MSc Project Websites

Dr Suhail A Islam
suhail.islam@imperial.ac.uk

Aim

The main aim of the presentation is to be a "**Getting Started**" guide for the data analysis project web sites for people with **no previous web site development experience**.

- Basic Web Terminology
  HyperText (HT) and Markup Language (ML), HTML
- HTML – Document Structure
- Web Deployment on Servers
- Methods for deployment – HTML, Frameworks
- Examples

Documentation

- Presentation documentation
  http://teaching.bc.ic.ac.uk/msc/websites

- Example of former MSc Projects
  http://msc.bc.ic.ac.uk/examples2020
  Username: msc
  Password: examples

- Some simple examples
  http://msc.bc.ic.ac.uk/htmlexamples

- Excellent documentation at
  https://www.w3schools.com

Project Planning

- **Plan the main sections for your Website layout as if you were writing a Report.** For example
  - Home (Abstract, Lay term introduction, …)
  - Method
  - Results
  - Discussion
  - Conclusion and Future Work
  - References
- **Interactive Elements** to consider
  - Hyperlinks to References
  - Hover-over words - glossary
  - Interactive Charts
- **Use existing templates and examples**

Very rare for anyone to create Website documents from scratch. You are not being examined on web development technologies !

Terminology – Markup Language

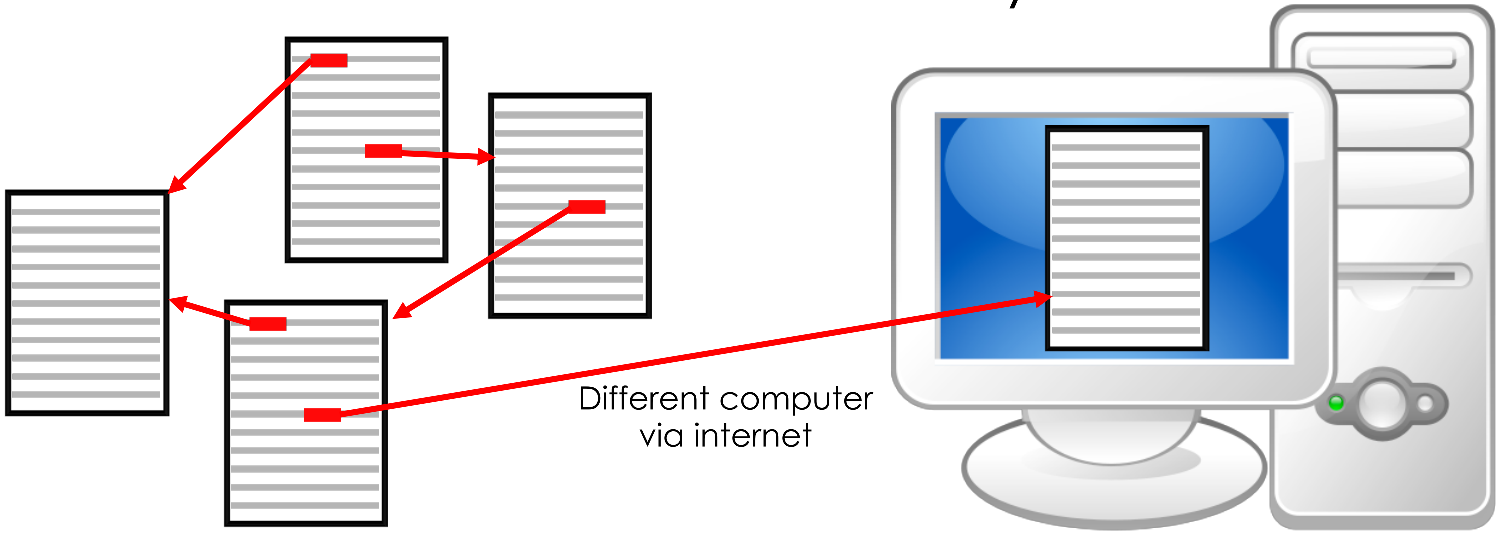
- The concept of a Markup Language dates back *circa* 1969 - Charles Goldfarb et al. (IBM)
- In computer text processing, a **Markup Language (ML)**, is a method for **annotating a document** (using so called **TAGS**) in a way that is visually distinguishable from the content and is typically not included in representations of the document for end users (e.g., font style, size, colour)
- In a document that is processed for display, the markup language does not appear. For example, you cannot see the "tags" for showing this TEXT in red or this **TEXT** in a different font.

Terminology – Markup Language

- Several **Markup Languages** exist – the two most popular one are **HTML** and **XML**
  - **HTML** – used for creating webpages
  - **XML** (Extensible Markup Language)– used for storing structured data
- **HTML** and **XML** document files are saved in a **plain text** format and can be viewed/edited in a standard text editor

Terminology – Hypertext

- Development / Ideas of 'Hypertext' date back to 1960's
- **Hypertext (HT)** is text displayed on a computer screen with *references* (called **hyperlinks**) to other text that a reader can immediately access



HTML - WWW

- 1980 - Tim Berners-Lee (**CERN**) proposed and prototyped **ENQUIRE**, a simply hypertext program for use by researchers to use and share documents.
- In 1989, Tim Berners-Lee proposed and later prototyped **a new hypertext project** in response to a request for a simple, immediate, information-sharing facility, to be used among physicists working at CERN and other academic institutions. He called the project "**WorldWideWeb**".
- 1991 – Firstly publicly available description of **HTML** (**Hypertext Markup Language**). 18 descriptive elements (**TAGS**).

HTML documents are text files

- The **HTML language** structures content, labelling different elements such as images and text in order to tell the browser how to display the content and in what order
- An **HTML Document** is a **text file**, written using the HTML protocol. Files usually have the extension '**.html**' or '**.htm**'
- A Web Browser reads an HTML document and displays its contents according to the rendering instructions
- **To create a Web site we need to create HTML documents**

HTML documents are text files

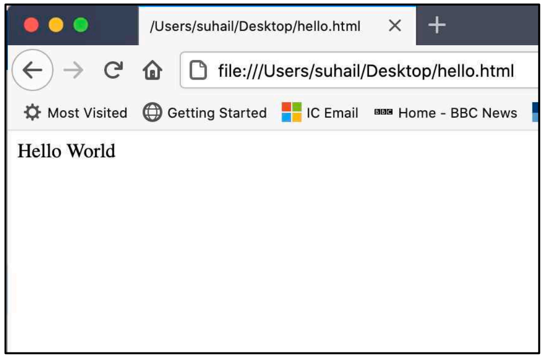
- An HTML text file, which can be created on your local computer and simply opened using a web browser

```
<HTML>

Hello Worlds

</HTML>
```

*Use sublime Text to create a file called **hello.htlm***



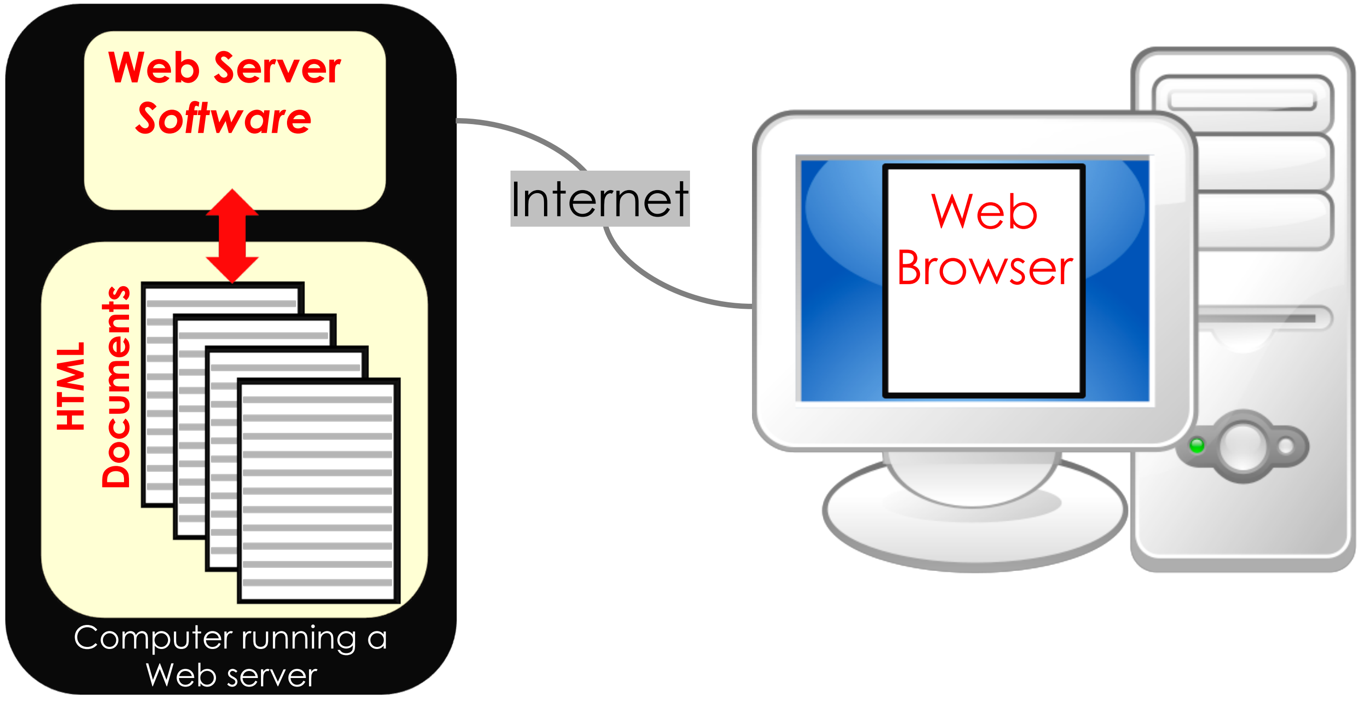
/Users/suhail/Desktop/hello.html

file:///Users/suhail/Desktop/hello.html

Hello World

- For a file to be accessible on a computer via the internet, the computers needs to be running software, simply referred to as a "**Web Server**" or "**HTTPD Daemon**" (*cf, database servers, print servers, Jupyter Hub Servers*)

WEB Server and HTML file location



- A **system administrator specifies which directories may may be accessed by the Web Server** software. Examples of two common locations are

  /var/www/html
  /User_Home_Directory/public_html

URL – Uniform Resource Locator

- Colloquially known as a ***Web Address***

  http://hostname/document.html

- If no document is specified in the URL e.g.
      http://hostname
  then the browser will look for a file **index.html**

URL	File on Web Server
http://teaching.bc.ic.ac.uk	**/var/www/html/index.html**
http://teaching.bc.ic.ac.uk/msc	**/var/www/html/msc/index.html**
http://msc.bc.ic.ac.uk/~suhail	**/project/home/suhail/public_html/index.html**

How do I create an HTML File

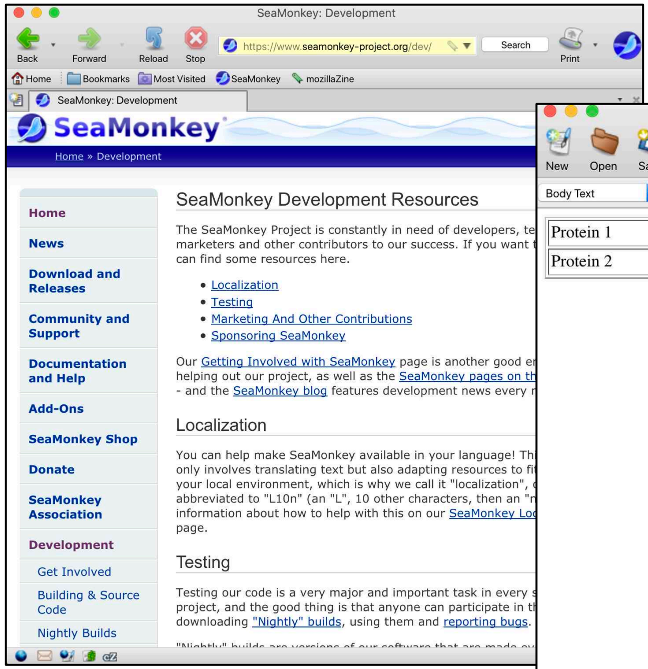
- Dedicated Web Design Software
  Adobe Dream Weaver

- Web Browsers can provide developer Tools e.g., **Seamonkey composer**
  https://www.seamonkey-project.org/

- Many application (e.g., MS word) can export a page as an HTML document

- **Use existing templates and a Text Editor like Sublime Text**

Seamonkey Example



Home
News
Download and Releases
Community and Support
Documentation and Help
Add-Ons
SeaMonkey Shop
Donate
SeaMonkey Association
Development
Get Involved
Building & Source Code
Nightly Builds

SeaMonkey Development Resources

The SeaMonkey Project is constantly in need of developers, te
marketers and other contributors to our success. If you want t
can find some resources here.

- Localization
- Testing
- Marketing And Other Contributions
- Sponsoring SeaMonkey

Our Getting Involved with SeaMonkey page is another good e
helping out our project, as well as the SeaMonkey pages on th
- and the SeaMonkey blog features development news every r

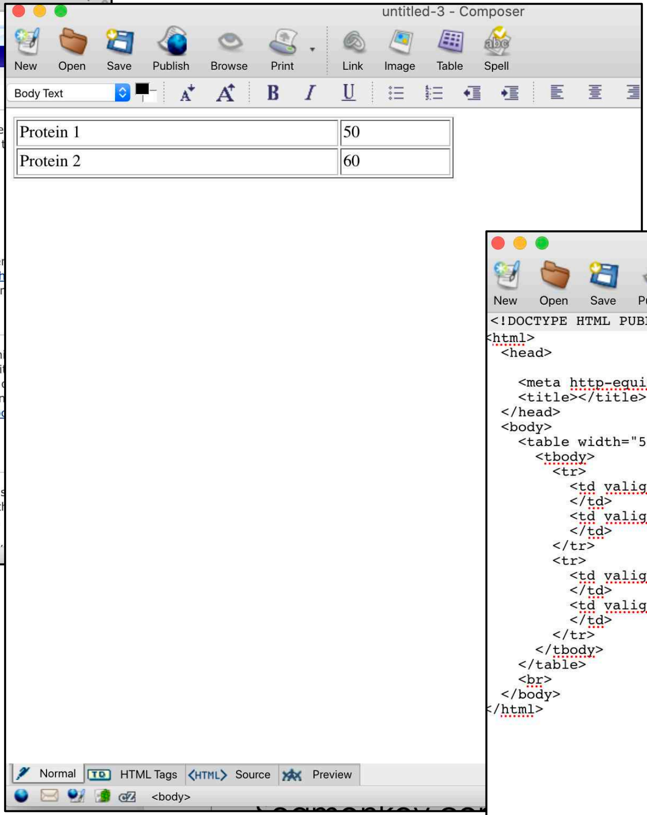
Localization

You can help make SeaMonkey available in your language! Thi
only involves translating text but also adapting resources to fi
your local environment, which is why we call it "localization", o
abbreviated to "L10n" (an "L", 10 other characters, then an "n
information about how to help with this on our SeaMonkey Lo
page.

Testing

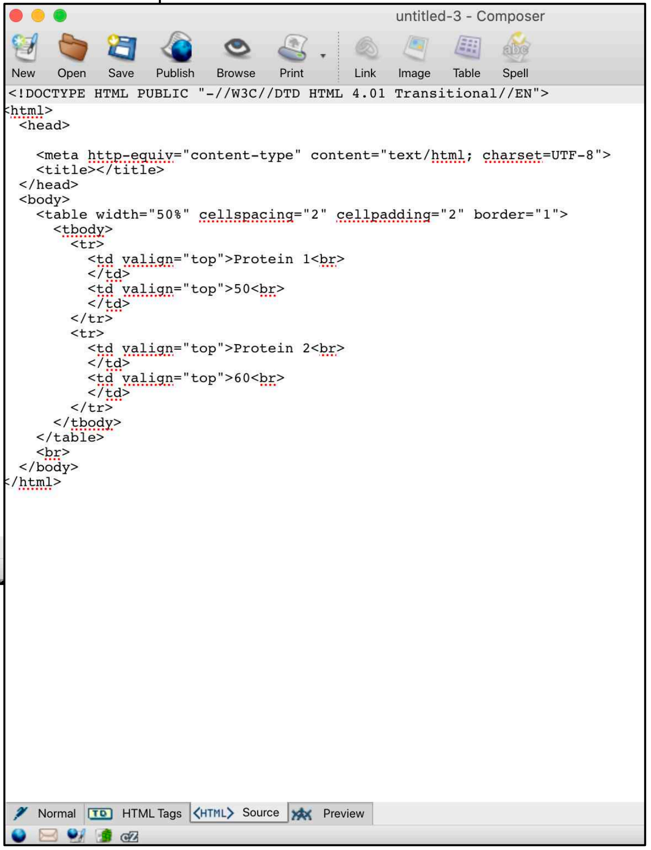
Testing our code is a very major and important task in every s
project, and the good thing is that anyone can participate in th
downloading "Nightly" builds, using them and reporting bugs.

Composer tool



Protein 1	50
Protein 2	60

Inserted Table Example



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>

    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
    <table width="50%" cellspacing="2" cellpadding="2" border="1">
      <tbody>
        <tr>
          <td valign="top">Protein 1<br>
          </td>
          <td valign="top">50<br>
          </td>
        </tr>
        <tr>
          <td valign="top">Protein 2<br>
          </td>
          <td valign="top">60<br>
          </td>
        </tr>
      </tbody>
    </table>
    <br>
  </body>
</html>
```

View HTML Source code

Creating Web contents

- Two types of approaches to creating and serving HTML files

1. **Static HTML Files** (e.g., create using a Text Editor)
2. **Web Application Frameworks (WAFs)** - **Django** *e.g., Instagram, YouTube, Google, NASA*), **Flask**, **Express**, **Rails** (*e.g., Airbnb, GitHub, Hulu*), **Shiny.** Django and Flask are python based and Shiny is R based.

   If using an Application framework, you will require a **PORT** (a unique number allocated to you by the system administrator) e.g., 202021

   http://msc.bc.ic.ac.uk:202021

Web Application Frameworks (WAF)

- Offer a level of flexibility and sophistication not feasible with *static html pages*. The backend can include writing to a database, processing information, retrieving information which is subsequently a response delivered to your web page.

- Handling
  - User requests
  - Dynamically generated webpages

What approach should you use ?

- Are WAFs overkill for a "*basic*" website ? I would recommend that you do try out/learn WAFs *sometime*
- If you are **not familiar** with WAFs and **do not** have a very specific **WAF** requirement - then I would suggest that you develop using the static HTML page approach.
- Most of our students in the past have used have developed websites using the static HTML page approach

http://msc.bc.ic.ac.uk/examples2020/
Username: **msc**
Password: **examples**

Creating Web contents with HTML

Three pilars for Web Development

- **HTML** (HyperText Markup Language)
  Defines the content to be displayed

And to add further sophistication

- **CSS** (Cascading Style Sheets)
  Tell the browser how to display the contents
- **JavaScript**
  Makes the content interactive

- Note: A web site can be created simply using HTML without CSS & JavaScript – but most people employ these to some extent

HTML - TAGS

- ML embeds codes, called "**TAGS**", in documents
- The codes ("TAGS")
    - Describe the structure documents
    - Include instructions for processing
- Tags are enclosed by angled brackets (**< >**) and **usually** come in pairs comprising an opening and closing tags. An opening tag begins a section of page content, and a closing tag ends it. Closing tags always proceed the element with a /. Think of Tags as "keywords" enclosed between angled brackets

HTML - TAGS

- Examples of Tags are: <**B**> , <**I**> , <**img**>
- Example: to make the words "Homo Sapiens" render as italics (the Tag to make text italic is <I>)

  … protein found in **<I>***Homo Sapiens***</I>** may not be …

- Example: to insert an image into a document use <img>

  … <**img** **src**="chart_1.jpg"> …

- Some Tags can have additional properties, call **attributes**. For example, when inserting an image we can define its size

  … <**img** **src**="chart_1.jpg" **size**="50%"> …

- Over 100 different TAGS currently defined

  https://www.w3schools.com/TAGS/default.ASP

HTML Elements

- An **HTML element** usually consists of a **start** tag and **end** tag, with the content inserted in between:

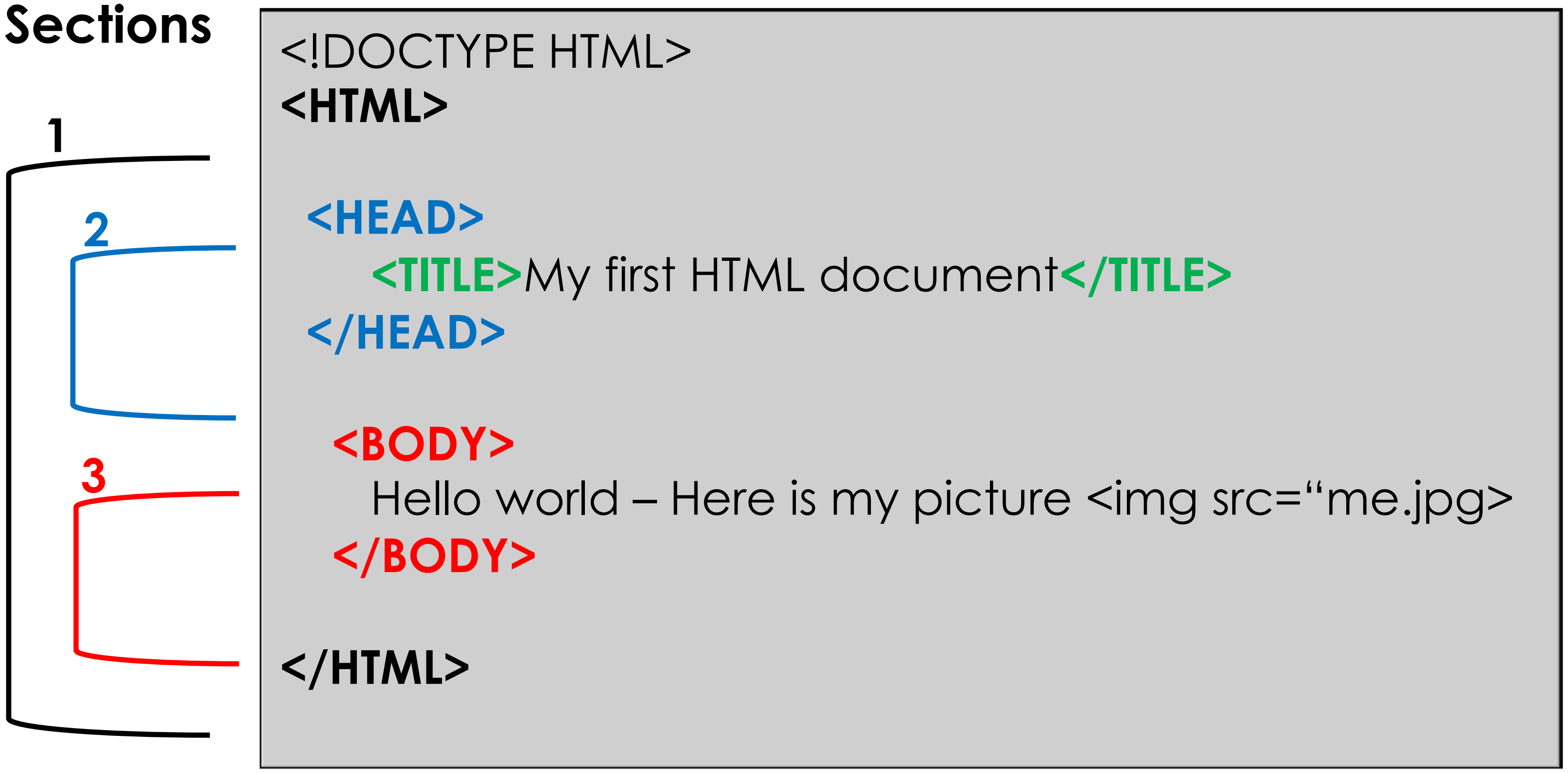
  <tagname>Content goes here...**</tagname>**

  The HTML element is everything from the start tag to the end tag

- Confusion arises between Tags and Elements. Although technically different definitions, in common usage the terms HTML element and HTML tag are interchangeable i.e. a tag is an element is a tag

HTML Document Structure

3 Main Sections



```
<!DOCTYPE HTML>
<HTML>

  <HEAD>
      <TITLE>My first HTML document</TITLE>
  </HEAD>

   <BODY>
      Hello world – Here is my picture <img src=“me.jpg>
   </BODY>

</HTML>
```

Opening Tag	Closing Tag	Description
<html>	</html>	Opens and closes an HTML document
<head>	</head>	The first of two main sections of an HTML document. The <head> section is used to provide information about the document for use primarily by search engines and browsers.
<title>	</title>	The title of document. This element is nested inside the <head> section. **In HTML5, this is the only required tag other than the DOCTYPE declaration.**
<body>	</body>	The second of two main sections of an HTML document. The <body> section contains all the content of the web page.

Examples

- Example files – HTML structure
  - **index1.html**
    Trivial example of an HTML document
  - **index2-MainHTMLSections.html**
    Basic HTML with the 3 main sections
- Example files – Common Tags
  - **index3-HoverOverHelp.html**
    Displaying help on a keyword
  - **index4-HyperlinkExtern.html**
    Hyperlinks to documents and sub-sections
  - **index8-Image.html**
    Insert an Image

Special Characters

- For rendering of non-English characters, special codes are needed and are usually preceded by an ampersand ("&").

  Examples

  Uppercase Alpha: **&Alpha;** or in Unicode **&#x0391;**

  Lowercase Alpha: **&alpha;** or in Unicode **&#x03B1;**

  https://www.w3schools.com/charsets/ref_utf_greek.asp

- Example files – Special Characters
  - **index9-GreekCharacters.html**

    Display Greek characters

Server Side Includes (SSI)

- Code shared by many documents (e.g., Main Menu) can be put into a document and then the document can be ‘included’ other files. Compare – python import.
- This is called **SSI.** Web servers may not allow this function unless explicitly configured (our MSc server does allow SSI)
- Example Files – top menu
  - **index7-Menu-included.html**
    Main menu included in document
  - **index7-Menu-IncludeFile.html**
    Menu read from file menus.shtml

Cascading Style Sheets (CSS)

- Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents.
- While HTML is used to structure a web document (defining things like headlines and paragraphs, and allowing you to embed images, video, and other media), CSS comes through and specifies your document's style—page layouts, colors, and fonts are all determined with CSS
- CSS code has its own syntax e.g., define tag <p>
  ```
  p { color:red;  font-weight:bold; }
  ```
- CSS code can be
  - Imported from a file into an HTML document
  - Embedded within the HTML document
  - Defined inline (inline elements occur within a block of text) within the HTML
- **https://www.w3schools.com/css/css_howto.asp**

Cascading Style Sheets (CSS)

- Example files – Common Tags
  - **index5-CSS.html**
    CSS defined in files css/styles.css and also define internally within document

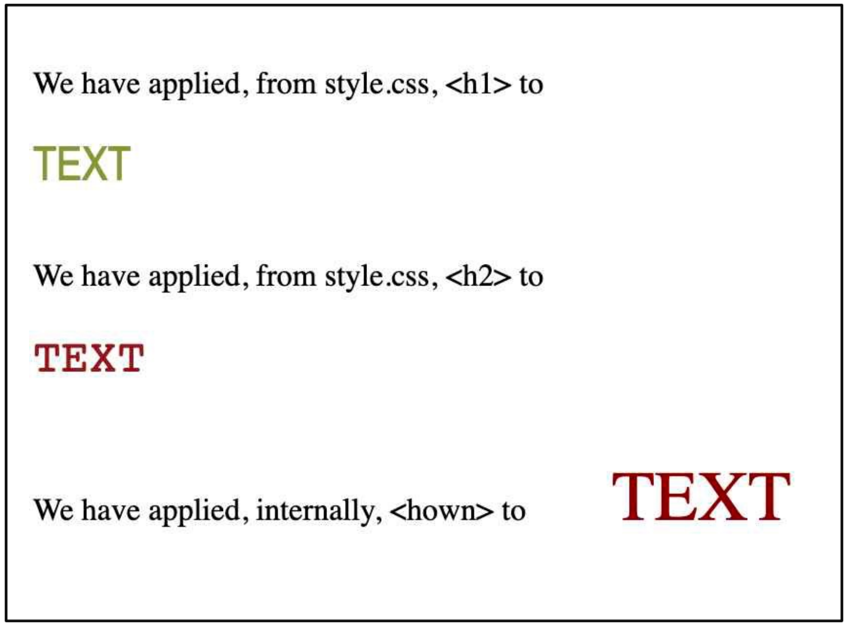
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
         "http://www.w3.org/TR/html4/strict.dtd">
<HTML>

 <HEAD>
    <TITLE>My first HTML document</TITLE>
    <!--- This is a Comment: Import Style Sheet file --->
    <LINK type="text/css" rel="stylesheet" href="css/style.css">
    <!--- This is a Comment: Define a CSS internally --->
    <STYLE>
     body {background-color: white; margin: 20px;}
     hown {color: maroon; margin-left: 40px; font-size: 36px;}
    </STYLE>
 </HEAD>

  <BODY>

    We have applied, from style.css,  &lt;h1&gt; to <h1>TEXT</h1><br>
    We have applied, from style.css,  &lt;h2&gt; to <h2>TEXT</h2><br>
    We have applied, internally,     &lt;hown&gt; to <hown>TEXT</hown><br>
  </BODY>

</HTML>
```



We have applied, from style.css, <h1> to

TEXT

We have applied, from style.css, <h2> to

TEXT

We have applied, internally, <hown> to TEXT

JavaScript

- Early specifications of HTML produced only static pages

- September 1995, a **Netscape** programmer named **Brandan Eich** developed a new scripting language in just 10 days. It was originally named **Mocha**, but quickly became known as **LiveScript** and, later, **JavaScript**.

- **JavaScript** is a scripting language that is considered one of the three core languages used to develop websites.

Whereas **HTML** and **CSS** give a website structure and style, JavaScript lets you add functionality and behaviours to your website, allowing your website's visitors to interact with content in many imaginative ways

JavaScript

- **JavaScript** (*do not confuse with JAVA – not related*)
  - **Programming Language** for the Web. Variable types

    Numbers
    Strings
    Objects
    Arrays
    Functions

  - Update and change both **HTML** and **CSS**
  - Can **calculate**, **manipulate** and **validate** data
- Examples – Embedded within the **<script>** tag in **HTML**

```
var x = myFunction(4, 3);   // Call Function & return value
function myFunction(a, b) {
  return a * b;
}

document.write("Some simple maths 10*3.141592 =",10*3.141592);
```

JavaScript

- **There are MANY pre-written JavaScript libraries** available (*cf* python and R libraries). These libraries can be **download** onto the **local file system** or **linked to remotely** within HTML code

- Your main use for project websites will be to **create interactive charts**. Two popular libraries are
    - https://d3js.org/
    - https://www.anychart.com

JavaScript

- Example files
  - **index6-JavaScript-BarGraph.html**

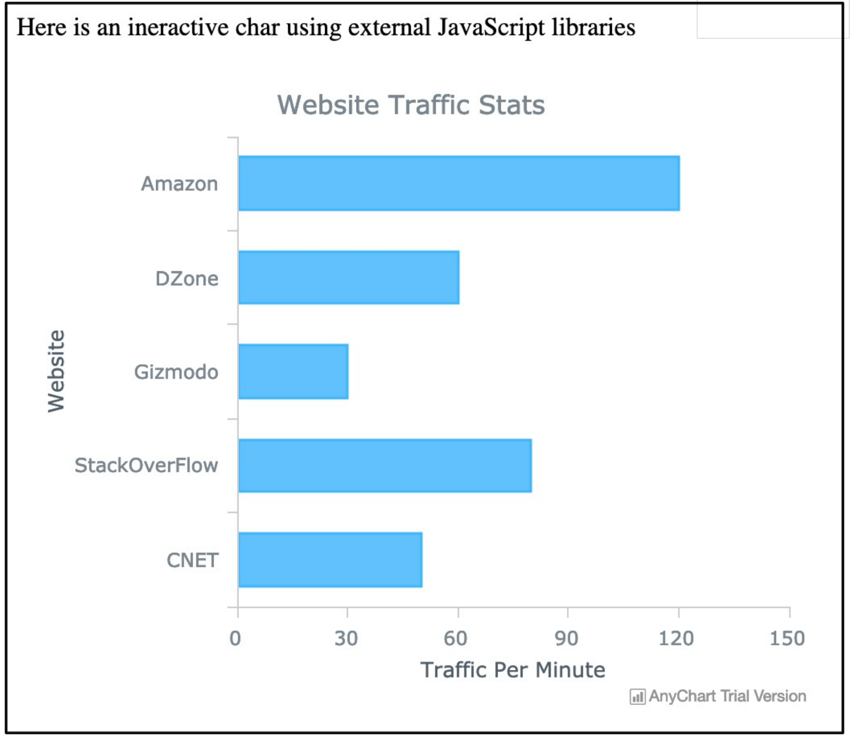
    Interactive BarGraph

```
<!DOCTYPE HTML>
<HTML>

 <HEAD>
     <TITLE>My first HTML document</TITLE>
     <!--- Read just one of MANY java script codes --->
     <script src="https://cdn.anychart.com/releases/v8/js/anychart-core.min.js"></script>
     <script src="https://cdn.anychart.com/releases/v8/js/anychart-cartesian.min.js"></script>
 </HEAD>
   <BODY>
…

     <div id="container" style="width: 500px; height: 400px">
     <script>
     anychart.onDocumentLoad(function() {
     // create the data
     var data = [
           {x: 'Amazon', y: 120},
           {x: 'DZone', y: 60},
           {x: 'Gizmodo', y: 30},
           {x: 'StackOverFlow', y: 80},
           {x: 'CNET', y: 50}
        ];
     var chart = anychart.bar(); // create a chart
     chart.title('Website Traffic Stats'); // create title for the chart
     chart.xAxis().title("Website"); // create name for X axis
     chart.yAxis().title("Traffic Per Minute"); // create name for Y axis
     var series = chart.bar(data); // create bar series and pass data
     chart.container("container"); // reference the container Id
     chart.draw(); // initiate drawing the bar chart
       });
…

</HTML>
```


Here is an ineractive char using external JavaScript libraries

JavaScript

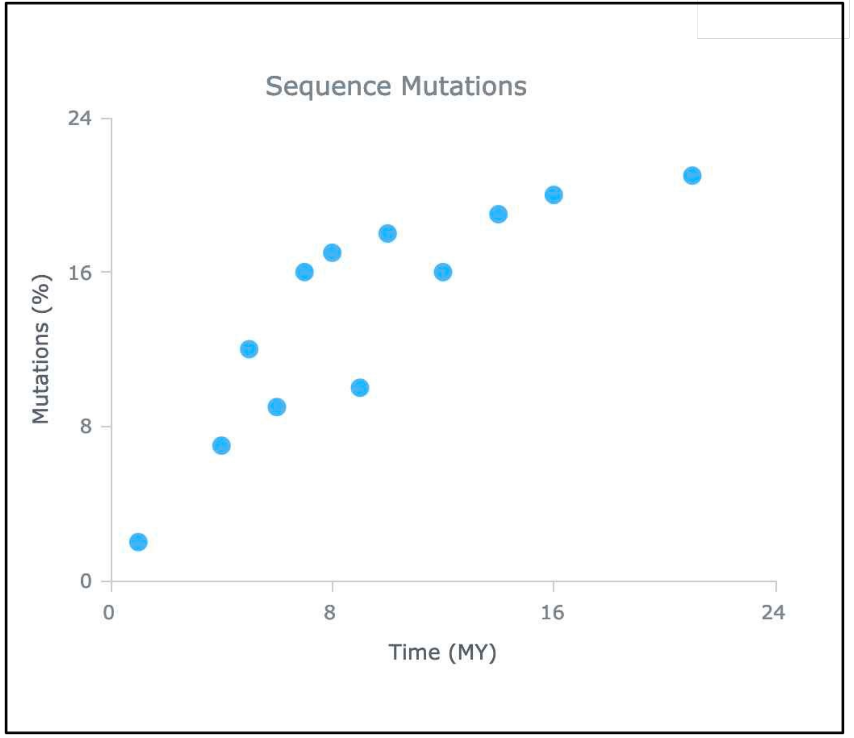
- Example files
  - **index6-JavaScript-Scatter.html**

    Interactive Chart – read data from data.csv

```
<!DOCTYPE HTML
<HTML>

 <HEAD>
      <TITLE>My first HTML document</TITLE>
      <!--- Read just one of MANY java script codes --->
   <script src="https://cdn.anychart.com/releases/8.7.1/js/anychart-base.min.js"></script>
   <script src="https://cdn.anychart.com/releases/8.7.1/js/anychart-data-adapter.min.js"></script>
   <script src="https://cdn.anychart.com/releases/v8/js/anychart-core.min.js"></script>
   <script src="https://cdn.anychart.com/releases/v8/js/anychart-cartesian.min.js"></script>
</HEAD>
  <BODY>
...
<div id="container" style="width: 500px; height: 400px">
      <script>
      anychart.onDocumentReady(function() {
       anychart.data.loadCsvFile("data.csv", function (data) {
          // create the chart
          chart = anychart.scatter();
          // assign the data to a series
          var series1 = chart.marker(data);
          // set title
          chart.title("Sequence Mutations");
          // set axes titles
          chart.xAxis().title("Time (MY)");
          chart.yAxis().title("Mutations (%)");
          // draw chart
          chart.container("container").draw();
        });
      })
      </script>
      </div>

  </BODY>
</HTML>
```



CGI - Common Gateway Interface

- An interface specification that enables web servers to execute an external program (e.g., written in Python, Perl, …), typically to process user requests

- Example files
  - **index10-CGI.html**

    Collect user data and call external python script

```
<FORM ACTION="/cgi-bin/htmlexamples/processform.py" METHOD="POST">
...
<SELECT NAME="title">
<option value="non">Please Choose</option>
<option value="Ms">Ms</option>
<option value="Mr">Mr</option>
<option value="Dr">Dr</option>
<option value="Professor">Professor</option>
</SELECT>
</TD>
</TR>
<TR><TD>Surname:</TD><TD><INPUT SIZE=25 NAM
...
</FORM>
```

process.py

```
…
formData = cgi.FieldStorage()
inputTitle = formData.getvalue("title")
inputSurna = formData.getvalue("surname")
…
```

Password Protecting Website

- BTW - if you ever want to restrict web access to a directory by requiring a username and password, here are some notes. This 'username' and 'password' should not be related to your existing credentials.
- Go to directory ~/public_html and create a file called '**.htaccess**'. The file contains the following lines and you need to set the path to the file '**.htpasswd**' (which you will create in the next step, to store the actual password)

```
AuthName "Restricted Area"
AuthType Basic
AuthUserFile /somewhere-under-your-home-dir/.htpasswd
AuthGroupFile /dev/null
require valid-user
AddType text/html .shtml .htm
AddHandler server-parsed .shtml
AddHandler server-parsed .html
AddHandler server-parsed .htm
Options Indexes FollowSymLinks Includes
```

Password Protecting Website

- You are now going to create the file (not in public_html) called '.htpasswd', which will contain the encrypted password. This file can be stored anywhere (make sure you set the path in file .htaccess) e.g.,

```
/project/home19/suhail/data/.htaccess
```

- Create a user name, let's say, called 'username' (obviously you decide on the username) and set a password

```
cd /somewhere-under-your-home-dir/

htpasswd -c /somewhere-under-your-home-dir/.htpasswd username
```

You will now be asked to type in a password for this user