HTML5

What is HTML

HyperText Markup Language (HTML) is a markup language used to create and structure web pages. The language is interpreted by web browsers which create a web page based on the browser interpretation of the HTML document.
HTML was first introduced in 1991 and was highly influenced by another markup language called SGML.
At it's basic form HTML is a very simple text file with rules that are not heavily enforced by the web browsers, meaning even if the document contains bad structure most of the browsers will be able to display a page based on the understanding of the browser.
Due to the fact that browsers could display a page even with bad HTML the web started to fill with non standard pages and browsers displayed the pages differently, in 1994 the W3C was founded with the goal to bring a standard to the web. Development of the HTML language was now supervised by W3C.
Due to the HTML language being for non restrictive in 2002 HTML was combined with a more strict language called XML to create XHTML this standard allowed us to enforce stricter rules on our HTML documents.
In 2014 HTML5 was released and that will be the version we will cover in this course. HTML pages can also be combined with CSS files which describes the design styles of an HTML document. HTML can also be combined with JavaScript code which is the interpreted scripting languages that the browsers understand.

HTML Tags

Tag Types

There are two types of tags in HTML:

- Pair tags
  Pair tags will have an opening tag and a closing tag. The syntax of the opening tag is: **<tag_name>** and the syntax of the closing tag is: **</tag_name>** and between those tags is the content of the tag. example of such a tag is the paragraph tag: ** paragraph text ** .
- Unpaired tags
  Certain tags are embedded with the opening and closing tag combined in one. The general syntax is: **<tag_name />**. For example the horizontal line tag: **<hr/>**
  There is no reason to put content in that tag so this tag is Unpaired

HTML Skeleton

This is the general structure of an HTML page:

```
<!DOCTYPE html>
<html>
<head>
  <title>Example Page</title>
</head>
<body>

</body>
</html>
```

Let's go over the lines here:

- **DOCTYPE -** the doctype line must be the first line in your HTML document, this is not an HTML tag it's used for the browsers to determine what version of HTML we are writing our document. And this course we will always write **<!DOCTYPE html>** which instructs the browser that we are using HTML5
- **html -** the html tag tells the browser that this is an HTML Document
- **head -** the head tag contains the following
    - Meta information about our page
    - Js scripts to load
    - CSS styles to include
    - Title of our page in the browser
- **title -** the title of our site in the browser window
- **body -** our page layout will be here

Student exercise

Open your notepad and write your first HTML document with an hello world message. Try and open the file with your browser to check that everything is working

Important Meta tags

The job of these tags is to present information about the data that is currently presented and to instruct the browser how to serve you page.
The meta tags can either have the **name** attribute or **charset** attribute or **http-equiv** attribute. where each of them has it's own job:

- the **name** supply data about the page
- **http-equiv** tells the browser how to serve you page
- **charset** tells the browser about the encoding in your page

Let's go over the **meta** tags that are considered a must in web pages or are often used

- **charset -** character encoding in this page, use **UTF-8**
- **<meta http-equiv="refresh" content="10" />** set the browser to refresh the page every 10 seconds
- **<meta name="author" content="Nerdeez Ltd" />** author of the page
- **<meta name="description" content="This site explain HTML" />** short description about your site
- **<meta name="keywords" content="html, meta, div" />** contains comma separated keywords that relate to your app most searchbots are not looking at this list
- **<meta name="robots" content="index, nofollow, noarchive, nocache" />** instruction for the search bots, this can be comma separated values according to the table in this link for now we set the bots to crawl our site but not follow any links
- **<meta name="viewport" content="width=device-width, initial-scale=1.0" />** this tells the browser how to serve the site on mobile devices

Text Tags

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Text Tags</title>
</head>
<body>

<!-- Text Tags -->
<h1>
  H1 - header1
</h1>
<h2>
  H2 - header2
</h2>
<h3>
  H3 - header3
</h3>
<h4>
  H4 - header4
</h4>
<h5>
  H5 - header5
</h5>
<h6>
```

```
  H6 - header6
</h6>
<p>
  P - paragraph
</p>

<!-- Other text styling -->
<p>
  <b>bold</b>
  <i>italic</i>
  <strong>Important text</strong>
  <small>small text</small>
</p>

</body>
</html>
```

Table

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Table</title>
</head>
<body>

  <h1> Table one row one col - without border</h1>
  <table>
      <tr>
          <td>1st col</td>
      </tr>
  </table>

  <h1> Table one row two cols - with border</h1>
  <table border="1">
      <tr>
          <td>1st col</td>
          <td>2nd col</td>
      </tr>
  </table>
```

```
  <h1>Table two rows 3 cols - colspan</h1>
  <table border="1">
      <tr>
          <td>1 col</td>
          <td>2 col</td>
      </tr>
      <tr>
          <td colspan="2">3 col</td>
      </tr>
  </table>

  <h1>Table two rows 3 cols - rowspan</h1>
  <table border="1">
      <tr>
          <td>1 col</td>
          <td rowspan="2">2 col</td>
      </tr>
      <tr>
          <td>3 col</td>
      </tr>
  </table>

</body>
</html>
```

div

div is the tag we are going to use when structuring our web page. Each div represents a block containing a part of the layout of our page. Each div can take an entire row on the screen or just part of the row. Let's try and demonstrate how we layout our page using **div**

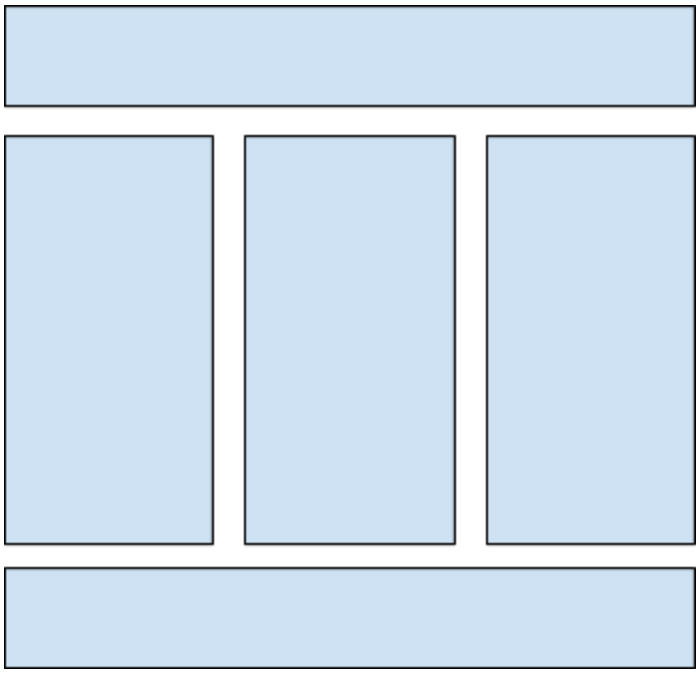
```
<!DOCTYPE html>
<html style="height: 100%">
<head>
  <meta charset="UTF-8">
  <title>Div Example1</title>
</head>
<body style="height: 100%">

  <!-- begin menu section -->
  <div style="border: 1px solid black; height: 100px; background-color: pink">
  </div>
```

```
  <!-- end menu section -->

  <!-- begin left side -->
  <div style="float: left; width: 30%; border: 1px solid black; height: calc(100% - 100px);
background-color: olive">
  </div>
  <!-- end left side -->

  <!-- begin right side -->
  <div style="float: right; width: 69%; border: 1px solid black; height: calc(100% - 100px);
background-color: antiquewhite;">
  </div>
  <!-- end right side -->

</body>
</html>
```

Notice that we added inside the div tags an attribute for styling the div, later in the course we will learn how to embed the styling in an external css file.

Student exercise

Use div to create the following layout:



Lists

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>

  <h1>UnOrdered List</h1>
  <h3>Shopping list</h3>
  <ul>
      <li>
          Banana
      </li>
      <li>
          Apple
      </li>
      <li>
          Potatoes
      </li>
  </ul>

  <h1>Ordered List</h1>
  <h3>Ordered Todo</h3>
  <ol>
      <li>
          wake up
      </li>
      <li>
          Brush my teeth
      </li>
      <li>
          Goto work
      </li>
  </ol>

</body>
</html>
```

Links

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>

  <h1>Links</h1>

  <!-- relative link -->
  <a href="div1.html" >div1</a>

  <!-- absolute -->
  <a href="https://www.google.com" >google</a>

  <!-- targets -->
  <a href="Lists.html" target="_blank">blank</a>

  <!-- Scroll link -->
  <a href="#down">go down</a>

  <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
  <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
  <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
  <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
  <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
  <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
  <h1 id="down">
     scroll here
  </h1>
</body>
</html>
```

Images

You can load an image by placing the **img** tag. The path of the image is inside the **src** attribute which can be a relative path or an absolute pate.
Example:

```
<!-- Relative path -->
```

```
<img src="img.png" height="30" width="40" />

<!-- Absolute path -->
<img src="https://www.nerdeez.com/assets/img.png" height="30" width="40" />
```

scripts

We can load external JS scripts by using the script tag:
```
<script src="app.js"></script>
```
When the browser loads a script it pause the rendering of the page until the script is loaded, it's recommended to put the loading of the scripts at the bottom of the **body** so the entire page will be rendered before the scripts load.

link

Although we can place CSS styling inside each HTML element as an attribute, it is common to place the styling in an external CSS file. You place the **link** inside the head section in the following syntax:
```
<link rel="stylesheet" href="style.css" />
```

Forms

One of the main ways for user to interact with our web application is by using **form**. A form is a location in the screen for user to place data and submit the form and send the data back to the server. The form contains an attribute called **method** which determines the request type and an attribute called **action** which determines the location to send the request with the data from the form.
Example of a form
```
<form method="get" action="./"></form>
```

Method can be **get** and the data in the form will be submitted with URL get params.
And the method can be **post** where the data will be sent in the request **body.**
Let's try and create a simple form with a single text input and a submit button.

```
<form method="get" action="">
  <label>Username</label>
  <input type="text" name="username" />
  <button type="submit">
    Submit
  </button>
```

</form>

We placed the form and by leaving the action to empty it will submit the data to the same page we are currently in.
We are submitting the data with the method **get** so we should see in the same page the data is appended to the url via query params.
We want to place a text input in our form so we placed an **input** of type **text** . in our **input** we also placed an attribute in our tag that is called **name** which will map the data in the input to a url query param with the name set to **username**.
We also added a **button** tag with the attribute **type** equals **submit** which will submit the form when pressing the button.
If you open this with your browser you should see that if you place a text in the input and press submit it will add to the url: **username=<text you placed in the text box>**

Inputs

Let's go over the type of inputs we have.

Text
<input type="text" name="username" maxlength="5" />

Regular text input, we also placed a **maxlength** attribute so the user wont be able to type more than 5 characters

Password
<input type="password" name="password" minlength="5" />

This will hide what the user is typing. We also placed a **minlength** attribute, try and press the submit button when typing under 5 characters you should see that the browser will mark the field as error. You can also disable the automatic validation on form by placing on the wrapping form the attribute **novalidate**

Checkbox
<input type="checkbox" name="agree" value="agreeToTerms" required />

This will place a checkbox on the form and when marked in will send the **value** attribute, otherwise it won't sent nothing. Placing a **required** attribute will validate when the form is submitted that this checkbox is checked

Radio button
<input type="radio" name="agree" value="1" required/>
<input type="radio" name="agree" value="2" required/>

<input type="radio" name="agree" value="3" required/>

You can use the radio buttons when you have a single variable (in our example the name of the variable is **agree**) with multiple value options (in our example the values are: 1,2,3)

Button
You can place a **submit** button that will submit the form or a regular button, you can use the **button** tag or the **input type=”button”**

Student Exercise

Create a form to create a Task object.
Task object contains:

- title : string
- description : string
- group : string
- when : string (we will replace it in the future to date)

Submitting the form should add all the params to the current url

HTML Attributes

We can add attributes to HTML tags,
Attribute syntax is like this:
<html_tag attribute_name=”attribute_value”></html_tag>
so let’s go over some common attributes we can add:

id

Id is a way to give a unique name to an html element. The syntax is:
** **
You value of the id can’t be repeated on another element. This attribute is usefull when trying to reference the element with JavaScript or style the element with css

class

Class is an attribute that allows us to name a group of HTML elements. This will be useful when we go over css, but let’s imagine that we have a navigation bar, probably every link in that navigation bar will look the same. So our navigation bar code will be similar to this:
<nav>

```
        <ul>
                <li class=”nav-li”>
                        <a href=”about.html”>About</a>
                </li>
                <li class=”nav-li”>
                        <a href=”contact.html”>Contact</a>
                </li>
        </ul>
</nav>
```

Notice that every **li** in the navbar we added a class attribute to allow us to place style that will effect all items in the class.
The class also allows us to reference a group of elements in JavaScript code.

style

You can style an HTML element by placing a style attribute and placing a css code inside the expression. For example:
<h1 style=”color:red; background-color: blue”>Style me</h1>
We just place a title with red color font and a background color of blue. In our next chapter we will cover CSS and you will learn what other attributes you can add here.

Summary

This is just the tip of the iceberg in HTML topic and there are many more to cover, but all the other stuff we will learn as we start to become professional web developers