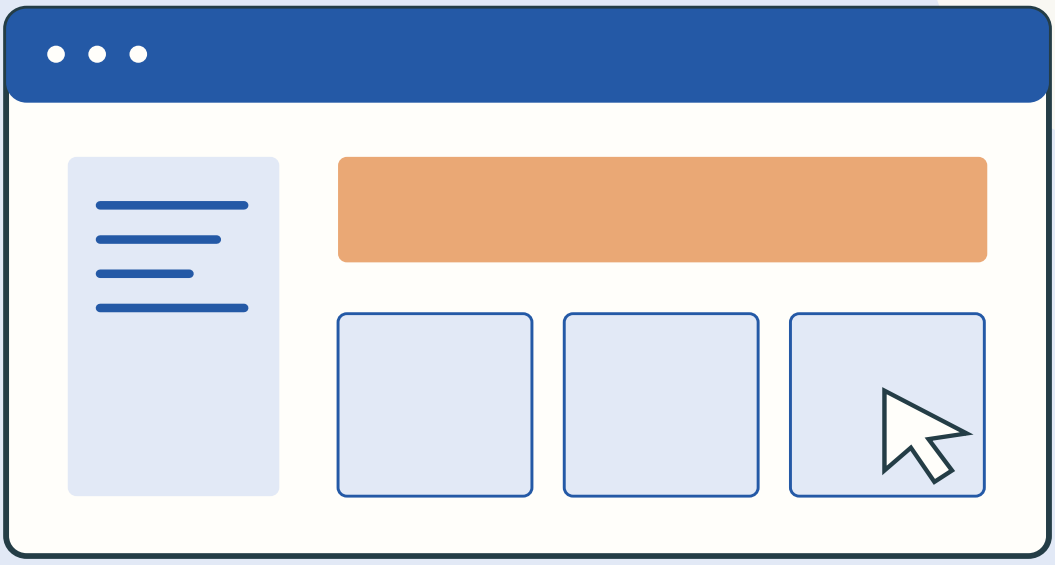
LABELED HTML FORMS

Building Labeled Form Mockups in HTML



LABELS / FIELDS

DIGITAL & TECHNOLOGY

Building Labeled Form Mockups in HTML

Learning outcomes

- Create explicit labels and help text for a fictional request mockup.
- Indicate required information in text as well as markup.
- Keep the exercise incapable of transmitting data.
- Translate a fictional field brief into exact labels, controls, required states and help associations.
- Repair duplicate ids, missing/wrong-topic descriptions and placeholder-only labels.
- Distinguish native constraint metadata, editing behavior and unimplemented validation workflows.
- Audit local non-submission boundaries across elements, attributes and resources.
- Apply synchronized field amendments and deliver a reviewable independent mockup with honest evidence.

Original training material with synthetic examples. Local practice only; no certification or production-readiness claim.

Study the foundations, work through the applied units, complete the practice tasks, and then attempt the assessment. Use only the supplied fictional practice data. Keep your original inputs and your revised work separate.

Core concepts

Model an interaction without collecting information

A static mockup is a design exercise, not an invitation to supply real details. State “Practice only: do not enter personal information” above the controls. Use invented item codes and request categories rather than names, contact details or passwords. For a deliberately non-submitting exercise, place the controls in a section rather than a form and use type="button" for the demonstration button. Omit scripts, external resources, action URLs and remote embeds. A missing form action alone does not reliably mean no submission: a real form may submit to the current URL. Removing submission behavior by design is clearer than relying on a warning.

Connect names, controls and instructions

Give every control a unique id and connect its visible label using for. Placeholder text is not a durable label because it disappears when typing begins and may be hard to perceive. Put help text beside the relevant control; aria-describedby can connect the control to the help text’s id. For a select, write a label that explains the requested choice and a neutral initial option. A fieldset with a legend can group related radio buttons if you use them. Explain required fields in words, not only a red asterisk. In this non-submitting mockup, required metadata documents intent but is not a substitute for a real validation workflow.

Evaluate the experience and the boundary

Click each label and confirm that the intended control receives focus or activates as appropriate. Traverse the controls with Tab and use the keyboard to make selections. Verify that a required marker is explained before users encounter it. Test the demonstration button and Enter in a text field: neither should send a request or navigate. Inspect the source for form actions, submit controls, network scripts and external asset URLs. Record the difference between design acceptance, such as a clear label, and backend acceptance, which this mockup does not implement. Do not claim the page securely collects anything; collection is intentionally absent.

Worked example

Worked example: <label for="kit">Kit code (required)</label><input id="kit" required aria-describedby="kit-help"> Use a fictional code such as KIT-X. Do not enter a person’s name. provides a visible name and separate instruction. Place it in a section, not a submitting form. A button labeled “Preview only” with type="button" and no script is inert; explain this visibly so a learner does not mistake it for a functioning request system.

Foundation practice

Mock fields

Item 1

Name: Kit code

Type: text

Required

True

Help: Use KIT-A, KIT-B or KIT-C only.

Item 2

Name: Request category

Type: select

Choices

- Choose a category
- Practice setup
- Practice replacement

Required

True

Item 3

Name: Number of sample cards

Type: number

Min

1

Max

6

Required

True

Item 4

Name: Demo note

Type: textarea

Required

False

Help: Write only a fictional item observation; no personal information.

Boundary

Use section, not form. No script, no external assets, no network URL and no submit-type control. Include an inert Preview only button.

Foundation tasks

- Sketch labels, unique ids and help-text associations before coding.
- Test every label/control pair and the keyboard sequence.
- Capstone: hand over a non-transmitting request mockup with all four controls, explained required fields, and a source-level no-submission checklist.

01 Define an interaction specimen without creating a collection service

A form mockup can demonstrate labels, choices and help without collecting anything. In this course controls are placed in a section, not a form. The Preview only button has type="button" and no behavior. There are no scripts, external assets, network URLs or submission endpoints. These are structural boundaries, not merely a warning pasted over a live form. A real form with its action omitted can still submit to the current document, so omission of an action attribute is not our non-submission strategy.

The controls remain usable for local exploration: a learner can type a fictional code, choose an option and inspect the labels. Non-submitting does not mean every control is disabled. It means the page has no implemented request workflow. Do not add a fake success message, an email link, a storage script or a remote preview service to make the specimen feel finished. Those changes would alter both the learning objective and the data boundary.

Allowed in this course	Not part of this course
Fictional item codes	Names, emails, passwords or payment details
Visible labels and local help	Remote validation or data collection
Native text/select/number/textarea controls	Account creation or live requests
Inert type=button preview	Submit controls or form action URLs

A local static file is not a promise that every browser forgets what was typed. Browser features may retain local state or offer autofill depending on settings. Therefore use synthetic values throughout and do not claim secure collection or guaranteed deletion. The page demonstrates an interface specification. Its success is judged by clear relationships and an intact non-submission boundary, not by whether it fulfills a real request.

E01-X1 State the boundary precisely

Write a three-sentence notice for the original kit mockup: fictional input only, required markers describe the proposed design, and Preview only performs no submission. Then list four source features to exclude.

Submit: A visible notice and a source checklist.

02 Translate the field brief into a design contract

Before writing controls, describe each field’s purpose, control type, required state, permitted fictional values and help. The original kit brief specifies a text code, a category select, a number from one to six, and an optional note. Preserve those four roles. Do not substitute an email field because a template already has one, or add a requester name to make the example look more like a real ticket. The supplied brief is the authority for what belongs in the mockup.

The worked folio case has a separate field contract. It uses FOL-A or FOL-B, two print activities, a count from one to four and an optional print observation. Keep that contract separate from the original KIT codes and six-card range. A copied control can carry a stale help paragraph or max attribute even when its visible label has changed. A field matrix exposes these inconsistencies before they become a misleading interface.

Field	Control	Required?	Supplied instruction
Folio code	Text input	Yes	FOL-A or FOL-B only
Practice activity	Select	Yes	Choose an activity; Compare prints; Arrange prints
Number of prints	Number input	Yes	Whole number 1–4
Practice observation	Textarea	No	Fictional print only; no personal information

The matrix is a design specification, not proof of enforced validation. A help sentence saying FOL-A or FOL-B does not automatically prevent other strings. A min and max express native numeric constraints but do not create a backend. Keep separate columns in your review for stated intent, encoded attributes and observed behavior. This prevents a common overclaim: assuming every rule described in prose is already enforced by the browser.

E02-X1 Build the original field matrix

Using the original practice brief, write four rows with label, type, required state and choices/range/help. Identify two differences from the worked folio case.

Submit: A four-field contract and a contrast note.

03 Give each control a persistent visible name

An explicit label is a visible label element whose for value exactly matches the control’s id. It tells the reader what the control is for and establishes a programmatic relationship. Place the label immediately before its control in these single-column specimens. That keeps the visual and source order straightforward. A label that says Code while the help discusses a quantity is not merely stylistically vague; it may cause the reader to supply the wrong kind of fictional value.

```
<label for="folio-code">Folio code (required)</label>
<input id="folio-code" type="text" required
  aria-describedby="folio-code-help">
<p id="folio-code-help">Use FOL-A or FOL-B only.</p>
```

Placeholder text is not a durable substitute for this label. It can disappear after typing, often has different visual treatment, and leaves the reader dependent on memory. In these exercises the controls do not need placeholder text at all because the persistent label and help provide the purpose and example. If a placeholder is present in a broken fixture, it is a teaching defect to repair, not an unfinished authoring instruction. Keep the visible name available while the field contains a value.

A control’s id, name and visible label are different concepts. Id identifies an element within the document and connects it to label for. A name can participate in form-data submission, which this mockup does not implement and does not need. The visible label is the reader-facing wording. Do not assume that name="folio-code" repairs a missing id, or that a matching placeholder repairs a wrong for value. The explicit relationship is specifically for to id.

For a button, the text inside the button provides its visible name. It does not need an extra external label for this exercise. The four data-entry controls do need their explicit labels. This distinction keeps the checker from demanding a fifth label element for a self-named Preview button while still auditing every field.

E03-X1 Repair a placeholder-only field

A draft contains <input id="folio-code" placeholder="Folio code"> and no label. Add the correct persistent label, required state and help association from the folio contract. Explain why adding name alone is not the repair.

Submit: A complete label/control/help fragment.

04 Audit identifiers across labels, controls and help

Every id in the document must be unique, including ids on help paragraphs and section headings. A field can fail even when its own label looks correct. If two controls share folio-code, a label for folio-code no longer has an unambiguous target. If two help paragraphs share activity-help, a describedby reference is ambiguous. Count all declarations across the document, not just inputs. Copy-and-paste errors often occur in nearby explanatory text.

Use a predictable naming pattern: folio-code for the control and folio-code-help for its description. The pattern is an authoring convenience, not a requirement of HTML. It makes reviews easier because a reader can see the relationship. Do not generate identifiers from field position alone if fields might be reordered; field-2 reveals less purpose than activity. Whatever pattern you choose, for and describedby must point to the actual declared ids exactly.

Reference	Expected target	Failure example
label for=activity	One select id=activity	No element id=activity-choice
describedby=observation-help	One help paragraph	Only observation-hint exists
label for=print-count	One number input	Two controls copied with id=folio-code
section labelledby=mock-heading	One visible heading	Heading id removed during editing

Repair the mistaken declaration or reference according to the field contract, then check all affected edges. Renaming a control id requires updating its label; renaming a help id requires updating describedby. Do not add an empty element just to satisfy the missing reference. A help target should contain the intended instruction, and a label target should be the intended labelable control. Existence alone does not prove the relationship is meaningful.

E04-X1 Diagnose a copied identifier

A text input and number input both have id="folio-code". The number label uses for="print-count" and its help paragraph is id="print-count-help". State the minimal control repair and all checks to rerun.

Submit: One source change and a relationship checklist.

05 Keep names and descriptions separate but connected

A label answers what information is requested. Help answers how to provide it or what limits apply. Combining every instruction into a long label can make the control’s name cumbersome, while leaving instructions far away makes them easy to miss. In these mockups use a concise label and a nearby help paragraph. Connect the paragraph with aria-describedby so the relationship is available programmatically as well as visually.

```
<label for="print-count">Number of prints (required)</label>
<input id="print-count" type="number" required
  min="1" max="4" step="1" aria-describedby="print-count-help">
<p id="print-count-help">Use a whole number from 1 to 4.</p>
```

The describedby value contains an id, not the actual help sentence and not a file path. If multiple separate descriptions are genuinely needed, the value can list multiple ids separated by spaces. Every referenced id must exist exactly once and contain relevant text. Do not point every field at the same long warning paragraph when each also needs a field-specific range or choice instruction. The general practice notice belongs before the fields; the specific rule belongs with the relevant field.

A visible help paragraph without a reference is still readable visually, but it does not fulfill the explicit association requirement of this exercise. Conversely, a describedby reference to nonexistent text creates no useful explanation. Audit both the declaration and content. A code field described by the number-range paragraph technically points somewhere, but it describes the wrong rule. This is the help-text equivalent of a correctly linked navigation item reaching the wrong topic.

Do not add aria-label with different wording to a control that already has a correct visible label simply to make a checker happy. Conflicting naming strategies can create a mismatch between what is seen and what is exposed as the control’s name. For this simple specimen use one clear explicit label and one field-specific description.

E05-X1 Repair a wrong-topic description

The folio-code input has aria-describedby="print-count-help". Both ids exist. Explain why a reference-existence check passes but the field remains wrong, then write the correct association and help.

Submit: A corrected fragment and a semantic check.

06 Express required intent in words and attributes

Required status should be understandable without relying on a red asterisk. In this course append required or optional to each visible label and explain before the controls that these words describe the proposed design. Add the native required attribute to the three required controls. Leave it absent from the optional observation. This combination communicates the intent in readable text and encodes the corresponding control metadata.

Required is a boolean HTML attribute: its presence represents true. Writing required="false" still leaves the attribute present and does not make the field optional. To make the note optional, remove required entirely. This is a common copy-and-paste defect when authors treat all attributes like ordinary key/value configuration. The visible label and the attribute must agree, or a reader receives conflicting instructions.

```
<!-- Required field -->
<label for="activity">Practice activity (required)</label>
<select id="activity" required aria-describedby="activity-help">
  <option value="">Choose an activity</option>
  <option value="compare">Compare prints</option>
  <option value="arrange">Arrange prints</option>
</select>
<!-- Optional field: no required attribute -->
<label for="observation">Practice observation (optional)</label>
<textarea id="observation" rows="3"
  aria-describedby="observation-help"></textarea>
```

The mockup has no submission path and no validation workflow. Required metadata may describe a native validity constraint, but pressing Preview only does not submit or validate a request. Do not write “the system rejects incomplete requests” when there is no system implementing that process. A design specification can say what a future real workflow would require, but this course stops at the local specimen and must not claim backend acceptance.

E06-X1 Resolve conflicting optional state

A textarea label says optional but the element has required="false". State the actual attribute meaning, make the correct repair, and explain what Preview only does afterward.

Submit: A repaired textarea and a behavior boundary.

07 Use numeric constraints without overstating enforcement

Number of prints is a count, so use a number input with min="1", max="4" and step="1" in the folio case. The visible help states the same whole-number range. The original kit case instead uses one to six sample cards. Keep the source’s range synchronized across text and markup. A label that says cards while the help says prints is also a defect, even if both ranges happen to be numeric.

Native number controls can present steppers or numeric-oriented keyboards depending on the browser and device. Do not promise an identical interface everywhere. More importantly, min and max do not guarantee that an out-of-range value can never be typed. They describe validity constraints; browser editing behavior can vary. In this non-submitting specimen, entering 9 may remain visible even though it does not meet the intended 1–4 rule. That is not evidence of a working server validation failure because there is no server workflow.

Test value	Design intent for folio count	Interpretation
1	Within range	Lower boundary allowed

Test value	Design intent for folio count	Interpretation
4	Within range	Upper boundary allowed
0	Outside range	Below minimum
5	Outside range	Above maximum
2.5	Not a whole count	Step mismatch under this design
Empty	Missing required value	Required intent not fulfilled

Use these values only as fictional design tests. You may inspect how the browser behaves when available, but do not add scripts to enforce rules or a submit button to trigger validation. The task is to encode and explain the intended constraints and distinguish that from observed editing behavior. If a reviewer sees 5 in the control, they should consult the field contract rather than assuming the control prevented all invalid input.

E07-X1 Reconcile range text and markup

The label says Number of prints, help says 1–4, and the input has min 1 max 6 step 1. Diagnose the inconsistency. State the design classification for 0,1,4,5 and 2.5 after repair.

Submit: Corrected attributes and five classifications.

08 Make a select communicate an unchosen state

A required category select should not silently choose a real activity before the learner makes a choice. Use a neutral first option such as Choose an activity with an empty value, followed by the two supplied activities. Keep the label outside the select. The neutral option is a prompt, not a substitute for the visible field name. A reader who has chosen Compare prints still needs to know that the control represents Practice activity.

The option text communicates the choice to the reader; its value is an internal value associated with that option. In this non-submitting mockup those values are not transmitted. Use simple stable values rather than real account identifiers or private data. Do not add options that the brief did not supply. An extra Emergency replacement choice would invent a new service category and imply a workflow that does not exist.

```
<label for="activity">Practice activity (required)</label>
<select id="activity" required aria-describedby="activity-help">
  <option value="">Choose an activity</option>
  <option value="compare">Compare prints</option>
  <option value="arrange">Arrange prints</option>
</select>
<p id="activity-help">Choose one fictional activity.</p>
```

A first option with an empty value supports the intended unchosen state for this single-choice required select, but the Preview button still does not run a submission workflow. Do not make the neutral option look like a successful choice in the test ledger. Record initial state, chosen state and help text separately. Keyboard selection behavior can vary in details, so test the

actual control rather than writing a universal sequence of keys that supposedly behaves identically everywhere.

E08-X1 Repair a preselected category

A select begins with Compare prints selected and has no neutral option. Add the required initial prompt and exact two choices, retain the explicit label, and explain why a selected choice does not remove the need for a label.

Submit: A corrected select fragment and naming explanation.

09 Keep optional notes bounded and genuinely optional

An optional textarea can demonstrate longer input without asking for personal information. The folio brief requests only a fictional print observation. Its help should say exactly that and explicitly exclude personal information. Avoid examples that resemble names, addresses, account numbers or confidential incidents. A suitable sample is “The fictional blue print has a diagonal stripe.” The sample concerns the supplied object, not a real person or organization.

Optional means the learner may leave the note empty in the proposed design. Do not add required to the textarea or describe a blank note as an error. The control should still have a persistent label and help association. Optional fields are not exempt from naming requirements. A blank control with no label is not self-explanatory simply because it is not mandatory.

```
<label for="observation">Practice observation (optional)</label>
<textarea id="observation" rows="3"
  aria-describedby="observation-help"></textarea>
<p id="observation-help">
  Describe only a fictional print; no personal information.
</p>
```

Rows suggests an initial visible height; it is not a maximum number of lines of content. A textarea may contain more text and provide internal scrolling. Do not use CSS to hide the text overflow or falsely claim that all typed text is visible at once. For this course the test note can be short, but reviewers should still verify that label and help remain readable when the control grows or the viewport narrows. The type system itself is not the focus; preserving instructions is.

A reset button is unnecessary. It would introduce another action and another behavior to explain, while the exercise only requires an inert preview. If a learner wants to clear a fictional entry, they can edit it locally. Do not add storage or restore features to the specimen, and do not imply the page guarantees removal from all browser history or local state.

E09-X1 Review a note-field redesign

A proposed note field is required, asks for the requester’s email, and has rows=3 described as a three-line limit. Identify all three problems and restore the folio design.

Submit: A corrected field contract and three diagnoses.

10 Inspect the complete non-submitting worked page

The complete folio page combines the field contract, visible labels, help references and non-submitting boundary. It is supplied as assets/folio-complete.html and printed below so you can inspect every element. There is no omitted script or hidden service connection. The fields are inside a section in main, and the Preview button has an explicit type and a visible explanation of its inactivity.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Folio practice mockup</title>
  <style>
    body { margin: 1rem; font: 1rem/1.5 system-ui, sans-serif; }
    main { max-width: 40rem; }
    label { display: block; font-weight: bold; }
    input, select, textarea, button { font: inherit; }
    input, select, textarea { max-width: 100%; box-sizing: border-box; }
    .field { margin-block: 1.25rem; }
    .help { margin-block: 0.35rem; }
    :focus-visible { outline: 3px solid #18509b; outline-offset: 2px; }
  </style>
</head>
<body>
  <header><h1>Folio practice mockup</h1></header>
  <main>
    <section aria-labelledby="mock-heading">
      <h2 id="mock-heading">Local design preview</h2>
      <p>Practice only: do not enter personal information.</p>
      <p>Fields marked required are required in the proposed design.</p>
      <p>This page does not submit a request or save it to a service.</p>
      <div class="field">
        <label for="folio-code">Folio code (required)</label>
        <input id="folio-code" type="text" required
          aria-describedby="folio-code-help">
        <p class="help" id="folio-code-help">
          Use FOL-A or FOL-B only.
        </p>
      </div>
      <div class="field">
        <label for="activity">Practice activity (required)</label>
        <select id="activity" required
          aria-describedby="activity-help">
          <option value="">Choose an activity</option>
          <option value="choice-1">Compare prints</option>
          <option value="choice-2">Arrange prints</option>
        </select>
        <p class="help" id="activity-help">
          Choose one fictional activity.
        </p>
      </div>
      <div class="field">
        <label for="print-count">Number of prints (required)</label>
        <input id="print-count" type="number" required
```

```
          min="1" max="4" step="1"
          aria-describedby="print-count-help">
        <p class="help" id="print-count-help">
          Use a whole number from 1 to 4.
        </p>
      </div>
      <div class="field">
        <label for="observation">Practice observation (optional)</label>
        <textarea id="observation" rows="3"
          aria-describedby="observation-help"></textarea>
        <p class="help" id="observation-help">
          Describe only a fictional print; no personal information.
        </p>
      </div>
      <p id="preview-help">Preview only is inactive; it performs no action.</p>
      <button type="button" aria-describedby="preview-help">
        Preview only (inactive)
      </button>
    </section>
  </main>
</body>
</html>
```

The wrapper’s heading is named through aria-labelledby, while each field uses label for and describedby. These are different relationships with different target types. The section points to a heading, each label points to its control, and each description points to a help paragraph. The checker must distinguish them. It should not require the section heading to be an input or count the preview-help paragraph as another field.

The stylesheet preserves ordinary font inheritance and a visible focus outline. It does not use external resources, fixed-height note clipping or hidden labels. The source is therefore suitable for a local design inspection. Actual label-click focus, keyboard order and button behavior still need a browser test before they can be reported as observed. The code’s existence is not a substitute for that evidence.

E10-X1 Trace the complete relationship graph

For all four fields, identify label for, control id, describedby value, help id and required state. Add the section-heading relationship and preview-button type.

Submit: A four-row field graph plus two boundary checks.

11 Group fields in a logical reading and keyboard order

The field order should support the reader’s task. In the folio case, identify the folio, choose the activity, state the print count, then optionally describe an observation. That sequence is reflected in the source order. A single-column layout makes the relationship easy to inspect. Do not visually reorder fields with CSS while leaving keyboard navigation in a different sequence. For this exercise there is no reason to use positive tabindex values to force a custom order.

Group each label, control and help paragraph in a simple field div. The div has no special form semantics; it keeps the related pieces together for spacing and maintenance. A section

with a heading groups the overall mockup. If a future design uses a set of radio buttons, fieldset and legend can identify the group, but that control family is not required for these four-field tasks. Adding more control types merely to use more tags would dilute the goal of accurate label relationships.

Source order	Reason
Folio code	Identify the fictional object first
Practice activity	Choose the action context
Number of prints	Provide the bounded count
Practice observation	Optional detail after essentials
Preview only	Demonstration control after the fields

A disabled control is not the same as an inert preview workflow. Disabling every field would prevent the learner from exploring labels and keyboard entry. The required specimen uses active local controls but no submitting mechanism. Similarly, adding the inert attribute to the entire section would suppress interaction rather than model the requested experience. The word inert in this textbook describes the button’s absence of action, not an instruction to apply that HTML attribute to all controls.

E11-X1 Repair a keyboard-order workaround

A draft visually shows code/activity/count/note but uses tabindex values 4, 3, 2, 1 on those controls. Describe the simplest repair and explain why disabling all fields is not the intended solution.

Submit: A source-order decision and two distinctions.

12 Test labels and controls as observable pairs

A source audit can verify that for and id match, but label behavior should also be tested in a browser when available. Click the Folio code label and observe whether its intended text control receives focus. Repeat for the select, number input and textarea. Do not test one pair and assume the others match; a copied for value can silently connect a later label to the first field. Record the actual focused control id for each observation.

Then traverse the page with the keyboard. Check that each field is reachable in logical order and that focus is visible. Enter only synthetic values, such as FOL-A, Compare prints, 2 and a fictional note. Press the Preview button and Enter in the text field. The acceptance condition is no implemented request submission or navigation. Do not invent a “Preview successful” result, because the button intentionally performs no action. A visible inactivity note explains this to the learner.

Action	Expected design behavior	Evidence to record
Click code label	Intended text control receives focus	Observed focused id

Action	Expected design behavior	Evidence to record
Tab through fields	Logical order; visible focus	Actual sequence
Choose activity	Local selection changes	Chosen visible option
Activate Preview only	No action/submission	URL/network observation if tested
Press Enter in code	No form submission	Observed outcome in named browser

A source inspection that finds no submission mechanism is strong structural evidence for this specimen, but do not call it a network-monitor test. If you actually inspect the browser's network activity, record the tool and observed requests. Browser background traffic unrelated to the page should not be misattributed to a field action without investigation. Keep the claim bounded to the local page and tested action.

E12-X1 Design a label-click ledger

Write four label-click test rows with expected target ids for the folio page. Add keyboard order and two no-submission actions. Mark browser outcomes not tested unless you actually run them.

Submit: An actionable observation ledger.

13 Repair a complete mockup without adding a live workflow

The broken folio specimen is complete and still local, but its relationships are defective. The code field has only a paragraph and placeholder instead of a label. The activity label points to a nonexistent id. The number input duplicates the code id and keeps the wrong maximum from another case. The optional note points to missing help. The count label also omits its required text. These defects are independent; correcting one id does not restore the whole contract.

Repair a copy named folio-repaired.html. Start from the field matrix, then reconcile visible labels, declared ids, help references, required states, exact option text and numeric range. Keep the non-submission boundary unchanged. There is no need to add form, submit, JavaScript, a network endpoint or a fake confirmation. A repair should reduce defects within the authorized task, not turn a specimen into a different product.

```
<!-- Two intentional source defects, shown together. -->
<label for="activity-choice">Practice activity (required)</label>
<select id="activity" required>
  <option value="">Choose an activity</option>
  <option value="compare">Compare prints</option>
</select>
<!-- The label must target activity, not activity-choice. -->
```

Use a before/after ledger that names each wrong relationship and the exact repair. "Improved labels" is not enough. Write that for=activity-choice became for=activity, or that

the number input id became print-count. Preserve the original flawed file so the detector can demonstrate the expected failures. The final repaired file should pass the entire field contract, including facts that were not known to be faulty.

E13-X1 Repair the full folio specimen

Use folio-fields.json and folio-broken.html. Restore all four labels, unique ids, help associations, required wording and the 1–4 range. Keep the preview inert and all source boundaries unchanged.

Submit: Complete repaired HTML and categorized before/after ledger.

14 Audit the non-submission boundary as well as labels

A specimen can have excellent labels and still violate the task if a form or script transmits the values. The boundary audit therefore runs independently of the association audit. Reject form, script, iframe, object, embed and external resource links in the supplied files. Inspect attributes such as action, formaction and event handlers, not just visible text. A harmless-looking button can acquire behavior through an onclick attribute even when there is no separate script element.

Our fixtures intentionally use only a small set of local elements and inline CSS. The checker rejects URL-bearing external resources, CSS imports and url() references. It also checks every button has type="button" and no event handler. No control is associated with a form elsewhere through a form attribute. A file can be non-submitting by design while still allowing local typing; the checker should not mistake ordinary input elements for a network operation.

Source feature	Why inspect it
form or form association	Could establish submission behavior
button without explicit type	May acquire submit behavior in a form context
action or formaction	Can specify a submission destination
script or on-event attribute	Can implement transmission or navigation
External resource URL	Breaks the self-contained local boundary
CSS import or url()	Can load external resources despite no script

Do not provide live network test endpoints in a broken fixture. A teaching asset can describe a forbidden pattern as text without creating an executable collecting form. The supplied broken folio page contains association and range defects but retains the local-only boundary so it is safe to inspect as a design exercise. The absence of a transmitting mechanism is not a claim that a real data-collection service has been secured; no such service exists here.

E14-X1 Review a proposed feature addition

A reviewer suggests adding form with no action and changing Preview to a default button so required fields can be tested. Explain why this changes the boundary and propose a within-scope alternative.

Submit: A rejection with a safe test alternative.

15 Distinguish design acceptance from completed validation

A design acceptance record asks whether the intended field names, choices, required states and help are present and coherent. A validation record asks whether a working process accepts or rejects values according to its rules. This mockup provides the first kind of artifact, with some native constraint metadata. It does not implement a full validation or backend process. Keep those concepts separate in the handover so a recipient does not mistake a design specimen for a working request service.

Consider this supplied hypothetical test record: source associations pass; max 4 and step 1 are present; entering5 leaves5 visible; Preview does nothing. The correct interpretation is not “the system accepted5.” There is no request acceptance event. The browser allowed an editing state, and the inert button performed its specified no-action behavior. The field still expresses a1–4 design constraint. A future validation workflow would need additional design and testing, but adding it is outside this task.

Another hypothetical report says “no network request was observed” without recording whether the page was ever interacted with. That evidence is weaker than a named sequence of actions and observations. State which button and key actions were exercised, the local filename, browser/version and whether the URL changed. If actual browser access is unavailable, keep these observations pending and report only the source-level boundary checks that were executed.

Claim	Supported here?
All four explicit associations match	Yes, if source audit passed
Help and max attributes agree	Yes, if compared with field contract
Every invalid value is blocked while typing	No general guarantee
A real request was securely collected	No; collection is absent
Preview has no implemented action	Yes from inspected source; behavior observation separately recorded

A useful acceptance summary can be short without becoming vague. Name the four-field contract, report the static association and boundary results, then state any unperformed interaction tests. Do not bury a limitation inside a broad claim of complete accessibility. Assistive-technology testing, browser keyboard behavior and source relationships each

provide different evidence. The reader should be able to tell which layer was actually examined.

E15-X1 Rewrite an overstated status

Rewrite: “All validation and security checks passed because the fields have required and Preview does nothing.” Use the hypothetical record above and identify the missing evidence.

Submit: A bounded design acceptance statement.

16 Revise a field rule consistently across all its representations

A bounded amendment changes the folio print count from 1–4 to 1–3 and adds a third code, FOL-C. The amendment does not change the field labels, control types, required states, activity choices or non-submission boundary. Apply the new rules to a copy. Update the code help and the count’s max attribute and visible help. A change to only one representation leaves the interface contradicting itself.

The source matrix is also part of the review packet. Keep the original and revised contracts clearly named rather than silently overwriting the only input. Record each affected place: code-help text, number max, number-help text, and any test ledger’s boundary values. If the ledger still says4 is allowed after the max becomes 3, the report is stale even though the HTML has been corrected. Verification artifacts should describe the final version they accompany.

Retest the boundaries that changed:1 remains allowed,3 becomes the upper allowed count,4 is now outside the proposed range. The optional observation remains optional, and the preview remains type="button" with no action. It is easy to introduce an unrelated change while editing a copied control, so rerun uniqueness and association checks as well. A small amendment does not authorize adding a new service category or a real contact field.

E16-X1 Apply the folio amendment

Use folio-amendment.txt to produce folio-v2.html. State every changed source location and classify 1, 3, 4 under the revised design. Verify all four associations remain intact.

Submit: Revised file and change/test ledger.

17 Compare field-state variants when all references already exist

The tray-count diagnostic is a separate one-field specimen, not a replacement for the four-field capstone. Its source contract says Number of trays is required, accepts a whole count from 2 through 5, and has visible help stating that same range. Only fictional counts are used. The control remains inside a section, and Preview only is inactive. Two complete local files implement different versions of this contract. Compare them by meaning as well as by reference existence.

```
<!doctype html>
```

```
<html lang="en">
<head><meta charset="utf-8"><title>Tray count specimen A</title></head>
<body>
<main>
  <section aria-labelledby="tray-heading">
    <h1 id="tray-heading">Tray count specimen A</h1>
    <p>Practice only. Use fictional counts; no personal information.</p>
    <label for="tray-count">Number of trays (optional)</label>
    <input id="tray-count" type="number" required="false"
      min="2" max="5" step="1" aria-describedby="tray-help">
    <p id="tray-help">Use a whole number from 2 to 6.</p>
    <p>Preview only is inactive and sends no request.</p>
    <button type="button">Preview only (inactive)</button>
  </section>
</main>
</body>
</html>
```

Variant A has a real label, a unique control id and an existing help target. A narrow relationship checker can therefore pass all its for/id and describedby existence checks. Yet the visible label says optional, the required attribute is present despite its false-looking value, and help permits6 while max allows only 5. The specimen is not consistent with its contract. This is a useful counterexample to the idea that every important defect is a missing identifier.

```
<!doctype html>
<html lang="en">
<head><meta charset="utf-8"><title>Tray count specimen B</title></head>
<body>
<main>
  <section aria-labelledby="tray-heading">
    <h1 id="tray-heading">Tray count specimen B</h1>
    <p>Practice only. Use fictional counts; no personal information.</p>
    <label for="tray-count">Number of trays (required)</label>
    <input id="tray-count" type="number" required
      min="2" max="5" step="1" aria-describedby="tray-help">
    <p id="tray-help">Use a whole number from 2 to 5.</p>
    <p>Preview only is inactive and sends no request.</p>
    <button type="button">Preview only (inactive)</button>
  </section>
</main>
</body>
</html>
```

Variant B changes the visible label to required, expresses required as a plain boolean attribute and corrects the help upper bound to 5. The control id, label target, description target and numeric attributes are otherwise unchanged. Both variants remain non-submitting; neither repair introduces a form or a validation script. The comparison demonstrates that an interface can improve its stated design intent without gaining any backend behavior.

Contract dimension	Variant A	Variant B
Explicit name association	for/id match	for/id match

Contract dimension	Variant A	Variant B
Help reference exists	Yes: tray-help	Yes: tray-help
Visible required intent	Wrong: optional	Correct: required
Native required attribute	Present; value false does not negate it	Present as required
Visible range versus attributes	Help2–6 conflicts with max 5	Help2–5 matches min 2/max 5
Submission boundary	No form/script; type="button"	Same local boundary

Use the contract to classify hypothetical inputs before discussing browser behavior. Empty does not meet the required design intent. The values2 and 5 are within the range. The values1 and 6 are outside it. A value of 2.5 is not a whole count under step 1. These classifications do not assert that the browser blocks every keystroke. A user may reach a temporary editing state that does not satisfy the constraints, especially in an inert specimen with no submission event. Keep that distinction visible in the review.

The following acceptance records are supplied fictional evidence for analysis, not results from browser tests performed by this author. Record S1 reports that Variant B passed the explicit association and source-boundary checks. S2 reports that the label text and help range were reconciled with tray-state-contract.json. S3 says label clicking and keyboard traversal were not performed. S4 says no Preview or Enter interaction was observed. A reviewer summarizes these as all interaction and validation tests passed. That summary is unsupported even though the corrected source is coherent.

A defensible summary is that Variant B meets the inspected source contract and local boundary, while label-click, keyboard, Preview and Enter observations remain unperformed. Do not downgrade an unperformed test to a failure unless an actual failure was observed; equally, do not upgrade it to a pass. The next steps are concrete: open the correct final file, use fictional values, observe label focus and keyboard order, and verify the specified no-action behavior. If browser execution is unavailable, preserve the pending states rather than fabricating evidence.

This small case also sharpens repair review. Required=false should not be counted as a false native required state; its presence still expresses true. The actual contradiction is between that state and the optional visible label. Changing it to plain required makes the source clearer but does not by itself change the boolean meaning. By contrast, changing help6 to 5 repairs the reader-facing range while leaving max unchanged. A precise ledger describes these different effects rather than saying that three bugs were all fixed in the same way.

E17-X1 Compare two field-state variants

Use tray-state-contract.json and both complete specimens. Identify which checks already pass in A, which meanings conflict, and what B changes. Classify empty, 1, 2, 5, 6 and 2.5 under the contract. Rewrite the hypothetical S1–S4 acceptance verdict without claiming unperformed interactions.

Submit: A field-state comparison, six input classifications and a bounded acceptance summary.

18 Plan a new sample-sleeve mockup without borrowing stale facts

The independent case is a sample-sleeve design preview. It has four fields: Sample set code, Handling activity, Number of sleeves and Sleeve observation. Its source is assets/sleeve-fields.json. The allowed codes are SET-P and SET-Q. The activities are Inspect samples and Pack samples. The count is a whole number from 1 to 5. The observation is optional and concerns only a fictional sleeve. These choices differ from both the original kit case and the worked folio case.

Create the association plan before coding. Each field needs a distinct control id, an exact visible label, a required/optional state and a relevant help id. The three required fields should have native required metadata; the note should not. The select needs the neutral prompt Choose an activity, then the two exact choices. The number input needs min 1 max 5 step 1, with the same range in help text. Use a section container and one inert preview button.

Field	Proposed stable id	Help content
Sample set code	sample-code	Use SET-P or SET-Q only.
Handling activity	handling	Choose one fictional handling activity.
Number of sleeves	sleeve-count	Use a whole number from 1 to 5.
Sleeve observation	sleeve-note	Fictional sleeve only; no personal information.

The proposed ids are a convenient model, not the only acceptable names. An alternative scheme can pass if every declaration is unique and every reference matches the intended control or description. Exact labels, choices, ranges and boundaries are not optional preferences because they come from the source contract. This distinction lets an instructor accept equivalent implementation details without tolerating changed requirements.

E18-X1 Prepare the independent relationship plan

Read sleeve-fields.json and create a four-row matrix containing label, id, tag/type, required state, help id and exact help. Add the neutral option and two choices for the select.

Submit: A complete pre-code field specification.

19 Deliver the mockup with source and interaction evidence

Create sleeve-preview.html independently from the sample-sleeve packet. The final file must open locally without external resources and must contain no form, script, submit control or other implemented transmission path. It should expose four well-labeled controls, useful visible help, readable required/optional intent and an explicitly inactive Preview button. The instructor model is separate and should not be used as the starting point.

Submit a field-contract ledger and a source-boundary report. The ledger compares every label, control type, id relationship, help reference, required state, choice and range with the supplied source. The boundary report checks both elements and attributes that could introduce behavior. Run a detector exercise on a temporary copy: change the Handling label’s for value to a nonexistent id and duplicate a help id, confirm both are detected, then restore the correct final file. Keep the deliberately broken copy clearly identified.

If browser testing is available, record all four label-click outcomes, the keyboard order, visible focus, local select behavior, Preview activation and Enter in the text field. Use only synthetic values. Inspect the page at 320px and 1024px widths so long labels and help can be checked for clipping. If any of these observations cannot be performed, mark them not tested and explain the precise limitation. A complete static artifact and incomplete behavioral evidence can be reported honestly together.

Finish with two explanations: why required attributes do not turn this specimen into a working validation process, and why a missing form action would not have been an adequate non-submission design. These questions test the boundary that makes the exercise safe and focused. The outcome is a reviewable interface specimen, not a live request service, a privacy guarantee or a certification of universal accessibility.

E19-X1 Independent sample-sleeve capstone

Build sleeve-preview.html, field ledger, source-boundary results and intentional-defect detection evidence. Include actual interaction/width results if performed, otherwise explicit not-tested states. Keep all controls local and all inputs fictional.

Submit: Complete local mockup and concise review packet.

Assessment

Choose one best answer per question. Questions 1–10 check the foundation concepts; questions 11–20 apply the extended study units. Complete and submit the practical tasks separately. A strong quiz score does not replace correct practical work.

1. After typing a value, the field’s only identifying text disappears. Why is placeholder-only labeling insufficient?

A. It always submits the form

B. It creates a second input

C. It can disappear when the user types

D. It prevents keyboard focus

2. A label has for="kit". Which property establishes its explicit target?

A. Its name is any value

B. Its placeholder says kit

C. The input has id="kit"

D. The input is blue

3. The task forbids submission even to the current document. Which container meets the design?

A. An embedded external survey

B. A form with no action attribute

C. A form with a remote action

D. A section containing controls and an inert button

4. The Preview control must have no submit behavior. What explicit type is intended?

A. image

B. button

C. submit

D. reset attached to a remote form

5. A required field is marked only with an unexplained red star. What should the marker communicate?

A. Only a decorative icon

B. The requirement in readable text

C. A guessed server error

D. Only the color red

6. A control has aria-describedby="kit-help". What must exist for this reference?

A. The id of explanatory help text

B. A secret validation endpoint

C. The label’s font size

D. A file extension

7. Source for/id values match. Which browser observation checks the explicit label behavior?

A. Change the browser language

B. Click the label and observe its control

C. Count characters in CSS

D. Upload the page

8. You need a sample input for the local specimen. Which data fits the boundary?

A. A payment-card number

B. A real customer’s full details

C. A colleague’s password

D. KIT-A and a fictional observation

9. There is no form or script. What should pressing Enter in the text control do here?

A. The page emails the instructor

B. A remote ticket is created

C. No request is sent

D. The input is automatically published

10. The handover describes a static local mockup. Which claim exceeds the implemented scope?

A. This is a local design exercise

B. This demonstrates label associations

C. This securely collects real requests

D. This contains four mock controls

11. An optional textarea has required="false". What is the correct repair?

A. Remove the required attribute

B. Change false to 0

C. Keep it; false disables the attribute

D. Add a submit button

12. Both code and count controls declare id="folio-code". What must be repaired?

A. Remove all labels

B. Only change the visible color

C. Add a hidden duplicate label

D. Assign the count its own intended id and update/verify references

13. A code field points to an existing numeric-range help paragraph. Which check is missing?

A. Whether the description matches the field’s meaning

B. Only presence of an id

C. Only font size

D. Only HTML file extension

14. Why use an empty-value neutral first option?

A. To submit an empty request

B. To represent an unchosen state without removing the label

C. To replace required metadata

D. To create a hidden account

15. What do min/max/step establish here?

A. A backend acceptance service

B. A guarantee invalid keys cannot be typed

C. A network destination

D. Native constraint intent that must be distinguished from observed editing

16. Why is a form without action not the chosen boundary?

A. It cannot contain labels

B. It may still submit to the current document

C. It removes all help text

D. It automatically disables inputs

17. Which source feature must a boundary audit catch besides script tags?

A. Inline event handlers such as onclick

B. Label for values

C. Textarea rows

D. Ordinary paragraphs

18. What does rows="3" on a textarea mean in this course?

A. An initial visible height suggestion, not a maximum line count

B. Exactly three allowed lines

C. Three required values

D. Three submissions

19. The count maximum changes 4→3. Which set must be synchronized?

A. Only page title

B. Only button text

C. Max attribute, visible help, field contract and boundary test expectations

D. Only font size

20. Which report is accurate after static checks only?

A. All keyboard users tested

B. All security and accessibility verified

C. Every invalid value rejected by a server

D. Four associations and source boundary passed; interaction/width tests not performed

Sources and further reading

W3C WAI: Labeling controls
https://www.w3.org/WAI/tutorials/forms/labels/

W3C WAI: Form instructions
https://www.w3.org/WAI/tutorials/forms/instructions/

MDN: required attribute
https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Attributes/required

MDN: button element
https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/button

MDN: number input
https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/input/number