Poznań University of Technology

Faculty of Control, Robotics and Electrical Engineering

Institute of Robotics and Machine Intelligence

Division of Control and Industrial Electronics

Web application
HTML and JavaScript

Mobile and embedded applications for Internet of Things

Teaching materials for laboratory

Dominik Łuczak, Ph.D.; Adrian Wójcik, M.Sc.

Dominik.Luczak@put.poznan.pl
Adrian.Wojcik@put.poznan.pl

I. Goal

Knowledge

The aim of the course is to familiarize yourself with:

- the role of HTML in web applications,
- syntax of HTML and basic HTML tags,
- HTML document structure,
- the role of the Document Object Model in web applications,
- the role of JavaScript in web applications,
- basics of JavaScript language syntax.

Skills

The aim of the course is to acquire skills in:

- creating HTML documents,
- creating scripts in JavaScript language,
- communicating JS scripts with HTML documents.

Social competences

The aim of the course is to develop proper attitudes:

- proper management of web application code base,
- strengthening understanding and awareness of the importance of non-technical aspects and effects of the engineer's activities, and the related responsibility for the decisions taken,
- choosing the right technology and programming tools for the given problem,
- testing developed IT system.

II. Laboratory report

Complete laboratory tasks as per the instructor's presentation. Work alone or in a team of two. **Keep safety rules while working!** Prepare laboratory report documenting and proving the proper execution of tasks. Editorial requirements and a report template are available on the *eKursy* platform. The report is graded in two categories: tasks execution and editorial requirements. Tasks are graded as completed (1 point) or uncompleted (0 points). Compliance with the editorial requirements is graded as a percentage. The report should be sent as a *homework* to the *eKursy* platform by Sunday, May 16, 2021 by 23:59.

III. Prepare to course

a) Know the safety rules

All information on the laboratory's safety instructions are provided in the laboratory and on Division website [1]. All inaccuracies and questions should be clarified with the instructor. It is required to be familiar with and apply to the regulations.

Attend the class prepared. Knowledge from all previous topics is mandatory.

B) Introduction to HTML

HTML (*HyperText Markup Language*) is the most basic component of the World Wide Web. HTML defines **structure** of web content. (*Hypertext*) refers to links that connect web pages to each other, both within single domain page and between different domains. The HTML element is distinguished from plain text in the document by *markup* or more commonly 'tags', which consist of the name of the element surrounded by „<" and „>". The element name inside the tag is not case-sensitive. This means that it can be in uppercase, lowercase or mixed letters. For example, the `<title>` tag can be also written as `<Title>`, `<TITLE>`, or in other ways [2].

Currently (since May 2019) the WHATWG organization (*Web Hypertext Application Technology Working Group*) is the publisher of HTML and DOM standrads. The language standard can be found on the WHATWG page [3] . The current version of HTML language is HTML5 [4]. A list of tags with descriptions and examples is available on the devdocs.io portal [5].

Separate technologies apart from HTML used to create web applications are: **CSS** used to describe the appearance/presentation of the website and **JavaScript** used to describe the functionality/behavior of the application [2].

C) Document object model

DOM (*Document Object Model*) is an API that represents HTML documents that allows to programmatically communicate with them. DOM is a model of a document presented in a browser represented as a tree, in which each *node* represents a part of the document (see: fig. 1).

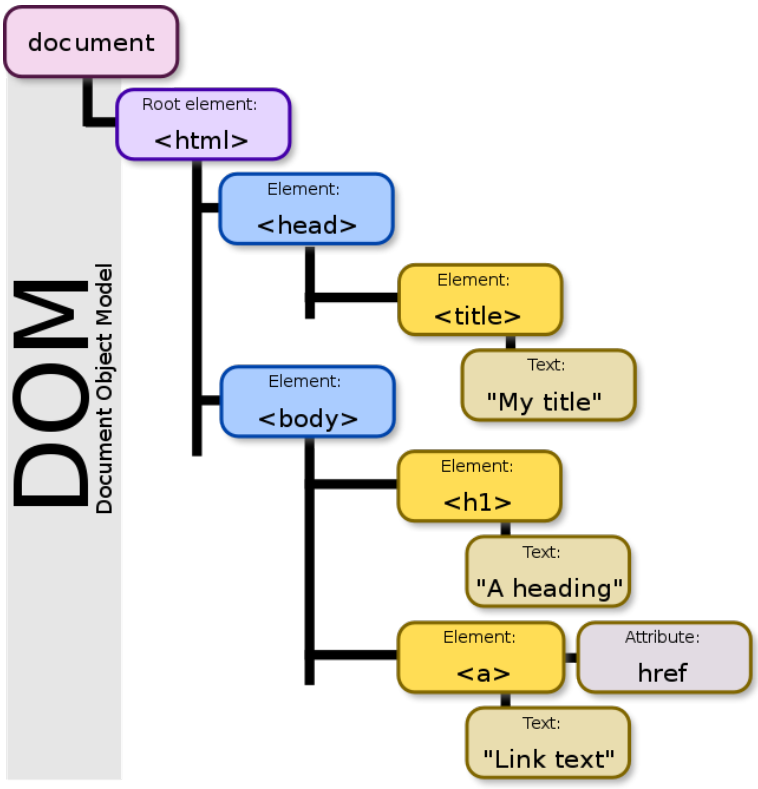

Fig. 1. An example of the DOM hierarchy in an HTML document [6].

The introduction of a structural representation of the document, allows to modify its content and visual presentation. DOM is one of the most commonly used APIs in the WWW because it allows access and complex interaction with *nodes* of the document. These nodes can be created, moved and changed.

Also, event listener can be add to nodes. The DOM therefore combines web pages with JavaScript (or other programming languages). The DOM standard can be found at WHATWG website [7].

d) Introduction to JavaScript

JavaScript (in short: *JS*) is a scripting programming language - interpreted or compiled using the JIT (just-in-time) method . A distinctive feature of JS is that functions are „first-class citizens", i.e. objects that can be stored as references and passed as any other objects. JavaScript is a multi-paradigm language: prototype-based, dynamic syntax, object-oriented, imperative and declarative (functional programming) [8].

The standard for JavaScript is ECMAScript [9]. In June 2015, ECMA International published the sixth major version of ECMAScript. Since 2012, all modern browsers completely support ECMAScript 5.1. Older browsers support at least ECMAScript 3. Currently, ECMAScript standards are issued on an annual basis.

Do not confuse JavaScript with the Java programming language. Both „Java" and „JavaScript" are trademarks of Oracle. However, both of these programming languages have very different syntax, semantics, and uses. This name is primarily the result of marketing efforts.

JavaScript is best known as a scripting language for websites, but it is also used by many non-browser environments such as Node.js, Apache CouchDB and Adobe Acrobat.

1. JavaScript in HTML document

On the listing 1. an example of an HTML document with a script written in JavaScript language is presented. Script allows handling of the event in the form of pressing a button. Both HTML and JS are in single **.htm** file.

Listing 1. An example of an HTML document with JavaScript in single file.

```
<!DOCTYPE html>
<html>
  <!-- HTML file head: web page metadata -->
  <head>
    <title>HTML/JavaScript examples</title>
    <script>
      var buttonCounter = 0;
      function myOnClickMethod() {
        buttonCounter += 1;
        var paragraphDispText ="<b>Click␣counter:␣</b>"+buttonCounter.toString();
        document.getElementById("paragraph").innerHTML = paragraphDispText;
      }
    </script>
  </head>
  <!-- HTML file body: web page content -->
  <body>
    <h1>Simple button example: signle file</h1>
      <button onclick="myOnClickMethod()">Click me</button>
      <p id="paragraph"></p>
  </body>
</html>
```

When creating web applications, however, it is good practice to separate the description of *document structure* (HTML) from *software functionality* (JavaScript). On the listings 2. and 3. are presented the **.htm** file (containing information about used **.js** file) and the **.js** file, respectively. Of course, both presented web applications work in an identical way.

Listing 2. An example of an HTML document with JavaScript in a separate file.

```
<!DOCTYPE html>
<html>
  <!-- HTML file head: web page metadata -->
  <head>
    <title>HTML/JavaScript examples</title>
    <script type="text/javascript" src="scripts/button_example_v2.js"></script>
  </head>
  <!-- HTML file body: web page content -->
  <body>
    <h1>Simple button example: separte HTML and JavaScript</h1>
      <button onclick="myOnClickMethod()">Click me</button>
      <p id="paragraph"></p>
  </body>
</html>
```

*Listing 3. An example of JavaScript in a separate **.js** file.*

```
var buttonCounter = 0; ///< button onClick event counter

/* @brief button onClick event handling */
function myOnClickMethod() {
  buttonCounter += 1;
  var paragraphDispText = "<b>Click␣counter:␣</b>" + buttonCounter.toString();
  document.getElementById("paragraph").innerHTML = paragraphDispText;
}
```

Using the `<script>` tag, external JS libraries that are available both locally and as an online resource, can be add to document.

2. JSON in JavaScript

The JSON (*JavaScript Object Notation*) data exchange format is derived directly from JavaScript - JSON objects are **textual** representation of objects in JavaScript. The function enabling parsing, i.e. obtaining a JS object from text in JSON format (`JSON.parse()`) and encoding, i.e. obtaining text in JSON format from a JS object (`JSON.stringify()`) are both part of the JS language and do not require use of additional libraries. On listing 4. an example use case of these functions is shown.

Listing 4. An example use case of functions `JSON.parse()` and `JSON.stringify()`.

```
const jsObjExample = {
    temperature: { value: 20.3,   unit: "C"   },
    pressure:    { value: 1023.0, unit: "hPa" },
    humidity:    { value: 63,     unit: "%"   }
};

const jsonTextExample ='{"temperatur":{"value":20.3,"unit":"C"},␣"pressure":{"
    value":1023.0,"unit":"hPa"},"humidity":{"value":63,"unit":"%"}}';
var jsObj = JSON.parse(jsonTextExample);
var jsonText = JSON.stringify(jsObjExample);
```

3. JavaScript code debugging in web browser

Most modern browsers contain *web console*, that allows to view variables and execute JavaScript code in the current *context*. There are also tools available that allows full debugging of the web application directly in the browser - e.g. Firefox or Chrome. Fig. **??** shows an example of debugging a simple button application.

Fig. 2. An example of debugging a JS script in the Chrome browser - stopping script execution using breakpoint in line 8. after pressing „Click me" button.

IV. Scenario for the class

A) Teaching resources

Hardware
- computer,

Software
- web browser (eg. Firefox, Chrome),
- text editor (eg. Notepad++),
- web server (eg. XAMPP, Lighttpd) (optional),

B) Tasks

1. Create a simple HTML document with a single button.
   (a) Using a text editor, create HTML document (*.htm) and JavaScript script (*.js).
   (b) Place the header section and body section in the HTML document. Define documents title, add script information, and put a single „CLICK ME" button in page body. Assign the name of your JavaScript function to `onClick` attribute.
   (c) In the JavaScript script, define the function to handle button clicking. The function should change the content of the button to „HELLO WORLD" and increment some global variable.
   (d) Open the HTML document using a browser and check if the defined functionality works correctly.
   (e) Open the browser's web console and check that the global variable defined in the script changes the value correctly each time the button is clicked.
   (f) Alternatively, place the document and script on the local server, then open the document by entering the appropriate URL in your browser.
2. Create a simple web application: unit converter.
   (a) Create an HTML document containing elements that allow the user to enter a numerical value (e.g. rotational speed), select the input and output unit (min. four diffrent units, e.g. rpm, rps, rad/s, rad/min, deg/s, deg/min), confirmation of input data and presentation of the result. Use the `input`, `select` and `output` tags.

(b) Create a JavaScript script for unit conversions. Try to implement this functionality without using conditional statements.

3. Modify a simple web application using JSON as input and output.

   (a) Based on the solution to the previous task, create a web application that converts units (e.g., rotational speed) based on the structure provided by the user in JSON format. The structure must contain information about the input value, input unit and output unit. Include an example structure as the starting value for the text input element.

   (b) The application output should also be presented in JSON format, containing information about the output value and output unit.

   (c) Validate the input data, inform the user about any errors (e.g. invalid JSON format, incorrect input value, incorrect / unavailable input unit, incorrect / unavailable output unit).

4. Create a web application containing a function graph using an external JavaScript library.

   (a) Familiarize yourself with the documentation of the Chart.js library available on the website: chartjs.org.

   (b) Create an HTML document and add the Chart.js library to it with:

```
<script type="text/javascript"
    src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0">
<\script>
```

   Add your own script. The order of placing scripts in the document is important in the context of visibility of the variables declared in the scripts.

   (c) Place a (*line chart*) in the document - based on examples in the Chart.js library documentation. Add elements to allow user input. The document should allow input of information about the sample time, amplitude, phase and period of the harmonic signal.

   (d) In your script, implement the functionality that allows to generate data for the chart based on the harmonic signal parameters provided by the user and update the data and description of the x axis of the chart.

5. Create basic web application consisting of multiple HTML documents.

   (a) Create an HTML document with three elements: a heading (title), 4 hyperlinks, and a frame.

   (b) Hyperlinks should be links to files with solutions to previous tasks. After clicking the hyperlink, the frame should be filled with the content of the appropriate document.

   (c) Answer the question: Is JavaScript necessary to achieve this functionality?

References

1. *Regulations, health and safety instructions* [online]. [N.d.] [visited on 2019-09-30]. Available from: http://zsep.cie.put.poznan.pl/materialy-dydaktyczne/MD/Regulations-health-and-safety-instructions/.
2. *HTML: Hypertext Markup Language* [online]. [N.d.] [visited on 2020-04-14]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTML. Library Catalog: developer.mozilla.org.
3. *HTML Standard* [online]. [N.d.] [visited on 2020-04-14]. Available from: https://html.spec.whatwg.org/multipage/.
4. *HTML5* [online]. [N.d.] [visited on 2020-04-14]. Available from: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5. Library Catalog: developer.mozilla.org.

5. *HTML documentation — DevDocs* [online]. [N.d.] [visited on 2020-04-14]. Available from: https://devdocs.io/html/.
6. *Document Object Model* [online]. 2019 [visited on 2020-04-14]. Available from: https://en.wikipedia.org/w/index.php?title=Document_Object_Model&oldid=932262242. Page Version ID: 932262242.
7. *DOM Standard* [online]. [N.d.] [visited on 2020-04-14]. Available from: https://dom.spec.whatwg.org/.
8. *JavaScript* [online]. [N.d.] [visited on 2020-04-14]. Available from: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Library Catalog: developer.mozilla.org.
9. *Standard ECMA-262* [online]. [N.d.] [visited on 2020-04-14]. Available from: https://www.ecma-international.org/publications/standards/Ecma-262.htm.