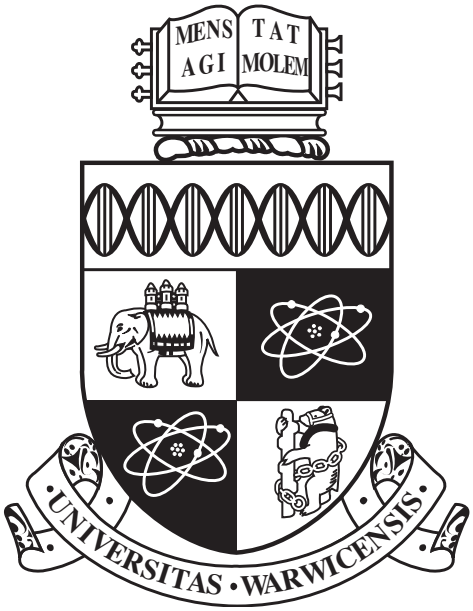
University of Warwick
Department of Computer Science

CS139

Web Development Technologies



Cem Yilmaz
February 11, 2022

Contents

1 HTML

1.1 Syntax

HTML stands for HyperText Markup Language and is semantic. This means that it describes the structure of the document and not the content. It is intended to modify the appearance of HTML elements and can be in fact frustrating to use for page layouts.

A lot of HTML is done with the <> brackets. For example,

```
<h1>Welcome to CS139</h1>
```

Listing 1: Heading

This would set the header tag to the text "Welcome to CS139". For this module, we will be using JSFiddle, that is available online.

1.1.1 Doctypes

Every HTML documents should have a doctype definition on top. In particular, HTML5 uses

```
<!DOCTYPE html>
```

Listing 2: DOCTYPE

It helps the browser to know what to expect.

1.1.2 Example HTML5 Document

```
<!DOCTYPE html>
<html>

  <head>
    <meta charset="UTF-8">
    <title>Title for Hello World </title> // Title that is seen at the top of the browser
  </head>
  <body>
    <h1>Hello world</h1> // Biggest header for the website
  </body>
</html>
```

Listing 3: Example Document

1.1.3 Head tag

This tag is used by the browser, web-crawlers and bots. IT includes meta-tags required by these applications and includes location of supporting documents e.g. JavaScript and CSS.

1.1.4 Text-encoding

Familiar with ASCII, but that is only 128 characters. In particular, UTF-8 has 107000 characters and is denoted with

```
<meta>
```

Listing 4: Text-encoding

1.1.5 Body tag

This is the tag where main information goes into that the user gets to read. Its syntax is

```
<body>
</body>
```

Listing 5: Body

1.1.6 Syntax

```
<a href="google.com">Google search</a>
```

Listing 6: Syntax

In the code above, a is the element name. The hyperlink in href is called the *Attribute*. The content is the *Google search* text. However, there are also empty tags e.g.

```
<meta charset="utf-8">
```

Listing 7: Empty Tag

1.1.7 Nesting Definitions

Definition 1.1. Child and parent

A syntax is a child if and only if there exists a tag that is at a lower level than the upper tag.

```
<body>
  <p>
  This is some text
  </p>
</body>
```

Listing 8: Parent Child

In particular, $< body >$ is the parent of $< p >$ and $< p >$ is the child of $< body >$.

Definition 1.2. Sibling

Sibling is when the tag is on the same level. For example,

```
<body>
  <p>
  This is some text
  </p>
  <p>
  Another text
  </p>
</body>
```

Listing 9: Sibling

In here, the p are siblings.

Similarly, the term descendants would be group of tags of in comparison to a tag that is a parent of all.

1.1.8 Lists

There are 3 types of lists:

- Ordered lists denoted with $< ol >$ and then listed items with $< li >$
- Unordered lists denoted with $< ul >$ and then listed items with $< li >$
- Description list would list terms and then list descriptions. In particular, the tags that are used are $< dl >$, $< dt >$ and $< dd >$ which are list, item and description respectively.You can also created nested list if you simply begin another list inside a list.

For example

```
<ul>
  <li> shopping </li>
  <ol>
    <li> eggs </li>
    <li> bread </li>
  </ol>
  <li> cooking </li>
</ul>
```

Listing 10: Lists

1.1.9 Hyperlinks

Hyperlinks are listing websites to a specific piece of text. For example

```
<a href="www.google.com"> This is google hyperlink </a>
```

Listing 11: Hyperlink

You can also hyperlink inside the website using ids. For example,

```
<body>
<h1> Links </h1>
<p id = "#top">
  This is some paragraph text
</p>
<a href="#top"> Go to top </a>
```

Listing 12: IDs

1.1.10 Images

You can also include images with a singular tag that use the src and alt attributes. For example,

```
<img src="http://warwick.ac.uk/logo.gif" alt="Warwick Logo" title="Warwick Logo" width=200
    height=60 />
```

Listing 13: Image embed

1.1.11 Character entities

```
  //Nonbreakable space
&lt; //<
&gt; //>
&copy; //Copyright symbol
&trade; //Trademark symbol
```

Listing 14: Character entities

1.1.12 Break Line

You can get a new line or break a line using the tag

```
<br>
```

Listing 15: Break

1.2 Semantic Mark-up

Some semantics include but are not limited to

```
<abbr>
<cite>
<time>
<span>
<div>
```

Listing 16: Semantics

That is, these do not change the looks but are important regardless. Usually, semantic mark-up refers to creating IDs on code to give meaning to piece of text and make it readable. In particular, the example in Hyperlinks with the ID is an example of a semantic mark-up.

1.3 Validation

You can make sure that your HTML is valid using a validation tool provided by W3C. The website is http://validator.w3.org

1.4 Types of Style Sheet

There are three different types of style sheet:

- Author created style sheets
- User style sheets
- Browser style sheets

1.5 Tables

A lot of data in HTML conveys data in rows and columns, i.e., a table.

```
<table>
  <tr>
    <td> The death of Marata </td>
    <td> Jacques-Louis David </td>
    <td> 1793 </td>
    <td> 162cm </td>
    <td> 128cm </td>
  </tr>
  <tr>
    <td> Burial at Ornans </td>
    <td> Gustave Courbet </td>
    <td> 1849 </td>
    <td> 314cm </td>
    <td> 663cm </td>
  </tr>
</table
```

Listing 17: Table

In here, $< /tr >$ stands for table row. $< td >$ stands for table data. Therefore cells in a row are declared by td, whereas rows are declared by tr. There is also $< th >$ which characterises table header. You can also add rowspan i.e.

```
<table>
  <tr>
    <th> Artist </th>
    <th> Title </th>
    <th> Year </th>
  </tr>
  <tr>
    <td rowspan="3"> Jacques-Louis David </td>
    <td> The death of Marat </td>
    <td> 1793 </td>
  </tr>
  <tr>
    <td> The Intervention of Sabine Woman </td>
    <td> 1789 </td>
  </tr>
  <tr>
    <td> Napoleon Crossing the Alps </td>
    <td> 1800 </td>
  </tr>
</table>
```

Listing 18: Rowspan

Column span works in a similar fashion. Furthemore

```
<thead>
// Top of the table. It is always the top rows of a table.
</thead>
<tbody>
// Middle of the table. It is always inbetween header and foot.
<tbody>
<tfoot>
// Sets the last row of the table. This will always be at the bottom of the table.
</tfoot>
```

Listing 19: Table Head and Footer

1.6 Forms

Sometimes we require to get data from the user, e.g. log-in page. It has the attributes:

- Action - the destination of the form data when submitted
- Method - the way in which the data is sent (GET or POST)
- Accept-charset - The charset accepted by the form

```
<form method="post" action="process">
  <fieldset>
    <legend> Details </legend>
    <p>
      <label> Title: </label>
      <input type="text" name="title> />
    </p
    <p>
      <label> Country: </label>
      <select name="where">
        <option> Choose a country </option>
```

```
        <option> Canada </option>
        <option> Finland </option>
        <option> United States </option>
      </select>
    </p>
    <input type ="submit" />
  </fieldset>
</form>
```

Listing 20: A basic form

1.6.1 Attributes

- Type - the kind of input to accept
- Name - the name submitted to the server
- The value that you want the field to have
- Other attributes related to type...

```
<input type="text" name="city name" value="Warwick" >
<input type="text" name="uname" placeholder="what is your name" >
```

Listing 21: Attributes

1.6.2 Radio Buttons

Radio buttons must have the same name in a group and allows you to select only a singular option

```
<input type="radio" name="where" value="1">Coventry<br>
<input type="radio" name="where" value="2" checked>Warwick<br>
<input type="radio" name="where" value="3">Nottingham<br>
```

Listing 22: Radio Buttons

1.6.3 Dropdown menus

Dropdowns need $< option >$ elements with the $< select >$.

```
<select name="choices">
  <option>First</option>
  <option selected>Second </option>
  <option> Third </option>
</select>
```

Listing 23: Dropdown menus

1.6.4 Accessibility

Sometimes we prefer the use of tables for data, not layout. In these cases, we use the caption element. This connect cells with a textual description in the header. You can also use "id" and "label for" to easen the accessability

2 Introduction to Python/Flask

2.1 Flask and Python

Flask is a server-side web micro-framework and it mainly uses Python as its scripting language. Its extensions provide further functionality. For example,

- Dynamic HTML markup uses Jinja2
- Database access via SQLAchamy
- OThers such as Flask-WTF (forms) and Flask-Login (authentication)

It was released in 2010 and its version 2.0.2 was released in October. Reddit, Linkedin, Netflix and Pinterest use it.

2.2 General Architecture

Python code is executed on a server and the output html is then sent to the client's browser.

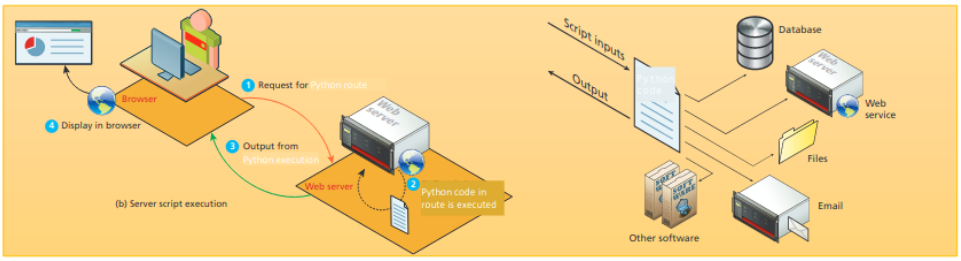


Figure 1: Architecture of web server

2.3 Flask and Python

=Python is used to execute a web server and is written within a .py file. HTMl code is written in .html file. Python uses indentation for scoping without braces or semicolons. Comments use ectomorph simple or triple apostrophes e.g.

```
133 # comment
134 ''' comment '''
```

Listing 24: Commenting

2.4 Variable Naming

Variables are duck=typed, i.e. dynamically and an integer. Variable names must begin with a letter or an underscore and cannot have spaces. Case sensitive value is not equal Value.

2.5 Python Data Types

Scalar types include

- Integer
- Float
- Complex
- Boolean
- String

And compound types

- Dictionary
- List and tuples
- Set
- Object

2.6 Python Containers

Python has 4 containers types: Lists, Sets, Dictionaries and Tuples.

- Lists are ordered and mutable, uses []
- Sets are unordered and cannot have duplicates, uses set ()
- Tuples are ordered and immutable, uses (,)
- Dictionaries are associative arraways - unordered values are references by their key, use {}

In particular, for examples, dictionaries can have "keys", e.g.

```
135 ages = {"Adam":27, "Dean":20, "Louise":30}
```

Listing 25: Keys

Then,

```
136 ages["Adam"] = 27
137 ages.update("Dean",20)
```

Listing 26: Keys 2

The key, which is their name, is linked to data which is their age.

2.7 Python Functions

You can define your own functions so that code is reusable. It is essentially creating methods. The code is

```
138 def <function name> (arg1, arg2...):
```

Listing 27: Functions

And these can return any object.

2.8 Dealing with Forms

Form values are submitted via a GET or POST method.

- GET - encodes the parameters in the url, some of you have already discovered this
- POST - encodes form values in the message body.

Flask stores these values in the args(GET) or the form(POST) of the request object and we can pull out the values and do whatever we wish with them.

2.9 Templates in Flask

Web pages don't need to be created in Python and we can create HTML pages and render them from Python instead. An HTML page in Flask is called a Template and is stored in the Templates folder. IT is rendered using

```
render_template(<pagename>)
```

Listing 28: Template Render

And these templates have Jinja2 scripting. That is, these are not just HTML and they can be scripted and accept data from Python. For example,

- Statements/scripts are written in $\{\%\%\}$
- Expressions evaluated to the output are written in $\{\{\}\}$
- Comments are written in $\{\#\#\}$

And this is useful because we can put shared elements into each page easily, i.e. headers, footers, menus etc. Furthermore, by keeping them in one place we can edit and have changes made in all pages.

2.10 Object Oriented Python

You can make python object oriented

```
from flash impot Flask
app = Flash(__name__)

class Person :
  name = "unset"

  def get_name(self):
  return self.name;

  def set_name(self, new_name):
  self.name = new_name;

@app.route('/oop')
def oop():
  p = Person()
  p.set_name("Jane")
  return p.get_name()
```

Listing 29: Object Oriented Python

Is an example of such, where we have a member variable called name and two functions get_name and set_name. To call methods on objects, we use the syntax

```
object.method(<args>)
```

Listing 30: Methods in objects

The . is used regularly in Python to represent hierarchies (classes, modules etc.) If you want to import constructors, then you wrap the name of the constructor around double underscore. i.e. __name__.
Lastly, there is no such thing as private or protected in python. Instead, in variables, we name them with _name which is protected and __name which is a private variable. Lastly, inheritance can be done using

```
class Admin(user):
```

Listing 31: Inheritance

This creates a subclass by using a class as a parameter to a class and you can also call parent class by

```
super().method()
```

Listing 32: Parent class

3 Databases

A web page can have the same structure: text,images,forms,tables,etc. even when the content is different. Web servers get the different data from a database. For databases, we use SQL. SQL stands for structured query language and is the language which you interact with Relational DataBase Management Systems (RDBMS). SQL is also a standard, however, database vendors extend their variants of SQL. It is possible to make a website with noSQL, and anther way to store is key-value pairs. No need for indexes and fast retrieval through other means e.g. hash-function. It is the same idea as dictionaries in Python.

Definition 3.1. Database

A data base is

- A structure collection of data
- Arranged into tables
- Data may be queried

3.1 Management

Databases often require permissions to access the data, such as users, passwords, access rights and etc. Managing the database is a complex process and is beyond the scope of this web-dev module. For simplicity we will use SQL called sqlite3.

3.2 Database table

A table is placed to store data of the same type. You decide on the columns in the table, and these usually represent the data which you will store. The rows of the table are the data that is entered. A table has a fixed number of columns, but may have an unlimited number of rows. You can think of it like a spreadsheet.

3.3 Relating data

For example, in an online forum, Users require to write leading to posts. Posts are a part of threads which belong to forums. Sites have many forums. Forum features must be view all posts by a user and get all threads in a forum. It should be able to find all users who posted in forum A, but not forum B on Tuesday.

3.4 SQL Datatypes

The SQL datatypes are

- integer - whole numbers
- real - decimal numbers
- text
- blob - binary data

3.5 SQL Operations

The basic operations we are going to consider are:

3.5.1 Creating a table

Listing 33: Creating table

3.5.2 Deleting a table

4 CSS

4.1 Cascade

Styles are applied in the following order:

- Browser default
- External style sheet
- Internal style sheet
- Inline styling

However, in terms of inheritance, only some properties are inherited because it is complicated.

4.2 Syntax

The commands modify the styling of your HTML, and for instance,

```
h1 {
  color: blue;
  font-size: 12px;
}
```

Listing 34: CSS example

4.3 Selectors

Notice how all $< h1 >$ tags will be modified the same way. A class selector can be used to modify just some HTML elements: we use fullstops to denote a class. An ID selector can e used to modify unique HTML element as well. These are done by denoting classes and then modifying the style of these classes

4.4 Pseudo-elements

A pseudo-selector targets a particular state or relationship e.g. the $< a >$ tag. An example of pseudo-element is the following:

```
<style>
p::selection {background-color:green;}
</style>
```

Listing 35: Pseudo selector

4.5 Box Model

All elements on a page are boxes. They all have width and height. Block level elements start and end with a new line. For example, $< h1 >$, $< p >$, $< table >$ are all block level elements. That is, they would become separate lines.
There are also inline elements, that is, boxes which are not in separate lines. Examples include $< a >$, $< img >$ etc.