JavaScript DOM

Lili Nemec Zlatolas

Annotation

This course introduces the document object model in JavaScript.

Objectives

The course provides a quick overview of prior knowledge needed, such as the basics of HTML and JavaScript. The pupil will learn on Document object model in JavaScript.

Keywords

JavaScript, DOM, HTML

Date of Creation

15.4.2022

Duration

12 hours

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

Basics of HTML

To manipulate websites via Document object model (DOM), some prior knowledge of HTML and JavaScript is required. HTML is a markup language that describes the structure of a website. HTML consists of elements that are labelled and send the information to the browser on how to display the content of the document.

The last accepted standard is HTML5. HTML consists of elements that are defined by a start tag, content of the element and end tag: <tag> Content </tag>

Here is an example of a HTML document:

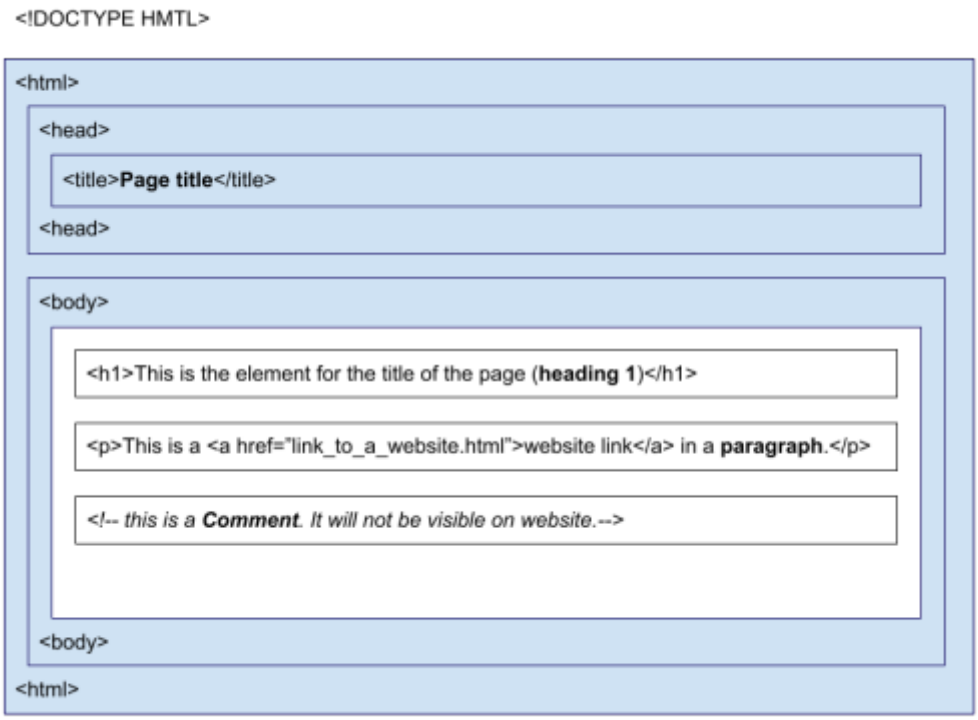

Fig. 1. HTML page structure and some elements

To edit the documents, you can use Notepad or similar programs (e.g. Notepad++, Visual Studio Code). The HTML document needs to have .html as a file extension.

HTML is not case sensitive, but it is recommended to use lowercase in HTML tags.

Table 1. Some of the basic HTML elements

Tag	Information
`<! DOCTYPE html>`	This is the declaration for HTML5. It should appear at the top of the HTML document.
`<h1>`	There are HTML headings from <h1> to <h6>, the first one being the largest heading.
`<p>`	This is a tag for a paragraph.
` `	This is a tag for a break into the next row.
`<a href="https://www.websitelink.com">`	This is a tag for a website link. It uses attributes.
`<img src="image_location.jpg" alt="image_description">`	This is a tag for inserting an image. It also uses different attributes.
`<b>`	This is a tag for bolding the text.

Some elements are called empty elements. An example of an empty element is
, which does not have content or the end tag. It could however include the end tag inside the start tag if we are using XHTML guidelines:

HTML tags can have attributes. Some elements do not have any functions without the attributes. Attributes appear in start tags. One of the often used attributes is style, which can be used for example in paragraph:

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

Interaktivní prvek

HTML tables are often used and consist of cells inside table columns and rows. A simple HTML table is:

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

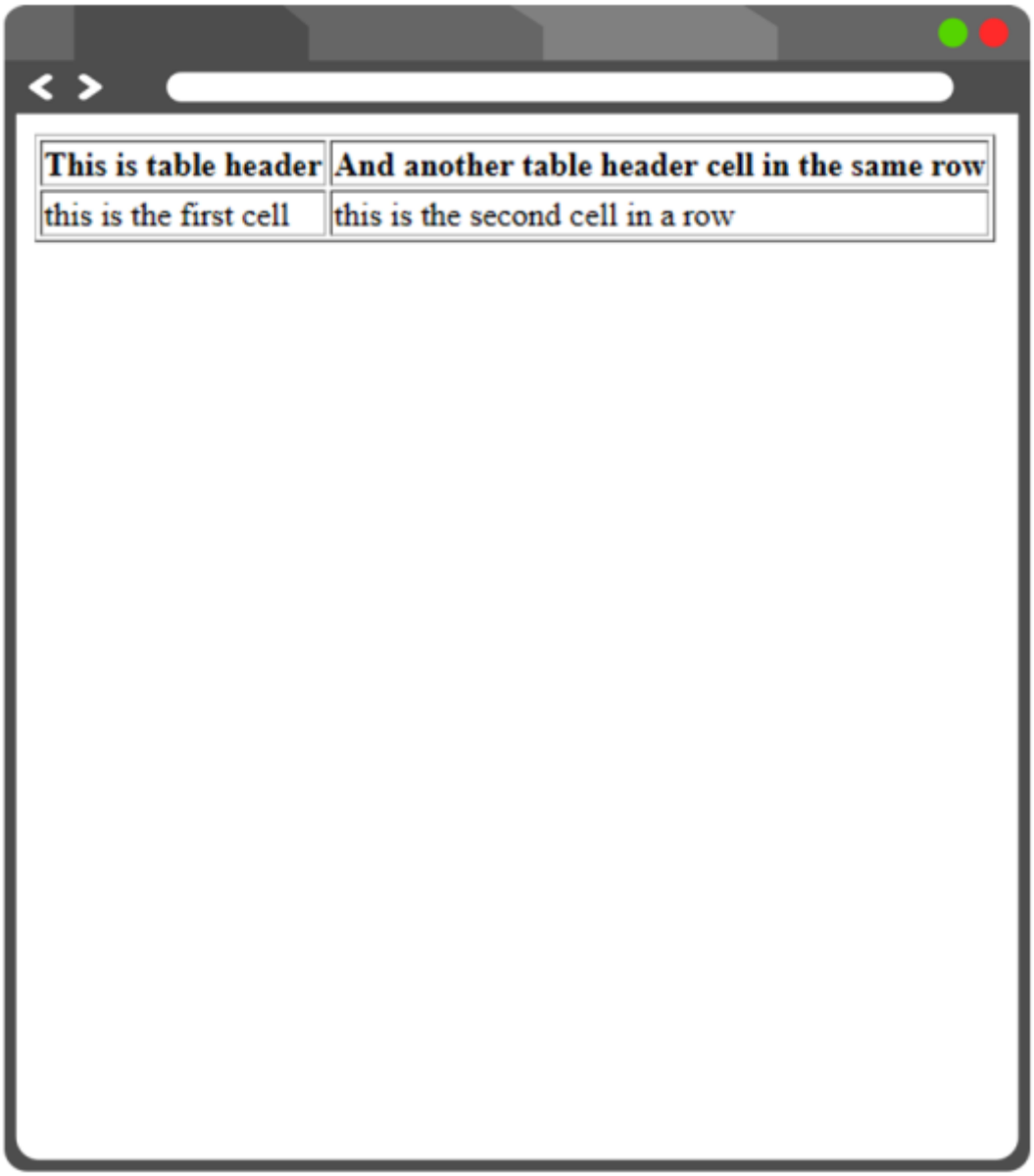

Fig. 2. How the table looks in the browser.

HTML lists are for creating ordered or unordered HTML lists. Ordered lists are numbered and unordered lists normally have dots.

Table 2. HTML lists example

List example	Browser view

`<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

This was a short introduction to HTML. In the next chapter, we will look into some basics of JavaScript syntax.

CHAPTER 2

Basics of JavaScript

JavaScript is a programming language that can make websites more interactive and dynamic. It is used in HTML between <script> and </script> tag either in <body> or <head> section of HTML page or both.

JavaScript can also be placed in html file as external file with .js extension. Example:

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

There are a few possibilities of how to display JavaScript data. We can use:

innerHTML (e.g. document.getElementById(id))

- We will look at this methods in the next chapters, because this is DOM

document.write()

- This is normally used for testing, because it deletes all existing HTML from the document and only displays the script content.

window.alert()

- It creates an alert box to display data. Window can be omitted.

console.log()

- This is normally used for debugging purposes and displays data in the browser.

JavaScript code is separated by semicolons (**;**) at the end of each executable code.

Code blocks are grouped together inside **{curly brackets}**.

Interaktivní prvek

Single line comments use // and multiple line comments use /* to start and */ to finish.

Some of the most important keywords in JavaScript are presented in Table. These are the reserved words that cannot be used for names of variables.

Table 3. Reserved words in JavaScript

Reserved words / keywords	Description
`var`	Declares a variable, which can be re-declared and updated.
`let`	Declares a block variable that can be updated, but not re-declared.
`const`	Declares a variable that cannot be updated or re-declared.

`return`	Exits a function
`if`	Start of the condition code
`for`	Start of the loop code
`function`	Declares a function

Interaktivní prvek

JavaScript syntax are rules of how the programs are constructed. First, the variables are created with declaring them and then they are used in the program.

Decimal **numbers** are separated by a **dot**. **Texts** are written withing **single or double quotes**. **Equal signs** assign values to variables.

Variable names must begin with letters of the alphabet (A-Z), a dollar sign or with an underscore. They are **case sensitive**.

Example of creation of different variables:

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

Table 4. Equal signs in JavaScript

Equal sign	Description
=	This is an assignment operator (e.g. **var a = a * 2;**) and does not have the same meaning as algebra sign.
==	This is the equal operator that is used in algebra for example in 7+2=9, and in JavaScript it is used as e.g. **if (x == 2) {};**
===	This is the equal value and also equal type.

You can use = and + to concatenate or calculate the variable value.

```
let wholeName = "Maya" + " " + "Surname";
```

Interaktivní prvek

JavaScript has 3 logical operators:

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

Functions are blocks of code that need to be called to be executed. They can have more parameters and the code is within curly brackets. They are useful when we use the functions more times with different arguments. Example of a function:

EXAMPLE

```
function circle_area( a, b)
{
return a * a * b;  // Function returns the product of a, a and b. If we
want to calculate the area of circle, we need to enter PI number instead
of B.
}
let area = circle_area (15, Math. PI);  // Function is called with the
length 15 cm and number PI.
document. write( "The area of circle is " + area + " m2. ");
```

When running the above function, the browser will display: "**The area of circle is 706.8583470577034 m2.**"

There is a lot more to learn in JavaScript, but this are some basic and important things we need to start with JavaScript DOM.

CHAPTER 3

Introduction to Document Object Model

The browser creates a Document Object Model (DOM) when it loads a website. DOM is a W3C (World Wide Web Consortium) standard for accessing documents.

The HTML DOM is constructed as a tree of Objects.

An example of the following code is presented in the animation.

```
<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. A tree of Objects in HTML DOM model

A tree has **nodes** for elements, which represent HTML tags and determine the structure of the document. The standard is designed so that we first create a node, then add a child to it and attributes to the child. The code can therefore be quite long.

In the animation, <html> is the *root node* without parents but is a *parent* to the *first child* <head> and *last child* <body>.

JavaScript can access DOM and change elements and attributes in the HTML page.

In DOM, all HTML elements are defined as **objects**.

DOM is a standard where information on how we get to HTML elements, how we change them, add them or delete them in HTML documents. DOM uses **methods** to access and do actions on HTML elements.

Interaktivní prvek

3.1 DOM navigation

The nodes in the node tree have a hierarchical relationship, meaning that every node has exactly one parent, except the first node, which has no parent. A node can have more children. And sibling nodes are nodes with the same parent.

To navigate between nodes with JavaScript we can use the following properties:

- parentNode
- childNodes[nodenumber]
- firstChild
- lastChild
- nextSibling
- previousSibling

Animation 2. Example of nodes in a document

Property childNodes[0] is the same as firstChild. It holds an array-like object with the length property with which it accesses the child nodes.

Interaktivní prvek

We will look more into nodes and how to get the content from elements in the chapter DOM Nodes.

CHAPTER 4

DOM Methods

With DOM **Methods** we can perform actions on HTML elements. DOM **properties** are values of HTML elements that we can set or change.

EXAMPLE

```
<p id="example">This will not be shown on the page, because Javascript
will overwrite the text inside the paragraph node. </p>
<script>
document.getElementById("example").innerHTML = "This is the text that
will be displayed on the website. ";
</script>
```

In this example, "**This is the text that will be displayed on the website.** " will be shown in the browser when the code will be executed.

In this example, getElementById is a **method** and innerHTML is a **property**. Very often, id (in the example id=”example”) is used to find the element in the document. Otherwise, we can use parent and child nodes to get to certain elements.

Property innerHTML is used to set or return HTML content of an element, in the above example, it is changing the text within the <p> tag with the h id element.

Table 5. Methods for finding, changing, adding and deleting elements HTML elements

Method	Description
document.getElementById(id)	Finding element by element id
document.getElementsByTagName(name)	Finding element by tag name
document.getElementsByClassName(name)	Finding element by class name
element.setAttribute(attribute, value)	Changing the attribute value of an HTML element
document.createElement(element)	Creating an HTML element
document.removeChild(element)	Removing a child HTML element
document.appendChild(element)	Adding a child HTML element
document.replaceChild(new, old)	Replacing a child HTML element

Interaktivní prvek

Video 1. Examples of get element by X

Interaktivní prvek

CHAPTER 5

DOM Nodes

All **HTML elements, their attributes and texts are nodes**. Some elements contain other nodes.

Navigation between nodes was already described in chapter DOM navigation.

Here is an example of how we can access the value of the nodes.

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

The element <p> contains a **text node** with the value „This is the first paragraph.“ The value of the text can be accessed by node's innerHTML property:

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

The same can be done by accessing the **nodeValue**:

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

Video 2. Child nodes and node values

Root nodes can access the full document:

- document.body – The body of the document
- document.documentElement – The full document

Property **nodeValue** specifies the value of a node. It is null for element nodes, but it is useful for text nodes, where it presents the text itself. The property for attribute nodes returns the attribute value.

Property **nodeName** specifies the name of the node and it is read-only. nodeName of an element node returns tag name and it returns attribute name of an attribute node. nodeName of a text node is *#text* and of the documents node is *#document*.

We will use some of the following methods in the subchapters for creating, removing and replacing DOM HTML elements (Nodes). In this table, we will see the methods for creating elements and text nodes.

Table 6. Methods for creating new DOM HTML elements

Method	Description
createElement(type)	Creates an element node with specific type (e.g. type “p”).
createTextNode()	Creates a text node.

In the next table, methods for creating, removing and replacing nodes are presented. Normally, we have to create an element, then create a text node within it and add this to an existing structure. We will see examples of use in the next subchapters.

Table 7. Methods for creating, removing and replacing Nodes

Method	Description
appendChild(node)	Adds a new child element (node) to element.
insertBefore(new, existing)	Inserts a child node before an existing child.
replaceChild(new, old)	Replaces a child node with a new node.
remove()	Removes an element from the document. Has no parameters.
removeChild(node)	Removes an element's child.

5.1 Creating DOM HTML elements (Nodes)

In this chapter, we will look at how we can add, remove, or replace HTML DOM elements.

To add a new element, we must first create the element node and then add it to the existing element. Here is an example of the code where we added a new paragraph with nodes.

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>
```

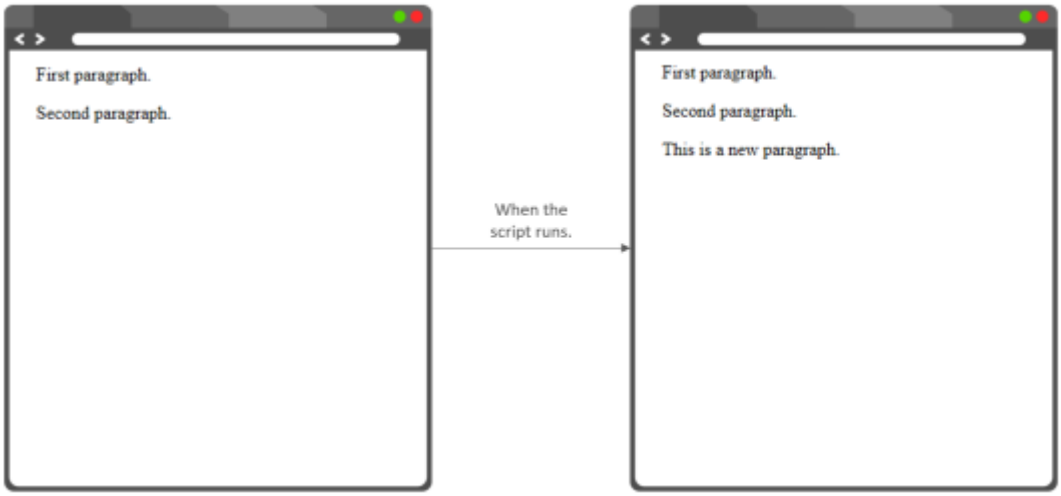

Fig. 3. How the code looks in a browser without the script and with the script.

createElement("p") will create a new <p> element. To add text to this element, we must create a text node with **createTextNode**. Then, we have to add the text node to the <p> element with **appendChild**. Last, we must add the new element to an existing div element.

Now, we will use the same example, but insert the paragraph in the first position with insertBefore instead of the last position as we did with appendChild.

Video 3. Example of insertBefore method

Interaktivní prvek

5.2 Replacing DOM HTML elements (Nodes)

To replace an element, we must first create the element node and then replace it with the existing element. Here is an example of the code where we replaced a new paragraph with the old paragraph.

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>
```

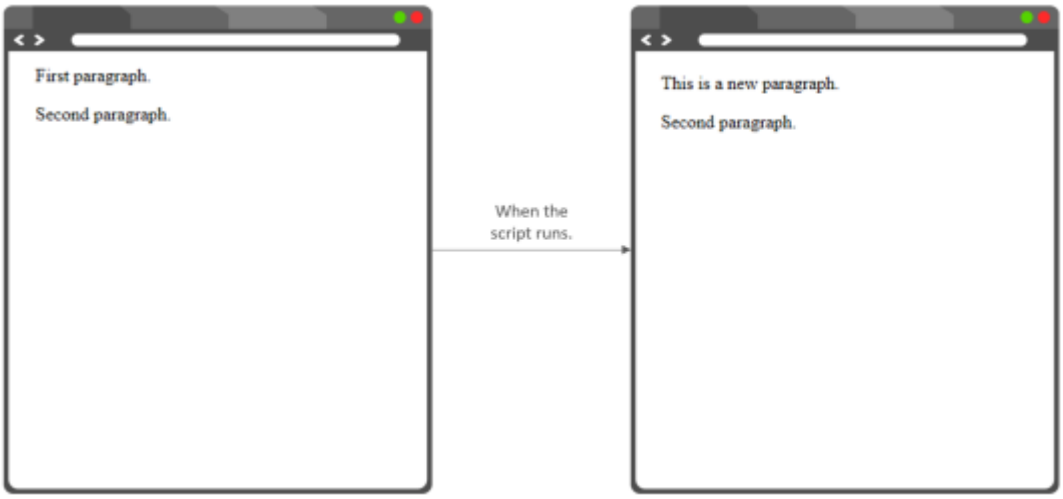

Fig. 4. How the code looks in a browser without the script and with the script.

Interaktivní prvek

5.3 Removing DOM HTML elements (Nodes)

In this chapter we will see an animation where we will remove one element with remove() method and we will remove one element via accessing the parent with removeChild() method. The remove() method does not work in older browsers and that is why sometimes we need to use removeChild().

In removeChild, we need to look for the parent, to get the child removed.

Animation 3. Removing DOM HTML elements with two examples

Interaktivní prvek

CHAPTER 6

DOM Form Validation

We can do HTML form validation by JavaScript via DOM. Normally, we want to check whether the user has filled in the correct data in the correct format. We want to ensure that the user's input is correct. In this example, we will create a simple form and check if the user has entered the correct data with a function, using DOM.

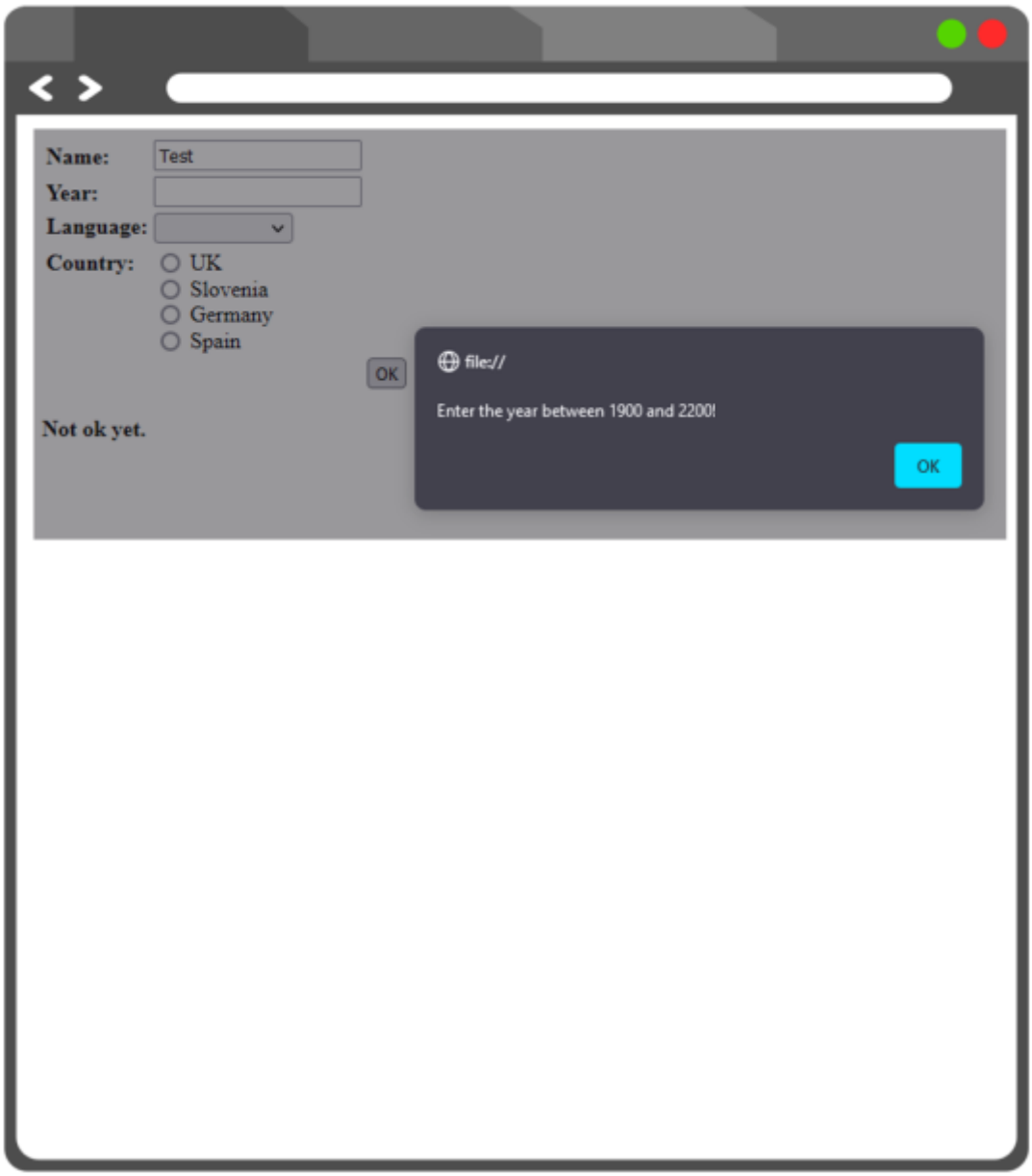

Fig. 5. Example of an alert and form that will be presented in the code below.

EXAMPLE

We are creating a form with 4 different options. We have used a HTML table for a better display.

In the form, text, option and input with type radio button were used as well as a submit button.

```
<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">
```

Here, a validation function is created. When something will not be correctly filled in by the user, an alert window will appear.

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

With DOM, we have accessed the form, name input and its value. If this is empty, an alert will appear.

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

Next, if the year entered is out of range 1900 and 2200, the alert will appear. With isNaN we also check if the user has entered a number at all.

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

Next, if for the language dropdown list, the user did not select any index, the alert appears.

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

Next, if the user did not select any of the countries with radio button, alert will appear.

```
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 changing CSS

Often, DOM is used to change the style of HTML elements. The syntax for it is:

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

Usually, a visitor of a website should click on some element like a button and the changing of the CSS style in the document can occur. These are also called DOM events. The syntax for it is:

onclick=JavaScript

Interaktivní prvek

In the example below, we will look at some of the properties and events we can use.

Animation 4. Example of DOM changing CSS

EXAMPLE

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

This button below will change the color of the text above into green color.

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

This button below will change the font size of the text above into 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>
```

This button will call a function called date that is in the <script> part and will display current time in the above paragraph.

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

The below input presents a box, where the user enters color. The next input is a checkbox and the last has a function that will change the background to the text written in a box.

```
<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>
```

This function will display the current date and time.

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

This function will change the background color to what the user has entered into a box and only if the checkbox is checked.

```
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 changing tables

In this example, we will do a to-do list with an HTML table. We can manipulate tables by adding or deleting rows, headers, cells etc.

Table 8. Methods for adding, deleting different elements in tables

Method	Description
deleteRow()	Removes a <tr> from a table.
insertRow()	Creates an empty <tr> element in a table.
deleteCell()	Deletes cell from the current table row.
insertCell()	Creates a cell into a current table row.

In the example below, we will create a to-do list where we can add and remove tables from a table by clicking on the checkbox.

EXAMPLE

First, we will create two buttons and a text field, where a user can add new task, delete new task or write what the task is. The functions are called via onclick and are in the <script> section.

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

This is a simple table with only table head, without any tasks. Those will be added via functions.

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

<script>
```

First, we will create a function for adding new tasks to the tables. We need to look for the table and in this example, we are using getElementById.

```
var count_lines=0;
function addTask() {
```

```
 var table = document.getElementById("table");
```

Next, inserting the rows from the bottom to top needs to be done with the insertRow method and -1 (so the task goes to the bottom). Afterwards, three cells are inserted into the table with three columns.

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

Next, we are adding 1 to count_lines variables for the second column that counts the current number of tasks added.

```
count_lines++;
```

In the first cell, a checkbox is inserted. In the second cell, the line counter is inserted and in the third box, the text value from the textbox is inserted.

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

Next, we will create a function for deleting finished tasks to the tables. We need to look for the table again and in this example, we are using getElementById.

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

Next, we have created an i variable to find the row where the checkbox is checked and then we can delete the row the user has selected as done. We have used chilNodes and checked if the checkbox is checked.

```
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

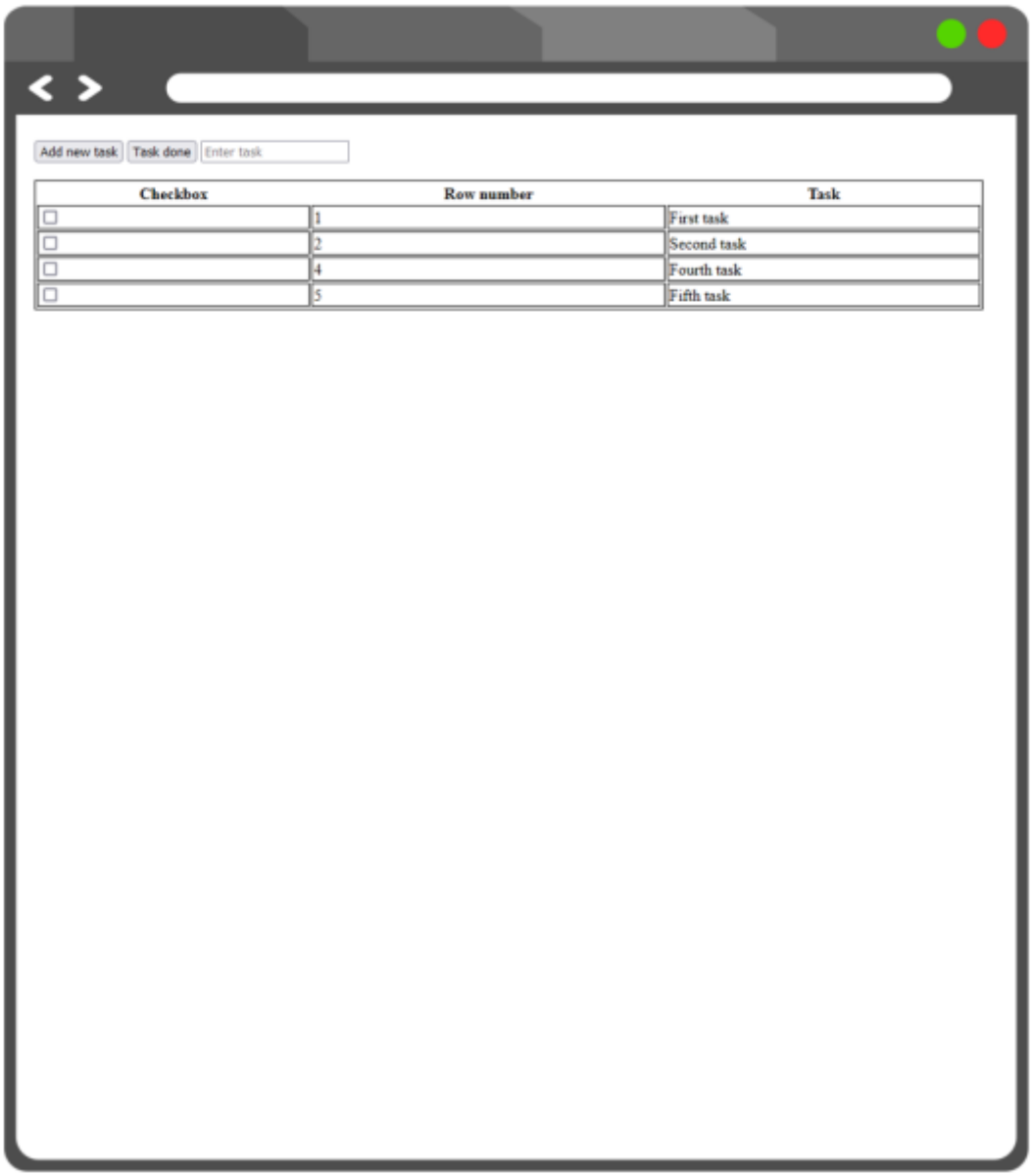

Fig. 6. How the to-do list looks in the browser.

Interaktivní prvek

This was an introductory course to JavaScript DOM.

CHAPTER 9

Test

Where does the <title> tag appear?

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

What can HTML elements consist of?

- ☐ username
- ☐ start tag
- ☐ content
- ☐ end tag

Which tag is for making the text bold?

- ◯ <u>
- ◯ <i>
- ◯ <b>
- ◯ <q>

What is an empty element?

- ☐ an element <>
- ☐ an element that does not have an end tag

- ☐ an element that does not have content
- ☐ an element without an attribute

What good practice says about where to put <script> element in?

- ☐ in <head> tag
- ☐ in <table> tag
- ☐ in <body> tag
- ☐ before <!DOCTYPE html> tag

How do you create a comment in HTML?

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

Which tag creates a list with dots in front?

- ◯
- ◯ <dl>
- ◯
- ◯

Text in JavaScript is written within:

- ☐ backslash
- ☐ single quotes
- ☐ no signs are needed to start the text
- ☐ double quotes

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

- ○ <js>
- ○ <src>
- ○ <script>
- ○ <body>

Which words declare a variable in JavaScript?

- ☐ var
- ☐ const
- ☐ for
- ☐ let

Can a function in JavaScript have more parameters?

- ○ yes
- ○ no

Which methods can we use to navigate between <html> and <head>?

- ☐ sibling
- ☐ 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) {};

- ○ ==
- ○ =

- ○ ===
- ○ ====

What relationships can <body> and <head> have?

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

What is the first object in HTML DOM?

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

What is a node from these example?

- ☐ HTML element
- ☐ texts in HTML elements
- ☐ attribute of an HTML element
- ☐ the whole html document

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

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

Which node methods can we use to create a new paragraph?

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

Which elements can we use for creating new HTML elements?

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

Which methods can we use if we want to remove an element?

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

Which methods do we need if we want to create a new <td> in a new <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