Web Programming

Tutorial 5

To begin this tutorial, please download `tut05-starter.zip` from the course website. When you finish, zip all your deliveries to submit to this tutorial's submission box. The zip file's name should follow this format: `tclass_sid.zip` where `tclass` is your tutorial class name (e.g. `tut01`, `tut02`, `tut03`, etc.) and `sid` is your student's ID (e.g. `2101040015`).

Activity 1 – From `promise/then` to `async/await`

In the `activity_1` folder, edit `async-await.js`. Replace all usage of `.then` / `.catch` in the code with `async` / `await` so that the output is the same. You will need to define at least one `async` function.

Open `index.html` in the browser and use the console to check your work.

Activity 2 – JS Object

Given the following JS Object:

```
let data = {
  "age": 21,
  "age-median": 20,
  "company": "H City Zoo",
  "ticket-prices": {
    "adult": 9.95,
    "child": 5.95
  },
  "people": [{
    "first_name": "Aylmar",
    "last_name": "Avison",
    "email": "aavison0@scribd.com",
    "favorite_animal": "Honey badger"
  },
  {
    "first_name": "Jean",
    "last_name": "Jorg",
    "email": "jjorg1@i2i.jp",
    "favorite_animal": "Elephant, asian"
  },
```

```
  {
    "first_name": "Bernardo",
    "last_name": "McDuff",
    "email": "bmcduff2@purevolume.com",
    "favorite_animal": "Cockatoo, slender-billed"
  }]
};
```

Complete the table below mapping JavaScript code to its value or mapping the value to its JavaScript code.

JavaScript	Value
`data.company`	
`data['company']`	
	`'jjorg1@i2i.jp'`
`data['ticket-prices'].adult`	
	`5.95`
`data.ticket-prices`	
`data.age-median`	
	`20`

Activity 3 – AJAX Pets

Given the `ajaxpets` folder as a starter, create an AJAX-powered gallery of pet images that allows you to switch between kitty and puppy images without reloading the page.

You are provided with a Pets API:

Service URL: `http://103.159.50.237/wpr/api/pets/index.php`

Query Parameters (required): `?animal=<value>`

Details: `animal` is the name of the query parameter you need to assign a value to. This API recognizes either a value of `puppy` or `kitty`.

Example Request (with puppy as the value):
`http://103.159.50.237/wpr/api/pets/index.php?animal=puppy`

Pets API Response comes in plain text and has the following format:

```
https://path/to/pet/img0.jpg

https://path/to/pet/img1.jpg

https://path/to/pet/img2.jpg

https://path/to/pet/img3.jpg

...
```

Ajax Pets Implementation

The provided starter code includes a module-pattern template we've been using to get you started, named `ajaxpets.js`. You will need to implement the JavaScript to incorporate AJAX and make a request with the Pets API URL with the parameter `animal` of value `kitty` or `puppy`, depending on which radio button is selected.

When a request returns a response successfully with the plain text response of image paths, write JS to add `img` tags as children to the `#pictures` div for each image path returned on a new line.

Hint: you should listen for the `change` event of the radio buttons.

Activity 4 – Making a POST request with Fetch

Given the `login` folder as starter. Your task is to update `post.html` so that the input elements have the proper attributes (both input elements should be required to be filled for the form to be submitted). Update `post.js` to get user inputs, send a POST request to the Login API and show the result returned by the API. Upon a the form being submitted, the response from the API should be added to the `#response` container.

This login page has only one valid username/password combo (username: `rainbowdash`, password: `ponyta`).

Login API Documentation

Service URL: `http://103.159.50.237/wpr/api/login.php`

Request method: POST

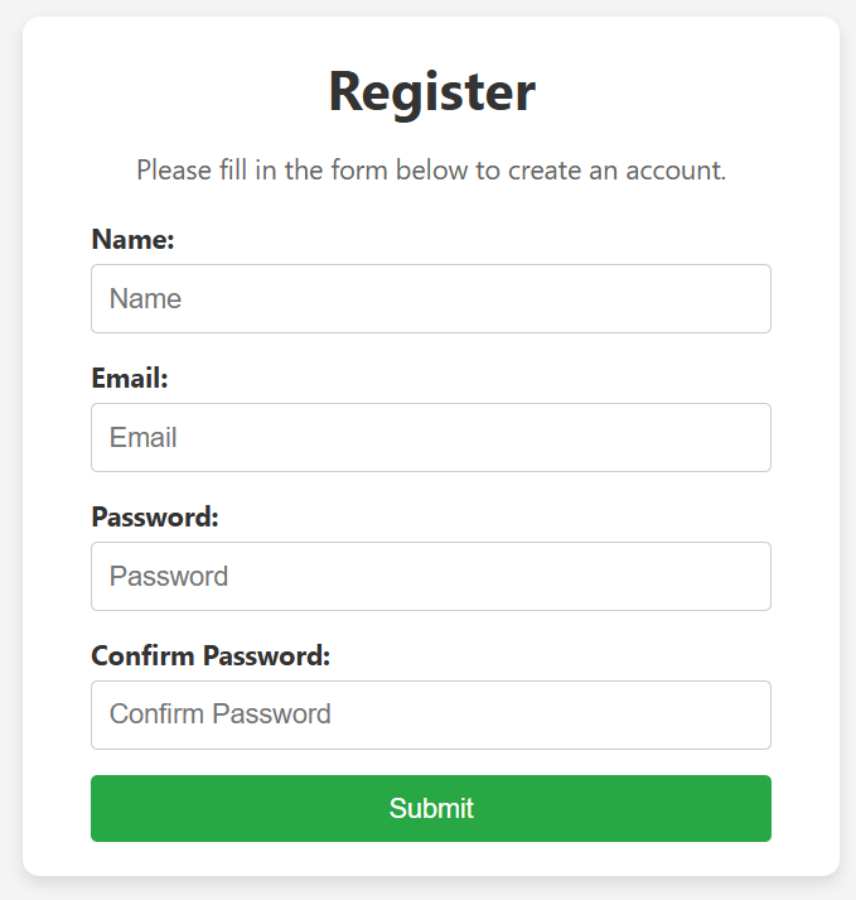
Response format: plain text

Body Parameters (2 params):

`user` : Obtained from the first input element for the username

`password` : Obtained from the second input element for the password

Activity 5 - Form validation



In the `form_validation` folder, these starter files (HTML, CSS, and part of the JS) provide the basic structure for the form validation exercise. You need to implement the JavaScript logic to validate the form, display error messages, and handle the countdown timer after a successful validation.

You have to do following tasks:

1. Validate the form fields as follows:

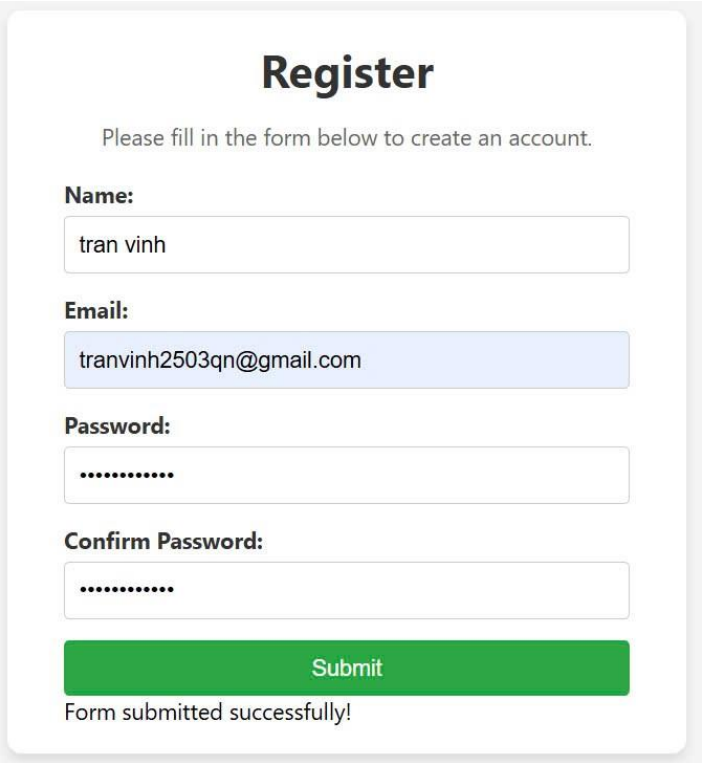
- **Name**: Required, minimum 3 characters

- **Email**: Required, must be a valid email address
- **Password**: Required, minimum 8 characters, must include at least one uppercase letter, one lowercase letter, and one number
- **Confirm Password**: Must match the Password field.

2. Display dynamic error messages as the user inputs data

3. Prevent form submission if any validation fails

3. After successful validation, display a 3-second countdown and then show a success message.



Hints:

- Use regular expressions for email and password validation
- Use "`addEventListener`" for real-time validation
- Use "`preventDefault()`" to stop form submission if validation fails