In-class Lab 01

ASP.NET Core MVC

Contents

0 Introduction **1**

1 Basic HTML **2**
1.1 outer elements . . . 2
1.2 head and body elements . . . 2
1.3 head tags . . . 2
1.4 headers . . . 2
1.5 paragraphs . . . 3
1.6 formatting . . . 3
1.7 generic elements . . . 3
1.8 empty elements . . . 3
1.9 hyper links (anchors) . . . 4

2 HTML Tables **4**
2.1 table element . . . 4
2.2 table row elements . . . 5
2.3 row header elements . . . 5
2.4 row data elements . . . 5
2.5 spanning multiple rows and columns . . . 6

3 HTML Forms **7**
3.1 creating a form . . . 7
3.2 single valued form inputs . . . 8
3.3 multi-valued form inputs . . . 8
3.4 text areas . . . 9
3.5 submit and reset inputs . . . 9

4 Navigation, JavaScript event handlers **10**
4.1 tabindex, accesskey . . . 10
4.2 onmouseover, onmouseut . . . 10
4.3 onfocus, onblur . . . 10
4.4 onload, onunload . . . 10
4.5 onclick, ondblclick . . . 10

0 Introduction

In this lab, we will have a very brief review of HTML. If you need more than this brief review, you can find some good tutorials online. One very good tutorial is at `https://www.w3schools.com/html/default.asp`. For this lab, we will be using the HTL 4.01 specificatio found at `https://www.w3.org/TR/html401`. The canonical reference to all things HTML is the official specification, at `https://www.w3.org/TR/html52/`.

This lab consists of four parts. Part 1 is basic HTML, Part 2 is HTML tables. Part 3 is HTML forms. Part 4 is JavaScript event handlers. You will complete this lab step by step.

1 Basic HTML

1.1 outer elements

HTML consists of a series of tabs, like this: `<element> ...</element>`. Tags can have content, which may consists of ordinary text or other tags, like this: `<paragraph>This is a very short paragraph.</paragraph>`. Tags can be empty, which means that they have no content, like this hard line break: `<br />`. Optionally, tags may have attributes, which are key-value pairs containing some setting, like this: `<paragraph id='first-paragraph' color='red'>This paragraph has a couple of attributes.</paragraph>`.

An HTML page has outer HTML tabs, with a DOCTYPE header. The DOCTYPE header tells the rendering agent what type of document to render. Type these commands in your favorite editor, save the file with a name and a file extension of `.html`. Then, open the document in your favorite browser using `Cntl-O` (control-O opens a dialog box that will allow you to select the file to open. Note that the text "Hello world" renders as plain text with no formatting.

```
1 <!DOCTYPE html>
2 <html>
3     Hello world
4 </html>
```

1.2 head and body elements

HTML documents usually contain a `head` element and a `body` element. Comments are distinguished by the comment characters: `<!-- ...-->`.

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4     </head>
5     <body>
6         <!-- this is a comment -->
7         Hello world
8     </body>
9 </html>
```

1.3 head tags

An HTML head element may contain tags that apply globally to the document, as shown. Revise your document to include these.

```
1     <head>
2         <title>My first web page</title>
3         <meta name='author' content='Charles Carter' />
4         <meta name='description' content='This is my first web page.' />
5         <script>
6             <!-- no script yet -->
7         </script>
8         <style>
9             <!-- no style yet -->
10        </style>
11    </head>
```

1.4 headers

HTML has six levels of headers. Headers are *block elements*, meaning that they are stacked vertically, on to of each other. Normally, only the first two levels are used, but sometimes you will see three levels. Rarely do you see more than three levels of headers. Header tags have both an opening and a closing tag.

```
    <body>
        <h1>H1 Level Header</h1>
        <h2>H2 Level Header</h2>
        <h3>H3 Level Header</h3>
        <h4>H4 Level Header</h4>
        <h5>H5 Level Header</h5>
        <h6>H6 Level Header</h6>
        Hello world
    </body>
```

1.5 paragraphs

Paragraphs are also block elements. Paragraph tags have both an opening and a closing tag.

```
    <body>
        <h1>My first web page</h1>
            <p>Hello world</p>
            <p>How do you like it?</p>
    </body>
```

1.6 formatting

In older versions of HTML, formatting tags were used to format text. *This is not recommended, so DON'T DO IT!!!* The following is simply an illustration of *in-line elements*. In-line elements are "stacked" horizontally.

```
    <body>
        <h1>My first web page</h1>
            <p>Hello world. How do you like it?</p>
            <p>In-line tags display across the page.
                <i>This is italic.</i>
                <b>This is bold.</b>
                <b><i>This is bold italic. Notice that the tags must be closed
                in the reverse order of opening.</i></b></p>
    </body>
```

1.7 generic elements

HTML allows both generic block elements and in-line elements. The generic block element is `<div>`. The generic in-line element is `<span>`. NOTE: I have formatted these lines to emphasize the behavior of these elements — this is an example of how *NOT* to format HTML.

```
        <h1>Examples of DIV and SPAN</h1>
             <div>This is the first div element.</div> <div>This is the second div
                 element.</div>
             <div>This is the third div element.</div> <span>This is the first span
                 element.</span>
             <span>This is the second span element.</span> <span>This is the third
                 span element.</span>
```

1.8 empty elements

Two empty elements that you will see are the hard line break, `<br />` and the horizontal rule, `<hr />`. Using the previous example (with good formatting) we place a line at the top and bottom, and a break between each span. Notice that empty tags are terminated by the forward slash, which indicates to the rendering agent that no closing tag follows.

```
        <h1>Examples of HR and BR</h1>
            <hr />
             <div>This is the first div element.</div>
             <div>This is the second div element.</div>
             <div>This is the third div element.</div>
            <br />
             <span>This is the first span element.</span>
            <br />
             <span>This is the second span element.</span>
            <br />
             <span>This is the third span element.</span>
            <hr />
```

1.9 hyper links (anchors)

You may be wondering why HTML is **Hypertext Markup Language** amd HTTP is **Hypertext Transfer Protocol**. "Hypertext" allows links between resources, usually between resources on different pages, sites, or servers, but also within the same document. The *anchor element* accomplishes this. Using the previous example, we place a link at the top of the document to the bottom of the document and *vice versa*. We also link to various sites of interest. Anchor elements are in-line elements.

```
            <a id='top' />
            <a href='#bottom'>Go To Bottom</a>
        <h1>Examples of anchor tags</h1>
            <hr />
             <div>This is the first div element.</div>
             <div>This is the second div element.</div>
             <div>We need to create some space on the page to see the jumps.</div>
            <br /> <br /> <br /> <br /> <br /> <br /> <br /> <br />
            <br /> <br /> <br /> <br /> <br /> <br /> <br /> <br />
            <br /> <br /> <br /> <br /> <br /> <br /> <br /> <br />
             <span>This is the first span element.</span>
            <br />
             <span>This is the second span element.</span>
            <br />
             <span>This is the third span element.</span>
            <hr />
        <h1>Examples of hyperlinks</h1>
        <a href='http://microsoft.com'>Microsoft</a> <br />
        <a href='http://erau.edu'>Embry Riddle</a> <br />
        <a href='http://www.benning.army.mil/garrison/dhr/ACES.html'>ACES</a> <br />
            <a id='bottom' />
            <a href='#top'>Go To Top</a>
```

2 HTML Tables

2.1 table element

HTML tables are built by adding rows. There is no way to build HTML tables by adding columns. Unfortunately, this adds to the complexity of HTML tables. Despite this, tables are very common and very useful.

Tables are created by the `<table>` tag. Table elements have content, so they must be closed by an end tag. This example uses the *rules* attribute, which places a line between the named elements. NOTE: The rule attribute has been removed from HTML 5 — the preferred way to separate table elements is by using CSS. The permitted values of the rule attribute are: none, all, rows, columns, and groups.

Tables may also have an optional `<caption>` element. Captions must have opening and closing tags. The content of the caption is informational text about the table.

```
        <h1>Example of an HTML table</h1>
            <table rules='all'>
```

```
            <caption>This is a table</caption>
        </table>
```

2.2 table row elements

HTML table rows are created by the <tr> tag. Note that TR means "Table Row." The table row element has both opening and closing tags.

```
    <h1>Example of an HTML table</h1>
        <table rules='all'>
            <caption>This is a table</caption>
            <tr>
                <!-- the table cells go here -->
            </tr>
        </table>
```

2.3 row header elements

HTML row headers are created by the <th> tag. Note that TH means "Table Header." The row header element has both opening and closing tags.

```
    <h1>Example of an HTML table</h1>
        <table rules='all'>
            <caption>Programming Languages</caption>
            <tr>
                <th>Language</th>
                <th>Creator</th>
                <th>Year</th>
                <th>Paradigm</th>
            </tr>
        </table>
```

2.4 row data elements

HTML row headers are created by the <td> tag. Note that TH means "Table Data." The row data element has both opening and closing tags.

```
    <h1>Example of an HTML table</h1>
        <table rules='all'>
            <caption>Programming Languages</caption>
            <tr>
                <th>Language</th>
                <th>Creator</th>
                <th>Year</th>
                <th>Paradigm</th>
            </tr>
            <tr>
                <td>Fortran</td>
                <td>John Backus</td>
                <td>1953</td>
                <td>Procedural</td>
            </tr>
            <tr>
                <td>Lisp</td>
                <td>John McCarthy</td>
                <td>1958</td>
                <td>Functioal</td>
            </tr>
            <tr>
                <td>COBOL</td>
                <td>Grace Hopper</td>
                <td>1959</td>
                <td>Procedural</td>
```

```
                </tr>
                <tr>
                    <td>C</td>
                    <td>Thompson &amp; Ritchie</td>
                    <td>1972</td>
                    <td>Procedural</td>
                </tr>
            </table>
```

2.5 spanning multiple rows and columns

The `rowspan` and `colspan` attributes to the table header and table data elements permit the spanning of multiple rows and columns.

spanning multiple columns

```
        <h1>Example of spanning multiple columns</h1>
            <table rules='all'>
                <caption>Programming Languages by Paradiogm</caption>
                <tr>
                    <th colspan='3'>Procedural Languages</td>
                </tr>
                <tr>
                    <td>COBOL</td>
                    <td>Grace Hopper</td>
                    <td>1959</td>
                </tr>
                <tr>
                    <td>C</td>
                    <td>Thompson &amp; Ritchie</td>
                    <td>1972</td>
                </tr>
                <tr>
                    <td>Perl</td>
                    <td>Larry Wall</td>
                    <td>1975</td>
                </tr>
                <tr>
                    <th colspan='3'>Functional Languages</td>
                </tr>
                <tr>
                    <td>Lisp</td>
                    <td>John McCarthy</td>
                    <td>1958</td>
                </tr>
                <tr>
                    <td>Haskell</td>
                    <td>FPCA Conference</td>
                    <td>1990</td>
                </tr>
                <tr>
                    <th colspan='3'>Object Oriented Languages</td>
                </tr>
                <tr>
                    <td>Java</td>
                    <td>James Gosling</td>
                    <td>1990</td>
                </tr>
                <tr>
                    <td>C Sharp</td>
                    <td> Anders Hejlsberg</td>
                    <td>1999</td>
                </tr>
            </table>
```

spanning multiple rows

```
        <h1>Example of spanning multiple rows</h1>
            <table rules='all'>
                <caption>Programming Languages by Paradiogm</caption>
                <tr>
                    <th rowspan='3'>Procedural Languages</th>
                    <td>COBOL</td>
                    <td>Grace Hopper</td>
                    <td>1959</td>
                </tr>
                <tr>
                    <td>C</td>
                    <td>Thompson &amp;amp; Ritchie</td>
                    <td>1972</td>
                </tr>
                <tr>
                    <td>Perl</td>
                    <td>Larry Wall</td>
                    <td>1975</td>
                </tr>
                <tr>
                    <th rowspan='2'>Functional Languages</th>
                    <td>Lisp</td>
                    <td>John McCarthy</td>
                    <td>1958</td>
                </tr>
                <tr>
                    <td>Haskell</td>
                    <td>FPCA Conference</td>
                    <td>1990</td>
                </tr>
                <tr>
                    <th rowspan='2'>Object Oriented Languages</th>
                    <td>Java</td>
                    <td>James Gosling</td>
                    <td>1990</td>
                </tr>
                <tr>
                    <td>C Sharp</td>
                    <td> Anders Hejlsberg</td>
                    <td>1999</td>
                </tr>
            </table>
```

3 HTML Forms

The HTTP protocol, and HTML, were originally developed to allow nuclear physicists to read the research of other nuclear physicists. As such, the primary purpose was downloading content on a server to a client. However, HTTP works in both directions. The HTML `<form>` element allows content to be uploaded from the client to the server.

3.1 creating a form

The form element has a number of attributes, two of which are the `method` and the `action` attributes. The method attribute specifies HTTP method, usually either GET or POST. You will also see the `enctype` attribute, which specifies the document's content.

```
        <h1>My first HTML form</h1>
        <form method='get' action = ''>
            <p>This is my form.</p>
        </form>
```

3.2 single valued form inputs

HTML form controls send key/value pairs to the web server. The key consists of the *name* of the control. The value consists of the content that the user specifies. To begin with, we will look at the `<input>` element. Input has a large number of attributes, and we will look at five in this section. All of these inputs are single valued inputs.

The *hidden* input control does not display to the user. It is used to pass useful information back to the web server without any action by users. Even though this control does not appear to the user, it is not secure, as users can easily edit the control by hand and change the value specified by the author. In this example, the name of the control is "not_shown" and the value is "This does not appear to the user." When the form inputs are returned to the server, this input will result in this: `not_shown='This does not appear to the user.'`.

The *text* input control results in a box where the user may enter some text. The name of this control is "my_name". The control also has an ID attribute, which is common as many server interfaces do not recognize the name attribute. The value is set to "Enter your first name." The text `onfocus='my_name.value = "" '` is a JavaScript command that states that, when the element receives focus (that is, when the user navigates to the element, usually indicated by some sort of highlighting) the value of the control is set to the empty string. In effect, the message "Enter your first name" disappears so that the user can enter some text.

The *password* input control has the same use as the text input control, except that the input is obscured by dots or asterisks.

The *range* input control results in a slider. This control has both a name and an ID, each set to "my_range." The slider has a minimum value of 0, a maximum value of 100, and initial value of 50, and an increment/decrement of 5 (5, 10, 15, etc.) The text `oninput='range_display.value = my_range.value'` is a JavaScript command that specifies that the control whose ID is "range_display" receives the value of the control whose ID is "my_range." The element with the ID of "range_display" is the `output` element. This results in the value of the slider appearing to the user.

The `color` input control is self explanatory. Play with it and see what it does.

```
1        <h1>My first HTML form</h1>
2        <form method='get' action = '' name='first' id='first'>
3            <p>This is my first form.</p>
4            Hidden input: <input type='hidden' name='not\_shown' value=
5                'This does not appear to the user.' /> <br />
6            Text input: <input type='text' name='my\_name' id='my_name' value=
7                'Enter your first name' onfocus='my\_name.value = "" '/> <br />
8            Password input: <input type='password' name='my\_password' /> <br />
9            Range input: <input type='range' name='my\_range' id='my_range' value='50'
10               min='0' max='100' step='5' oninput='range\_display.value = my_range.value'>
11               <output  id='range\_display'></output> <br />
12           Color input: <input type='color' name='my-color' /> <br />
13       </form>
```

3.3 multi-valued form inputs

```
1        <h1>My second HTML form</h1>
2        <form method='get' action = '' name='second' id='second'>
3            <p>This is my second form.</p>
4        </form>
5        Radio buttons:<br />
6                Male: <input type="radio" name="sex" value="male" /> |
7                Female: <input type="radio" name="sex" value="female"  /> |
8                Other: <input type="radio" name="sex" value="other" checked="checked" />
9                <br />
10               Checkboxes:<br />
11               Pepperoni: <input type="checkbox" name="pizza" value="pepperoni" /> |
12               Anchovi: <input type="checkbox" name="pizza" value="anchovi" /> |
13               Pineapple <input type="checkbox" name="pizza" value="pineapple" /> |
14               Mushrooms: <input type="checkbox" name="pizza" value="mushrooms" /> |
15               Peppers: <input type="checkbox" name="pizza" value="peppers" />
```

```
                <br />Select car:<br />
                <select name="car">
                        <option value="">Please select 1</option>
                        <option value="ford">Ford</option>
                        <option value="dodge">Dodge</option>
                        <option value="honda">Honda</option>
                        <option value="kia">Kia</option>
                        <option value="bmw">BMW</option>
                        <option value="fiat">Fiat</option>
                </select>
                <br />Select accessories:<br />
                <select name="acc" multiple="multiple" size="3" >
                        <option value="">Please select several</option>
                        <option value="ac">Air Conditioner</option>
                        <option value="radio">Stereo Radio</option>
                        <option value="manual-trans">Manual Transmission</option>
                        <option value="leather">Leather Interior</option>
                        <option value="windows">Power Windows</option>
                        <option value="liners">Floor Windows</option>
                </select>
                <br />
        <br />
                Pets:<br />
                <input type="text" list="pets" name="pets" autocomplete="on">
                <datalist id="pets">
                  <option>Cat</option>
                  <option>Dog</option>
                  <option>Hamster</option>
                  <option>Turtle</option>
                </datalist>
                <br />
```

3.4 text areas

```
        <h1>My third HTML form</h1>
        <form method='get' action = '' name='third' id='third'>
            <p>This is my third form.</p>
            <br />
                Describe your car.<br />
                <textarea rows="6" cols="60">Describe your dream car.
                </textarea>
                <br />
        </form>
```

3.5 submit and reset inputs

```
        <h1>My fourth HTML form</h1>
        <form method='get' action = '' name='fourth' id='fourth'>
            <p>This is my fourth form.</p>
        <br />
                <input type="button" value="Click␣me"
                        onclick="window.alert('You␣clicked␣me␣---␣it␣tickled!')" />
                <br />
                <br />
                <br />
                <input type="submit" name="mysubmit" value="Submit" />
                <input type="reset" name="mysubmit" value="Reset" />
                <br />
                </form>
        </form>
```

4 Navigation, JavaScript event handlers

4.1 tabindex, accesskey

4.2 onmouseover, onmouseut

4.3 onfocus, onblur

4.4 onload, onunload

4.5 onclick, ondblclick