00:00-13:18

Building a Website with HTML, CSS, and JavaScript

Introduction

In the previous video, we developed a Python Flask server that serves as the backend for our website. Now, in this tutorial, we will build the frontend of our website using HTML, CSS, and JavaScript. We will create three files: `app.html`, `app.css`, and `app.js`. These files will define the structure, style, and dynamic behavior of our web application.

HTML Structure

In the `app.html` file, we will define the structure of our user interface (UI). The HTML code will consist of two main sections: the head and the body. The head section includes the title of the application and the import of necessary files such as jQuery, `f.js`, and `f.css`. The body section contains the main elements of our application, including an input for area, buttons for BHK (bedroom, hall, kitchen) selection, a button bar for bathroom options, and a dropdown for location selection.

Styling with CSS

To style our website, we will use CSS. We have defined several CSS classes in the `app.css` file that correspond to the UI elements mentioned earlier. These classes define properties like background color, font style, and other visual aspects of the application.

Running the Application

To see our UI in action, we can open the `app.html` file in a web browser. At this point, the locations in the dropdown are still hard-coded. We will dynamically populate the dropdown using JavaScript in the next step.

Communicating with the Backend using JavaScript

In the `app.js` file, we will write JavaScript code to communicate with the backend server and retrieve data. First, we will implement a function to load the locations. Using JavaScript's `window.onload` event, we can make an HTTP GET request to the server to fetch the list of locations. We will then iterate through the locations and dynamically add them to the dropdown.

Next, we will implement the `onEstimatePriceClick` function, which will be triggered when the user clicks the "Estimate Price" button. This function will retrieve the values of the UI elements (area, BHK, bathroom, and location) using helper functions `getBHKValue` and `getBathValue`. With these values, we will make an HTTP POST request to the server's "predict home price" endpoint. The response will contain the estimated price, which we will display on the UI.

Testing the Application

After refreshing the page, we can test the application by selecting a location and clicking the "Estimate Price" button. The estimated price will be displayed based on the selected features. Different locations and property configurations will yield different prices.

Deployment and Conclusion

At this point, our application is working fine locally. However, if we want to deploy it in a production environment, additional steps and procedures need to be followed. In a typical workflow, a data scientist builds models, a Python Flask server is developed, and a separate software engineering team handles the UI and backend application. However, it's beneficial for data scientists to have some knowledge of web development to test their models effectively.

In summary, this tutorial series has provided insights into how a data scientist working for a large organization builds an application from end to end. We covered data cleaning, dimensionality reduction, feature engineering, model building, creating a Python Flask server, and developing the UI using HTML, CSS, and JavaScript. We hope you found this series informative and useful for your journey as a data science enthusiast.

Remember to give a thumbs up and share this series with others who may find it helpful. Thank you for watching!

Bye!