtener la clave/key en una posición determinada que se pasa como parámetro.

Almacenamiento interno

Ejemplo

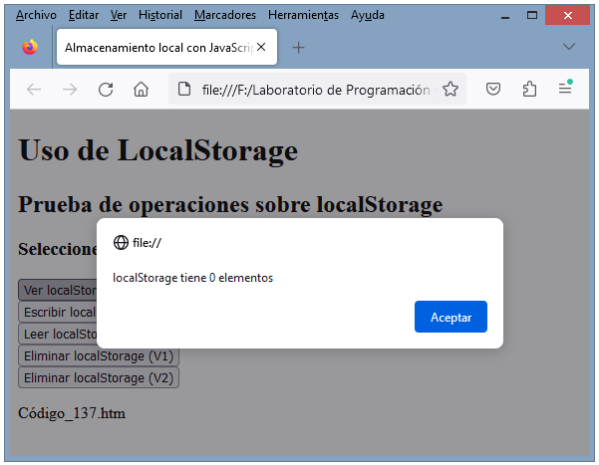

```
<!-- Codigo_137.htm -->
<!DOCTYPE html>
<html lang="es">
<head>
<title>Almacenamiento local con JavaScript</title>
<script type="text/javascript" src="Codigo_137.js">
</script>
</head>

<body>
<header>
    <h1>Uso de LocalStorage</h1>
</header>
<section>
    <h2>Prueba de operaciones sobre localStorage</h2>
    <article>
        <h3>Seleccione una operación</h3>
        <form id="idFrmLocalStorage" name="nomFrmLocalStorage">
        <input  type="button" id="btnVerCantElementos" value="Ver localStorage"
                onClick="verCantElemLocalStorage();"><br>
        <input  type="button" id="btnEscribir" value="Escribir localStorage"
                onClick="escribirLocalStorage();leerLocalStorage();"><br>
        <input  type="button" id="btnLeer" value="Leer localStorage"
                onClick="leerLocalStorage();"><br>
        <input  type="button" id="btnEliminarV1" value="Eliminar localStorage (V1)"
                onClick="eliminarLocalStorageV1();leerLocalStorage();"><br>
        <input  type="button" id="btnEliminarV1" value="Eliminar localStorage (V2)"
                onClick="eliminarLocalStorageV2();leerLocalStorage();"><br>
        </form>
        <p>
        <div id="divTitulo"></div>
        <div id="divNombrePersona"></div>
        <div id="divApellidoPersona"></div>
        <div id="divCorreoPersona"></div>
        </p>
    </article>
</section>
<footer><p>Código_137.htm</p></footer>
</body>
</html>
```

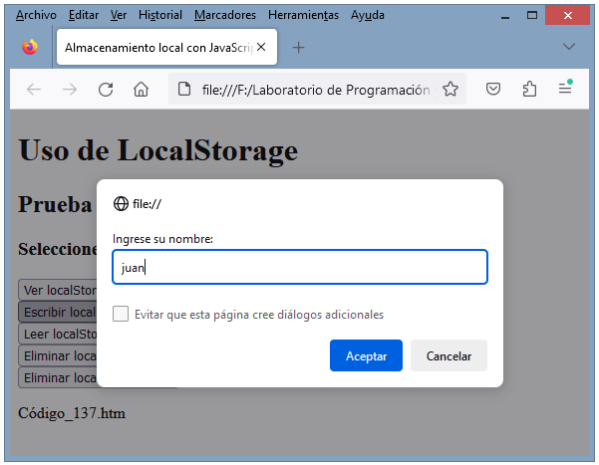
Almacenamiento interno

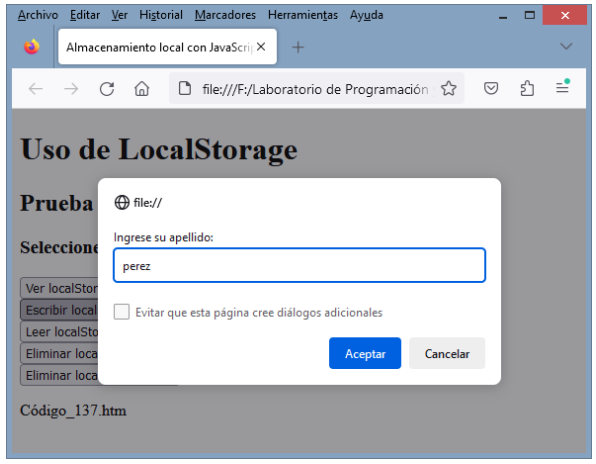
```
<!-- Codigo_137.js -->

var mensajeError = "localStorage no es soportado por este navegador web";

function verCantElemLocalStorage()
{
    if (typeof localStorage != "undefined")
    {   alert("localStorage tiene "+localStorage.length+" elementos");  }
    else
    {   alert(mensajeError);    }
}

function imprimirTitulo(titulo)
{
    document.getElementById("divTitulo").innerHTML = titulo;
}

function imprimirElemento(item, lugar)
{
    var aux = "";
    if (!localStorage.getItem(item))
        {   aux = "Sin valor guardado"; }
    else
        {   aux = localStorage.getItem(item);   }
    document.getElementById(lugar).innerHTML = aux;
}

function escribirLocalStorage()
{
    var nombrePersona = "";
    var apellidoPersona = "";
    var emailPersona = "";

    if (typeof localStorage != "undefined")
    {
        nombrePersona = window.prompt("Ingrese su nombre:");
        localStorage.setItem("nombrePersona",nombrePersona);
        apellidoPersona = window.prompt("Ingrese su apellido:");
        localStorage.setItem("apellidoPersona",apellidoPersona);
        emailPersona = window.prompt("Ingrese su correo electrónico:");
        localStorage.setItem("correoPersona",emailPersona);
    }
    else
    {   alert(mensajeError);    }
}
```

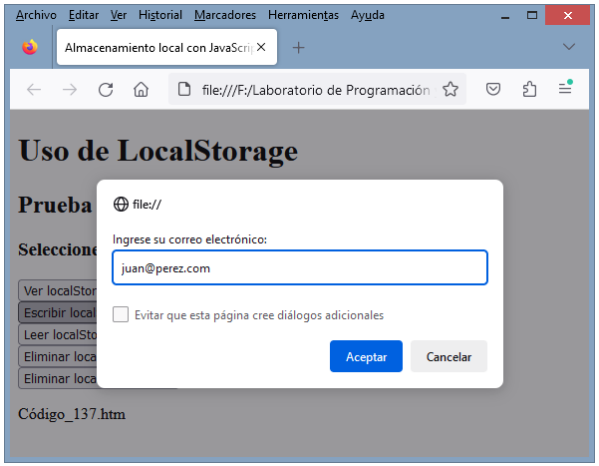
```

function leerLocalStorage()
{
    if (typeof localStorage != "undefined")
    {
        imprimirTitulo("Contenido del localStorage:");
        imprimirElemento("nombrePersona","divNombrePersona");
        imprimirElemento("apellidoPersona","divApellidoPersona");
        imprimirElemento("correoPersona","divCorreoPersona");
    }
    else
    {   alert(mensajeError);    }
}

function eliminarLocalStorageV1()
{
    if (typeof localStorage != "undefined")
    {
        localStorage.removeItem("nombrePersona");
        localStorage.removeItem("apellidoPersona");
        localStorage.removeItem("correoPersona");
    }
    else
    {   alert(mensajeError);    }
}

function eliminarLocalStorageV2()
{
    if (typeof localStorage != "undefined")
    {
        localStorage.clear();
    }
    else
    {   alert(mensajeError);    }
}
```

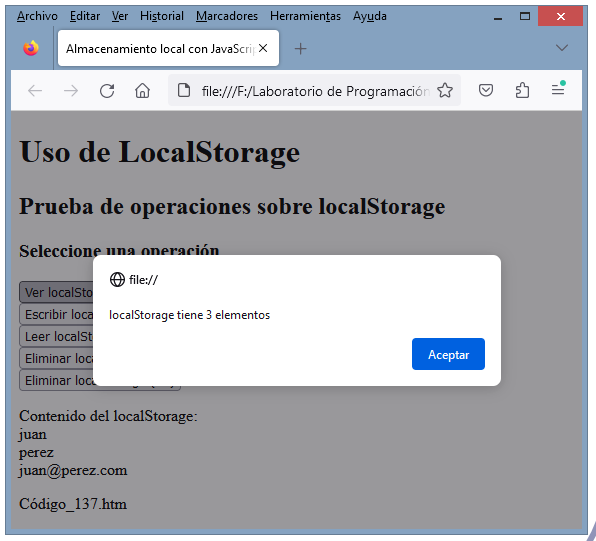
Almacenamiento interno











Aplicación: ubicación geográfica

Geolocalización

- HTML5 incorpora una API para funciones de geolocalización.
- Esto permite al navegador, previo soporte y autorización, determinar una ubicación aproximada del usuario.
  - Combinado con servicios web de representación de mapas (Google Maps), es posible mostrar al usuario su ubicación.
- Para esto, se puede usar el objeto *navigator.geolocation*

Aplicación: ubicación geográfica

- El objeto *navigator.geolocation* tiene un método llamado *getCurrentPosition*, que tiene los siguientes parámetros:
  - Función éxito: indica la función que debe ejecutarse si fue posible determinar la ubicación del cliente
  - Función de error: indica la función que debe ejecutarse en caso de error en la determinación de la ubicación.
  - Otros parámetros opcionales son: *enableHighAccuracy* y *Timeout*
- Finalmente, la determinación de la latitud y longitud de logra con el objeto *position*.

Aplicación: ubicación geográfica

- El objeto *position* tiene, a su vez, otro objeto llamado *coords*, que tiene los atributos *longitude* y *latitude*, con los que es posible determinar longitud y latitud respectivamente.

Aplicación: ubicación geográfica

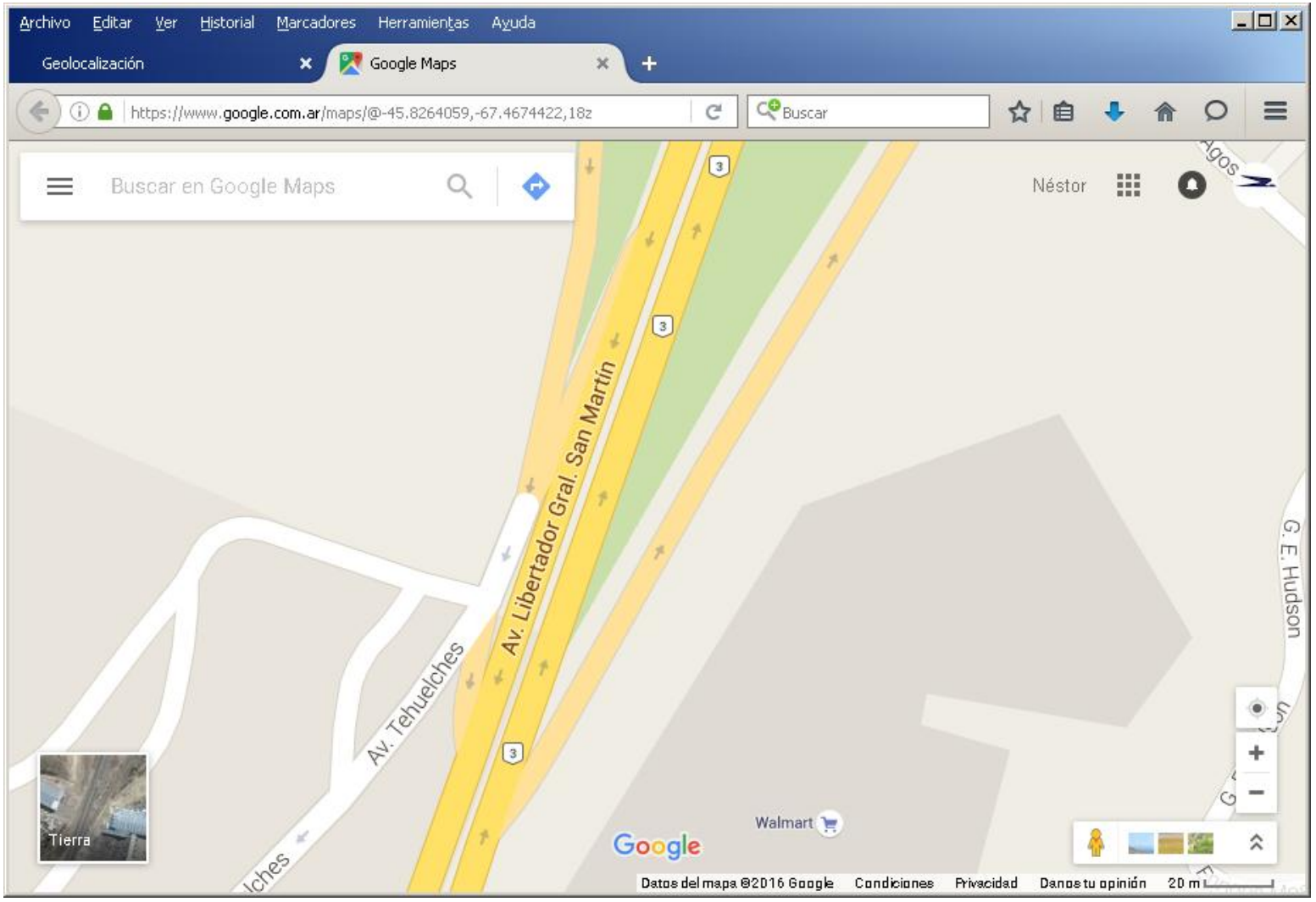
Ejemplo

```
<!-- Codigo_110.js -->
function cargaLocalizacion()
{
    navigator.geolocation.getCurrentPosition(obtengoCoordenadas, MuestraError, {timeout:10000});
}

function obtengoCoordenadas(posicion)
{
    var longitud = posicion.coords.longitude;
    var latitud = posicion.coords.latitude;
    var enlace = "http://maps.google.com.ar/?ll=" + latitud  + "," + longitud + "&z=18";
    document.getElementById("long").innerHTML = "Longitud: " + longitud ;
    document.getElementById("lat").innerHTML = "Latitud: " + latitud;
    document.getElementById("enlace").href= enlace;
}

function MuestraError(error)
{
    alert(error.code);
}
```

```
<!-- Codigo_110.htm -->
<!DOCTYPE html>
<html lang="es">
<head>
<title>Geolocalizaci&oacute;n</title>
<script type="text/javascript" src="Codigo_110.js">
</script>
</head>
<body onLoad="cargaLocalizacion();">
<header>
    <h1>Ejemplo de geolocalizaci&oacute;n</h1>
</header>
<section>
    <article>
    <p id="long"></p>
    <p id="lat"></p>
    <a id="enlace" target="_blank">Enlace de mapa</a>
    </article>
</section>
<footer>
    <p>Archivo: Codigo_110.htm</p>
</footer>
</body>
</html>
```

Aplicación: ubicación geográfica

Aplicación: ubicación geográfica



Aplicación: Google Maps

Ejemplo

```
1  <!-- Codigo_127.htm -->
2  <!DOCTYPE html>
3  <html lang="es">
4  <head>
5  <meta name="viewport" content="initial-scale=1.0, user-scalable=yes"/>
6  <title>Geolocalizaci&oacute;n</title>
7  <script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCR-9VEi1z8uJakLAr-xTE7H5hYIYVXLWQ"></script>
8  <script type="text/javascript" src="Codigo_127.js"></script>
9  </head>
10 <body onLoad="cargoMapa();">
11 <header>
12     <h1>Ejemplo de geolocalizaci&oacute;n</h1>
13 </header>
14 <section>
15     <article>
16         <p id="long"></p>
17         <p id="lat"></p>
18         <div id="miMapa" style="width:100%; height:100%"></div>
19     </article>
20 </section>
21 <footer>
22     <p>Archivo: Codigo_127.htm</p>
23 </footer>
24 </body>
25 </html>
```

Aplicación: Google Maps

```
<!-- Codigo_127.js -->
var longitud, latitud;

function cargoMapa()
{
    var opciones = {
        enableHighAccuracy: true,
        timeout: 10000
    };

    if (!navigator.geolocation)
    {   alert("Su navegador no soporta la geolocalización");    }
    else
    {
        navigator.geolocation.getCurrentPosition(obtengoCoordenadas, muestraError, opciones);
        generoMapa();
    }

    function generoMapa()
    {
        //var centroMapa = new google.maps.LatLng(latitud,longitud);
        var centroMapa = new google.maps.LatLng(-45.8733,-67.5088);
        var opcionesGoogleMap =
        {
            zoom: 8,
            center: centroMapa,
            mapTypeId: 'roadmap'
        }
        var miMapa = new google.maps.Map(document.getElementById("miMapa"),opcionesGoogleMap);
    }
}

function obtengoCoordenadas(posicion)
{   //El parametro 'posicion', en tiempo de ejecución se genera automaticamente con un objeto position
    longitud = posicion.coords.longitude;
    latitud = posicion.coords.latitude;
    document.getElementById("long").innerHTML = "Longitud: " + longitud;
    document.getElementById("lat").innerHTML = "Latitud: " + latitud;
}

function muestraError(error)
{
    alert("No se puede determinar la posición debido al error: "+error.code);
}
```

Aplicación: Google Maps