Lab 1: Introduction to Knockout.js and the MVVM Pattern

Lab 1.1: Hello World with Knockout.js

The following example demonstrates the core concepts of Knockout.js: **observables**, **ViewModel**, and **declarative data binding**.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Knockout.js Hello World</title>
  <!-- Include Knockout.js from CDN -->
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/knockout/3.5.1/knockout-
min.js"></script>
</head>
<body>

  <!-- View: The UI that displays data -->
  <div>
    <p>Hello, <span data-bind="text: userName"></span>!</p>
    <p>Enter your name: <input type="text" data-bind="value: userName" /></p>
  </div>

  <script>
    // ViewModel: JavaScript object that holds the data and logic
    function AppViewModel() {
      // Observable property - automatically notifies the UI when changed
      this.userName = ko.observable("World");
    }

    // Apply bindings: Connect the ViewModel to the DOM
    ko.applyBindings(new AppViewModel());
  </script>

</body>
</html>
```

Explanation of Key Concepts

Code Segment	Explanation
`ko.observable("World")`	Creates a reactive property. Whenever this value changes, any bound UI elements automatically update.
`data-bind="text: userName"`	Declarative binding that sets the text content

Code Segment	Explanation
	of the to the value of userName.
data-bind="value: userName"	Binds the input field's value to the same userName observable, enabling two-way binding.
ko.applyBindings(new AppViewModel())	Activates Knockout and links the ViewModel to the DOM. After this, all data-bind attributes are processed.

Lab 1.2: Building a Dynamic Greeting App with Knockout.js

Lab Title: Create a Reactive Name Greeting Application

In this hands-on exercise, you will create a simple web page that dynamically updates a greeting message as the user types their name. You'll use Knockout.js to implement the MVVM pattern, leveraging observables and declarative bindings.

Files to Create

You will create the following file:

- `greeting-app.html`

Step-by-Step Instructions

Step 1: Create the HTML File

Create a new file named `greeting-app.html` in your project directory.

Step 2: Set Up the HTML Structure

Add the basic HTML boilerplate and include the Knockout.js library from a CDN.

> **Note**: Always include the Knockout script before using ko in your code.

Step 3: Design the View

Add a simple UI with: - A greeting message that displays the user's name. - An input field where the user can type their name.

Use `data-bind` attributes to connect the UI elements to the ViewModel.

Step 4: Define the ViewModel

Inside a `<script>` tag, define a JavaScript function called `GreetingViewModel`. It should contain: - An observable property `userName` initialized to `"Guest"`.

Step 5: Apply Bindings

Use `ko.applyBindings()` to connect the ViewModel to the DOM.

Step 6: Test the Application

Open the file in a modern web browser (e.g., Chrome, Firefox). Type into the input field and verify that the greeting updates in real time.

Expected Output

When opened in a browser:

```
Hello, Guest!

Enter your name: [__________]
```

As the user types, for example, “Alice”, the page should instantly update to:

```
Hello, Alice!

Enter your name: [Alice________]
```

The change happens **without page refresh or manual DOM updates**.

---

4. Complete Implementation of the Lab Exercise

Below is the full, working implementation of the `greeting-app.html` file as described in the lab.

File: `greeting-app.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Knockout.js Greeting App</title>
  <!-- Load Knockout.js from CDN -->
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/knockout/3.5.1/knockout-
min.js"></script>
  <style>
```

```
    body {
      font-family: Arial, sans-serif;
      margin: 40px;
      line-height: 1.6;
    }
    input {
      padding: 5px;
      font-size: 16px;
    }
    span {
      font-weight: bold;
      color: #0056b3;
    }
  </style>
</head>
<body>

  <!-- View: The user interface -->
  <h2>Dynamic Greeting Application</h2>
  <p>Hello, <span data-bind="text: userName"></span>!</p>
  <p>Enter your name: <input type="text" data-bind="value: userName"
placeholder="Type your name" /></p>

  <!-- ViewModel and Binding Logic -->
  <script>
    // Define the ViewModel
    function GreetingViewModel() {
      // Observable to hold the user's name
      this.userName = ko.observable("Guest");
    }

    // Apply Knockout bindings to connect ViewModel to the View
    ko.applyBindings(new GreetingViewModel());
  </script>

</body>
</html>
```

How to Run the Application

1. Save the above code into a file named `greeting-app.html`.
2. Ensure you have an internet connection (to load the Knockout.js CDN).
3. Open the file in a web browser by double-clicking it or using a local server.
4. Start typing in the input box and observe the `<span>` updating instantly.

Key Takeaways from the Lab

- **No Manual DOM Updates**: You did not use `document.getElementById()` or `.innerHTML`. Knockout handles updates automatically.
- **Reactivity via Observables**: The `userName` is wrapped in `ko.observable()`, making it reactive.
- **Declarative Binding**: The `data-bind` syntax keeps the HTML clean and expressive.
- **MVVM in Action**:
    - **Model**: (Implied) The user's name data.
    - **View**: The HTML with `data-bind`.
    - **ViewModel**: `GreetingViewModel` function that exposes data and behavior.
