Lesson:

HTML Forms

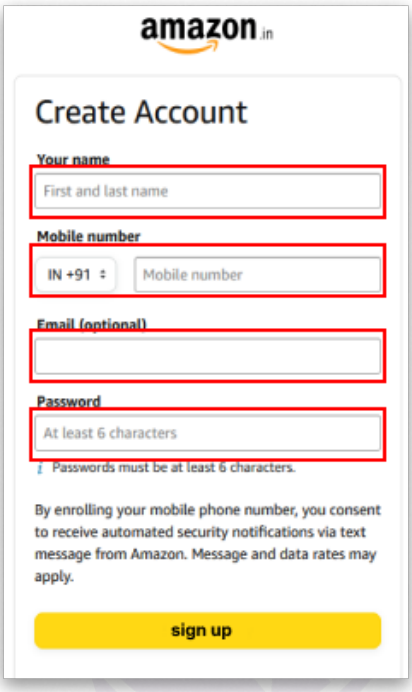
PW SKILLS

Topics Covered

- Introduction to HTML form
- HTML form attributes
- HTML form elements
- HTML form submission
- Fieldset and legend

Introduction to HTML form

An HTML form is like a virtual questionnaire that you find on websites. It allows you to interact with a website by filling in information and submitting it



Imagine you're signing up for a new website account. The website asks you for your name, phone number, email, and password. You provide this information by typing it into boxes on the website. These boxes are called "input fields." Once you've filled out everything, you click a button that says "Sign Up" or "Submit."

When you click that button, the website takes all the information you provided in the input fields and sends it to the website's server, kind of like sending a letter. The server then processes that information and does whatever the website needs it to do, like creating your account with the details you provided.

Basic form code example

To declare a form in HTML, you use the <form> element. Inside the <form> element, you place various form elements to collect data from users. Here's the basic syntax for declaring a form and placing form elements inside it:

```
Unset
    <h1>Contact Form</h1>
    <form action="/submit_form" method="post">
        <!-- Form elements go here -->
    </form>
```

Form attributes

Behind the scenes, HTML forms use attributes to control the behaviour and how the data is processed. Let's understand these attributes using our Amazon sign-up example:

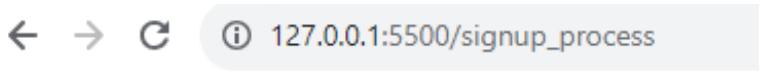
- **action**

The action attribute is like specifying the address where you want to send your form. It tells the browser where to submit the form data. The action attribute might look like this: action="/signup_process". This means the form data will be sent to the "/signup_process" URL on some website server.

Example:

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

As we click on “submit” button, you will see action url on browser url,



- **method**

The method attribute determines how the form data is sent to the server.

Syntax

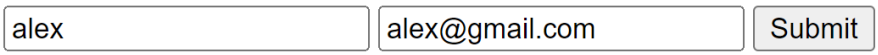
```
Unset
<form method="get"></form>
<form method="post"></form>
```

The most common methods are "**GET**" and "**POST**". In our Amazon sign-up form, you can use the **method="POST"** attribute, which means the data is sent securely and privately.

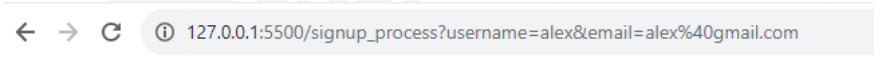
Note: In the case of the GET method, after submitting the form, form data is visible in the address bar. But the POST method prevents form data from appearing in the address bar after the form has been submitted, via sending data in the request body.

Example:

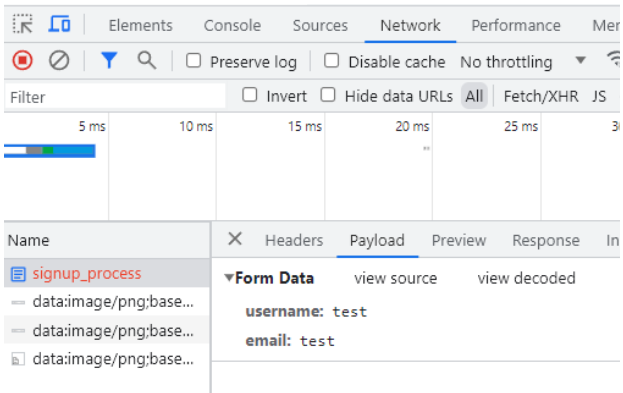
```
Unset
<form action="/signup_process" method="get">
    <input name="email" />
    <input name="username" />
    <input type="submit" />
</form>
```



As we click on “submit” button, you will see action url on browser url, you will notice browser url changes and all form values are visible in url



In case of method=”post”, form values will be sent to server as part of body(payload), it can be visulised in network tab of browser dev tool as shown below,



- **target**

The action attribute accepts different values, each influencing how the form response is handled.

The target attribute can have one of the following values:

- **_self**: This is the default value if the "target" attribute is not specified. It means that the form response will be displayed in the same window or tab where the form was submitted.
- **_blank:** Using **target="_blank"** opens the form submission response in a new browser window,
- _parent: Using frames or iframes, **target="_parent"** Opens the form submission response in the parent form.
- **_top:** If the form is within a nested set of frames or iframes, **target="_top"** Cause the form response to open in the top-level browsing context, replacing any frames. If the form is not within a frame or iframe, it behaves the same as **_self.**

Syntax

```
Unset
<form target="_self"></form>
<form target="_blank"></form>
<form target="_parent"></form>
<form target="_top"></form>
```

Example: In previous example, added target="_blank"

```
Unset
<form action="/signup_process" method="post" target="_blank">
    <input name="username" />
    <input name="email" />
    <input type="submit" />
</form>
```

- **autocomplete**

Controls whether the browser should remember and autofill form input values. Values can be "on" (default) or "off"

Example:

```
Unset
<form action="/signup_process" method="post" >
    <label>Name</label>
    <input name="username" autocomplete="on" />
    <input type="submit" />
</form>
```

As you see above, browser is suggesting text based on input history.

- **enctype**

specifies how the form data should be encoded before sending it to the server. There are three common enctype values:

- **application/x-www-form-urlencoded** (Default): It encodes form data in a URL-like format, suitable for regular form submissions with text input fields.
- **multipart/form-data:** Used for forms with file uploads. It properly handles binary data, like images or documents.
- **text/plain:** Rarely used, sends form data as plain text without special formatting.

Syntax

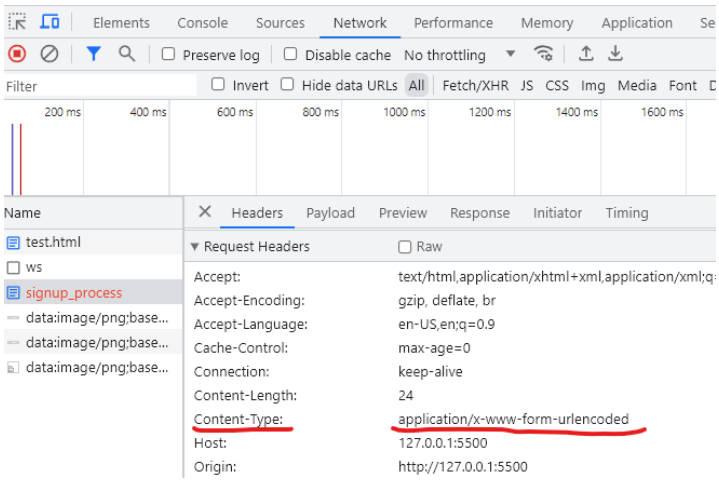
```
Unset
<form enctype="application/x-www-form-urlencoded"></form>


 <form enctype="multipart/form-data"></form>
 <form enctype="text/plain"></form>
```

Choose the appropriate enctype based on your form content. For regular forms, you can stick to the default. For file uploads, use **multipart/form-data**.

Example:

```
Unset
<form action="/signup_process" method="post"
enctype="multipart/form-data">
      <input name="filename" />
      <input name="file" />
      <input type="submit" />
</form>
```

Upon Submitting the form, you can see enctype as Content-Type in Reaquest Headers in Netowork tab of browser dev tool.



- **name**

Name of the form, it must a unique non-emply string.

Syntax

```
Unset
<form name="signup"></form>
```

- **novalidate**

A boolean attribute, which tells the browser to not validate field values. We will study about validation in next lesson.

Syntax

```
Unset
<form novalidate></form>
```

- **rel**

It defines the relationship between current document and the linked document. Its values could be,

- **external**: Indicates that the referenced document exists outside the scope of the current website.
- **help:** Provides a link to a document designed to offer assistance or guidance.
- **license:** Directs to copyright-related information associated with the document.
- **next:** Refers to the subsequent document within a series or selection.
- **nofollow:** Links to a document without an endorsement, often used for paid links.
- **noreferrer:** Instructs the browser to not send an HTTP referrer header.
- **nopener:** Advises the browser to open the link in a way that prevents the new page from controlling the originating page.
- **prev:** Points to the preceding document in a series or selection.
- **search:** Connects to a search tool designed for exploring the document

Example:

```
Unset
<form rel="noreferrer nofollow" action="mypage.php">
   Username <input type="text" /><br /><br />
   Password<input type="password" /><br />
   <input type="submit" value="submit" />
</form>
```

In above example, rel=”noreferrer nofollow nopener”, protects your site users from potentially malicious external links.

Form elements

Form elements in HTML are used to create interactive sections where users can input data or make selections. Forms are a fundamental part of web development, allowing users to submit information to a server for processing. Here are some elements that form contains:

1. Input and label Tag

The **<input>** HTML element is one of the most important form elements which is used to accept data/input from the use

A wide variety of types of input data and control widgets are available.

Example:

```
Unset
<label for="email">Enter Email: </label>
<input type="email" name="email" id="email" required>
```

Above code defines an **email input** for entering an email address. It also checks for a valid email address. Here <label> tags help users to know the purpose of the form control element (input), and for attribute specifies the target control element, which allows users to click on the label to focus on the associated form control.

2. Fieldset and Legend Tags

The **<fieldset>** tag is used to group related form elements together. It will give a border to the grouped elements.

The **<legend>** tag is **used within a <fieldset> to provide a caption or title** for the fieldset. It is often used to group related form controls together and provide context or instructions for the user. The text within the **<legend> tag** is typically styled to stand out from the rest of the form controls, making it easier for the user to identify the purpose of the group of controls.

Let's take the **form submission** example, instead of <h1> add fieldset and legend.

index.html

```
Unset
<body>
    <fieldset>
        <legend>sign up</legend>
        <form action="/signup_process" method="post"
name="sign-up>
        <label for=" username">Username:</label>
            <input type="text" id="username" name="username"
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" required><br>


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

             <input type="submit" value="Sign Up">
    </fieldset>
    </form>
</body>
```

Browser output

3. Output Tag

We use <output> tag, to show result of any calculation or any user action,

Syntax

```
Unset
<output name="" for="" form=""></output>
```

Here, **name** represents elements name, **for** attribute has space separated ids, which were involved in producing the output, and **form** attribute has id of **<form>** tag, which is the correct form owner of **output**.

Example:

```
Unset
<form id="add_form">
   <input type="number" id="x" name="x" value="10" /> +
   <input type="number" id="y" name="y" value="20" /> =
   <output name="result" for="x y" form="add_form"></output>
</form>
```

Output

10 + 20 = 30

In above code, <output> tag name is “result”, its final results is depend on values of x and y, amd it is associated with form “add_form”.

Note: In above code, we have not written logic for adding two numbers, because we have not studied javascript yet.

4. Button Tag

It offers submitting a form or triggering specific actions, like resetting the form fields.

Syntax

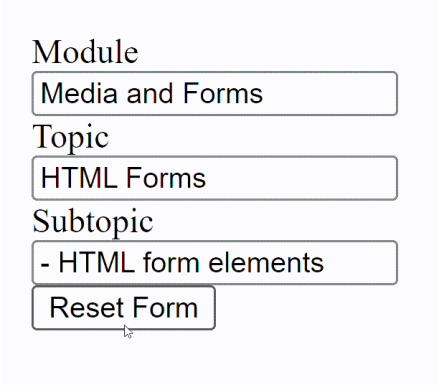
```
Unset
<button type="">Button Text<button/>
```

Here, type attribute can be either submit, reset, or button, depending on use cases

- **submit:** When this value is used, the button initiates the submission of form data to the server.
- **reset:** Selecting this value causes the button to reset all associated input fields to their original values.
- **button:** Opting for this value means, button will not have predefined behavious, like reset or submit

Example: Reset form,

```
Unset
<form>
    <label for="module">Module</label><br />
    <input type="text" id="module" name="module" required
/><br />

    <label for="topic">Topic</label><br />
   <input type="text" id="topic" name="topic" required /><br
/>

   <label for="subtopic">Subtopic</label><br />
   <input type="text" id="subtopic" name="subtopic" required
/><br />

   <!-- Reset Button -->
   <button type="reset">Reset Form</button>
</form>
```



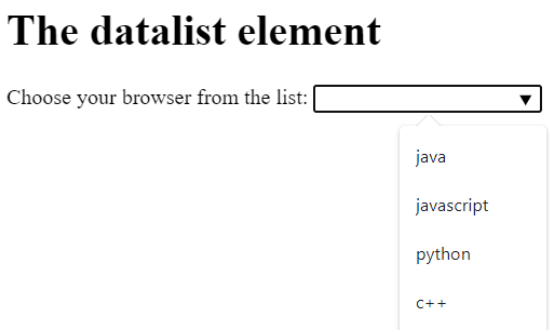
As you see above, on clicking button with type “reset”, sets all field to its original value.

5. Datalist Tag

The **<datalist>** tag is used to provide an **autocomplete** feature for **<input>** elements (We will study about input tags in detail in next lesson). Users will see a drop-down list of predefined options as they input data

The <datalist> element's **id** attribute must be equal to the **<input>** element's list attribute (this binds them together).

```
Unset
<h1>The datalist element</h1>
<label for="language">Choose your browser from the
list:</label>
<input list="languages" name="language" id="language" />
    <datalist id="languages">
      <option value="java"> </option>
      <option value="javascript"> </option>
      <option value="python"> </option>
      <option value="c++"> </option>
</datalist>
```



6. Textarea Tag

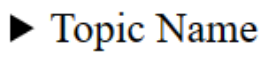
The <textarea> element is used to create a multi-line text input field that allows users to enter and edit text. The <textarea> element can be useful when you want users to provide longer comments, descriptions, or other types of textual input.

7. Details Tag

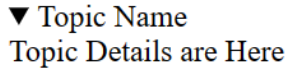
The **<details>** HTML element generates a disclosure widget that displays information only when the widget is switched to the open state.

```
Unset
<details>
    <summary>Topic Name</summary>
    Topic Details Here
</details>
```

Close state



Open State

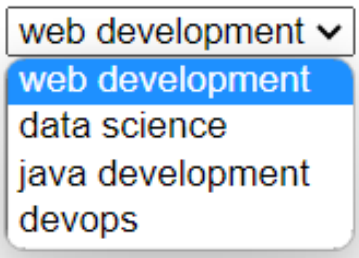


NOTE: All the input elements types with attributes will be discussed in the next section in detail.

8. Select and Options

In HTML, the <select> element is used to create a dropdown list, and the <option> element is used to define the individual options within that dropdown list. This Allows users to select one option from a dropdown of multiple options. Let's understand with an example of Dropdown,

```
Unset
<select name="selected">
      <option value="web development">web development</option>
      <option value="data science">data science</option>
      <option value="java development">java
development</option>
      <option value="devops">devops</option>
</select>
```



In the above example, it allows users to select one option from the dropdown options.

Form Submission

The <input> element with type="submit" attribute is the most common way to create a submit button in a form. When the user clicks on this button, the form data is automatically submitted to the URL specified in the action attribute of the <form> element, using the HTTP method specified in the method attribute.

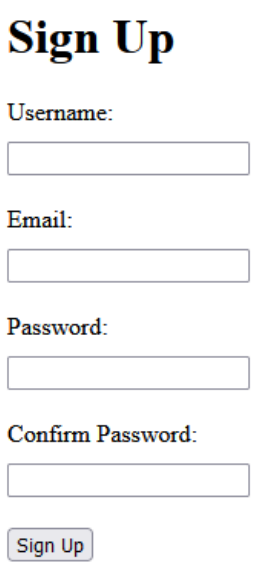
Let's create a simple sign-up example, submit the form, and also observe the request data in the network section of our browser.

index.html

```
Unset
<body>
    <h1>Sign Up</h1>
    <form action="/signup_process" method="post"
name="sign-up>


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

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

         <input type="submit" value="Sign Up">
    </form>
 </body>
```

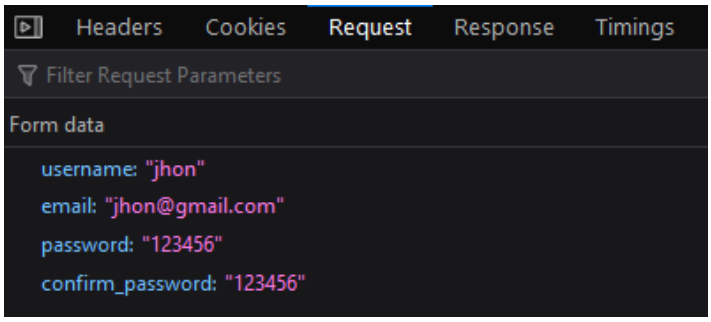
In the above code example, we create a basic sign-up that takes user name, email, password and confirm password, and finally, we add the submit button using input submit type (<button type = ”submit”>… </ button> is an alternative way to create a submit button.)

Browser output

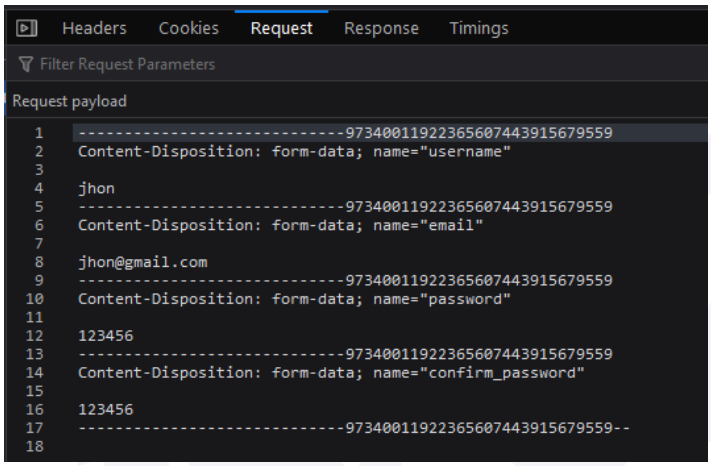


In the next step, fill in the required data in a sign-up form and hit the sign-up submit button.

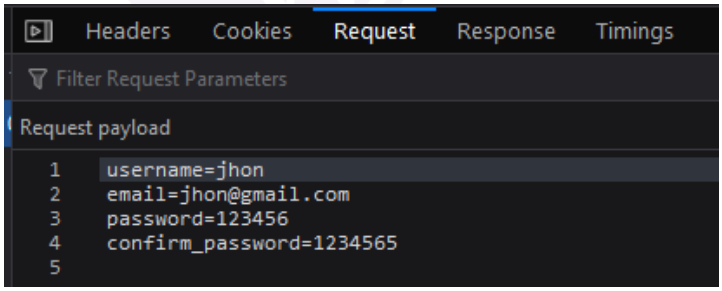
Request object case of **enctype = "application/x-www-form-urlencoded"** (default) form attribute



Request object case of **enctype = "multipart/form-data"** form attribute



Request object case of **enctype = "text/plain"**



We have created a basic HTML form to demonstrate the concept of form submission. However, we understand that there are various form elements used in HTML, and we will explore them in the next section to provide a comprehensive understanding of each element's usage.