**Seminar 8**

**Intro to HTML**

# Learning Outcomes

1. Special software to create html pages
2. Use different html tags to create content on web pages

# Outline

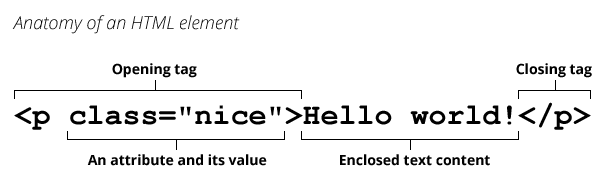
1. Terminology
2. Before we start
3. HTML basics

# Terminology

### Element

An **element** is a part of a webpage. In HTML, an element may contain a data item or a chunk of text or an image, or nothing. A typical element includes an opening tag with some attributes, enclosed text content, and a closing tag.

Some elements can be empty. **Empty element** **cannot** have any children (i.e., nested elements or text).



**Question:** are HTML element and HTML tag the same?

# Before we start

## Tools

*Information is adapted from* [*Mozilla Developer Network*](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/Installing_basic_software)

To work with HTML, we need to have basic software installed. It includes:

* A **text editor**, to write code in. This could be a text editor (e.g. Notepad++, Sublime Text, Brackets, Atom or Visual Studio Code), or a hybrid editor (e.g. Dreamweaver). Office document editors are not suitable for this use, as they rely on hidden elements that interfere with the rendering engines used by web browsers.
* **Web browsers**, to test code in. Currently the most-used browsers are Firefox, Chrome, Opera, Safari, and Internet Explorer. You should also test how your site performs on mobile devices and on any old browsers your target audience may still be using extensively (such as IE 6–8.)

Apart from software mentioned above, Web professionals utilize the following tools:

* A **graphics editor**, like GIMP, Paint.NET, or Photoshop, to make images for webpages.
* A **version control system**, to manage files on servers, collaborate on a project with a team, share code and assets, and avoid editing conflicts. Right now Git is the most popular version control tool, and the GitHub code hosting service, based on Git, is also very popular.
* A **FTP program**, used on older webhosting accounts to manage files on servers (Git is increasingly replacing FTP for this purpose). There are loads of (S)FTP programs available including FileZilla, Fetch, and Cyberduck.
* An **automation system**, like Grunt or Gulp, to perform repetitive tasks automatically, for example minifying code and running tests.
* Templates, libraries, frameworks, etc., to speed up writing common code.
* Even more specific tools.

*You can explore them as a part of your homework.*

## Dealing with files

*Information is adapted from* [*Mozilla Developer Network*](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/Dealing_with_files)

Although you can open a text editor like Notepad and start writing HTML code right away, it is better to set up proper file structure first to organize the work.

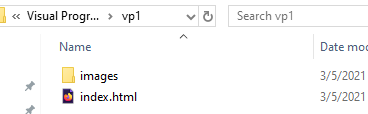
1. Choose a place to store your website projects. E.g., create a new folder called *web-projects* (or similar). This is where all your website projects will live.
2. Inside this first folder, create another folder to store our first webpage in.

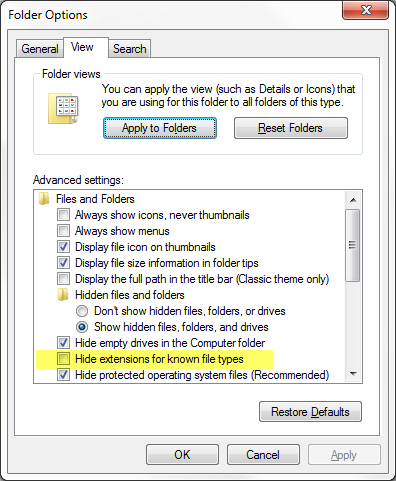
Next, let's look at what structure our test site should have. The most common things we'll have on any website project we create are an index HTML file and folders to contain images, style files, and script files:

* **index.html**: This file will generally contain your homepage content, that is, the text and images that people see when they first go to your site.
* **images** folder: This folder will contain all the images that you use on your site.
* **styles** folder: This folder will contain the CSS code used to style your content (for example, setting text and background colors). For this seminar we do not have any styles; therefore, you do not need to create it now.
* **scripts** folder: This folder will contain all the JavaScript code used to add interactive functionality to your site (e.g. buttons that load data when clicked). Similarly, we do not have any scripts for this seminar, thus no need for such folder yet.

## Task 1

1. Create a folder and call it ***vp1*** (or something more suitable).
2. Using your text editor (Sublime3), create a new file called ***index.html*** and save it inside your ***vp1***folder.
3. Create a folder called ***images*** inside your ***vp1*** folder.
4. Download ***wiut-logo.jpg*** and place it inside *images* folder





**Note:** On Windows computers, you might have trouble seeing the file names, because Windows has an annoying option called *Hide extensions for known file types* turned on by default.

Generally, you can turn this off by following these steps:

* open any folder →
* hit ***Alt*** key to reveal the menu →
* ***Tools*** →
* ***Folder options…*** →
* ***View*** tab →
* uncheck ***Hide extensions for known file types***

**Attention:** acquire a habit to name folders and files completely in lowercase with no spaces.

The reasons are:

* Many computers, particularly web servers, are case-sensitive. So, for example, if you put an image on your website at *test-site/MyImage.jpg*, and then in a different file you try to use the image in code as *test-site/myimage.jpg*, it may not work.
* Browsers, web servers, and programming languages do not handle spaces consistently. For example, if you use spaces in your filename, some systems may treat the filename as two filenames. Some servers will replace the spaces in your filenames with "%20" (the character code for spaces in URIs), breaking all your links. It's better to separate words with dashes or underscores: *my-file.html* or *my\_file.html*.

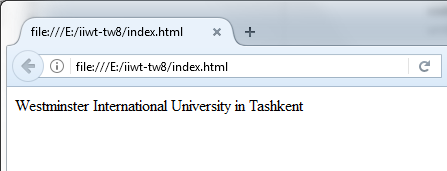
For these reasons, it is best to get into the habit of writing your folder and file names lowercase with no spaces, at least until you know what you're doing. That way you'll bump into fewer problems.

# HTML basics

*Information is adapted from* [*Mozilla Developer Network*](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics)

## 1. Working with elements

* 1. Inside your **index.html** file, paste the following line:

Westminster International University in Tashkent

Now open the index.html using one of the browser installed at WIUT PCs – Firefox, Chrome, Opera, Internet Explorer, etc. You will be able to see that the text appeared inside the web page.

* 1. Go to new line and add more text to the page:

WIUT was founded on the 16th January 2002 in partnership with the University of Westminster (UK, London) and the UMID Foundation of the President of the Republic of Uzbekistan.

Save the page (Ctrl + S) and reload it in browser window (F5). You will see that new text appeared on the same line with the previous. It will remain the same regardless of how many white space (empty lines or spaces) you will add between text pieces in code.

* 1. Make both lines to be **paragraph** elements <p>, so that they will stand by themselves:

<p>Westminster International University in Tashkent</p>

<p>WIUT was founded on the 16th January 2002 in partnership with the University of Westminster (UK, London) and the UMID Foundation of the President of the Republic of Uzbekistan.</p>

Identify where each element starts and ends, what are opening and closing tags are. Save page and reload it inside browser to see the results.

## 2. Nesting elements

You can put elements inside other elements — this is called **nesting**. If we wanted to stress the foundation date of WIUT, we could wrap it in a <strong> element, which means that the word is to be strongly emphasized:

<p>WIUT was founded on the <strong>16th January 2002</strong> in partnership with the University of Westminster (UK, London) and the UMID Foundation of the President of the Republic of Uzbekistan.</p>

You do however need to make sure that your elements are properly nested: in the example above we opened the <p> element first, then the <strong> element, therefore we have to close the <strong> element first, then the <p>. The following is incorrect:

<!-- WRONG: -->

<p>WIUT was founded on the <strong>16th January 2002.</p></strong>

The elements have to open and close correctly so they are clearly inside or outside one another. If they overlap like above, then your web browser will try to make a best guess at what you were trying to say, and you may well get unexpected results. Therefore, don’t do it.

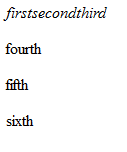
## 3. Block versus inline elements

*The information is taken from* [*Mozilla Developer Network*](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started#Block_versus_inline_elements)

There are two important categories of elements in HTML, which you should know about — block level elements and inline elements.

* **Block level elements** form a visible block on a page — they will appear on a new line from whatever content went before it, and any content that goes after it will also appear on a new line. Block level elements tend to be structural elements on the page that represent e.g. paragraphs, lists, navigation menus, footers, etc. A block level element wouldn't be nested inside an inline element, but it might be nested inside another block level element.
* **Inline elements** are those that are contained within block level elements and surround only small parts of the document’s content, not entire paragraphs and groupings of content. An inline element will not cause a new line to appear in the document: they would normally appear inside a paragraph of text, for example an <a> element (hyperlink) or emphasis elements such as <em> or <strong>.

Take the following example:

<em>first</em><em>second</em><em>third</em>

<p>fourth</p><p>fifth</p><p>sixth</p>

<em> is an inline element, so as you can see in results to the right, the first three elements sit on the same line as one another with no space in between. <p> on the other hand is a block level element, so each element appears on a new line, with space above and below each (the spacing is due to default CSS styling that the browser applies to paragraphs).

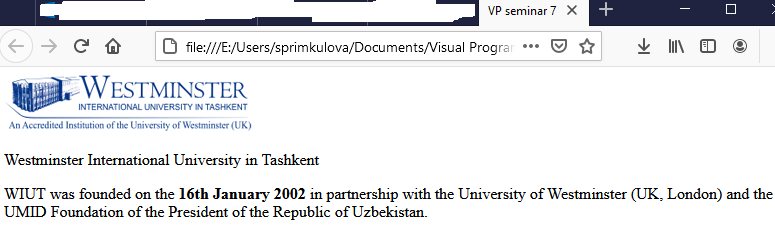
## 4. Empty elements

Some elements have no content, and are called empty elements. For example, insert this <img> element before all paragraphs:

<img src="images/wiut\_logo.jpg" alt="WIUT logo">

This element contains two **attributes**, but there is no closing </img> tag, and no inner content. This is because an image element doesn't wrap content to have an effect on it. Its purpose is to embed an image in the HTML page, in the place it appears.

Save page and refresh it to see the results:



## 5. Anatomy of an HTML document

There are multiple individual HTML elements, but they aren't very useful on their own. Now we'll look at how individual elements are combined to form an entire HTML page. Examine the following code:

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>VP seminar 7</title>

</head>

<body>

<img src="images/wiut-logo.png" alt="WIUT logo">

<p>Westminster International University in Tashkent</p>

<p>WIUT was founded on the <strong>16th January 2002</strong> in partnership with the University of Westminster (UK, London) and the UMID Foundation of the President of the Republic of Uzbekistan.</p>

</body>

</html>

Here we have:

* <!doctype html> — the **doctype**. In 1991-1992, when HTML was young, doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML, which could mean automatic error checking and other useful things. However, these days no-one really cares about them, and they are really just a historical artifact that needs to be included for everything to work right. For now, that's all you need to know.
* <html></html> — the <html> element. This element wraps all the content on the entire page, and is sometimes known as the **root element**.
* <head></head> — the **<head> element**. This element acts as a container for all the stuff you want to include on the HTML page that *isn't* the content you are showing to your page's viewers. This includes things like keywords and a page description that you want to appear in search results, CSS to style our content, character set declarations, and more.
* <body></body> — the **<body> element**. This contains *all* the content that you want to show to web users when they visit your page, whether that's text, images, videos, games, playable audio tracks, or whatever else.
* <meta charset="utf-8"> — this element sets the **character set** your document should use to utf-8, which includes most characters from all known human languages. Essentially it can now handle any textual content you might put on it, which can help avoid some problems later on.
* <title></title> — this sets the **title** of your page, which is the title that appears in the browser tab the page is loaded in, and is used to describe the page when you bookmark/favourite it.

***Homework task:*** *explore the most common <meta> tags. You can use* [*this article*](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/The_head_metadata_in_HTML) *as a start.*

## 6. Images

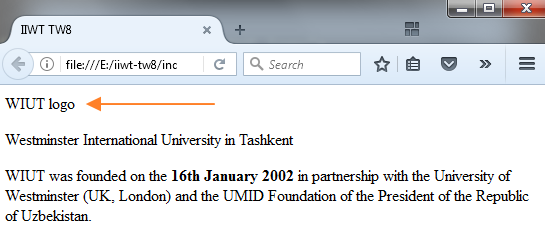
Let's turn our attention to the image element again:

<img src="images/wiut\_logo.jpg" alt="WIUT logo">

As was mentioned before, it embeds an image into our page in the position it appears. It does this via the src (source) attribute, which contains the path to our image file.

We have also included an alt (alternative) attribute. In this attribute, you specify descriptive text for users who cannot see the image, possibly because:

1. They are visually impaired. Users with significant visual impairments often use tools called Screen Readers to read out the alt text to them.
2. Something has gone wrong causing the image to not display. For example, try deliberately changing the path inside your src attribute to make it incorrect. If you save and reload the page, you should see something like this:



The alt text should be descriptive. The alt text you write should provide the reader with enough information to have a good idea of what the image conveys. In this example, our current text of "WIUT logo" is fine, but could be improved.

A better option could be "WIUT logo: the WIUT building and name of the University in blue", since it allows visually impaired people to know in more details what information WIUT logo contains.

***Homework task:*** *Find out more about accessibility at Mozilla Developer Network’s* [*Accessibility page*](https://developer.mozilla.org/en-US/docs/Web/Accessibility)*.*

## 7. Marking up text

### Headings

Heading elements allow you to specify that certain parts of your content are headings — or subheadings — of your content. In the same way that a book has a main title, chapter titles and subtitles, an HTML document can too. HTML contains six heading levels, <h1>–<h6> although you'll commonly only use 3–4 at most:

<h1>My main title</h1>

<h2>My top level heading</h2>

<h3>My subheading</h3>

<h4>My sub-subheading</h4>

Now make a suitable title to your HTML page:

<h1>Westminster International University in Tashkent</h1>

### Paragraphs

As explained before, <p> elements are for containing paragraphs of text; you'll use these frequently when marking up regular text content:

<p>This is a single paragraph</p>

Add more text (you can take it from [www.wiut.uz](http://www.wiut.uz) pages) into one or a few paragraphs, placed after existing content. E.g.:

<p>WIUT adopted the English language as teaching and communication medium. The courses offered by WIUT are important in terms of the development of the national economy and society of Uzbekistan. The university accumulates valuable potentials of the country in education, and the European standards and approaches.</p>

<p>WIUT provides the following courses as a part of its undergraduate programme: BA (Hons) Business Administration, BSc (Hons) in Economics, BSc (Hons) Business Information Systems, BA (Hons) Commercial Law, BSc (Hons) in Economics with Finance</p>

### Lists

A lot of the Web's content is lists and HTML has special elements for these. Marking up lists always consist of at least two elements. The most common list types are ordered and unordered lists:

1. **Unordered lists** are for lists where the order of the items doesn't matter, like a shopping list. These are wrapped in a <ul> element.
2. **Ordered lists** are for lists where the order of the items does matter, like a recipe. These are wrapped in an <ol> element.

Each item inside the lists is put inside an <li> (list item) element.

For example, we can turn the part of our last paragraph into a list by slightly modifying the markup:

<p>WIUT provides the following courses as a part of its undergraduate programme:</p>

<ul>

<li>BA (Hons) Business Administration</li>

<li>BSc (Hons) in Economics</li>

<li>BSc (Hons) Business Information Systems</li>

<li>BA (Hons) Commercial Law</li>

<li>BSc (Hons) in Economics with Finance</li>

</ul>

***Homework task:*** *Try adding an ordered or unordered list to your example page, explore nested lists in HTML. More information can be found in* [*this article*](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals#Nesting_lists)*.*

### Links

Links are very important — they are what makes the Web A WEB. To add a link, we need to use a simple element — <a> — the *a* being short for "anchor". To make text within your paragraph into a link, follow these steps:

1. Choose some text. In our example, we can use "courses" to provide a link to information about courses at WIUT website.
2. Wrap the text in an <a> element, like so:

<a>courses</a>

1. Give the <a> element an href attribute, like so:

<a href="">courses</a>

1. Fill in the value of this attribute with the web address that you want the link to link to:

<a href="http://www.wiut.uz/study-at-wiut/wiut-courses">courses</a>

You might get unexpected results if you omit the https:// or http:// part, called the **protocol**, at the beginning of the web address. After making a link, click it to make sure it is sending you where you wanted it to.

**Note:** href might appear like a rather obscure choice for an attribute name at first. If you are having trouble remembering it, remember that it stands for **Hypertext REFerence**.

Add a link to your page if you haven’t done so. Save the page and reload it to see the results.

***Homework task:*** *read about creating links in more details* [*here*](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Creating_hyperlinks)

## Task 2

If you have followed all the instructions above, you should end up with a page that looks like the one below:

**5-10 minute exercise:** use materials from [www.wiut.uz](http://www.wiut.uz) to add more content to the page. Try to add the following:

* **headings** of different levels
* **more images** – take them from [University news page](http://www.wiut.uz/university-news) or use image search online
* **lists** – both ordered and unordered
* **abbreviations** – explore <abbr> tag and try to add it where appropriate

**Answer the following questions** (you can search online to find the answers):

1. What is the difference between
   1. <strong> and <b>?
   2. <em> and <i>?
2. What <u> tag does and why do we need to be careful while using it?

***Hint:***[*read here*](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals#Italic_bold_underline...)

# Special characters in HTML

*Information is adapted from* [*Mozilla Developer Network*](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started#Entity_references_including_special_characters_in_HTML)

In HTML, the characters < (less than), > (greater than), " (double quotation mark), ' (apostrophe or single quotation mark) and & (ampersand) are special characters. They are parts of the HTML syntax itself, so how do you include one of these characters in your text, for example if you really want to use an ampersand or less than sign, and not have it interpreted as code as some browsers may do?

We have to use character references — special codes that represent characters, and can be used in these exact circumstances. Each character reference is started with an ampersand (&), and ended by a semi-colon (;).

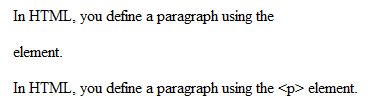
|  |  |
| --- | --- |
| Literal character | Character reference equivalent |
| < | &lt; |
| > | &gt; |
| " | &quot; |
| ' | &apos; |
| & | &amp; |

In the below example, you can see two paragraphs, which are talking about web technologies:

<p>In HTML, you define a paragraph using the <p> element.</p>

<p>In HTML, you define a paragraph using the &lt;p&gt; element.</p>

In the output below, you can see that the first paragraph has gone wrong, because the browser thinks that the second instance of <p> is starting a new paragraph. The second paragraph looks fine, because we have replaced the angle brackets with character references.



**Note:** A chart of all the available HTML character entity references can be found on Wikipedia: [List of XML and HTML character entity references](https://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references).

# HTML comments

*Information is adapted from* [*Mozilla Developer Network*](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started#HTML_comments)

In HTML, as with most programming languages, there is a mechanism available to write comments in the code — comments are ignored by the browser and invisible to the user, and their purpose is to allow you to include comments in the code to say how your code works, what the different parts of the code do, etc. This can be very useful if you return to a code base that you've not worked on for 6 months, and can't remember what you did — or if you hand your code over to someone else to work on.

To turn a section of content inside your HTML file into a comment, you need to wrap it in the special markers <!-- and -->, for example:

<p>I'm not inside a comment</p>

<!-- But I am -->

The first paragraph appears in the output, but the second one doesn't.



# Task 3. Explore advanced text formatting

Use the following article to explore the advanced text formatting: <https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting>

After reading the article, practice with formatting imaginary letter – a fictional response from a research fellow to a prospective PhD student concerning their application to work at the university.

The task details can be found there: <https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Marking_up_a_letter>

**Tips:**

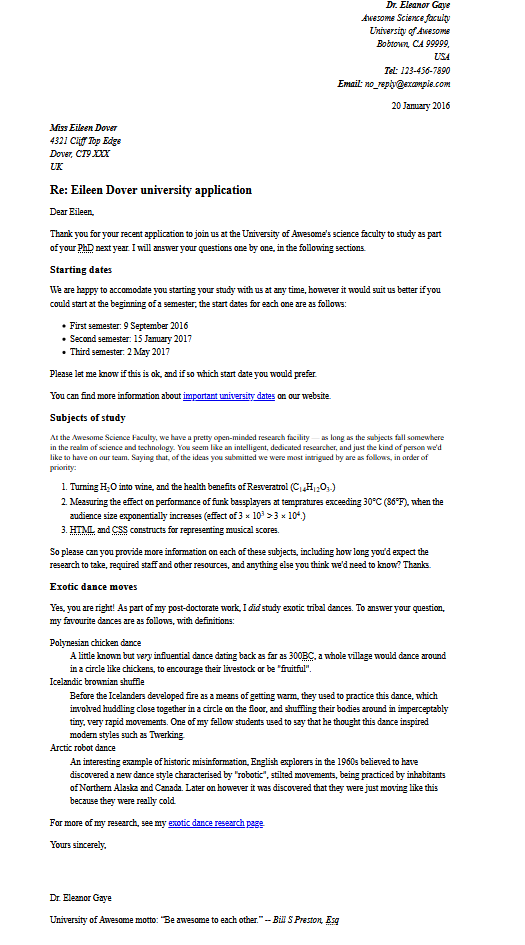
1. Work in new HTML file called **letter.html**
2. Create **styles** folder and put there **letter.css** file that you can download from Intranet
3. Inside <head> element of letter.html paste the following code. It will allow to use the styles from letter.css in your HTML page

<link rel="stylesheet" href="styles/letter.css">

1. Wrap the recipient details into the <address> element with the class attribute set to “receiver-column”. It will allow to use styles set inside letter.css

<address class="receiver-column">

1. Use <br> element for line breaks
2. The letter.html should have the following formatting at the end.



# Homework

1. Explore the most common <meta> tags and discuss them during online seminars. You can use [this article](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/The_head_metadata_in_HTML) as a start.
2. Find out more about accessibility at Mozilla Developer Network’s [Accessibility page](https://developer.mozilla.org/en-US/docs/Web/Accessibility).
3. Try adding an ordered or unordered list to your example page. Explore nested lists in HTML. More information can be found in [this article](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals#Nesting_lists).
4. Read about creating links in more details [here](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Creating_hyperlinks). Focus on:
   1. Using title attribute to provide more information
   2. Block level links
   3. Links to document fragments
   4. Absolute vs. relative links
   5. Link best practices
   6. Mailto links
5. Read about debugging HTML code [here](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Debugging_HTML).