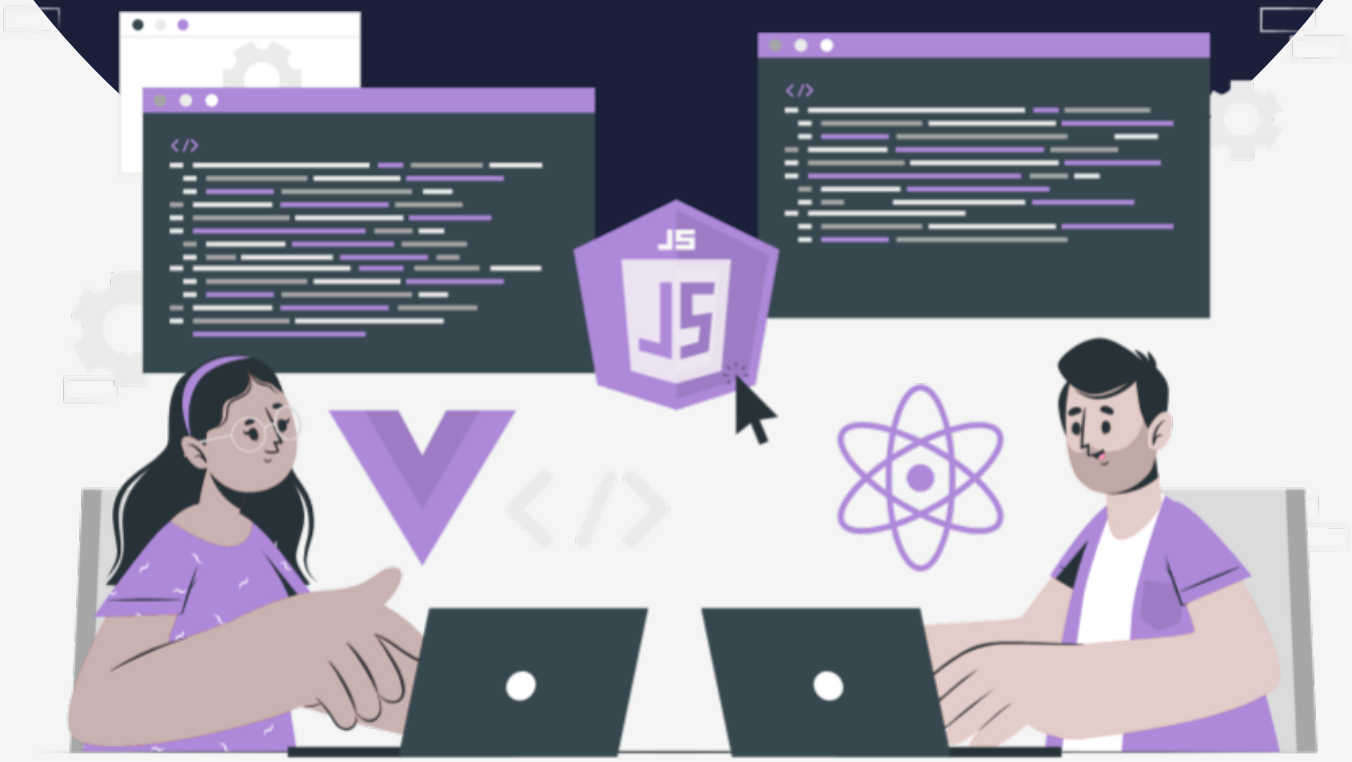
Lesson:

Working with environment variables using .env



PW SKILLS

Topics

- What are environmental variables and why do we use them?
- Good practices to follow while defining environment variables in .env file.
- Set up and read a .env file?

What are environmental variables, and why do we use environment variables?

Environment variables are variables available to your program/application dynamically during runtime. The value of these variables can come from a range of sources — text files, third-party secret managers, calling scripts, etc.

Environment variables are used to store sensitive data such as passwords, API credentials, and other information that should not be written directly in code. Environment variables must be used to configure any variables or configuration details that may differ between environments.

Good practices to follow while defining environment variables in .env file.

Defining environment variables in a well-organized and consistent way can make it easier to manage and maintain your application's configurations. Here are some good practices for defining environment variables in a .env file

- **Use descriptive names:** Use clear and meaningful names for your environment variables. This makes it easier for other developers to understand what the variables are used for.
- **Use uppercase letters:** It's common convention to use all uppercase letters for environment variables. This makes it easier to distinguish them from other variables in your code.
- **Separate words with underscores:** To make your environment variables more readable, separate words with underscores. For example, use "DATABASE_URL" instead of "databaseUrl".
- **Keep the file consistent:** Use a consistent format for your .env file. For example, you could use the format "VARIABLE_NAME=value" for each variable.
- **Don't commit .env file to version control:** Make sure your .env file is not committed to version control, as this could expose sensitive information to unauthorized users.
- **Update .env file when changes are made:** Whenever you make changes to your application's configuration, make sure to update the corresponding variables in your .env file.

By following these good practices, you can create a well-organized and secure .env file that makes it easier to manage your application's configuration.

Set up and read .env file in node.js.

The dotenv package is widely recognized as the most popular choice in the Node.js ecosystem for managing environment variables. You can create an .env file in the application's root directory, and it should contain all the necessary key/value pairs for the required environment variables in the project. The dotenv library is responsible for reading the .env file and appending its contents to the process.env object.

To set up and read .env file, follow these general steps.

1. create a new file in the root directory of your project called **.env.** This file should not have a file extension and should be in the same directory as your main code file.
2. Add your environment variables to the .env file using the following format: **KEY=VALUE.** For example:

```
Unset
PORT=localhost
DB_URL=url
DB_PASSWORD=mypassword
DB_USERNAME=myusername
```

3. Save the .env file.
4. Now Install dotenv npm package. Using the command given below.

```
Unset
npm install dotenv
```

5. In your main code file, import the package that you just installed and load the environment variables from the **.env** file:

```
JavaScript
require('dotenv').config();
```

6. You can now access the environment variables in your code using the process.env object. For example:

```
JavaScript
const dbConfig = {
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
};
```

Note that you should never commit your **.env** file to version control, as this can expose sensitive information. Instead, you should add the file to your **.gitignore** file to prevent it from being tracked. Additionally, you should be careful to use unique key names for your environment variables to ensure that your application is using the correct variables for its intended purpose.