Forms

<form> tag

The <form> tag in HTML is a container used to create an interactive form on a web page. Forms are crucial for user input, allowing visitors to submit data to a server for processing. The <form> tag encompasses various form elements like text fields, checkboxes, radio buttons, and more.

Basic Structure:

```
<form action="/submit" method="post">
  <!-- Form elements go here -->
</form>
```

Attributes of the <form> Tag:

- **'action'**: Specifies the URL or script to which the form data should be submitted. It indicates where the form data will be processed.

```
<form action="/submit" method="post">
```

- '**method**': Defines the HTTP method to send form data to the server. Typical values are "**get**" and "**post**".

```
<form action="/submit" method="post">
```

- '**enctype**': Specifies the encoding type for form data when using the post method. Common values include "**application/x-www-form-urlencoded**" and "**multipart/form-data**". This is often used when dealing with file uploads.

```
<form action="/submit" method="post" enctype="multipart/form-data">
```

- **‘target’**: Determines where the form response should be displayed. Values include _blank (opens in a new window or tab), _self (default, opens in the same window), _parent, and _top.

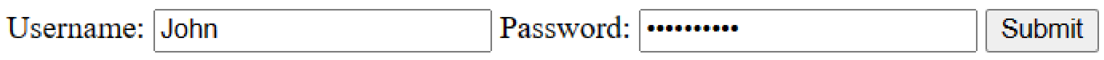
```
<form action="/submit" method="post" target="_blank">
```

Example:

```
<form action="/submit" method="post" enctype="multipart/form-data">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required>

  <label for="password">Password:</label>
  <input type="password" id="password" name="password" required>

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



In this example, the form collects a username and password, and when submitted, the data is sent to the server-side script specified in the action attribute. The method is set to "post" for more secure data transmission, and the enctype is specified for potential file uploads. The required attribute ensures that users must fill in the specified fields before submitting the form.

- **‘novalidate’**: The novalidate attribute is used to disable the default browser validation of form fields. When this attribute is present on the <form> tag, it prevents the browser from validating the form before submission. This can be useful when you want to perform custom validation on the server side or using JavaScript..

```
<form action="/submit" method="post" novalidate>
  <!-- form fields go here -->
</form>
```

- **‘autocomplete’**: The autocomplete attribute is used to control whether a browser should automatically complete form fields or not. It is applied to individual form elements or the <form> tag itself. The values it can take are "**on**" or "**off**".
  - **autocomplete="on"**: This is the default behaviour, where the browser may store and automatically fill in form data based on the user's previous input

```
<form action="/submit" method="post" autocomplete="on">
  <!-- form fields go here -->
</form>
```

  - **autocomplete="off"**: This disables the browser's autocomplete feature, preventing it from suggesting or filling in values for form fields.

```
<form action="/submit" method="post" autocomplete="off">
  <!-- form fields go here -->
</form>
```

Elements of the <form> tag:

- **‘input’**: The <input> tag is used to create various types of form controls, such as text fields, checkboxes, radio buttons, and more. The specific type is specified using the type attribute.

```
<input type="text" name="username" placeholder="Enter your username">
```

- **‘label’:** The <**label**> tag associates a text label with a form control, enhancing accessibility and user experience. It is often used with form elements like <**input**>, <**select**>, and <**textarea**>.

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

- **'<select> and <option>':** The <select> tag creates a dropdown list, and <option> tags define the options within the list.

```
<select name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="mercedes">Mercedes</option>
  <option value="audi">Audi</option>
</select>
```

- **'<textare>':** The <**textarea**> tag is used to create a multiline text input field, typically for longer text entries or comments.

```
<textarea name="message" rows="4" cols="50">Enter your message here...
</textarea>
```

- **<button>:** The <button> tag creates a clickable button, which can be used for various purposes such as form submission or triggering JavaScript functions.

```
<button type="submit">Submit</button>
```

- **<fieldset> and <legend>:** The <fieldset> tag groups related form elements together, and <legend> provides a caption or title for the <fieldset>.

```
<fieldset>
  <legend>Personal Information</legend>
  <!-- Form elements go here -->
</fieldset>
```

- **<datalist> and <option>:** The <datalist> tag contains a list of <option> elements used to provide suggestions or autocompletion for an <input> field

```
<input list="browsers" name="browser" id="browser">
<datalist id="browsers">
  <option value="Chrome">
  <option value="Firefox">
  <option value="Safari">
  <option value="Edge">
  <option value="Opera">
</datalist>
```

- **<optput>:** The <output> tag is used to represent the result of a calculation or user action.

```
<form oninput="result.value=parseInt(a.value)+parseInt(b.value)">
  <input type="range" id="a" value="50">+
  <input type="number" id="b" value="50">
  =<output name="result" for="a b">100</output>
</form>
```

- **<optgroup>:** The <optgroup> tag groups <option> elements within a <select> dropdown, providing a way to organize and structure the options.

```
<select>
  <optgroup label="Fruits">
    <option value="apple">Apple</option>
    <option value="banana">Banana</option>
  </optgroup>
  <optgroup label="Vegetables">
    <option value="carrot">Carrot</option>
    <option value="broccoli">Broccoli</option>
  </optgroup>
</select>
```

These HTML tags play crucial roles in creating interactive and user-friendly forms on web pages. They provide a variety of input methods and customisation options to cater to different types of user interactions.

HTML Input Types

Here are the different input types you can use in HTML:

- **<input type="button">**: Creates a clickable button, typically used in conjunction with JavaScript for custom actions.
- **<input type="checkbox">**: Creates a checkbox for selecting multiple options in a group.
- **<input type="color">:** Provides a colour picker interface for selecting a colour.
- **<input type="date">:** Generates a date input field, allowing users to select a date from a calendar.
- **<input type="datetime-local">:** Allows users to input a date and time in the local time zone.
- **<input type="email">:** Creates a text box specifically for email input, with built-in validation for email addresses.
- **<input type="hidden">:** Hides the input value from the user interface but allows it to be submitted with the form.
- **<input type="image">:** Creates an image as a submit button. When clicked, it sends the coordinates of the click location to the server.
- **<input type="month">:** Allows users to input a month and year.
- **<input type="number">:** Creates a numeric input field, restricting input to numeric values.
- **<input type="password">:** Generates a password input field, obscuring the entered text.

- **<input type="radio">:** Creates a radio button for selecting a single option from a group.
- **<input type="range">:** Creates a slider control, allowing users to select a value within a specified range.
- **<input type="reset">:** Resets all form elements to their default values.
- **<input type="search">:** Generates a search input field.
- **<input type="submit">:** Creates a submit button for submitting form data to the server.
- **<input type="tel">:** Allows users to input a telephone number.
- **<input type="text">:** Generates a single-line text input field.
- **<input type="time">:** Allows users to input a time value.
- **<input type="url">:** Creates a text box for inputting a URL, with built-in validation for URLs.
- **<input type="week">:** Allows users to input a week and year.

Each input type serves a specific purpose, providing a wide range of options for collecting user input in web forms.

File Input

The <**input**> element with **type**="**file**" is used to create a file input field, allowing users to select and upload files from their devices. The associated attributes -**accept**, -**capture**, -**id**, and -**name** are not standard HTML attributes for the <input> element, but they might be used in certain contexts or frameworks.

Here is the explanation of the standard and commonly used attributes for the <input type="file"> element:

1. **accept**: Specifies the types of files the file input should accept. It is a comma-separated list of file type specifiers, such as file extensions or MIME types. This attribute can help filter the types of files that users can select.

```
<input type="file" accept=".pdf, .doc, .docx">
```

2. **capture**: Used in mobile devices to specify the device's camera or file picker. It can take values like "**user**" (for the front-facing camera), "**environment**" (for the rear-facing camera), or "**filesystem**" (for file pickers).

```
<input type="file" capture="user">
```

3. **id**: Specifies a unique identifier for the input element. This can be useful for associating the input with a label or for targeting the element with JavaScript or CSS.

```
<input type="file" id="fileInput">
```

4. **name**: Specifies the input element's name, which is sent to the server when the form is submitted. This attribute is crucial for identifying the input field in server-side processing.

```
<input type="file" name="fileUpload">
```

Button Tag

The <button> tag in HTML is used to create a clickable button on a webpage. It can contain text, images, or other HTML elements, and it is often used to trigger JavaScript functions, submit forms, or handle user interactions.

Attributes of the <button> Tag:

1. **type**: Specifies the type of button. The most common values are
    a. "**button**" (default): A regular button.
    b. "**submit**": Submits the form data.
    c. "**reset**": Resets the form fields to their default values.

```
<button type="submit">Submit</button>
```

2. **name**: Assigns a name to the button, which is sent to the server when the form is submitted. Useful when multiple buttons exist in the same form.

```
<button type="submit" name="submitButton">Submit</button>
```

3. **value**: Assigns a value to the button, which is also sent to the server when the form is submitted. Useful when multiple buttons submit the same form.

```
<button type="submit" value="submit">Submit</button>
```

4. **disabled:** Disables the button, preventing user interaction. The button appears grayed out.

```
<button type="button" disabled>Disabled Button</button>
```

5. **onclick:** Specifies a JavaScript function to be executed when the button is clicked.

```
<button type="button" onclick="myFunction()">Click me</button>
```

6. **autofocus:** Automatically focuses the button when the page loads.

```
<button type="button" autofocus>Click me</button>
```

HTML Input Form Attributes

1. **formaction**: Specifies the URL to which the form data should be submitted when the user clicks a submit button. This attribute can be applied to <input type="submit">, <button type="submit">, and <button formaction="..."> elements.
   Note: This attribute overrides the action attribute of the <form> element.

```
<form action="/default" method="post">
  <input type="submit" formaction="/custom">
</form>
```

2. **formenctype**: Specifies the encoding type for form data when the form is submitted. It can take values such as "**application/x-www-form-urlencoded**", "**multipart/form-data**", or "**text/plain**".
   Note: This attribute overrides the enctype attribute of the <form> element.

```
<form action="/submit" method="post" enctype="multipart/form-data">
  <input type="submit" formenctype="text/plain">
</form>
```

3. **formmethod**: Specifies the HTTP method to be used when the form is submitted. It can take values like "get" or "post".
   Note: This attribute overrides the method attribute of the <form> element.

```
<form action="/submit" method="post">
  <input type="submit" formmethod="get">
</form>
```

4. **formtarget**: Specifies the target window or frame in which the form response should be displayed. It can take values like "blank", "self", "parent", or "top".
   Note: This attribute overrides the method attribute of the <form> element.

```
<form action="/submit" method="post">
  <input type="submit" formtarget="_blank">
</form>
```

5. **formnovalidate**: When present, this boolean attribute disables form validation before submission. It can be used to submit the form even if some fields do not meet validation criteria.

```
<form action="/submit" method="post">
  <input type="submit" formnovalidate>
</form>
```

CONCLUSION:

In conclusion, the attributes and elements discussed in the context of HTML forms provide developers with a powerful toolkit to create interactive and customisable user interfaces. Starting with the <form> tag as the foundational element, various attributes such as action, method, and enctype define how form data is submitted and processed on the server. Attributes like autocomplete, novalidate, and target offer control over the form's behaviour and user experience.

Further customisation is achieved through input elements with diverse types such as text, number, date, and file, each enriched with attributes like readonly, disabled, and placeholder. Validation is enhanced using attributes like required, maxlength, and pattern. Additionally, formaction, formenctype, formmethod, and formtarget allow fine-tuning form behavior on a per-element basis, offering flexibility in form submission and response handling. The novalidate attribute, whether applied to the entire form or individual elements, provides an option to override default browser validation.

Collectively, these HTML form elements and attributes empower developers to create forms that are not only visually appealing but also functional, accessible, and tailored to specific requirements. As web technologies continue to evolve, these tools remain fundamental for crafting seamless and user-friendly interactions on the web.

REFERENCES:

Explore these tags more by going through the links below.

1. Form Tag
2. Form Input Types
3. Input Attributes
4. Form Elements