MAS115 PRESENTATION LAB, WEEK 10

1. Mathematical webpages

Today we will put mathematical content on to webpages. The first approach is by using a script known as *Mathjax*.

1.1. **Mathjax.** Open a text editor (Notepad or Notepad++) and create a basic webpage with title and `h1` (that is, major heading) 'Newton's laws'. Include a doctype declaration by copying it from the source of the course webpage.

> *If you can't remember the basic set-up of an HTML page, view the source of my lab attempt from Week 8 on the course website.*

Use the CSS file in 'My lab attempts' from Week 8 on the course webpage by saving it to your folder as `week10lab.css` and linking to it from the HTML. Alternatively, use one of your own CSS files.

Create an `h2` (subheading) called 'Newton's Laws of Motion' and a paragraph reading 'The following is taken from Wikipedia'. Find the Wikipedia page for Newton's Laws of Motion and copy and paste the introductory section (the bit above the contents block) into a new paragraph in your HTML file. Turn the word 'Wikipedia' in your introductory sentence into a link to the Wikipedia page.

You'll see that most of the formatting for the text has gone. Tidy it up by doing the following.

(1) Delete all the footnote references that are in the text.
(2) Find the three laws and turn them into an *ordered list*. To do this, use the tag `<ol>` before the list and `</ol>` afterwards. Then, add `<li>` before each list item and `</li>` afterwards.
    (Think of this like LaTeX's enumerate environment, with `<ol>` similar to `\begin{enumerate}` and `<li>` similar to `item`.)
(3) Change the 'ol's to 'ul's to create an *unordered list*. Change them back if you prefer.
(4) Find the mathematical symbols in the text and put them in dollar-signs, like you would in LaTeX.
(5) Go to the course website and look for the Mathjax-code in the HTML section under 'Extras'. Copy and paste this into the head of your HTML document. Any change?
(6) Change the equation describing Newton's second law to $F = \frac{d}{dt}(mv)$ and put it in display-math mode using double-dollars.
(7) Rewrite the paragraph so that it talks of 'change in momentum' rather than acceleration.

You should have found that after pasting the Mathjax code into the head of your document, the LaTeX maths was formatted in the way it was intended.

Create a second subheading called 'Newton's Law of Gravitation', find the corresponding Wikipedia page and copy and paste a similar amount of material from there, crediting Wikipedia as before. Tidy things up as best you can, including the formula.

(8) Create a div surrounding the definition of the law itself, and set its class to 'definition'. (See last week's sheet on using divs.)
(9) Use CSS to change the width of the 'definition' div to 80%, the font-style to italic and center it (as you did for the body of last week's page).

Add a section at the bottom called 'Appendix' and follow it with the paragraph 'The table below shows the SI units for the quantities involved in the above discussion'.

By looking at the source code for the course webpage, build a table as below, filling in the rows corresponding to the symbols used in the formulas on the webpage. You should be looking for tags such as `<table>`, `<thead>` (table head), `<tbody>` (table body), `<tr>` (table row), `<th>` (heading cell) and `<td>` (standard cell).

Symbol	Quantity	SI unit
F	Force	Newton (N)
⋮	⋮	⋮

If you aren't happy with the format of the table, use CSS to change it by using the `table`, `th` and `td` selectors. (You could borrow ideas from the CSS file on the course webpage).

1.2. **Mini-project revisited.** Let's find out how easy it is to turn a LaTeX file into a webpage.

(10) Start a new HTML file called `miniproject.html`. Take the code from the body of your mini-project LaTeX file and paste it into the body of the HTML document.
(11) Replace all the LaTeX commands for sections, titles, itemized lists, etc. with their HTML equivalents to get your mini-project working as a webpage. (See the note below.)
(12) Add links to your Python scripts in appropriate places.

Note: To get blocks of computer code to display well in webpages, you should put it within `<pre>` (which stands for 'pre-formatted') and `<code>` elements. That is, you should write

```
<pre>
<code>
(Code goes here)
</code>
</pre>
```

For inline code, you can just use `<code>...</code>`. If you don't like the default style of font, you can change it using the font-family property in CSS (Courier New would be a good choice).

You may find that some characters don't display properly (for example, < signs will cause problems). To correct this, you should use convert your code to 'safe' HTML characters using an online converter, such as the one linked to from the Extras section of the course webpage. You probably want to choose options to convert line breaks and change spaces to ` `. (What does ` ` do?)

To make things much prettier, change the opening `pre` tag to

```
<pre class="prettyprint linenums">
```

and enable Google Prettify by copying and pasting the relevant line from the Extras section of the course webpage just above it.

1.3. **Embedding Python scripts into a webpage.** There are ways to embed simple Python scripts into a webpage. One option is to use `repl.it`.

(13) Go to `http://repl.it` and choose Python 3.
(14) Paste a Python script in the left-hand side. (It may not work for scripts which use packages like `numpy` and `matplotlib`.)
(15) Check that it runs by pressing the 'run' button.
(16) Click the 'share' button, and copy the share link.
(17) Add the following line to your HTML file, replacing the source URL with your link as appropriate.

```
<iframe src="your link" style="width:100%; height:500px"></iframe>
```

There other similar ways to get basic Python scripts working on a page. One similar site is Trinket (`https://trinket.io/python`). Perhaps use the discussion board to share ideas if you find a better one.

1.4. **LaTeX to HTML converters.** There are programmes that will take LaTeX code and turn it into an HTML file. Most of them are hard to install and configure, and most of them don't do a very good job.

The easiest one to get working is called *tex4ht*. This comes with MiKTeX, so should be ready to use. However, I tried to get it working on the managed desktop and failed.

To use tex4ht requires some knowledge of calling programmes from the *command line*. The command you need to use to run tex4ht is

```
htlatex *.tex "xhtml,mathml"
```

where *.tex is your tex file, and you've already navigated to the folder containing your tex file.

> *If these instructions mean nothing to you, then read about the Windows command prompt by following the link in the extras section of the MAS115 website.*

There are other LaTeX to HTML converters. Search online, and see what you find. There is even one called *Plastex* which is programmed in Python.

2. Going further with HTML and CSS

That ends the material on HTML and CSS for this course. There are loads of online tutorials on using HTML and CSS. If you find a really good one, why not share it with the rest of the MAS115 students on the course discussion board?

Homework

Continuing as described above, finish the process of turning your mini-project into a functioning webpage. Think about

- whether you want one page, or more than one page (a site);
- whether you want to have the Python embedded (running) in the page, linked to as a file, or simply displayed as code;
- whether there are any other features of a webpages that can enhance your project.

You do not need to print out anything. Instead, we will ask you to show us your webpage in the Week 11 lab.

This homework will count towards your homework mark for the module.