SWitCH – Computing systems and networks (SCOMRED)

2020/2021

Laboratory 05

HTML. Active contents with JavaScript programming.

Team project start - Active and Dynamic Web site.

1. HTML pages

HTML is a **presentation notation** in text format, it contains data but it's totally focused on how data is presented to end users.

Web browsers are client applications that are able to fetch through the network, contents from web servers, among others, contents in HTML format, and are able to present those contents to end users.

Strictly speaking, an HTML content is text, so it's organized in lines. And yet, when interpreted as HTML, line breaks don't have any meaning, the whole content could as well be in a single line. Yet, line breaks are very useful for humans to read HTML contents.

- HTML is based in tags, in HTML, tags are names with special meanings.
- Tags are specified between angle brackets.
- Tags should be closed by repeating the tag's name with a slash prefix, e.g. <body></body>.
- What's between a **tag's open** and that **tag's close** it the **tag's content**.
- Tags with no content may be self-closed by adding the slash suffix on it's open, e.g. <hr/>.
- In HTML some tags (e.g.
, <hr>, <img>) are not required to be closed.
- Tags may be nested, one tag's content may have several other tags within it.
- Because an HTML content has a single mandatory tag embracing the whole content (<html></html>), nested tags also become a tree, and the <html> tag is the tree's root (root tag).
- The <html> root tag embraces the whole document's content, it may contain two tags (branches), <head> and <body>.
- When a browser loads an HTML content, tags become elements of the page.
- The <html> tag should be preceded by a the DOCTYPE preamble: **<!DOCTYPE html>**, meaning the following content is HTML.

So, the basic structure for an HTML document is:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

a) Getting help and learning HTML

Recommended sites for information about HTML and other incoming subjects:

1. **https://www.w3schools.com**
2. https://html.com
3. https://www.htmldog.com

PRACTICE:

Take a look around information available at **https://www.w3schools.com/html/**

b) Editing HTML contents.

We will see later an HTML contents may be dynamically generated by a running application. Focusing for now on static HTML contents, they are usually stored in files, of course, **text files**.

It's convenient for text files with an HTML content to have names ended by **.html**. This is because some operating systems like MS-windows use that to judge the type of content the file has.

Typically, web browsers get the HTML content to be displayed from web servers. Nevertheless, on the web server side, static HTML contents are stored in files as well.

For now, you are going to create and edit HTML files (text files with an HTML content) locally on your personal laptop and then test them, locally as well, with your personal web browser. Yet, if those contents were supposed to be made available by a web server, it would be a simple matter of copying those files to the server.

Any text editor can be used to create and edit text files with an HTML content. Yet, a syntax aware text editor turns out to be more productive. A syntax aware editor is one that is capable of checking syntactically, and even semantically, the text while it's being written.

If you have an IDE (Integrated Development Environment) installed on your laptop, it has its own syntax aware text editor. Thus if it supports (understands) HTML, you will be ok with it.

You can also install a generic, syntax aware, text editor, like for instance Notepad++. Such a text editor is capable of checking the syntax for several programming languages and text data formats.

c) Create and test local web pages.

PRACTICE:

In your personal laptop, create a new folder named **SCOMRED-HTML** to hold the web pages. You may create it wherever you want, for instance in your personal Desktop area.

Use your HTML text editor to create a new file in folder SCOMRED-HTML, named **index.html** with the following content:

```
<!DOCTYPE html>
<html>
<head><title>P1</title></head>
<body bgColor=gray>
<img src=https://www.dei.isep.ipp.pt/images/logo.png />
<h1>SCOMRED Test Page 1</h1>
<ul>
<li><a href=https://portal.isep.ipp.pt/>ISEP portal</a></li>
<br>
<li><a href=https://moodle.isep.ipp.pt/ target=moodle>ISEP Moodle service</a></li>
<br>
<li><a href=test2.html>SCOMRED Test Page 2</a></li>
<br>
</ul><hr>
</body>
</html>
```

Save the file content and test the page by opening the index.html file with your personal web browser.

Match what you see in your web browser with the HTML content.

The **<ul>** tag establishes an unordered list of items, each identified with **<li>** tags (list item). We could also use **<ol>** (ordered list) instead of <ul>, then list elements will be numbered, HTML lists can also be nested.

By default, live links defined by the **<a>** tag are loaded into the current browser window, and thus replace the current content (first list item). If the **target** attribute is used, the link is opened in a window with that given name, if it's not the current window, then it's opened in a new window (second list item).

The link in the last list item is broken because the referenced page doesn't exist for now, unlike previous links that have absolute locations, this last one points to a relative location.

It's a relative location because it only has a filename, it says nothing about the network location (server name) or about the path (folder) within the server. In this case, the browser assumes the location is the same as the currently loaded file, so this link refers to file named **test2.html** in folder **SCOMRED-HTML** of your laptop. You may see that by checking the link the browser loads when you click it.

Browsers provide several navigation buttons, by pressing the back button, usually represented by an arrow pointing left, you are able to navigate to the previous location where you were. Another important bowser button is reload, when you change a page (file) to view the new content you must use this button.

PRACTICE:

Now let's create the page for the broken link, a file named **test2.html** in folder **SCOMRED-HTML** of your laptop. Create it with the following content:

```
<!DOCTYPE html>
<html>
<head><title>P2</title></head>
<body bgColor=lightblue>
<img src=https://www.isep.ipp.pt/img/logo3.png />
<h1>SCOMRED Test Page 2</h1>
<p align=center><font color=red face=arial>
This is the second HTML test page. This paragraph is aligned to the page center. It's
using the ARIAL font in red colour.
</font></p>
<ol>
<li><a href=https://www.dei.isep.ipp.pt/ target=dei>www.dei.isep.ipp.pt</a></li><br>
<li><a href=https://rede.dei.isep.ipp.pt/ target=rede>rede.dei.isep.ipp.pt</a></li><br>
<li><a href=index.html>SCOMRED Test Page 1</a></li>
</ol>
<center><hr width=90%><table border=1 bgColor=yellow width=50%><br><br>
<tr><td colspan=3 align=center><big>Sample table</big></td></tr>
<tr><td align=center><b>Item</b></td><td align=center>Date</td>
<td align=center>Size</td></tr>
<tr><td align=center><b>A</b></td><td align=center bgColor=gray><i>2018-10-30</i></td>
<td align=center>10 cm</td></tr>
<tr><td align=center><b>B</b></td><td align=center>2018-10-31</td>
<td align=center>20 inches</td></tr>
</table><hr size=10></center>
</body>
</html>
```

The last link of the first page should now work, click it.

Again, match what you see with the HTML content.

Now we have an ordered list (<ol>) instead of an unordered list (<ul>).

In this page there's a table at the bottom, defined by the **<table>** tag. Within it, the **<tr>** defines a table row, within each table row the **<td>** tag defines each column's value. These tags support several attributes.

d) Deploy locally created HTML files into a real web server.

We are now going to place the two created files (index.html and test2.html) in a real web server, the server you already have in the DEI cloud.

Your server is named **vsXX.dei.isep.ipp.pt**, where XX is a unique number for your server, it has an Apache web server running, thus you can access it in URL **http://vsXX.dei.isep.ipp.pt** , when you open this link the server is sending the content of the local file **/var/www/html/index.html** we have created in LAB02. This happens because folder **/var/www/html** is defined in Apache as the documents root and because if no filename is specified Apache tries to load a file named **index.html**.

One safe way to upload files into a server is by using SSH, beyond a safe terminal session access, SSH also allows a safe file transfer, this is called SFTP (SSH File Transfer Protocol or Secure File Transfer Protocol). Most likely you already have an SFTP client on your laptop´s command line, it may be the **sftp** or **psftp** command, the last one is a version provided together with PuTTY.

You can also use a SFTP client with GUI (Graphical User Interface), like for instance Bitvise (https://www.bitvise.com/) for Windows. In the following guidelines we are going to use the command line version of the SFTP client.

IMPORTANTE: Next it's assumed you have your own server running, with the DNS name **vsXX.dei.isep.ipp.pt**, and that your laptop is connected to a DEI VPN service for you to be able to reach it.

PRACTICE:

Open a command prompt on your laptop, then change your working directory to be your local **SCOMRED-HTML** folder.

Now run the local SFTP client to connect to your server's SSH service:

```
sftp root@vsXX.dei.isep.ipp.pt
```

You are logging in as user root, type the correct password and you should be allowed to login.

Now change the remote current working directory to **/var/www/html**:

```
cd /var/www/html
```

Upload both files:

```
put index.html
put test2.html
```

You may now exit the SFTP client by typing the **exit** command.

Now let's test, on your local browser open URL **http://vsXX.dei.isep.ipp.pt**, if everything went ok you should see the first HTML page create today in this class. Check that both pages are working as before.

File **test2.html** is now available on URL **http://vsXX.dei.isep.ipp.pt/test2.html**.

File **index.html** is now available on URL **http://vsXX.dei.isep.ipp.pt/index.html**, because if no filename is specified Apache, and most other web servers, load the **index.html** file, this file is also accessible on URL **http://vsXX.dei.isep.ipp.pt**.

2. JavaScript in HTML pages

JavaScript is a programming language all modern web browsers understand, and thus are able to execute, so from this point of view web browsers are JavaScript interpreters.

The approach for embedding JavaScript execution into an HTML web page is by defining functions to be called by HTML elements (tags), more exactly when some **events occur on HTML elements**. One most notorious event is when the page (body) is loaded in the first place.

The <body> tag may contain an **onLoad** attribute defining the JavaScript function to be executed immediately after the content being loaded and displayed.

Typically JavaScript functions are not part of what is to be presented to the user when the page is loaded, so they should be defined within the <head> tag and not within the <body> tag. Elements loaded from the <body> tag will later call the defined JavaScript functions when some events occur.

PRACTICE:

a) Now, let's test the first JavaScript example from lectures. On your **SCOMRED-HTML** folder create a new file named **rgb.html** with the following content:

```
<!DOCTYPE html>
<html><head>
<script>
var colour="green";
function switchColour() {
if(colour=="green") colour="lightblue";
else if(colour=="lightblue") 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>
```

This is a very simple example, and yet it uses both BOM and DOM.

In this example, the BOM object **window** is used to settle the periodic execution of the **switchColour()** function every two seconds. This is done when the <body> tag is loaded (onLoad).

The DOM object **document** is used to access the body element and settle the background colour, it's also used to get one specific element of the page (by its id) and change its content.

Test this page in your personal web browser.

PRACTICE:

b) Now let's test the second example from lectures. On your **SCOMRED-HTML** folder create a new file named **multiples.html** with the following content:

```
<!DOCTYPE html>
<html><head>
<script>
function calculate() {
var m=parseInt(document.getElementById("m").value); if(isNaN(m)) m=0;
var a=parseInt(document.getElementById("a").value); if(isNaN(a)) a=0;
var b=parseInt(document.getElementById("b").value); if(isNaN(b)) b=0;
if(m<0)m=-m;
if(a<0)a=-a;
if(b<0)b=-b;
if(a>b) { // auto swap limits
var c=a; a=b; b=c;
}
document.getElementById("m").value=m;
document.getElementById("a").value=a;
document.getElementById("b").value=b;
if(m<1 || b<1) return;
var txt="<p><b>Multiples of " + m + " in [ " + a + " , " + b + " ] are:</b> ";
var n; 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>
```

Now, the onLoad attribute of the body element is not used because the page isn't supposed to run JavaScript when loaded. The JavaScript function **calculate()** is executed when the user clicks the FIND button (**onClick** attribute of the <input type=button> tag).

Some improvements were added regarding user provided input data validation and automatic correction. Also, in the case of automatic corrections, the corrected values are placed in the input fields, replacing the originals.

- If not numbers (**isNaN()** function) they are replaced by zero. NaN stands for **Not a Number**.
- Negative numbers are replaced by their opposites.
- If interval limits are reversed, they are swapped.
- If values don't make sense the calculation is not performed (e.g. multiples of zero).

Each result is appended as a new paragraph to the tag on the bottom of the page, so the user can see the results of previous calculations.

Test this page several times with valid and invalid input data.

PRACTICE:

c) Remember the little game we have implemented in a BASH script? Something like: **Guess a number between 0 and 100**. Now, try creating your own HTML page with JavaScript functions to implement that same game.

To get a random number you may use the **Math.random()** function, it returns a random floating point number in the [0 , 1 [interval.

Because **Math.random()** never returns number 1, multiplying it by 100 is not enough (the 100 number would never be generated), you must multiply it by 101:

Math.floor((Math.random() * 101));

The **Math.floor()** function returns the highest integer bellow its argument. Because in this case the argument will never be 101, the highest resulting number is going to be 100. Ultimately this expression results in a random integer number in the [0 , 100] interval.

To deliver this task:

1st Inspire yourself on previous examples.

2nd Search on earlier recommended sites.

3rd Ask the teacher.

3. Team project – Active and Dynamic Web Site/API

In this project, each team is enrolled in developing four web applications. All four applications have a frontend and a backend, some provide an HTTP API and others are clients of those APIs.

Teams are to use the following technologies and protocols to design and implement the requested applications:

Frontend: HTML, JavaScript, AJAX, HTTP.

Backend: HTTP, Apache on Linux, and CGI based BASH scripts.

The applications are to be developed/deployed in DEI Virtual Servers Cloud virtual servers belonging to the team members.

The teacher assessment is based in a final live demo and the sprint review.