Web — Master 1 IFI
Lab Session #4: Client-Side Programming

Andrea G. B. Tettamanzi
Université côte d'Azur
andrea.tettamanzi@univ-cotedazur.fr

Academic Year 2019/2020

Abstract

The goals of this lab session are familiarize ourselves with DOM object manipulation and HTML event handling.

Introduction

We will write a JavaScript program to generate on-the-fly a table of contents for the HTML document in which it is included. To do this, we will proceed step by step.

Here is a sample HTML document you can use to test your script, which will have to be placed in a file named `toc.js`:

```
<!DOCTYPE html>
<html lang="fr">

  <head>
    <title>Document de test</title>
    <script src="./toc.js"></script>
    <meta charset="ISO-8859-1" />
  </head>

  <body>
    <h1>Introduction</h1>
    <p> bla </p>
    <h1>Section 1</h1>
    <h2>Section 1.1</h2>
    <p> bla </p>
    <h2>Section 1.2</h2>
    <p> bla </p>
    <h1>Section 2</h1>
    <p> bla </p>
    <h1>Section 3</h1>
    <p> bla </p>
    <h1 id="ccl">Conclusion</h1>
    <p> bla </p>
  </body>

</html>
```

1 Basic Table of Contents

Write a function, for instance `toc()`, which, based on the first-level headings (`h1`) found in the document, generates an HTML table at the beginning of the document in which all the first-level headings of the document are listed.

Some useful DOM methods are:

- `var element = document.createElement(tagName);`
- `var elements = document.getElementsByTagName(name);`
- `var aChild = element.appendChild(aChild);`
- `var insertedNode = parentNode.insertBefore(newNode, referenceNode);`

The `innerHTML` property of an HTML element will also be useful.

Use the following event handler to make sure the construction of the table of content begins once the document has finished loading:

```
window.onload = toc;
```

2 Linked Table of Contents

Write a second version of this function where each element of the table is encapsulated in an anchor tag, with an `href` attribute allowing to jump from the table of contents to the selected first-level heading.

To this aim, each first-level heading will have to be uniquely identified by an `id` attribute (like "Conclusion" in the sample document). For all first-level headings missing such attribute, the `toc()` function will have to generate it.

3 Highlighting the Selected Heading

Write a third version of the `toc()` function to dynamically change the style of a first-level heading when the mouse pointer hovers the reference to it in the table of contents. For instance, the background color of the heading might become yellow as long as the mouse stays over its reference in the table of contents and go back to the original color as soon as the mouse leaves that area.

To implement this new version of the function, you could use the `EventTarget.addEventListener()` method, as well as the `mouseover` and `mouseleave` events.

Adding stylesheet rules will have to be done programmatically, by addding a `<style>` tag in the head of the document. An attribute can be dynamically assigned to the selected heading by means of the `setAttribute` method so that the style can be applied to it while it is selected.

4 Processing Second-Level Headings

Write a fourth version of the `toc()` function to include in the table of contents second-level headings as well. Make the highlighting style for second-level headings different from first-level headings. For instance, second-level headings might become green (instead of yellow) when highlighted.