

🏠 → El navegador: Documentos, Eventos e Interfaces → Documento

📅 24 de octubre de 2022

# Árbol del Modelo de Objetos del Documento (DOM)

La estructura de un documento HTML son las etiquetas.

Según el Modelo de Objetos del Documento (DOM), cada etiqueta HTML es un objeto. Las etiquetas anidadas son llamadas "hijas" de la etiqueta que las contiene. El texto dentro de una etiqueta también es un objeto.

Todos estos objetos son accesibles empleando JavaScript, y podemos usarlos para modificar la página.

Por ejemplo, `document.body` es el objeto que representa la etiqueta `<body>`.

Ejecutar el siguiente código hará que el `<body>` sea de color rojo durante 3 segundos:

```
1 document.body.style.background = 'red'; // establece un color de fondo rojo
2
3 setTimeout(() => document.body.style.background = '', 3000); // volver atrás
```

En el caso anterior usamos `style.background` para cambiar el color de fondo del `document.body`, sin embargo existen muchas otras propiedades, tales como:

- `innerHTML` – contenido HTML del nodo.
- `offsetWidth` – ancho del nodo (en píxeles).
- ..., etc.

Más adelante, aprenderemos otras formas de manipular el DOM, pero primero necesitamos conocer su estructura.

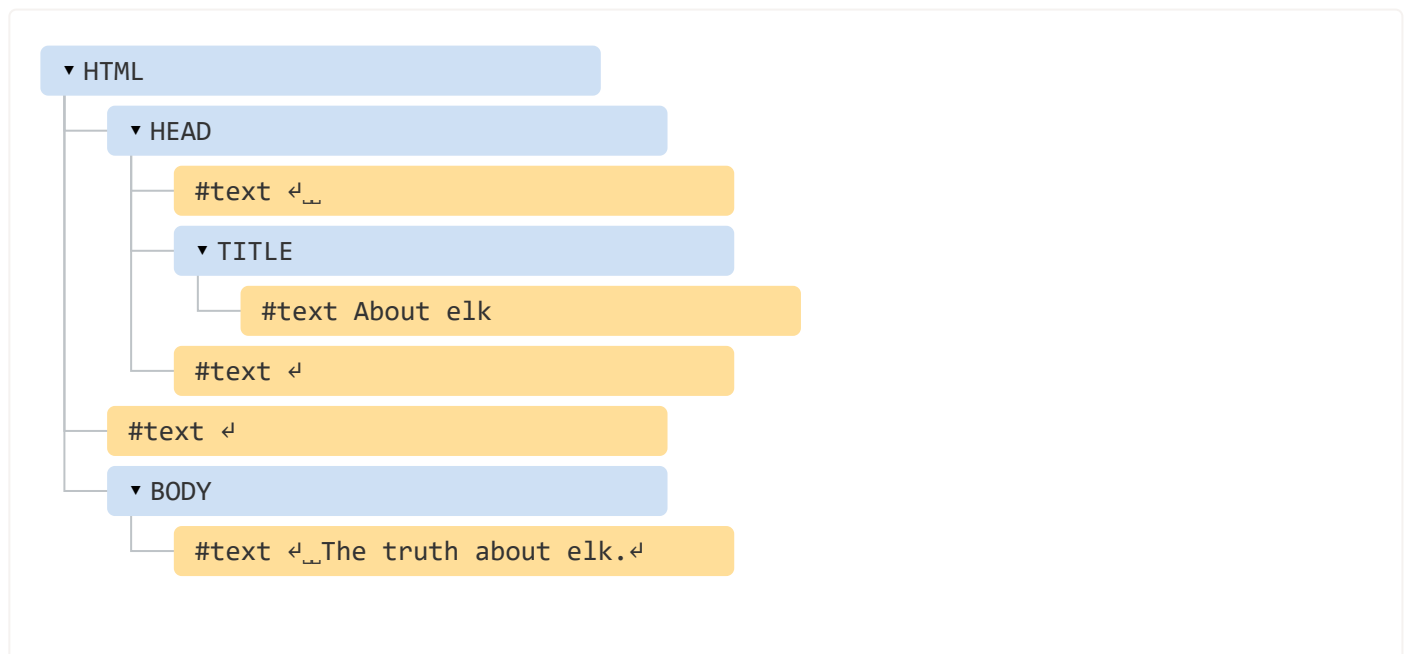
## Un ejemplo del DOM

Comencemos con el siguiente documento simple:

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <title>About elk</title>
5 </head>
6 <body>
7   The truth about elk.
8
```

```
9 </body>
  </html>
```

El DOM representa HTML como una estructura de árbol de etiquetas. A continuación podemos ver como se muestra:



En la imagen de arriba, puedes hacer clic sobre los nodos del elemento y como resultado se expanden/colapsan sus nodos hijos.

Cada nodo del árbol es un objeto.

Las etiquetas son *nodos de elementos* (o solo elementos) y forman la estructura del árbol: `<html>` está ubicado en la raíz del documento, por lo tanto, `<head>` y `<body>` son sus hijos, etc.

El texto dentro de los elementos forma *nodos de texto*, etiquetados como `#text`. Un nodo de texto contiene solo una cadena. Esta puede no tener hijos y siempre es una hoja del árbol.

Por ejemplo, la etiqueta `<title>` tiene el texto "About elk".

Hay que tener en cuenta los caracteres especiales en nodos de texto:

- una línea nueva: `↵` (en JavaScript se emplea `\n` para obtener este resultado)
- un espacio:

Los espacios y líneas nuevas son caracteres totalmente válidos, al igual que letras y dígitos. Ellos forman nodos de texto y se convierten en parte del DOM. Así, por ejemplo, en el caso de arriba la etiqueta `<head>` contiene algunos espacios antes de la etiqueta `<title>`, entonces ese texto se convierte en el nodo `#text`, que contiene una nueva línea y solo algunos espacios.

Hay solo dos excepciones de nivel superior:

1. Los espacios y líneas nuevas antes de la etiqueta `<head>` son ignorados por razones históricas.
2. Si colocamos algo después de la etiqueta `</body>`, automáticamente se sitúa dentro de `body`, al final, ya que, la especificación HTML necesita que todo el contenido esté dentro de la etiqueta `<body>`, no puede haber espacios después de esta.

En otros casos todo es sencillo – si hay espacios (como cualquier carácter) en el documento, se convierten en nodos de texto en el DOM, y si los eliminamos, entonces no habrá ninguno.

En el siguiente ejemplo, no hay nodos de texto con espacios en blanco:

```
1 <!DOCTYPE HTML>
2 <html><head><title>About elk</title></head><body>The truth about elk.</body></html>
```



### **i Los espacios al inicio/final de la cadena y los nodos de texto que solo contienen espacios en blanco, por lo general, están ocultos en las herramientas**

Las herramientas del navegador (las veremos más adelante) que trabajan con DOM usualmente no muestran espacios al inicio/final del texto y nodos de texto vacíos (saltos de línea) entre etiquetas.

De esta manera, las herramientas para desarrolladores ahorran espacio en la pantalla.

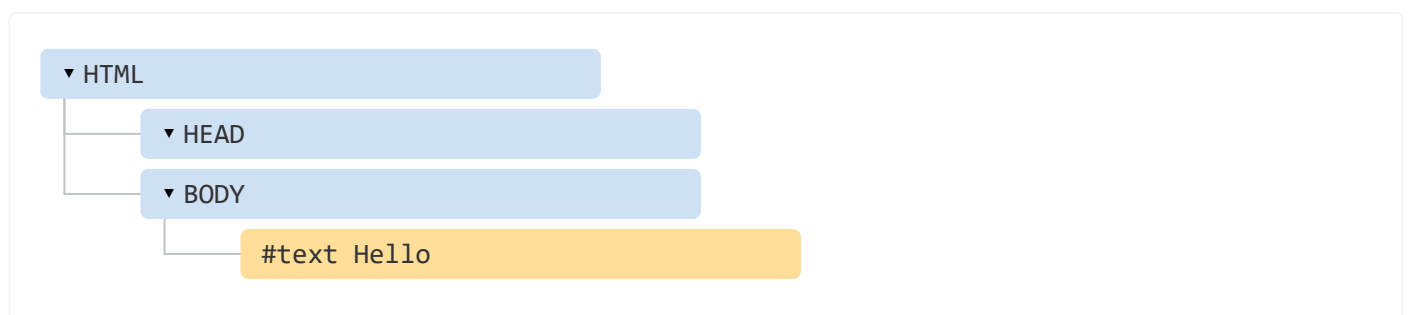
En otras representaciones del DOM, las omitiremos cuando sean irrelevantes. Tales espacios generalmente no afectan la forma en la cual el documento es mostrado.

## Autocorrección

Si el navegador encuentra HTML mal escrito, lo corrige automáticamente al construir el DOM.

Por ejemplo, la etiqueta superior siempre será `<html>`. Incluso si no existe en el documento, ésta existirá en el DOM, puesto que, el navegador la creará. Sucede lo mismo con la etiqueta `<body>`.

Como ejemplo de esto, si el archivo HTML es la palabra "Hello", el navegador lo envolverá con las etiquetas `<html>` y `<body>`, y añadirá la etiqueta `<head>` la cual es requerida, basado en esto, el DOM resultante será:

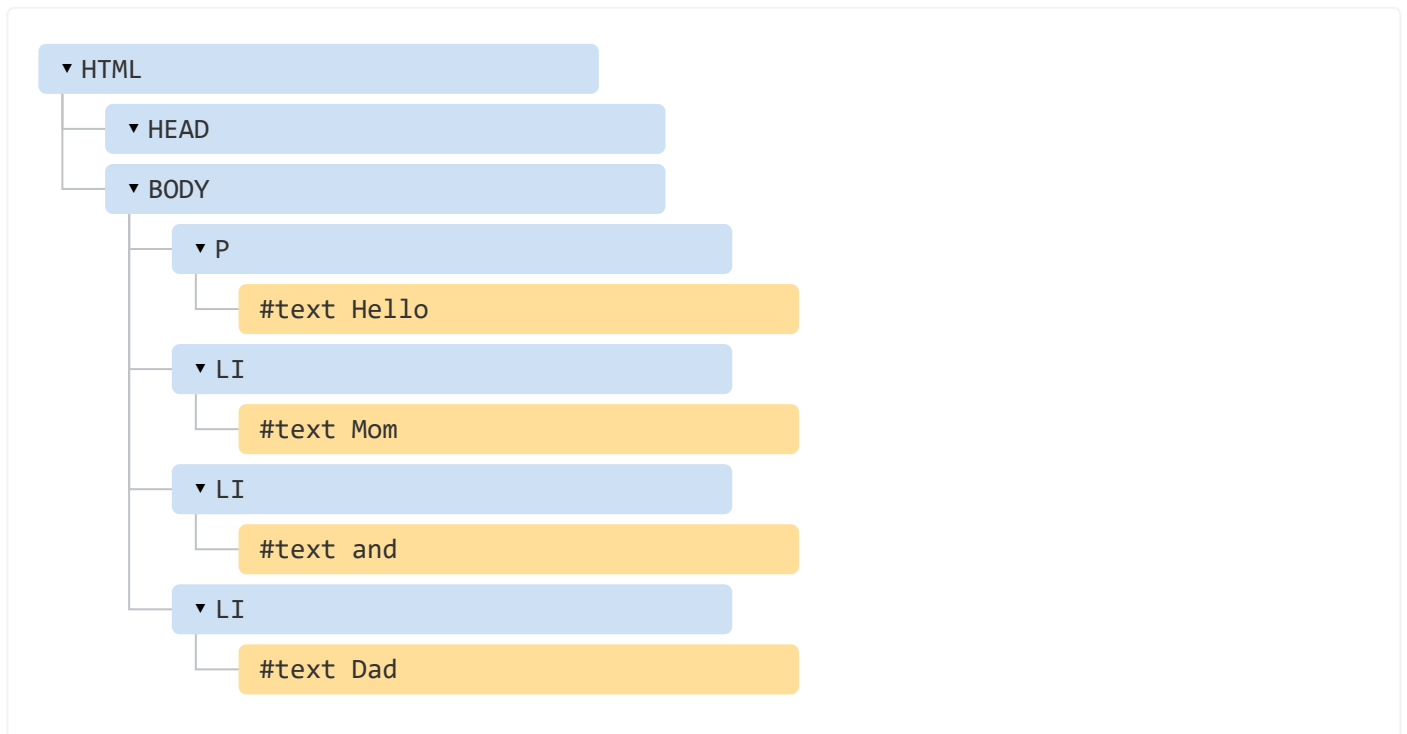


Al generar el DOM, los navegadores procesan automáticamente los errores en el documento, cierran etiquetas, etc.

Un documento sin etiquetas de cierre:

```
1 <p>Hello
2 <li>Mom
3 <li>and
4 <li>Dad
```

...se convertirá en un DOM normal a medida que el navegador lee las etiquetas y compone las partes faltantes:



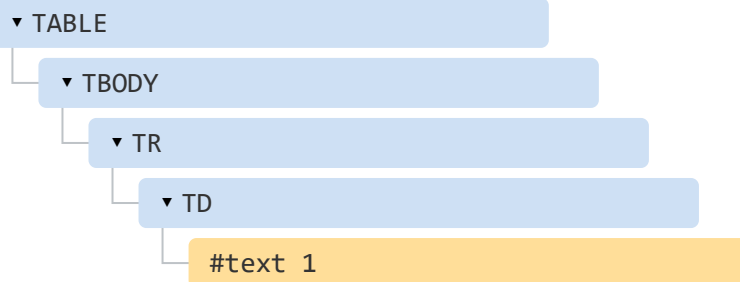
## ⚠ Las tablas siempre tienen la etiqueta `<tbody>`

Un caso especial interesante son las tablas. De acuerdo a la especificación DOM deben tener la etiqueta `<tbody>`, sin embargo el texto HTML puede omitirla: el navegador crea automáticamente la etiqueta `<tbody>` en el DOM.

Para el HTML:

```
1 <table id="table"><tr><td>1</td></tr></table>
```

La estructura del DOM será:



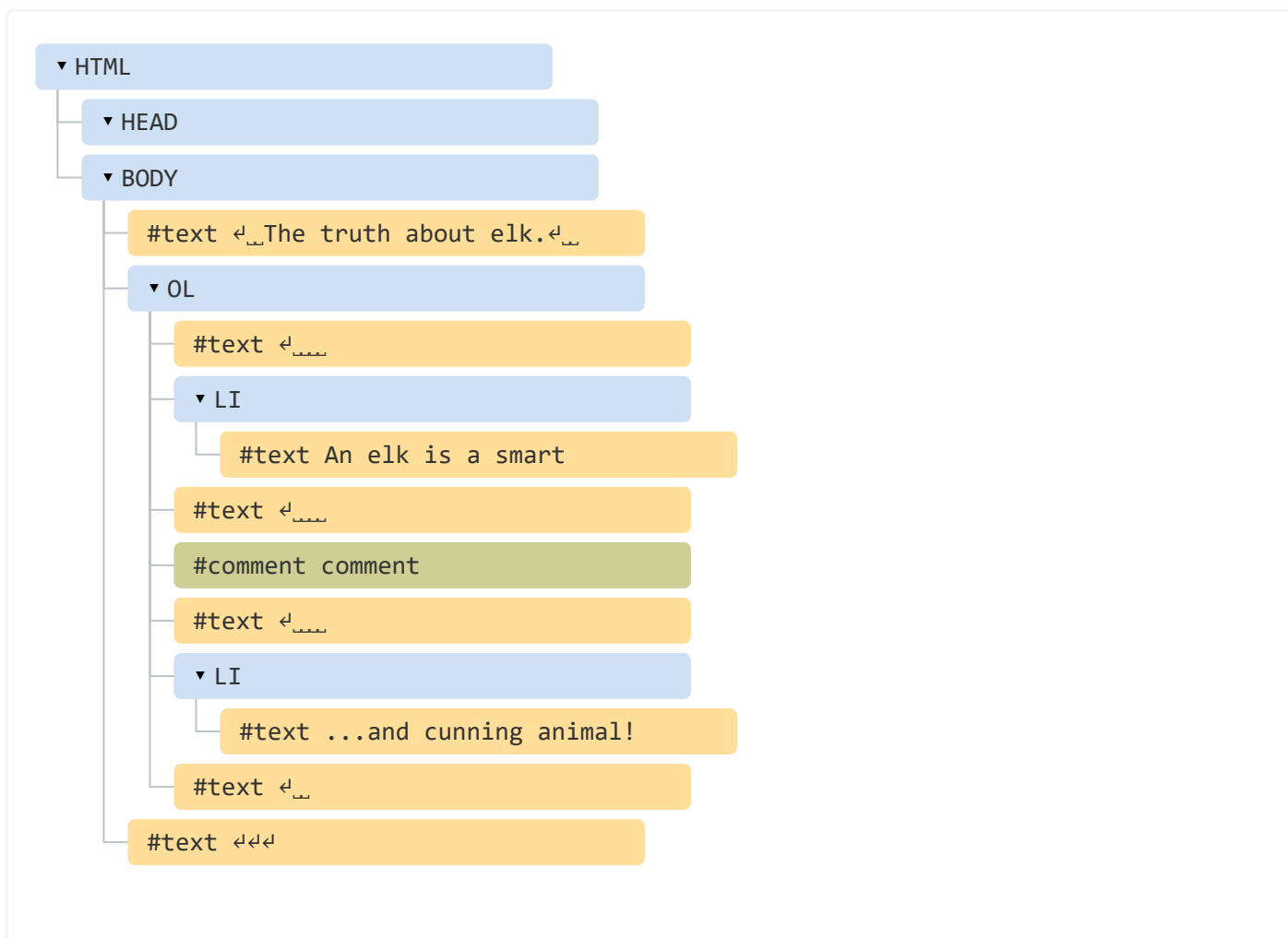
¿Lo ves? La etiqueta `<tbody>` apareció de la nada. Debemos tener esto en cuenta al trabajar con tablas para evitar sorpresas.

## Otros tipos de nodos

Existen otros tipos de nodos además de elementos y nodos de texto.

Por ejemplo, los comentarios:

```
1 <!DOCTYPE HTML>
2 <html>
3 <body>
4   The truth about elk.
5   <ol>
6     <li>An elk is a smart</li>
7     <!-- comentario -->
8     <li>...y el astuto animal!</li>
9   </ol>
10 </body>
11 </html>
```



Aquí podemos ver un nuevo tipo de nodo de árbol – *nodo de comentario*, etiquetado como `#comment`, entre dos nodos de texto.

Podemos pensar – ¿Por qué se agrega un comentario al DOM? Esto no afecta la representación de ninguna manera. Pero hay una regla – si algo está en el código HTML, entonces también debe estar en el árbol DOM.

**Todo en HTML, incluso los comentarios, se convierte en parte del DOM.**

Hasta la declaración `<!DOCTYPE...>` al principio del HTML es un nodo del DOM. Su ubicación en el DOM es justo antes de la etiqueta `<html>`. No vamos a tocar ese nodo, por esa razón ni siquiera lo dibujamos en diagramas, pero esta ahí.

El objeto `document` que representa todo el documento es también, formalmente, un nodo DOM.

Hay **12 tipos de nodos**. En la práctica generalmente trabajamos con 4 de ellos:

1. `document` – el “punto de entrada” en el DOM.
2. nodos de elementos – Etiquetas-HTML, los bloques de construcción del árbol.
3. nodos de texto – contienen texto.
4. comentarios – a veces podemos colocar información allí, no se mostrará, pero JS puede leerla desde el DOM.

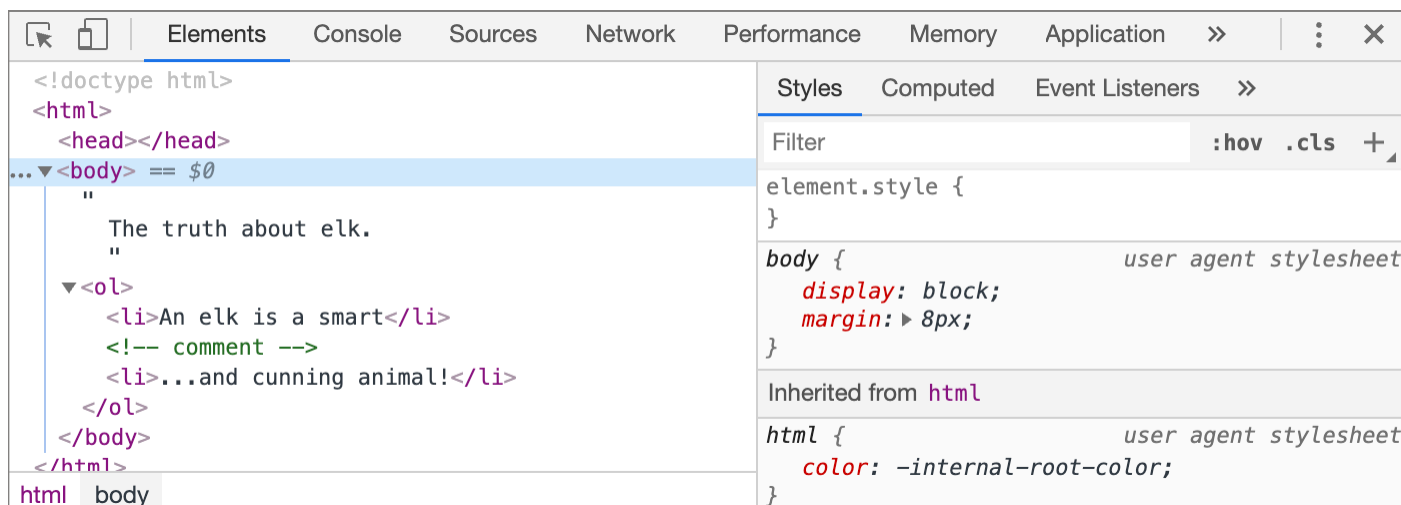
## Véalo usted mismo

Para ver la estructura del DOM en tiempo real, intente [Live DOM Viewer](#). Simplemente escriba el documento, y se mostrará como un DOM al instante.

Otra forma de explorar el DOM es usando la herramienta para desarrolladores del navegador. En realidad, eso es lo que usamos cuando estamos desarrollando.


Para hacerlo, abra la página web [elk.html](#), active las herramientas para desarrolladores del navegador y cambie la pestaña a elementos.

Debe verse así:

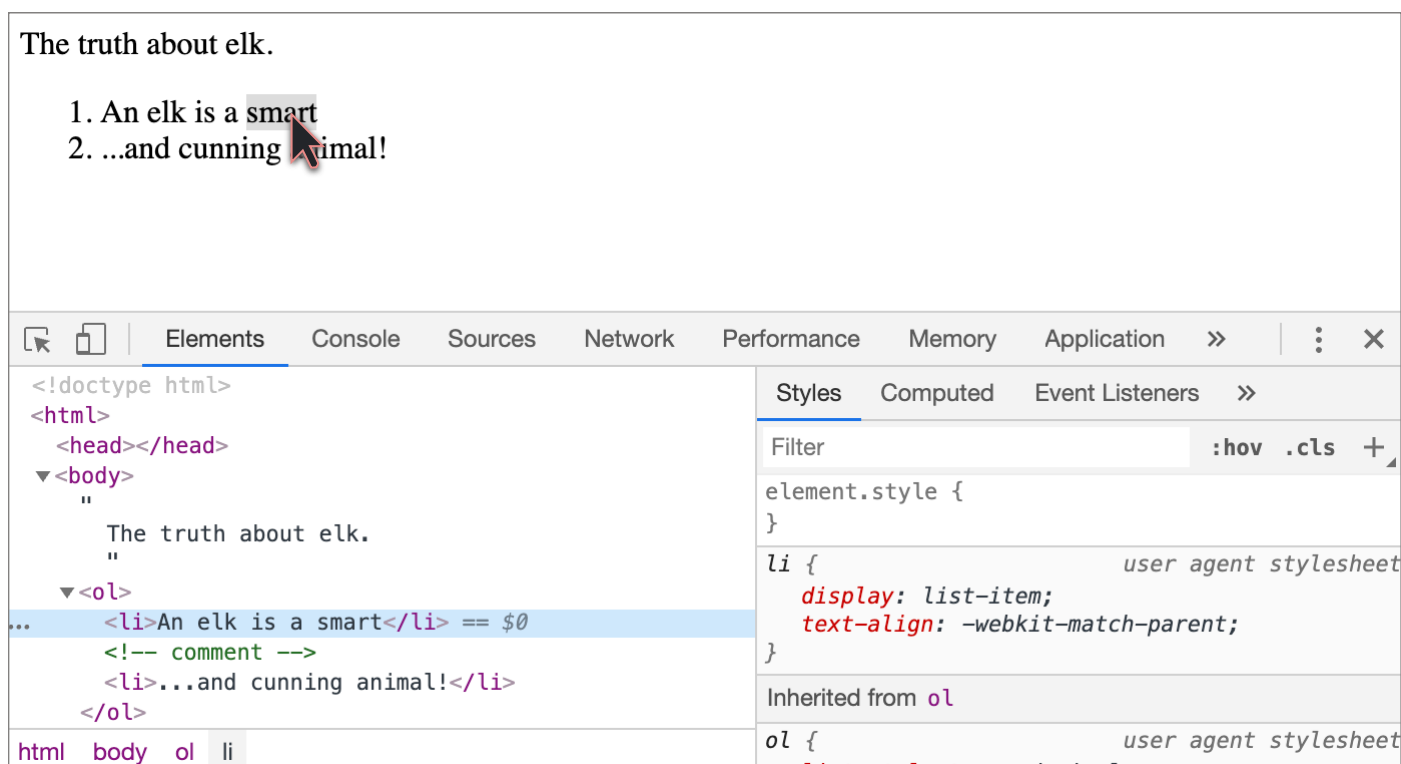


Puedes ver el DOM, hacer clic sobre los elementos, ver sus detalles, etc.

Tenga en cuenta que la estructura DOM en la herramienta para desarrolladores está simplificada. Los nodos de texto se muestran como texto. Y absolutamente no hay nodos de texto con espacios en blanco. Esto está bien, porque la mayoría de las veces nos interesan los nodos de elementos.

Hacer clic en el botón  ubicado en la esquina superior izquierda nos permite elegir un nodo desde la página web utilizando un "mouse" (u otros dispositivos de puntero) e "inspeccionar" (desplazarse hasta él en la pestaña elementos). Esto funciona muy bien cuando tenemos una página HTML enorme (y el DOM correspondiente es enorme) y nos gustaría ver la posición de un elemento en particular.

Otra forma de realizarlo sería hacer clic derecho en la página web y en el menú contextual elegir la opción "Inspeccionar Elemento".



En la parte derecha de las herramientas encontramos las siguientes sub-pestañas:

- **Styles** – podemos ver CSS aplicado al elemento actual regla por regla, incluidas las reglas integradas (gris). Casi todo puede ser editado en el lugar, incluyendo las dimensiones/márgenes/relleno de la siguiente caja.
- **Computed** – nos permite ver cada propiedad CSS aplicada al elemento: para cada propiedad podemos ver la regla que la provee (incluida la herencia CSS y demás).
- **Event Listeners** – nos ayuda a ver los listener de eventos adjuntos a elementos del DOM (los cubriremos en la siguiente parte del tutorial).
- ...,etc.

La manera de estudiarlos es haciendo clic en ellos. Casi todos los valores son editables en el lugar.

## Interacción con la consola

A medida que trabajamos el DOM, también podemos querer aplicarle JavaScript. Al igual que: obtener un nodo y ejecutar algún código para modificarlo, para ver el resultado. Aquí hay algunos consejos para desplazarse entre la pestaña elementos y la consola.

Para empezar:

1. Seleccione el primer elemento `<li>` en la pestaña elementos.
2. Presiona la tecla `Esc` – esto abrirá la consola justo debajo de la pestaña de elementos.

Ahora el último elemento seleccionado esta disponible como `$0`, el seleccionado previamente es `$1`, etc.

Podemos ejecutar comandos en ellos. Por ejemplo, `$0.style.background = 'red'` hace que el elemento de la lista seleccionado sea rojo, algo así:

The truth about elk.

1. An elk is a smart
2. ...and cunning animal!

Elements Console Sources Network Performance Memory Application >> ⋮ ✕

```
<!doctype html>
<html>
  <head></head>
  <body>
    "
      The truth about elk.
    "
    <ol>
      <li style="background: red;">An elk is a smart
    </ol>
  </body>
</html>
```

Styles Computed Event Listeners >>

Filter :hov .cls +

```
element.style {
  background: red;
}

li {
  display: list-item;
  text-align: -webkit-match-parent;
}
```

html body ol li

Console ✕

top Filter Default levels ⚙

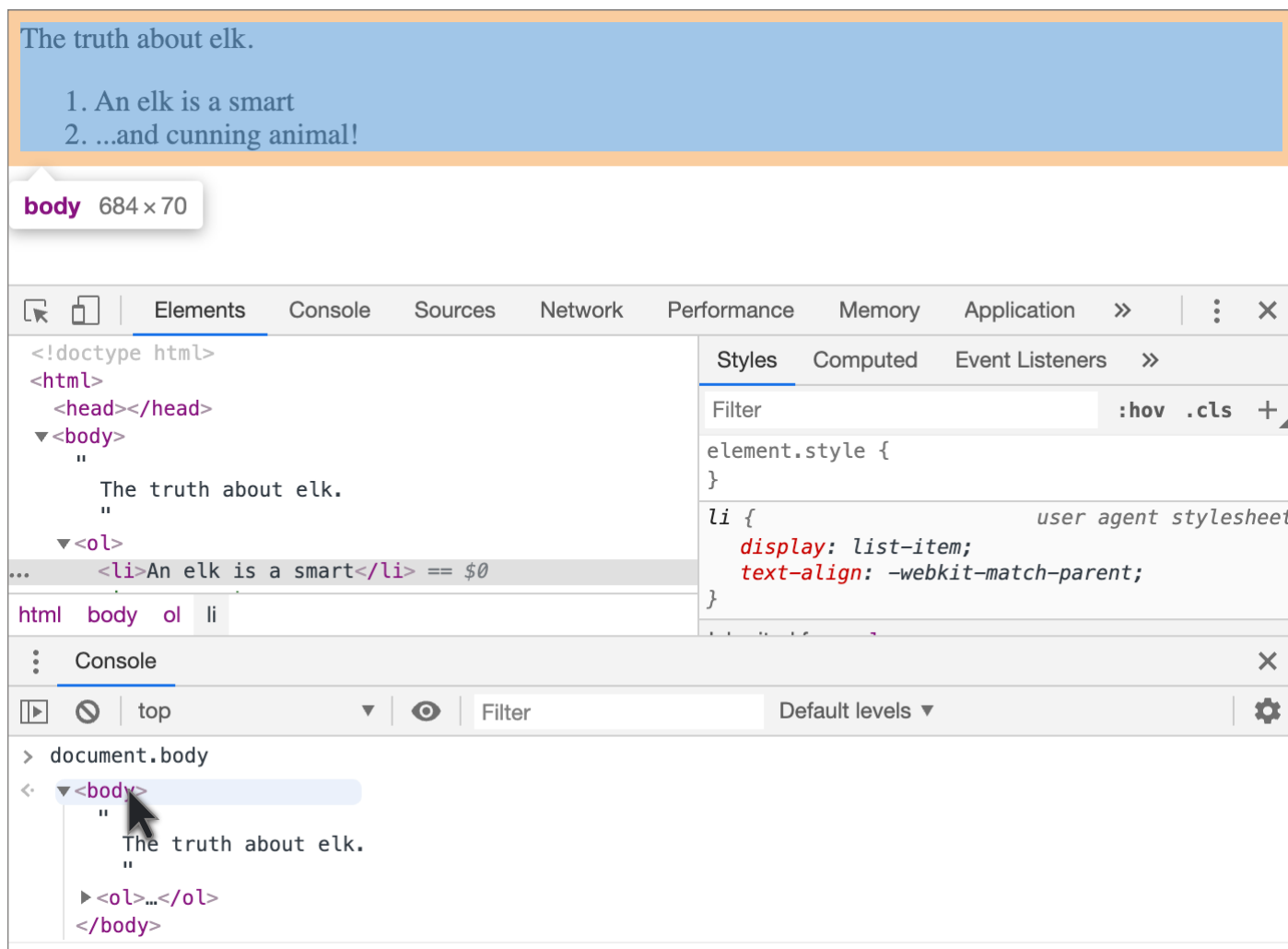
```
> $0.style.background = 'red'
< "red"
> |
```

Así es como en la consola, se obtiene un nodo de los elementos.



También hay un camino de regreso. Si hay una variable que hace referencia a un nodo del DOM, usamos el comando `inspect(node)` en la consola para verlo en el panel de elementos.

O simplemente podemos generar el nodo del DOM en la consola y explorar en el lugar, así como `document.body` a continuación:



Desde luego, eso es para propósitos de depuración del curso. A partir del siguiente capítulo accederemos y modificaremos el DOM usando JavaScript.

Las herramientas para desarrolladores del navegador son de mucha ayuda en el desarrollo: podemos explorar el DOM, probar cosas y ver que sale mal.

## Resumen

Un documento HTML/XML está representado dentro del navegador como un árbol de nodos (DOM).

- Las etiquetas se convierten en nodos de elemento y forman la estructura.
- El texto se convierte en nodos de texto.
- ...etc, todos los elementos de HTML tienen su lugar en el DOM, incluso los comentarios.

Podemos utilizar las herramientas para desarrolladores para inspeccionar el DOM y modificarlo manualmente.

Aquí cubrimos los conceptos básicos, las acciones más importantes y más utilizadas, para comenzar. Hay una extensa documentación acerca de las herramientas para desarrolladores de Chrome en <https://developers.google.com/web/tools/chrome-devtools>. La mejor forma de aprender a usar las herramientas

es hacer clic en ellas, leer los menús: la mayoría de las opciones son obvias. Más adelante, cuando tenga conocimiento general sobre ellas, lea los documentos y elija el resto.

Los nodos del DOM tienen propiedades y métodos que nos permiten desplazarnos entre ellos, modificarlos, moverlos por la página, y más. Empezaremos a realizar todo esto en los siguientes capítulos.



Lección anterior

Próxima lección



Compartir



Mapa del Tutorial

## Comentarios

- Si tiene sugerencias sobre qué mejorar, por favor [enviar una propuesta de GitHub](#) o una solicitud de extracción en lugar de comentar.
- Si no puede entender algo en el artículo, por favor explique.
- Para insertar algunas palabras de código, use la etiqueta `<code>` , para varias líneas – envolverlas en la etiqueta `<pre>` , para más de 10 líneas – utilice un entorno controlado (sandbox) ([plnkr](#), [jsbin](#), [codepen...](#))