WEB222 - Web Programming Principles

Week 6: More on HTML

Agenda

- More HTML Elements
  - Html table,
  - Image and figure elements
  - Multimedia – audio and video
- HTML element categories
- Document type and HTML validation
- Using JavaScript in HTML page

HTML Table

- An HTML table is used for presenting tabular data in a grid-like fashion.
- A table is not (/no longer) for the purposes of laying out a web page, or the sections within a web page.
  - unless you have to do so.
- Basic table tags

Tag	Description
<table>	Specifies a table
<caption>	Specifies a table caption
<tr>	Specifies a table row
<th>	Specifies a table header
<td>	Specifies a table cell / detail

Table Structure

- Table structure
  - in a table, each piece of information is displayed in a cell (<td>).
  - The cells in a line across the page make up a row (<th> or <tr>).
  - The cells in a line down the page make up a column.
- Table header <th> vs table cell / detail <td>
  - The text in <th> elements are bold and centered by default.
  - The text in <td> elements are regular and left-aligned by default.

Table Structure

- Example

```
<table border="1">
   <caption>This is the table caption</caption>
      <tr>
         <th>Column Heading 1</th>
         <th>Column Heading 2</th>
      </tr>
      <tr>
         <td>This is row 1 cell 1</td>
         <td>This is row 1 cell 2</td>
      </tr>
      <tr>
         <td>This is row 2 cell 1</td>
         <td>This is row 2 cell 2</td>
      </tr>
  </table>
```

This is the table caption

Column Heading 1	Column Heading 2
This is row 1 cell 1	This is row 1 cell 2
This is row 2 cell 1	This is row 2 cell 2

- tables.html

Table Attributes - border

- <table border='value'>: value (integer) is the thickness of the table border in pixels.
- This attribute has been deprecated in HTML5, so use it only when necessary.
  - Use CSS instead.
- By default, a table has no borders (border="0")

<td> & <th> Attributes – rowspan

- rowspan

  <th rowspan='value'>

  <td rowspan='value'>

  - Value: non-negative integer value that indicates on how many rows does the cell extend.
  - By default value ='1’.
  - If value ='0', it extends until the end of the table section (<thead>, <tbody> or <tfoot>).

<td> & <th> Attributes – rowspan

- example

```
<table border="2">
    <caption>rowspan example</caption>
    <tr>
        <th rowspan="2" >row 1 cell 1: rowspan="2"</th>
        <td>row 1 cell 2</td>
    </tr>
    <tr>
        <td>row 2 cell 1</td>
    </tr>
</table>
```

rowspan example

row 1 cell 1: rowspan="2"	row 1 cell 2
	row 2 cell 1

<td> & <th> Attributes – colspan

- colspan

  <th colspan='value'>

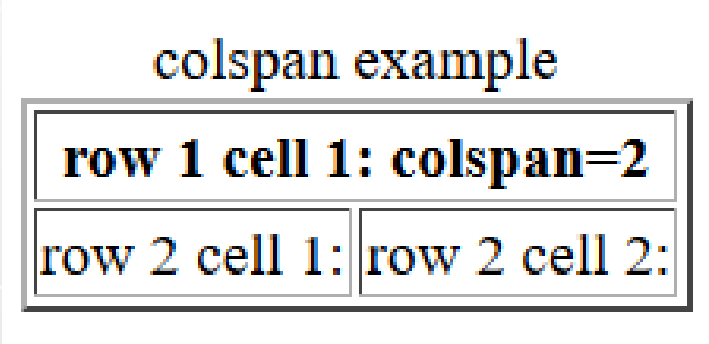
  <td colspan='value'>

  - Value: non-negative integer value that indicates on how **many rows** does the cell extend.
  - By default value ='1'.

<td> & <th> Attributes – colspan

- example

```
<table border=2>
    <caption>colspan example</caption>
    <tr>
        <th colspan=2>row 1 cell 1: colspan=2</th>
    </tr>
    <tr>
        <td>row 2 cell 1:</td>
        <td>row 2 cell 2:</td>
    </tr>
</table>
```



- tables-col-rowspan.html

Table with thead, tbody and tfoot tags

- The <thead>, <tbody> and <tfoot> elements are used to specify each part/section of a table: table header, body and table footer.
  - <thead> - table head tags - group the first one or more rows of a table for formatting
  - <tbody> - table body tags - group the middle rows of a table for formatting
  - <tfoot> - table foot tags - group the last one or more rows of a table for formatting

Table with thead, tbody and tfoot tags

- Example

Header content 1	Header content 2
Body content 1	Body content 2
Body content 3	Body content 4
Footer content 1	Footer content 2

```
<table>
    <thead style="background-color:lightblue;">
        <tr>
            <th>Header content 1</th>
            <th>Header content 2</th>
        </tr>
    </thead>
    <tfoot style="background-color:lightgreen;">
        <tr>
            <td>Footer content 1</td>
            <td>Footer content 2</td>
        </tr>
    </tfoot>
    <tbody style="background-color:lightyellow;">
        <tr>
            <td>Body content 1</td>
            <td>Body content 2</td>
        </tr>
        <tr>
            <td>Body content 3</td>
            <td>Body content 4</td>
        </tr>
    </tbody>
</table>
```

- tables-sections.html

HTML5 - <figure> and <figcaption> tags

- The HTML5 <figure> tag specifies self-contained content, frequently with a caption (<figcaption>), and is typically referenced as a single unit.
- The HTML5 <figure> tag can be used to hold <img>, <video> or <audio> elements.
- The <figcaption> can be positioned either above or below the contained element.

HTML5 - <figure> and <figcaption> tags

- Example

```
<div class="picture">
  <figure>
    <figcaption>
      This is a figure caption
    </figcaption>
    <img src="image-01.jpg" alt="landscape 1"
         title="landscape 1" />
  </figure>
</div>
```

- html5figure-1.html
- html5figure-2.html

<audio> and <video> tags

- About multimedia
  - On the web, multimedia comes in many different formats.
  - It can be almost anything you can hear or see. e.g.
    - Pictures, music, sound, videos, records, films, animations
- HTML5 introduced a built-in multimedia support via the <audio> and <video> elements, offering the standard and easy way to embed media into HTML documents.
  - Before HTML5, most audio/video files are played through a plug-in (like flash).
  - Supported by IE 9, Firefox, Opera, Chrome, and Safari

HTML5 <audio> Tags

- Example

```
<figure>
  <audio controls>
    <source src="Track03.mp3" type="audio/mpeg" />
    <source src="Track03.ogg" type="audio/ogg" />
    Your browser does not support the audio tag used.
  </audio>
  <figcaption>Audio Caption</figcaption>
</figure>
```

- Multiple <source> elements can link to different audio files.
  - The browser will use the first recognized format.

- html5_audio.html

Attributes of <audio> Element

- controls
  - Displays the standard HTML5 controls for the audio on the web page.
- src
  - It's optional. You may instead use the <source> element with src attribute.
- autoplay
- loop
- preload
- buffered, muted, played

The <source> element

- The source element is used to specify multiple media resources for audio and video elements in HTML5. It is an empty element.
- It is commonly used to serve the same media in multiple formats supported by different browsers.
- Attributes
  - src, type, media

HTML5 video Tags

- e.g.

```
<figure>
  <video width="320" height="240"  controls>
     <source src="movie.mp4"      type="video/mp4"/>
     <source src="movie.ogg"      type="video/ogg" />
     <source src="movie.webm"    type="video/webm" />
     Your browser does not support the video tag / type
  </video>
  <figcaption>Video Caption</figcaption>
</figure>
```

- The width and height specify the size of the video's display area.
- The autoplay and loop are additional attributes that can be used with the video tag.
- html5_video.html

Attributes of <video> Element

- The <video> Element shares many attributes with the <audio> element but has its own attributes:
  - Height, width
  - poster – specifies an image to show while the mediaitem is loading

About Audio/Video Formats

- Audio file formats
  - mp3 audio format (Wikipedia)
  - ogg audio/video format (Wikipedia)
- Video file formats
  - mp4 video format (Wikipedia)
  - webm audio/video format (Wikipedia)

HTML Block and Inline Elements

HTML elements/tags are classified in two different categories depend on their display features:

- Block-level elements:
  - A block-level element is a tag that creates large blocks of content. E.g.
    - <table>, (division), <hr> (horizontal rule),
    - <p>, <h1>, <ul>, <dl>, …
  - By default, a block-level element starts on a new line.
  - They can contain other block tags as well as inline tags and text.

HTML Block and Inline Elements

- Inline-level elements:
  - An inline element is a tag that defines the text or data in the document. E.g.
    - , <a>, <img>, <td>, <input>, …
  - Inline elements don't start new lines when they are used.
  - they generally only contain other inline tags, text or data.

HTML Empty element

- An empty element does not have closing tags or they are not paired.
- An empty element does not contain any text/content.
- Empty tags are simply used as markers.
- e.g.
  - <img>, <input>,
, <hr>, …
  - or older (xHTML) way: <img />, <input />,
, <hr />, …
  -
- Using a closing tag on an empty element is usually invalid. e.g. <input type="text"> ~~</input>~~.

HTML Grouping Tags

- The (division) and elements have no special meaning, but they can group HTML elements into sections.
- You group sections of an HTML page when you want to perform an action on multiple elements.

Tag	Description	
	Defines a section in a document	block-level element
	Defines a section in a document	inline element

- Note: It should be used only when no other semantic element (such as <article>, <nav>, <section>) is appropriate.

HTML Grouping Tags

- e.g.

```
<div id="menu">
  <a href="/index.htm">HOME</a> |
  <a href="/about/contact_us.htm"><span class="highlight">
     CONTACT</span></a> |
  <a href="/about/index.htm">ABOUT</a>
</div>

<div class="content" id="cnt1">
  <h5>Content</h5>
  <p>
    This is the example of <span class="highlight">span</span>
    tag and the <span class="highlight">div</span> tag along with
    <span class="highlight">CSS</span>.
  </p>
</div>
```

- tags-grouping.html

Document Type Definition (DTD)

- Document type definition (DTD) is a set of markup declarations that define a document type for an Standard Generalized Markup Language (SGML), e.g. XML, HTML.
- DTD Examples - Doctype Declarations List
  - **HTML5 document**
  - XHTML 1.0 Strict document
  - XHTML 1.0 Transitional document
  - HTML 4 Strict document
  - HTML 4 Transitional document

Validating your HTML for Correctness

- HTML validator – used to check HTML syntax errors based on Document Type.
- Throughout this course we will be using an HTML Validator officially supported by the World Wide Web Consortium (W3C) through this link.
- This links directly to the "Validate by Direct Input" tab, which allows you to copy & paste your HTML code to be checked by the validation service
- The rule of thumb for the course is that Warnings are fine, but Errors are not.
- You should get in the habit of always checking your HTML code and correcting errors, as invalid HTML incurs large penalties when marking assignments.

Using JavaScript in an HTML Page

- Now that we know how to create a simple HTML page, why don't we use it to enhance our JavaScript output and finally move out of the console!
- To begin, we will discuss 3 ways of including JavaScript in a webpage:

  - **Inline** JavaScript code: Basic event handlers.
  - **Internal** JavaScript code: Using script tags.
  - **External** JavaScript code: Using code stored in a separate .js file.

Inline (embedded) JavaScript code

- **Inline** JavaScript code: Scripts that handle events are referred to, appropriately, as event handlers.

- e.g.

  <input type="button" id="hello" value="Hello" onClick = “myFunction()“>

Internal JavaScript code

- Internal JavaScript code: Using <script> tags.
  - NOTE: the type attribute is optional because "text/javascript" is its default value.
- Scripts can be inserted anywhere on a page, e.g.

```
<!DOCTYPE html>
<html>
  <head>
    <title>WEB222</title>
    <script>
      // include your JavaScript code here
    </script>
  </head>
  <body>
    <script type='text/javascript'>
      // you can also include it here!
    </script>
  </body>
</html>
```

External JavaScript code

- External JavaScript code: Using <script> tags with a "src" attribute.
- The "src" attribute will contain a path (absolute or relative) to a separate JavaScript (.js) file containing only JavaScript code
- This is the preferred way to include your JavaScript.
- The <script></script> tags are typically included either in the <head>…</head> section or at the end of the <body>...</body> section (for large JS libraries).

External JavaScript code

- Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>WEB222</title>
      <script src="js/myFile.js"></script>
   </head>
   <body>
      <script type='text/javascript' src="js/myOtherJSFile.js"></script>
   </body>
</html>
```

Updating Page Text from JavaScript

- In order for us to be able to actually update some of the text in our HTML pages we need to be able to programmatically reference the elements on the page.
- This is done via the **inner.HTML** of the Document Object Model (DOM) :
  - The Document Object Model (DOM) is a 2-way application programming interface (API) for HTML (and XML) documents.
  - Essentially, you can think of it as a way for JavaScript to access / manipulate the complex "document" object hierarchy that is created when a browser loads an HTML page (document).
  - With the DOM, we have read/write access to any element on the page!

Creating a Container in HTML

- To able to write to the HTML page using the inner.HTML property of the DOM, we need to have HTML code that helps to identify our target element
- for example, we can use the "**id**" attribute to uniquely identify any element in the way of DOM object (Duplicate id values are NOT allowed in HTML documents).
- Here, we set the id value of our h3 element to be "outputContainer"

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example 1</title>
    <script src="js/myFile.js"></script>
  </head>
  <body>
    <h3 id="outputContainer"></h3>
  </body>
</html>
```

Writing to the Container

- In Visual Studio Code, create a new Folder in your Example1 directory (call it "js")
- Inside the "js" folder, create a file called myFile.js
- In our myFile.js we need to do a couple of things to ensure that we correctly write to our "outputContainer".
- First: We need to make sure that the page is completely loaded before we even attempt to update the DOM
- This is done by making use of the "onload" property of the global "window" object.
- We can assign this to be a function that will be executed once the page has completed loading:

```
window.onload = function() {
        // start accessing the DOM here
};
```

Writing to the Container (cont'd)

- Second: We need to get a reference to the element
- This is done by making use of the "querySelector" function on the global "document" object
- Since the element we want is identified by an ID, we use the following syntax to gain a reference to the element, ie:
- document.querySelector(#someID);

```
window.onload = function() {
  // start accessing the DOM here
  var myContainer = document.querySelector("#outputContainer");
};
```

Writing to the Container (cont'd)

- Third: We need to call a function to actually update the text
- Fortunately, we have access to a very useful element property – "innerHTML".
- Be careful! You can very easily insert invalid code into the page this way and it can be difficult to debug, so only use the "innerHTML" property for updating text or simple HTML

```
window.onload = function() {
    // start accessing the DOM here
    var myContainer = document.querySelector("#outputContainer");

    // update it's "innerHTML" property
    myContainer.innerHTML = "Hello World!"
};
```

Writing to the Container (Full Example)

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example 1</title>
    <script src="js/myFile.js"></script>
  </head>

  <body>
    <h1 id="title"></h1>
    <div id="outputContainer"></div>
  </body>
</html>
```

JavaScript (myFile.js)

```
var listTitle = "Students List (Alphabetical)";
var studentArray = ["John", "Bob", "Amy", "Haley", "Kimberly"];

window.onload = function() {

  var myTitle = document.querySelector("#title");
  myTitle.innerHTML = "<em>" + listTitle + "</em>";

  var myContainer = document.querySelector("#outputContainer");

  studentArray.sort();

  for (var i = 0; i < studentArray.length; i++) {
    myContainer.innerHTML += "<p>" + studentArray[i] + "</p>";
  }
};
```

- student-list.html

More examples of updating DOM using **inner.HTML**

- Using inner.HTML to populate tables

  populate-table.html

- Using inner.HTML to add images

  add-image.html

Resourceful Links

- MDN - HTML element reference
- MDN - Articles tagged: Multimedia

Thank you!

Any Questions?