CS7026: Authoring for Digital Media

HTML5 Global Attributes
Global Attributes

- There are several new global attributes, which can be added to any element.

**contenteditable**

- Invented by Microsoft, and reverse-engineered and implemented by all other browsers, *contenteditable* is now officially part of HTML5.

- It means two things for browsers:

  1. that users can edit the contents of elements with this attribute, so the element must be selectable and the browser must provide a caret to mark the current editing position;
  2. that changes made to the document affect the selected content specifically selected and editable, that is you can make the text bold, change the font, add lists, headings, and so on.
Global Attributes

- It is a Boolean attribute, so it can be set to true or false.

- Although markup capitalisation is irrelevant, the DOM attribute requires `contentEditable` (note the capital E).

- The DOM also has `isContentEditable` to assess whether this element is editable.
Global Attributes

- You can also set `document.designMode = 'on'` (notice, not 'true') to enable the entire document to be editable. This can only be done using JavaScript.

- Any selected (that is, highlighted) content by the user can have a number of commands run against it, such as `document.execCommand('bold')`.

- Typical keyboard commands to make text bold or italic affect the DOM in the editable element.
Global Attributes

**draggable**
- draggable indicates that the element can be dragged using the drag-and-drop API (more later...).

**hidden**
- This tells the browser that the content of this element shouldn’t be rendered in any way.
- It doesn’t only hide the content, but keeps it “in the wings,” so that for instance you could use JavaScript later on to remove the attribute and cause the element to “pop” into being.
CS7026 – HTML5

Forms
HTML Forms and Input

- HTML forms are used to collect data and act on it and/or pass it to a server.

- A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more.

- The basic construction of a HTML form is:
  ```html
  <form> begin a form
  <input /> ask for information in one of several different ways...
  <input /> ...there can be as many input areas as you wish
  </form> end a form
  ```
The most important form element is the *input* element.

The input element is used to select user information.

An input element can vary in many ways, depending on the *type* attribute.

Up until HTML5 an input element could be of type *text field, checkbox, password, radio button, submit button,* and more. The following are the most commonly used...
Text Fields

- `<input type="text" />` defines a one-line input field that a user can enter text into:
  ```html
  <form>
  First name: <input type="text" name="firstname" />
  </form>
  ```

- Every input requires a **name**. When the user types in her name (eg. “Mary”), it will become the input's **value** and be paired with **firstname** so the end result after processing will be:

  ```
  firstname = Mary
  ```

- **Note:** The form itself is not visible. Also note that the default width of a text field is 20 characters.
Text Fields

- We can if we want, type in a value.

  ```html
  <form>
  <input type="text" name="address" value="44 Cherry St" />
  </form>
  ```

- This will automatically pair the `value` 44 Cherry St with the `name` address, unless the user changes it.
We can specify the size of the text input box.

```html
<form>
  <input type="text" name="address" value="44 Cherry St" size="10" />
</form>
```
Text Fields

- If we want, we can specify how many characters a user can input.

```html
<form>
  <input type="text" name="address" value="44 Cherry St" size="30" maxlength="10" />
</form>
```
Password Field

- `<input type="password" />` defines a password field:

```
<form>
  Password: <input type="password" name="pwd" />
</form>
```

- **Note:** The characters in a password field are masked (shown as asterisks or circles).
Radio Buttons

- `<input type="radio" />` defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices:

```html
<form>
  <input type="radio" name="gender" value="male" /> Male<br />
  <input type="radio" name="gender" value="female" /> Female
</form>
```
Checkboxes

- `<input type="checkbox" />` defines a checkbox. Checkboxes let a user select ONE or MORE options of a limited number of choices.

```html
<form>
<input type="checkbox" name="vehicle" value="Bike" /> I have a bike<br />
<input type="checkbox" name="vehicle" value="Car" /> I have a car
</form>
```
Submit and Reset Buttons

```html
<form>
  <input type="submit" />
</form>

- submit of course, sends the data...
- ...and reset, clears the form.

```html
<form>
  <input type="reset" />
</form>
```
Submit and Reset Buttons

- We can easily change what the buttons say.

```html
<form>
  <input type="submit" value="Send it away" />
  <input type="reset" value="Clear the form" />
</form>
```
We must tell the browser what to do with the data we gather. There are three options:

- You can have the data emailed directly to you using the `mailto` action. This is simple but not very reliable, OR
- Use a scripting language to perform all form processing on the client side (as opposed to sending the form data to the server for processing), OR
- Send the data to a server-side script for processing. (preferred method).
Processing Forms – The mailto Action

<html>
<head>
<title>My First Form</title>
</head>
<body>
<form method="post" action="mailto:abc@123.com">
</form>
</body>
</html>
Processing Forms – The mailto Action

- The only thing you have to do is put in your email address after mailto:

- To send the data to more than one email address, separate them with commas...

  mailto:larry@3stooges.org,curly@3stooges.org,mo@3stooges.org
Processing Forms – The mailto Action

- This method is simple, but not entirely reliable - good for practicing.

- Every so often, depending on the sender's browser/email configuration, using the mailto action might just bring up an empty New Mail window instead of sending the form data.

- Sometimes (rarely) you'll click on the submit button and nothing will happen. In this case you will not be able to even practice.

- There are also the obvious spam risks.
Processing Forms – Javascript

Using a scripting language to perform form processing

- E.g., you might have a small form at the bottom of a page that contains only two radio buttons labeled Yes and No, plus a Submit button. The form action might be a JavaScript function defined in the head section of the document that displays one alert if the user selects Yes and another alert if the user selects No:
function processForm() {
    if (document.forms[0].elements[0].checked) {
        alert('Yes');
    } else {
        alert('No');
    }
}
Processing Forms – Javascript

- To use a client-side JavaScript function as the form action:

  - Add a `processForm()` JavaScript function to the `<head>` element of your document.

  - Call the function from your submit button:
    ```html
    <input type="submit" name="Submit" value="Submit" onclick="processForm()" />
    ```
You can handle many form-processing tasks using client-side scripting, but you can't save the data entered by the user or send it to someone else.

For such purposes you need a server-side application such as a Common Gateway Interface (CGI) script.
Processing Forms – Server Side

- You define the processing file in the `action` attribute and this usually does something with the received input:

```html
<form name="input" action="html_form_action.asp" method="get">
  Username: <input type="text" name="user" />
  <input type="submit" value="Submit" />
</form>
```

- If you type some characters in the text field, and click the "Submit" button, the browser will send your input to a file called "html_form_action.asp".
HTML5

Lots of new stuff for forms...
HTML5 New Input Types

- HTML5 has several new input types for forms. These new features allow for better input control and validation.
  - email
  - url
  - number
  - range
  - date pickers (date, month, week, time, datetime, datetime-local)
  - search
  - color
Browser Support

- Opera and Chrome have the best support for the new input types. However, you can already start using them in all major browsers. If they are not supported, they will behave as regular text fields.

- Try caniuse.com for compatibility tables.
Input Type - email

- The email type is used for input fields that should contain an e-mail address.

  E-mail: <input type="email" name="user_email" />

- Safari on the iPhone recognizes the email input type, and changes the on-screen keyboard to match it (adds @ and . options).
Input Type - url

- The url type is used for input fields that should contain a URL address.

    Homepage: <input type="url" name="user_url" />

- Safari on the iPhone recognizes the url input type, and changes the on-screen keyboard to match it (adds ., / and .com options).
Input Type - number

- The number type is used for input fields that should contain a numeric value.

- Use the following attributes to specify restrictions for the number type:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>number</td>
<td>Specifies the maximum value allowed</td>
</tr>
<tr>
<td>min</td>
<td>number</td>
<td>Specifies the minimum value allowed</td>
</tr>
<tr>
<td>step</td>
<td>number</td>
<td>Specifies legal number intervals (if step=&quot;3&quot;, legal numbers could be -3,0,3,6, etc)</td>
</tr>
<tr>
<td>value</td>
<td>number</td>
<td>Specifies the default value</td>
</tr>
</tbody>
</table>
Input Type - number

- An example with all the restriction attributes:

```html
<form action="demo_form.asp" method="get">
  <input type="number" name="points" min="0" max="10" step="3" value="6" />
  <input type="submit" />
</form>
```

- Safari on the iPhone recognizes the number input type, and changes the on-screen keyboard to match it (shows numbers).
Input Type - number

- In the desktop versions of Opera and Chrome, the type="number" field is rendered as a “spinbox” control, with little up and down arrows that you can click to change the value:

  ![Spinbox example](image)

- They respects the min, max, and step attributes, so you’ll always end up with an acceptable numeric value.

  ![Spinbox example](image)
The range type is used for input fields that should contain a value from a range of numbers.

The range type is displayed as a slider bar (can look a bit different depending on the browser).

You can also set restrictions on what numbers are accepted:

```html
<input type="range" name="points" min="1" max="10" />
```
**Input Type - range**

- Use the following attributes to specify restrictions for the range type:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>number</td>
<td>Specifies the maximum value allowed</td>
</tr>
<tr>
<td>min</td>
<td>number</td>
<td>Specifies the minimum value allowed</td>
</tr>
<tr>
<td>step</td>
<td>number</td>
<td>Specifies legal number intervals (if step=&quot;3&quot;, legal numbers could be -3,0,3,6, etc)</td>
</tr>
<tr>
<td>value</td>
<td>number</td>
<td>Specifies the default value</td>
</tr>
</tbody>
</table>
Input Type - Date Pickers

- HTML5 has several new input types for selecting date and time:
  - date - Selects date, month and year
  - month - Selects month and year
  - week - Selects week and year
  - time - Selects time (hour and minute)
  - datetime - Selects time, date, month and year (UTC time)
  - datetime-local - Selects time, date, month and year (local time)
Input Type - Date Pickers

- The following example allows you to select a date from a calendar:

```html
Date: <input type="date" name="user_date" />
```
HTML5 Form Elements - **datalist Element**

- HTML5 has several new elements and attributes for forms.

- The **datalist** element specifies a list of options for an input field.

- The list is created with option elements inside the **datalist**.

- The option elements should always have a value attribute.
To bind a datalist to an input field, let the `list` attribute of the input field refer to the `id` of the datalist:

```html
Webpage: <input type="url" list="url_list" name="link" />
<datalist id="url_list">
<option label="TCD" value="http://www.tcd.ie" />
<option label="Google" value="http://www.google.com" />
<option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```

The option elements should always have a value attribute.
output Element

- The output element is used for different types of output, like calculations or script output.

- I.e, it is the semantically correct element for displaying the results of a calculation from form elements. (works in Opera).

```html
<form onsubmit="return false">
  <input name=a type=number> +
  <input name=b type=number> =
  <output onforminput="value = a.valueAsNumber + b.valueAsNumber">
  </output>
</form>
```
HTML5 New Form Attributes

- New **form** attributes:
  - autocomplete
  - novalidate

- New **input** attributes:
  - autocomplete
  - autofocus
  - form
  - form overrides (formaction, formenctype, formmethod, formnovalidate, formtarget)
  - height and width
  - list
  - min, max and step
  - multiple
  - pattern (regexp)
  - placeholder
  - required
autocomplete Attribute

- Most browsers have some kind of autocomplete functionality. When the user starts to type in an autocomplete field, the browser should display options to fill in the field:

- The autocomplete attribute lets you control how this works.

- The default state is for the input to inherit the autocomplete state of its form owner. Forms have autocomplete on by default.

- If the autocomplete attribute of a form element is set to on, the field is fine for autocompletion.
**autocomplete Attribute**

- To quote the specification’s description of the off state: “The off state indicates either that the control’s input data is particularly sensitive (for example, the activation code for a nuclear weapon); or that it is a value that will never be reused (for example, a one-time-key for a bank login) and the user will therefore have to explicitly enter the data each time.”

- **Note:** The autocomplete attribute works with `<form>`, and the following `<input>` types: text, search, url, telephone, email, password, datepickers, range, and color.
autocomplete Attribute

<form autocomplete="on">
First name: <input type="text" name="fname" />
<br />
Last name: <input type="text" name="lname" />
<br />
E-mail: <input type="email" name="email" autocomplete="off" /><br />
<input type="submit" />
</form>
autofocus Attribute

- The autofocus attribute specifies that a field should automatically get focus when a page is loaded.

- The autofocus attribute works with all <input> types.

User name: <input type="text" name="user_name" autofocus="autofocus" />
**form Attribute**

- The form attribute specifies one or more forms the input field belongs to.

- The form attribute works with all `<input>` types.

- The form attribute must refer to the id of the form it belongs to.
form Attribute

```html
<form action="demo_form.asp" method="get"
id="user_form">
First name:<input type="text" name="fname" />
<input type="submit" />
</form>
Last name: <input type="text" name="lname"
form="user_form" />
```

- To refer to more than one form, use a space-separated list.
The height and width attributes specify the height and width of the image used for the input type image (which defines an image as the submit button).

Note: The height and width attributes only works with `<input>` type: image.

```html
<input type="image" src="img_submit.gif" width="24" height="24" />
```
list Attribute

- The list attribute specifies a datalist for an input field. A datalist is a list of options for an input field.

- The list attribute works with the following <input> types: text, search, url, telephone, email, date pickers, number, range, and color.

  Webpage: `<input type="url" list="url_list" name="link" />`

  `<datalist id="url_list">`

  `<option label="The Irish Times" value="http://www.irishtimes.com" />`

  `<option label="Google" value="http://www.google.com" />`

  `<option label="Microsoft" value="http://www.microsoft.com" />`

  `</datalist>`
The min, max and step attributes are used to specify restrictions for input types containing numbers or dates.

- The max attribute specifies the maximum value allowed for the input field.
- The min attribute specifies the minimum value allowed for the input field.
- The step attribute specifies the legal number intervals for the input field (if step="3", legal numbers could be -3,0,3,6, etc).
**min, max and step Attributes**

- **Note:** The min, max, and step attributes works with the following `<input>` types: date pickers, number, and range.

- This example shows a numeric field that accepts values between 0 and 10, with a step of 3 (legal numbers: 0, 3, 6, 9):

```html
Points: <input type="number" name="points" min="0" max="10" step="3" />
```
**multiple Attribute**

- The multiple attribute specifies that multiple values can be selected for an input field.

- The multiple attribute works with the following `<input>` types: email, and file.

Select images: `<input type="file" name="img" multiple="multiple" />`
pattern Attribute

- The pattern attribute specifies a pattern used to validate an input field.

- The pattern is a javascript regular expression.

**Note:** The pattern attribute works with the following `<input>` types: text, search, url, telephone, email, and password.
pattern Attribute

- The example below shows a text field that can only contain three letters (no numbers or special characters):

```html
Country code: <input type="text" name="country_code" pattern="^[A-z]{3}$" title="Three letter country code" />
```
**placeholder Attribute**

- The placeholder attribute provides a hint that describes the expected value of an input field.

- The placeholder attribute works with the following `<input>` types: text, search, url, telephone, email, and password.

- The hint is displayed in the input field when it is empty, and disappears when the field gets focus:

  `<input type="search" name="user_search" placeholder="Search TCD website" />`
required Attribute

- The required attribute specifies that an input field must be filled out before submitting.

- The required attribute works with the following `<input>` types: text, search, url, telephone, email, password, date pickers, number, checkbox, radio, and file.

Name: <input type="text" name="usr_name" required="required" />


Lab Exercise

- Design and code the front end of an airline booking form for a top-end, first-class airline company.

- Think about what information the airline needs to process the booking.

- What about extra customer requirements?