HTML Documentation

What is HTML

HTML, also known as HTML5, or HyperText Markdown Language, is a web development languaged used to create webpages. You can specify headers, paragraphs, links, images, and more on a webpage. Consider HTML the template or strucure of a webpage. HTML is not designed to make the webpage look pretty...that's CSS, which we can directly add to the HTML file, or use an external CSS file with a link to the HTML page.

Parts of the HTML Page

There are 2 main parts of the HTML page.

- Header
- Body

Header

The header is usually written in the top part of the HTML document. It contains information about the webpage, such as the refresh rate, the title or the tab name, and links to other resources such as CSS and JavaScript resources.

Linking an External CSS Page

To link an external CSS page, we use the link tag. The link tag will allow us to link an external CSS page. We can use the href argument to point to the CSS resource on the webpage. We will use the rel argument to specificy what resource is being accessed. An example of this is

```
<link rel="stylesheet" href="{{ url_for('static', filename='css/create.css') }}">
```

This format is used when working with Flask, the href argument will look different when serving a static webpage without Flask

Linking CSS within an HTML Page

To quickly add CSS within a HTML page, we can use the style tag. This will allow CSS to be added within the HTML page. An example would be:

```
<style>
    background-color: red;
</style>
```

Linking an External JavaScript Page

To link an external JavaScript page, we use the script tag. This will allow us to connect external JavaScript code with our HTML page. We will use the src argument to point to the JavaScript file. An example of this would be:

```
<script src="../../static/js/create.js"></script>
```

You can also use the script tag to make inline JavaScript code. Like this:

```
<script>
    //Some JavaScript Code
</script>
```

Body

The body contains all the elements that will be displayed on the webpage. The headers (h1-h7), paragraphs (p), tables (table, thead, tbody, tr, td), images (img), links (a), and dividers (div). Depending on what you want to do, you will need to specify the element you will use. Not only do elements have different functions, but elements also help screen readers or visually impaired users have an easier time navigating the page. **So make sure you use the correct elements!**

Title or Header Tags

h tags are very important in specifying headers, or titles of parts of the webpage. We can use numbers to specify the importance of the header. From 1 being the largest or most important title, til about 7, which specifies the least important header on the webpage. In addition, make sure to close the header tag with a /. This signifies that the tag is closed. Many HTML elements will be closed the same way, in exception to elements such as `<img>`. An example of a header is:

```
<h1>This is a very imporant title</h1>
<h3>This is a lesser important titel</h3>
```

Images

Image tags are used to display an image on the webpage. We use the argument src to tell the HTML page where the image is found on the webpage. We use the img tag to display images. Example:

```
<img src="../../static/img/triumf-trans-white.png">
```

Links

Links allow clickable buttons to link to different resources and different locations on the webpage. Anything that is published on the internet can be accessed with a link. We use the a tag and the href argument to speciy the location we want to go to when words, images, or buttons on clicked on. A few examples are:

```
    // Allows clickable image to different webpage
    <a href="https://www.triumf.ca/" target="__blank"><img
src="../../static/img/triumf-trans-white.png" id="logo"></a>
    // Allows clickable button to different route
    <a href="/view/{{dname}}"><button id="input-btn2">Back</button></a>
```

Tables

Tables are a very important when looking at data on the TRIUMF webpage. Tables allow us to display data in a sorted fashion. We use multiple tags to create a table.

- table - Specifies the general table
- thead - Specifies the table head, which will include the table header
- th - Specifies the table header, the names of each column
- tbody - Specifies the table body, will display the table data
- tr - Specifies each row of the table
- td - Specifies each box or data in each table row

A example of our code is:

```
<table id="table-data" class="table table-striped styled-table" border=1
frame=void rules=rows>
            <thead class="thead-dark">
                <th>Process Variable</th>
                <th>Reading</th>
            </thead>
            <tbody>
            {% if data %}
            {% for pv in data["readPvDict"] %}
            <tr class="data-row">
                <td class="elem">{{ pv }}</td>
                <td class="elem" id="reading">{{ data["readPvDict"][pv] }}</td>
            </tr>
            {% endfor %}
            {% endif %}
            </tbody>
        </table>
```

For more information on tables Visit Mozilla

Using Flask with HTML

With Flask, we are able to send data to display on the HTML page. To allow this data to be displayed or used in the html page, we use {{}}.
Let's create a simple example! Our Flask Python route will look like this:

```
@app.route("/")
def homepage():
    "This is the route for the homepage"
    return render_template("home.html", somedata=data), 200
```

Seen above, we can see that we are not only rendering the HTML page home.html, but we are sending or giving it data called somedata from the data variable in python.
Now lets see our simple HTML page. We must use {{}} to display the data. Our HTML page will look like this:

```
<h1>{{ somedata }} </h1>
```

Seen above, we specify that we want to use the Flask data with {{}} and we specify which data we want to use by inputting the variable name of somedata. With flask we can give or send multiple variables of data to the frontend. Like this:

```
return render_template("home.html", somedata=data, moredata=data2,
somemoredata=data3), 200
```

Depending on the type of data we went with Flask, we may need to use a loop to loop over dictionary items or lists. To do this, we use {{}} and specify the loop within this. An example of this is like this:

```
{% for pv in data["readPvDict"] %}
    <td class="elem">{{ pv }}</td>
{% endfor %}
```

We also use % to specify loops or conditional in the HTML page. The loop above is a simple loop that loops through the variable data and will name each element in the loop pv. It will then display the current pv in the loop. It will continue to add each pv into the table data. The endfor will then end the loop.