Module 1: Getting Started with React

Lecture 1 - Introduction to React: The Power of Declarative UI

What is React?

- React is a JavaScript library developed by Facebook.
- Used for building **user interfaces**, especially single-page applications.
- It focuses only on the **view layer** of the application (UI).
- Allows you to create **reusable components**.
- It uses a **virtual DOM** to improve performance.

Why React?

- **Fast Rendering**: Uses virtual DOM for efficient updates.
- **Reusable Components**: Build once, reuse anywhere in the UI.
- **Rich Ecosystem**: Supported by tools like React Router, Redux, etc.
- **Large Community**: Tons of support, documentation, and tutorials.
- **Used by top companies** like Meta, Netflix, and Uber.

Why React?

- **Fast Rendering**: Uses virtual DOM for efficient updates.
- **Reusable Components**: Build once, reuse anywhere in the UI.
- **Rich Ecosystem**: Supported by tools like React Router, Redux, etc.
- **Large Community**: Tons of support, documentation, and tutorials.
- **Used by top companies** like Meta, Netflix, and Uber.

Module 1: Getting Started with React

Declarative UI

- In React, you **describe what the UI should look like** for a given state.
- React updates the DOM automatically when state changes.
- Eliminates manual DOM manipulation (like `document.getElementByID()`).
- Makes code **simpler**, **cleaner**, and **predictable**.

Single Page Applications (SPA)

- The entire app loads once, and **routing happens via JavaScript**.
- No full-page reloads – only parts of the UI update.
- Fast and smooth user experience.
- Uses tools like **React Router** for client-side routing.
- Ideal for modern, responsive web apps.

Module 1: Getting Started with React

Lecture 2 - Understanding the Virtual DOM & Reconciliation Process

1. **What is Virtual DOM?**
    - A lightweight copy of the actual DOM.
    - It allows React to perform updates more efficiently by minimizing direct manipulation of the real DOM.
2. **How Virtual DOM Works:**
    - When the state or props of a component change, a new Virtual DOM is created.
    - React compares this new Virtual DOM with the previous one to identify changes (diffing).
3. **Reconciliation Process:**
    - The process of updating the actual DOM to match the Virtual DOM.
    - React calculates the minimal number of changes needed and applies these changes to the actual DOM.
4. **Benefits of Virtual DOM:**
    - Improves performance by reducing direct manipulation of the DOM.
    - Makes UI updates faster and smoother.
5. **Key Points:**
    - Virtual DOM updates are batched and optimized.
    - Only the parts of the DOM that changed are updated, not the entire DOM.

Module 1: Getting Started with React

Lecture 3- Setting Up the Development Environment

Install Node.js:

- Required for running and managing JavaScript projects.

- Download from the official Node.js website.

Windows users: Ensure your environment variable path is correctly set up. Make sure you add your Node.js paths to the environment variables. If you don't do this, NODE might not work correctly.

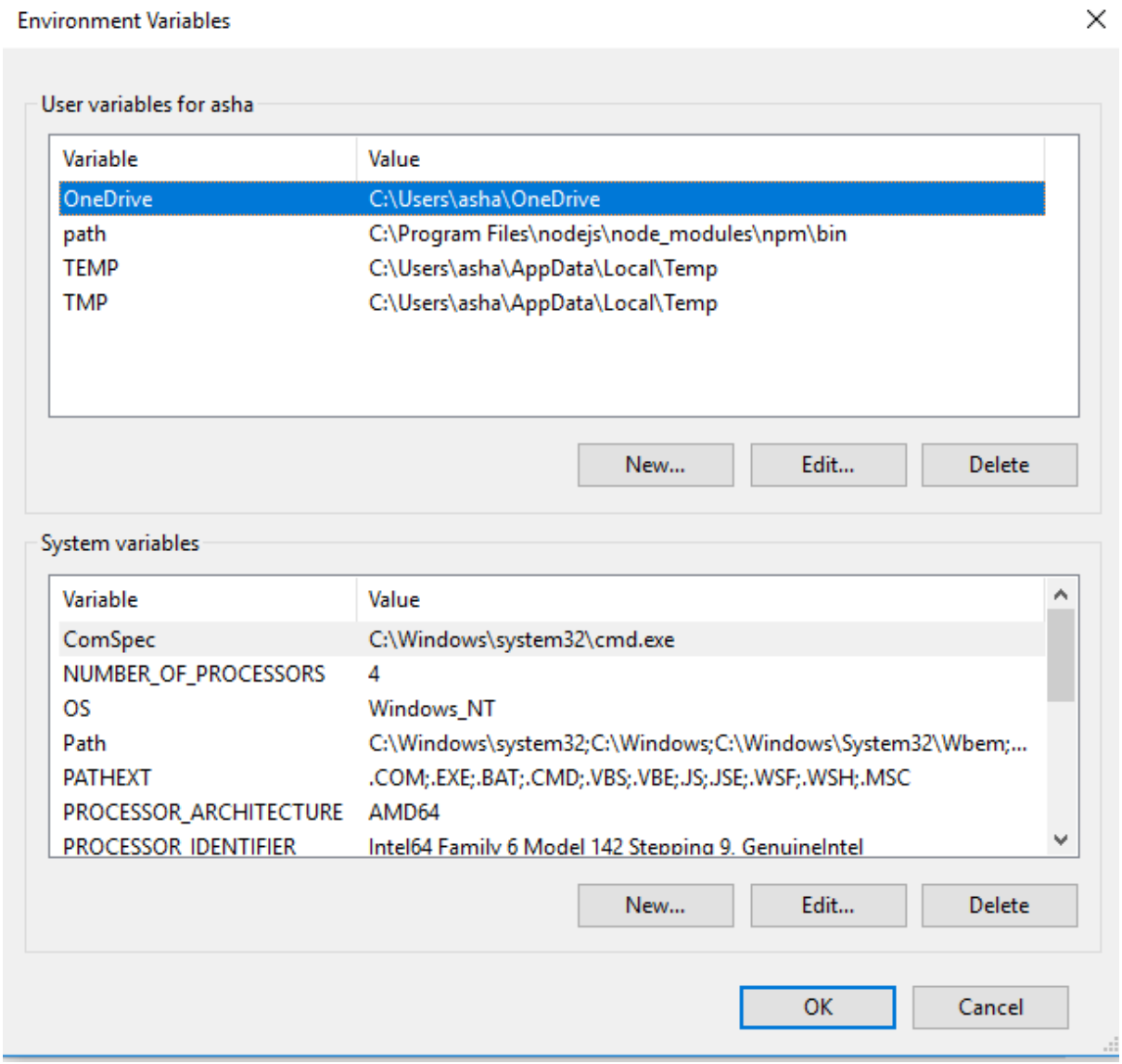
And to open the environment variables in Windows, follow these steps:

1. Right-click on 'This PC' or 'My Computer' and select 'Properties'.

2. Click on 'Advanced system settings'.

3. In the System Properties window, click on the 'Environment Variables...' button.

4. Under 'System variables', find and edit the 'Path' variable to add your desired paths.



Install VS Code:

- A powerful code editor for development.

- Download from the VS Code website.

Recommended VS Code Extension:

- ES6/ES7 Babel: Helps with modern JavaScript syntax highlighting and snippets.
- Prettier extension

Module 1: Getting Started with React

Lecture 4- Creating Your First React Application

Using Create React App (Deprecated):

- Although `create-react-app` is deprecated, you can still use it to create a React application.

Command: npx create-react-app my-app

Using Vite (Recommended):

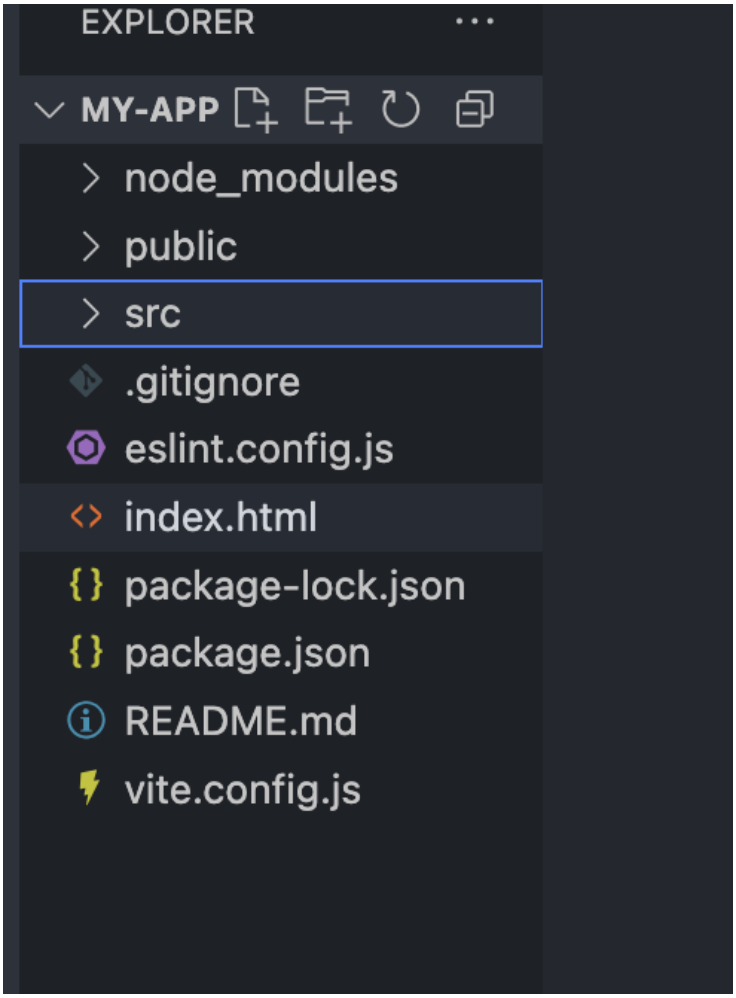
- For a faster and more modern setup, you can use Vite.

Command: npm create vite@latest

This command initializes a new React project with Vite, providing a more efficient development experience.

Module 1: Getting Started with React

Lecture 5 - Understanding the React Project Structure



1. **`node_modules/`**
   - Contains all the installed dependencies from `package.json`.
   - Auto-generated after `npm install`.
2. **`public/`**
   - Static files go here (e.g., images, `favicon.ico`)
   - Files are copied as-is to the final build.

3. **`src/`**

- Main source code folder.
- All React components, CSS files, and logic reside here.
- Entry point typically is `main.jsx` or `main.tsx`.

4. **`.gitignore`**
   - Specifies which files/folders Git should ignore (e.g., `node_modules`, `.env`).
5. **`eslint.config.js`**
   - Configuration for ESLint – helps maintain code quality and consistent style.
6. **`index.html`**
   - Root HTML file.
   - Vite injects the React app inside the `<div id="root"></div>` of this file.
   - Script tag uses `type="module"`.
7. **`package.json`**
   - Lists project metadata and dependencies.
   - Includes scripts like `dev`, `build`, and `preview`.
8. **`package-lock.json`**
   - Auto-generated lock file.
   - Ensures consistent dependency versions across all environments.
9. **`README.md`**
   - Markdown file with project description, setup steps, etc.

10. `vite.config.js`

Vite configuration file.

Used for customizing build behavior, aliases, plugins, etc.

Module 1: Getting Started with React

Lecture 6 - NPM vs NPX vs NVM Explained | Caret (^) & Tilde (~) Simplified

NPM (Node Package Manager)

- Installs packages from the npm registry.
- Can install locally (`node_modules/`) or globally.
- Comes by default with Node.js.

NPX (Node Package Execute)

- Executes a package without installing it permanently.
- Great for one-time commands or CLIs like `create-react-app`, `vite`, etc.

Where does NPX store temporary packages?

- **Mac:**
  `~/.npm/_npx/<random-id>/node_modules/.bin/`
- **Windows:**
  `C:\Users\<You>\AppData\Local\Temp\npx\<temp>`

NVM (Node Version Manager)

- Allows switching between multiple Node.js versions.
- Useful when working on projects needing different versions.

✅ Common Commands:

nvm install 20

nvm use 20

nvm alias default 20

🔧 **NVM Setup for Mac:**

Refer to this - https://github.com/nvm-sh/nvm

1. **Create and open the profile**

   touch ~/.zprofile

   open ~/.zprofile

2. Add these lines

   export NVM_DIR="$HOME/.nvm"

   [-s "$NVM_DIR/nvm.sh"] && \. "$NVM_DIR/nvm.sh"

3. Apply changes:

   source ~/.zprofile

🔧 **NVM Setup for Windows :**

Refer to this - Releases · coreybutler/nvm-windows

For Windows users, it's important to set up your environment variable path properly. Make sure you add your **NVM** paths to the environment variables. If you don't do this, NVM might not work correctly.

And to open the environment variables in Windows, follow these steps:

1. Right-click on 'This PC' or 'My Computer' and select 'Properties'.

2. Click on 'Advanced system settings'.

3. In the System Properties window, click on the 'Environment Variables...' button.

4. Under 'System variables', find and edit the 'Path' variable to add your desired paths.

Module 1: Getting Started with React

Caret (^) vs Tilde (~) in `package.json`

`^ – Caret`

- `Meaning: Allows updates to minor and patch versions.`
- `Example: "^1.2.3"`
- `Acceptable Updates:`
  - ✅ 1.3.0, 1.4.5
  - ❌ 2.0.0 (major version change not allowed)

`Use when: You want to get the latest minor updates safely.`

`~` – Tilde

- Meaning: Allows updates to patch versions only.
- Example: `"~1.2.3"`
- Acceptable Updates:
  - ✅ `1.2.4`, `1.2.9`
  - ❌ `1.3.0` (minor version change not allowed)

- Use when: You want strict version control with only safe patch updates.

Module 1: Getting Started with React

Lecture 7 - Writing Code from Scratch

Creating a Simple React Application (Without Default Structure)

1. Import React and ReactDOM:

   At the top of your `main.jsx` file, import the necessary libraries:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
```

2. Create a Root:

   Use `ReactDOM.createRoot` to create a root for your React app:

```
const root = ReactDOM.createRoot(document.getElementById('root'));
```

3. Render a Simple Component:

   Render a simple React element (like a heading) to the root:

```
root.render(<h1>Hello, React!</h1>);
```

4. **Minimal Setup:**

   No complex directory structure is used (like the one generated by Vite or CRA).

The `src` folder is kept simple, containing just the `main.JSX` file for demonstration.

Lecture 8 - Creating & Publishing Your First NPM Package

1. **Initialize Your Package:**

   Create a new directory for your package and navigate into it.

   Run `npm init -y` to create a `package.json` file.

   Set the `name` of your package and the `license` to MIT.

2. **Write Your Code:**

   Create a JavaScript function (e.g., a function to add two numbers) and export it.

3. **Prepare for Publishing:**

   Ensure your package is ready and all files are correct.

   Optionally, add a `README.md` for documentation.

4. **Publish the Package:**

   Run `npm login` and enter your NPM credentials.

   Publish your package with `npm publish`.

5. **Use the Published Package:**

   In another project, install your package using `npm install your-package-name`.

   Import and use the function you created.