JavaScript DOM

Lili Nemec Zlatolas

Annotation

Este curso presenta el modelo de objetos del documento en JavaScript.

Objectives

El curso proporciona una rápida visión general de los conocimientos previos necesarios, como los fundamentos de HTML y JavaScript. El alumno aprenderá sobre el modelo de objetos del documento en JavaScript.

Keywords

JavaScript, DOM, HTML

Date of Creation

15.4.2022

Duration

12 horas

Language

English

License

ISBN

Literature

[1] Matt Frisbie. Professional JavaScript for Web Developers. Publishing place: John Wiley & Sons, 2019. 978-1-119-36644-7.

[2] R. Ferguson. Beginning JavaScript: The Ultimate Guide to Modern JavaScript Development. Apress, Ocean, 2019. 3rd edition.

[3] M. Haverbeke. Eloquent JavaScript - A Modern Introduction to Programming. Third Edition. No Starch Press, San Francisco, 2018.

[4] W3schools. Javascript HTML DOM. https://www.w3schools.com/js/js_htmldom.asp.

CHAPTER 1

Fundamentos de HTML

Para manipular sitios web a través del modelo de objetos del documento (DOM), se requieren algunos conocimientos previos de HTML y JavaScript. HTML es un lenguaje de marcado que describe la estructura de un sitio web. El HTML consta de elementos que se etiquetan y envían la información al navegador sobre cómo mostrar el contenido del documento.

El último estándar aceptado es HTML5. HTML consta de elementos que se definen mediante una etiqueta de inicio, el contenido del elemento y la etiqueta de fin : <tag> Content </tag>

Este es un ejemplo de documento HTML:

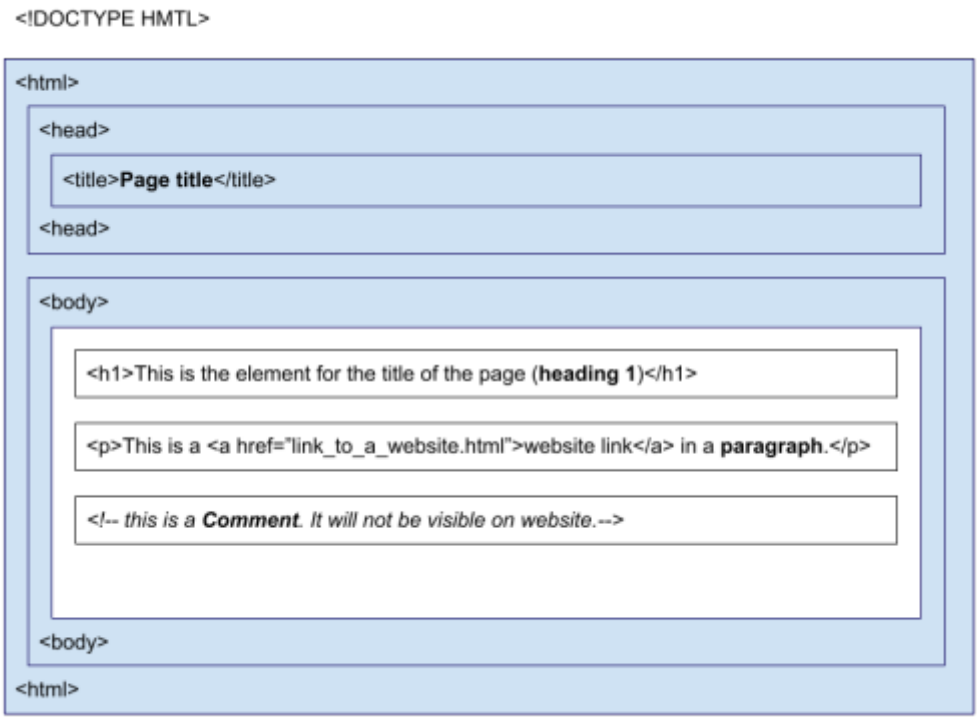

Fig. 1. HT Estructura de la página HTML y algunos elementos

Para editar los documentos, puede utilizar el Bloc de notas o programas similares (por ejemplo, Notepad++, Visual Studio Code). El documento HTML debe tener la extensión .html.

El HTML no distingue entre mayúsculas y minúsculas, pero se recomienda utilizarlas en las etiquetas HTML.

Table 1. Algunos de los elementos básicos de HTML

Tag	Información
<! DOCTYPE html>	Esta es la declaración de HTML5. Debe aparecer en la parte superior del documento HTML.
<h1>	Hay encabezados HTML desde <h1> hasta <h6>, siendo el primero el encabezado más grande.
<p>	Esta es una etiqueta para un párrafo.
 	Se trata de una etiqueta para pasar a la siguiente fila.
<a href="https://www.websitelink.com">	Se trata de una etiqueta para un enlace web. Utiliza atributos.
<img src="image_location.jpg" alt="image_description">	Es una etiqueta para insertar una imagen. También utiliza diferentes atributos.
<b>	Esta es una etiqueta para poner el texto en negrita.

Algunos elementos se denominan elementos vacíos. Un ejemplo de un elemento vacío es
, que no tiene contenido ni la etiqueta final. Sin embargo, podría incluir la etiqueta final dentro de la etiqueta de inicio si estamos utilizando las pautas XHTML:

Las etiquetas HTML pueden tener atributos. Algunos elementos no tienen ninguna función sin los atributos. Los atributos aparecen en las etiquetas de inicio. Uno de los atributos más utilizados es el estilo, que se puede utilizar, por ejemplo, en el párrafo:

```
<p style="color: blue;"> The text will be blue. </p>
```

Interaktivní prvek

Las tablas HTML se usan a menudo y consisten en celdas dentro de columnas y filas de tablas. Una tabla HTML simple es:

EXAMPLE

```
<table border="1">
  <tr> <!--this is a table row which does not have content -->
    <th> This is table header </th>
    <th> And another table header cell in the same row </th>
  </tr>
  <tr>
    <td>this is the first cell</td>
    <td>this is the second cell in a row</td>
  </tr>
</table>
```

Interaktivní prvek

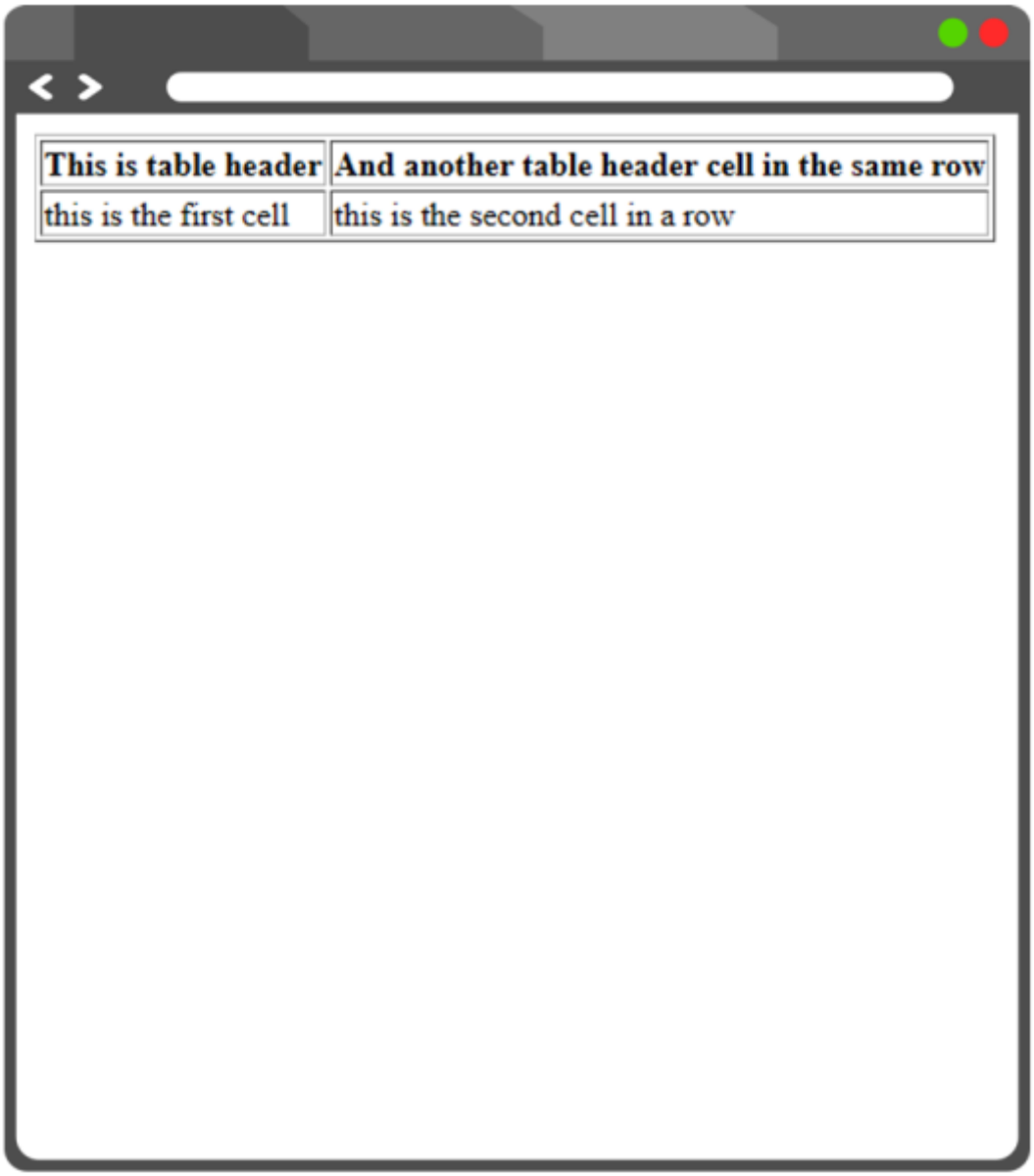

Fig. 2. Cómo se ve la tabla en el navegador.

Las listas HTML son para crear listas HTML ordenadas o no ordenadas. Las listas ordenadas están numeradas y las listas no ordenadas normalmente tienen puntos.

Table 2. Ejemplo de lista HTML

ejemplo de lista	Vista en navegador

`<ul>`
`<li>Dog</li>`
`<li>Cat</li>`
`<li>Fish</li>`
`</ul>`
	• Dog
	• Cat
	• Fish

`<ol>`
`  <li>Dog</li>`
`  <li>Cat</li>`
`  <li>Fish</li>`
`</ol>`
	1. Dog
	2. Cat
	3. Fish

Interaktivní prvek

Esta fue una breve introducción a HTML. En el siguiente capítulo, veremos algunos conceptos básicos de la sintaxis de JavaScript.

CHAPTER 2

Conceptos básicos de JavaScript

JavaScript es un lenguaje de programación que puede hacer que los sitios web sean más interactivos y dinámicos. Se utiliza en HTML entre <script> y </script> etiqueta, ya sea en <body> o <head> sección de la página HTML o ambos.

eJavaScript también se puede colocar en un archivo html como un archivo externo con .js extensión. Ejemplo:

```
<script src="JavascriptFile.js"></script>
```

Hay algunas posibilidades de cómo mostrar datos de JavaScript. Podemos utilizar:

innerHTML (ej. document.getElementById(id))

Veremos estos métodos en los próximos capítulos, porque esto es DOM

document.write()

Esto se usa normalmente para las pruebas, ya que elimina todo el HTML existente del documento y solo muestra el contenido del script.

window.alert()

Crea un cuadro de alerta para mostrar los datos. Se puede omitir la ventana.

console.log()

Esto se utiliza normalmente con fines de depuración y muestra datos en el navegador.

El código JavaScript está separado por punto y coma (;) al final de cada código ejecutable.

Los bloques de código se agrupan dentro de **{curly brackets}**.

Interaktivní prvek

Los comentarios de una sola línea usan // y los comentarios de varias líneas usan /* para comenzar y */ para terminar.

Algunas de las palabras clave más importantes en JavaScript se presentan en la tabla. Estas son las palabras reservadas que no se pueden usar para nombres de variables.

Table 3. Palabras reservadas en JavaScript

Palabras reservadas / palabras clave.	Descripción
`var`	Declara una variable, que se puede volver a declarar y actualizar.
`let`	Declara una variable de bloque que se puede actualizar, pero no volver a declarar.
`const`	Declara una variable que no se puede actualizar ni volver a declarar.
`return`	Sale de una función
`if`	Inicio del código de condición
`for`	Inicio del código de condición
`function`	Declara una función

Interaktivní prvek

La sintaxis de JavaScript son reglas de cómo se construyen los programas. Primero, las variables se crean declarándolas y luego se usan en el programa.

Los **números** decimales están separados por **un punto**. Los **textos** se escriben **entre comillas simples o dobles**. Los **signos iguales** asignan valores a las variables.

Los nombres de las variables deben comenzar con letras del alfabeto (A-Z), un signo de dólar o con un guión bajo. Distinguen entre mayúsculas y minúsculas.

Ejemplo de creación de diferentes variables:

```
let firstNumber;
var firstText;
firstNumber = 13;
var secondNumber = 17;
firstText = "This is number 13."
```

Table 4. Signos igual en JavaScript

Signo igual	Descripción
=	Se trata de un operador de asignación (por ejemplo, var a = a * 2;) y no tiene el mismo significado que el signo de álgebra.
==	Este es el operador igual que se usa en álgebra, por ejemplo, en 7 + 2 = 9, y en JavaScript se usa como, por ejemplo, **if (x == 2) {};**
===	Este es el valor igual y también el tipo igual.

Puede usar = and + para concatenar o calcular el valor de la variable.

let wholeName = "Maya" + " " + "Apellido";

Interaktivní prvek

JavaScript tiene 3 operadores lógicos:

- && logical and
- || logical or
- ! logical not

Las funciones son bloques de código que necesitan ser llamados para ser ejecutados. Pueden tener más parámetros y el código está entre corchetes rizados. Son útiles cuando usamos las funciones más veces con diferentes argumentos. Ejemplo de una función:

EXAMPLE

```
function circle_area(a, b)
{
return a * a * b;  // La función devuelve el producto de a, a y b. Si
queremos calcular el área del círculo, debemos ingresar el número PI en
lugar de B.
}
let area = circle_area (15, Math.PI);  // La función se llama con la
longitud 15 cm y el número PI.
document.write("The area of circle is " + area + " m2.");
```

Al ejecutar la función anterior, el navegador mostrará: "**The area of circle is 706.8583470577034 m2.**"

Hay mucho más que aprender en JavaScript, pero estas son algunas cosas básicas e importantes que necesitamos para comenzar con JavaScript DOM.

CHAPTER 3

Introducción al modelo de objetos de documento

El explorador crea un modelo de objetos de documento (DOM) cuando carga un sitio web. DOM es un estándar del W3C (World Wide Web Consortium) para acceder a documentos.

El DOM HTML se construye como un árbol de objetos.

En la animación se presenta un ejemplo del código siguiente.

```
<html>
  <head>
    <title>This is page title.</title>
  </head>
<body>
  <h1 align="left">This is the heading.</h1>
  <p style="background-color: blue">This is a paragraph.</p>
  <img src="image.jpg" />
</body>
</html>
```

Interaktivní prvek

Animation 1. Arbol de objetos en modelo HTML DOM

Un árbol tiene nodos para los elementos, que representan etiquetas HTML y determinan la estructura del documento. El estándar está diseñado para que primero creemos un nodo, luego le agreguemos un hijo y atributos al niño. Por lo tanto, el código puede ser bastante largo.

En la animación, <html> es el nodo raíz sin padres pero es un padre para el primer hijo <head> y el último hijo<body>.

JavaScript puede acceder a DOM y cambiar elementos y atributos en la página HTML.

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

DOM es un estándar donde se informa sobre cómo llegamos a los elementos HTML, cómo los cambiamos, los añadimos o los eliminamos en documentos HTML. DOM utiliza métodos para acceder y realizar acciones en elementos HTML.

Interaktivní prvek

3.1 Navegación DOM

Los nodos en el árbol de nodos tienen una relación jerárquica, lo que significa que cada nodo tiene exactamente un padre, excepto el primer nodo, que no tiene padre. Un nodo puede tener más hijos. Y los nodos hermanos son nodos con el mismo padre.

Para navegar entre nodos con JavaScript podemos utilizar las siguientes propiedades:

- parentNode (nodo padre)
- childNodes[nodenumber] (nodohijo)
- firstChild (primer hijo)
- lastChild (ultimo hijo)
- nextSibling (próximo hermano)
- previousSibling (previo hermano)

Animation 2. Ejemplo de nodos en un documento

Property childNodes[0] es lo mismo que firstChild. Contiene un objeto similar a una matriz con la propiedad length con la que accede a los nodos secundarios.

Interaktivní prvek

Veremos más sobre los nodos y cómo obtener el contenido de los elementos en el capítulo Nodos DOM.

CHAPTER 4

Métodos DOM

Con **métodos DOM** podemos realizar acciones sobre elementos HTML. Las **propiedades DOM** son valores de elementos HTML que podemos establecer o cambiar.

EXAMPLE

```
<p id="example"> Esto no se mostrará en la página, porque Javascript
sobrescribirá el texto dentro del nodo de párrafo.</p>
<script>
document.getElementById("example").innerHTML = " Este es el texto que se
mostrará en el sitio web.";
</script>
```

En este ejemplo, **"Este es el texto que se mostrará en el sitio web.** " se mostrará en el navegador cuando se ejecutará el código.

En este ejemplo, getElementById es un **método** e innerHTML es una **propiedad.** Muy a menudo, id (en el ejemplo id="example") se utiliza para encontrar el elemento en el documento. De lo contrario, podemos usar nodos padre e hijo para llegar a ciertos elementos.

La propiedad innerHTML se utiliza para establecer o devolver contenido HTML de un elemento, en el ejemplo anterior, está cambiando el texto dentro de la <p> etiqueta con el elemento h id.

Table 5. Métodos para encontrar, cambiar, agregar y eliminar elementos HTML elementos

Método	Descripción
document.getElementById(id)	Búsqueda de id de elemento por elemento
document.getElementsByTagName(name)	Búsqueda de elementos por nombre de etiqueta
document.getElementsByClassName(name)	Búsqueda de elemento por nombre de clase
element.setAttribute(attribute, value)	Cambiar el valor de atributo de un elemento HTML
document.createElement(element)	Creación de un elemento HTML
document.removeChild(element)	Eliminación de un elemento HTML secundario
document.appendChild(element)	Agregar un elemento HTML secundario
document.replaceChild(new, old)	Sustitución de un elemento HTML secundario

Interaktivní prvek

Video 1. Ejemplos de get element by X

Interaktivní prvek

CHAPTER 5

Nodos DOM

Todos **los elementos HTML, sus atributos y textos son nodos**. Algunos elementos contienen otros nodos.

La navegación entre los nodos ya se describió en el capítulo Navegación en el DOM.

Aquí hay un ejemplo de cómo podemos acceder al valor de los nodos.

```
<p id='paragaph'>This is first paragraph.</p>
```

El elemento <p> contiene un **nodo de texto** con el valor " This is the first paragraph". Se puede acceder al valor del texto mediante la propiedad innerHTML del nodo:

```
textFromParagraph = document.getElementById('p').innerHTML;
```

Lo mismo puede hacerse accediendo al **nodeValue**:

```
textFromParagraph
= document.getElementById('p').childNodes[0].innerHTML.nodeValue;
```

Video 2. Child nodes y node values

Los nodos raíz pueden acceder al documento completo:

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

La propiedad **nodeValue** especifica el valor de un nodo. Es nula para los nodos de elementos, pero es útil para los nodos de texto, donde presenta el propio texto. La propiedad para los nodos atributo devuelve el valor del atributo.

La propiedad **nodeName** especifica el nombre del nodo y es de sólo lectura. nodeName de un nodo elemento devuelve el nombre de la etiqueta y devuelve el nombre del atributo de un nodo atributo. nodeName de un nodo texto es #text y del nodo documentos es #document.

Utilizaremos algunos de los siguientes métodos en los subcapítulos para crear, eliminar y reemplazar elementos HTML del DOM (Nodos). En esta tabla, veremos los métodos para crear elementos y nodos de texto.

Table 6. Mètodos para crear nuevos elementos DOM HTML

Método	Descripción

createElement(type)	Crea un nodo de elemento con un tipo específico (por ejemplo, tipo "p").
createTextNode()	Crea un nodo de texto.

En la siguiente tabla se presentan los métodos para crear, eliminar y sustituir nodos. Normalmente, tenemos que crear un elemento, luego crear un nodo de texto dentro de él y añadirlo a una estructura existente. Veremos ejemplos de uso en los próximos subcapítulos

Table 7. Métodos para crear, eliminar y sustituir nodos

Método	**Descripción**
appendChild(node)	Añade un nuevo elemento hijo (nodo) al elemento.
insertBefore(new, existing)	Inserta un nodo hijo antes de un hijo existente
replaceChild(new, old)	Sustituye un nodo hijo por un nuevo nodo.
remove()	Elimina un elemento del documento. No tiene parámetros.
removeChild(node)	Elimina a un elemento hijo.

5.1 Creación de elementos DOM HTML (Nodos)

En este capítulo, veremos cómo podemos añadir, eliminar o reemplazar elementos del DOM de HTML.

Para añadir un nuevo elemento, primero debemos crear el nodo del elemento y luego añadirlo al elemento existente. Aquí tenemos un ejemplo de código en el que añadimos un nuevo párrafo con nodos.

EXAMPLE

```
<div id="div01">
 <p id="p01">First paragraph.</p>
 <p id="p02">Second paragraph.</p>
</div>
<script>
var newParagraph = document.createElement("p");
var textForParagraph = document.createTextNode("This is a new
paragraph.");
newParagraph.appendChild(textForParagraph);
var element = document.getElementById("div01");
element.appendChild(newParagraph);
</script>
```

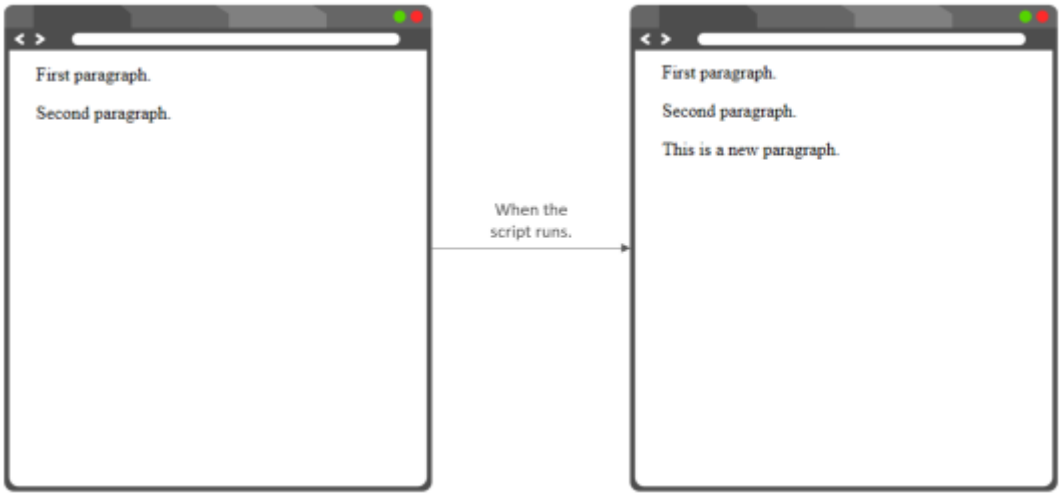

Fig. 3. Cómo se ve el código en un navegador sin el script y con el script.

createElement("p") creará un nuevo elemento <p>. Para añadir texto a este elemento, debemos crear un nodo de texto con **createTextNode.** Luego, debemos añadir el nodo de texto al elemento <p> con **appendChild**. Por último, debemos añadir el nuevo elemento a un elemento div existente.

Ahora, usaremos el mismo ejemplo, pero insertando el párrafo en la primera posición con insertBefore en lugar de la última posición como hicimos con appendChild.

Video 3. Ejemplo de método de insertBefore

Interaktivní prvek

5.2 Sustitución de elementos DOM HTML (Nodos)

Para reemplazar un elemento, primero debemos crear el nodo del elemento y luego reemplazarlo con el elemento existente. Aquí hay un ejemplo del código en el que reemplazamos un nuevo párrafo por el antiguo.

EXAMPLE

```
<div id="div01">
 <p id="p01">First paragraph.</p>
 <p id="p02">Second paragraph.</p>
</div>
<script>
var newParagraph = document.createElement("p");
var textForParagraph = document.createTextNode("This is a new
paragraph.");
newParagraph.appendChild(textForParagraph);
var parent = document.getElementById("div01");
var child = document.getElementById("p01");
parent.replaceChild(newParagraph,child);
</script>
```

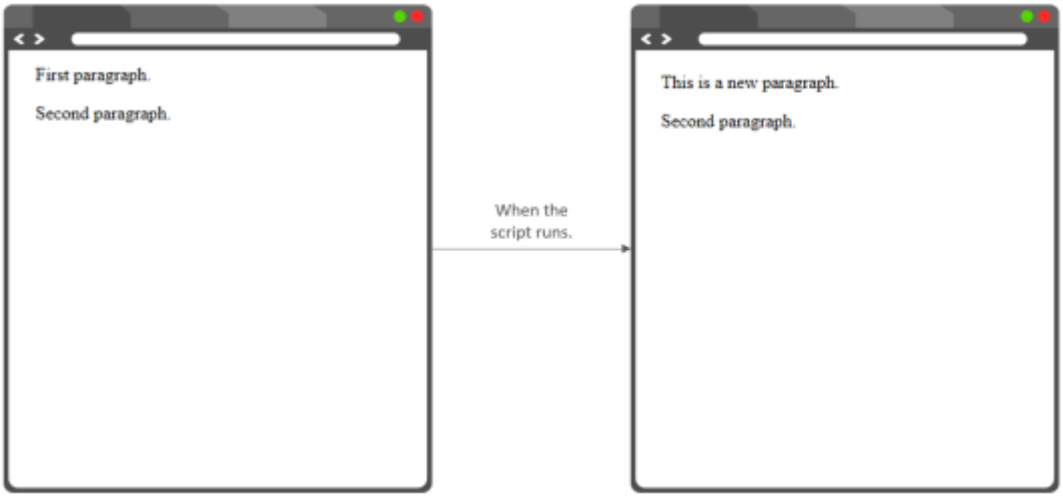

Fig. 4. Cómo se ve el código en un navegador sin el script y con el script.

Interaktivní prvek

5.3 Eliminación de elementos DOM HTML (Nodes)

En este capítulo veremos una animación donde eliminaremos un elemento con el método remove() y eliminaremos un elemento accediendo al padre con el método removeChild(). El método remove() no funciona en los navegadores más antiguos y por eso a veces tenemos que usar removeChild().

En removeChild, necesitamos buscar el padre, para conseguir eliminar el hijo.

Animation 3. Eliminación de elementos DOM HTML con dos ejemplos

Interaktivní prvek

CHAPTER 6

Validación de formularios DOM

Podemos hacer la validación de formularios HTML mediante JavaScript a través del DOM. Normalmente, queremos comprobar si el usuario ha rellenado los datos correctos en el formato correcto. Queremos asegurarnos de que la entrada del usuario es correcta. En este ejemplo, crearemos un formulario sencillo y comprobaremos si el usuario ha introducido los datos correctos con una función, utilizando el DOM.

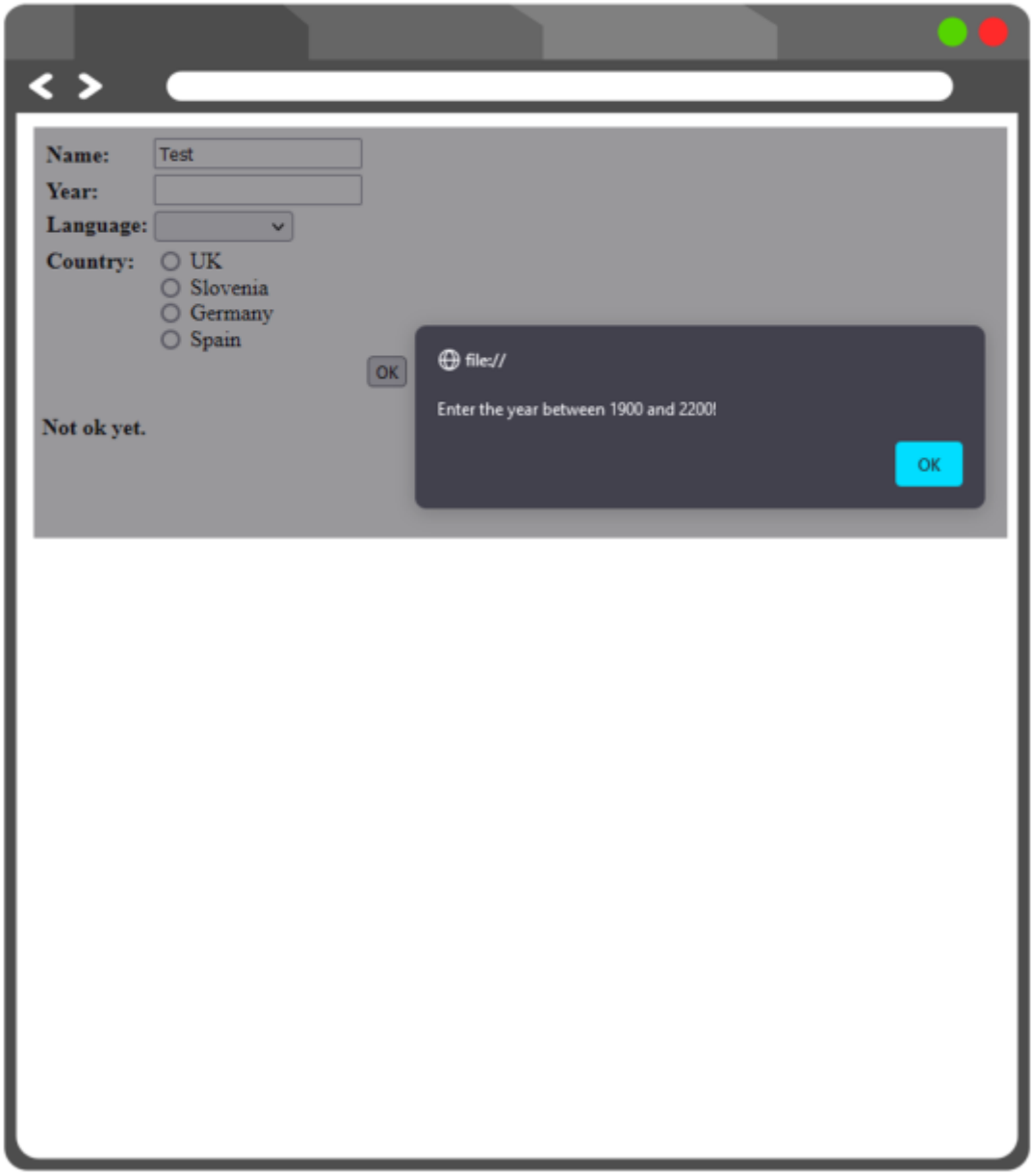

Fig. 5. Ejemplo de alerta y formulario que se presentará en el código siguiente.

EXAMPLE

Estamos creando un formulario con 4 opciones diferentes. Hemos utilizado una tabla HTML para una mejor visualización.

En el formulario se han utilizado texto, opción y entrada con tipo botón de radio, así como un botón de envío.

```
<form onsubmit="return validation( this)" action="#">
<table>
<tr>
<td><b>Name: </b></td>
<td><input type="text" name="name"></td>
</tr>
<tr>
<td><b>Year: </b></td>
<td><input type="text" name="year"></td>
</tr>
<tr>
<td valign="top"><b>Language: </b></td>
  <td>
  <select name="language">
<option> </option>
<option value="slo">english</option>
<option value="ang">slovenian</option>
<option value="nem">german</option>
<option value="fra">french</option>
</select>
</td>
</tr>
<tr>
<td valign="top"><b>Country: </b></td>
<td>
<input type="radio" name="country" value="1"> UK<br/>
<input type="radio" name="country" value="2"> Slovenia<br/>
<input type="radio" name="country" value="3"> Germany<br/>
<input type="radio" name="country" value="4"> Spain
  </td>
</tr>
<tr>
<td></td>
<td></td>
  <td><input type="submit" value="OK"></td>
</tr>
</table>
</form>
<script type="text/javascript">
```

Aquí se crea una función de validación. Cuando algo no sea rellenado correctamente por el usuario, aparecerá una ventana de alerta.

```
function validation( form)
{
```

Con el DOM, hemos accedido al formulario, a la entrada del nombre y a su valor. Si éste está vacío, aparecerá una alerta.

```
if ( form. name. value == "")
{
alert( "Enter the name! ")
return false
}
```

A continuación, si el año introducido está fuera del rango 1900 y 2200, aparecerá la alerta. Con isNaN también comprobamos si el usuario ha introducido algún número.

```
if ( isNaN( form. year. value) ||  form. year. value < 1900 ||  form. year. value >
2200)
{
alert( "Enter the year between 1900 and 2200! ")
return false
}
```

A continuación, si el año introducido está fuera del rango 1900 y 2200, aparecerá la alerta. Con isNaN también comprobamos si el usuario ha introducido un número en a.

```
if ( form. language. selectedIndex <= 0)
  {
alert( "Select the language! ")
return false
  }
```

A continuación, si el usuario no ha seleccionado ninguno de los países con botón de opción, aparecerá una alerta.

```
var i = 0
while ( i < form. country. length && ! form. country[ i]. checked)
++i

if ( i == form. country. length)
{
alert( "Choose a country! ")
return false
}
return true;
}
</script>
```

Interaktivní prvek

CHAPTER 7

DOM que cambia el CSS

A menudo, el DOM se utiliza para cambiar el estilo de los elementos HTML. La sintaxis para ello es:

document.getElementById(id).style.property = new style;

Por lo general, un visitante de un sitio web debe hacer clic en algún elemento como un botón y el cambio de estilo CSS en el documento puede ocurrir. Esto también se llama eventos DOM. La sintaxis para ello es:

onclick=JavaScript

Interaktivní prvek

En el siguiente ejemplo, veremos algunas de las propiedades y eventos que podemos utilizar.

Animation 4. Ejemplo de DOM que cambia el CSS.

EXAMPLE

```
<form name="newForm">
<p id='p01'>Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
```

Este botón de abajo cambiará el color del texto de arriba a color verde.

```
<input type='button' onclick='document.getElementById("p01").style.color
= "green";' value='Change text color'/><br/><br/>
```

Este botón de abajo cambiará el tamaño de la fuente del texto de arriba a 27px.

```
<input type='button'
onclick='document.getElementById("p01").style.fontSize = "27px";'
value='Change font size'/><br/><br/>
<p id='p02'>Current time will be displayed here. </p>
```

Este botón llamará a una función llamada fecha que está en la parte <script> y mostrará la hora actual en el párrafo anterior.

```
<input type='button' onclick='date()' value='Display current
time'/><br/><br/>
```

La siguiente entrada presenta una caja, donde el usuario introduce el color. La siguiente entrada es una casilla de verificación y la última tiene una función que cambiará el fondo al texto escrito en una caja.

```
<input type='text' id='entry' value='Enter color for background'
size='40' /> <br/><br/>
<input type="checkbox" id="box" value="box"> Change background <input
type='button' onclick='changeBackground()' value='Change background'/>
</form>
<script>
```

Esta función mostrará la fecha y la hora actuales.

```
function date() {
document.getElementById("p02").innerHTML = Date();
}
```

Esta función cambiará el color de fondo a lo que el usuario haya introducido en una casilla y sólo si la casilla está marcada.

```
function changeBackground() {
var checkbox = document.forms[0].box;
if (checkbox.checked) {
var body = document.getElementsByTagName("body")[0];
body.style.background = document.getElementById("entry").value;
}
}
</script>
```

Interaktivní prvek

CHAPTER 8

DOM cambiando tablas

En este ejemplo, haremos una lista de tareas con una tabla HTML. Podemos manipular las tablas añadiendo o eliminando filas, cabeceras, celdas, etc.

Table 8. Métodos para añadir, eliminar diferentes elementos en las tablas

Método	Descripción
deleteRow()	Elimina un <tr> de una tabla.
insertRow()	Crea un elemento <tr> vacío en una tabla.
deleteCell()	Elimina la celda de la fila de la tabla actual.
insertCell()	Crea una celda en una fila de la tabla actual.

En el ejemplo siguiente, crearemos una lista de tareas en la que podremos añadir y eliminar tablas de una mesa haciendo clic en la casilla de verificación.

EXAMPLE

En primer lugar, crearemos dos botones y un campo de texto, donde un usuario puede añadir una nueva tarea, eliminar una nueva tarea o escribir cuál es la tarea. Las funciones se llaman a través de onclick y están en la sección <script>.

```
<button onclick="addTask()">Add new task</button>
<button onclick="deleteTask()">Task done</button>
<input type="text" id="textbox" placeholder="Enter task"><br><br>
```

Esta es una tabla simple con sólo la cabeza de la tabla, sin ninguna tarea. Éstas se añadirán mediante funciones.

```
<table id="table" style="width:50%">
<tr>
  <th>Checkbox</th>
  <th>Row number</th>
  <th>Task</th>
</tr>
</table>

<script>
```

Primero, crearemos una función para añadir nuevas tareas a las tablas. Necesitamos buscar la tabla y en este ejemplo, estamos usando getElementById.

```
var count_lines=0;
function addTask() {
 var table = document.getElementById("table");
```

A continuación, hay que insertar las filas de abajo a arriba con el método insertRow y -1 (para que la tarea vaya al fondo). Después, se insertan tres celdas en la tabla con tres columnas.

```
var row = table.insertRow(-1);
 var cell1 = row.insertCell(0);
 var cell2 = row.insertCell(1);
 var cell3 = row.insertCell(2);
```

A continuación, añadimos 1 a las variables count_lines para la segunda columna que cuenta el número actual de tareas añadidas.

```
count_lines++;
```

En la primera celda se inserta una casilla de verificación. En la segunda celda se inserta el contador de líneas y en la tercera casilla se inserta el valor del texto del cuadro de texto.

```
cell1.innerHTML = '<input type="checkbox" name="box" value="0">';
 cell2.innerHTML = count_lines;
 cell3.innerHTML = document.getElementById("textbox").value;
}
```

A continuación, vamos a crear una función para eliminar las tareas terminadas a las tablas. Tenemos que buscar la tabla de nuevo y en este ejemplo, estamos utilizando getElementById.

```
function deleteTask() {
 var table = document.getElementById("table");
```

A continuación, hemos creado una variable i para encontrar la fila en la que está marcada la casilla de verificación y así poder eliminar la fila que el usuario ha seleccionado como hecha. Hemos utilizado chilNodes y comprobado si la casilla de verificación está marcada.

```
let i;
 for (i = table.rows.length-1; i >= 1; i--) {
if(table.rows[i].cells[0].childNodes[0].checked==true){
table.deleteRow(i);
}
  }
}
</script>
```

Interaktivní prvek

Interaktivní prvek

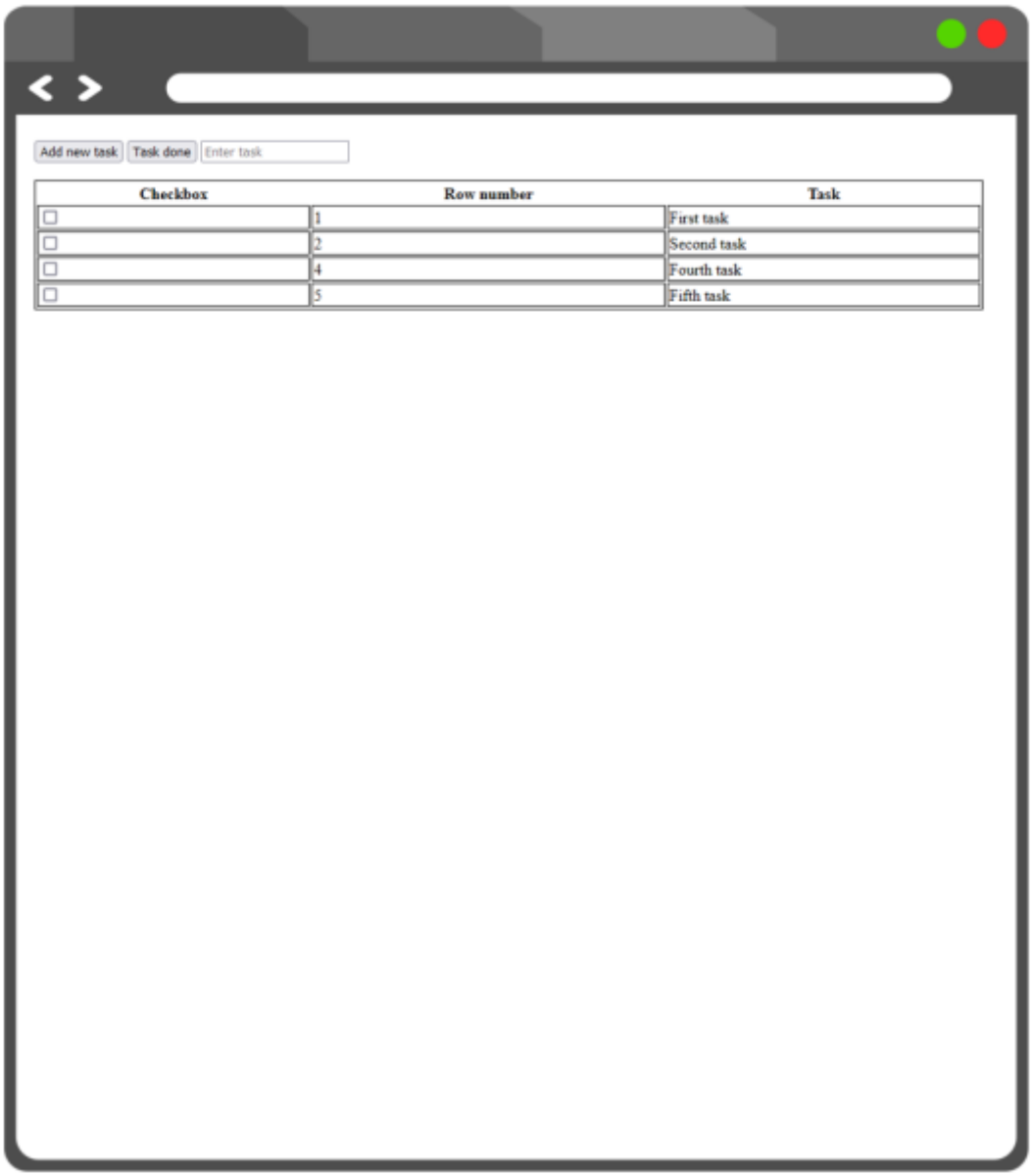

Fig. 6. Cómo se ve la lista de tareas en el navegador.

Interaktivní prvek

Este ha sido un curso de introducción a JavaScript DOM.

CHAPTER 9

Test

Where does the <title> tag appear?

- ◯ under <body> tag
- ◯ under <head> tag
- ◯ under <p> tag
- ◯ under <img> tag

¿Qué son los elementos en HTML?

- ☐ nombre de usuario
- ☐ etiqueta de inicio
- ☐ contenido
- ☐ etiqueta final

Which tag is for making the text bold?

- ◯ <u>
- ◯ <i>
- ◯ <b>
- ◯ <q>

¿Qué es un elemento vacío?

- ☐ un elemento <>
- ☐ un elemento que no tiene una etiqueta de fin

- ☐ un elemento que no tiene contenido
- ☐ un elemento sin atributo

¿Dónde se puede ubicar la etiqueta

- ☐ en la etiqueta <head>.
- ☐ en la etiqueta <tab>
- ☐ en la etiqueta <body>
- ☐ antes de la etiqueta <!DOCTYPE html>

How do you create a comment in HTML?

- ◯ <!-- comment -->
- ◯ /* comment */
- ◯ // comment

Which tag creates a list with dots in front?

- ◯
- ◯ <dl>
- ◯
- ◯

El texto en JavaScript se escribe dentro:

- ☐ barra inclinada
- ☐ comillas simples
- ☐ no se necesitan signos para iniciar el texto
- ☐ comillas dobles

Which tag needs to be in the HTML to run Javascript code?

- ◯ <js>
- ◯ <src>
- ◯ <script>
- ◯ <body>

¿Qué palabras declaran una variable en JavaScript?

- ☐ var
- ☐ const
- ☐ for
- ☐ let

Can a function in JavaScript have more parameters?

- ◯ yes
- ◯ no

¿ Qué nodos podemos usar para navegar entre <html> y <head>?

- ☐ hermano
- ☐ firstChild
- ☐ sisterNode
- ☐ parentNode

Which sign do we need to put in this example if we want to find if the value of x is 5 and the type of the variable is not important? if (x ??? 5) {};

- ◯ ==
- ◯ =

- ○ ===
- ○ ====

¿ que relaciones puden tener <body> y <head>?

- ☐ previousSibling
- ☐ nextSibling
- ☐ firstChild
- ☐ parentNode

What is the first object in HTML DOM?

- ○ <p>
- ○ <html>
- ○ document
- ○ <head>

¿Qué es un nodo de este ejemplo?

- ☐ elemento HTML
- ☐ texto en elementos HTML
- ☐ atributo de un elemento HTML
- ☐ documento html completo

We want to change <p> tag. Which method can we use if there is no id in <p> tag?

- ○ setAttribute
- ○ getElementsByClassName
- ○ getElementById
- ○ getElementsByTagName

¿Qué métodos podemos utilizar para crear un nuevo párrafo con nodos?

- ☐ value
- ☐ createTextNode
- ☐ getElementById
- ☐ createElement

Which elements can we use for creating new HTML elements?

- ◯ setAttribute
- ◯ createElement
- ◯ getElementById
- ◯ appendChild

¿Qué métodos podemos utilizar si queremos eliminar un elemento?

- ☐ removeParent()
- ☐ removeSibling()
- ☐ remove()
- ☐ removeChild()

Qué métodos necesitamos si queremos crear un nuevo <td> en un nuevo <tr>?

- ☐ insertRow()
- ☐ checked()
- ☐ insertCell()
- ☐ deleteCell()

Why is innerHTML used?

- ◯ to access the content of an HTML document
- ◯ to access the content of a <html> tag
- ◯ to get to a child node from the current node

What does appendChild() method do?

- ◯ appends new child to the child
- ◯ appends new element to the parent
- ◯ appends child element to a sibling

What does childNodes[1] mean?

- ◯ it will take the second element of the kind we are looking for
- ◯ it will take the first element of the kind we are looking for
- ◯ it is the same as lastChild

Which method can we use if we want to add a new element to existing element on the last place?

- ◯ appendChild
- ◯ insertBefore
- ◯ replaceChild

Which option can we use in a form to call a function?

- ◯ action
- ◯ onsubmit
- ◯ href

What property can we use to access the content of the field in a HTML form?

- ◯ isNaN
- ◯ year
- ◯ value

With what can we access the number of the selected option in a drop-down list?

- ◯ checked
- ◯ selectedIndex
- ◯ value

What can we use to see if the box in the form has been selected?

- ◯ checked
- ◯ selectedIndex
- ◯ value

Can we change CSS with DOM without using events?

- ◯ yes
- ◯ no

With what property can we change the text color?

- ◯ style.font-color
- ◯ style.color
- ◯ style.background-color

With what property can we change the background color?

- ◯ style.background

- ◯ style.color
- ◯ style.background-color

How can we create a new <tr> in a table?

- ◯ insertCell()
- ◯ insertRow
- ◯ insertTableRow