COMP 2406 A : Fall 2020

Written Tutorial : Forms and Receiving User Input

Author: Michael Grewal

Introduction

Websites that receive user input is fundamental for building interactive web experiences. This has a wide variety of applications such as user profile creation, posting to blogs and social media, creating chat rooms, building databases, paying taxes online, conducting online banking/business, etc… Can you think of others?

Objectives

- Build a webpage that contains a form.
- Display the contents received on the form to another webpage.
- Practice HTML/CSS while utilizing some DOM Javascript code.
- By the end of this tutorial we will have created a typical *Register for an account* login form.

Expectations

- You should know about tags, attributes, and general HTML structure.

Problem 1 : Setup

1. Create a folder and create a file index.html. This will serve as our main page and hold our register form.
2. Copy and paste the following starter HTML code to begin or use the starterhtml.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Register Form</title>
</head>
<body>

</body>
</html>
```

3. Create a file called submit.html. This page will display our submitted information.
4. Copy and paste the same code as above (Step #2) and change the <title> to something appropriate.

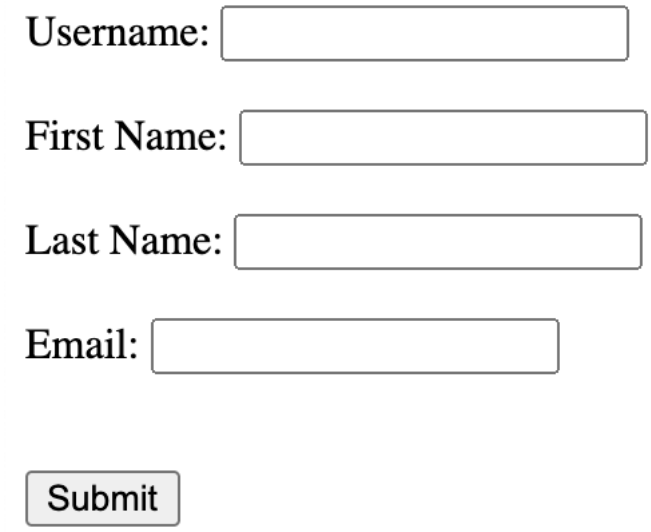
When you open your webpage, you should see a blank page. We have not added any content to the body yet.

Problem 2: Create Form Content

1. In the <body> section of our index.html we will include the tags <form> and close it with </form>. This will create a form on our page and allow any information within this section to be submitted.
2. Within the <form> tags, we will type <input> and <label></label> tags so that we have an input field for a username, and a label for it as well. The <input> tag allows the user to input data, and the <label> tags will allow the user to know what information is being asked.
3. Make the label "Username: " by typing that between the <label></label> tags (no quotations, and include the attribute for="username".
4. Give the <input> tag the attribute name="username" and id="username".
5. Next, we will add a submit button by typing <button>Submit</button>. Your code should look like this:

```
<form>
    <label for="username">Username: </label>
    <input name="username" id="username">
    <button>Submit</button>
</form>
```

6. Add some more <label> and <input> tags for more information (First Name, Last Name, email).
7. Save your HTML and open it in the web browser.
8. Your webpage should look like this:



Hint: Experiment with and
 tags to get the Submit button at the bottom.

Load your webpage, type a username and click the Submit button. What happens? Your typed information vanishes, and the page reloads. This is because the information has been submitted! However, this information is just being sent to the current webpage, which is kind of useless and unpractical. Next we will discover how to send the information to somewhere useful.

Problem 3: Submitting information using "GET" method

1. Give the <form> tag the attribute action="submit.html". The action tag provides a location for the submitted information.
2. Give the <form> tag the attribute method="GET". The "GET" method appends the information to the URL and allows it to be passed to another page on our site. We will see more on this later…

Hint: There is another method="POST" which sends information to a server, and allows it to be stored in a database.

Now that we have routed our form information to the submit.html file, let's go ahead and write some Javascript to process the information.

Problem 4: Processing the information

Before processing the information, our submit.html file needs a bit of work.

1. In the <body>, create a container with the id="results" attribute. This container will contain the results submitted by the form with the help of some Javascript.
2. After the , include <script></script> tags with the following Javascript code inside:

```
<script>
    let resultsList = document.getElementById("results");
    new URLSearchParams(window.location.search).forEach((value, name) => {
        resultsList.append(`${name}: ${value}`)
        resultsList.append(document.createElement("br"))
    })
</script>
```

Hint: Make sure to use `backticks` for the ${name}: ${value} string formatting.

The code above is grabbing the "results" container and holding it as resultsList variable. Then it's going to scan the URL for each value and name combination provided by the form and insert it into the "results" along with the given string formatting.
Read through the code slowly and carefully so that you understand what it's doing.

3. Include a hyperlink to go back to the form page by using <a> tags:

```
<a href="./index.html">Back To Form</a>
```

Load your webpage, and try using your form. Click submit and notice the URL. Do you see the name=value pairs?

```
/submit.html?username=thetraveler31&firstname=Eris&lastname=Morn&email=destiny2%40b...
```

Problem 5: Add CSS Style and More

This section is meant to be fun!
Explore some CSS styling to make your form more appealing and add more functionality.

1. Try to format your form differently.
2. Add color.
3. Can you include more user input fields? Perhaps some with checkboxes, radio buttons, or drop down menus? By default the <input> type is text, but you can explicitly specify the type by including the type attribute.
4. You can also create functionality to submit files, and create passwords.

Below is an example. It's a form to register for an online gaming forum. The left is the form webpage, and the right is the resulting information.