<input>: Checkbox Type: The HTML

<input> element with type="checkbox" creates a single checkbox option. To group checkboxes under the same category, assign them the same name attribute. Since checkboxes allow multiple selections, users can choose multiple options within the same category.

```
<input type="checkbox" name="fruit"
value="apple"> Apple<br>
<input type="checkbox" name="fruit"
value="orange"> Orange<br>
<input type="checkbox" name="fruit"
value="banana"> Banana<br>
```

<textarea> Element

When you need a multi-line text input, such as for comments, the <textarea> element is employed. Adjust the size of the textarea using the rows and cols attributes. Although users can resize the textarea, its initial size is determined by these attributes. Unlike single-line inputs, <textarea> has both opening and closing tags, and its content lies between them, akin to <p> elements.

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

<form> Element

Use the HTML <form> element to gather and transmit data to an external destination. Within a <form>, include various input elements. Upon submission, the form forwards the input data to the location specified by the action attribute.

```
<form action="/submit_form"
method="post">
  <input type="text" name="username"
placeholder="Username"><br>
  <input type="password"
name="password"
placeholder="Password"><br>
  <input type="submit" value="Submit">
</form>
```

<input>: Number Type

Define input fields as type="number" to restrict entries to numerical values and specific characters. For instance, a form containing an input with type="number" and name="balance" will transmit data in key-value pairs upon submission.

```
<form action="/submit_balance"
method="post">
  Balance: <input type="number"
name="balance" min="0"><br>
  <input type="submit" value="Submit">
</form>
```

<input> Element

Employ the <input> element to generate diverse input fields like text fields or checkboxes on a webpage. The type attribute dictates the field's appearance. For example, the provided code will display a text input and a checkbox.

<input>: Range Type

By setting type="range", an HTML <input> creates a slider enabling users to select values within a defined range. Customize the slider's parameters using min, max, and step attributes. Range sliders are suitable for visually adjusting values, such as volume control.

```
<input type="range" name="volume"
min="0" max="100" step="1">
```

<select> Element

Utilize the <select> element to establish dropdown lists. Populate the list with <option> elements, allowing users to choose a single option at a time. When the form submits, the selected option's name and value pair are transmitted.

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

Submitting a Form

After gathering data in a form, specify where it should be sent using the action attribute. The method attribute determines how the data is handled upon submission, employing HTTP verbs.

<input>: Text Type

HTML <input> elements support text input with type="text", presenting a single-row input field for textual entries. Upon form submission, the input's name and value attributes form a key-value pair.

```
<form action="/submit_name"
method="post">
  Name: <input type="text"
name="name"><br>
  <input type="submit" value="Submit">
</form>
```

<datalist> Element

Combine an <input> with a <datalist> to create a basic search or autocomplete feature. The datalist stores predefined options shown in a dropdown when users interact with the input field.

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

<input>: Radio Button Type

Utilize type="radio" attribute to create single radio buttons. Group related radio buttons by assigning the same name attribute. Upon selection, the chosen option's name and value pair are transmitted when the form submits.

```
<input type="radio" name="gender"
value="male"> Male<br>
<input type="radio" name="gender"
value="female"> Female<br>
<input type="radio" name="gender"
value="other"> Other<br>
```

Submittable Input

Transform an <input> into a submit button by setting type="submit". By default, this button submits the associated form, executing its action. Customize the button's text using the value attribute.

```
<form action="/submit_button"
method="post">
  <input type="submit" value="Submit">
</form>
```

<input> name Attribute

Assigning a name attribute to an <input> facilitates data submission in key-value pairs. Note that the name differs from the ID in terms of form submission.

```
<form action="/submit_data"
method="post">
  <input type="text"
name="first_name"><br>
  <input type="text"
id="first_name"><br>
  <input type="submit" value="Submit">
</form>
```

<label> Element

Link <label> elements with <input> fields using matching id and for attributes. Clicking the <label> focuses on the associated <input> field.

```
<form action="/submit_password"
method="post">
  <label
for="password">Password:</label>
  <input type="password" id="password"
name="password"><br>
  <input type="submit" value="Submit">
</form>
```

<input> Password Type

Designate type="password" for sensitive data entry, obscuring characters as they're typed. Upon submission, the actual value, not the obscured version, is transmitted.

```
<form action="/submit_login"
method="post">
  Username: <input type="text"
name="username"><br>
  Password: <input type="password"
name="password"><br>
  <input type="submit" value="Login">
</form>
```

required Attribute

Enforce field completion by adding the required attribute to input fields, ensuring they contain a value before submission.

```
<form action="/submit_email"
method="post">
  Email: <input type="email"
name="email" required><br>
  <input type="submit" value="Submit">
</form>
```

max Attribute

Limit input values by setting a maximum value using the max attribute, useful for numerical inputs to prevent entries beyond a defined limit.

```
<form action="/submit_age"
method="post">
  Age: <input type="number" name="age"
max="120"><br>
  <input type="submit" value="Submit">
</form>
```

maxlength Attribute

Define the maximum character count for text input fields with the maxlength attribute, restricting user input accordingly.

```
<form action="/submit_message"
method="post">
  Message: <input type="text"
name="message" maxlength="100"><br>
  <input type="submit" value="Submit">
</form>
```

pattern Attribute

Employ the pattern attribute with a regular expression to validate input values against specific criteria.

```
<form action="/submit_zip"
method="post">
  Zip Code: <input type="text"
name="zip" pattern="[0-9]{5}"><br>
  <input type="submit" value="Submit">
</form>
```

minlength Attribute

Specify a minimum character count for text input fields using the minlength attribute, ensuring entries meet a specified length requirement.

```
<form action="/submit_username"
method="post">
  Username: <input type="text"
name="username" minlength="4"><br>
  <input type="submit" value="Submit">
</form>
```

HTML Form Validators

HTML forms support various validation mechanisms, including minimum and maximum value checks, length constraints, etc., applied through input field attributes.

```
<form action="/submit_form"
method="post">
  <label
for="username">Username:</label>
  <input type="text" id="username"
name="username" required><br>

  <label for="age">Age:</label>
  <input type="number" id="age"
name="age" min="18" max="100"
required><br>

  <label for="email">Email:</label>
  <input type="email" id="email"
name="email" required><br>

  <label
for="password">Password:</label>
  <input type="password" id="password"
name="password" minlength="8
```

min Attribute

Utilize the min attribute to set a minimum value for number input fields, restricting entries to values above a defined threshold.

```
<form action="/submit_quantity"
method="post">
  Quantity: <input type="number"
name="quantity" min="1"><br>
  <input type="submit" value="Submit">
</form>
```

Exercise:
Create a registration form for a website. The form should include the following fields:

1. Username (required, minimum length of 5 characters)
2. Email (required, valid email format)
3. Password (required, minimum length of 8 characters)
4. Confirm Password (required, must match the password field)
5. Date of Birth (required, must be at least 18 years old)
6. Gender (optional)
7. Terms and Conditions (required, must be checked)

Your task is to create the HTML code for this registration form, including appropriate input fields and validators.

Guide to Creating a Registration Form in HTML: Helpful Hints and Tips

1. Start with the Structure: Begin by setting up the basic structure of the HTML document using the `<!DOCTYPE html>` declaration and the `<html>`, `<head>`, and `<body>` tags.

2. Create the Form Tag: Use the `<form>` tag to create the registration form. Set the `action` attribute to specify where the form data should be submitted and the `method` attribute to specify the HTTP method (usually `post`).

3. Add Input Fields: Inside the `<form>` tag, add input fields for each piece of information you want to collect from the user (e.g., username, email, password). Use the appropriate `<input>` types (`text`, `email`, `password`, etc.).

4. Set Required Fields: Use the `required` attribute to make certain fields mandatory. This ensures that users must fill in these fields before they can submit the form.

5. Define Input Field Constraints: Utilize attributes like `minlength`, `min`, `max`, and `pattern` to set constraints on input fields. For example, you can specify a minimum length for the password field or require a certain format for the email address.

6. Add Labels: Use `<label>` tags to provide labels for each input field. This improves accessibility and usability by associating each label with its corresponding input field.

7. Include Additional Fields: Consider including additional fields such as confirm password, date of birth, gender, and terms and conditions. Use appropriate input types (`date`, `radio`, `checkbox`) and validators for each field.

8. Test Your Form: Once you've created the form, test it by filling out each field and submitting the form. Make sure it behaves as expected and validates input correctly.

By following these hints, you should be able to create a registration form that collects the necessary information from users and validates their input effectively. Feel free to refer back to the example provided earlier for guidance.