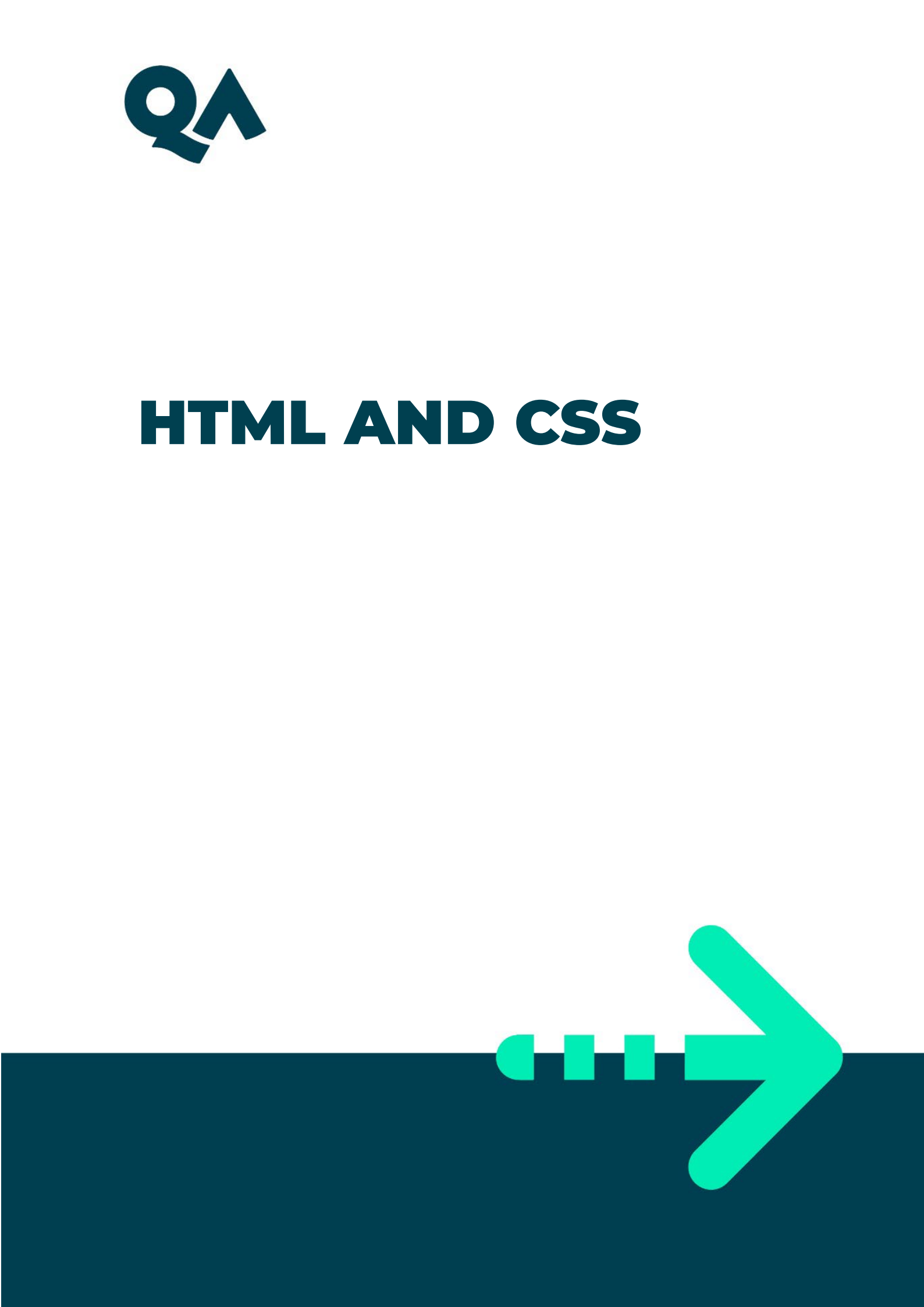
HTML AND CSS

Get user input using forms

Objectives

To understand how to use forms in HTML.

Reference material

This exercise is based on material from the Get user input using forms chapter.

Overview

- In this exercise you will:
  - Create a form.
  - See how information from you form is sent to a web server.

Estimated duration

The estimated duration for this lab is 20 minutes.

Completed solution

There is a completed solution for this lab.

Step by step instructions

1. Explanation: shortly, you will add a form to your webpage, asking the user to submit their details so you can send them further information.

   The details the user enters on this form will need to be handled by your web server – probably stored on a database so that you can retrieve them later.

   For the purpose of this exercise, we are not going to store anything on a database. But we are going to send the details to a web server, then use ASP to generate a webpage that includes some of those details. You do not need to understand ASP because we have already written and supplied you with the ASP web page.

   However, it is important that you understand that ASP is a technology which runs on the web *server* (*not* the web browser, or client), which generates an HTML page.

2. Navigate to the Resources\Chapter 4 - Get User Input Using Forms folder. In there, you will find a file called MoreDetails.asp. Select this file, and add it to your website.

3. Now, open your webpage, **index.html**, in Code. To the bottom of the page (or somewhere else if there's a more suitable place), add a heading: **Sign Up for More Information**. Add some text explaining to users what they will be signing up for.
4. After any text, add a form as follows:
    - Action: **MoreDetails.asp**
    - Method: **get**
    - Contents:
        I. One <input> with a **type** of **text** and a **name** of **Name**.
        II. One <input> with a **type** of **email** and a **name** of **Email**.
            - Ensure that the **name** attributes of the input fields are set correctly.
            - This is important, because the asp page will be looking for a piece of data with the correct name.
        III. One <input> with a **type** of **submit**.
            - Give the submit button a suitable value.
    - Ensure that the <input> tags are labelled with some appropriate text, so the user knows what each is for.

    Hint: <form> elements may contain <p> elements, but not the other way around. Use this (and any other formatting skills you like) to ensure your form looks attractive.
5. Load your page by pressing **Alt-B** in Code, and try it. It doesn't work. Why do you think that is?

    The reason it doesn't work is because ASP requires a **web server**. You are not telling your browser to get the page from a web server – you are loading it directly from the file system. To fix this, navigate to **C:\inetpub\wwwroot**, and create a new folder called QA. Copy your website files to the new QA folder, then open the following page in your web browser: http://localhost/QA/index.html. Now try your form again and check it works correctly this time. This is because the web browser is now fetching the web page from the IIS web server that's running on your computer (which you enabled in Lab 3).
6. Notice that, at the bottom of the More Details page, it says that the form was submitted using the Get method. The reason it says that is because we've put some code into the ASP page which determines what method was used to submit the form, and to include an appropriate message on the generated page.

    Look at the address bar, and notice that the URL includes the name and e-mail address that you entered. (Some of the characters may have been encoded to make sure they get sent correctly, and the web server will have decoded them, but this should not prevent you from seeing the information.)

7. Close your browser. Edit the <form> element in your page, and change its **method** to **post** (make sure you edit the version in **C:\inetpub\wwwroot\qa)**. Test it again.

   This time, notice that the message at the bottom of the page has changed. This clearly shows that the server is able to tell whether a page has been submitted using Get or Post.

   Also, notice that the URL no longer contains any clues as to what information you entered. This means that this page cannot be bookmarked, and it would be harder for the user to modify the submitted information.

   Discussion: when should we use Get, and when should we use Post?

   In general, you should use Get when a) the information being submitted is used to query a database, and not actually being stored in the database, and b) you want the user to be able to bookmark the page.

   You should use Post when you don't want the user to be able to bookmark the page together with the submitted data. This particularly applies when the submitted data is going to be stored in the database – you don't want the user to be able to store the same piece of information twice.

   You should also use Post when there is a significant amount of data being submitted. Get requests are not suitable for lots of data because all of the data has to be added to the URL, which makes the URL unmanageably long.

When you have time:

8. Have a look through the PowerPoint presentation for different types of input fields that can be used in forms. Add some of these to your web page. Experiment with how to make them look good, and learn when to use each one.

When you have even more time:

9. Have a look at the ASP page we have supplied. See if you can work out how it works. See if you can add extra code to it, to handle the new inputs you've added to your form.