**Assignment - Bootstrap Web Components with DOM Manipulation**

**Assignment Instructions:** Create a custom Web Component that accepts two properties: data and config. Both data and config should be objects.

* **data:** Contains the content or image paths for elements such as heading, subheading, description, image, etc.
* **config:** Provides the class names for each HTML element used in the Web Component.

You should build your Web Component using document.createElement for every element and utilize proper DOM manipulation methods like appendChild, setAttribute, and innerText to structure your component.

**Folder Structure:**

The folder structure for your component should be as follows:

less

Copy code

component-name/

│

├── script.js // Main file containing the Web Component class

├── index.html // Example usage of your Web Component

└── form.json // Formly form configuration file with data and config input fields

**Details of Each File:**

* **script.js**: Contains the definition of your custom Web Component using the document.createElement method for creating elements and other DOM methods to manipulate them.
* **index.html**: A standalone HTML file that demonstrates how to use your Web Component. Include examples where the component is initialized with different data and config.
* **form.json**: A JSON configuration file for a Formly form with fields to accept data and config as inputs. This file should be structured to allow users to input different values for the component.

**Sample Code: Card Component**

Here’s a sample card component to help you get started:

**script.js**

javascript

Copy code

class CardComponent extends HTMLElement {

constructor() {

super();

// Create container element

const cardContainer = document.createElement('div');

// Set default classes

cardContainer.className = 'card';

// Create image element

const imgElement = document.createElement('img');

imgElement.className = 'card-img-top';

imgElement.alt = 'Card image';

cardContainer.appendChild(imgElement);

// Create card body container

const cardBody = document.createElement('div');

cardBody.className = 'card-body';

// Create title element

const cardTitle = document.createElement('h5');

cardTitle.className = 'card-title';

cardBody.appendChild(cardTitle);

// Create text element

const cardText = document.createElement('p');

cardText.className = 'card-text';

cardBody.appendChild(cardText);

cardContainer.appendChild(cardBody);

// Append the card container to the component root

this.appendChild(cardContainer);

}

connectedCallback() {

const data = JSON.parse(this.getAttribute('data')) || {};

const config = JSON.parse(this.getAttribute('config')) || {};

// Setting data to elements

const imgElement = this.shadowRoot.querySelector('img');

const cardTitle = this.shadowRoot.querySelector('.card-title');

const cardText = this.shadowRoot.querySelector('.card-text');

imgElement.src = data.image || '';

cardTitle.textContent = data.title || 'Default Title';

cardText.textContent = data.description || 'Default Description';

// Setting config to elements

imgElement.className = config.imageClass || 'card-img-top';

cardTitle.className = config.titleClass || 'card-title';

cardText.className = config.textClass || 'card-text';

}

}

// Define the new element

customElements.define('card-component', CardComponent);

**index.html**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

<title>Card Component Example</title>

</head>

<body>

<h2>Card Component Example</h2>

<!-- Example Usage of Card Component -->

<card-component

data='{

"title": "Card Title",

"description": "This is a sample card component description.",

"image": "https://via.placeholder.com/150"

}'

config='{

"imageClass": "card-img-top",

"titleClass": "card-title",

"textClass": "card-text"

}'>

</card-component>

<script src="./script.js"></script>

</body>

</html>

**form.json**

json

Copy code

{

"fields": [

{

"key": "title",

"type": "input",

"templateOptions": {

"label": "Card Title",

"placeholder": "Enter card title",

"required": true

}

},

{

"key": "description",

"type": "textarea",

"templateOptions": {

"label": "Description",

"placeholder": "Enter card description",

"required": true

}

},

{

"key": "image",

"type": "input",

"templateOptions": {

"label": "Image URL",

"placeholder": "Enter image URL",

"required": true

}

},

{

"key": "config.imageClass",

"type": "input",

"templateOptions": {

"label": "Image CSS Class",

"placeholder": "Enter CSS class for image",

"required": true

}

},

{

"key": "config.titleClass",

"type": "input",

"templateOptions": {

"label": "Title CSS Class",

"placeholder": "Enter CSS class for title",

"required": true

}

},

{

"key": "config.textClass",

"type": "input",

"templateOptions": {

"label": "Text CSS Class",

"placeholder": "Enter CSS class for text",

"required": true

}

}

]

}

**Assignment Requirements:**

1. Use document.createElement to create each element of the component.
2. Use DOM methods like appendChild, setAttribute, and textContent to manipulate elements.
3. Apply classes to elements based on the config object values.
4. Populate content from the data object into the respective HTML elements.

**Submission Instructions:**

* Fork the bootstrap-component-assignments repository from GitHub.
* Create your assignment inside the SubmittedAssignments folder using the naming format: FirstName\_LastName\_designNumber (e.g., Vilas\_Shetkar\_01).
* Ensure the folder structure follows the specified format.
* Once completed, push your code to GitHub and raise a pull request.

This example should help guide you in creating your own Web Component using the document.createElement method and DOM manipulation techniques.