3.1 HTML containers

Containers and parent containers

A ***container*** is any part of a web document body that has opening and closing tags. Web developers typically create many containers as a convenience to assist in organizing and formatting content. Ex: Containers can be formatted by applying styles to adjust margins, padding, horizontal and vertical alignment, and other visual presentation attributes.

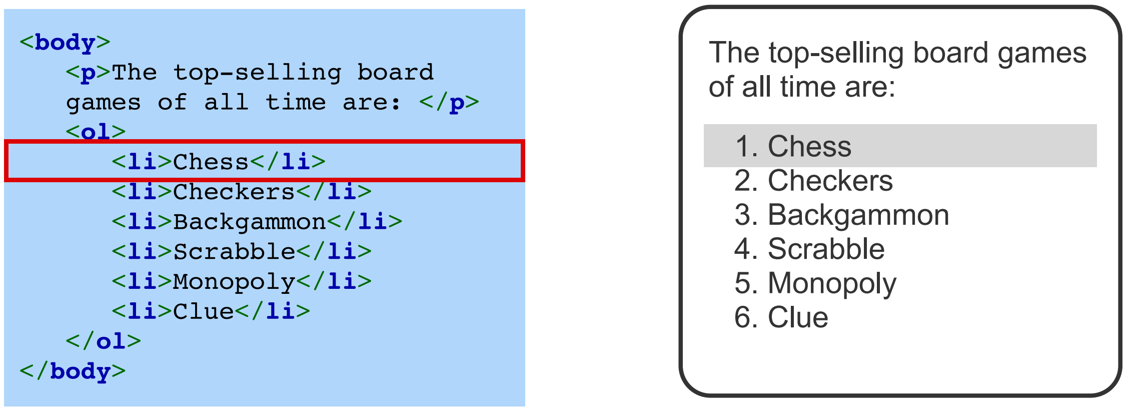
A ***parent container*** is the container in which another element resides.

Table 3.1.1: Common HTML containers.

Container	Description
<header>	Container for introductory content
<footer>	Container for content descriptive information about the webpage like author, copyright, or date modified
<address>	Container for person's or organization's contact information
<main>	Container for the document's primary content
<section>	Container for distinct parts of a document, such as a chapter
<article>	Container for self-contained content that can be reused independently, such as a news article
<nav>	Container for content relating to website navigation
<aside>	Container for content not directly related to the main topic of a document
<div>	Generic tag for creating block containers
<span>	Generic tag for creating inline containers

PARTICIPATION ACTIVITY

3.1.1: Containers and parent containers.


```
<body>
   <p>The top-selling board
   games of all time are: </p>
   <ol>
      <li>Chess</li>
      <li>Checkers</li>
      <li>Backgammon</li>
      <li>Scrabble</li>
      <li>Monopoly</li>
      <li>Clue</li>
   </ol>
</body>
```

The top-selling board games of all time are:

1. Chess
2. Checkers
3. Backgammon
4. Scrabble
5. Monopoly
6. Clue

Animation content:

The following code snippet is displayed:
```
<body>
   <p>The top−selling board games of all time are: </p>
   <ol>
     <li>Chess</li>
     <li>Checkers</li>
     <li>Backgammon</li>
     <li>Scrabble</li>
     <li>Monopoly</li>
     <li>Clue</li>
   </ol>
</body>
```
End of code snippet. A monitor displays a paragraph with the text, "The top−selling board games of all time are: " and below it is an ordered list beginning with "1. Chess", and below it is "2. Checkers", "3. Backgammon", "4. Scrabble", "5. Monopoly", and "6. Clue".

Animation captions:

1. The <p> element is the container for the text "The top-selling board games of all time are:".
2. The <ol> element is the parent container for all list items.
3. Each <li> element is the container for one list item.

PARTICIPATION ACTIVITY | 3.1.2: Parent containers.

Given the following HTML:

```
<section>
   <ol>
      <li><img src="George-Washington.jpg" alt="George Washington"></li>
      <li><img src="John-Adams.jpg" alt="John Adams"></li>
      <li><img src="Thomas Jefferson.jpg" alt="Thomas Jefferson"></li>
   </ol>
</section>
```

1) The `<ol>` element is the `<li>` element's parent container.
   - ○ True
   - ○ False

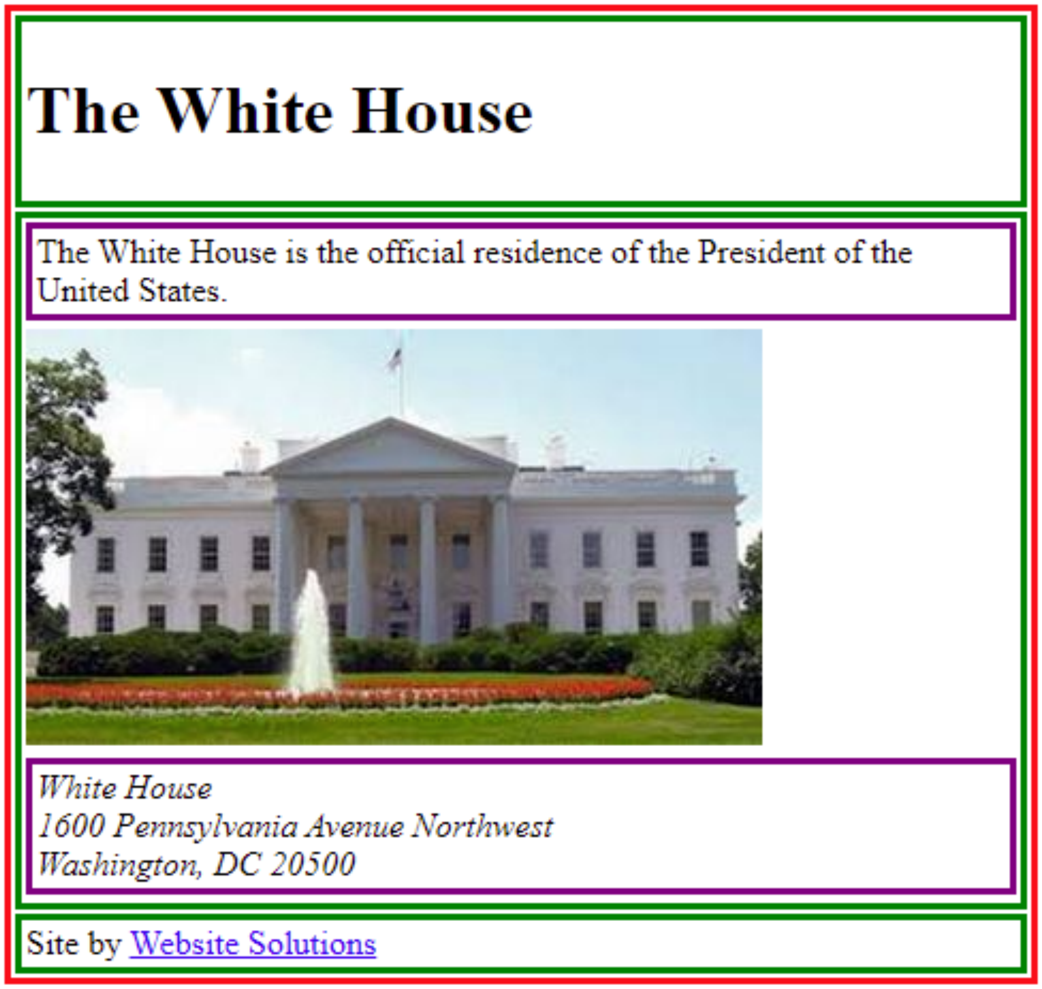
2) The `<ol>` element is the `<img>` element's parent container.
   - ○ True
   - ○ False

3) The `<ol>` element has a parent container.
   - ○ True
   - ○ False

PARTICIPATION ACTIVITY | 3.1.3: Container structure.

Use the following visible containers created with the provided HTML below to answer the questions.

```
<body>
   <header>
      <h1>The White House</h1>
   </header>
   <main>
      <p>
         The White House is the official residence of the President of
         the United States.
      </p>
      <img src="WhiteHouse.jpg" alt="White House">
      <address>
         White House<br>
         1600 Pennsylvania Avenue Northwest<br>
         Washington, DC 20500
      </address>
   </main>
   <footer>
      Site by <a href="mailto:solutions@example.com">Website
Solutions</a>
   </footer>
</body>
```



The White House

The White House is the official residence of the President of the United States.

White House
1600 Pennsylvania Avenue Northwest
Washington, DC 20500

Site by Website Solutions

1) Which tag creates the container that holds the entire webpage?

- ○ `<body>`
- ○ `<header>`
- ○ `<main>`

2) Which tag creates the container that holds the picture?

- ○ `<body>`
- ○ `<main>`
- ○ `<img>`

3) Which tag creates the container that holds the White House address text?

- ○ `<footer>`
- ○ `<main>`
- ○ `<address>`

4) Which tag creates the parent container that holds the paragraph?

- ○ `<body>`
- ○ `<main>`
- ○ `<p>`

Block elements

HTML elements can be categorized as either block or inline. A ***block element*** (sometimes called a ***block-level element***) fills the width of the element's parent container and can contain other block elements, inline elements, and text. Block elements include `<h1>`, `<table>`, and `<p>`.

Some block elements cannot be contained within certain other block elements when the semantics are unclear. Ex: The `<p>` element cannot contain another `<p>` element.

A block element is typically displayed starting and ending on new lines. Ex: The `<ol>` tag is a block tag that fills the entire width of the parent container, and each ordered list starts on a new line separate from previous and following blocks.

The `<div>` element is a generic element for creating block containers. Unlike other block elements, such as `<p>` and `<table>`, `<div>` is the only block element with no semantic meaning.

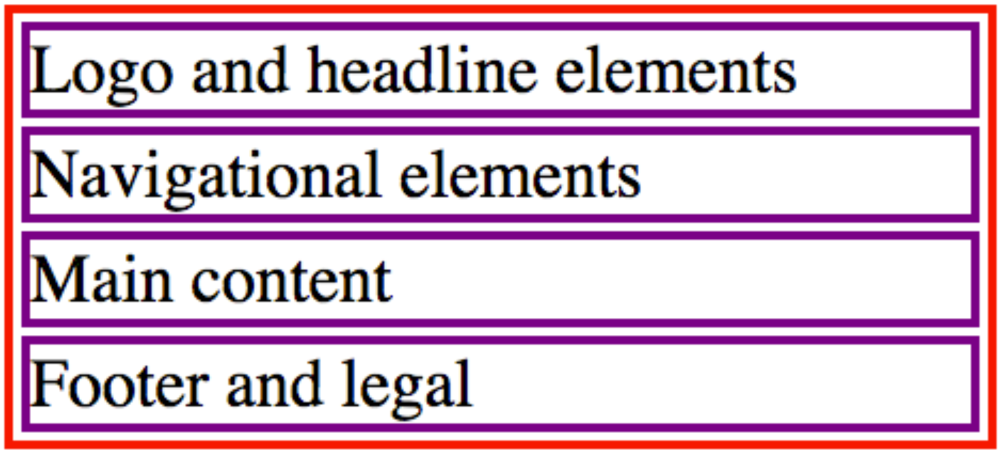
Note

By default, web browsers do not display container borders. For clarity, the container borders are visible in the following examples.

Example 3.1.1: Block elements with visible containers.

The following HTML breaks up a document into logical components using a `<div>` tag for each component. The corresponding image displays the border for each container.

```
<body>
   <div>Logo and headline elements</div>
   <div>Navigational elements</div>
   <div>Main content</div>
   <div>Footer and legal</div>
</body>
```



PARTICIPATION ACTIVITY 3.1.4: Block elements.

1) Which element is a generic block element without any implied meaning?

   - ○ `<block>`
   - ○ `<div>`
   - ○ `<section>`

2) What is the width of a block element?

   - ○ The minimum width necessary to hold the block's contents.
   - ○ The full width of the webpage.
   - ○ The full width of the block's parent container.

Inline elements

An ***inline element*** fills the minimum space possible in the element's parent container and can only contain text or other inline elements. Ex: The `<a>` element is an inline element that creates a hyperlink container as big as the link's internal content; a hyperlink does not fill the width or height of the link's parent paragraph.

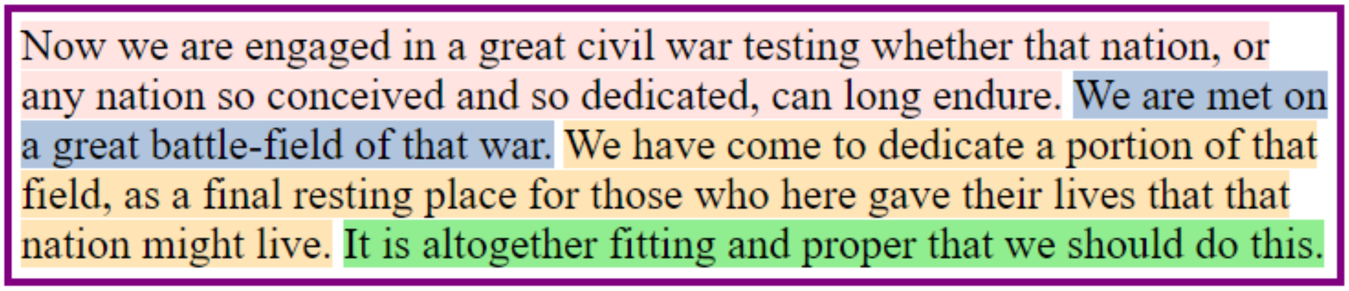
The `<span>` element is a generic inline element. Unlike other inline elements, such as `<a>` and `<em>`, the `<span>` element has no semantic meaning.

Since `<div>` and `<span>` do not have semantic meaning, `<div>` and `<span>` are used primarily for presentation and interaction purposes. *Good practice is to use tags such as `<address>` and `<article>` that convey semantic meaning when creating containers, and use `<div>` and `<span>` only when no other tags are appropriate.*

Example 3.1.2: Span tags with visible containers.

The following HTML breaks up the second paragraph from Lincoln's Gettysburg Address into sentences using a `<span>` tag for each component. The corresponding image displays a different background color for each inline container.

```
<p>
   <span>Now we are engaged in a great civil war testing whether that
   nation, or any nation so conceived and so dedicated, can long
   endure.</span>
   <span>We are met on a great battle-field of that war.</span>
   <span>We have come to dedicate a portion of that field, as a final
   resting place for those who here gave their lives that that nation
might
   live.</span>
   <span>It is altogether fitting and proper that we should do this.
</span>
</p>
```


Now we are engaged in a great civil war testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

PARTICIPATION ACTIVITY 3.1.5: Block and inline elements.

Refer to the following HTML, and determine if each element is a block or inline element.

```
<p>Click on the "swoosh" to go to Nike Headquarters. <br>
<a href="http://nike.com/"><img src="nike-swoosh.jpg" alt="Nike swoosh">
</a></p>
```

1) `<p>`

- block
- inline

2) `<img>`

- ○ block
- ○ inline

3) `<a>`

- ○ block
- ○ inline

4) `<br>`

- ○ block
- ○ inline

PARTICIPATION ACTIVITY 3.1.6: Hierarchy of block and inline elements.

1) What types of elements can be inside a block element?

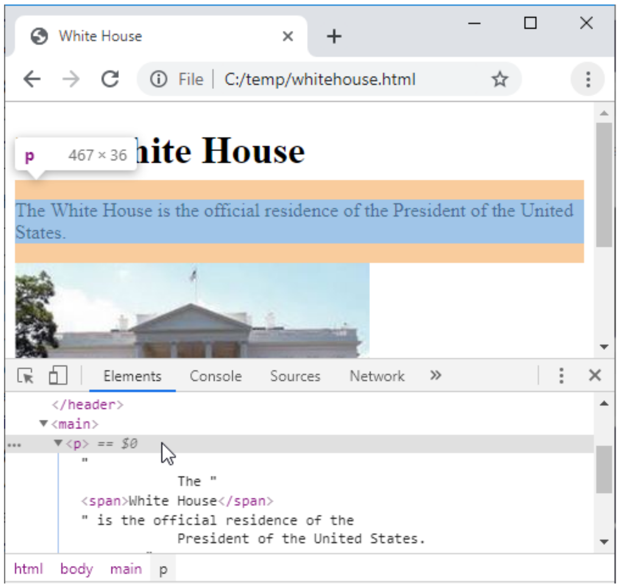
- ○ Only inline elements
- ○ Only block elements
- ○ Both inline and block elements

2) What types of elements can be inside an inline element?

- ○ Only inline elements
- ○ Only block elements
- ○ Both inline and block elements

Block vs. inline in Chrome DevTools

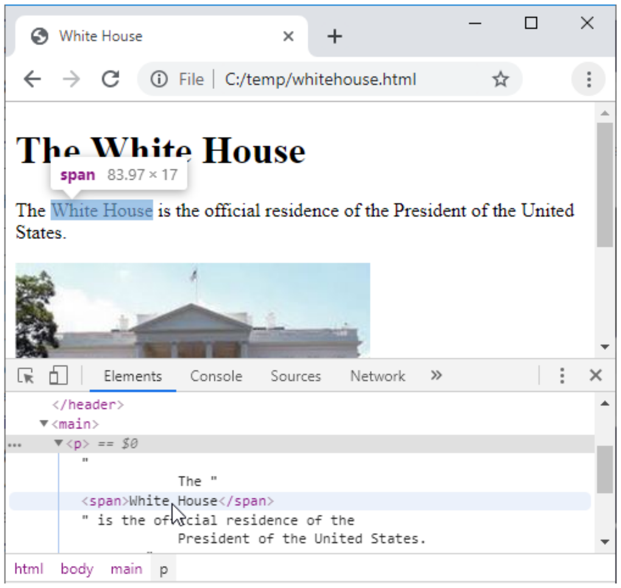
The difference between block and inline elements is visible in Chrome's DevTools. In the screenshot below-left, the mouse hovers over the `<p>` tag in the DevTools, and a rectangle appears around the entire paragraph in the webpage. The rectangle spans the browser width because `<p>` is a block element. Below-right, the mouse hovers over the `<span>` tag, but the rectangle is only as wide as the span's contents because `<span>` is an inline element.



The White House

The White House is the official residence of the President of the United States.

```
</header>
▼<main>
▼<p> == $0
  "
            The "
  <span>White House</span>
  " is the official residence of the
            President of the United States.
```



The White House

The White House is the official residence of the President of the United States.

```
</header>
▼<main>
▼<p> == $0
  "
            The "
  <span>White House</span>
  " is the official residence of the
            President of the United States.
```

PARTICIPATION ACTIVITY 3.1.7: Block vs. inline.

Change the `<div>` tags to `<span>` tags in the following HTML to see the difference between block and inline layout.

HTML CSS

```
<section>
   <div>According to Wikipedia, J.R.R. Tolkien's <u>The Lord of the Ri
   best-selling novel ever written.</div>

   <div>Wikipedia states that "only A Tale of Two Cities by Charles Di
   worldwide (over 200 million)".</div>

   <div>Tolkien's <u>The Hobbit</u> is the fourth best-selling novel c
</section>
```

Render webpage

Reset code

Your webpage

According to Wikipedia, J.R.R. Tolkien's The Lord of the Rings is the second best-selling novel ever written.

Wikipedia states that "only A Tale of Two Cities by Charles Dickens has sold more copies worldwide (over 200 million)".

Tolkien's The Hobbit is the fourth best-selling novel of all time.

▶ View solution

CHALLENGE ACTIVITY 3.1.1: HTML containers.

550544.4142762.qx3zqy7

Start

Modify the <h1> to be in a <header>, the section content to be in a <section>, and the copyright to be in a <footer>. **SHOW EXPECTED**

```

<h1>Section 1</h1>
<p>First section</p>
<p>&copy; 2012 - Jan Moorekins</p>

```

1 2 3 4

Check Next

View your last submission

3.2 Forms

<form> tag

The ***<form>*** tag allows the web browser to submit information from the user to the server. The `<form>` tag has two primary attributes:

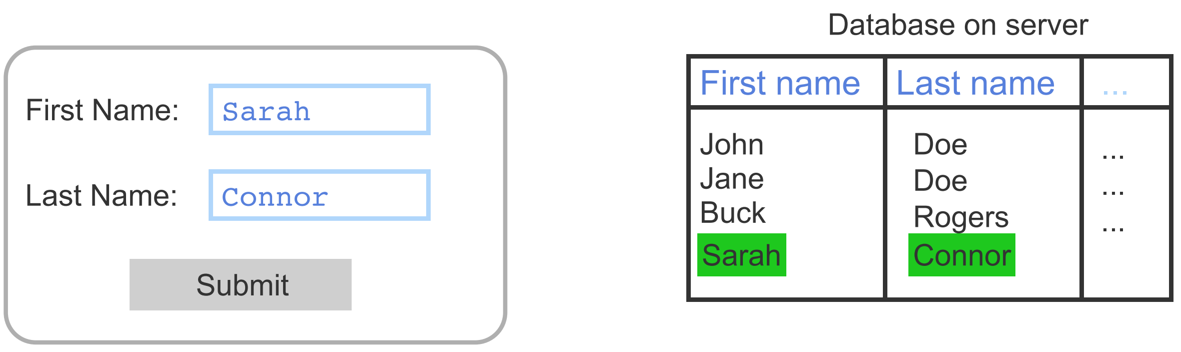
1. The ***action attribute*** indicates the URL where the form data should be sent. Typically the URL uses HTTPS so the form data is encrypted.

2. The ***method attribute*** indicates the HTTP request type the browser will use to communicate with the server. The method is either GET or POST. GET is the default method if no method is specified.

Figure 3.2.1: Partial HTML form sending data to Twitter using a secure HTTP POST request.

```
<form action="https://twitter.com/status"
method="POST">
  ...
</form>
```

PARTICIPATION ACTIVITY 3.2.1: Submitting form data to a server.



Animation content:

A form is displayed with the input boxes "First Name" and "Last Name" with a button at the bottom called "Submit." The user inputs "Sarah" for the first name, and "Connor" for the last name. When the user presses the button, the information is sent to a database on the server. The "Database on server" is also shown, with "Sarah" highlighted in the "First Name" column, and "Connor" highlighted in the "Last Name" column.

Animation captions:

1. The user enters information in the form.
2. The user clicks the submit button.
3. The browser collects the data and sends the data to the server.

PARTICIPATION ACTIVITY 3.2.2: <form> tags.

Refer to the HTML below.

```
<form action="https://google.com/">
</form>
```

1) To which server will the browser send the form data?

Check Show answer

2) Which method will the browser use to communicate with the server?

Check Show answer

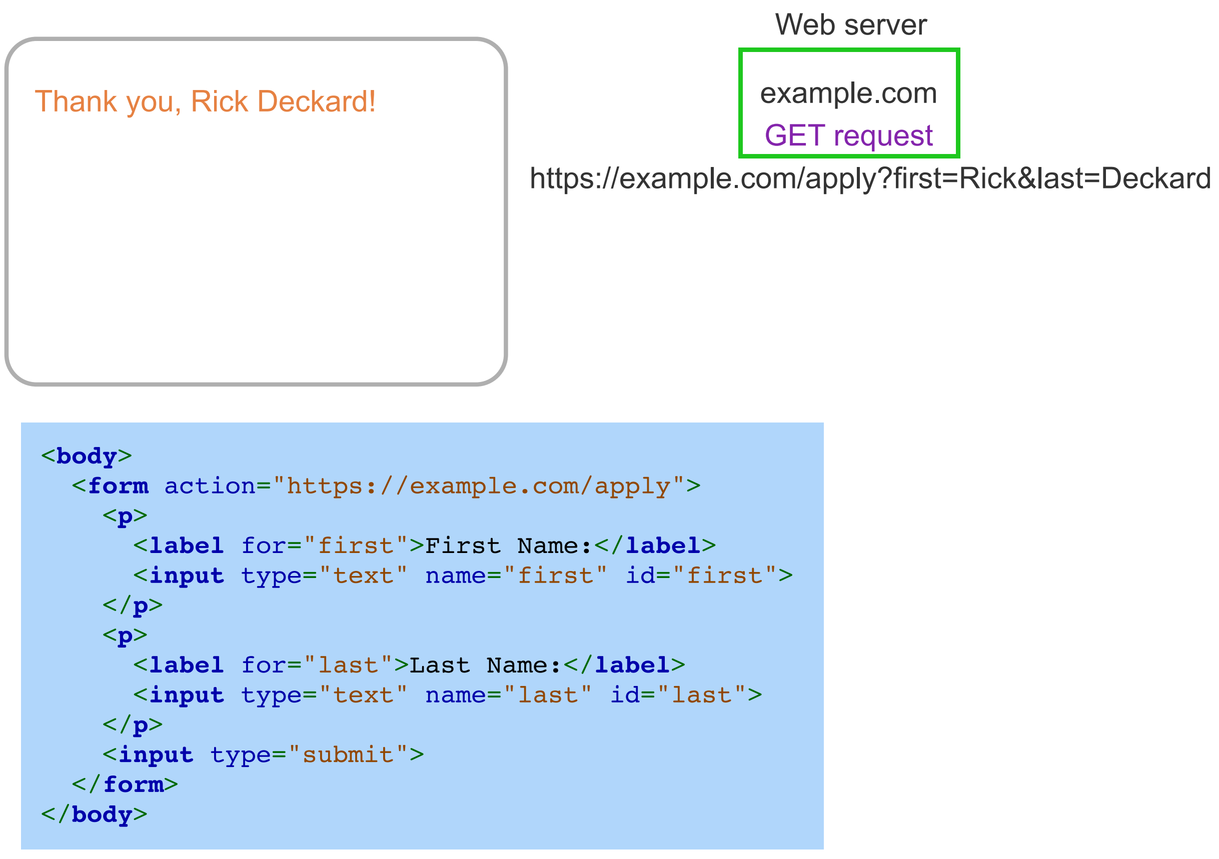
GET method

The ***GET method*** is a technique used by a web browser to submit information to a web server by altering the URL of the HTTP request. When a user clicks the submit button in a form that uses the GET method, the browser performs the following steps:

1. Collect all data from the form fields into a query string. The ***query string*** is a set of name=value pairs separated by the ampersand character (&). Each name is specified as an attribute of the HTML field, and the value is the user-entered data. Ex: The first and last field names and values in the animation below become the string: `first=Rick&last=Deckard`

2. Create a URL with the server page and name=value pairs. The URL is composed of the action attribute specified in the form, the question mark character (?), and the query string. Ex: `http://example.com/apply?first=Rick&last=Deckard`
3. Use the newly created URL to create and send an HTTP GET request.
4. Display or update the webpage using the HTTP response received from the server.

PARTICIPATION ACTIVITY | 3.2.3: Using the GET method to submit form data to a server.



```
<body>
  <form action="https://example.com/apply">
    <p>
      <label for="first">First Name:</label>
      <input type="text" name="first" id="first">
    </p>
    <p>
      <label for="last">Last Name:</label>
      <input type="text" name="last" id="last">
    </p>
    <input type="submit">
  </form>
</body>
```

Animation content:

The following code snippet is displayed:
<body>
 <form action="https://example.com/apply">
  <label for="first">First Name:</label>
  <input type="text" name="first" id="first">

```
    <label for="last">Last Name:</label>
    <input type="text" name="last" id="last">
    <input type="submit">
  </form>
</body>
```

End of code snippet. A form is displayed with the inputs "First Name" and "Last Name" and a button called "Submit." The first name is filled in with "Rick" and the last name is filled in with "Deckard." When the submit button is clicked, the first and last name are used to create the URL for example.com with the first=Rick and last=Deckard appended onto the end. The browser then uses a GET request with the created URL to submit the form data to the web server. Then the web server responds with a new page that shows one line "Thank you, Rick Deckard!"

Animation captions:

1. The user enters information in the form and clicks the submit button.
2. The browser creates a query string of name=value pairs for each form field, separated by &.
3. The query string is appended to the URL from the form action.
4. The browser uses the new URL to submit the form data to the web server.
5. The web server responds with a new webpage.

Warning

Because submitting form data with the GET method places the form data in the URL, the data is visible to anyone who sees the URL. The GET method should not be used to submit private information like phone numbers, credit card information, etc.

PARTICIPATION ACTIVITY 3.2.4: GET method.

Refer to the form below.

```
<form method="GET" action="http://example.org/">
   <input type="text" name="item">
   <input type="text" name="price">
   <input type="submit">
</form>
```

1) What is the query string if the user submits "bananas" and "2.50" in the text boxes?

- ○ `bananas=2.50`
- ○ `method=bananas&action=2.50`
- ○ `item=bananas&price=2.50`

2) What URL does the browser request if the user submits "bananas" and "2.50" in the text boxes?

- ○ http://example.org/
- ○ http://example.org/item=bananas&price=2.50
- ○ http://example.org/?item=bananas&price=2.50

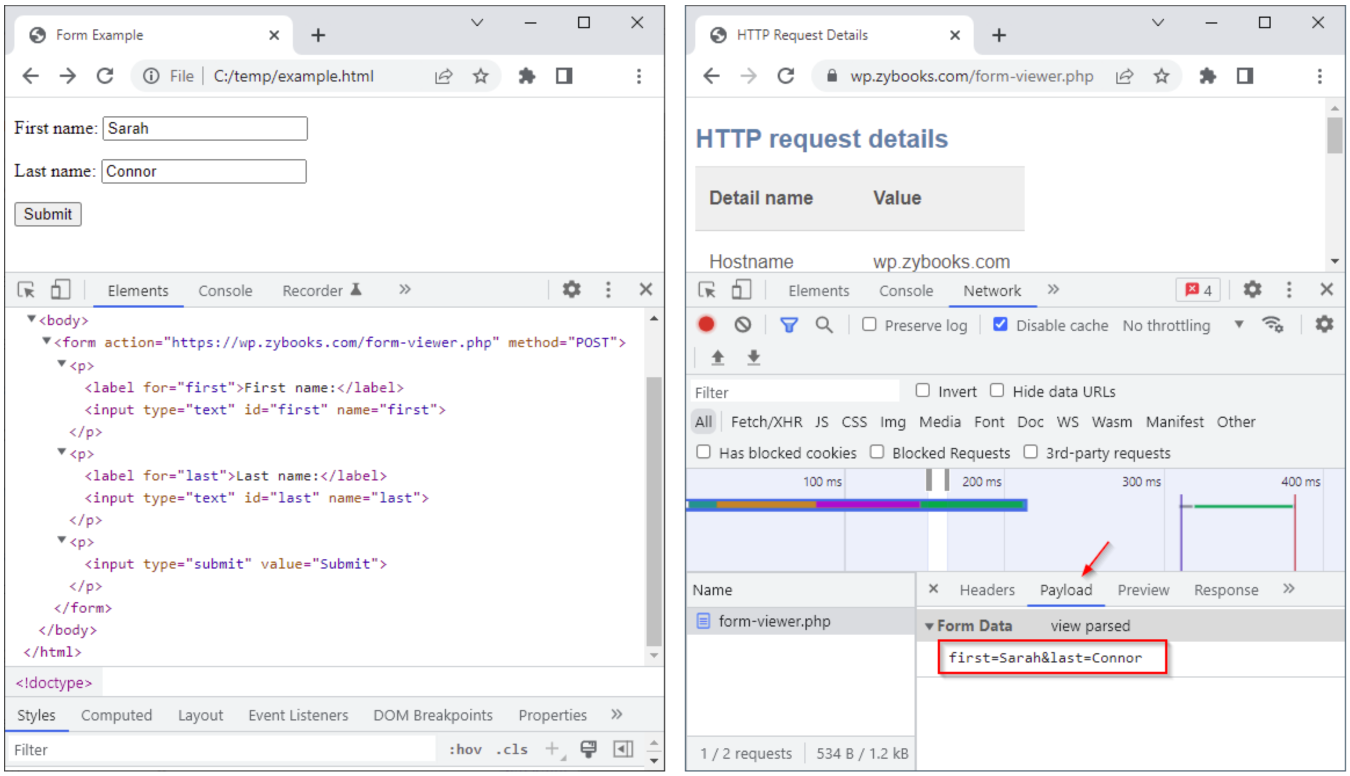
POST method

The ***POST method*** is a technique used by a web browser to submit information to a web server by sending the information in the HTTP request body. When the user clicks the submit button in a form that uses the POST method, the browser performs the following:

1. Create an HTTP POST request using the URL from the form's `action` attribute.
2. Create a query string from the form data. Ex: `first=Sarah&last=Connor`
3. Place the query string in the HTTP request message body, and send the request.
4. Display or update the webpage using the HTTP response received from the server.

The left side of the figure below shows a webpage opened in Chrome. The DevTools show the page's form, which is POSTing to a zyBooks URL. The user entered her name Sarah Connor before pressing Submit. On the right, the user has pressed Submit, and the DevTools show the details of the HTTP request. The Payload tab shows the query string created from the form data.

Figure 3.2.2: Chrome DevTools showing form data in POST request.



First name: Sarah

Last name: Connor

Submit

```
<body>
  <form action="https://wp.zybooks.com/form-viewer.php" method="POST">
    <p>
      <label for="first">First name:</label>
      <input type="text" id="first" name="first">
    </p>
    <p>
      <label for="last">Last name:</label>
      <input type="text" id="last" name="last">
    </p>
    <p>
      <input type="submit" value="Submit">
    </p>
  </form>
</body>
</html>
```

HTTP request details

Detail name	Value
Hostname	wp.zybooks.com

Name: form-viewer.php

Form Data view parsed

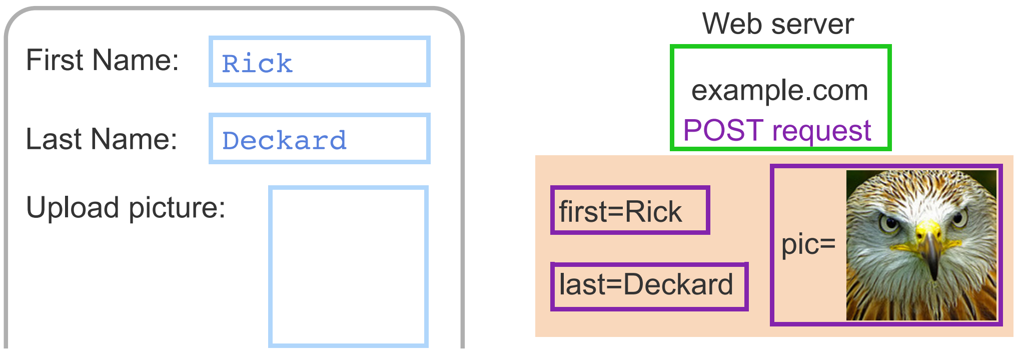
```
first=Sarah&last=Connor
```

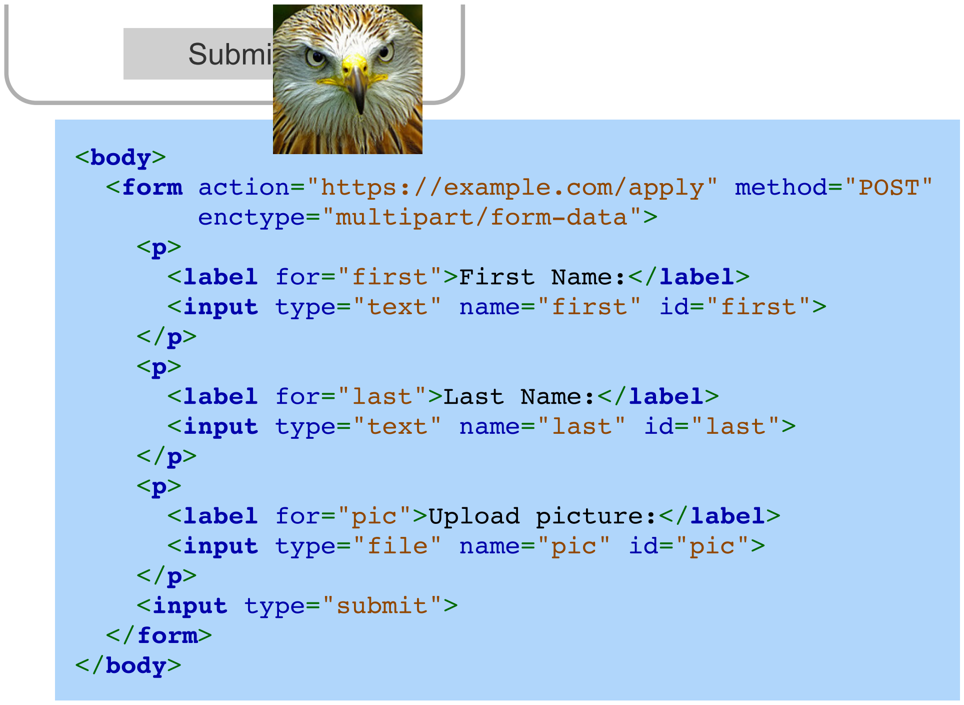
1 / 2 requests | 534 B / 1.2 kB

Form Example
C:/temp/example.html
Elements Console Recorder
Styles Computed Layout Event Listeners DOM Breakpoints Properties
HTTP Request Details
wp.zybooks.com/form-viewer.php
Elements Console Network
Preserve log Disable cache No throttling
Filter Invert Hide data URLs
All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other
Has blocked cookies Blocked Requests 3rd-party requests
100 ms 200 ms 300 ms 400 ms
Headers Payload Preview Response

If a form field contains binary data such as an image, the normal format of the query string is not sufficient to encode the binary data. To accommodate binary data, a POST request can be split into multiple parts. The `<form>` tag's ***enctype attribute*** value "multipart/form-data" indicates the web browser should split a POST request into multiple parts, where each input field is sent as a separate part of the HTTP request message.

PARTICIPATION ACTIVITY 3.2.5: Using the POST method to submit form data to a server.





```
<body>
  <form action="https://example.com/apply" method="POST"
        enctype="multipart/form-data">
    <p>
      <label for="first">First Name:</label>
      <input type="text" name="first" id="first">
    </p>
    <p>
      <label for="last">Last Name:</label>
      <input type="text" name="last" id="last">
    </p>
    <p>
      <label for="pic">Upload picture:</label>
      <input type="file" name="pic" id="pic">
    </p>
    <input type="submit">
  </form>
</body>
```

Animation content:

The following code snippet is displayed:

```
<body>
  <form action="https://example.com/apply" method="POST"
        enctype="multipart/form-data">
    <label for="first">First Name:</label>
    <input type="text" name="first" id="first">
    <label for="last">Last Name:</label>
    <input type="text" name="last" id="last">
    <label for="pic">Upload picture:</label>
    <input type="file" name="pic" id="pic">
    <input type="submit">
  </form>
</body>
```

End of code snippet. A form is displayed with the inputs "First Name", "Last Name", and "Upload Picture" with a button called "Submit." The first name is filled in with "Rick" and the last name is filled in with "Deckard", and the uploaded picture is an eagle head. When the submit button is clicked, the first name, last name and picture were added to the POST request, which is then sent to the server.

Animation captions:

1. The user enters information in the form, including a picture to upload to the server, and presses Submit.
2. The browser collects the form data into multiple parts and adds each part to the POST request.
3. The POST request is then sent to the server.

PARTICIPATION ACTIVITY 3.2.6: POST method.

Refer to the form below.

```
<form method="POST" action="http://example.org/">
   <input type="text" name="item">
   <input type="text" name="price">
   <input type="submit">
</form>
```

1) What is the query string if the user submits "oranges" and "1.99" in the text boxes?

- ○ `oranges=1.99`
- ○ `method=oranges&action=1.99`
- ○ `item=oranges&price=1.99`

2) What URL does the browser request if the user submits "oranges" and "1.99" in the text boxes?

- ○ http://example.org/
- ○ http://example.org/oranges=1.99
- ○ http://example.org/?item=oranges&price=1.99

3) What changes about the form data when `enctype="multipart/form-data"` is added to the `<form>` tag?

- ○ Nothing changes since binary data is not submitted.
- ○ Each name/value pair is separated into multiple parts.
- ○ The form data is encrypted.

Escaping form data

The &, ?, and = characters have special meaning in a query string. If a user enters characters like &, ?, =, or white space characters like space, newline, or tab, the characters must be ***escaped***, meaning the characters must be transformed into other representations. The browser rules for escaping form data are as follows:

- Space → +
- All other reserved characters, newline, and tab → %XX where XX is the ASCII hex value of the character

Ex: If "1 + 2 = ?" is entered into a textbox, the browser escapes the values producing "1+%2B+2+%3D+%3F". 2B is the ASCII hex value for "+", 3D is the ASCII value for "=", and 3F is the ASCII value for "?".

The web server ***unescapes*** the form data to determine what the original values are.

Form widgets

A ***widget*** is an interactive component (usually graphical) that the browser uses to interact with a user. Ex: Buttons, drop-down menus, and data entry fields.

The ***<input>*** tag allows the user to enter information into a webpage. The `<input>` tag cannot enclose any additional page content, and thus does not have a closing tag. The `<input>` tag has five primary attributes:

- The ***type attribute*** indicates the widget type. Common types include text, password, submit, and button.
- The ***name attribute*** names the widget and sends the widget's value when the widget's form is submitted.
- The ***id attribute*** is used to give a widget a unique identifier.
- The ***placeholder attribute*** specifies text that first appears in a text widget, typically for giving the user a hint as to the expected value.
- The ***value attribute*** specifies a default value for a widget.

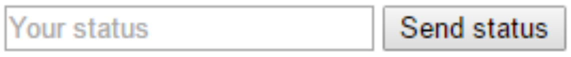
A ***text box*** widget is an `input` element with the `type` attribute of "text" that allows users to enter a single line of text.

The web browser displays a ***submit button*** widget for an `<input>` tag with the `type` attribute of "submit", which sends the associated form's data to the server when clicked. A submit button uses the `value` attribute to specify the button's text.

The HTML below asks for a message to tweet. The text box widget does not use the `value` attribute because no default tweet message makes sense. The submit button does not use the `name` attribute because the submit button's value is not needed by the web server.

Figure 3.2.3: Complete HTML form sending status to Twitter using a secure HTTP POST request.

```
<form action="https://twitter.com/status" method="POST">
  <input type="text" name="status" id="status" placeholder="Your
status">
  <input type="submit" value="Send status">
</form>
```



PARTICIPATION ACTIVITY 3.2.7: <input> attributes.

Match each `<input>` attribute to the corresponding effect.

If unable to drag and drop, refresh the page.

placeholder name type id value

	Indicates which kind of widget is displayed by the browser.
	Allows the server to identify which data came from which widget.
	Uniquely identifies the specific input tag to the browser.
	Allows the input to start with a default value.
	Provides a hint to the user about the information being requested.

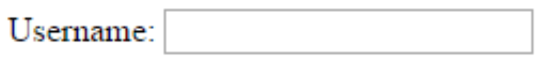
Reset

Labels and text areas

The ***<label>*** tag displays descriptive text associated with a specific widget. A label has a ***for attribute*** whose value should match the `id` attribute for the widget being labeled. Labels help people using screen readers understand what input is expected.

Figure 3.2.4: HTML for a label associated with a text box.

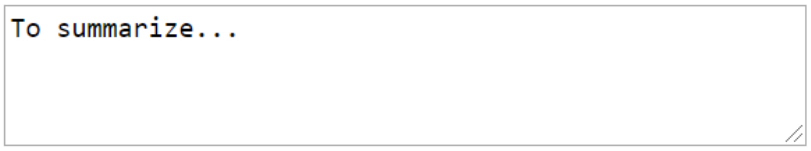
```
<label for="username">Username:
</label>
<input type="text" id="username">
```


Username:

A ***text area*** widget is an input element specified by ***<textarea>*** opening and closing tags that allows users to enter multiple lines of text. A `<textarea>` tag has optional ***rows*** and ***cols attributes*** to specify the initial size of the text area.

Figure 3.2.5: HTML for a textarea.

```
<textarea name="summary" rows="4" cols="50">To summarize...
</textarea>
```



To summarize...

PARTICIPATION ACTIVITY | 3.2.8: Text inputs.

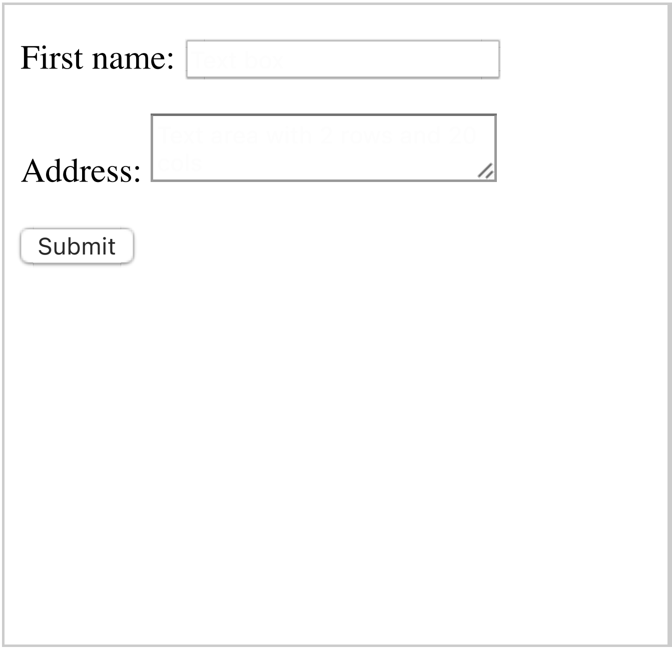
The following HTML form contains a text box, text area, and submit button. Pressing the submit button submits the form data to form-viewer.php, which displays the submitted form data.

1. Add another text box with a "last" `name`. Also change the size of the text area so that the text area has 7 rows and 50 columns of text. Make your webpage match the expected webpage.
2. Type some data into the form and press Submit. Note that the form data appears in the query string of the resulting webpage because the form uses the GET method.
3. Change the form's `method` from "GET" to "POST". Render the webpage, type some data into the form, and press Submit. The form data no longer appears in the query string of the resulting webpage although the submitted data is present.

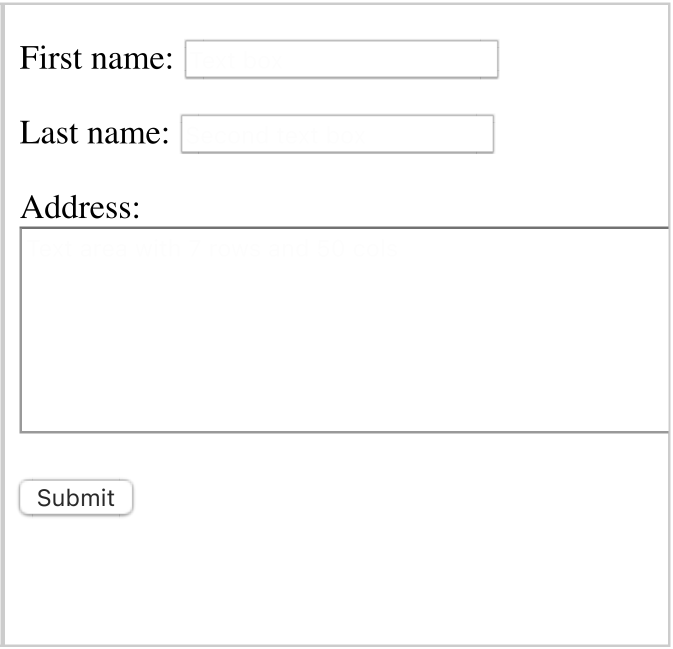
```
<form action="https://wp.zybooks.com/form-viewer.php" target="_blank'
   <p>
      <label for="first">First name:</label>
      <input type="text" id="first" name="first" placeholder="Text bo
   </p>
   <p>
      <label for="address">Address:</label>
      <textarea id="address" name="address" placeholder="Text area wi
   </p>
   <p>
      <input type="submit" value="Submit">
   </p>
</form>

```

Render webpage   Reset code

Your webpage



Expected webpage



▶ View solution

PARTICIPATION ACTIVITY 3.2.9: Text inputs.

1) Which tag creates a text box widget?

- ○ `<input>`
- ○ `<text>`
- ○ `<textarea>`

2) Which attribute must be set to create a text box widget?

- ○ `type`
- ○ no attribute
- ○ `name`
- ○ `id`

3) Which tag creates a widget capable of holding multiple lines of text?

- ○ `<input>`
- ○ `<text>`
- ○ `<textarea>`

4) Which attribute inside the `<label>` tag is used to associate the label with a widget?

- ○ `id`
- ○ `for`
- ○ `name`

CHALLENGE ACTIVITY | 3.2.1: Building forms.

550544.4142762.qx3zqy7

Start

Add a <label> with content "Last name", associated to a text <input> with name and id of lastName, and placeholder of Jones. **SHOW EXPECTED**

```
<form action="https://wp.zybooks.com/form-viewer.php" target="_blank"
   <p>

      <!-- Your solution goes here -->

   </p>
   <p>
      <input type="submit" value="Submit">
   </p>
</form>
```

1 2 3

Check Next

View your last submission

Exploring further:

- enctype attribute from W3Schools
- Sending form data from Mozilla Developer Network

3.3 Common form widgets

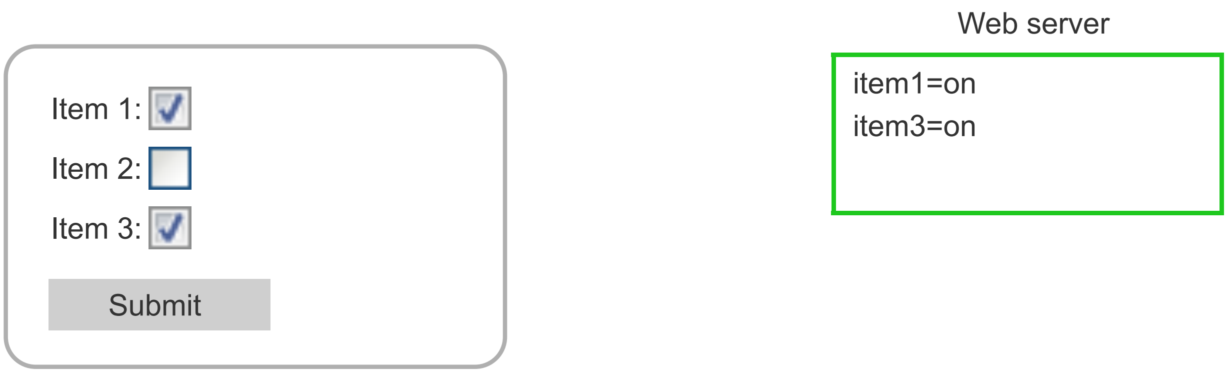
Checkbox

A ***checkbox*** is a widget for input elements with the `type` attribute of "checkbox", which allows users to check, or select, a value. A checkbox initially appears selected if the ***checked attribute*** is set. Ex: `<input type="checkbox" checked>` creates a checked checkbox. The `checked` attribute is an example of a boolean attribute. A ***boolean attribute*** is an attribute that is true when present and false when absent. No value must be assigned to a boolean attribute.

For each checkbox selected, the browser sends the checkbox's name and value to the server. If the value attribute is not specified, the default value of "on" is sent. If a checkbox is not selected, the browser does not send anything to the server. *A common error on the server is failing to record all checkboxes in the form as the browser doesn't report any values for checkboxes not selected by the user.*

Good practice is to use label elements with checkboxes so the user can click the label to check and uncheck the associated checkbox.

PARTICIPATION ACTIVITY 3.3.1: Submitting checkboxes to the server.



```
item1=on
item3=on
```

```
<form action="http://example.com/survey">
  <p>
     <label for="item1">Item 1:</label>
     <input type="checkbox" name="item1" id="item1">
  </p>
  <p>
     <label for="item2">Item 2:</label>
     <input type="checkbox" name="item2" id="item2">
  </p>
  <p>
     <label for="item3">Item 3:</label>
     <input type="checkbox" name="item3" id="item3">
  </p>
  <input type="submit">
</form>
```

Animation content:

The following code snippet is displayed:

```
<form action="http://example.com/survey">
  <p>
    <label for="item1">Item 1:</label>
    <input type="checkbox" name="item1" id="item1">
  </p>
  <p>
    <label for="item2">Item 2:</label>
    <input type="checkbox" name="item2" id="item2">
  </p>
  <p>
    <label for="item3">Item 3:</label>
    <input type="checkbox" name="item3" id="item3">
  </p>
  <input type="submit">
</form>
```

End of code snippet. A form is displayed with 3 checkboxes called "Item 1", "Item 2", and "Item 3" with a button called "Submit." Boxes 1 and 3 are selected and when the submit button is clicked, "Item 1" and "Item 3" are sent to the web server with the value on. The "Web Server" is displayed, showing "item1=on" and "item3=on."

Animation captions:

1. Each checkbox is displayed with a corresponding label.
2. The user checks Item 1 and Item 3 checkboxes and clicks the submit button.
3. For each selected checkbox, the checkbox's name and the value "on" are sent to the server.

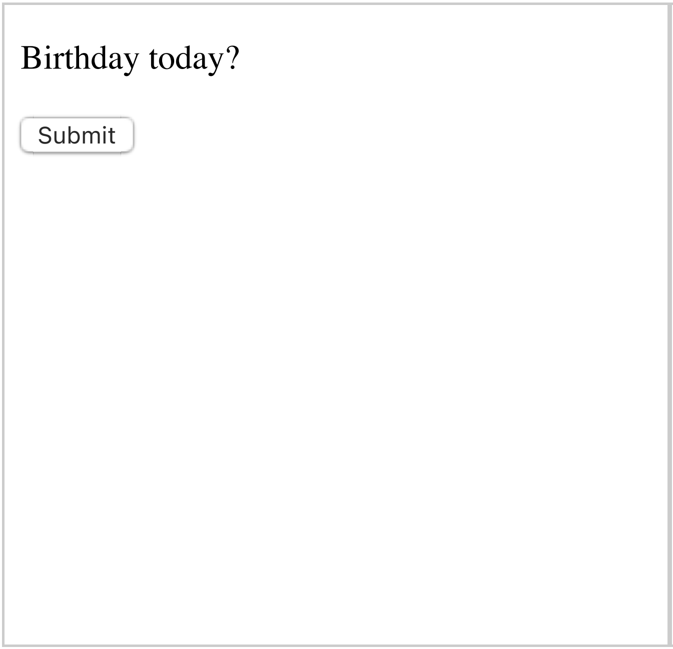
PARTICIPATION ACTIVITY 3.3.2: Checkbox inputs.

The following form contains a checkbox, which is initially checked. Click the submit button to view the form data sent to the server. Try adding another checkbox with the name "throw_party".

```
<form action="https://wp.zybooks.com/form-viewer.php" target="_blank'
   <p>
      <label for="birthday_today">Birthday today?</label>
      <input type="checkbox" id="birthday_today" name="birthday_today
   </p>

   <p>
      <input type="submit" value="Submit">
   </p>
</form>

```

Render webpage Reset code

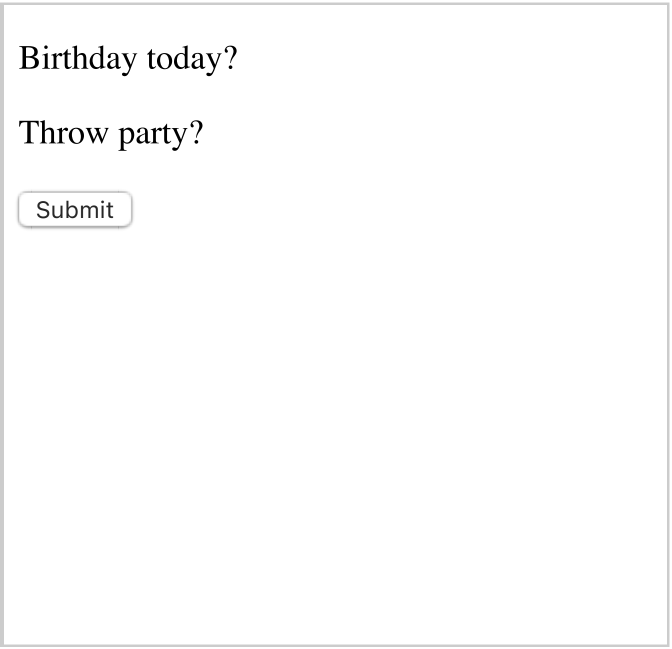
Your webpage



Birthday today?

Submit

Expected webpage



Birthday today?

Throw party?

Submit

▶ View solution

Radio button

A ***radio button*** is a widget for input elements with the `type` attribute of "radio", which allows users to select exactly one value from possibly many values. The web browser groups radio buttons together with the same `name` attribute, where each possible value in a group has an associated input. When submitting a form, the browser sends the selected radio button's `name` and `value` attribute. Ex: If the radio button `<input type="radio" name="movie" value="ET">` is selected, "movie=ET" is sent to the server.

The main difference between a radio button and checkbox is that only one radio button in a group can be selected, while any number of checkboxes can be selected.

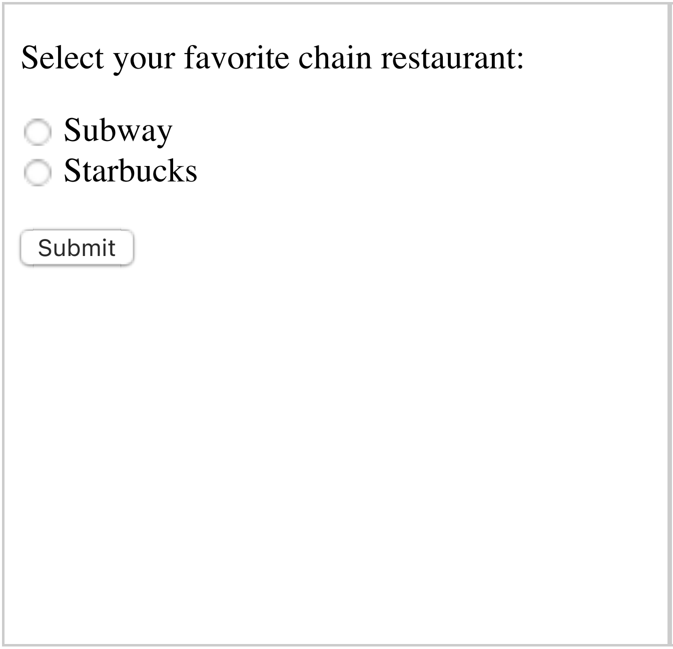
PARTICIPATION ACTIVITY 3.3.3: Radio buttons.

Try adding another radio option for the restaurants group with a value of Pizza Hut. Click the submit button to view the form data sent to server. Then, try adding another set of radio buttons to match the expected rendered HTML.

```
<form action="https://wp.zybooks.com/form-viewer.php" target="_blank'
   <p>Select your favorite chain restaurant:</p>

   <div>
      <input type="radio" name="restaurants" value="Subway" id="sub">
      <label for="sub">Subway</label>
   </div>
   <div>
      <input type="radio" name="restaurants" value="Starbucks" id="st
      <label for="starB">Starbucks</label>
   </div>


   <p>
      <input type="submit" value="Submit">
   </p>
```

Render webpage   Reset code

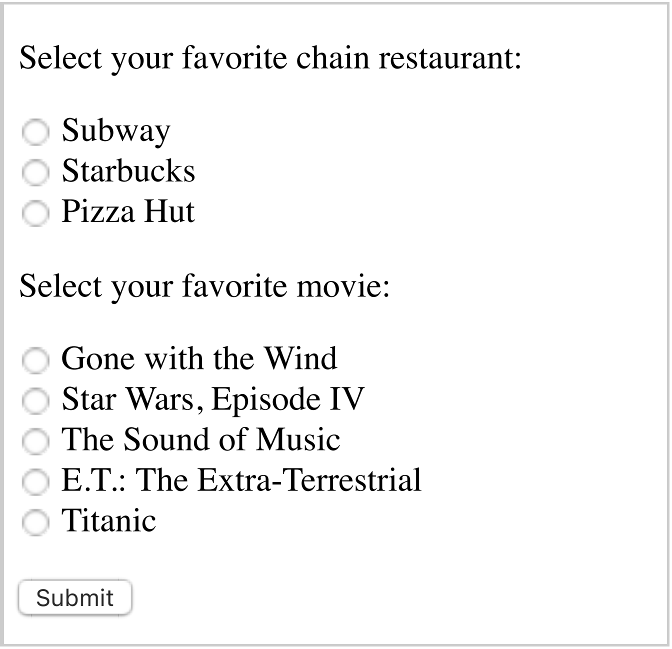
Your webpage



Select your favorite chain restaurant:

- ◯ Subway
- ◯ Starbucks

Expected webpage



Select your favorite chain restaurant:

- ◯ Subway
- ◯ Starbucks
- ◯ Pizza Hut

Select your favorite movie:

- ◯ Gone with the Wind
- ◯ Star Wars, Episode IV
- ◯ The Sound of Music
- ◯ E.T.: The Extra-Terrestrial
- ◯ Titanic

▶ View solution

PARTICIPATION ACTIVITY 3.3.4: Checkboxes and radio buttons.

Refer to the HTML below.

```html
<form action="form-processor" method="POST">
   <p>Select a flavor:</p>
   <div>
      <input type="radio" name="flavor" value="1" checked>
      <label for="choc">Chocolate</label>
   </div>
   <div>
      <input type="radio" name="flavor" value="2">
      <label for="van">Vanilla</label>
   </div>
   <div>
      <input type="radio" name="flavor" value="3">
      <label for="straw">Strawberry</label>
   </div>

   <p>Select toppings:</p>
   <div>
      <input type="checkbox" name="topping-whip">
      <label for="whip">Whipped Cream</label>
   </div>
   <div>
      <input type="checkbox" name="topping-nuts" checked>
      <label for="nuts">Nuts</label>
   </div>

   <p>
      <input type="submit" value="Order">
   </p>
</form>
```

1) If the user clicks Order without changing anything in the form, how many items will be sent to the server?

Check Show answer

2) What is the maximum number of items that can be sent to the server?

Check Show answer

3) What is the minimum number of items that will be sent to the server?

Check Show answer

4) The labels are not correctly associated with each input element. Which attribute should be added to each input element to correctly associate the labels with the input elements?

Check Show answer

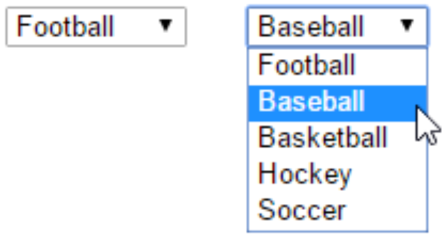
Drop-down menu

The ***<select>*** opening and closing tags create a ***drop-down menu*** (or ***drop-down list***), which allows users to select one of several predefined values. The ***<option>*** opening and closing tags create a value, or option, the user can select within a drop-down menu. When the user is not interacting with the menu, the drop-down menu usually displays the selected option.

The difference between a drop-down menu and a radio button widget is that the drop-down menu only displays the options when interacting with the user, while a radio button widget always displays all options.

Figure 3.3.1: Drop-down menu's default appearance (left) and when selecting an option (right).

```
<select name="sport">
  <option value="football">Football</option>
  <option value="baseball">Baseball</option>
  <option
value="basketball">Basketball</option>
  <option value="hockey">Hockey</option>
  <option value="soccer">Soccer</option>
</select>
```



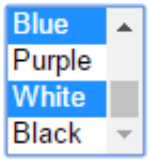
List box

A ***list box*** widget is created by specifying a size with the select element's `size` attribute. Ex: `<select size="4">` creates a list box that shows four options at a time. If the list box contains more than `size` options, the browser adds a vertical scrollbar so the user can scroll through the list of options.

The `multiple` attribute allows the user to select multiple options. On Windows, the user must hold down the control (Ctrl) button to select multiple options, and on a Mac, the user must hold down the command button. *Many users are unaware of how to choose multiple options from a list box, so good practice is to use checkboxes instead.*

Figure 3.3.2: List box that allows multiple options to be selected.

```
<select name="flagcolors" size="4"
multiple>
  <option value="red">Red</option>
  <option value="orange">Orange</option>
  <option value="yellow">Yellow</option>
  <option value="green">Green</option>
  <option value="blue">Blue</option>
  <option value="purple">Purple</option>
  <option value="white">White</option>
  <option value="black">Black</option>
</select>
```



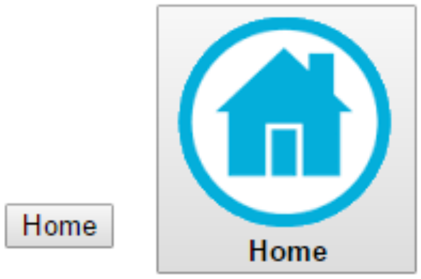
Buttons

A ***button*** widget can be created using the ***<button>*** opening and closing tags or with `<input type="button">`. The `<button>` element allows text and images to be displayed in a button, but an `<input>` button only allows text.

The `<button>` element has a `type` attribute that can be set to various values like "button" or "submit". The "button" type is typically used with JavaScript to perform an action when clicked. The "submit" type creates a submit button for a form. *If the `type` attribute is not specified, different browsers may choose different default types, so good practice is to always specify the type.*

Figure 3.3.3: HTML buttons.

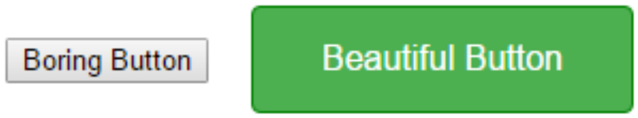
```
<input type="button" value="Home">

<button type="button">
  <img src="home.png" alt="home">
<br>
  <strong>Home</strong>
</button>
```



Styling widgets

The default look of a form's widgets may differ depending on the browser and operating system. Developers use CSS to give widgets a more uniform look or to increase the widgets' visual appeal.

The image on the left shows the default button in Chrome. The button on the right has been styled with CSS.



Password field

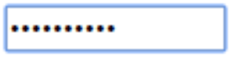
A ***password field*** is a widget for input elements with the `type` attribute of "password", which allows users to enter a password without the password contents being displayed on-screen. Web browsers usually provide facilities to remember passwords at various websites to help users.

Forms that submit passwords or any sensitive data should always submit with URLs that use HTTPS. Form data submitted with HTTP are not encrypted, but HTTPS encrypts form data.

The HTML below uses the `size` attribute to limit the password field's width and uses the `maxlength` attribute to limit the maximum number of characters the user can enter. The `size` and `maxlength` attributes can be used on text boxes as well.

Figure 3.3.4: Password field that limits the number of characters to 10.

```
<input type="password" name="secret" size="10"
maxlength="10">
```

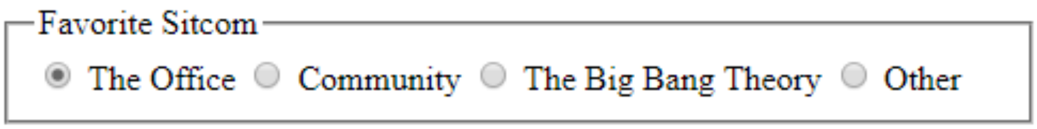


Fieldset

The ***<fieldset>*** tag groups related form widgets together and draws a box around the related widgets. The ***<legend>*** tag defines a caption for a `<fieldset>`.

Figure 3.3.5: Fieldset around related radio buttons.

```
<fieldset>
  <legend>Favorite Sitcom</legend>

  <input type="radio" name="sitcom" value="The Office" id="theOffice">
  <label for="theOffice">The Office</label>

  <input type="radio" name="sitcom" value="Community" id="community">
  <label for="community">Community</label>

  <input type="radio" name="sitcom" value="The Big Bang Theory"
id="bigBang">
  <label for="bigBang">The Big Bang Theory</label>

  <input type="radio" name="sitcom" value="Other" id="other">
  <label for="other">Other</label>
</fieldset>
```



PARTICIPATION ACTIVITY 3.3.5: Menus, buttons, and passwords.

1) A `<fieldset>` can group drop-down menus, buttons, and other widgets.

- True
- False

2) A drop-down menu only allows one option to be selected.

- True
- False

3) An `<option>` tag must have a `value` attribute.

- True
- False

4) Which element can create a button with an image?

- <button>
- <input>

5) Buttons always submit form data to a server.

- True
- False

6) Passwords from the password widget that are sent to the server using HTTP are safe from prying eyes.

- True
- False

7) Which input element attribute limits the number of characters the user can type in a text box or password field?

- ○ size
- ○ maxlength

CHALLENGE ACTIVITY | 3.3.1: Building common widgets.

550544.4142762.qx3zqy7

Start

Add a checkbox associated with each label. The first checkbox should have name and id of "first" and be initially checked. The second checkbox should have name and id of "second" and be initially not checked. **SHOW EXPECTED**

```
<form action="https://wp.zybooks.com/form-viewer.php" target="_blank"

   <p>
      <label for="first">Like cats?</label>
      <!-- TODO: Add checkbox -->
   </p>

   <p>
      <label for="second">Like dogs?</label>
      <!-- TODO: Add checkbox -->
   </p>


   <p><input type="submit" value="Submit"></p>
</form>
```

1 2 3 4

Check Next

View your last submission ⌄

Exploring further:

- HTML <input> tag from W3Schools
- HTML <select> tag from W3Schools
- HTML <button> tag from W3Schools

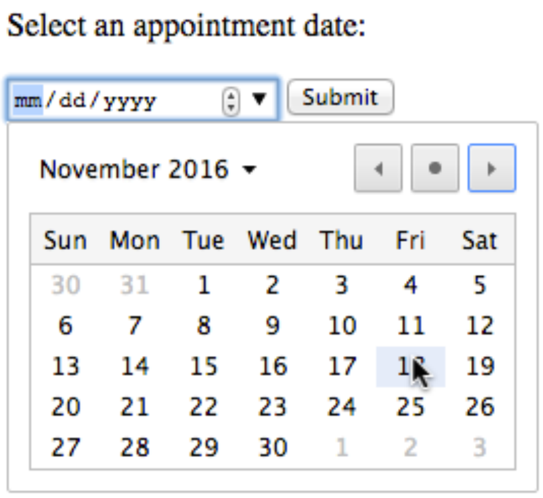
3.4 Additional form widgets

Date picker

An input ***picker*** is a widget that allows the user to interactively pick a choice using a popup or other guided selection method. The ***date picker*** is an input picker that allows the user to enter a date or choose a date from a calendar popup.

The basic syntax for the date picker is `<input type="date">`. Two common attributes for the date input are `min` (the earliest date permitted) and `max` (the latest date permitted).

Example 3.4.1: Date picker.



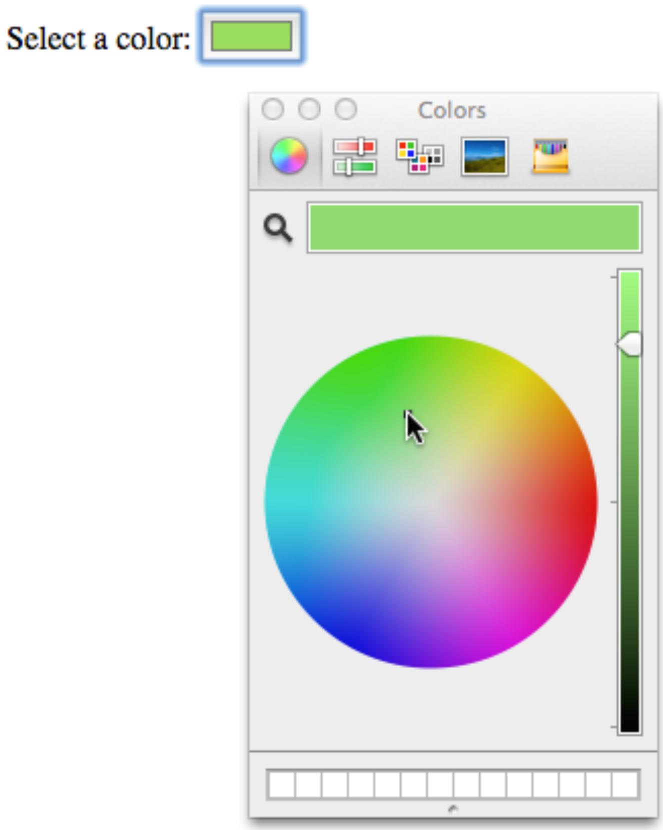
Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3

Color picker

Clicking on the ***color picker*** creates a color selector popup that helps the user explore and choose a color.

The basic syntax for the color picker is `<input type="color">`.

Example 3.4.2: Color picker.

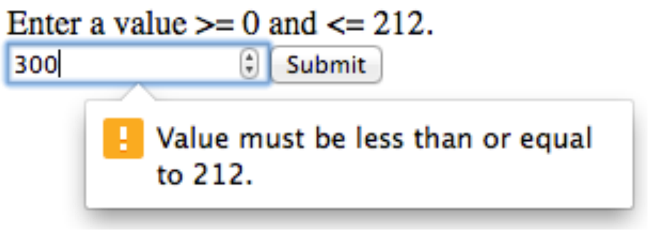


Number input

The ***number input*** ensures user input is a valid number.

The basic syntax for the number input is `<input type="number">`. The `min` and `max` attributes are commonly used with the number input.

Example 3.4.3: Number input.


Enter a value >= 0 and <= 212.

300

Value must be less than or equal to 212.

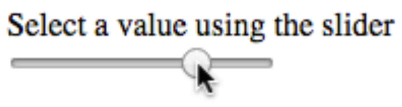
The number widget attributes are used to automatically validate input.

Range input

The ***range input*** widget allows the user to select a value by dragging a sliding control along the length of a line.

The basic syntax for the range input is `<input type="range">`. Three commonly used attributes for the range input are `min`, `max`, and `value`.

Example 3.4.4: HTML range input.

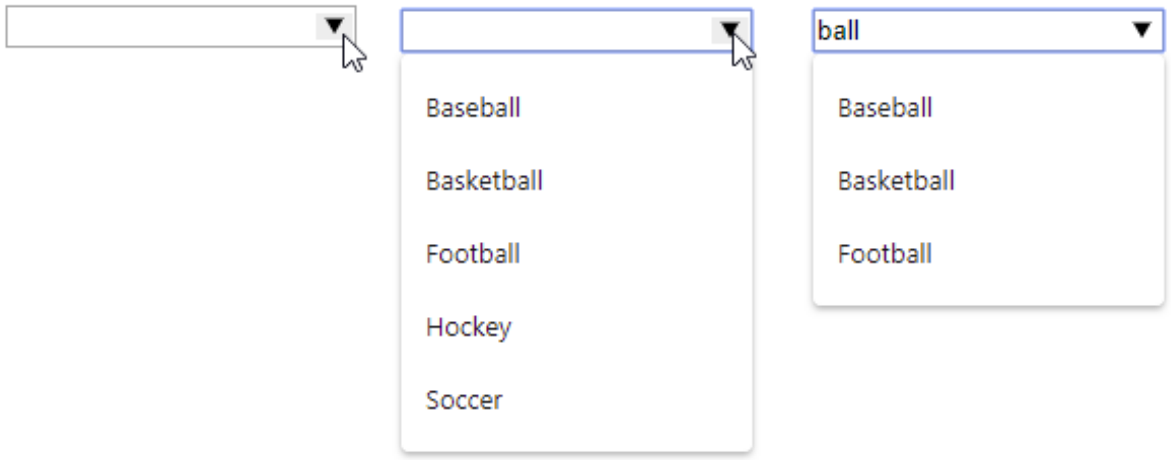


Combo box

A ***combo box*** is the combination of a text box and drop-down menu into a single widget. A combo box is created with an `<input>` element, which creates the text box, and a ***<datalist>*** element, which provides the drop-down list options.

Example 3.4.5: Combo box widget.

```
<input list="sport">

<datalist id="sport">
   <option value="Baseball">
   <option value="Basketball">
   <option value="Football">
   <option value="Hockey">
   <option value="Soccer">
</datalist>
```



Options match what the user types.

Specialized text input

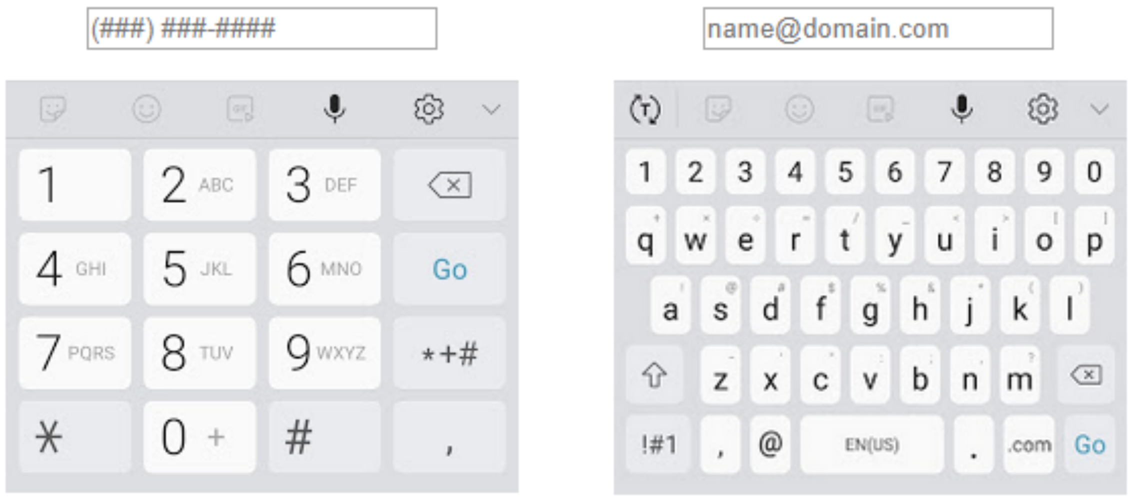
Four input types exist for entering specific types of text:

- **url** - For typing a URL
- **tel** - For typing a telephone number
- **email** - For typing an email address
- **search** - For typing search terms

Most mobile browsers display keyboards configured for entering the input type.

Example 3.4.6: Mobile browser keyboards for tel and email types.

```
<input type="tel" placeholder="(###) ###-####">

<input type="email" placeholder="name@domain.com">
```



PARTICIPATION ACTIVITY 3.4.1: Some additional widgets.

If unable to drag and drop, refresh the page.

week datetime-local month time

	Allows input of a date and time.
	Allows selecting a week and year.
	Allows selecting a month and year.
	Allows entering a time more easily.

Reset

PARTICIPATION ACTIVITY | 3.4.2: Input types.

Enter the missing values.

1) Allow the user to enter a day, month, and year.

```
<input type="        ">
```

Check Show answer

2) Allow the user to enter a month and year.

```
<input type="        ">
```

Check Show answer

3) Allow the user to select some combination of red, blue, and green.

```
<input type="        ">
```

Check Show answer

4) Allow the user to use a slider to select a number between 0 and 10.

```
<input type="        "
min="0" max="10">
```

Check Show answer

5) Ensure the user types a valid number between 0 and 10.

```
<input type="[          ]"
min="0" max="10">
```

Check Show answer

6) Create a combo box that uses the datalist with id "states".

```
<input [          ]
="states">
```

Check Show answer

7) Create a text box for entering the user's favorite website.

```
<input type="[          ]">
```

Check Show answer

Input attributes

Input attributes that restrict user input are listed in the table below.

Table 3.4.1: Input attributes.

Attribute	Description	Example
maxlength	Sets the maximum number of input characters.	`<!-- Only 4 chars max can be entered -->` `<input type="password" maxlength="4">`
minlength	Sets the minimum number of input characters.	`<!-- At least 5 chars must be entered -->` `<input type="password" minlength="5">`
max	Sets the maximum value that the input can have.	`<!-- Number may not exceed 212 -->` `<input type="number" max="212">`
min	Sets the minimum value that the input can have.	`<!-- May not be earlier than July 4, 1976 -->` `<input type="date" min="1976-07-04">`
pattern	Provides a pattern (called a regular expression) that the input must match.	`<!-- Value must be A, B, or C followed by single digit -->` `<input type="text" pattern="[ABC][0-9]">`
required	States that the input is required and must not be left empty.	`<!-- At least one character must be entered -->` `<input type="password" required>`
step	Sets the amount by which the value can change.	`<!-- Number is changed by multiples of 5 -->` `<input type="range" step="5">`

PARTICIPATION ACTIVITY 3.4.3: Input attributes.

Enter the missing attributes.

1) Ensure that the value will change by multiples of 100.

```
<input type="range"
           ="100">
```

Check Show answer

2) Ensure that the user types only one letter in the range A through F.

```
<input type="text"
           ="[A-F]">
```

Check Show answer

3) Ensure that the temperature entered will not be less than -273.16.

```
<input type="number"
           ="-273.16">
```

Check Show answer

4) Ensure that the date entered will not be greater than the end of 2025.

```
<input type="date"
           ="2025-12-31">
```

Check Show answer

5) Ensure the user enters a value in the input field.

```
<input type="number"
 >
```

Check Show answer

Fallbacks and polyfills

Not all widgets and attributes are fully supported by all browsers. Ex: Older browsers may not show a color input picker, so users must instead enter a hex string representing the red, green, and blue values of the color, like `#4268D3`, into a text input field.

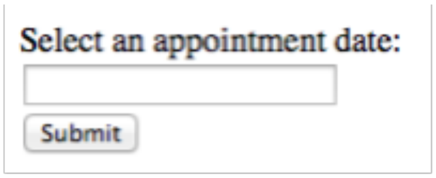
A ***fallback*** is a mechanism that allows a webpage element to function correctly even if the browser does not support a particular element. *Good practice is to implement a fallback mechanism if a particular widget is not widely supported by browsers at the time.*

A ***polyfill*** is a fallback using JavaScript code that makes certain HTML features (Ex: the date picker) work on browsers that do not natively support those features. Developers often use a JavaScript library such as Modernizr to detect which features the browser does not support, and then load one or more polyfills to provide fallback mechanisms for the non-supported features.

Example 3.4.7: A date polyfill.

When the date input element is not supported by a particular browser, the browser treats the date input box like an ordinary text input field.

```
<form>
  <p>Select an appointment date:</p>
  <p><input type="date" name="appointmentDate"></p>
  <p><input type="submit"></p>
</form>
```

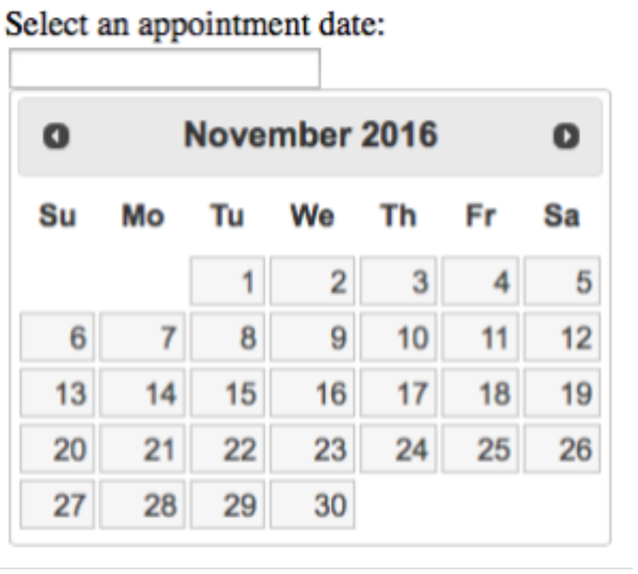


Unsupported date input

The date picker problem can be solved by detecting the problem using a library called Modernizr and adding a date picker from one of the widely-used JavaScript libraries such as jQuery.

The highlighted lines are responsible for loading the necessary Modernizr and jQuery files and for activating the polyfill if the browser does not support the date input. Notice that the original HTML form contents are not modified.

```
<link href="https://cdnjs.cloudflare.com/ajax/libs/jqueryui/1.12.0/jquery-
ui.min.css" rel="stylesheet">
<script
src="https://cdnjs.cloudflare.com/ajax/libs/modernizr/2.8.3/modernizr.min.js">
</script>
<script src="https://code.jquery.com/jquery-3.1.0.min.js"></script>
<script src="https://code.jquery.com/ui/1.12.0/jquery-ui.min.js"></script>

<script>
$(function() {
    if (!Modernizr.inputtypes['date']) {
        $('input[type=date]').datepicker({
            dateFormat: 'mm-dd-yy'
        });
    }
});
</script>

<form>
  <p>Select an appointment date:</p>
  <p><input type="date" name="appointmentDate"></p>
  <p><input type="submit"></p>
</form>
```


Select an appointment date:

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Polyfill for date input using Modernizr and the jQuery date picker

PARTICIPATION ACTIVITY 3.4.4: Fallbacks and polyfills.

1) A polyfill is code that creates a browser fallback for a feature if the web browser does not support that feature.

- True
- False

2) Browser fallbacks are only used to make new features work on old browsers.

- True
- False

3) Having users enter dates, colors, email, and numbers using the specialized widgets is preferable when an appropriate widget is available.

   ○ True

   ○ False

Exploring further:

- W3 Consortium official date input element description
- Modernizr on GitHub
- Polyfill on MDN
- Check browser feature support at caniuse.com

3.5 Audio and video

Audio element

The ***<audio>*** element plays an audio file in a webpage. The ***<source>*** element is used inside the `<audio>` tag to specify an audio file to play. Some common `<audio>` attributes include:

- `autoplay` - Boolean attribute that makes the audio begin playing automatically.
- `controls` - Boolean attribute that displays audio controls for the user to play, pause, and control the volume.
- `loop` - Boolean attribute that replays the audio upon reaching the end of the audio.
- `muted` - Boolean attribute that initially mutes the audio.

Note: Some browsers block `<audio autoplay>` from autoplaying so users are not jolted with sound when visiting a webpage.

Browser plug-ins

Prior to HTML5, developers used ***<embed>*** *and* ***<object>*** *elements to embed audio or video in a webpage. The embedded audio or video required browser plug-in to play the media. A browser* ***plug-in*** *is software that can properly read and interpret a file format that the browser cannot. Ex: The Flash Player plug-in allows a browser to play Flash files. Plug-ins have traditionally been a source of security risks for web browsers, and some browsers have stopped supporting some types of plug-ins.*

PARTICIPATION ACTIVITY 3.5.1: Audio element.

The `<audio>` tag below loads the MP3 file specified with the `<source>` tag. The text "Your browser does not support the audio element" message only displays on older browsers that do not support the `<audio>` tag.

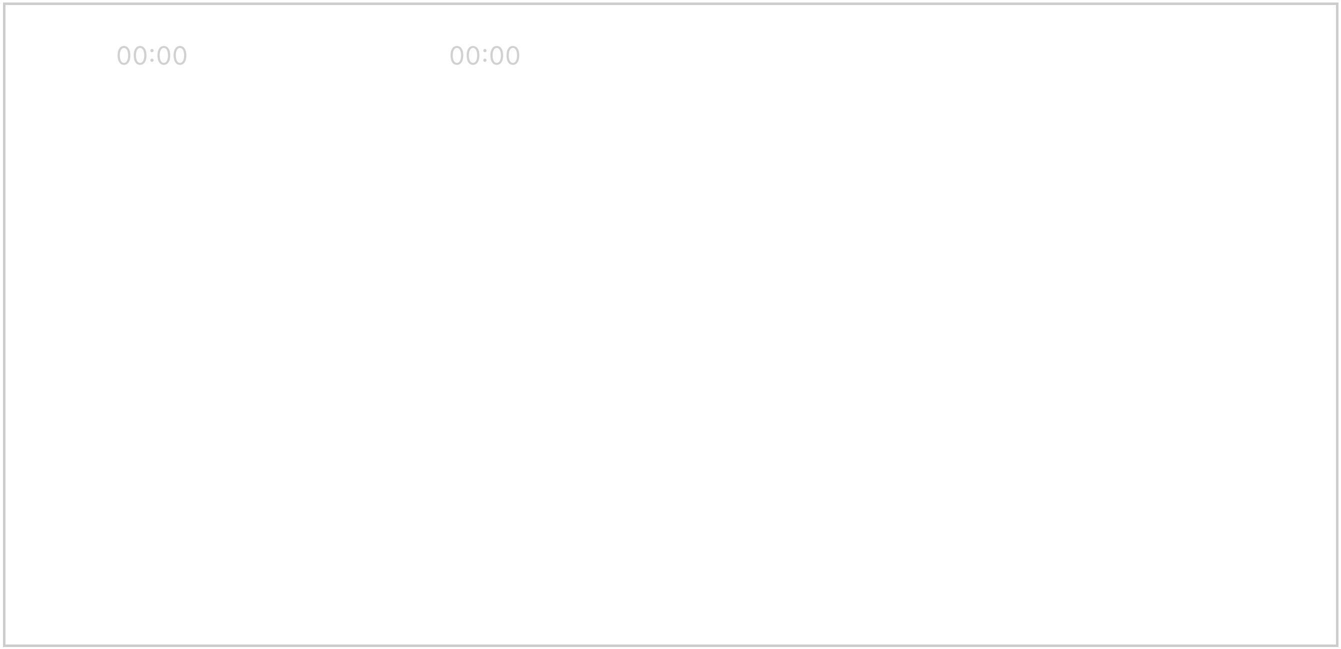
1. Press the play button to play the audio and the pause button to pause the audio.
2. Add the `autoplay` attribute to the `<audio>` tag and render the webpage. Note how the MP3 plays automatically.
3. Add the `loop` attribute to the `<audio>` tag and render the webpage. The audio will restart upon reaching the end of the audio.
4. Add the `muted` attribute to the `<audio>` tag and render the webpage. Note how the audio is initially muted.

Vivaldi meets hip hop by tyops: Creative Commons Attribution License

```
<audio controls>
  <source src="https://resources.zybooks.com/WebProgramming/tyops_viva
Your browser does not support the audio element.
</audio>
```

Render webpage Reset code

Your webpage



▶ View solution

Different web browsers support different audio formats, so multiple `<source>` tags can be used to supply alternate file formats. The MP3 and AAC formats have wide browser support.

Table 3.5.1: Common audio formats.

Format	File extension	Description
AAC (Advanced Audio Coding)	.aac	Designed to be the successor of the MP3 format with better sound quality.
MP3	.mp3	The sound portion of an MPEG file. Most popular format for music players.
Ogg Vorbis	.ogg	Open-source audio coding format developed by Xiph.Org Foundation.
Wave (Waveform Audio File Format)	.wav	Developed by Microsoft and IBM, mainly for storing uncompressed audio on Windows.

PARTICIPATION ACTIVITY 3.5.2: Audio element.

1) The audio element's audio can always be paused by the user.
   - True
   - False

2) The audio element's _____ attribute causes the audio file to play as soon as the audio file has downloaded to the browser.
   - autoplay
   - loop

3) The AAC and MP3 formats are supported by nearly all web browsers.
   - True
   - False

4) A browser that is only capable of playing MP3 audio will download which file?

```
<audio controls>
   <source src="sound.ogg">
   <source src="sound.mp3">
</audio>
```

- ○ sound.mp3
- ○ sound.ogg

Video element

The ***<video>*** element displays a video in a webpage. The `<source>` element is used in a `<video>` tag to specify the name of the video file to play. Some common `<video>` attributes include:

- `autoplay` - Boolean attribute that makes the video begin playing automatically.
- `controls` - Boolean attribute that displays video controls for the user to play, pause, and control the volume.
- `loop` - Boolean attribute that replays upon reaching the end of the video.
- `muted` - Boolean attribute that initially mutes the video.
- `width` - Specifies the pixel width of the video's display area.

Some browsers will not autoplay a video unless the video is muted. Ex: `<video autoplay muted>` will play a muted video successfully in all browsers. Muted video is less jarring to users when visiting a webpage.

PARTICIPATION ACTIVITY 3.5.3: Playing a video.

The `<video>` tag below plays the MP4 file specified by the `<source>` tag.

1. Press the play button to play the video and the pause button to pause the video.
2. Change the `width` attribute to "300" and render the webpage. The video's display area is now 300 pixels wide.
3. Add the `autoplay` attribute to the `<video>` tag and render the webpage. Note how the video plays automatically.
4. Add the `muted` attribute to the `<video>` tag and render the webpage. Note how the

video is initially muted.

New York City Subway Arriving: CC0 License from pexels.com

```
<video controls width="500">
  <source src="https://resources.zybooks.com/WebProgramming/subway.mp4
</video>

```

Render webpage Reset code

Your webpage

▶ View solution

Different web browsers support different video formats, mainly because of patent issues. Multiple `<source>` tags can be used to supply alternate file formats. MP4, WebM, and Ogg are ideal video formats for playing video on the web.

Table 3.5.2: Common video formats.

Format	File extension	Description
MPEG-4 or MP4	.m4a .mp4	Developed by the Moving Pictures Expert Group. m4a is audio only, but mp4 may contain video.
Ogg Theora	.ogg .ogv .ogm	Open video compression format developed by Xiph.Org Foundation.
WebM	.webm	Open media format developed by Mozilla, Opera, Adobe, and Google for the web.

PARTICIPATION ACTIVITY 3.5.4: Video element.

1) The size of the video displayed by a video element is configurable.
   - ○ True
   - ○ False

2) Multiple `<source>` tags can be used to specify different video formats.
   - ○ True
   - ○ False

3) All browsers autoplay unmuted video.
   - ○ True
   - ○ False

4) Playing video uses significantly more network bandwidth than playing just audio.

○ True

○ False

Controlling media playback with JavaScript

The audio and video elements can be controlled with JavaScript. Ex: JavaScript can start and stop audio, change the volume, and jump to a specific location in a video. See the HTMLMediaElement link in Exploring further below for a list of JavaScript properties and methods to control audio and video.

Showing YouTube videos

A YouTube video may be embedded in a webpage with the `<iframe>` element. The ***<iframe>*** element allows a webpage to be embedded in a rectangular area of the current webpage. The `<iframe>` element uses the `src` attribute to specify the URL of the webpage to display and the `width` and `height` attributes to define the width and height in pixels of the rectangular iframe.

The YouTube Help pages give instructions for uploading a video to YouTube and embedding a video into a webpage.

PARTICIPATION ACTIVITY 3.5.5: Embedding a YouTube video.

The webpage below embeds a YouTube video in the webpage. The YouTube URL ends with "7g7kP_Trp0g?rel=0". `7g7kP_Trp0g` is the ID assigned to the video when the video was first uploaded to YouTube, and `?rel=0` disables showing related videos when the video finishes.

1. Click the video to play the video, and the pause button to pause the video.
2. Add `&autoplay=1` to the end of the YouTube URL and render the webpage. The video will play automatically on some browsers but not in others.
3. Add the attribute `allow="autoplay"` to the `<iframe>` tag and render the

webpage. For some browsers like Chrome, the video should now play automatically.

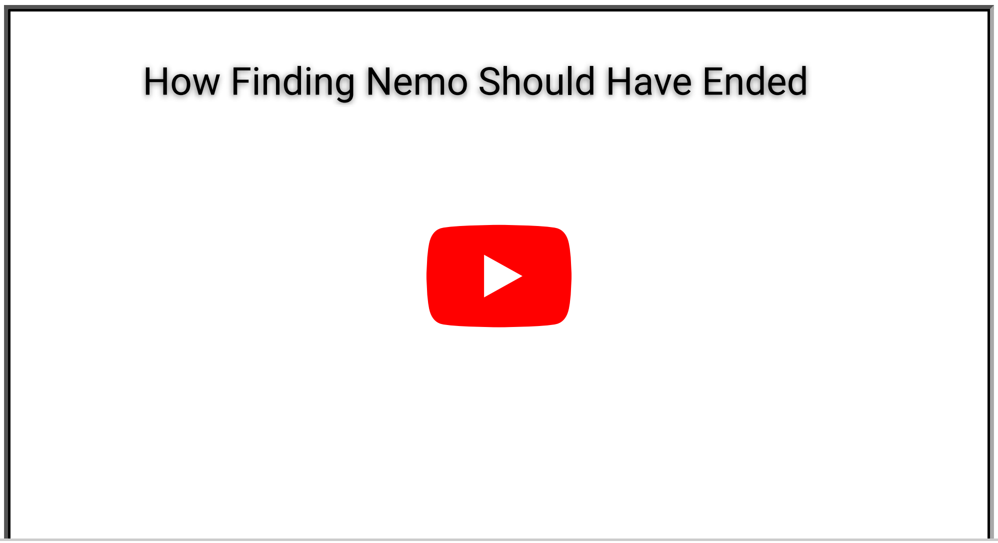
4. Change the video ID in the URL to `g4hvUvBmoaA` and render the webpage. A different video should display in the iframe.

```
<p>Watch this funny video!</p>

<iframe width="460" height="250" src="https://www.youtube.com/embed/7g
</iframe>

```

Render webpage Reset code

Your webpage

Watch this funny video!



▶ View solution

PARTICIPATION ACTIVITY 3.5.6: YouTube videos and <iframe>.

1) What `<iframe>` attribute indicates the URL of the webpage to display in the iframe?

- ○ link
- ○ src
- ○ href

2) What is missing to play the YouTube video automatically?

```
<iframe allow="autoplay" width="460"
height="250"
   src="_____"></iframe>
```

- ○ https://www.youtube.com/embed/g4hvUvBmoaA
- ○ https://www.youtube.com/embed/g4hvUvBmoaA?autoplay=0
- ○ https://www.youtube.com/embed/g4hvUvBmoaA?autoplay=1

Exploring further:

- HTML Video from W3Schools
- HTML Audio from W3Schools
- HTML Plug-ins from W3Schools
- Media formats supported by the HTML audio and video elements from MDN
- HTMLMediaElement from MDN

3.6 <script> and <style>

The ***<script>*** tag allows a webpage to include executable code, which the browser assumes to be

JavaScript unless indicated otherwise. The optional ***type attribute*** is used to indicate the content type when the content is not JavaScript. The ***src attribute*** provides the URL of an external file containing JavaScript code. If a `<script>` tag does not have the src attribute, then the JavaScript code is contained directly within the tag. The HTML below shows two ways the `<script>` tag can be used.

Example 3.6.1: Two ways of using <script> tags.

```
<script src="https://example.com/interactive_content.js">
</script>

<script type="text/javascript">
  alert("Hello, World!");
</script>
```

A common error is to forget the closing `</script>` tag when using the src attribute. Even when the JavaScript code is located in a separate external file, the closing `</script>` tag must be included.

Good practice is to use the src attribute to separate content and functionality and promote modularity. An external JavaScript file can be edited separately from an HTML file, which allows a webpage's interactive content to be updated even if the content is unchanged. A separate JavaScript file can also be reused on many webpages to provide the same functionality for different pieces of content. Additionally, when the JavaScript file is separate, a browser that doesn't understand JavaScript, such as a screenreader for blind users, can avoid downloading the JavaScript file that will not be used.

The ***<style>*** tag allows the webpage to introduce presentational directives, usually CSS. A `<style>` tag is placed in an HTML document prior to the `<body>` tag, because the style section is designed to describe the presentation of the entire document. Although only needed for non-CSS content and rarely used, the `<style>` tag has an optional type attribute that describes the content inside the tag.

Example 3.6.2: Using the <style> tag.

```
<html>
  <style>
    p {
      margin: 1em;
    }
  </style>
  <body>
   <p>The White House is the official residence of
the
      President of the United States.
    </p>
  </body>
</html>
```

Note

Unlike all other HTML tags, the contents within the `<script>` and `<style>` tags are not displayed by the browser. The `<script>` and `<style>` tags' purpose is to provide interactive functionality and presentational styling.

PARTICIPATION ACTIVITY 3.6.1: <script> and <style> tags.

If unable to drag and drop, refresh the page.

text/javascript **text/css** `<script>` `<style>`

	Tag that surrounds interactive content in an HTML document.
	Tag that surrounds presentational content in an HTML document.
	Default type for the `<script>` tag.

Default type for the `<style>` tag.

Reset

3.7 HTML developer guidelines

The following developer guidelines are helpful best practices for web development. *Although the guidelines and suggestions are not mandatory, good practice is to consistently follow the guidelines, which results in maintainable webpages and helps avoid errors.* Cleanly-organized HTML is also easier to read and understand by other developers.

Use closing tags

For elements that have a closing tag, always use the closing tag to mark the end of the element. Ex: Standard HTML does not require closing `</p>` tags because the web browser can infer the ending of a paragraph, but always including the `</p>` closing tag to end the paragraph explicitly is safer. *A common error is to expect all web browsers to add a missing closing tag at the same location in the HTML.*

Avoid self-closing tags

A ***self-closing tag*** is an open tag that also closes by putting a forward slash at the end of the tag. Ex: `<br />`. Standard HTML does not require or promote using self-closing tags for void elements. Ex: Use `<br>` not `<br />`, and use `<img ... >` not `<img ... />`.

PARTICIPATION ACTIVITY 3.7.1: Use closing tags.

Does the HTML follow the developer guidelines to always use closing tags and avoid self-closing tags?

1)
```
<ol>
   <li>Avatar
   <li>Titanic
   <li>Jurassic World
</ol>
```

Yes

No

2)
```
<p>We have updated our company
logo to
<img src="new-logo.jpg"
alt="New company logo">
from the older logo of
<img src="old-logo.jpg"
alt="Old company logo">
</p>
```

Yes

No

3)
```
<footer class="center-align">
<p>Copyright &copy; 2020</p>
```

Yes

No

4)
```
<address>
John Smith <br />
123 Main St. <br />
Anytown, USA
</address>
```

Yes

No

Use quotes for attribute values

Always use quotes around attribute values. Ex: Use `value="Start"` instead of `value=Start`. *While HTML does not require quotes around an attribute value that does not contain a space, a common error is forgetting the quotes for attribute values that contain spaces. Ex:* *`value=Start animation`* *is illegal and results in an error.*

Use double quotes

Although either double or single quotes are acceptable around HTML attribute values, good practice is to use double quotes, which in general results in more readable HTML. Ex: Use `value="Start"` instead of `value='Start'`. Inconsistent use of quotes can also lead to problems when using some web development frameworks, which assume the developer uses double quotes.

If a double quote (") needs to be placed within an attribute value, the internal double quote must be escaped. A special character such as the double quote can be ***escaped*** within an attribute value by placing the backslash character (\) before the special character, causing that character to lose any special meaning. Ex: `alt="Dwight D. \"Ike\" Eisenhower"`.

PARTICIPATION ACTIVITY 3.7.2: Identify correct use of quotes around attribute values.

Which quote guideline is violated ?

1)
```
<p id="preamble">We the
people...</p>
```
- ○ Use quotes around attribute values
- ○ Use double quotes
- ○ No guideline violations

2)
```
<input type=checkbox checked>
```
- ○ Use quotes around attribute values
- ○ Use double quotes
- ○ No guideline violations

3)
```
<input type="checkbox"
checked>
```
- ○ Use quotes around attribute values
- ○ Use double quotes
- ○ No guideline violations

4) `<input type='text' name='zip'>`

- ○ Use quotes around attribute values
- ○ Use double quotes
- ○ No guideline violations

5) `<a href="http://example.com" rel='nofollow'>`

- ○ Use quotes around attribute values
- ○ Use double quotes
- ○ No guideline violations

6) `<p class="assertion">He says he is 'innocent'.</p>`

- ○ Use quotes around attribute values
- ○ Use double quotes
- ○ No guideline violations

Use boolean attributes concisely

A ***boolean attribute*** is an attribute that is true when present and false when absent. Ex: The `checked` attribute for a checkbox widget is a boolean attribute. If the `checked` attribute is set, the checkbox is initially selected.

Older versions of HTML required all attributes to have values, so developers would use `checked="checked"` to indicate the `checked` attribute was true. Browsers continue to support the older syntax, but *good practice is to only use the attribute name for a boolean attribute without specifying a value.* Ex:
`<input type="checkbox" name="foodPreference" value="vegetarian" checked>` is the preferred way to use the checked boolean attribute.

PARTICIPATION ACTIVITY 3.7.3: Boolean attributes.

Does the HTML follow the developer guidelines to use boolean attributes concisely?

1)
```
<select>
  <option selected>Pick an
option below</option>
  <option>Lions</option>
  <option>Tigers</option>
  <option>Bears</option>
  <option>Oh, my!</option>
</select>
```

- Yes
- No

2)
```
<div hidden="hidden">
```

- Yes
- No

3)
```
<button type="submit"
disabled="">Submit</button>
```

- Yes
- No

Use lowercase for all tags and attributes

Good practice is to use lowercase for all tags and attributes. Ex: Use `<p>` and `value="Start"` instead of `<P>` and `VALUE="Start"`.

PARTICIPATION ACTIVITY 3.7.4: Use lowercase.

Does the HTML follow the developer guidelines to use lowercase for tags and attributes?

1)
```
<H1>Table of Contents</H1>
```

- Yes
- No

2)
```
<img
Src="https://apple.com/apple.jpg">
```
○ Yes

○ No

Start block elements on new lines

Each block element should start on a new line. Ex: A `<p>`, `<table>`, or `<ol>` element should always begin on a new line. To highlight the nature of block elements, inline elements should only start on a new line for readability purposes. Ex: A `<a>`, `<span>`, or `<q>` element may start anywhere on a line.

PARTICIPATION ACTIVITY 3.7.5: Start block elements on a new line.

Does the HTML follow the developer guidelines for block-level and inline elements?

1)
```
<h1>Olympics</h1><p>The IOC
organises the modern summer
and winter Olympic Games and
Youth Olympic Games held every
four years.</p>
```
○ Yes

○ No

2)
```
<h1>Olympics</h1>
<p>The
<em>IOC</em>
organises the modern Olympic
Games and Youth Olympic Games,
held in summer and winter,
every four years. The first
Summer Olympics organised by
the IOC was held in Athens,
Greece, in 1896</p>
```
○ Yes

○ No

3)
```
<label for="city">City:
</label>
<input type="text" name="city"
id="name">
```

○ Yes

○ No

Indent nested elements consistently

Nested elements should be indented at least two spaces. Indentation should be consistent throughout a file.

Table 3.7.1: Good and bad examples of consistent indentation.

Guideline	Good example	Bad example

Indenting nested elements	`<ol>`
	`  <li>Item 1</li>`
	`  <li>Item 2</li>`
	`</ol>`
		`<ol>`
		`<li>Item 1</li>`
		`<li>Item 2</li>`
		`</ol>`

Consistent indenting	`<html>`
	`  <head>`
	`    <title>My Title</title>`
	`  </head>`
	`  <body>`
	`    <p>First paragraph</p>`
	`    <p>Second paragraph</p>`
	`  </body>`
	`</html>`
		`<html>`
		`  <head>`
		`    <title>My Title</title>`
		`     </head>`
		`  <body>`
		`       <p>First paragraph</p>`
		`      <p>Second paragraph</p>`
		`  </body>`
		` </html>`

PARTICIPATION ACTIVITY 3.7.6: Indent nested elements.

Does the HTML follow the developer guidelines to consistently indent nested elements?

1)
```
The all-time top grossing
movies domestically<ol>
<li>Avatar</li>
<li>Titanic</li>
<li>Jurassic World</li>
</ol>
```

- Yes
- No

2)
```
<tr>
   <td>Hematocrit</td>
   <td>34.9-44.5%</td>
   <td>38.8-50.0%</td>
</tr>
```

- Yes
- No

3)
```
<form>
  <label for="name">Name:
</label>
     <input type="text"
id="name" name="name">
  <label
for="address">Address:</label>
     <input type="text"
id="address" name="address">
  <label for="city">City:
</label>
     <input type="text"
id="city" name="city">
</form>
```

- Yes
- No

Separate content from presentation and functionality

Good practice is to not use embedded or inline CSS and JavaScript. Ex: Instead of using `<p style="color:red;">No!</p>`, use `<p class="attention">No!</p>` and define the "attention" class in a CSS file.

Good practice is to use separate files for CSS and JavaScript instead of placing CSS and JavaScript in the HTML document. Ex: Instead of using the `<style>` element for specifying CSS rules within the HTML document, specify the CSS rules in a separate CSS file, such as theme.css, and import the file using `<link rel="stylesheet" href="theme.css">`.

PARTICIPATION ACTIVITY | 3.7.7: Separate content from presentation and functionality.

Does the HTML follow the developer guidelines to separate content from presentation and functionality?

1)
```
<html>
   <link rel="stylesheet"
href="main_style.css">
   <body>
      <p>A page with style!
</p>
   </body>
</html>
```

- Yes
- No

2)
```
<button
onclick="processData()">Process
Data</button>
```

- Yes
- No

3)
```
<html>
   <style>p { color: blue }
</style>
   <body>
      <p>Once in a blue moon!
</p>
   </body>
</html>
```

- Yes
- No

Use CSS for layout

CSS is designed for layout; the `<table>` element is designed for holding tabular data. *Good practice is to avoid using tables to manage page layout and to avoid using CSS to manage tabular data.* Using tables to manage layout obscures the content and meaning of the webpage and can cause problems when trying to control layout correctly using CSS. Using CSS to build tables requires more complex JavaScript to manage data tables in more sophisticated web applications.

Validate HTML

Good practice is to validate HTML using an HTML validator and revise the HTML to eliminate any errors or warnings. One of the big challenges in web development is to make sure that the webpages are processed correctly by as many browsers as possible. Strict adherence to the HTML Living Standard generally reduces browser errors. The W3C official validator at https://validator.w3.org is free, works well, and allows validation by providing the URL, by uploading an HTML file, or by directly entering HTML.

PARTICIPATION ACTIVITY 3.7.8: Using the validator.

Does the following HTML pass the HTML validator with no errors or warnings? Copy the HTML, click on the validator link, paste the HTML into the validator, and click the validator's Check button.

1)
```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Test page</title>
</head>
<p>Does this page pass the
validator test?</p>
</html>
```

- Yes
- No

2)
```
<!DOCTYPE html>
<html lang="en">
<h1>Test page</h1>
<p>Does this page pass the
validator test?</p>
</html>
```

- Yes
- No

Implementing the guidelines

The webpage below implements the developer guidelines:

- Closing tags like `</p>` and `</li>` are always used.
- Double quotes are used around all attribute values, and boolean attributes like `checked` are used appropriately.
- All tags and attributes are lowercase, and tags are indented properly.
- Block elements like `<p>` and `<ol>` start on new lines.
- Separate CSS and JavaScript files are imported with `<link>` and `<script>` elements instead of placing the CSS and JavaScript in the HTML document.
- The `<table>` element is used for displaying tabular data and not for laying out the page contents.

Example 3.7.1: Example webpage implements all guidelines.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Example usage of HTML style rules</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="layout.css">
    <script src="functions.js"></script>
  </head>
  <body>
    <p>According to the <a href="https://usatoday.com/">USA Today</a>,
       the three busiest travel days are:</p>
    <ol>
      <li>Summer weekends</li>
      <li>Thanksgiving</li>
      <li>Christmas</li>
    </ol>
    <p>According to the <a href="https://www.mayoclinic.org">Mayo
Clinic</a>,
       the following values are typical complete blood count results:</p>
    <table>
      <caption>Normal complete blood count results for adults.</caption>
      <tr>
        <th>Count</th>
        <th>Female</th>
        <th>Male</th>
      </tr>
      <tr>
        <td>Red blood cell</td>
        <td>3.90-5.03 trillion cells/L</td>
        <td>4.32-5.72 trillion cells/L</td>
      </tr>
      <tr>
        <td>Hematocrit</td>
        <td>34.9-44.5%</td>
        <td>38.8-50.0%</td>
      </tr>
    </table>
  </body>
</html>
```

PARTICIPATION ACTIVITY

3.7.9: Consider all developer guidelines.

```
( 0) <!DOCTYPE html>
( 1) <html lang="en">
( 2)   <head>
( 3)   <title>Movies based on books</title>
( 4)   </head>
( 5)   <body>
( 6)         <P>The following movies were based on books that
( 7)         had been written previously.
( 8)         <ol>
( 9)              <li>The Wizard of Oz</li>
(10)              <li>Harry Potter and the Sorcerer's Stone</li>
(11)              <li>Willy Wonka and the Chocolate Factory</li>
(12)         </ol>
(13)         <p>Indicate which medium you think is better.</p>
(14)         <form action="http://example.org/vote"
(15)               method="post"><p>The Wizard of Oz</p>
(16)           <label for="wizOzBook">Book:</label>
(17)           <input type="radio" id="wizOzBook" name="wizOz"
(18)                  value="wizBook" />
(19)           <label for="wizOzMovie">Movie:</label>
(20)           <input type="radio" id="wizOzMovie" name="wizOz"
(21)                  value="wizMovie" >
(22)           <p>Harry Potter and the Sorcerer's Stone:</p>
(23)           <label for="PotterBook">Book:</label>
(24)           <input type="radio" id="PotterBook" name="potter"
(25)                  value="PotterBook" >
(26)           <label for="PotterMovie">Movie:</label>
(27)           <input type="radio" id="PotterMovie" name="potter"
(28)                  value="PotterMovie" >
(29)           <p>Willy Wonka and the Chocolate Factory</p>
(30)           <label for="WonkaBook">Book:</label>
(31)           <input type="radio" id="WonkaBook" name="wonka"
(32)                  value="WonkaBook" checked="checked" >
(33)           <label for="WonkaMovie">Movie:</label>
(34)           <input type="radio" id="WonkaMovie" name="wonka"
(35)                  value="WonkaMovie" ><input type='submit'>
(36)         </form>
(37)   </body>
(38) </html>
```

1) Which guideline is violated on line 3?

- Use closing tags
- Start block elements on new lines
- Indent nested elements consistently

2) Which line does not use double quotes correctly?

- ○ 10
- ○ 22
- ○ 35

3) Which guideline is violated on line 32?

- ○ Use closing tags
- ○ Use boolean attributes concisely
- ○ Start block elements on new lines
- ○ Indent nested elements consistently

4) Which guideline is violated on line 18?

- ○ Avoid self-closing tags
- ○ Use double quotes
- ○ Indent nested elements consistently

5) Which line did not start a block element on a new line?

- ○ 8
- ○ 14
- ○ 15
- ○ 35

6) Which guideline is violated in line 6?

- ○ Use closing tags
- ○ Start a block element on a new line
- ○ Use lowercase for all tags and attributes

7) Which guideline is violated in lines 9-11?

- ○ Start block elements on new lines
- ○ Indent nested elements consistently
- ○ Separate content from presentation and functionality
- ○ Validate HTML

Exploring further:

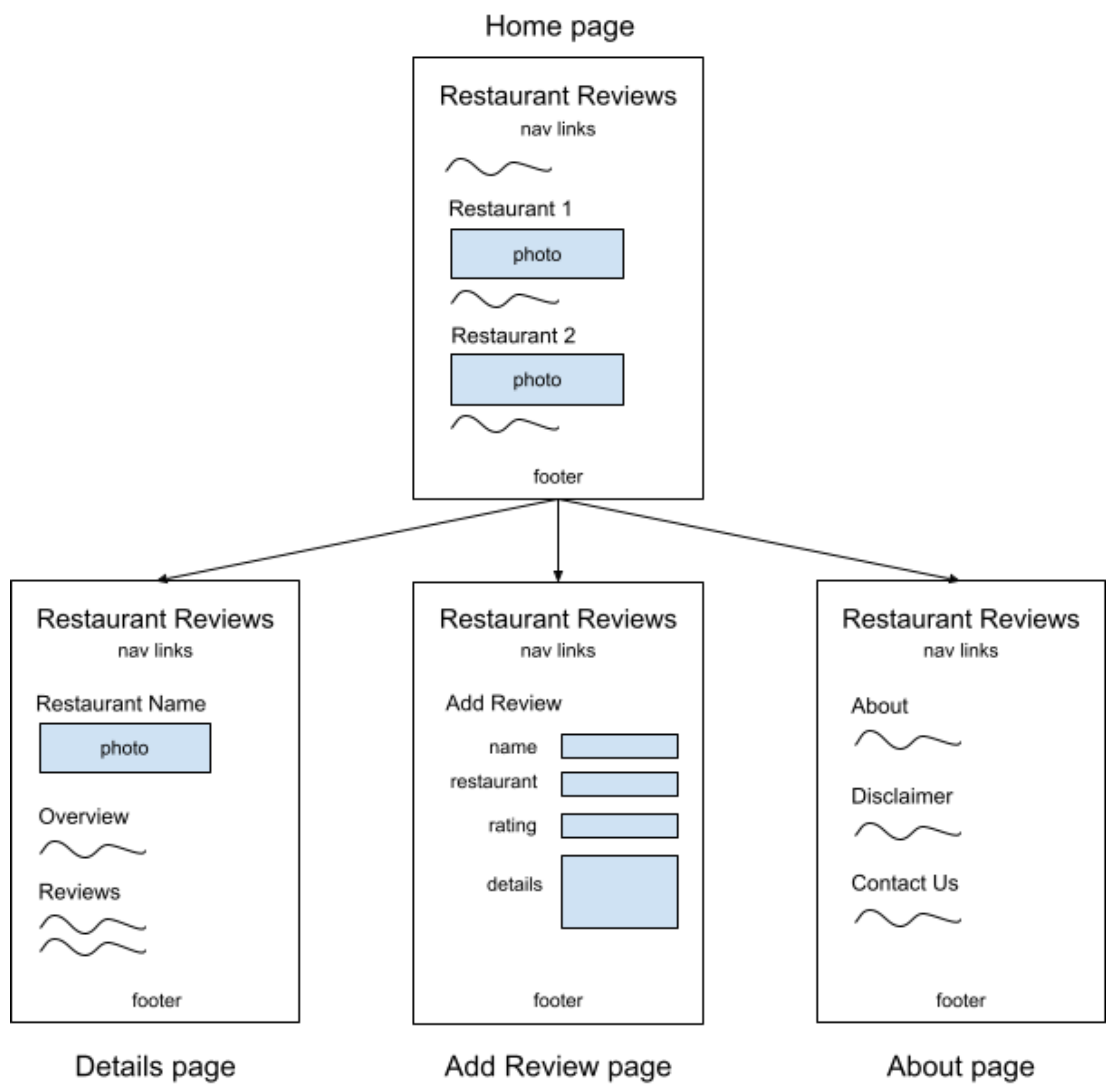
- Google's style guide for HTML and CSS.
- W3Schools' style guide for HTML, CSS, and JavaScript.

3.8 Example: Restaurant Reviews

Initial design

This section presents an example restaurant review website. The website collects reviews from users and displays the reviews on each restaurant's details page.

Figure 3.8.1: Wireframes of restaurant review webpages.



PARTICIPATION ACTIVITY 3.8.1: Restaurant review webpages.

If unable to drag and drop, refresh the page.

Details page **About page** **Add Review page** **Home page**

	Lists details for a restaurant including average rating and reviews.
	Lists featured restaurants.
	Provides information about the website.
	Allows a user to enter a restaurant review.

Reset

Home page

The home page for Restaurant Reviews is index.html, which is a common filename for a website's home page. Most web servers by default serve the ***index.html*** file when a URL does not specify an explicit filename. Ex: Accessing http://cs.harding.edu/ in a web browser loads http://cs.harding.edu/index.html.

To maintain a consistent look, all the website's pages have the same `<header>` tags with navigation links and `<footer>` tags that specify the copyright information. The home page is broken into several sections with `<section>` tags, and each of the featured restaurants are assigned to a section.

index.html is shown below. Clicking the links within the rendered webpage results in a 404 status code. All website files must be placed on a computer's file system or uploaded to a web server for the links to work.

HTML for index.html.

Bottom image from Wikimedia.org

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <title>Restaurant Reviews</title>
   </head>
   <body>
      <header>
         <h1>Restaurant Reviews</h1>
         <nav>
            <a href="index.html">Home</a> &nbsp;
            <a href="addreview.html">Add Review</a> &nbsp;
            <a href="about.html">About</a>
         </nav>
      </header>
```

Render webpage   Reset code

Your webpage

Restaurant Reviews

Home Add Review About

Main Street Cafe

PARTICIPATION ACTIVITY 3.8.2: Home page.

1) The Home page's navigation links link to three webpages.
   - True
   - False

2) The Home page links to every webpage in the website.
   - True
   - False

3) A four-star rating is displayed with a single image of four stars.
   - True
   - False

Detail pages

The website provides detailed restaurant reviews when the user clicks the restaurant name on the home page. Only two restaurants are listed, so only two detail webpages are required for this example:

1. mainstreetcafe.html - Details about Main Street Cafe
2. greekhouse.html - Details about Greek House

Dynamically-built webpages

As individuals enter new reviews, the static detail webpages need to be updated continually. Usually a website that displays frequently-changing content uses a server-side program to dynamically create webpages from data stored in a database. This example, however, uses static content for simplicity.

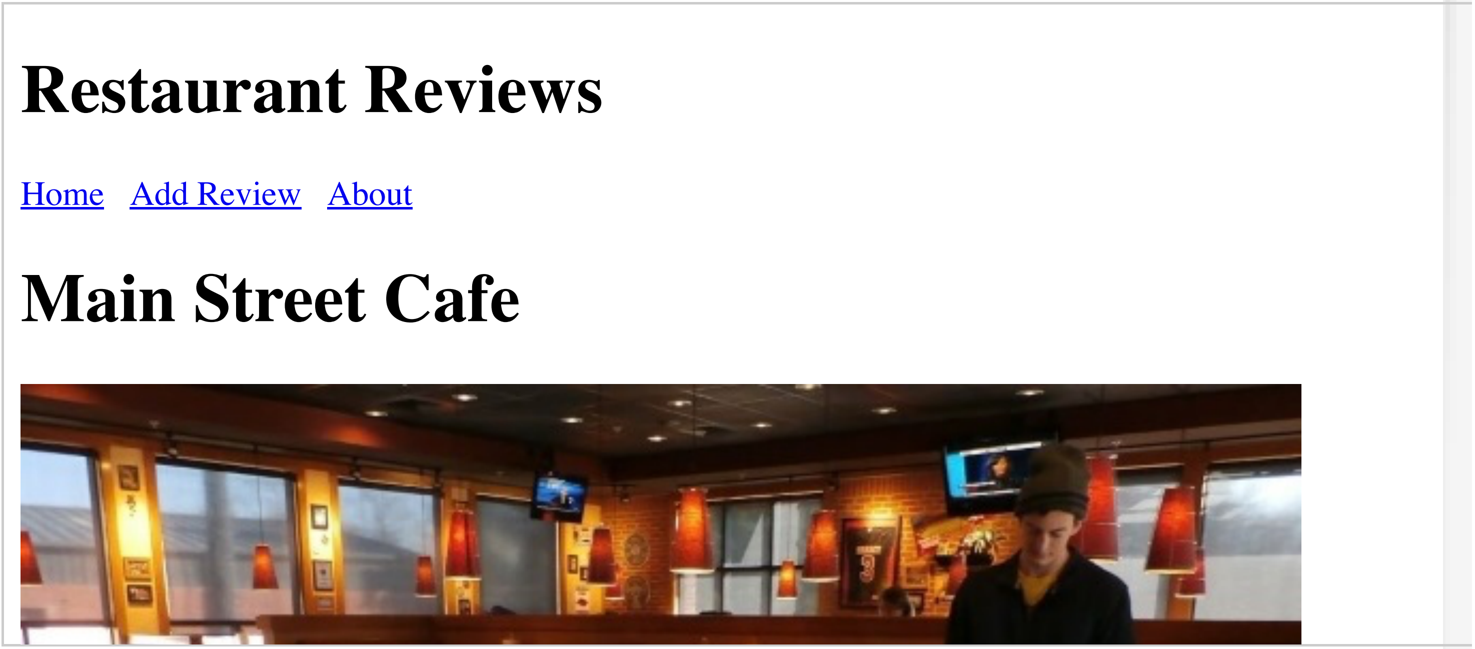
HTML for mainstreetcafe.html.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <title>Restaurant Reviews: Main Street Cafe</title>
   </head>
   <body>
      <header>
         <h1>Restaurant Reviews</h1>
         <nav>
            <a href="index.html">Home</a> &nbsp;
            <a href="addreview.html">Add Review</a> &nbsp;
            <a href="about.html">About</a>
         </nav>
      </header>
```

Render webpage

Reset code

Your webpage



Restaurant Reviews

Home Add Review About

Main Street Cafe

HTML for greekhouse.html.

Bottom image from Wikimedia.org

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <title>Restaurant Reviews: Greek House</title>
   </head>
   <body>
      <header>
         <h1>Restaurant Reviews</h1>
         <nav>
            <a href="index.html">Home</a> &nbsp;
            <a href="addreview.html">Add Review</a> &nbsp;
            <a href="about.html">About</a>
         </nav>
      </header>
```

Render webpage

Reset code

Your webpage

Restaurant Reviews

Home Add Review About

Greek House

PARTICIPATION ACTIVITY 3.8.3: Detail pages.

1) What container tags are used to surround each individual review?

- ○ `<section>`
- ○ `<article>`
- ○ `<div>`

2) If a new review is entered for Main Street Cafe, which webpages may need updating?

- ○ index.html only
- ○ mainstreetcafe.html only
- ○ index.html and mainstreetcafe.html

3) What tag should be added to the Greek House page to display a video advertisement of the Greek House?

- ○ `<video>`
- ○ `<audio>`
- ○ `<embed>`

Add Review webpage

The Add Review webpage displays a form for the user to submit a restaurant review. Normally the form would post the review to a server-side program that would save the review in a database. The example below posts the review to the zyBooks server, which only displays the submitted form values.

HTML for addreview.html.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <title>Restaurant Reviews - Add Review</title>
   </head>
   <body>
      <header>
         <h1>Restaurant Reviews</h1>
         <nav>
            <a href="index.html">Home</a> &nbsp;
            <a href="addreview.html">Add Review</a> &nbsp;
            <a href="about.html">About</a>
         </nav>
      </header>
```

Render webpage Reset code

Your webpage

Restaurant Reviews

Home Add Review About

Add Review

Name?

Restaurant? Choose one...

Date of visit? / /

Rating? 5

PARTICIPATION ACTIVITY 3.8.4: Add Review page.

1) What tag attribute is missing to give the Name text box the focus?

- `required`
- `autofocus`
- `action`

2) What happens if the user clicks the Submit button without entering their name?

- The form submits anyway.
- The browser beeps.
- The browser displays an error message.

3) What happens if the user enters a name, date, rating, and details but clicks the Submit button without choosing a restaurant?

- The form submits anyway.
- The browser chooses a restaurant at random to submit.
- The browser displays an error message.

4) Which HTML attribute prevents the user from entering "4.1" as a rating?

- `required`
- `max`
- `step`

About webpage

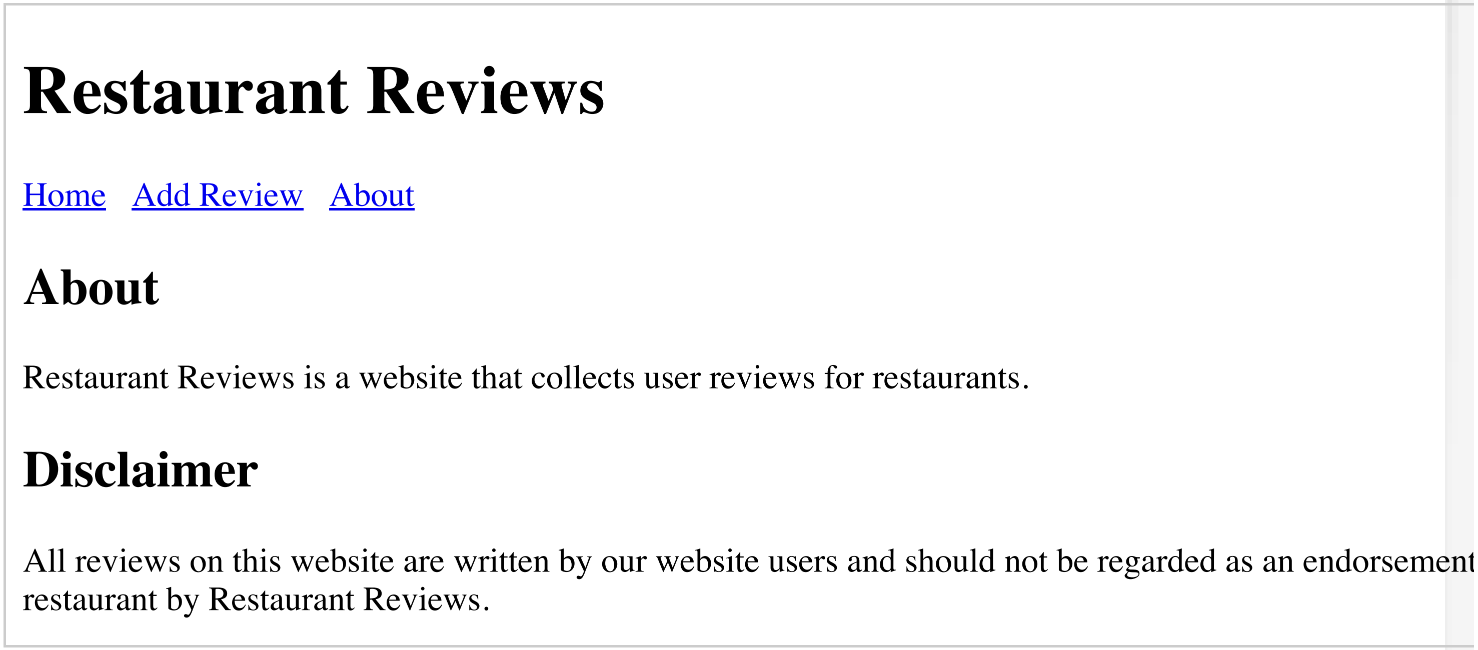
The About page displays information about the website, including an embedded Google Map showing the location of the Restaurant Review headquarters.

HTML for about.html.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <title>Restaurant Reviews - About</title>
   </head>
   <body>
      <header>
         <h1>Restaurant Reviews</h1>
         <nav>
            <a href="index.html">Home</a> &nbsp;
            <a href="addreview.html">Add Review</a> &nbsp;
            <a href="about.html">About</a>
         </nav>
      </header>
```

Render webpage   Reset code

Your webpage



Restaurant Reviews

Home Add Review About

About

Restaurant Reviews is a website that collects user reviews for restaurants.

Disclaimer

All reviews on this website are written by our website users and should not be regarded as an endorsement
restaurant by Restaurant Reviews.

PARTICIPATION ACTIVITY 3.8.5: About page.

1) The `<address>` tag contains only the company's physical address.
   - ○ True
   - ○ False

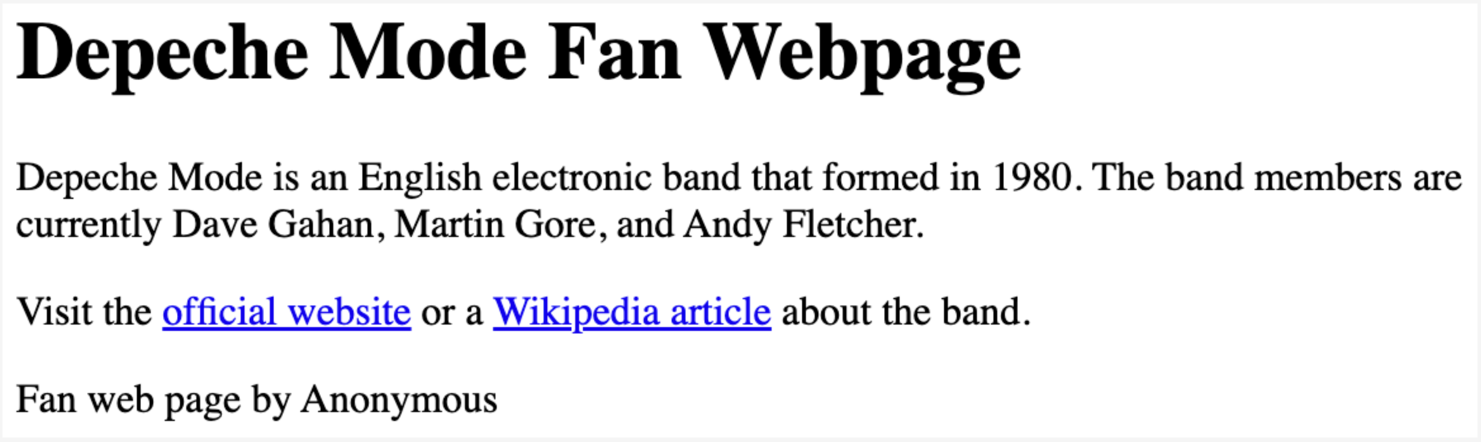
2) The text inside `<address>` is typically rendered in italics.
   - ○ True
   - ○ False

3) Clicking the phone number link in a mobile web browser starts the phone's dialer app.
   - ○ True
   - ○ False

4) The Google Map is embedded in an `<iframe>` tag.
   - ○ True
   - ○ False

3.9 LAB: Fan webpage (HTML)

Create a webpage in a file called index.html about one of your favorite musicians, similar to the example below.



Depeche Mode Fan Webpage

Depeche Mode is an English electronic band that formed in 1980. The band members are currently Dave Gahan, Martin Gore, and Andy Fletcher.

Visit the official website or a Wikipedia article about the band.

Fan web page by Anonymous

Use the following HTML containers:

- `<header>` with `<h1>` tags that contain the musician's name
- `<main>` with 2 paragraphs:
    - 1st paragraph gives some background information
    - 2nd paragraph links to at least one website about the musician
- `<footer>` that names the webpage's creator

550544.4142762.qx3zqy7

LAB ACTIVITY 3.9.1: LAB: Fan webpage (HTML) 10 / 10 ✓

Submission Instructions

Upload your files below by dragging and dropping into the area or choosing a file on your hard drive.

index.html

Drag file here
or
Choose on hard drive.

Submit for grading

Coding trail of your work What is this?

```
3/21 R10 min:2
```

Latest submission - 4:54 PM PDT on 03/21/24 Submission passed all tests ✓ Total score: 10 / 10

☐ Only show failing tests **Download this submission**

1:Unit test 1 / 1

Test for <header>

✓ <header> tag exists in <body>

2:Unit test 2 / 2

Test header's <h1> tag

✓ <h1> tag exists in <header>

✓ <h1> contains text

3:Unit test 1 / 1

Test for <main> tag

✓ <main> tag exists in <body>

4:Unit test 2 / 2

Test main's first <p>

✓ <p> tag exists in <main>

✓ <p> tag contains text

5:Unit test 2 / 2

Test main's second <p>

✓ Second <p> tag exists in <main>

✓ Second <p> tag contains at least 1 <a> tag

✓ <a> tag has href attribute

6:Unit test 2 / 2

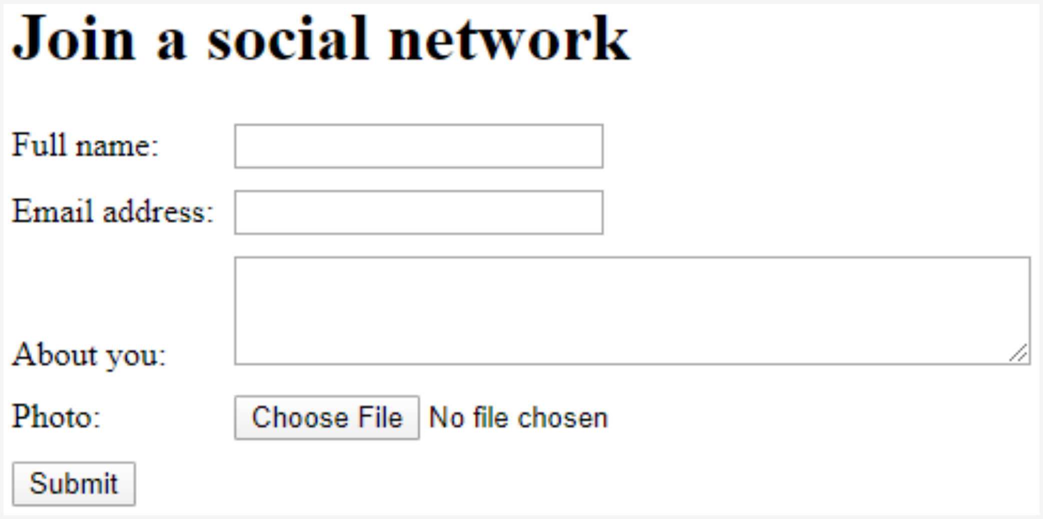
Test for <footer>

✓ <footer> tag exists in <body>

✓ <footer> contains text

3.10 LAB: Form for joining a social network (HTML)

Download the ZIP file below containing an HTML and CSS file. Add a web form to index.html as shown below that allows the user to enter personal information for joining a social network. The webpage uses CSS to align the form widgets vertically.



The form should:

- Use the POST method and the "multipart/form-data" enctype
- Submit to https://wp.zybooks.com/form-viewer.php
- Use a `<label>` for each form field and the label text shown in the image above
- Use a `<div>` to surround each label and form widget and a `<div>` around the submit button
- Use a text `<input>` with name and id "fullName" to get the user's name
- Use a text `<input>` with name and id "email" to get the user's email address
- Use a `<textarea>` with 3 rows, 50 columns, and name and id "about" to get a short description about the user
- Use a text `<input>` with type "file" and name and id "picture" to get the user's image
- Use a submit `<input>` button
- Use the `required` attribute for name and email fields

550544.4142762.qx3zqy7

LAB ACTIVITY 3.10.1: LAB: Form for joining a social network (HTML) 10 / 10 ✓

Submission Instructions

Downloadable files

`index.html` and `styles.css` **Download**

Upload your files below by dragging and dropping into the area or choosing a file on your hard drive.

index.html

Drag file here
or
Choose on hard drive.

Submit for grading

Coding trail of your work What is this?

```
3/22 F10 min:1
```

Latest submission - 11:04 AM PDT on 03/22/24 Submission passed all tests ✓ Total score: 10 / 10

☐ Only show failing tests **Download this submission**

1:Unit test 1 / 1

Test form tag

✓ <form> tag exists

✓ Form method is POST

✓ Form enctype is multipart/form-data

✓ Form action is zybooks.com URL

2:Unit test 1 / 1

Test full name label

- ✓ <div> is inside <form> tags
- ✓ <label> in div exists
- ✓ <label> has correct text
- ✓ <label> has correct for attribute value

3:Unit test 1 / 1

Test full name input

- ✓ <div> is inside <form> tags
- ✓ <input> in div exists
- ✓ <input> has correct name
- ✓ <input> has correct id
- ✓ <input> uses required attribute

4:Unit test 1 / 1

Test email label

- ✓ <div> is inside <form> tags
- ✓ <label> in div exists
- ✓ <label> has correct text
- ✓ <label> has correct for attribute value

5:Unit test 1 / 1

Test email input

- ✓ <div> is inside <form> tags

- ✓ <input> in div exists
- ✓ <input> has correct name
- ✓ <input> has correct id
- ✓ <input> uses required attribute

6:Unit test 1 / 1

Test about label

- ✓ <div> is inside <form> tags
- ✓ <label> in div exists
- ✓ <label> has correct text
- ✓ <label> has correct for attribute value

7:Unit test 1 / 1

Test about textarea

- ✓ <div> is inside <form> tags
- ✓ <textarea> in div exists
- ✓ <textarea> has correct name
- ✓ <textarea> has correct id
- ✓ <textarea> has correct cols attribute
- ✓ <textarea> has correct rows attribute

8:Unit test 1 / 1

Test picture label

- ✓ <div> is inside <form> tags

✓ <label> in div exists

✓ <label> has correct text

✓ <label> has correct for attribute value

9:Unit test 1 / 1

Test for file input

✓ <div> is inside <form> tags

✓ <input> in div exists

✓ <input> has correct name

✓ <input> has correct id

✓ <input> has correct type attribute

10:Unit test 1 / 1

Test for submit button

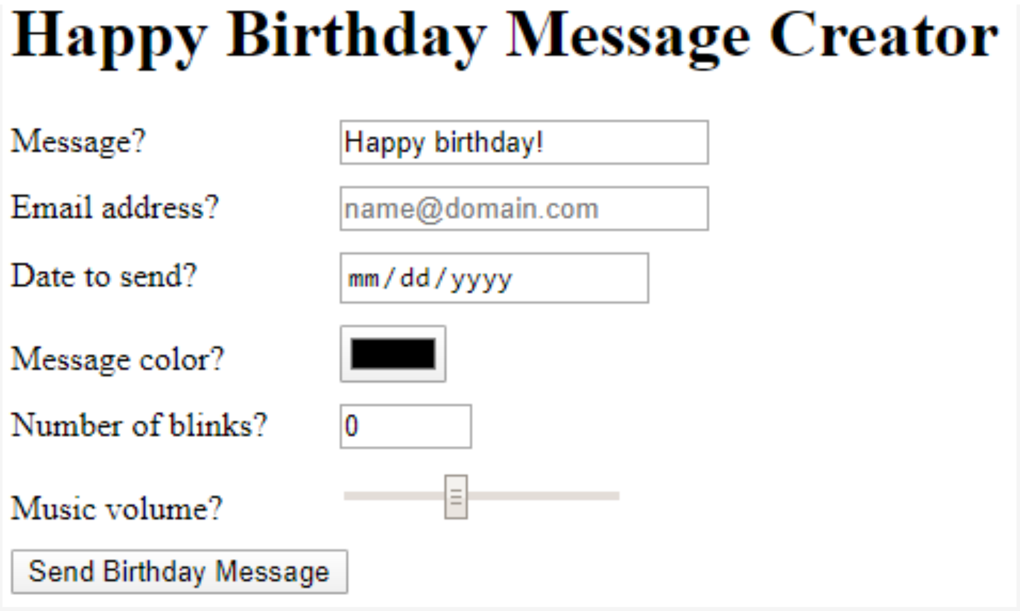
✓ <div> is inside <form> tags

✓ <input> in div exists

✓ <input> has correct type attribute

3.11 LAB: Happy birthday message creator (HTML)

Download the ZIP file below containing an HTML and CSS file. Edit the form in index.html so the user can enter a birthday message and select various message options. The webpage uses CSS to align the form widgets vertically. The webpage should look like the following:


Happy Birthday Message Creator

Message?	Happy birthday!
Email address?	name@domain.com
Date to send?	mm/dd/yyyy
Message color?	■
Number of blinks?	0
Music volume?	

The form should:

- Use the regular expression `[a-zA-Z0-9-_\.]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*` for the email address `<input>` to verify the user enters what looks to be an email address
- Use a date `<input>` for specifying a date to send the message
- Use a color `<input>` for selecting the message color
- Use a number `<input>` with a default value of 0, minimum of 0, and maximum of 10 for number of times the message blinks
- Use a range `<input>` with a default value of 20, minimum of 0, and maximum of 50 for the music volume
- Use a submit `<button>` that reads "Send Birthday Message"
- Specify an `id` attribute for all form elements that matches the given `for` attribute of each associated `<label>`
- Specify a `name` attribute for all form elements that matches each element's `id` attribute

550544.4142762.qx3zqy7

LAB ACTIVITY 3.11.1: LAB: Happy birthday message creator (HTML) 10 / 10 ✓

Submission Instructions

Downloadable files

`index.html` and `styles.css` Download

Upload your files below by dragging and dropping into the area or choosing a file on your hard drive.

index.html

Drag file here
or
Choose on hard drive.

Submit for grading

Coding trail of your work What is this?

```
3/22 F9,10 min:2
```

Latest submission - 11:25 AM PDT on 03/22/24 Submission passed all tests ✓ Total score: 10 / 10

☐ Only show failing tests Download this submission

1:Unit test 1 / 1

Test email <input> placeholder

✓ <input> exists with type="email" attribute

✓ <input> pattern attribute exists

✓ pattern has correct value

2:Unit test 1 / 1

Test date <input>

✓ <input> exists with type="date" attribute

✓ <input> has correct id

✓ <input> has correct name

3:Unit test 1 / 1

Test color <input>

✓ <input> exists with type="color" attribute

✓ <input> has correct id

✓ <input> has correct name

4:Unit test 1 / 1

Test number <input>

✓ <input> exists with type="number" attribute

✓ <input> has correct id

✓ <input> has correct name

5:Unit test 2 / 2

Test number <input> value, min, max

✓ <input> exists with type="number" attribute

✓ <input> has correct value

✓ <input> has correct min

✓ <input> has correct max

6:Unit test 1 / 1

Test range <input>

✓ <input> exists with type="range" attribute

✓ <input> has correct id

✓ <input> has correct name

7:Unit test 2 / 2

Test range <input> value, max, min

✓ <input> exists with type="range" attribute

✓ <input> has correct value

✓ <input> has correct min

✓ <input> has correct max

8:Unit test 1 / 1

Test submit <button>

✓ Submit <button> exists

✓ Button type is submit

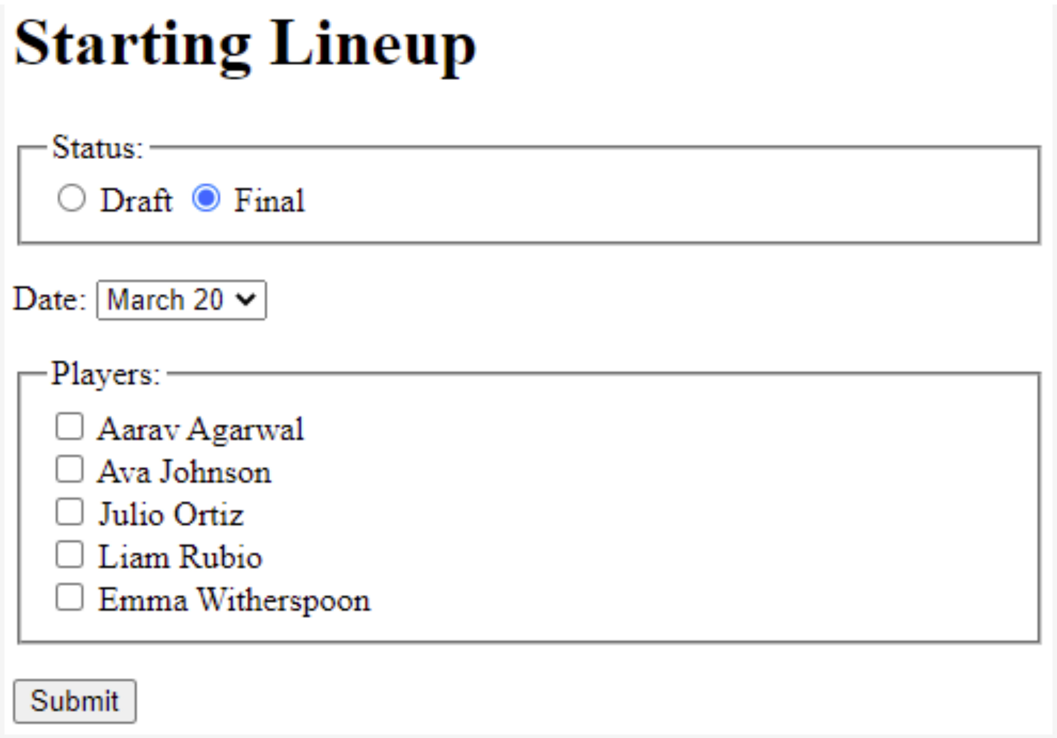
✓ Submit <button> inner text is correct

Previous submissions

11:23 AM on 3/22/24 9 / 10 View

3.12 LAB: Starting lineup (HTML)

Download the ZIP file below containing an HTML file. Add a form to index.html that allows the user to select a team's starting lineup for a game.



Starting Lineup

Status:

Draft [✓ Final]

Date: March 20

Players:

Aarav Agarwal
Ava Johnson
Julio Ortiz
Liam Rubio
Emma Witherspoon

Submit

The form should:

- Use the POST method
- Submit to https://wp.zybooks.com/form-viewer.php
- Use a `<p>` to surround related labels, form widgets, and fieldsets
- Use a `<fieldset>` and `<legend>` with radio buttons for choosing Draft or Final. The radio buttons' name attribute should be "status" with values "Draft" and "Final".
- Make the Final radio button checked by default
- Add a `<label>` to each radio button so clicking the label text selects the radio button
- Use a drop-down with the following dates: March 20, March 24, April 2, April 6, April 13. The `<select>` name attribute should be "gameDate", and each `<option>` should use a value identical to the option's date.
- Use a `<fieldset>` and `<legend>` with check boxes for choosing players: Aarav Agarwal, Ava Johnson, Julio Ortiz, Liam Rubio, Emma Witherspoon. The name attribute "player" should be used for all check boxes, and each check box should use a value identical to the player's name.
- Add a `<label>` to each checkbox so clicking the label text selects the checkbox
- Use a submit `<input>` with value "Submit"

550544.4142762.qx3zqy7

LAB ACTIVITY | 3.12.1: LAB: Starting lineup (HTML) 10 / 10 ✓

Submission Instructions

Downloadable files

`index.html` **Download**

Upload your files below by dragging and dropping into the area or choosing a file on your hard drive.

index.html

Drag file here
or
Choose on hard drive.

Submit for grading

Coding trail of your work What is this?

```
3/23 S6,9,9,9,10 min:3
```

Latest submission - 1:41 PM PDT on 03/23/24 Submission passed all tests ✓ Total score: 10 / 10

☐ Only show failing tests **Download this submission**

1:Unit test 1 / 1

Test <form> tag

✓ <form> tag exists

✓ Form method is POST

✓ Form action is zybooks.com URL

2:Unit test 1 / 1

Test radio buttons

- ✓ 2 radio buttons exist
- ✓ First radio button uses correct name
- ✓ Second radio button uses correct name

3:Unit test 1 / 1

Test labels for radio buttons

- ✓ 2 radio buttons exist
- ✓ At least one radio button has a <label> sibling
- ✓ First <label> has a for attribute
- ✓ First radio button has an id attribute
- ✓ First label's for attribute value matches first radio button's id
- ✓ Two radio buttons have a <label> sibling
- ✓ Second <label> has a for attribute
- ✓ Second radio button has an id attribute
- ✓ Second label's for attribute value matches second radio button's id

4:Unit test 1 / 1

Test drop-down <select> tag

- ✓ <select> in form exists
- ✓ <select> name is correct

5:Unit test 1 / 1

Test drop-down <option> tags

- ✓ Correct number of <option> tags in <select>
- ✓ <option> 1 value
- ✓ <option> 1 text
- ✓ <option> 2 value
- ✓ <option> 2 text
- ✓ <option> 3 value
- ✓ <option> 3 text
- ✓ <option> 4 value
- ✓ <option> 4 text
- ✓ <option> 5 value
- ✓ <option> 5 text

6:Unit test 2 / 2

Test checkbox <input> tags

- ✓ Correct number of <input> tags in <select>
- ✓ Checkbox <input> 1 name
- ✓ Checkbox <input> 1 value
- ✓ Checkbox <input> 2 name
- ✓ Checkbox <input> 2 value
- ✓ Checkbox <input> 3 name
- ✓ Checkbox <input> 3 value

- ✓ Checkbox <input> 4 name
- ✓ Checkbox <input> 4 value
- ✓ Checkbox <input> 5 name
- ✓ Checkbox <input> 5 value

7:Unit test 1 / 1

Test checkbox <label> tags

- ✓ Correct number of <input> tags in <select>
- ✓ Checkbox has an id attribute
- ✓ Checkbox's id attribute "aaravAgarwal" contains NO spaces
- ✓ Label has for attribute that matches checkbox id "aaravAgarwal"
- ✓ Label is sibling of checkbox with id "aaravAgarwal"
- ✓ Checkbox <input> 1 label text
- ✓ Checkbox has an id attribute
- ✓ Checkbox's id attribute "avaJohnson" contains NO spaces
- ✓ Label has for attribute that matches checkbox id "avaJohnson"
- ✓ Label is sibling of checkbox with id "avaJohnson"
- ✓ Checkbox <input> 2 label text
- ✓ Checkbox has an id attribute
- ✓ Checkbox's id attribute "julioOrtiz" contains NO spaces
- ✓ Label has for attribute that matches checkbox id "julioOrtiz"
- ✓ Label is sibling of checkbox with id "julioOrtiz"

- ✓ Checkbox <input> 3 label text
- ✓ Checkbox has an id attribute
- ✓ Checkbox's id attribute "liamRubio" contains NO spaces
- ✓ Label has for attribute that matches checkbox id "liamRubio"
- ✓ Label is sibling of checkbox with id "liamRubio"
- ✓ Checkbox <input> 4 label text
- ✓ Checkbox has an id attribute
- ✓ Checkbox's id attribute "emmaWitherspoon" contains NO spaces
- ✓ Label has for attribute that matches checkbox id "emmaWitherspoon"
- ✓ Label is sibling of checkbox with id "emmaWitherspoon"
- ✓ Checkbox <input> 5 label text

8:Unit test 1 / 1

Test <fieldset> tags

- ✓ Correct number of <fieldset> tags exist in <form>
- ✓ First <fieldset> contains a single <legend> tag
- ✓ First <fieldset> legend text
- ✓ Second <fieldset> contains a single <legend> tag
- ✓ Second <fieldset> legend text

9:Unit test 1 / 1

Test submit button

- ✓ A single submit <input> exists

✓ Submit button value is correct

Previous submissions

1:40 PM on 3/23/24	9 / 10	View
1:40 PM on 3/23/24	9 / 10	View
1:39 PM on 3/23/24	9 / 10	View
1:37 PM on 3/23/24	6 / 10	View