HTML, JavaScript and DOM

SCOMRED, October 2018

Support references:

- https://www.w3schools.com/html/
- https://www.w3schools.com/js/

HTML - Hyper Text Mark-up Language

HTML is a specification used to describe the appearance of WEB pages, as they should be presented to users by browsers.

It's based on TAGs, tags have names and they establish elements of the page in the general form:

<tagname>Element Content</tagname>

Each opened tag: <tagname> should be closed: </tagname>, the tag's content is what lies between them.

In HTML, some tags can´t have content, and those don't have to be closed. Nevertheless, tags with no content, can be explicitly self closed when opened:

<tagname />

In HTML tag names are not arbitrary they are meaningful for the browser. The whole page content must be within the HTML tag, this is the document's root tag for HTML:

<html>Document content</html>

HTML - <head> and <body>

To make things easier for browsers, the HTML content should be prefixed by the DOCTYPE declaration, in this case declaring it's an HTML content. The HTML content itself is split into two parts, head and body.

```
<!DOCTYPE html>
<html>
<head>Head Content</head>
<body>Body Content</body>
</html>
```

The head element contains meta information, standing for data the will not be directly presented, but may by used in the presentation of the body content ahead. For instance it may define a title for the page through the <title> tag, browsers may use that to give a name to the browser's window or tab. It may also establish the charset to be used when presenting the body (e.g. <meta charset="UTF-8">). The head element is optional.

The body establishes what is going to be presented and how.

Attributes and tags

Tags may also have attributes or properties, they are established when the tag is opened by a pair **attributename=value**. Many HTML tags support attributes, for instance in the body element you may define the background colour to be used on its content presentation, e.g.:

<body bgcolor=green>

The **Text** tag supports attributes regarding how within text is displayed, namely attributes **color**, **size** and **face**.

Some other basic HTML tags are:

<hr /> - draws an horizontal line.

**
** - introduces a line break (newline).

<h1>Text</h1> - Presents the content using header 1 format.

<h2>Text</h2> - Presents the content using header 2 format.

…

Tags <img> and <a>

HTML has hypertext on its name, this means a document may have links to other contents.

One case is the **<img>** tag, it's used to place an image on the document's presentation, but the image's content is in a separate source file:

****

The **src** attribute specifies the filename (resource) from where the image is to be fetched, to be displayed on that exact position.

The **<a>** tag establishes a live, user clickable, link through the **href** attribute:

Text

The text or whatever is within the tag is shown to the user as a clickable link, if clicked the current page being displayed is replaced by a new page referred by URI, the URI will usually represent a different HTML page, possibly in a different location.

Live links make the essence of the navigation throughout the WEB.

The <input> tag

The <input> tag, together with other special tags like <select>, <button> and <textarea> have been created to be used with forms (the <form> tag). They provide a wider range of user interactions namely regarding data submission.

In our study we are not going to use forms, instead we are going to use these tags directly through JavaScript and DOM (Document Object Model), but not right now.

The <input> tag has a key attribute the **type**, among others, the type attribute can be:

text – a single line text input field, it may be read-only.

password – same as before, but typed characters are not visible.

button – a user clickable button.

radio – a radio button

checkbox – a checkbox.

Depending on the type, they may support other attributes, like for instance **size** and **value** for the text type. For instance value also exists for the button type but in this case it represents the button face´s text.

JavaScript

JavaScript is an interpreted language, understood by web browsers, thus web browsers work as interpreters for JavaScript. It has some similarities with both C and Java.

Execution of JavaScript is mostly triggered by HTML elements, so it's widely based on functions definition that are latter called by HTML elements.

A function may receive arguments and also return a result, but it's an untyped language, neither variables or arguments have a defined type, for instance they may hold numbers, strings or booleans.

Next is an example a function definition in JavaScript:

```
function add(a,b) {
  c=a+b; return c;
}
```

Next an example on how to use it:

```
res=add(5,6);
```

Then variable named **res** would have value 11.

Including JavaScript functions in HTML

JavaScript code may be inserted into an HTML page by placing in within the <script> tag, if its on the body, then the code is executed when the document is loaded. Remember a function's definition itself is not executed, only when called.

To establish JavaScript functions to be made available for HTML, they are better placed in the <head> tag.

The JavaScript code defining functions may be placed inside a <script> tag, but an alternative is using the **src** attribute to load the JavaScript code from a different file into the current document:

<script src="URI"></script>

For instance:

```
<head>
<script src="myFunctions1.js"></script>
<script src="myFunctions2.js"></script>
</head>
```

Linking JavaScript to HTML

The call to JavaScript functions can be automatically triggered for several events.

To start with, when the document is first loaded, the **onload** attribute of the <body> tag can be used to establish a function to execute only once when the document is loaded, for instance:

```
<body onload="initData()">
```

For most HTML tags, possible events can be defined and the function to automatically call when they occur.

The simplest scenario is the event **onclick**, the attribute with the same name established the JavaScript function to be called when the user clicks the mouse on it. Example:

```
<button onclick="isDone()">Continue</button>
```

When the user clicks the **Continue** button the function **isDone()** is executed.

This is just one example of a possible event to be used to trigger a function call. There's a huge list.

Browser Object Model (BOM)

Modern WEB browsers provide several useful objects for JavaScript. Each object defines functions to interact with it, called **methods**, objects contain variables (data), called attributes or **properties**.

JavaScript can interact with objects by calling its methods or by directly retrieving or changing its properties.

The Browser Object Model (BOM) makes available to JavaScript the **window** object, this object represents the browser's window. Through it's possible for instance to close a window, open a new window, or query the current window size.

Some interesting methods implemented by the window object are:

window.setTimeout(function, milliseconds);

window.setInterval(function, milliseconds);

These functions allow execution scheduling of functions, the provided **function** is executed by the browser in background, exactly in the provided number of **milliseconds**. The setTimetout() method executes the function only once, the setInterval() method executes it repeatedly. By using these methods the periodic execution of functions in background can be scheduled. When a page is loaded, the initial scheduling may be done through the onload event of the body.

Document Object Model (DOM)

DOM provides the **document** object, this object belongs to the window object, so it can also be referred to as **window.document**.

The **document** object represents the loaded HTML web page, it has a tree structure to represent all tags/elements within the page. The tree's root is the <html> tag, then there are two branches, the <head> tag and the <body> tag, and so on. Within each tag, a new tag is a new branch.

By using the document object, JavaScript function are able to gain access to any tag/element within the web page and then interact with them.

To be able unambiguously get a single element (tag) of the page, the tag must have the **id** attribute defined. So every element required to be available to JavaScript through DOM should have the **id** attribute.

The **id** value is a string that must be unique within the document, this is because it's intended to uniquely identify that specific tag. For instance:

** **

Once this tag has the id attribute defined, then it can be retrieved by calling: **document.getElementById("result");**

DOM – interaction with elements

Once an element/tag has the **id** attribute defined, it may be retrieved by calling:

```
var elem = document.getElementById(idValue);
```

This method returns an object that represents the element (tag), if the id is not found it will return **null**. Now, by having access to the object we can interact with it.

Most tags have a content (what's between it's open and close), that is available through the **innerHTML** property. We can for instance change it:

```
elem.innerHTML = "<p>Teste</p>";
```

Or for instance copy its value: **a = elem.innerHTML;**

If an element has attributes, they can also be directly accessed.

For instance, the <body> tag is unique, so it may be directly access without id by: **document.body**

To change the background colour we can simply assign a new value to the bgColor attribute:

```
document.body.bgColor="green";
```

Mind in JavaScript attribute names are case sensitive.

JavaScript with BOM and DOM example 1

```
<!DOCTYPE html>
<html><head>
<script>
var colour="green";
function switchColour() {
          if(colour=="green") colour="blue";
          else if(colour=="blue") colour="red";
          else if(colour=="red") colour="green";
          document.body.bgColor=colour;
          document.getElementById("cname").innerHTML="The background colour is now " + colour;
          document.title=colour;
}
</script></head>
<body onLoad="window.setInterval(switchColour,2000)">
<hr/>
<h1>JavaScript, BOM and DOM demo</h1><hr>
<h1 id="cname">Starting ...</h1>
<hr/>
</body></html>
```

Once the document is loaded the switchColour() function is scheduled to be executed every two seconds (onLoad).

Notice the **colour** variable is declared outside the function (it's a **global** variable), if declared within the function it would be **local**, and thus freshly created on every execution of the function.

You may save this content to a file (e.g. rgb.html) and test it locally with your browser.

JavaScript with DOM example 2

```
<!DOCTYPE html>
<html><head>
<script>
function calculate() {
 m=parseInt(document.getElementById("m").value);
 a=parseInt(document.getElementById("a").value);
 b=parseInt(document.getElementById("b").value);
 if(a>b) {                                              // auto swap limits
          document.getElementById("a").value=b;
          document.getElementById("b").value=a;
          c=a; a=b; b=c;
 }
 txt="<p><b>Multiples of " + m + " in [ " + a + " , " + b + " ] are:</b> ";
 for(n=a;n<=b;n++) {
          if(n%m==0) txt=txt+ " " + n;               // integer division remainder equals zero
 }
 txt=txt+"</p>";
 document.getElementById("result").innerHTML= txt + document.getElementById("result").innerHTML;
}
</script></head>
<body bgColor="gray">
<hr/>
<h1>Multiples search within a closed interval</h1><hr>
<h2>Find multiples of <input id=m type=text value=0 size=1>
in [<input id=a type=text value=0 size=1> , <input id=b type=text value=0 size=1>]</h2>
<input type=button value=FIND onClick="calculate()">
<hr/>
<div id=result></div>
<hr/>
</body></html>
```

Save this content to a file (e.g. multiples.html) and test it.