ACCESSIBLE FORM ERRORS

What is form validation?

When users fill in web forms, developers need to **make sure all form data is sent to the server in the correct format**. Otherwise there may be difficulties processing the data.

To make sure data arrives in the correct format, the web application **should check each field as they are filled in by users**, or as the user submits the form.

If the data in each field is correct, then the form data **can be submitted to the server**.

If any of the data is incorrect, users should be **presented with an error message for each invalid field**, before the form is submitted to the server.

Types of form validation

There are two types of form validation - **client-side** and **server-side**.

Client-side validation **occurs in the browser**, before the data is submitted to the server.

Client-side validation is most commonly managed using **JavaScript**.

However, validation can also be managed to some degree **using native HTML5 form validation features**.

Server-side validation is **validation that occurs on the server**, after the data has been submitted.

Regardless of whether client-side validation is present, server-side validation **should still be in place as a backup**.

Form field validation

“Form field validation” is where individual form fields are validated immediately **after the user has entered data**, and focus has moved out of the form field.

Each form field that has errors **should be “flagged” as an error**.

Flagging form fields **should not use colour alone** to signify errors.

Error messages **should be descriptive** so that they provide information that will help users fill in the field correctly.

Error messages **need to be programmatically associated with the form control**, so that they are announced to screen readers along with the label content.

This can be achieved use **three different methods**.

Exercise 5a:
Inside the label

Open **exercise05-errors/exercise5-start.html**

Review your work against **exercise05-errors/exercise5-finished.html**

Using this method, information can be associated with form fields by **placing it inside the `<label>` element**.

This method only works if the `<label>` element **wraps around the label content, form control and instructions**.

This technique is **stable across all web browsers and assistive technologies** so is the recommended solution.

```
<label for="name">
  <span class="label">Name</span>
  <input class="input input__invalid" id="name"
type="text">
  <span class="error-message">Error: Incomplete
name</span>
</label>
```

Exercise 5b:
Using aria-describedby

Continue working in **exercise05-errors/exercise5-start.html**

Review your work against **exercise05-errors/exercise5-finished.html**

Using this method, the `aria-describedby` attribute can be used to **associate instructions with form controls**.

It can be used in cases where it is **not possible** to wrap the `<label>` element around the form control.

```
<label class="label" for="email">Email</label>
<input class="input input__invalid" id="email"
type="text" aria-describedby="error1">
<span id="error1" class="error-message">Error: The
email address is invalid</span>
```

This technique has **good support** across all modern web browsers and assistive technologies.

On-submit form validation

“On-submit form validation” is where the **entire form is validated** as the user attempts to submit the form.

If there is an error in any field, **the following should occur** when the user triggers the submit button:

1. An error message **should appear at the top of the form** alerting users that there are errors.

2. Focus must be **taken to the error message**.

3. The error message should **list all errors** in the order that they occur.

4. Ideally, each listed error **should be a link** that takes the user to the relevant form control.

Exercise 5c: Form error

Continue working in **exercise05-errors/exercise5-start.html**

Review your work against **exercise05-errors/exercise5-finished.html**

The error message container should exist on the page, **even when non-active**. However, it should not contain any content until triggered.

The **markup** is as follows:

```
<div
  class="error-container"
  aria-live="assertive"
  style="display:block"
  role="alert"
  tabindex="0"
  aria-label="Error Message">
  <p>The form has three errors:</p>
  <ol>
    <li><a href="#error01">Error 1: Incomplete
name</a></li>
</div>
```