HTML Forms Tutorial

Introduction

- HTML forms collect user input, usually sent to a server for processing.
- The <form> tag defines a container for input elements (text fields, checkboxes, radio buttons, submit buttons).

Basic Form Syntax

```
<form>
  <!-- form elements -->
</form>
```

Note: Without attributes, form data submits to the same page.

Text and Password Fields

```
<input type="text"> <!-- Single-line text input -->
<input type="password"> <!-- Password input -->
```

Example 1: Create an HTML file named form.html and start by adding the code snippet from Example 1-1. Continue updating this file as you progress through the rest of the example.

Example 1-1: <label> and <form>

```
<html>
<head><title>Form Example</title></head>
<body>
 <form>
  <label>Name</label>
  <input>
  <label>Password</label>
  <input>
 </form>
</body>
</html>
```

Note: The id attribute is valid for any HTML element and is used to uniquely identify elements in the document. The name attribute, however, is only valid on certain elements, such as <a>, <input>, <select>, <textarea>, and other form controls. When working with forms, use the name attribute on form controls because it determines the key names used in the POST or GET request when the form is submitted.

**Structuring with **

Example 1-2: Separating inputs clearly: to separate each block of code into different lines.

```
<html>
<head><title>Form Example</title></head>
<body>
 <form>
  <div>
   <label>Name</label>
   <input>
  </div>
  <div>
   <label>Password</label>
   <input>
  </div>
 </form>
</body>
</html>
```

Adding a Submit Button

Example 1-3: Using submit button: add a submit button: <input type="submit">.

```
<html>
<head><title>Form Example</title></head>
<body>
 <form>
  <div>
   <label>Name</label>
   <input>
  </div>
  <div>
   <label>Password</label>
   <input>
  </div>
  <button type="submit">Submit</button>
 </form>
</body>
</html>
```

Note: In most browsers, the default type for a <button> element is submit.

Example 1-4: Specifying action and method

Action and Method Attributes

Add the action and method attributes to the <form> tag to define how form data should be submitted.

- The **submit** button is used to send the form data to a **form handler**, which is typically a webpage or server-side script that processes the submitted data.
- The **action** attribute specifies **where** the form data should be sent (i.e., the URL or path to the form handler).
- The **method** attribute determines **how** the data is sent. It can be:
    - **GET** – Appends the form data to the URL as query parameters. Suitable for retrieving or searching data.
    - **POST** – Sends the form data in the request body. Commonly used when submitting sensitive or large amounts of data, or when modifying server-side resources.

```
<html>
<head><title>Form Example</title></head>
<body>
 <form action="results.html" method="GET">
  <div>
   <label>Name</label>
   <input>
  </div>
  <div>
   <label>Password</label>
   <input>
  </div>
  <button type="submit">Submit</button>
 </form>
</body>
</html>
```

Example 1-5: Results page: create the html page results.html. Make sure it's in the same folder as your form.html file.

```
<!DOCTYPE html>
<html>
<head>
 <title>Results</title>
</head>
<body>
 <h3>Results</h3>
 <script>
  // Writes the query string directly into the HTML
  document.write(window.location.search);
 </script>
</body>
</html>
```

Note: document.write() is generally discouraged because it can overwrite the entire page if called after the document has loaded.

Improved Version (using innerHTML)

```
<!DOCTYPE html>
<html>
<head>
 <title>Results</title>
</head>
<body>
 <h3>Results</h3>
  <!-- Placeholder for output -->
 <div id="results"></div>
 <!-- Link to return to the form -->
 <a href="form.html">Back to Form</a>
 <script>
  // Safer way to insert the query string into the page
  document.getElementById("results").innerHTML = window.location.search;
 </script>
</body>
</html>
```

Note: Uses innerHTML instead of document.write, which is safer and doesn't disrupt the DOM.

JavaScript DOM and URL Basics

- document:
  Represents the web page loaded in the browser. To access or manipulate any element in the HTML, you begin with the document object.
- getElementById():
  A method of the document object used to find and return an element by its id.
- innerHTML:
  A property that gets or sets the HTML content inside an element.
- window.location.search:
  Contains the query string (the part of the URL that comes after ?) of the current page's URL.

Naming Inputs

Example 1-6: Specifying input names: go back to form.html and specify names for the inputs.

```
<html>
<head><title>Form Example</title></head>
<body>
 <form action="results.html" method="GET">
  <div>
   <label>Name</label>
   <input name="name">
  </div>
  <div>
   <label>Password</label>
   <input name="password">
  </div>
  <button type="submit">Submit</button>
 </form>
</body>
</html>
```

Note:
The id attribute is valid for **any HTML element**. However, the name attribute is only valid on **specific elements**, such as <a>, <input>, <select>, and other **form controls**.
For form submissions, use the name attribute on form elements because it serves as the **key** in the data sent via **POST** or **GET** when the form is submitted.

Parsing Query Strings

Example 1-7: Extracting values: parse through window.location.search to extract the values for name and password

```
<html>
<head><title>Results</title></head>
<body>
 <div id="results"></div>
 <a href="form.html">Back to Form</a>
 <script>
  const params = new URLSearchParams(window.location.search);
  document.write("<br>" + params.get('name'));
  document.write("<br>" + params.get('password'));
 </script>
</body>
</html>
```

Working with URL Parameters in JavaScript

- URLSearchParams():
  The URLSearchParams API provides an easy and consistent way to work with the query string (the part of a URL after ?). It allows you to read, manipulate, and iterate over individual parameters.
- document.write():
  This method writes the given content directly into the HTML document at the time it is executed. Use with caution, as it can overwrite the entire page if used after the page has loaded.
- params.get('var_name'):
  Returns the value of the query parameter named 'var_name' from the URL. This is typically used after creating a URLSearchParams object, like:

  const params = new URLSearchParams(window.location.search);

  const value = params.get('var_name');

Using "for" and "id" Attributes

Example 1-8: Associating labels and inputs: add for and corresponding id and specify type of input.

```
<html>
<head><title>Form Example</title></head>
<body>
 <form action="results.html" method="GET">
  <div>
   <label for="name">Name</label>
   <input type="text" id="name" name="name">
  </div>
  <div>
   <label for="password">Password</label>
   <input type="password" id="password" name="password">
  </div>
  <button type="submit">Submit</button>
 </form>
</body>
</html>
```

Note: When used with a <label> element, the for attribute specifies which form element the label is associated with.

Checkboxes, Radio Buttons, and Drop-Down Lists

Example 2: Checkboxes

Checkboxes allow the user to select zero or more options from a set of choices.

The <input type="checkbox"> tag defines a checkbox.

```
<html>
<head><title>Checkboxes</title></head>
<body>
 <form>
  <p>Select your items:</p>

  <div>
   <label for="item1">Item 1</label>
   <input type="checkbox" name="item1" id="item1" value="item1">
  </div>

  <div>
   <label for="item2">Item 2</label>
   <input type="checkbox" name="item2" id="item2" value="item2">
  </div>

  <div>
   <label for="item3">Item 3</label>
   <input type="checkbox" name="item3" id="item3" value="item3">
  </div>
 </form>
</body>
</html>
```

Note: Use the value attribute if you need the selected data to be sent to the server.

Example 3: Radio Buttons

Radio buttons allow the user to select only one option from a group.

The name attribute must be the same for all radio buttons in the group to ensure only one can be selected.

```
<html>
<head><title>Radio Buttons</title></head>
<body>
 <form>
  <p>Gender:</p>

  <div>
   <label for="male">Male</label>
   <input type="radio" name="gender" id="male" value="male">
  </div>

  <div>
   <label for="female">Female</label>
   <input type="radio" name="gender" id="female" value="female">
  </div>

  <div>
   <label for="other">Other</label>
   <input type="radio" name="gender" id="other" value="other">
  </div>
 </form>
</body>
</html>
```

Example 4: Drop-Down List

A drop-down list allows the user to select one option from a predefined list.

```
<html>
<head><title>Drop-down Lists</title></head>
<body>
 <form>
  <label for="eyeColor">Eye Color:</label>
  <select name="eyeColor" id="eyeColor">
   <option value="Green">Green</option>
   <option value="Black">Black</option>
  </select>
 </form>
</body>
</html>
```

Useful Attributes for Form Elements

Attribute	Description
type	Defines the type of input (e.g. text, password, email, number, etc.)
name	The key used in form data sent to the server (paired with value)
size	Specifies the width of the input field (in characters)
maxlength	Maximum number of characters the user can enter
value	Sets a default value for the input field
placeholder	Displays a hint to the user inside the input field
required	Makes the field mandatory before submission
min	Minimum value for number, date, etc.
max	Maximum value for number, date, etc.
step	Specifies the interval between valid values (e.g., step="5")
multiple	Allows multiple selections (used with <select> and file inputs)
<textarea>	Element used to collect multiline text

References

- HTML Forms - W3Schools
- Forms Video Tutorial
- GET vs POST