JavaScript HTML DOM

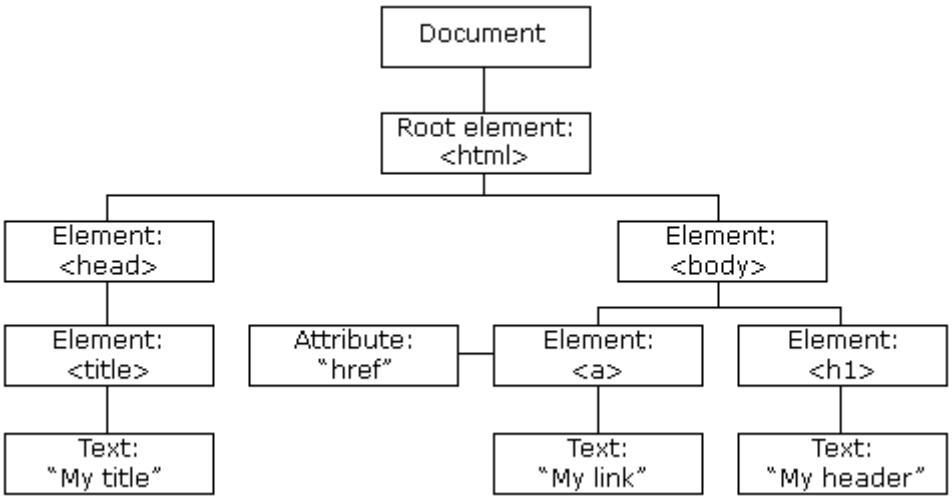
Con HTML DOM, JavaScript puede acceder y cambiar todos los elementos de un documento HTML.

El HTML DOM (Modelo de Objetos de Documento)

Cuando se carga una página web, el navegador crea
una D DOCUMENTO O bject M odelo de la página.

El modelo HTML DOM está construido como un árbol de Objetos :

El árbol de objetos HTML DOM



Con el modelo de objetos, JavaScript obtiene toda la potencia que necesita para crear HTML dinámico:

- JavaScript puede cambiar todos los elementos HTML en la página
- JavaScript puede cambiar todos los atributos HTML en la página
- JavaScript puede cambiar todos los estilos CSS en la página
- JavaScript puede eliminar elementos y atributos HTML existentes
- JavaScript puede agregar nuevos elementos y elementos HTML
- JavaScript puede reaccionar a todos los eventos HTML existentes en la página
- JavaScript puede crear nuevos eventos HTML en la página

¿Cuál es el DOM?

El DOM es un estándar W3C (World Wide Web Consortium).

El DOM define un estándar para acceder a los documentos:

"*El Modelo de Objetos de Documento (DOM) del W3C es una plataforma y una interfaz de lenguaje neutral que permite que los programas y scripts accedan y actualicen dinámicamente el contenido, la estructura y el estilo de un documento*".

El estándar W3C DOM está dividido en 3 partes diferentes:

- Core DOM: modelo estándar para todos los tipos de documentos
- XML DOM - modelo estándar para documentos XML
- HTML DOM - modelo estándar para documentos HTML

¿Cuál es el HTML DOM?

El HTML DOM es un modelo de objeto estándar y una interfaz de programación para HTML. Se define:

- Los elementos HTML como **objetos**
- Las **propiedades** de todos los elementos HTML
- Los **métodos** para acceder a todos los elementos HTML
- Los eventos para todos los elementos HTML

En otras palabras: **el HTML DOM es un estándar para obtener, cambiar, agregar o eliminar elementos HTML.**

JavaScript - Métodos DOM de HTML

Los métodos HTML DOM son acciones que puede realizar (en Elementos HTML).

Las propiedades HTML DOM son valores (de elementos HTML) que puede establecer o cambiar.

La interfaz de programación DOM

Se puede acceder al HTML DOM con JavaScript (y con otros lenguajes de programación).

En el DOM, todos los elementos HTML se definen como objetos .

La interfaz de programación es las propiedades y métodos de cada objeto.

Una **propiedad** es un valor que puede obtener o establecer (como cambiar el contenido de un elemento HTML).

Un **método** es una acción que puede hacer (como agregar o eliminar un elemento HTML).

Ejemplo

El siguiente ejemplo cambia el contenido (el innerHTML) del elemento <p> con id = "demo":

Ejemplo

```
<  html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

Inténtalo tú mismo "

En el ejemplo anterior, getElementById es un método , mientras que innerHTML es una propiedad .

El método getElementById

La forma más común de acceder a un elemento HTML es usar la identificación del elemento.

En el ejemplo anterior, el método getElementById utilizó id = "demo" para encontrar el elemento.

La propiedad innerHTML

La forma más fácil de obtener el contenido de un elemento es mediante el uso de la propiedad **innerHTML** .

La propiedad innerHTML es útil para obtener o reemplazar el contenido de elementos HTML.

> La propiedad innerHTML se puede usar para obtener o cambiar cualquier elemento HTML, incluidos <html> y <body>.

Documento DOM HTML de JavaScript

El objeto de documento HTML DOM es el propietario de todos los demás objetos en su página web.

Objeto de documento HTML DOM

El objeto del documento representa su página web.

Si desea acceder a cualquier elemento en una página HTML, siempre comienza con el acceso al objeto del documento.

A continuación se muestran algunos ejemplos de cómo puede usar el objeto de documento para acceder y manipular HTML.

Encontrar elementos HTML

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Cambiar elementos HTML

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element

Agregar y eliminar elementos

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Agregar controladores de eventos

Method	Description
document.getElementById(*id*).onclick = function() {*code*}	Adding event handler code to an onclick event

Encontrar objetos HTML

El primer HTML DOM Nivel 1 (1998) definió 11 objetos HTML, colecciones de objetos y propiedades. Estos siguen siendo válidos en HTML5.

Más tarde, en HTML DOM Nivel 3, se agregaron más objetos, colecciones y propiedades.

Property	Description	DOM
document.anchors	Returns all <a> elements that have a name attribute	1
document.applets	Returns all <applet> elements (Deprecated in HTML5)	1
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3
document.documentElement	Returns the <html> element	3
document.documentMode	Returns the mode used by the browser	3
document.documentURI	Returns the URI of the document	3
document.domain	Returns the domain name of the document server	1
document.domConfig	Obsolete. Returns the DOM configuration	3
document.embeds	Returns all <embed> elements	3
document.forms	Returns all <form> elements	1
document.head	Returns the <head> element	3
document.images	Returns all <img> elements	1
document.implementation	Returns the DOM implementation	3
document.inputEncoding	Returns the document's encoding (character set)	3
document.lastModified	Returns the date and time the document was updated	3
document.links	Returns all <area> and <a> elements that have a href attribute	1
document.readyState	Returns the (loading) status of the document	3
document.referrer	Returns the URI of the referrer (the linking document)	1
document.scripts	Returns all <script> elements	3
document.strictErrorChecking	Returns if error checking is enforced	3
document.title	Returns the <title> element	1
document.URL	Returns the complete URL of the document	1

Elementos de DOM HTML de JavaScript

Esta página le enseña cómo encontrar y acceder a los elementos HTML en una página HTML.

Encontrar elementos HTML

A menudo, con JavaScript, quiere manipular elementos HTML.

Para hacerlo, primero debes encontrar los elementos. Hay un par de formas de hacer esto:

- Encontrar elementos HTML por id
- Encontrar elementos HTML por nombre de etiqueta
- Encontrar elementos HTML por nombre de clase
- Encontrar elementos HTML por selectores de CSS
- Encontrar elementos HTML por colecciones de objetos HTML

Encontrar el elemento HTML por Id

La forma más fácil de encontrar un elemento HTML en el DOM, es mediante el uso de la identificación del elemento.

Este ejemplo encuentra el elemento con id = "intro":

Ejemplo

```
var myElement = document.getElementById("intro");
```

Inténtalo tú mismo "

Si se encuentra el elemento, el método devolverá el elemento como un objeto (en myElement).

Si no se encuentra el elemento, myElement contendrá nulo.

Encontrar elementos HTML por nombre de etiqueta

Este ejemplo encuentra todos los elementos <p>:

Ejemplo

```
var x = document.getElementsByTagName("p");
```

Inténtalo tú mismo "

Este ejemplo encuentra el elemento con id = "main", y luego encuentra todos los elementos <p> dentro de "main":

Ejemplo

```
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
```

Inténtalo tú mismo "

Encontrar elementos HTML por nombre de clase

Si desea buscar todos los elementos HTML con el mismo nombre de clase, use getElementsByClassName ().

Este ejemplo devuelve una lista de todos los elementos con class = "intro".

Ejemplo

```
var x = document.getElementsByClassName("intro");
```

Inténtalo tú mismo "

Encontrar elementos por nombre de clase no funciona en Internet Explorer 8 y versiones anteriores.

Encontrar elementos HTML por selectores de CSS

Si desea buscar todos los elementos HTML que coincidan con un selector CSS específico (id, nombres de clase, tipos, atributos, valores de atributos, etc.), utilice el método querySelectorAll ().

Este ejemplo devuelve una lista de todos los elementos <p> con class = "intro".

Ejemplo

```
var x = document.querySelectorAll("p.intro");
```

Inténtalo tú mismo "

El método querySelectorAll () no funciona en Internet Explorer 8 y versiones anteriores.

Encontrar elementos HTML por colecciones de objetos HTML

Este ejemplo encuentra el elemento de formulario con id = "frm1", en la colección de formularios, y muestra todos los valores de los elementos:

Ejemplo

```
var x = document.forms["frm1"];
var text = "";
var i;
for (i = 0; i < x.length; i++) {
    text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

Inténtalo tú mismo "

Los siguientes objetos HTML (y colecciones de objetos) también son accesibles:

- document.anchors
- document.body
- document.documentElement
- document.embeds
- document.forms
- document.head
- document.images
- document.links
- document.scripts
- titulo del documento

Ponte a prueba con ejercicios!

Ejercicio 1 » Ejercicio 2» Ejercicio 3 » Ejercicio 4» Ejercicio 5 »

JavaScript HTML DOM - Cambio de HTML

El HTML DOM permite a JavaScript cambiar el contenido de los elementos HTML.

Cambiar el flujo de salida HTML

JavaScript puede crear contenido HTML dinámico:

Fecha: Lun 04 Dic 2017 21:04:13 GMT + 0100 (CET)

En JavaScript, document.write () se puede utilizar para escribir directamente en la secuencia de salida HTML:

Ejemplo

```
<! !DOCTYPE html>
<html>
<body>

<script>
document.write(Date());
</script>

</body>
</html>
```

Inténtalo tú mismo "

Nunca use document.write () después de cargar el documento. Sobrescribirá el documento.

Cambio de contenido HTML

La forma más fácil de modificar el contenido de un elemento HTML es mediante el uso de la propiedad **innerHTML** .

Para cambiar el contenido de un elemento HTML, use esta sintaxis:

document.getElementById(	*id*).innerHTML = *new HTML*

Este ejemplo cambia el contenido de un elemento <p>:

Ejemplo

```
<!DOCTYPE html>
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New text!";
</script>

</body>
</html>
```

Inténtalo tú mismo "

Ejemplo explicado:

- El documento HTML de arriba contiene un elemento <p> con id = "p1"
- Usamos el HTML DOM para obtener el elemento con id = "p1"
- Un JavaScript cambia el contenido (innerHTML) de ese elemento a "¡Nuevo texto!"

Este ejemplo cambia el contenido de un elemento <h1>:

Ejemplo

```
<!DOCTYPE html>
<html>
<body>

<h1 id="id01">Old Heading</h1>

<script>
var element = document.getElementById("id01");
element.innerHTML = "New Heading";
</script>

</body>
</html>
```

Inténtalo tú mismo "

Ejemplo explicado:

- El documento HTML de arriba contiene un elemento <h1> con id = "id01"
- Usamos el HTML DOM para obtener el elemento con id = "id01"
- Un JavaScript cambia el contenido (innerHTML) de ese elemento a "Nuevo encabezado"

Cambiar el valor de un atributo

Para cambiar el valor de un atributo HTML, use esta sintaxis:

document.getElementById(	*id*).*attribute* = *new value*

Este ejemplo cambia el valor del atributo src de un elemento <img>:

Ejemplo

```
<!DOCTYPE html>
<html>
<body>

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```

Inténtalo tú mismo "

Ejemplo explicado:

- El documento HTML anterior contiene un elemento <img> con id = "myImage"
- Usamos el HTML DOM para obtener el elemento con id = "myImage"
- Un JavaScript cambia el atributo src de ese elemento de "smiley.gif" a "landscape.jpg"

Ponte a prueba con ejercicios!

Ejercicio 1 » Ejercicio 2» Ejercicio 3 » Ejercicio 4» Ejercicio 5 »

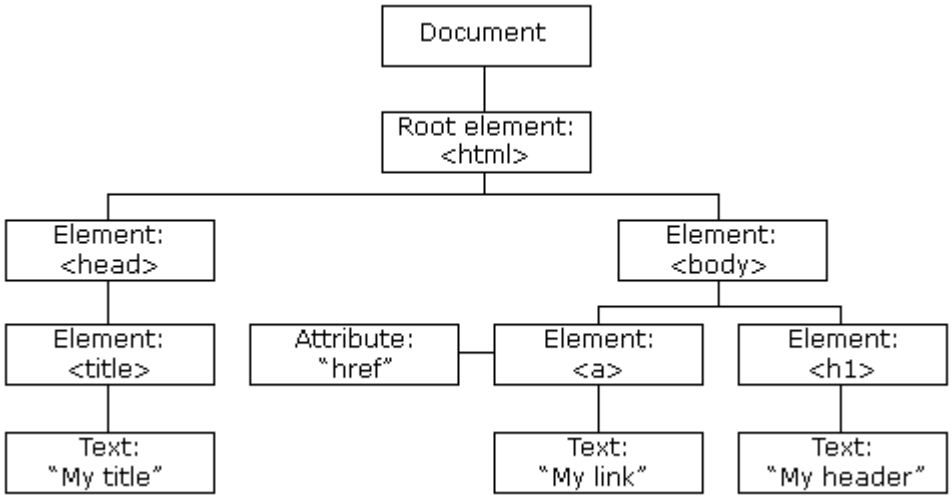
JavaScript HTML DOM Navigation

Con el HTML DOM, puede navegar por el árbol de nodos usando relaciones de nodo.

Nodos DOM

De acuerdo con el estándar W3C HTML DOM, todo en un documento HTML es un nodo:

- El documento completo es un nodo de documento
- Cada elemento HTML es un nodo de elemento
- El texto dentro de los elementos HTML son nodos de texto
- Cada atributo HTML es un nodo de atributo (obsoleto)
- Todos los comentarios son nodos de comentarios



Con el HTML DOM, se puede acceder a todos los nodos en el árbol de nodos mediante JavaScript.

Se pueden crear nuevos nodos y todos los nodos se pueden modificar o eliminar.

Relaciones de nodo

Los nodos en el árbol de nodos tienen una relación jerárquica entre ellos.

Los términos padre, hijo y hermano se usan para describir las relaciones.

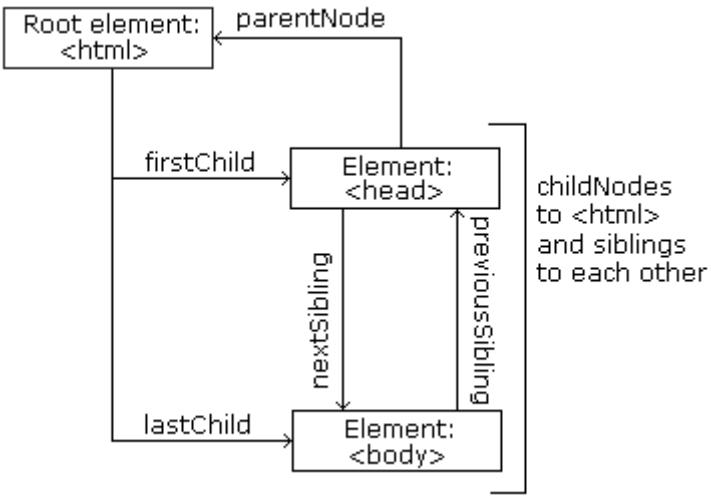
- En un árbol de nodos, el nodo superior se llama raíz (o nodo raíz)
- Cada nodo tiene exactamente un padre, excepto la raíz (que no tiene padre)
- Un nodo puede tener varios hijos
- Los hermanos (hermanos o hermanas) son nodos con el mismo padre

```
<html>

 <head>
   <title>DOM Tutorial</title>
 </head>

 <body>
   <h1>DOM Lesson one</h1>
   <p>Hello world!</p>
 </body>

</html>
```



Desde el HTML anterior puedes leer:

- <html> es el nodo raíz
- <html> no tiene padres
- <html> es el padre de <head> y <body>
- <head> es el primer hijo de <html>
- <body> es el último hijo de <html>

y:

- <head> tiene un hijo: <title>
- <title> tiene un hijo (un nodo de texto): "DOM Tutorial"
- <cuerpo> tiene dos hijos: <h1> y <p>
- <h1> tiene un hijo: "DOM Lesson one"
- <p> tiene un hijo: "¡Hola mundo!"
- <h1> y <p> son hermanos

Navegando entre nodos

Puede usar las siguientes propiedades de nodo para navegar entre nodos con JavaScript:

- parentNode
- childNodes [nodenumber]
- primer hijo
- último niño
- proximo hermano
- anteriorSibling

Nodos secundarios y valores de nodo

Un error común en el procesamiento DOM es esperar que un nodo elemento contenga texto.

Ejemplo:

```
<title id="demo">DOM Tutorial</title>
```

El elemento nodo <título> (en el ejemplo anterior) no contiene texto.

Contiene un **nodo de texto** con el valor "DOM Tutorial".

El valor del nodo de texto se puede acceder mediante la propiedad **innerHTML** del nodo :

```
var myTitle = document.getElementById("demo").innerHTML;
```

Acceder a la propiedad innerHTML es lo mismo que acceder al nodeValue del primer hijo:

```
var myTitle = document.getElementById("demo").firstChild.nodeValue;
```

El acceso al primer hijo también se puede hacer así:

```
var myTitle =
document.getElementById("demo").childNodes[0].nodeValue;
```

Todos los (3) siguientes ejemplos recuperan el texto de un elemento <h1> y lo copian en un elemento <p>:

Ejemplo

```
<! html>
<body>

<h1 id="id01">My First Page</h1>
<p id="id02"></p>

<script>
document.getElementById("id02").innerHTML =
document.getElementById("id01").innerHTML;
</script>

</body>
</html>
```

Inténtalo tú mismo "

Ejemplo

```
<! html>
<body>

<h1 id="id01">My First Page</h1>
<p id="id02"></p>

<script>
document.getElementById("id02").innerHTML =
document.getElementById("id01").firstChild.nodeValue;
</script>

</body>
</html>
```

Inténtalo tú mismo "

Ejemplo

```
<! html>
<body>

<h1 id="id01">My First Page</h1>
<p id="id02">Hello!</p>

<script>
document.getElementById("id02").innerHTML =
document.getElementById("id01").childNodes[0].nodeValue;
</script>

</body>
</html>
```

Inténtalo tú mismo "

InnerHTML

En este tutorial utilizamos la propiedad innerHTML para recuperar el contenido de un elemento HTML.

Sin embargo, aprender los otros métodos anteriores es útil para entender la estructura del árbol y la navegación del DOM.

Nodos raíz DOM

Hay dos propiedades especiales que permiten el acceso al documento completo:

- document.body - El cuerpo del documento
- document.documentElement - El documento completo

Ejemplo

```
<! html>
<body>

<p>Hello World!</p>
<div>
```

```
  <p>The DOM is very useful!</p>
<p>This example demonstrates the <b>document.body</b> property.</p>
</div>

<script>
  alert(document.body.innerHTML);
  </script>

  </body>
  </html>
```

Inténtalo tú mismo "

Ejemplo

```
  <html>
  <body>

  <p>Hello World!</p>
  <div>
  <p>The DOM is very useful!</p>
  <p>This example demonstrates
  the <b>document.documentElement</b> property.</p>
  </div>

  <script>
  alert(document.documentElement.innerHTML);
  </script>

  </body>
  </html>
```

Inténtalo tú mismo "

La propiedad nodeName

La propiedad nodeName especifica el nombre de un nodo.

- nodeName es de solo lectura
- nodeName de un nodo de elemento es el mismo que el nombre de etiqueta
- nodeName de un nodo de atributo es el nombre del atributo
- nodeName de un nodo de texto siempre es #text
- nodeName del nodo del documento siempre es #document

Ejemplo

```
<  h1 id="id01">My First Page</h1>
<p id="id02"></p>

<script>
document.getElementById("id02").innerHTML =
document.getElementById("id01").nodeName;
</script>
```

Inténtalo tú mismo "

Nota: nodeName siempre contiene el nombre de la etiqueta en mayúsculas de un elemento HTML.

La propiedad nodeValue

La propiedad nodeValue especifica el valor de un nodo.

- nodeValue para nodos de elemento no está definido
- nodeValue para los nodos de texto es el texto mismo
- nodeValue para los nodos de atributo es el valor del atributo

---

La propiedad nodeType

La propiedad nodeType es de solo lectura. Devuelve el tipo de un nodo.

Ejemplo

```
<  h1 id="id01">My First Page</h1>
<p id="id02"></p>

<script>
document.getElementById("id02").innerHTML =
document.getElementById("id01").nodeType;
</script>
```

Inténtalo tú mismo "

Las propiedades nodeType más importantes son:

Node	Type	Example

ELEMENT_NODE	1	<h1 class="heading">W3Schools</h1>
ATTRIBUTE_NODE	2	class = "heading" (deprecated)
TEXT_NODE	3	W3Schools
COMMENT_NODE	8	<!-- This is a comment -->
DOCUMENT_NODE	9	The HTML document itself (the parent of <html>)
DOCUMENT_TYPE_NODE	10	<!Doctype html>

El tipo 2 está en desuso en el HTML DOM (pero funciona). No está desaprobado en el XML DOM.

JavaScript HTML DOM elementos (nodos)

Agregar y eliminar nodos (elementos HTML)

Crear nuevos elementos HTML (nodos)

Para agregar un nuevo elemento al HTML DOM, primero debe crear el elemento (nodo del elemento) y luego añádalo a un elemento existente.

Ejemplo

```
<  div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var element = document.getElementById("div1");
element.appendChild(para);
</script>
```

Inténtalo tú mismo "

Ejemplo explicado

Este código crea un nuevo elemento <p>:

```
var para = document.createElement("p");
```

Para agregar texto al elemento <p>, primero debe crear un nodo de texto. Este código crea un nodo de texto:

```
var node = document.createTextNode("This is a new paragraph.");
```

Luego debe agregar el nodo de texto al elemento <p>:

```
para.appendChild(node);
```

Finalmente debe agregar el nuevo elemento a un elemento existente.

Este código encuentra un elemento existente:

```
var element = document.getElementById("div1");
```

Este código agrega el nuevo elemento al elemento existente:

```
element.appendChild(para);
```

Crear nuevos elementos HTML - insertBefore ()

El método appendChild () en el ejemplo anterior, agregó el nuevo elemento como el último elemento secundario del elemento primario.

Si no quiere, puede usar el método insertBefore ():

Ejemplo

```
<  div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var element = document.getElementById("div1");
var child = document.getElementById("p1");
element.insertBefore(para, child);
</script>
```

Inténtalo tú mismo "

Eliminar elementos HTML existentes

Para eliminar un elemento HTML, debe conocer el elemento primario del elemento:

Ejemplo

```
<  div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
```

```
parent.removeChild(child);
</script>
```

Inténtalo tú mismo "

El método node.remove () se implementa en la especificación DOM 4.
Pero debido al soporte deficiente del navegador, no deberías usarlo.

Ejemplo explicado

Este documento HTML contiene un elemento con dos nodos secundarios (dos elementos <p>):

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>
```

Encuentra el elemento con id = "div1":

```
var parent = document.getElementById("div1");
```

Encuentre el elemento <p> con id = "p1":

```
var child = document.getElementById("p1");
```

Eliminar al hijo del padre:

```
parent.removeChild(child);
```

Sería bueno poder eliminar un elemento sin consultar al padre.
Pero lo siento. El DOM necesita saber tanto el elemento que desea eliminar como su elemento primario.

Aquí hay una solución común: encuentre el elemento secundario que desea eliminar y use su propiedad parentNode para encontrar el elemento primario:

```
var child = document.getElementById("p1");
```

```
child.parentNode.removeChild(child);
```

Reemplazar elementos HTML

Para reemplazar un elemento al HTML DOM, use el método replaceChild ():

Ejemplo

```
<  div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.replaceChild(para, child);
</script>
```

Inténtalo tú mismo "

JavaScript H TML DOM Colecciones

El objeto HTMLCollection

El método getElementsByTagName () devuelve un objeto **HTMLCollection** .

Un objeto HTMLCollection es una lista (colección) similar a una matriz de elementos HTML.

El siguiente código selecciona todos los elementos <p> en un documento:

Ejemplo

```
var x = document.getElementsByTagName("p");
```

Se puede acceder a los elementos en la colección por un número de índice.

Para acceder al segundo elemento <p> puede escribir:

```
y = x[1];
```

Inténtalo tú mismo "

Nota: El índice comienza en 0.

HTML HTMLCollection Length

La propiedad de longitud define la cantidad de elementos en una HTMLCollection:

Ejemplo

```
var myCollection = document.getElementsByTagName("p");
document.getElementById("demo").innerHTML = myCollection.length;
```

Inténtalo tú mismo "

Ejemplo explicado:

1. Crea una colección de todos los elementos <p>
2. Muestra la longitud de la colección

La propiedad de longitud es útil cuando desea recorrer los elementos de una colección:

Ejemplo

Cambie el color de fondo de todos los elementos <p>:

```
var   myCollection = document.getElementsByTagName("p");
var i;
for (i = 0; i < myCollection.length; i++) {
    myCollection[i].style.backgroundColor = "red";
}
```

Inténtalo tú mismo "

¡Una HTMLCollection NO es una matriz!

Una HTMLCollection puede parecerse a una matriz, pero no lo es.

Puede recorrer la lista y referirse a los elementos con un número (como una matriz).

Sin embargo, no puede usar métodos de matriz como valueOf (), pop (), push () o join () en una HTMLCollection.

Listas de nodos DOM HTML de JavaScript

El objeto HTML DOM NodeList

Un objeto NodeList es una lista (colección) de nodos extraídos de un documento.

Un objeto NodeList es casi lo mismo que un objeto HTMLCollection.

Algunos navegadores (antiguos) devuelven un objeto NodeList en lugar de una HTMLCollection para métodos como getElementsByClassName () .

Todos los navegadores devuelven un objeto NodeList para la propiedad childNodes .

La mayoría de los navegadores devuelven un objeto NodeList para el método querySelectorAll () .

El siguiente código selecciona todos los nodos <p> en un documento:

Ejemplo

```
var myNodeList = document.querySelectorAll("p");
```

Se puede acceder a los elementos en NodeList mediante un número de índice.

Para acceder al segundo nodo <p> puede escribir:

```
y = myNodeList[1];
```

Inténtalo tú mismo "

Nota: El índice comienza en 0.

Longitud de lista de nodos DOM HTML

La propiedad de longitud define la cantidad de nodos en una lista de nodos:

Ejemplo

```
var   myNodelist = document.querySelectorAll("p");
document.getElementById("demo").innerHTML = myNodelist.length;
```

Inténtalo tú mismo "

Ejemplo explicado:

1. Crea una lista de todos los elementos <p>
2. Muestra la longitud de la lista

La propiedad de longitud es útil cuando desea recorrer los nodos en una lista de nodos:

Ejemplo

Cambie el color de fondo de todos los elementos <p> en una lista de nodos:

```
var   myNodelist = document.querySelectorAll("p");
var i;
for (i = 0; i < myNodelist.length; i++) {
    myNodelist[i].style.backgroundColor = "red";
}
```

Inténtalo tú mismo "

La diferencia entre una HTMLCollection y una NodeList

Una HTMLCollection (capítulo anterior) es una colección de elementos HTML.

Un NodeList es una colección de nodos de documentos.

Una NodeList y una colección de HTML son casi lo mismo.

Tanto un objeto HTMLCollection como un objeto NodeList son una lista (colección) de objetos similar a una matriz.

Ambos tienen una propiedad de longitud que define el número de elementos en la lista (colección).

Ambos proporcionan un índice (0, 1, 2, 3, 4, ...) para acceder a cada elemento como una matriz.

Se puede acceder a los elementos HTMLCollection por su nombre, id o número de índice.

A los elementos de NodeList solo se puede acceder por su número de índice.

Solo el objeto NodeList puede contener nodos de atributo y nodos de texto.

¡Una lista de nodos no es una matriz!

Una lista de nodos puede parecerse a una matriz, pero no lo es.

Puede recorrer la lista de nodos y referirse a sus nodos como una matriz.

Sin embargo, no puede usar métodos de matriz, como valueOf (), push (), pop () o join () en una lista de nodos.