Web Accessibility

Will Gregory

Course Outline (1 of 2)

- Accessibility introduction and benefits
- Legislation and guidelines
- Conformance levels – Level A, AA and AAA
- Screen reader and assistive technologies
- Problems with accessibility on sites
- Test and evaluating your website for accessibility
- Designing an accessible web site
- Page and HTML structure and navigation

Course Outline (2 of 2)

- Working with colour, images and multimedia
- Accessible tables and the header element
- Accessible links, tab structures and access keys
- Working with effective lists
- Creating accessible forms
- Accessible video
- Validating your website
- Working with conformance levels and logos

Why Web Accessibility? (1 of 2)

- When websites and web tools are properly designed, people with disabilities can use them
- Many sites contain accessibility barriers
- Making the web accessible benefits individuals, businesses, and society
- International web standards define what is needed for accessibility

Why Web Accessibility? (2 of 2)

> “The power of the Web is in its universality.
> Access by everyone regardless of disability is an essential aspect.”

Tim Berners-Lee, W3C Director and inventor of the World Wide Web

Disabilities to Consider

- Web accessibility encompasses all disabilities that affect access to the Web, including:
  - auditory
  - cognitive
  - neurological
  - physical
  - speech
  - visual

Other Situations to Consider

- Web accessibility also benefits people without disabilities, for example:
    - User’s device eg mobile phones, smart watches
    - Temporary disabilities eg broken arm or lost glasses
    - Environmental situation eg bright light
    - Bandwidth limitations eg train or countryside

Business Benefits

- Accessibility can enhance your brand, drive innovation and extend your market reach
- Accessibility can enhance SEO
  - improve ranking via alt text etc
  - increase traffic/sales
  - decrease bounce rate

Web Accessibility - Summary

- Good design is accessible to people with a diverse range of **hearing**, **movement**, **sight**, and **cognitive** ability
- Bad design creates barriers

Legal Requirements - UK Public Sector Bodies

- Your service must be accessible to everyone who needs it
- If it isn’t, you may be breaking the law
- Recent study found that 4 in 10 local council homepages failed basic tests for accessibility

Meeting Government Accessibility Requirements

- minimum level AA of the Web Content Accessibility Guidelines (WCAG 2.1)
- compatible with common assistive technologies - including screen magnifiers, screen readers and speech recognition tools
- include people with disabilities in user research
- have an accompanying accessibility page that explains how accessible the service is

Meeting Government Accessibility Regulations

“If your service meets government accessibility requirements, then you’ll also be meeting the new accessibility regulations that apply to public sector websites and apps.”

gov.uk, Service Manual

[*Click this link to visit the site*]

Current Government Accessibility Regulations

- The full name of current regulations is the **Public Sector Bodies (Websites and Mobile Applications) (No. 2) Accessibility Regulations 2018**

Deadlines for New Websites

- If you create a new public sector website on or after 23 September 2018, you need to meet accessibility standards and publish an accessibility statement by **23 September 2019**

Deadlines for Existing Websites

- Most existing websites that were published before 23 September 2018 need to comply with the 2018 regulations by **23 September 2020**

Exceptions for Existing Websites

- May not have to meet the requirements for your **whole** website or app if doing so would be a disproportionate burden
  - very expensive to make even simple changes
  - changes would bring very limited benefits to people with a disability
- Talk to legal adviser

Web Content Accessibility Guidelines (WCAG)

- Provides a standard for web content accessibility that meets the needs of individuals, organizations, and governments internationally
- Web “content” generally refers to the information in a web page or web application, including:
  - natural information such as text, images, and sounds
  - code or markup that defines structure, presentation, etc

WCAG 2.1

- Latest version, WCAG 2.1 was published on 5 June 2018
- Backward compatible with WCAG 2.0

Who is WCAG 2.1 For?

- Web content authors
- Web application developers
- Web accessibility evaluation consultants

WCAG 2.1 - Conformance

- WCAG 2.1 Success Criteria are written as testable criteria
- Combination of automated testing and human evaluation
- Usability testing is recommended
- It is recommended that users with disabilities be included in test groups

WCAG 2.1 - Conformance Levels

- Most standards only have one level of conformance
- WCAG 2.1 has three levels of conformance, and therefore, three levels of Success Criteria
- Level A, Level AA, Level AAA
- Quick Reference (https://www.w3.org/WAI/WCAG21/quickref/)

What is in the Guidelines?

- The guidelines are separated into 4 principles:
    - Perceivable
    - Operable
    - Understandable
    - Robust

Perceivable - Text alternatives for non-text content

- Short equivalents for images, including icons, buttons, and graphics
- Description of data represented on charts, diagrams, and illustrations
- Brief descriptions of non-text content such as audio and video files
- Labels for form controls, input, and other user interface components

Perceivable - Captions and other alternatives for multimedia

- Text transcripts and captions for audio content, such as recordings of a radio interview
- Audio descriptions, which are narrations to describe important visual details in a video
- Sign language interpretation of audio content, including relevant auditory experiences

Perceivable - Adaptability of Presentation

- Headings, lists, tables, input fields, and content structures are marked-up properly
- Sequences of information or instructions are independent of any presentation
- Browsers and assistive technologies provide settings to customize the presentation

Perceivable - Content is easier to see and hear

- Default foreground and background color combinations provide sufficient contrast
- No information is lost when zooming
- Images of text are resizable, replaced with actual text, or avoided where possible
- Users can pause, stop, or adjust the volume of audio that is played on a website
- Background audio is low or can be turned off, to avoid interference or distraction

Operable - Functionality is available from a keyboard

- Many people do not use the mouse
- Require keyboard access to:-
    - form controls
    - other user interface components
- All functionality that is available by mouse is also available by keyboard
- Keyboard focus does not get trapped in any part of the content

Operable - Timing

- Users must have enough time to read and use the content
- Pause, stop, or hide moving, blinking, or scrolling content
- Postpone or suppress interruptions, except where necessary
- Re-authenticate when a session expires without losing data

Operable - Content does not cause seizures and physical reactions

- Do not include content that flashes at particular rates and patterns
- Warn users before flashing content is presented, and provide alternatives
- Provide mechanisms to switch off animations, unless they are essential

Operable - Users can easily navigate

- Pages have clear titles and are organized using descriptive section headings
- There is more than one way to find relevant pages within a set of web pages
- Users are informed about their current location within a set of related pages
- There are ways to bypass blocks of content that are repeated on multiple pages
- The keyboard focus is visible, and the focus order follows a meaningful sequence
- The purpose of a link is evident, ideally even when the link is viewed on its own

Operable - Users can use different input modalities beyond keyboard

- Gestures that require dexterity or fine movement have alternatives that do not require high dexterity
- Components are designed to avoid accidental activation, for example by providing undo functionality
- Buttons, links, and other active components are large enough to make them easier to activate by touch

Understandable - Text is readable and understandable

- Identifying the primary language of a web page, such as Arabic, Dutch, or Korean
- Identifying the language of text passages, phrases, or other parts of a web page
- Providing definitions for any unusual words, phrases, idioms, and abbreviations
- Using the clearest and simplest language possible, or providing simplified versions

Understandable - Content appears and operates in predictable ways

- Navigation mechanisms that are repeated on multiple pages appear in the same place each time
- User interface components that are repeated on web pages have the same labels each time
- Significant changes on a web page do not happen without the consent of the user

Understandable - Users are helped to avoid and correct mistakes

- Descriptive instructions, error messages, and suggestions for correction
- Context-sensitive help for more complex functionality and interaction
- Opportunity to review, correct, or reverse submissions if necessary

Robust - Content is compatible with current and future user tools

- Ensuring markup can be reliably interpreted, for instance by ensuring it is valid
- Providing a name, role, and value for non-standard user interface components

Well Structured HTML

- Well-structured content allows more efficient navigation and processing
- Use HTML and WAI-ARIA to improve navigation and orientation on web pages

Page Regions

- Page header - <header>
- Page footer - <footer>
- Navigation - <nav>
- Main content - <main>
- Complementary content - <aside>

Workshop 1

- Start to create a well structured HTML document
- Define the top level regions of a web page

Test Early Test Often

- Test your web pages as you go
- Don’t treat accessibility as an afterthought
- Recommended to consider accessibility from the start

Tools For Testing

- Online testing - free/paid
  - webaccessibility.com
  - PowerMapper
- Browser extensions - free
  - ARC Toolkit
  - axe - Web Accessibility Testing
  - WAVE Evaluation Tool

Screen Readers

- Windows
    - JAWS - expensive
    - NVDA - free
- Apple
    - VoiceOver - built in
- Google Chrome
    - ChromeVox Classic Extension - free

Workshop 2

- Install tools for testing

Page Structure Concepts

- Page Regions: Identify and mark up regions on web pages using HTML5 and WAI-ARIA roles
- Labeling Regions: Label regions to allow users to distinguish and access them
- Headings: Add headings and nest them logically to label sections of web pages according to their relationships and importance
- Content Structure: Mark up the content on a page in a way that uses appropriate and meaningful elements

WAI-ARIA

- Accessible Rich Internet Applications
- Allows greater control over context and descriptions for screen readers and other assistive technologies
- WAI-ARIA addresses these accessibility challenges

ARIA Landmark Roles

- role=””
  - banner
  - complementary
  - contentinfo
  - form
  - main
  - navigation
  - region
  - search

ARIA Widget Roles

- role=””
    - button
    - checkbox
    - link
    - menuitem
    - option
    - progressbar
    - radio
    - searchbox
    - tab

ARIA Attributes

- aria-label="screen reader friendly label"
- aria-labelledby="id of another element"
- aria-describedby="id of another element"
- aria-current="true"

Headings

- <h1> to <h6>
- Correct use of heading structure essential within page sections

Content Structure

- Articles - <article>
- Sections - <section>
- Paragraphs - <p>
- Lists
    - Unordered lists - <ul>
    - Ordered lists - <ol>
    - Description lists - <dl>
    - Nested lists - <ul><li><ul>
- Quotes - <blockquote>
- Figures - <figure>
- Tables - <table>

Workshop 3

- Add headings and articles to the web page

Menu Concepts

- Structure: Mark up menus in a way that reflects their structure and appropriately labels them
- Styling: Use commonly recognized design patterns to distinguish menus and the state of menu items
- Fly-out Menus: Ensure fly-out (drop-down) submenus can be used appropriately by mouse and keyboard

Menu Structure

- Convey the menu structure using a list - <ul><li><a>
- Identify menus - <nav role=”navigation”>
- Label menus - <nav aria-labelledby=””>
- Indicate current item - <a href="#main" aria-current="page">

Menu Styling

- Consistent location on each page
- Easy identification of each menu on the page
- Easily readable
- Size menus with sufficient padding

Menu Link Items Styling

- Default state - nav a { … }
- Hover and focus states - nav a:hover, nav a:focus { … }
- Active state - nav a:active { … }
- Current state - nav [aria-current=page] { … }

Fly-out Menus

- Indicate submenus - <a aria-haspopup="true" aria-expanded="false">
- Provide other ways to the submenu items
- Functionality is created using CSS and scripting

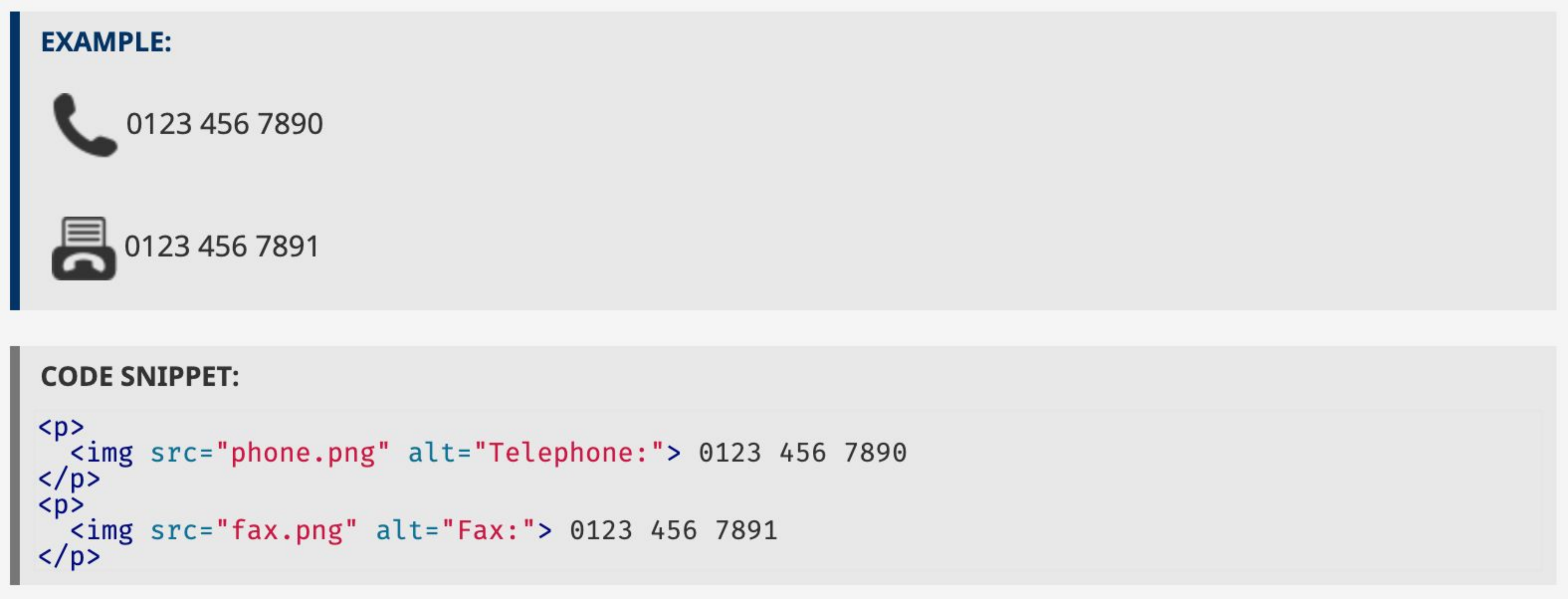
Workshop 4

- Add navigation menu to the web page banner

Image Concepts

- Informative images: Images that graphically represent concepts and information. The text alternative should convey essential information
- Decorative images: Provide a null text alternative (alt="")
- Functional images: The text alternative of an image used as a link or as a button should describe the functionality of the link or button
- Images of text: Avoid text in images or include words in alt attribute
- Complex images such as graphs and diagrams: Provide a full-text equivalent of the data or information

Informative: To label other information



EXAMPLE:

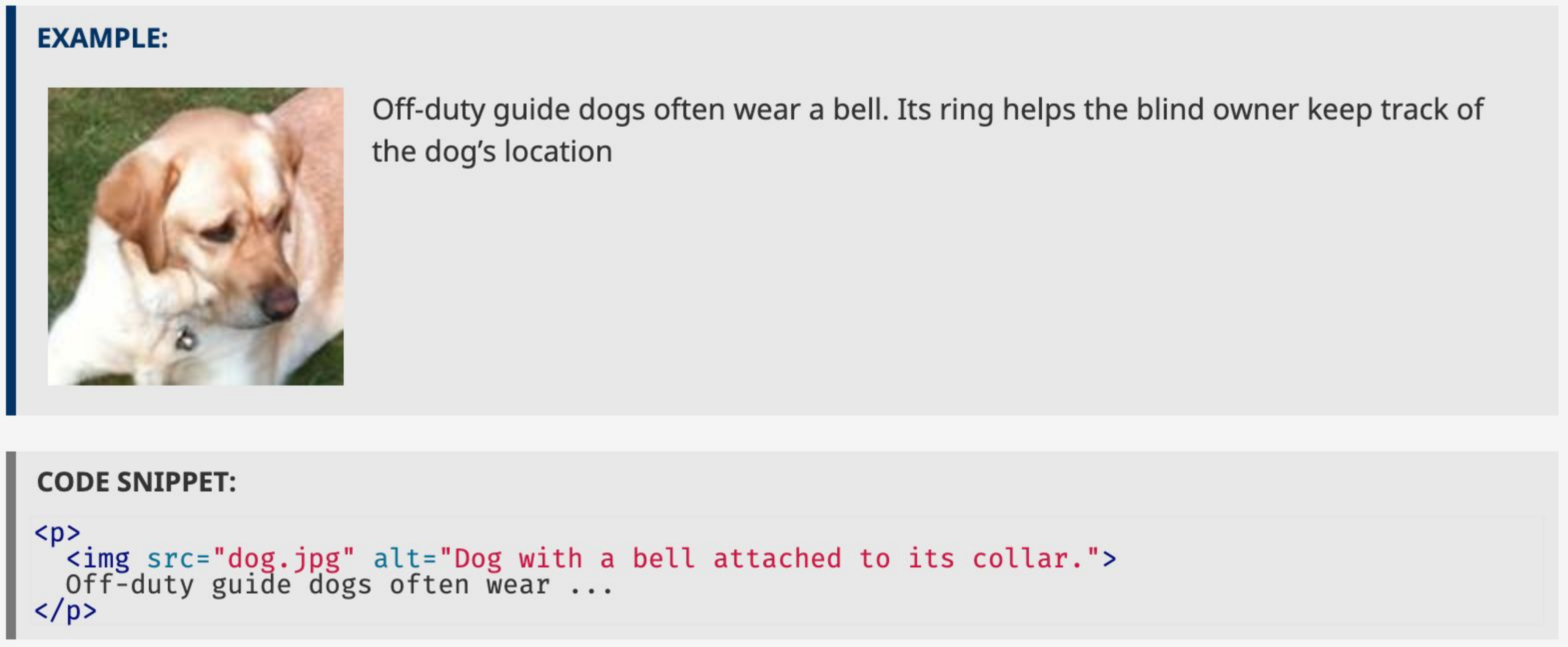
☎ 0123 456 7890

📠 0123 456 7891

CODE SNIPPET:

```
<p>
  <img src="phone.png" alt="Telephone:"> 0123 456 7890
</p>
<p>
  <img src="fax.png" alt="Fax:"> 0123 456 7891
</p>
```

Informative: To supplement other information



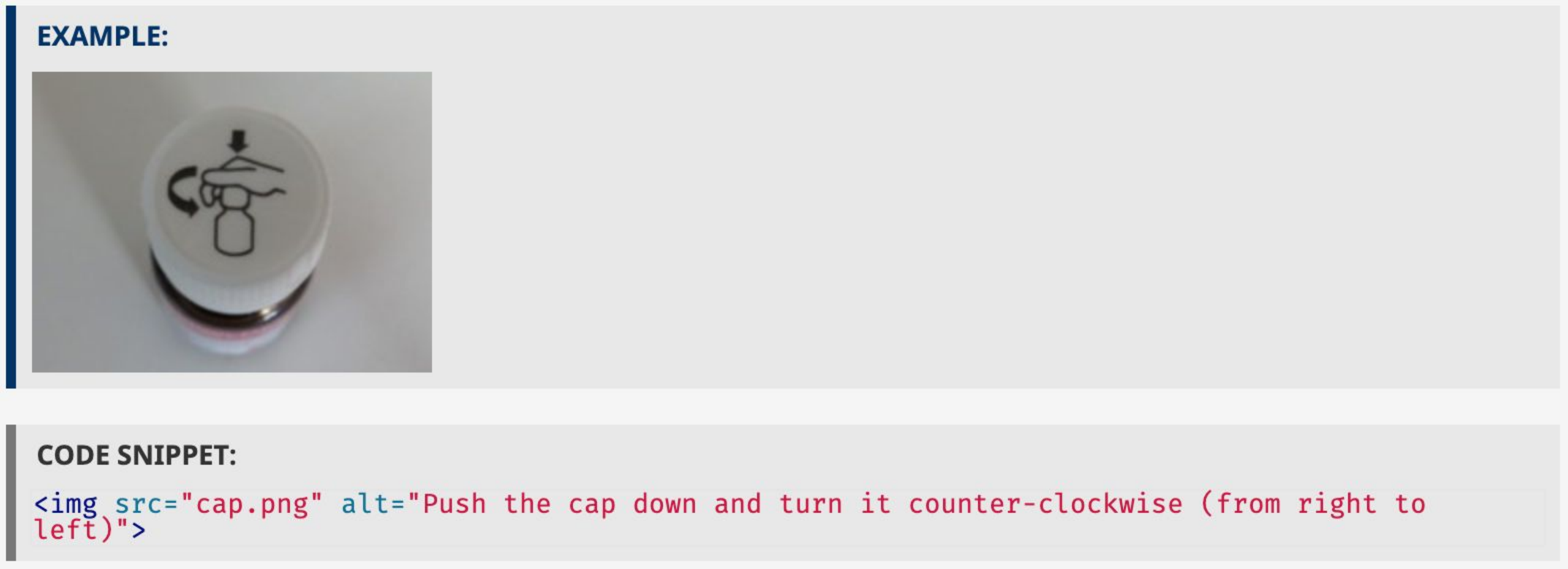
EXAMPLE:

Off-duty guide dogs often wear a bell. Its ring helps the blind owner keep track of the dog’s location

CODE SNIPPET:

```
<p>
  <img src="dog.jpg" alt="Dog with a bell attached to its collar.">
  Off-duty guide dogs often wear ...
</p>
```

Informative: Convey succinct information

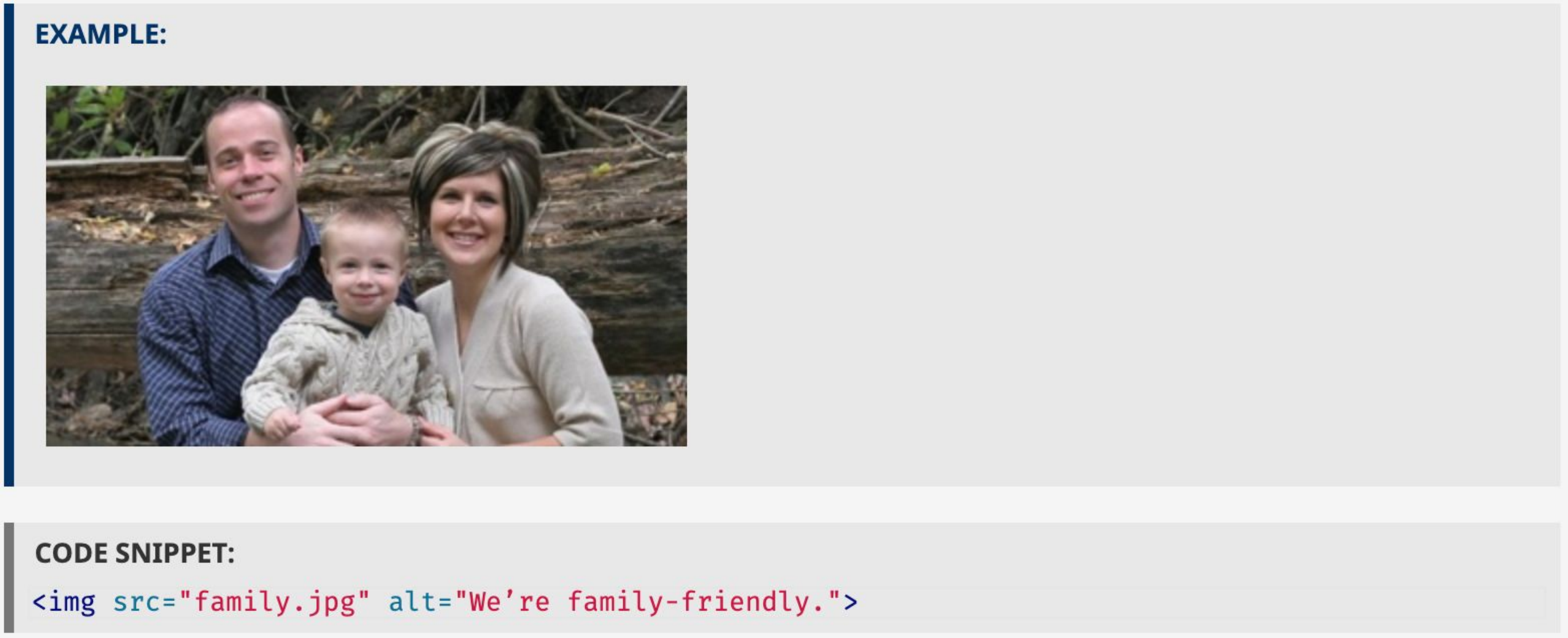


EXAMPLE:

CODE SNIPPET:

```
<img src="cap.png" alt="Push the cap down and turn it counter-clockwise (from right to
left)">
```

Informative: Convey an impression or emotion

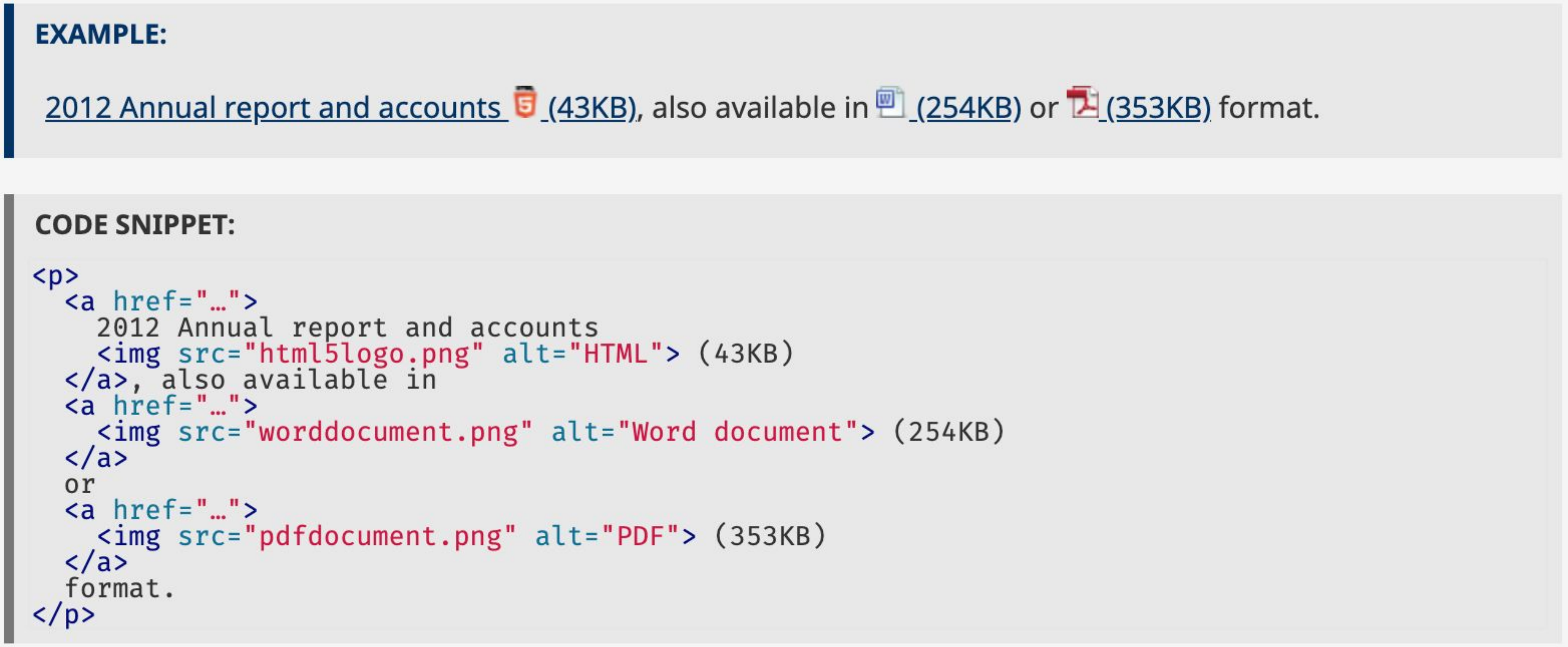


EXAMPLE:

CODE SNIPPET:

```
<img src="family.jpg" alt="We’re family-friendly.">
```

Informative: Convey file format



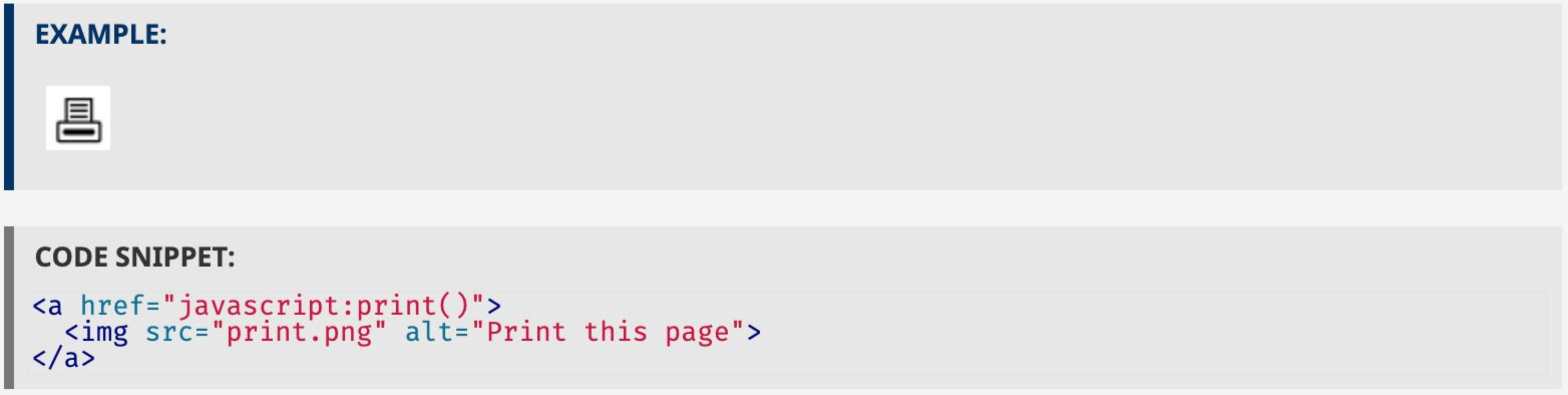
EXAMPLE:

2012 Annual report and accounts (43KB), also available in (254KB) or (353KB) format.

CODE SNIPPET:

```
<p>
  <a href="…">
    2012 Annual report and accounts
    <img src="html5logo.png" alt="HTML"> (43KB)
  </a>, also available in
  <a href="…">
    <img src="worddocument.png" alt="Word document"> (254KB)
  </a>
  or
  <a href="…">
    <img src="pdfdocument.png" alt="PDF"> (353KB)
  </a>
  format.
</p>
```

Functional: Stand-alone image that has a function



EXAMPLE:

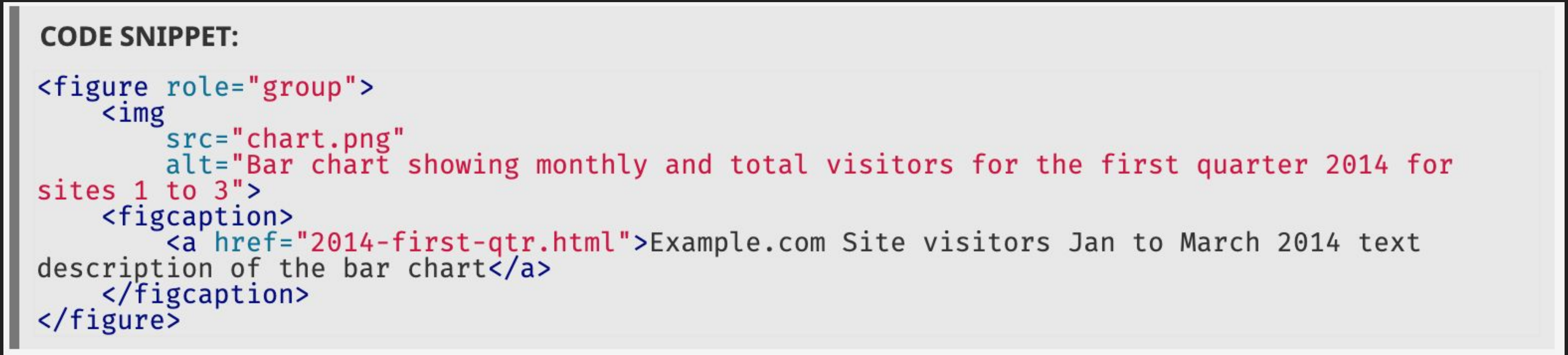
CODE SNIPPET:

```
<a href="javascript:print()">
  <img src="print.png" alt="Print this page">
</a>
```

Complex Images

- graphs and charts
- diagrams and illustrations
- maps

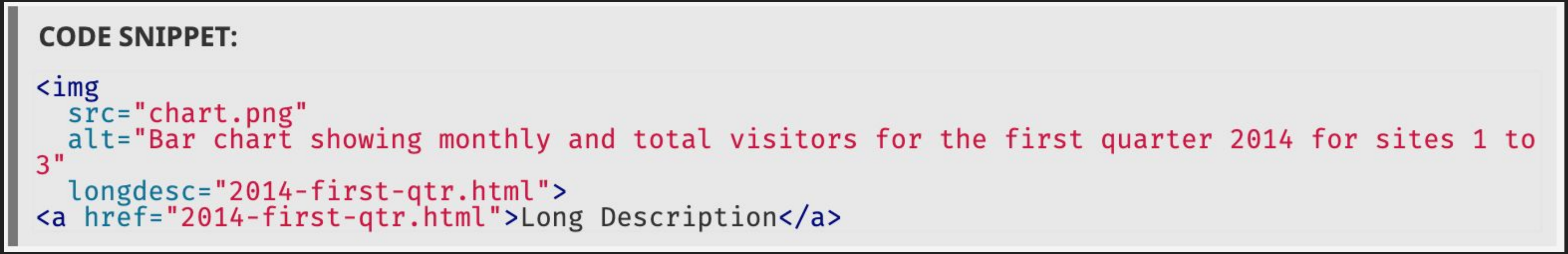
Complex Images: Link to detailed description



CODE SNIPPET:

```
<figure role="group">
    <img
        src="chart.png"
        alt="Bar chart showing monthly and total visitors for the first quarter 2014 for
sites 1 to 3">
    <figcaption>
        <a href="2014-first-qtr.html">Example.com Site visitors Jan to March 2014 text
description of the bar chart</a>
    </figcaption>
</figure>
```

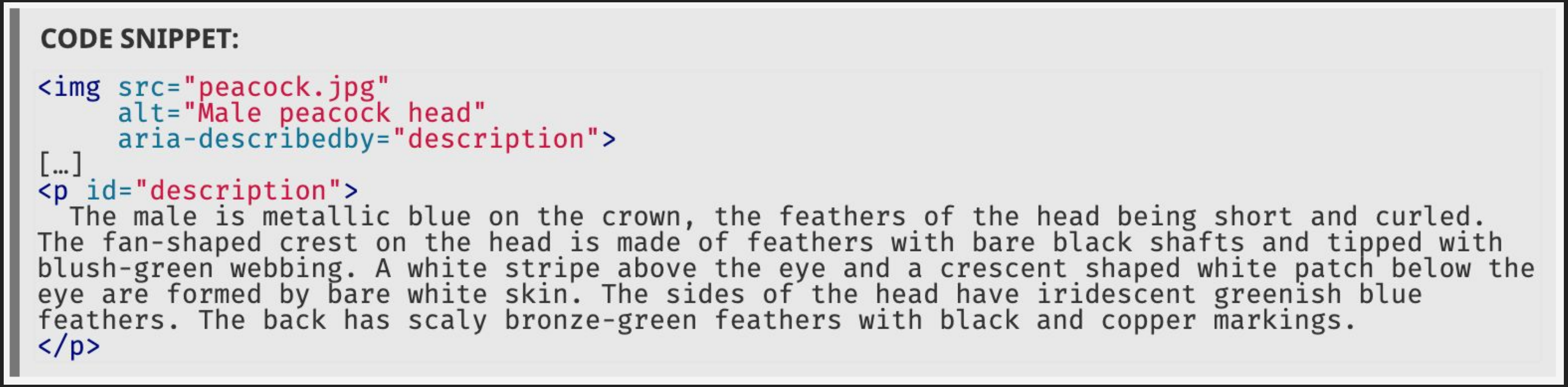
Complex Images: longdesc attribute



CODE SNIPPET:

```
<img
  src="chart.png"
  alt="Bar chart showing monthly and total visitors for the first quarter 2014 for sites 1 to
3"
  longdesc="2014-first-qtr.html">
<a href="2014-first-qtr.html">Long Description</a>
```

Complex Images: aria-describedby attribute



CODE SNIPPET:

```
<img src="peacock.jpg"
     alt="Male peacock head"
     aria-describedby="description">
[…]
<p id="description">
  The male is metallic blue on the crown, the feathers of the head being short and curled.
The fan-shaped crest on the head is made of feathers with bare black shafts and tipped with
blush-green webbing. A white stripe above the eye and a crescent shaped white patch below the
eye are formed by bare white skin. The sides of the head have iridescent greenish blue
feathers. The back has scaly bronze-green feathers with black and copper markings.
</p>
```

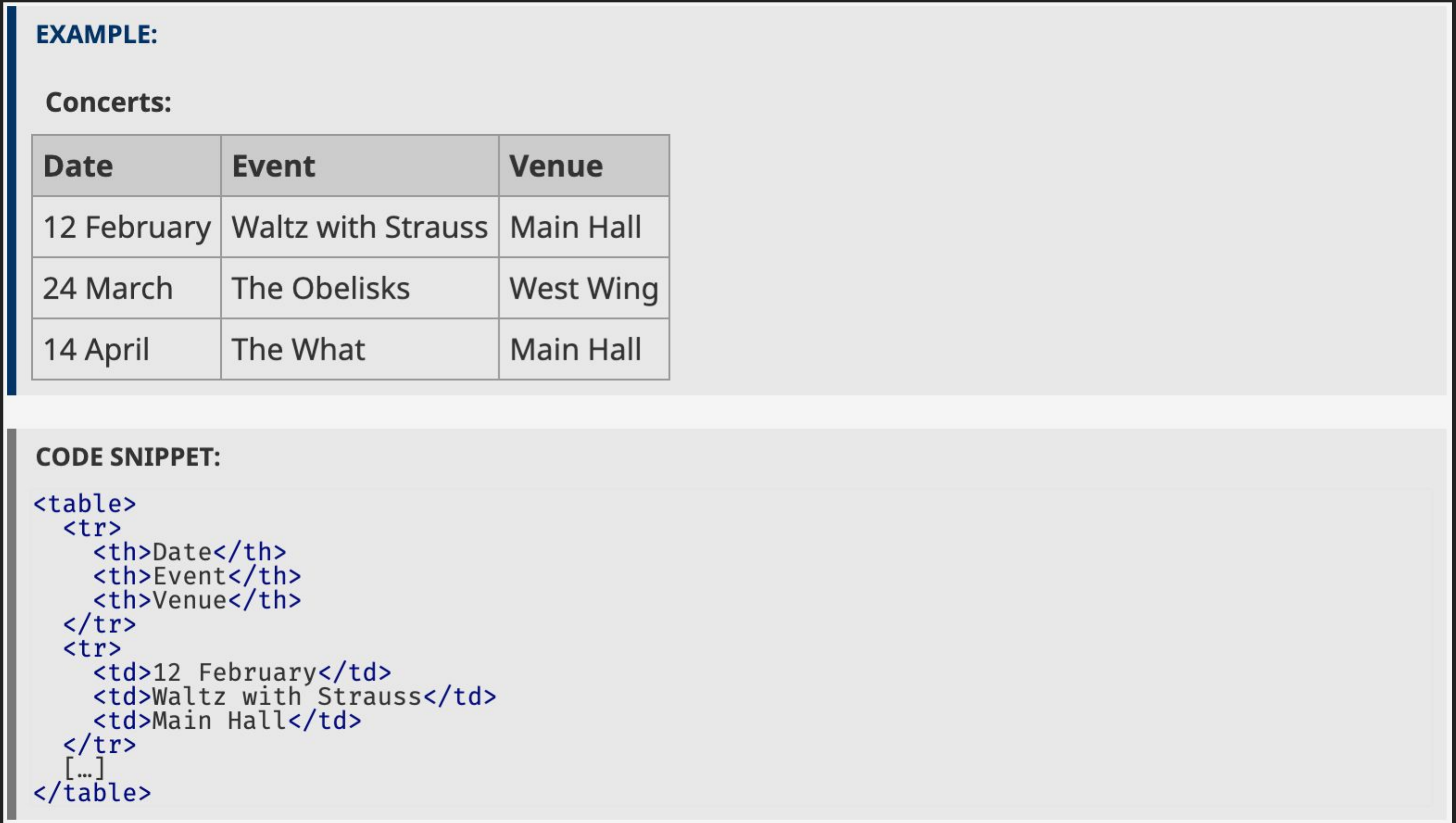
Workshop 5

- Add promotional aside to web page containing image

Table Concepts

- Header cells must be marked up with <th>, and data cells with <td> to make tables accessible
- For more complex tables, explicit associations may be needed using scope, id, and headers attributes

Tables: One header



EXAMPLE:

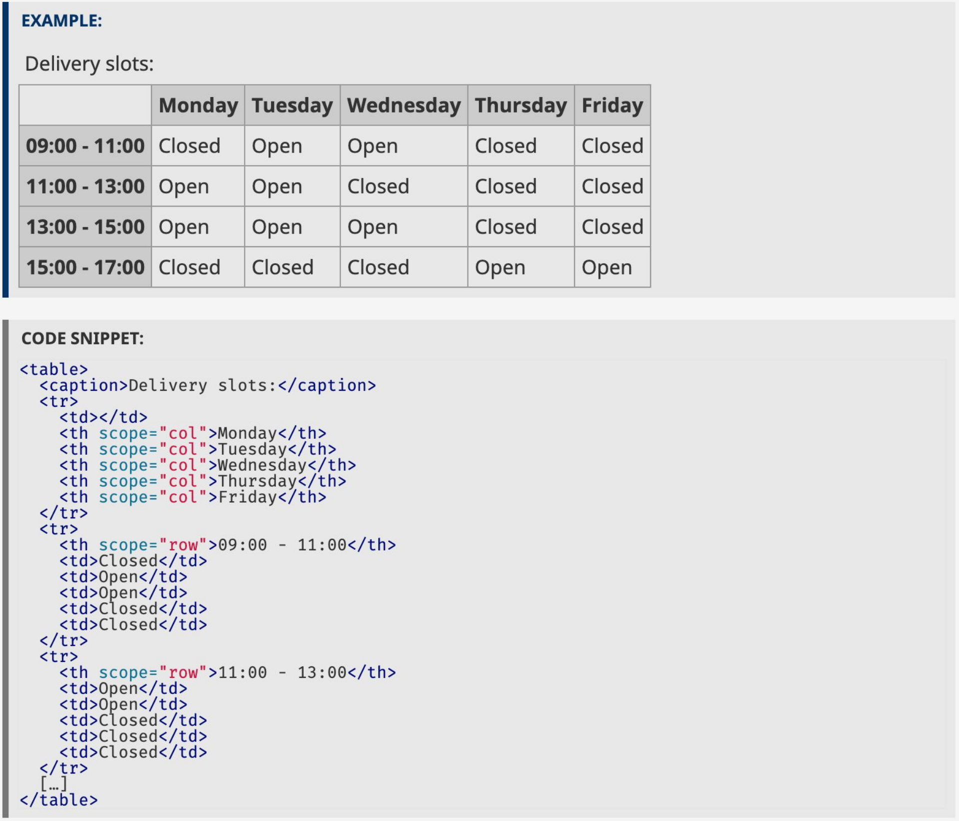
Concerts:

Date	Event	Venue
12 February	Waltz with Strauss	Main Hall
24 March	The Obelisks	West Wing
14 April	The What	Main Hall

CODE SNIPPET:

```
<table>
  <tr>
    <th>Date</th>
    <th>Event</th>
    <th>Venue</th>
  </tr>
  <tr>
    <td>12 February</td>
    <td>Waltz with Strauss</td>
    <td>Main Hall</td>
  </tr>
  […]
</table>
```

Tables: Two headers



EXAMPLE:

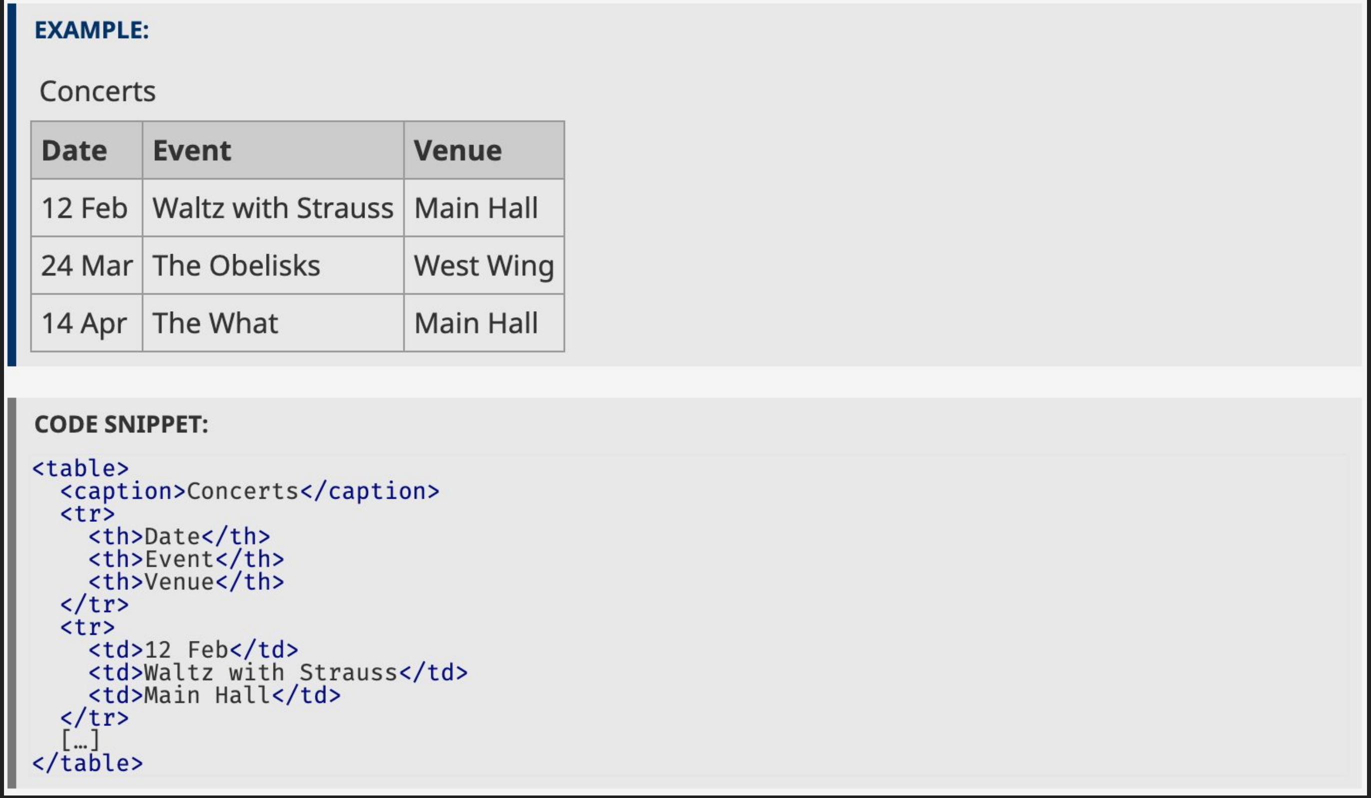
Delivery slots:

	Monday	Tuesday	Wednesday	Thursday	Friday
09:00 - 11:00	Closed	Open	Open	Closed	Closed
11:00 - 13:00	Open	Open	Closed	Closed	Closed
13:00 - 15:00	Open	Open	Open	Closed	Closed
15:00 - 17:00	Closed	Closed	Closed	Open	Open

CODE SNIPPET:

```
<table>
  <caption>Delivery slots:</caption>
  <tr>
    <td></td>
    <th scope="col">Monday</th>
    <th scope="col">Tuesday</th>
    <th scope="col">Wednesday</th>
    <th scope="col">Thursday</th>
    <th scope="col">Friday</th>
  </tr>
  <tr>
    <th scope="row">09:00 - 11:00</th>
    <td>Closed</td>
    <td>Open</td>
    <td>Open</td>
    <td>Closed</td>
    <td>Closed</td>
  </tr>
  <tr>
    <th scope="row">11:00 - 13:00</th>
    <td>Open</td>
    <td>Open</td>
    <td>Closed</td>
    <td>Closed</td>
    <td>Closed</td>
  </tr>
  […]
</table>
```

Tables: Caption



EXAMPLE:

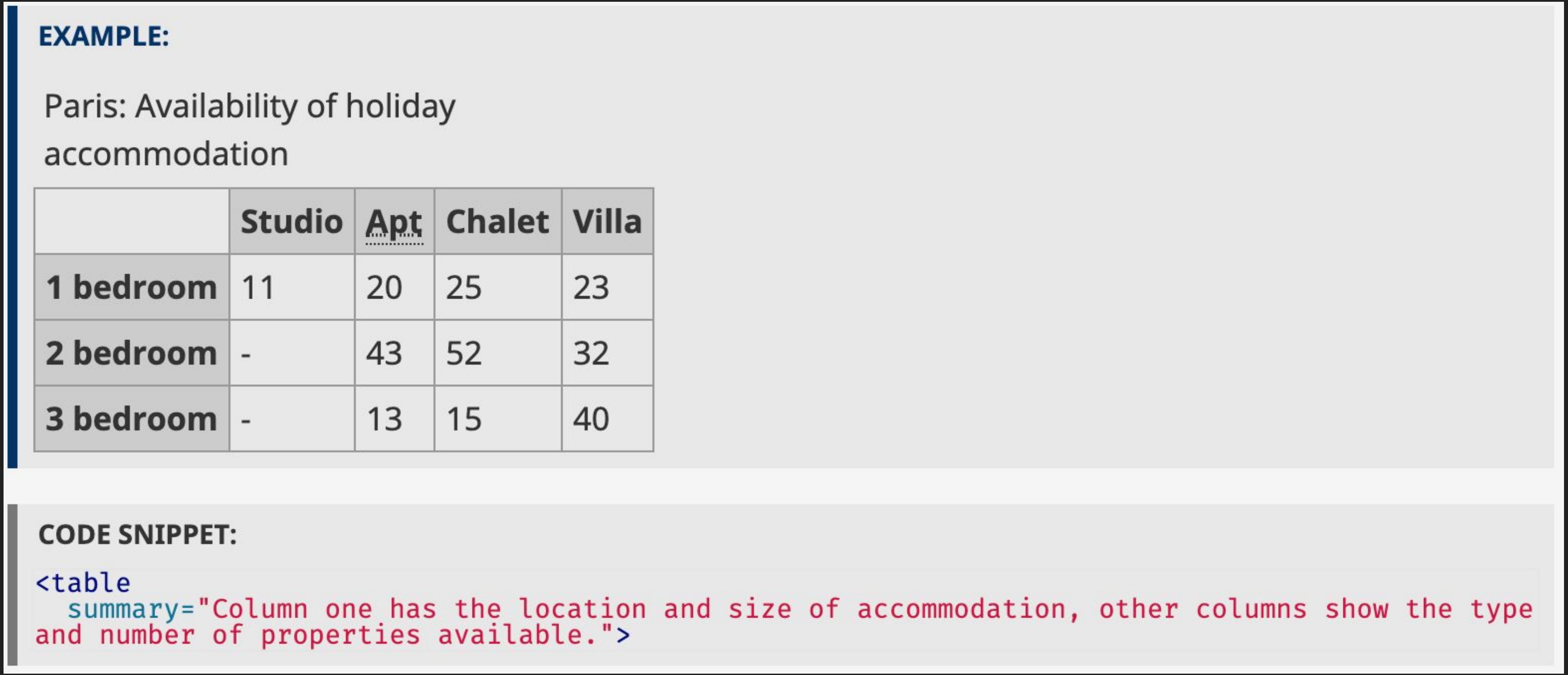
Concerts

Date	Event	Venue
12 Feb	Waltz with Strauss	Main Hall
24 Mar	The Obelisks	West Wing
14 Apr	The What	Main Hall

CODE SNIPPET:

```
<table>
  <caption>Concerts</caption>
  <tr>
    <th>Date</th>
    <th>Event</th>
    <th>Venue</th>
  </tr>
  <tr>
    <td>12 Feb</td>
    <td>Waltz with Strauss</td>
    <td>Main Hall</td>
  </tr>
  […]
</table>
```

Tables: Summary



EXAMPLE:

Paris: Availability of holiday accommodation

	Studio	Apt	Chalet	Villa
1 bedroom	11	20	25	23
2 bedroom	-	43	52	32
3 bedroom	-	13	15	40

CODE SNIPPET:

```
<table
  summary="Column one has the location and size of accommodation, other columns show the type
and number of properties available.">
```

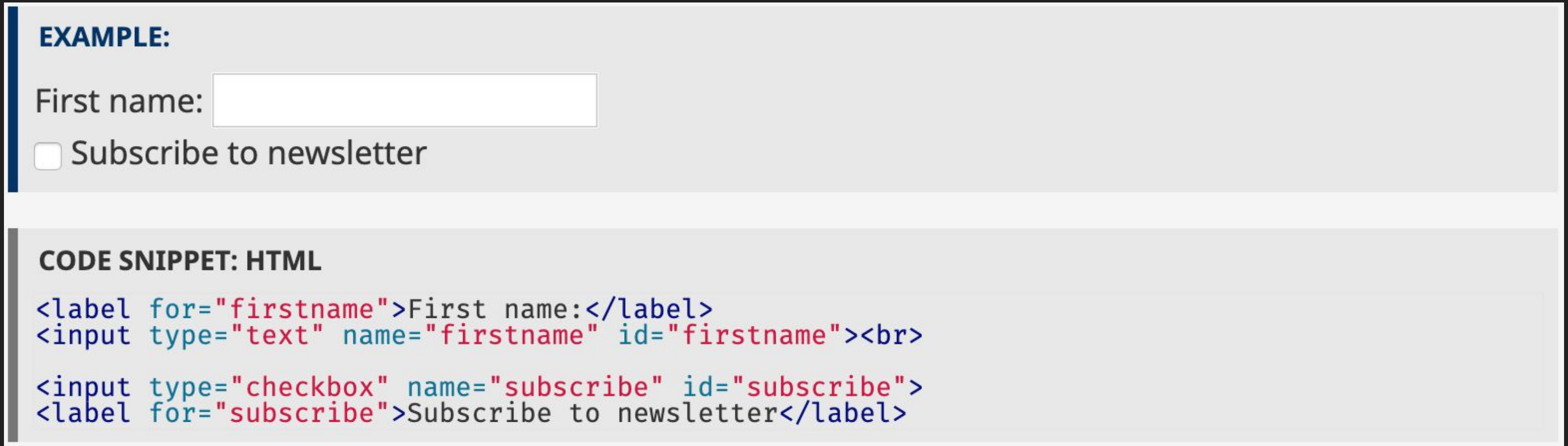
Workshop 6

- Create new Products page containing table of products

Form Concepts

- Labeling Controls: Use the <label> element to identify each form control
- Grouping Controls: Use the <fieldset> and <legend> elements to group and associate related form controls
- Form Instructions: Provide instructions to help users understand how to complete the form
- Validating Input: Validate input provided by the user and provide options to undo changes and confirm data entry
- User Notifications: Notify users about their progress

Labeling Controls: Explicit labels



EXAMPLE:

First name:

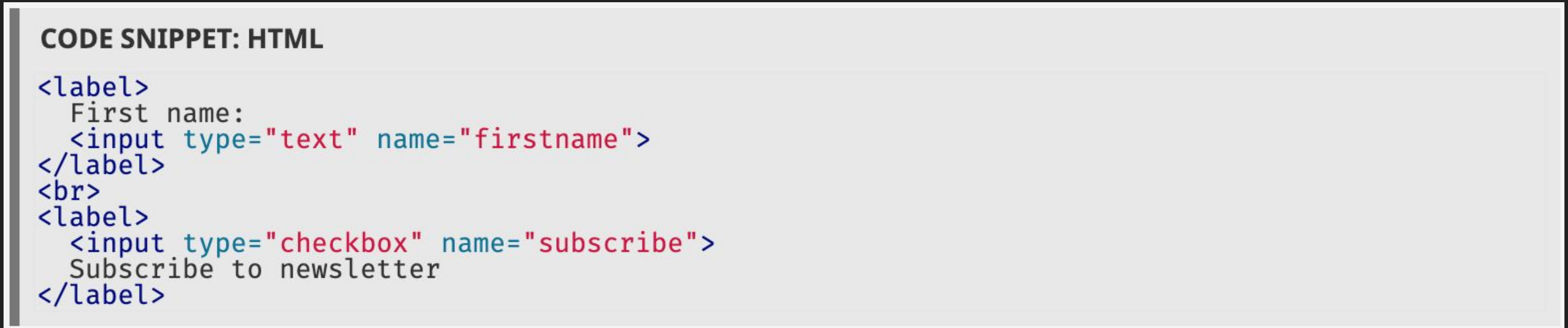
□ Subscribe to newsletter

CODE SNIPPET: HTML

```html
<label for="firstname">First name:</label>
<input type="text" name="firstname" id="firstname"><br>

<input type="checkbox" name="subscribe" id="subscribe">
<label for="subscribe">Subscribe to newsletter</label>
```

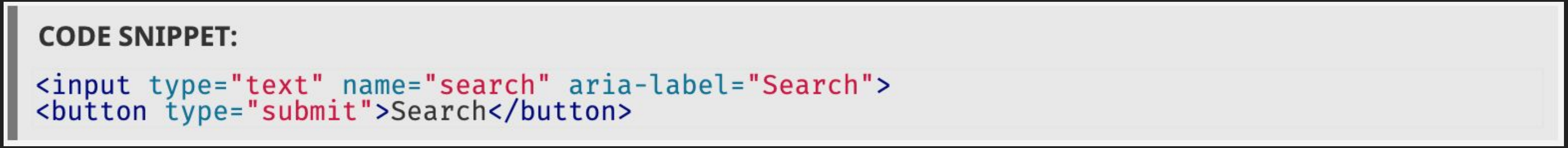
Labeling Controls: Implicit labels



CODE SNIPPET: HTML

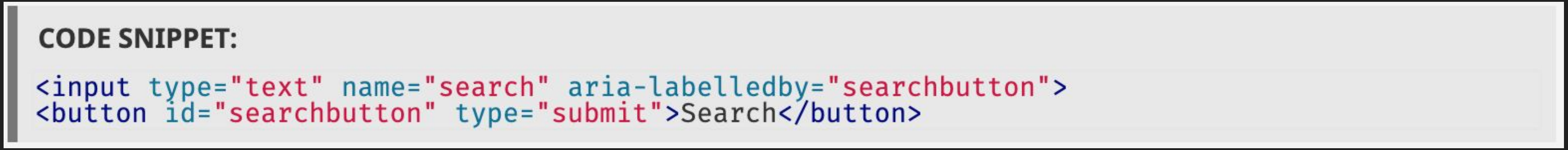
```html
<label>
  First name:
  <input type="text" name="firstname">
</label>
<br>
<label>
  <input type="checkbox" name="subscribe">
  Subscribe to newsletter
</label>
```

Labeling Controls: Hiding labels



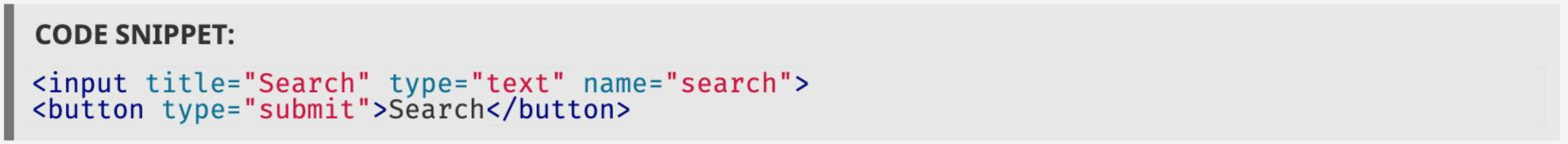
CODE SNIPPET:

```
<input type="text" name="search" aria-label="Search">
<button type="submit">Search</button>
```



CODE SNIPPET:

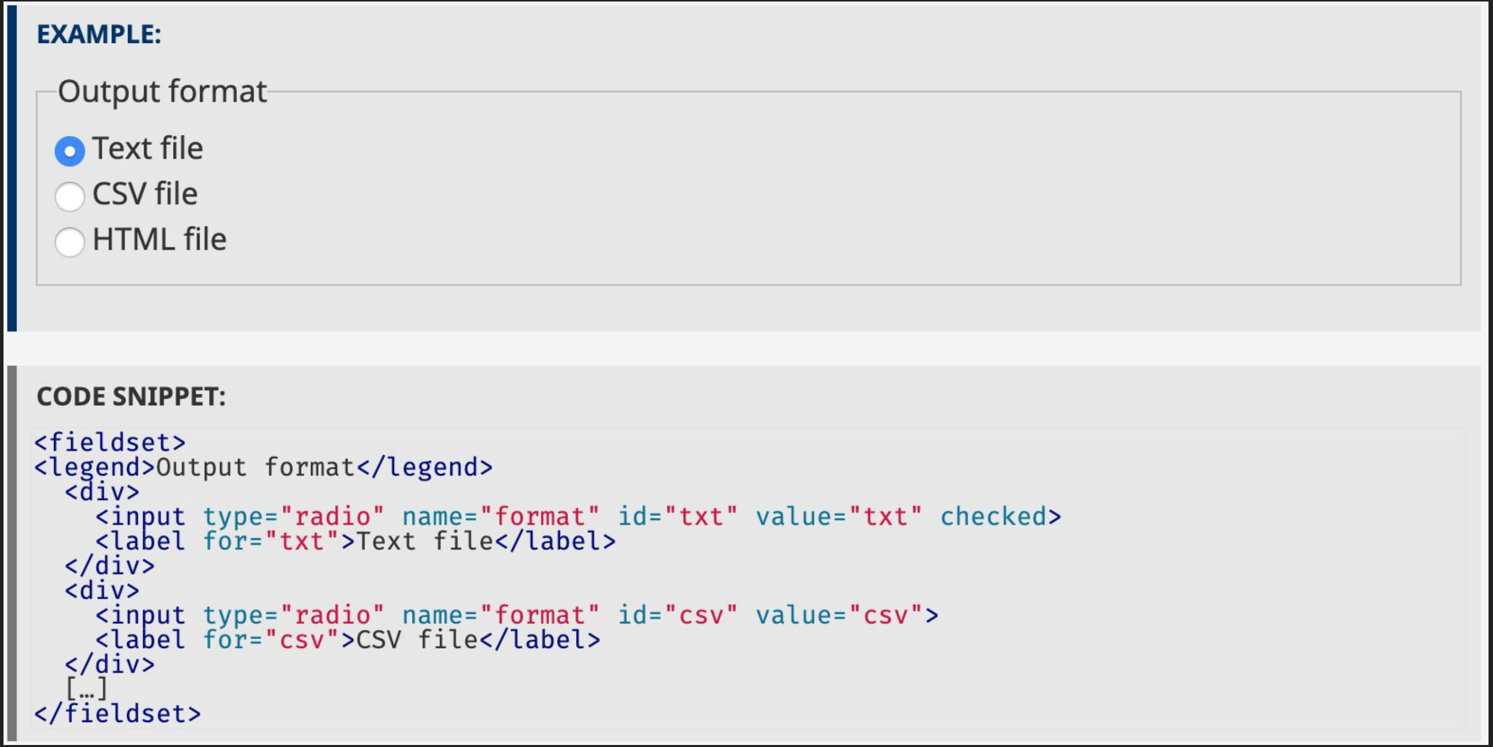
```
<input type="text" name="search" aria-labelledby="searchbutton">
<button id="searchbutton" type="submit">Search</button>
```



CODE SNIPPET:

```
<input title="Search" type="text" name="search">
<button type="submit">Search</button>
```

Grouping Controls: Associating related controls with fieldset



EXAMPLE:

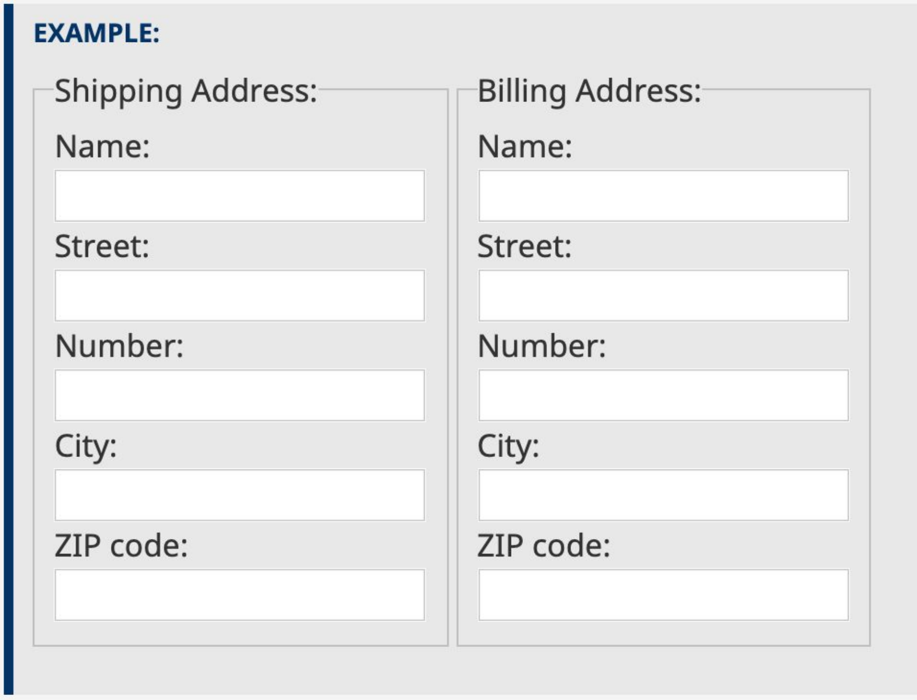
Output format

● Text file
○ CSV file
○ HTML file

CODE SNIPPET:

```
<fieldset>
<legend>Output format</legend>
  <div>
    <input type="radio" name="format" id="txt" value="txt" checked>
    <label for="txt">Text file</label>
  </div>
  <div>
    <input type="radio" name="format" id="csv" value="csv">
    <label for="csv">CSV file</label>
  </div>
  […]
</fieldset>
```

Grouping Controls: Related fields



EXAMPLE:

Shipping Address:

Name:

Street:

Number:

City:

ZIP code:

Billing Address:

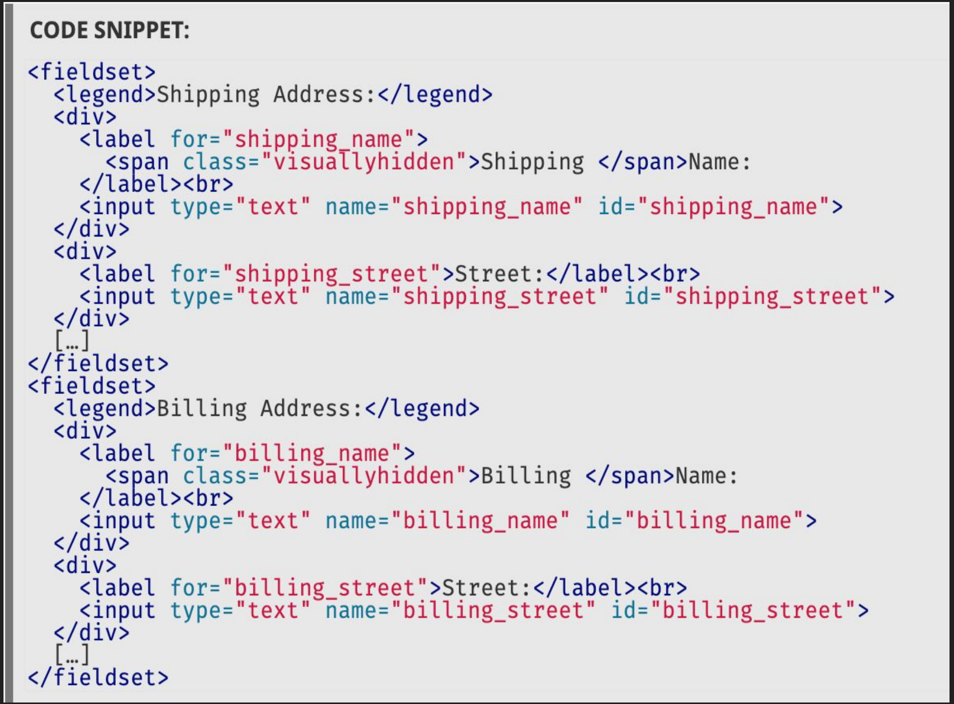
Name:

Street:

Number:

City:

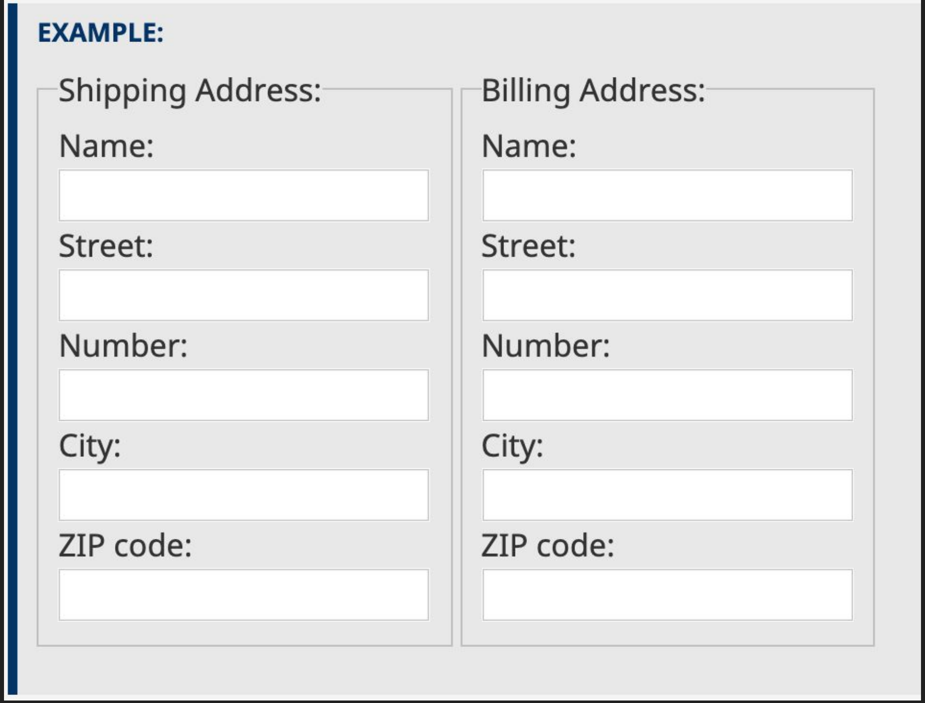
ZIP code:



CODE SNIPPET:

```
<fieldset>
  <legend>Shipping Address:</legend>
  <div>
    <label for="shipping_name">
      <span class="visuallyhidden">Shipping </span>Name:
    </label><br>
    <input type="text" name="shipping_name" id="shipping_name">
  </div>
  <div>
    <label for="shipping_street">Street:</label><br>
    <input type="text" name="shipping_street" id="shipping_street">
  </div>
  […]
</fieldset>
<fieldset>
  <legend>Billing Address:</legend>
  <div>
    <label for="billing_name">
      <span class="visuallyhidden">Billing </span>Name:
    </label><br>
    <input type="text" name="billing_name" id="billing_name">
  </div>
  <div>
    <label for="billing_street">Street:</label><br>
    <input type="text" name="billing_street" id="billing_street">
  </div>
  […]
</fieldset>
```

Grouping Controls: Associating related controls with WAI-ARIA



EXAMPLE:

Shipping Address:

Name:

Street:

Number:

City:

ZIP code:

Billing Address:

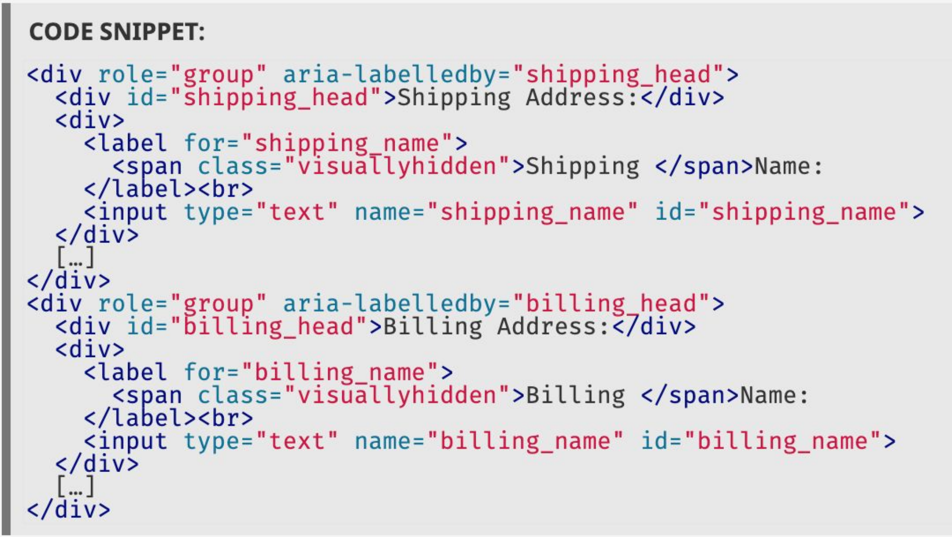
Name:

Street:

Number:

City:

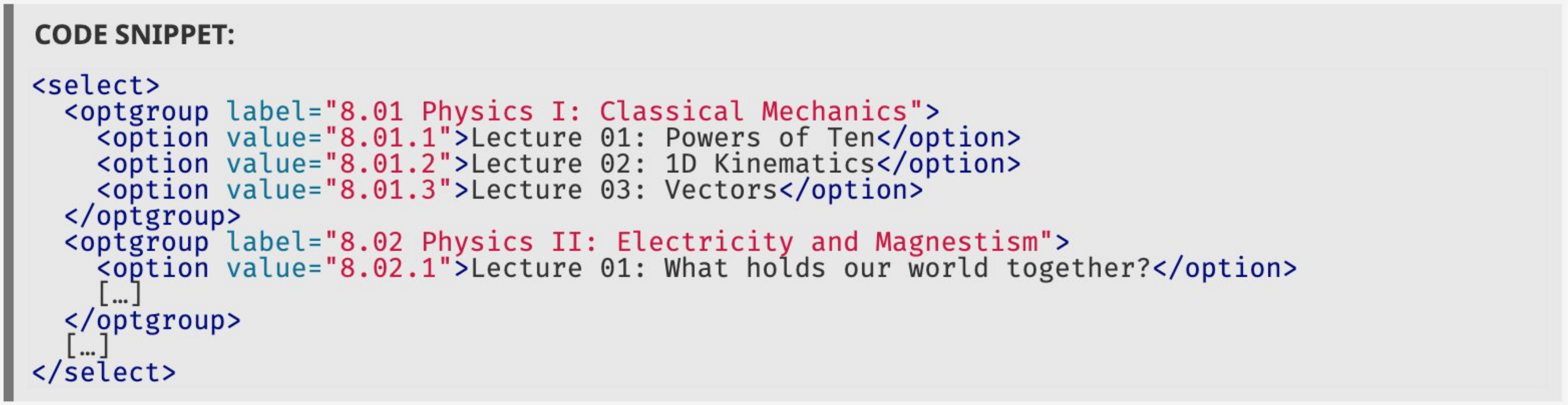
ZIP code:



CODE SNIPPET:

```
<div role="group" aria-labelledby="shipping_head">
  <div id="shipping_head">Shipping Address:</div>
  <div>
    <label for="shipping_name">
      <span class="visuallyhidden">Shipping </span>Name:
    </label><br>
    <input type="text" name="shipping_name" id="shipping_name">
  </div>
  […]
</div>
<div role="group" aria-labelledby="billing_head">
  <div id="billing_head">Billing Address:</div>
  <div>
    <label for="billing_name">
      <span class="visuallyhidden">Billing </span>Name:
    </label><br>
    <input type="text" name="billing_name" id="billing_name">
  </div>
  […]
</div>
```

Grouping Controls: Related items in select elements



CODE SNIPPET:

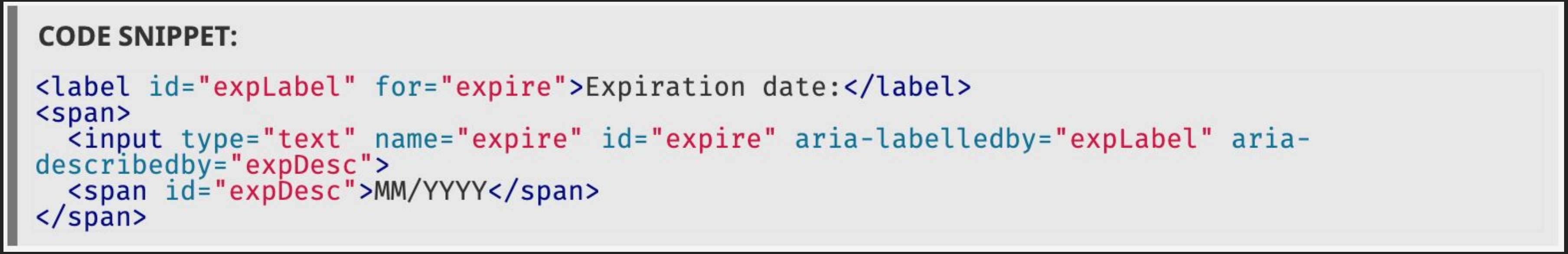
```
<select>
  <optgroup label="8.01 Physics I: Classical Mechanics">
    <option value="8.01.1">Lecture 01: Powers of Ten</option>
    <option value="8.01.2">Lecture 02: 1D Kinematics</option>
    <option value="8.01.3">Lecture 03: Vectors</option>
  </optgroup>
  <optgroup label="8.02 Physics II: Electricity and Magnestism">
    <option value="8.02.1">Lecture 01: What holds our world together?</option>
    […]
  </optgroup>
  […]
</select>
```

Instructions: Overall instructions

- Where relevant, provide overall instructions that apply to the entire form
- Provide such instructions before the <form> element:-
  - All fields marked “required” must be completed.
  - Dates should all be typed in the format dd/mm/yyyy, (as in 21/07/2013).
  - Passwords must contain at least 8 letters and/or numbers.

Instructions: Inline instructions

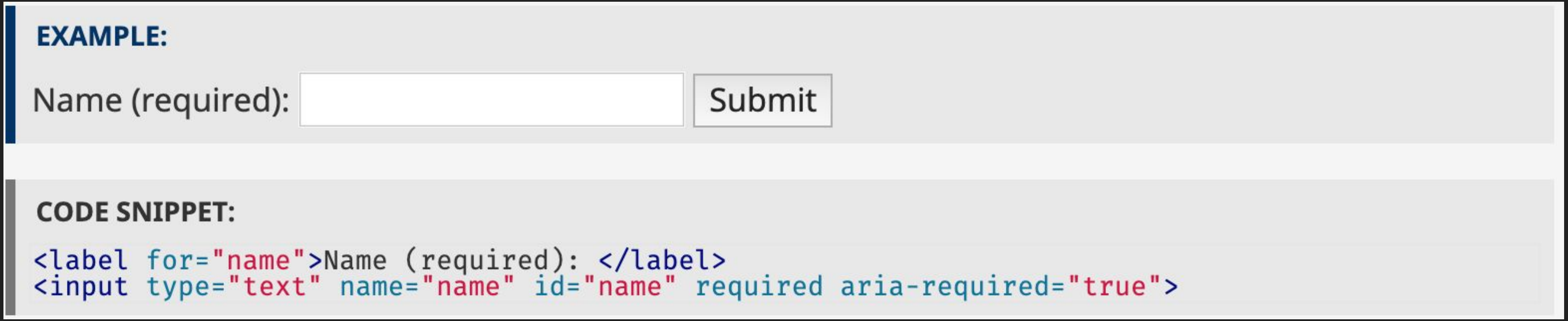
- Provide instructions within labels
- Provide using aria-labelledby and aria-describedby



CODE SNIPPET:

```html
<label id="expLabel" for="expire">Expiration date:</label>
<span>
  <input type="text" name="expire" id="expire" aria-labelledby="expLabel" aria-describedby="expDesc">
  <span id="expDesc">MM/YYYY</span>
</span>
```

Validating: Required input



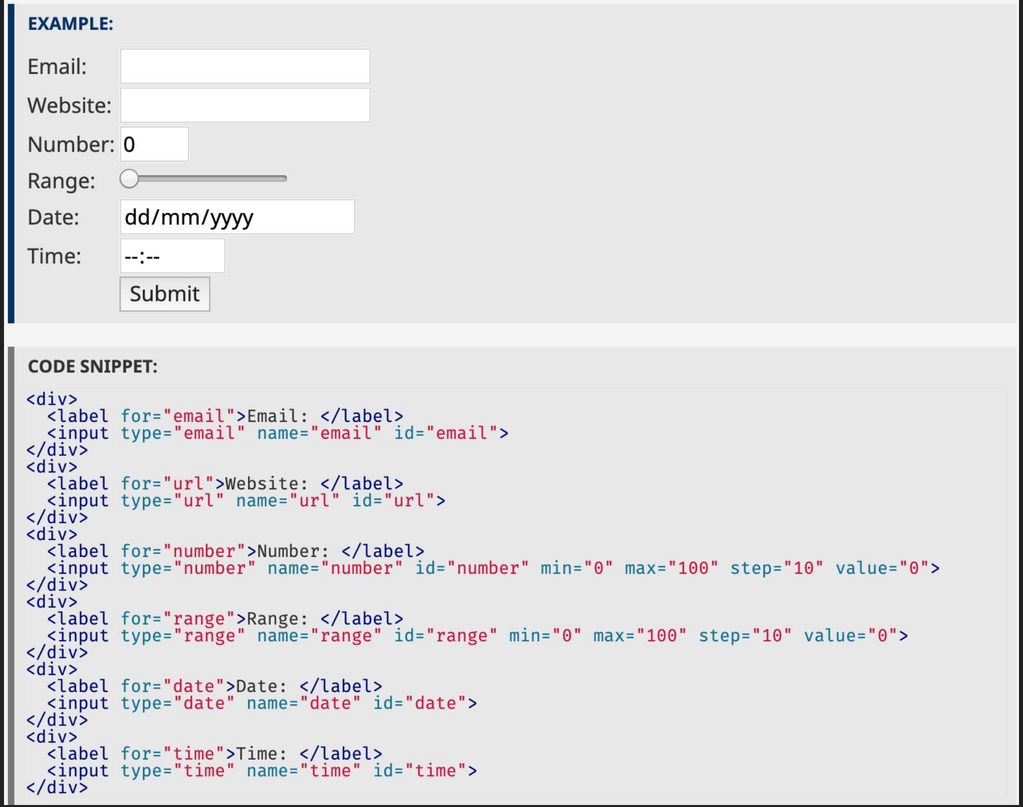
EXAMPLE:

Name (required):

CODE SNIPPET:

```html
<label for="name">Name (required): </label>
<input type="text" name="name" id="name" required aria-required="true">
```

Validating: Common input types



EXAMPLE:

Email:

Website:

Number: 0

Range:

Date: dd/mm/yyyy

Time: --:--

Submit

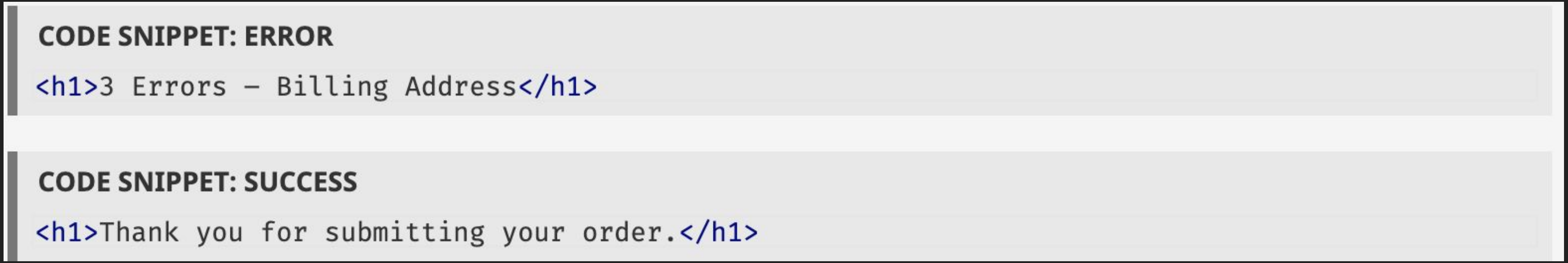
CODE SNIPPET:

```
<div>
  <label for="email">Email: </label>
  <input type="email" name="email" id="email">
</div>
<div>
  <label for="url">Website: </label>
  <input type="url" name="url" id="url">
</div>
<div>
  <label for="number">Number: </label>
  <input type="number" name="number" id="number" min="0" max="100" step="10" value="0">
</div>
<div>
  <label for="range">Range: </label>
  <input type="range" name="range" id="range" min="0" max="100" step="10" value="0">
</div>
<div>
  <label for="date">Date: </label>
  <input type="date" name="date" id="date">
</div>
<div>
  <label for="time">Time: </label>
  <input type="time" name="time" id="time">
</div>
```

Validating: Client-side validation benefits

- In general, client-side validation results in a better user experience and makes resolving validation errors more understandable
- However, not all web browsers support HTML5
- Validation needs to be carried out server-side as well

User Notification: Main headings



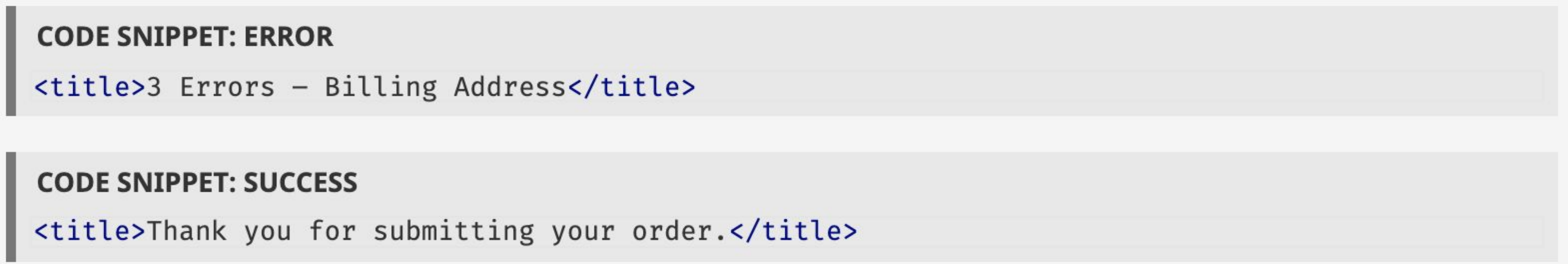
CODE SNIPPET: ERROR

```
<h1>3 Errors – Billing Address</h1>
```

CODE SNIPPET: SUCCESS

```
<h1>Thank you for submitting your order.</h1>
```

User Notification: Page title



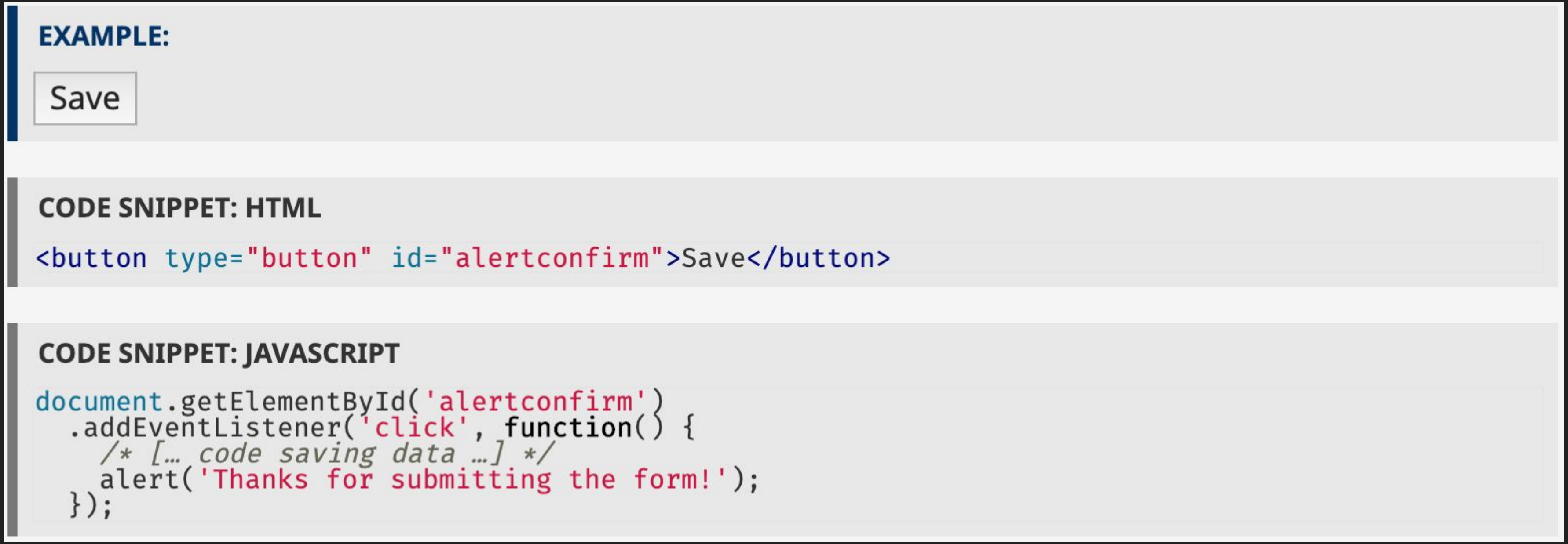
CODE SNIPPET: ERROR

```
<title>3 Errors – Billing Address</title>
```

CODE SNIPPET: SUCCESS

```
<title>Thank you for submitting your order.</title>
```

User Notification: Using a dialog



EXAMPLE:

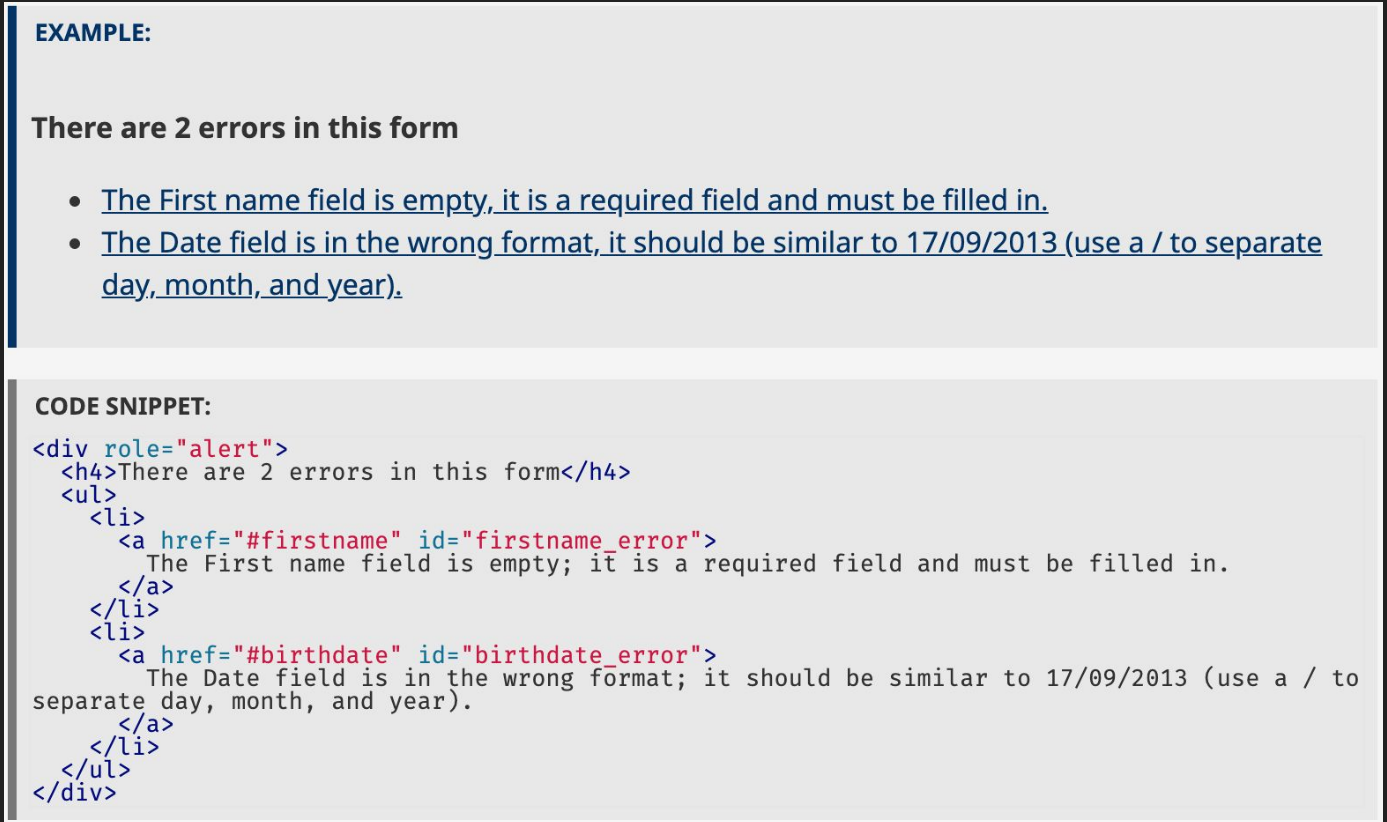
CODE SNIPPET: HTML

```html
<button type="button" id="alertconfirm">Save</button>
```

CODE SNIPPET: JAVASCRIPT

```javascript
document.getElementById('alertconfirm')
  .addEventListener('click', function() {
    /* [… code saving data …] */
    alert('Thanks for submitting the form!');
  });
```

User Notification: Listing errors



EXAMPLE:

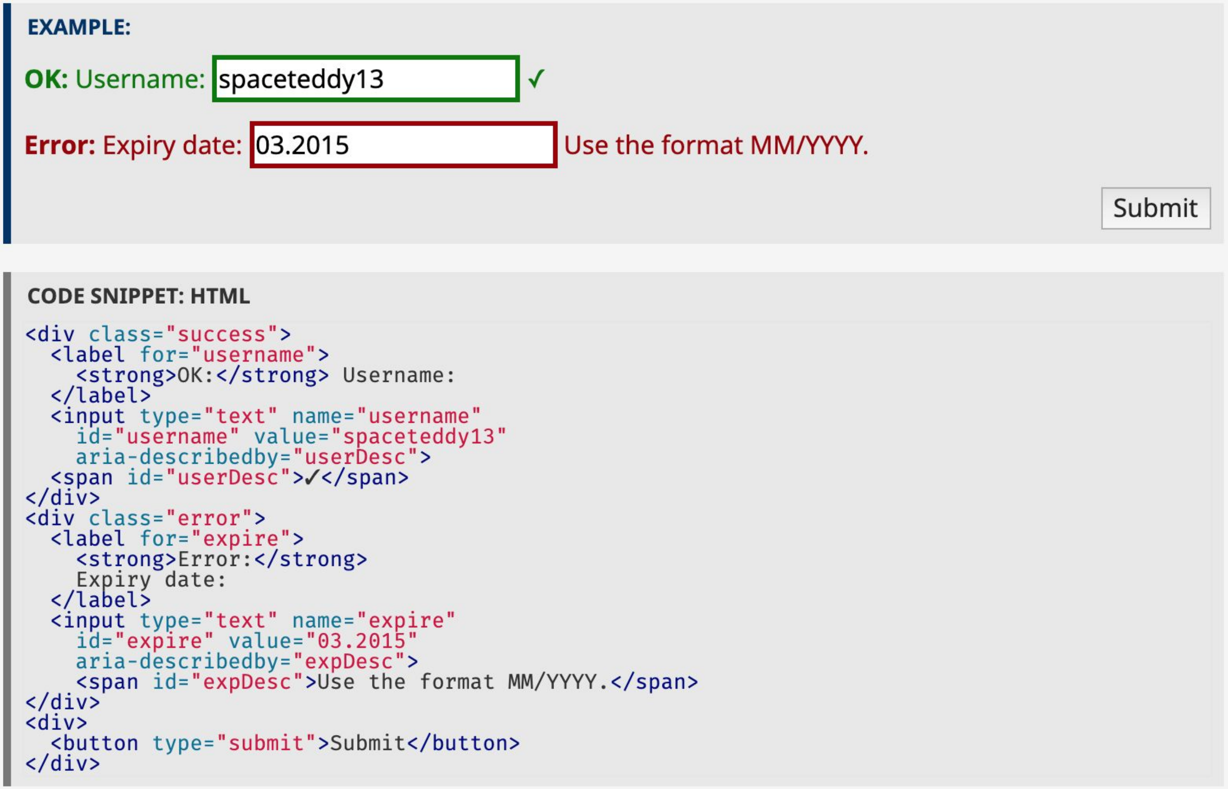
There are 2 errors in this form

- The First name field is empty, it is a required field and must be filled in.
- The Date field is in the wrong format, it should be similar to 17/09/2013 (use a / to separate day, month, and year).

CODE SNIPPET:

```
<div role="alert">
  <h4>There are 2 errors in this form</h4>
  <ul>
    <li>
      <a href="#firstname" id="firstname_error">
        The First name field is empty; it is a required field and must be filled in.
      </a>
    </li>
    <li>
      <a href="#birthdate" id="birthdate_error">
        The Date field is in the wrong format; it should be similar to 17/09/2013 (use a / to
separate day, month, and year).
      </a>
    </li>
  </ul>
</div>
```

User Notification: Inline feedback after submit



EXAMPLE:

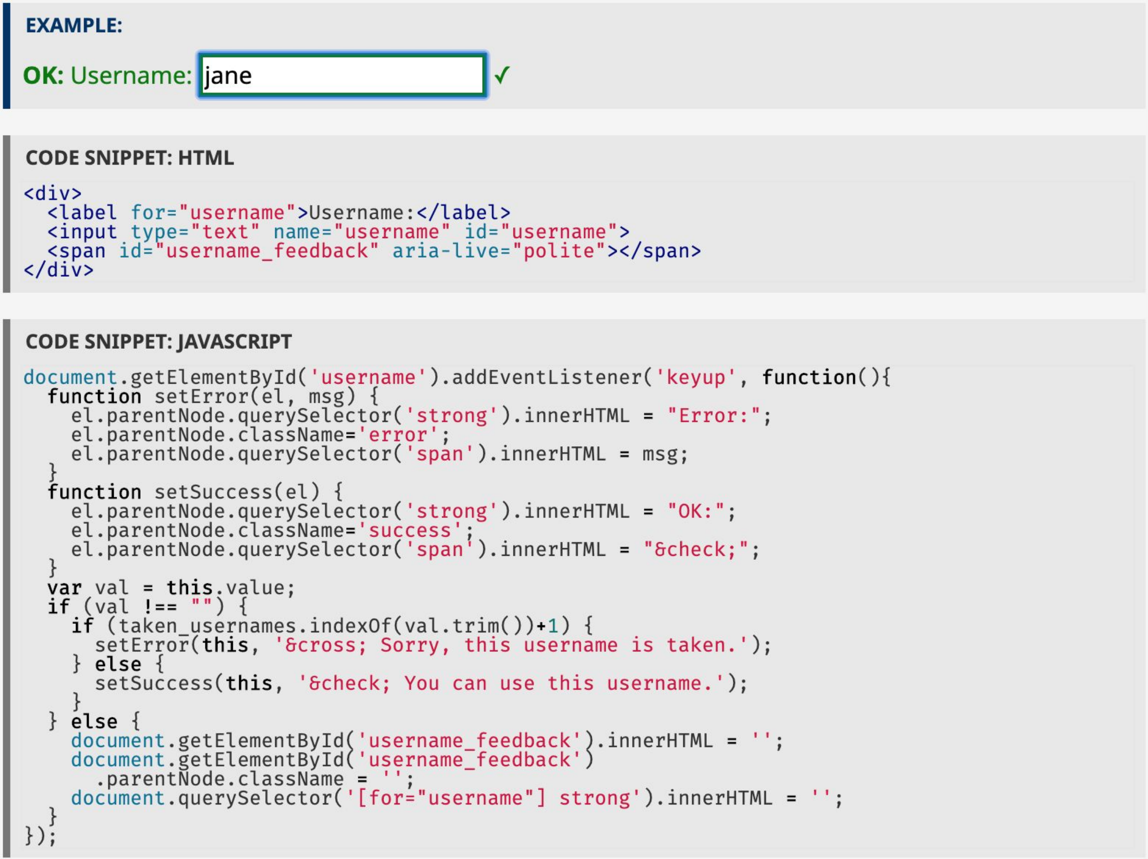
OK: Username: spaceteddy13 ✓

Error: Expiry date: 03.2015 Use the format MM/YYYY.

CODE SNIPPET: HTML

```
<div class="success">
  <label for="username">
    <strong>OK:</strong> Username:
  </label>
  <input type="text" name="username"
    id="username" value="spaceteddy13"
    aria-describedby="userDesc">
  <span id="userDesc">✓</span>
</div>
<div class="error">
  <label for="expire">
    <strong>Error:</strong>
    Expiry date:
  </label>
  <input type="text" name="expire"
    id="expire" value="03.2015"
    aria-describedby="expDesc">
    <span id="expDesc">Use the format MM/YYYY.</span>
</div>
<div>
  <button type="submit">Submit</button>
</div>
```

User Notification: Inline feedback during typing



EXAMPLE:

OK: Username: jane ✓

CODE SNIPPET: HTML

```
<div>
  <label for="username">Username:</label>
  <input type="text" name="username" id="username">
  <span id="username_feedback" aria-live="polite"></span>
</div>
```

CODE SNIPPET: JAVASCRIPT

```
document.getElementById('username').addEventListener('keyup', function(){
  function setError(el, msg) {
    el.parentNode.querySelector('strong').innerHTML = "Error:";
    el.parentNode.className='error';
    el.parentNode.querySelector('span').innerHTML = msg;
  }
  function setSuccess(el) {
    el.parentNode.querySelector('strong').innerHTML = "OK:";
    el.parentNode.className='success';
    el.parentNode.querySelector('span').innerHTML = "&check;";
  }
  var val = this.value;
  if (val !== "") {
    if (taken_usernames.indexOf(val.trim())+1) {
      setError(this, '&cross; Sorry, this username is taken.');
    } else {
      setSuccess(this, '&check; You can use this username.');
    }
  } else {
    document.getElementById('username_feedback').innerHTML = '';
    document.getElementById('username_feedback')
      .parentNode.className = '';
    document.querySelector('[for="username"] strong').innerHTML = '';
  }
});
```

Workshop 7

- Create a Contact page containing a form

Tab Indexing

- Some elements are automatically added to the tab index
  - Top level h1
  - All links
  - Form elements
- Use the tabindex attribute to manually add any element
- tabindex=”0” adds an element to the natural tab flow
- tabindex greater than 0 allows finer control of the order

Workshop 8

- Control use of tab navigation with the tabindex attribute

Video Content

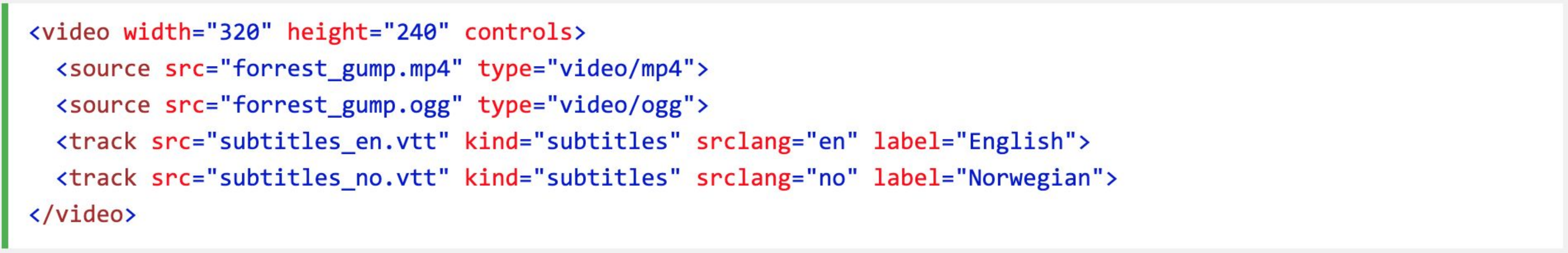
- Audio Description of Visual Information - eg “boy strokes dog”
- Captions/Subtitles - synchronized with the audio and usually shown in a media player when users turn them on
- Transcripts - text version of the speech and non-speech audio information needed to understand the content (provided as a link)
- Sign Languages - for many people who are deaf, sign language is their native language (AAA)

How to Add Captions

- Youtube videos offer functionality for built-in captions
- HTML5 videos can be pointed to VTT files

Referencing VTT Files

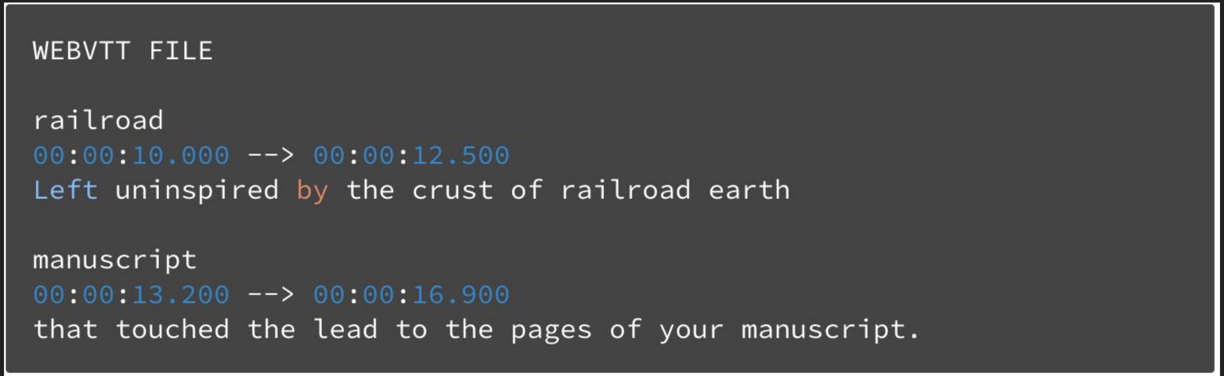
- Use the <track> element to point to 1 or more text tracks



```
<video width="320" height="240" controls>
  <source src="forrest_gump.mp4" type="video/mp4">
  <source src="forrest_gump.ogg" type="video/ogg">
  <track src="subtitles_en.vtt" kind="subtitles" srclang="en" label="English">
  <track src="subtitles_no.vtt" kind="subtitles" srclang="no" label="Norwegian">
</video>
```

VTT Files

- Text and timing controlled with simple format



```
WEBVTT FILE

railroad
00:00:10.000 --> 00:00:12.500
Left uninspired by the crust of railroad earth

manuscript
00:00:13.200 --> 00:00:16.900
that touched the lead to the pages of your manuscript.
```

Workshop 9

- Integrating a video and applying captions

Conformance Evaluation

- Conformance evaluation determines how well web pages or applications meet accessibility standards
- W3C's Website Accessibility Conformance Evaluation Methodology (WCAG-EM) is an approach for determining conformance to Web Content Accessibility Guidelines (WCAG)

Who WCAG-EM is For?

- Anyone who wants a common procedure for auditing websites
    - internal evaluators
    - external auditors
- Additionally, WCAG-EM can be referred to by managers, procurers, policy makers, regulators, and others

What is in WCAG-EM?

- Define the scope of the evaluation -
    - what is included in the evaluation
    - goal of the evaluation
    - conformance level (A, AA, AAA)
- Identifying key web pages
    - key functionality
    - types of web content
    - required web technologies
- Select a representative sample
- Evaluate the selected sample
- Report the evaluation findings

WCAG-EM Report Tool

- The WCAG-EM Report Tool: Website Accessibility Evaluation Report Generator helps you follow the steps of WCAG-EM and generate a structured report from the input that you provide

Accessibility Policy

- Declare to the world with an Accessibility Policy
- Usually a link from the footer
- Opportunity to emphasize your organization’s commitment to inclusion
- Gives contact information for an Accessibility Point Person in your organization
- Shows the past and future steps in your path towards accessibility
- Example of typical policy statement

Certification

- Certification typically includes a badge or seal and specific wording on your Accessibility Policy page stating that your website has been independently certified by an agency specializing in web accessibility exceeding minimum accessibility requirements

Links to Useful Resources

- UK Government Accessibility Guidelines
- WCAG Quick Reference
- Evaluating Web Accessibility Conformity Overview
- Easy Checks - A First Review of Web Accessibility
- WAI-ARIA Cheat sheet
- Accessible Video Guidelines
- Colour Contrast Checker