Hands-on Lab: Develop an Application to Fetch Health Articles using XMLHttpRequest

Estimated time needed: 30 minutes

What you will learn

In this lab, you will explore the dynamic creation of HTML elements using JavaScript, enabling the manipulation and real-time construction of webpage content. You will learn to parse JSON data to extract and structure information, facilitating seamless display on a webpage. Additionally, you will discover how to organize and present data by creating unordered lists.

Learning objectives

After completing this lab, you will be able to:

- **XHR integration proficiency:** Acquire the skill to integrate XMLHttpRequest (XHR) into web applications for fetching external JSON data, facilitating dynamic content retrieval.
- **Dynamic HTML construction:** Develop the ability to dynamically construct the HTML elements using JavaScript, enabling the creation of structured content and layout modification quickly.
- **JSON data manipulation:** Learn to handle and parse JSON data retrieved from external sources, facilitating its structured presentation within a webpage.
- **List generation competence:** Explore the creation of unordered lists to organize and display data, mastering techniques for efficient information organization on webpages.
- **Iterative content rendering:** Understand and implement iterative content rendering, allowing for the dynamic display of multiple articles fetched from JSON data onto a webpage, enhancing the user experience through comprehensive content presentation.

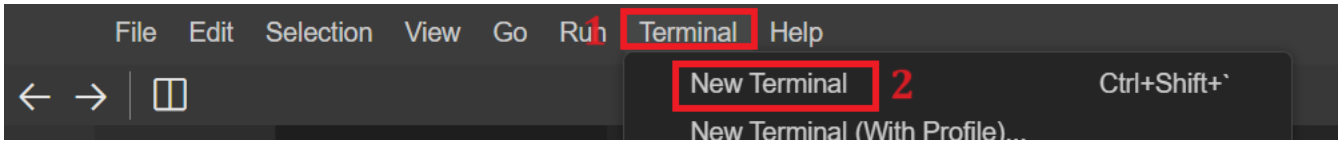
Prerequisites

- Basic Knowledge of HTML and GitHub.
- Web browser with a console (Chrome DevTools, Firefox Console, and so on).

Step 1: Setting up the environment

1. First, you need to clone your main repository in the **Skills Network Environment**. Follow the given steps:

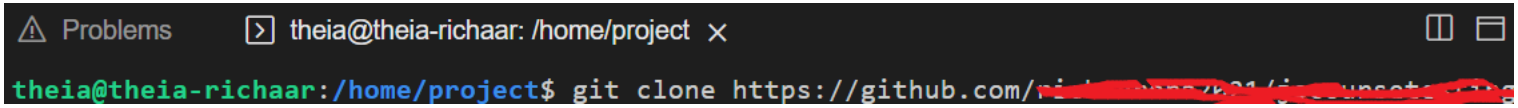
- Click on the terminal in the top-right window pane and then select **New Terminal**.



- Perform `git clone` command by writing given command in the terminal.

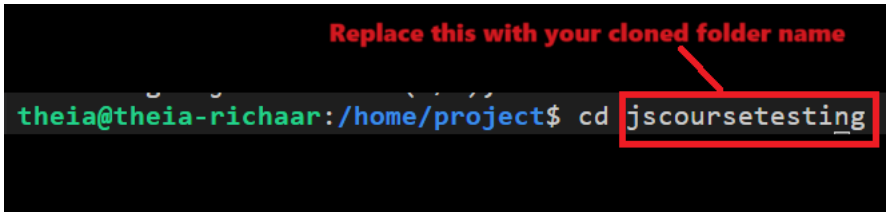
```
git clone <github-repository-url>
```

Note: Put your own GitHub repository link instead of `<github-repository-url>` and it should look like as given below:



- Above step will clone folder for your GitHub repository under project folder in explorer. You will also see multiple folders inside cloned folder.
- Now you need to navigate inside the cloned folder. For this write given command in the terminal:

```
cd <repository-folder-name>
```



Note: Write your cloned folder name instead of `<repository-folder-name>`. Perform `git clone` if you have logged out of **Skills Network Environment** and you cannot see any files or folder after you logged in.

2. Now, select the **cloned Folder Name** folder, right-click on it, and choose **New Folder**. Enter the folder name as **healthArticle**. This will create the folder for you. Then, select the **healthArticle** folder, right-click, and choose **New File**. Enter the file name as **health_article.html** and click OK. This will create your HTML file.
3. Now select **healthArticle** folder again, right click and select **New File**. Enter the file named **health_article.js** and click OK. It will create your JavaScript file.
4. Create the basic template structure of the HTML file by adding the provided content.
   - This HTML structure sets up a webpage titled **Health Articles** with a header and an empty `<div>` element intended to display health-related content dynamically using JavaScript. `<script>` tag is used to include javaScript file in HTML file.
   - For this, you need to include the given code in the **health_article.html** file.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Health Articles</title>
</head>
<body>
  <h1>Health Articles</h1>
  <div id="articles"></div>
<script src="./health_article.js"></script>
</body>
</html>
```

5. Additionally, create one more file in the project folder named **health_article.json**. This JSON file will have the health article data, including article titles, descriptions, and ways_to_achieve. You can access this data through the provided link.

   health_article.json

6. You need to include this data in the **health_article.json** file and save it.

Step 2: Perform Git commands

1. Perform `git add` to add latest files and folder by writing given command in terminal in git environment.

```
git add --a
```

Make sure terminal should have path as follows:

2. Then perform `git commit` in the terminal. While performing `git commit`, terminal can show message to set up your `git config --global` for user.name and user.email. If yes, then you need to perform `git config` command as well for `user.name` and `user.email` as given.

```
git config --global user.email "you@example.com"
```

```
git config --global user.name "Your Name"
```

Note: Replace data within qoutes with your own details.

Then perform commit command as given:

```
git commit -m "message"
```

3. Then perform `git push` just by writing given command in terminal.

```
git push origin
```

- After the push command, the system will prompt you to enter your username and password. Enter the username for your GitHub account and the password that you created in the first lab. After entering the credentials, all of your latest folders and files will be pushed to your GitHub repository.

Note: You should perform `git add`, `git commit` and `git push` whenever you are updating or creating new code.

Step 3: Defining object and variables for XMLHttpRequest

1. Declare variables for XMLHttpRequest and JSON file.

   - Declare a variable named **xhr** to create a new XMLHttpRequest object as follows in **health_article.js** file:

     ```
     var xhr = new XMLHttpRequest();
     ```

   - Create another variable named **url** to define the URL of the JSON file and fetched as follows:

     ```
     var url = './health_article.json';
     ```

Step 4: URL definition and request set up

1. Now, you need to prepare a GET request to the specified URL, which you have saved in a variable named **url** in asynchronous mode in the **health_article.js** file as follows:

   ```
   xhr.open('GET', url, true);
   ```

2. The open method configures an XHR request with the following parameters:

   - 'GET': Specifies the HTTP method used for the request (in this case, a GET request).
   - URL: Represents the URL from where you will fetch the data.
   - True: Indicates if the request is asynchronous (true) or synchronous (false). In this case, it's set to true for asynchronous operation, allowing other scripts to run while the request is processed.

Step 5: Response type specification

1. In this step, you need to inform the XMLHttpRequest object that the expected response from the server should be in JSON format.

2. For this, you need to set the expected response type to JSON format in **health_article.js** file as follows:

```
xhr.responseType = 'json';
```

Step 6: Handling the 'onload' event

1. You need to define what should happen when the data is successfully loaded using `xhr.onload = function() { ... }` function.

2. Inside this function, you need to include:

```
var articles = xhr.response.articles;
var articlesDiv = document.getElementById('articles');
```

- var articles = xhr.response.articles; to retrieve the articles array from the JSON response.
- var articlesDiv = document.getElementById('articles'); to retrieve the HTML element with the ID 'articles' where the fetched content will be displayed.

Step 7: Iterating through articles and constructing HTML

1. Now, you need to iterate health data to fetch on the front page using loops. For this, you need to use the `forEach` array method as follows:

```
articles.forEach(function(article) {
    var articleDiv = document.createElement('div');
    articleDiv.classList.add('article');
    var title = document.createElement('h2');
    title.textContent = article.title;
    var description = document.createElement('p');
    description.textContent = article.description;
    var waysHeader = document.createElement('h3');
    waysHeader.textContent = 'Ways to Achieve:';
    var waysList = document.createElement('ul');
    article.ways_to_achieve.forEach(function(way) {
      var listItem = document.createElement('li');
      listItem.textContent = way;
      waysList.appendChild(listItem);
    });
    var benefitsHeader = document.createElement('h3');
    benefitsHeader.textContent = 'Benefits:';
    var benefitsList = document.createElement('ul');
    article.benefits.forEach(function(benefit) {
      var listItem = document.createElement('li');
      listItem.textContent = benefit;
      benefitsList.appendChild(listItem);
    });
    articleDiv.appendChild(title);
    articleDiv.appendChild(description);
    articleDiv.appendChild(waysHeader);
    articleDiv.appendChild(waysList);
    articleDiv.appendChild(benefitsHeader);
    articleDiv.appendChild(benefitsList);
    articlesDiv.appendChild(articleDiv);
  });
```

2. In the above array method, your code needs to:

- Create HTML elements dynamically for example, `<div>`, `<h2>`, `<p>`, `<h3>`, `<ul>`, `<li>` for each article's title, description,ways_to_achieve, and benefits using `createElement` DOM method as follows:

  ```
  var articleDiv = document.createElement('div');
  ```

- Populate these HTML elements with corresponding content from the fetched JSON data as follows:

  ```
  articleDiv.classList.add('article');
  ```

- Attach these elements to the articlesDiv to display each article's details on the webpage as follows:

  ```
  articleDiv.appendChild(title);
  ```

Step 8: Sending the request

1. You need to send the XMLHttpRequest to fetch the data from the specified URL and include the given code at the end of the JavaScript file.

```
xhr.send();
```

2. To view how your HTML page, right-click the **health_article.html** file after selecting this file, then select "Open with Live Server".
3. The server should start on port 5500, indicated by a notification on the bottom right side.

4. It will open your default browser where you will see **healthArticle** folder name. Click on that folder name and then click again on **health_article.html**.
5. It will open the front page and output will be shown as below.

Health Articles

The Importance of Regular Exercise

This article discusses the numerous benefits of regular exercise, including improved cardiovascular health, stronger muscles, better mental health, and increas

Ways to Achieve:

- Engage in at least 150 minutes of moderate-intensity aerobic activity per week.
- Incorporate strength training exercises for major muscle groups at least twice a week.
- Find physical activities you enjoy, such as walking, cycling, swimming, or dancing.

Benefits:

- Enhanced cardiovascular fitness and reduced risk of heart diseases.
- Stronger muscles and improved bone density.
- Better mood and reduced risk of depression and anxiety.
- Increased lifespan and overall better quality of life.

Healthy Eating Habits for a Balanced Life

Learn about the fundamentals of a balanced diet, including tips on incorporating fruits, vegetables, whole grains, lean proteins, and healthy fats into your mea

Ways to Achieve:

- Consume a variety of colorful fruits and vegetables daily.
- Opt for whole grains like brown rice, quinoa, and whole wheat bread.
- Choose lean sources of protein such as poultry, fish, beans, and nuts.

Practice task

1. In this task, you need to create **XMLHttpRequest** to fetch data for news article.
2. For this you need to create one json file for news article just like you have worked with **health.json**.
3. Create javaScript code using **XMLHttpRequest**. For this you need to create object for xhr and define variable named **url** and assign link for json file.
4. Create javascript code to fetch news article from step 3 to step 6 provided in the previous lab instructions.

5. Then check the output for the same.

6. Perform `git add`, `git commit` and `git push` to push all your latest work.

Summary

1. Setting up fetch request:
    - You have created an object **xhr** to get data to define where the data is present.
    - You asked for data using `xhr.open('GET', url, true);` to prepare to receive data in a specific format (xhr.responseType = 'json';).
2. Using the received data:
    - You created the function `xhr.onload = function() { ... }` to retrieve specific information like articles, and decided where to show them on the webpage using `document.getElementById('articles')`.
3. Putting it all together:
    - For each article, you created webpage parts like **title descriptions** and filled them with data.
    - You then fetched these parts on the webpage and sent the request to get the data `xhr.send();`.