Episode 1

What is emmet?

Emmet, formerly known as Zen Coding, is a powerful and widely used web development tool that allows developers to write HTML and CSS code using shorthand syntax. It dramatically speeds up the process of writing and expanding code, making it a favorite among developers who work with HTML and CSS extensively.
Emmet works as a plugin or extension for various code editors, including Visual Studio Code, Sublime Text, Atom, and others. Once installed, you can use Emmet's shorthand syntax to quickly generate complex code snippets. It is particularly helpful for creating repetitive HTML structures and CSS properties.
Here are a few examples of how Emmet works:
HTML:
Typing `!` and then pressing the `Tab` key expands to a basic HTML boilerplate:

```
phpCopy code
<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Document</title> </head> <body> </body> </html>
```

Typing `ul>li*3` and then pressing the `Tab` key creates an unordered list with three list items:

```
cssCopy code
<ul> <li></li> <li></li> <li></li> </ul>
```

CSS:
Typing `m10` and then pressing the `Tab` key sets the `margin` property to `10px`:

```
cssCopy code
margin: 10px;
```

Typing `bgc-red` and then pressing the `Tab` key sets the `background-color` property to `red`:

```
cssCopy code
background-color: red;
```

Emmet is a time-saving tool that can significantly improve the productivity of web developers by reducing the amount of repetitive typing. It's essential to learn and use Emmet if you work frequently with HTML and CSS in your web development projects.

Difference between a library and a framework?

The terms "library" and "framework" are often used in the context of software development, and they refer to different types of tools or software components that developers use to build applications. Here's the key difference between a library and a framework:
Library:
A library is a collection of functions, classes, or modules that provide specific functionalities or features. It is designed to be used as a set of tools that can be integrated into your code as needed.
When using a library, the developer remains in control of the flow of the application. The developer decides when and how to use the library's functions or components to achieve the desired outcome.
Libraries are typically more flexible, and developers have the freedom to pick and choose specific components to use while maintaining the rest of the application's architecture and design.
Examples of libraries include React (for building user interfaces), Axios (for making HTTP requests), and Lodash (for utility functions).
Framework:

A framework, on the other hand, is a more comprehensive and opinionated set of tools, guidelines, and conventions that dictate the application's overall structure and flow.
When using a framework, the application's flow and structure are largely determined by the framework itself. The developer fills in the specific details and implements custom code within the framework's boundaries.
Frameworks often provide a complete solution for building applications, including predefined architecture, design patterns, and rules for interaction between different components.
Developers work within the framework's ecosystem and have less control over the overall application structure compared to using a library.
Examples of frameworks include Angular (a complete frontend framework), Django (a web framework for Python), and Ruby on Rails (a web framework for Ruby).
In summary, a library is a collection of tools that developers use to perform specific tasks, and they have more control over how and when to use these tools. On the other hand, a framework is a more comprehensive solution that dictates the application's structure, and developers work within the framework's guidelines and conventions to build their applications.

Why is React known as react?

React is known as "React" because of its core concept, which revolves around the idea of reacting to changes in data and efficiently updating the user interface (UI) based on those changes. The name "React" comes from the reactive nature of the library.
When building web applications, one of the critical challenges is efficiently updating the UI in response to changes in data or user interactions. Traditional approaches involve directly manipulating the DOM (Document Object Model), which can be inefficient and lead to performance issues, especially in complex applications.
React was developed by Jordan Walke, a software engineer at Facebook, and it was initially released by Facebook in March 2013. It was later open-sourced and became widely adopted in the web development community.
React introduces a concept called the "Virtual DOM." Instead of directly updating the actual DOM, React creates a lightweight virtual representation of the DOM. When the application's state or data changes, React compares the previous virtual DOM with the new one using a process called "reconciliation." It identifies the minimal set of changes needed to update the actual DOM efficiently, and then it applies those changes in a batch.
By taking advantage of the Virtual DOM and its efficient change detection algorithms, React minimizes the number of DOM manipulations, leading to better performance and improved user experience.
The name "React" highlights this reactive approach to updating the UI in response to data changes, making it a suitable and descriptive name for the library. Over time, React has gained immense popularity and has become one of the most widely used libraries for building user interfaces in web development.

What is cross-origin in script tag?

In the context of web development, the term "cross-origin" refers to interactions between web pages or web applications that originate from different domains, protocols, or ports. When a web page or script loaded from one origin tries to access resources from another origin, it is considered a cross-origin request.
To protect users from potential security risks, web browsers enforce a security feature known as the "same-origin policy." This policy prevents web pages from making direct requests to

resources (such as APIs or scripts) hosted on different domains or ports from the one that served the original web page.
The **<script>** tag is commonly used to include external JavaScript files in a web page. When you use the **<script>** tag to load a script from a different origin, it is known as a "cross-origin script."
For example, let's say you have a web page hosted on "https://www.example.com" and you include an external script using the **<script>** tag with a source from "https://api.example.com/data.js". This script tag will trigger a cross-origin request since it's attempting to load a script from a different domain ("api.example.com") than the one that served the web page ("www.example.com").
Cross-origin scripts are subject to some restrictions imposed by the same-origin policy. These restrictions typically prevent the cross-origin script from accessing sensitive data or interacting directly with the DOM of the web page that loaded it. However, there are mechanisms available, such as Cross-Origin Resource Sharing (CORS), that allow server-side configurations to explicitly allow or restrict cross-origin requests.
It's essential to understand cross-origin issues and how they are managed, especially when working with APIs or including external scripts in your web applications, to ensure security and prevent potential vulnerabilities.

What is difference between React and ReactDOM?

React and ReactDOM are two separate packages provided by React to handle different aspects of web development. Here's the difference between React and ReactDOM:
React:
React is the core library or framework for building user interfaces in JavaScript. It provides the foundational features and components needed to create interactive and reusable UI components.
React is not specific to the web browser environment; it can be used for building user interfaces in various environments, including web, mobile, and desktop applications.
With React, you can define components, manage component state, handle user interactions, and efficiently update the UI through its virtual DOM reconciliation process.
React focuses on the component-based architecture, where the user interface is broken down into smaller, reusable pieces, making it easier to manage and maintain complex applications.
ReactDOM:
ReactDOM is a package specifically designed to work with React in the web browser environment. It provides the integration between React and the DOM (Document Object Model) of the web page.
ReactDOM is responsible for rendering React components to the actual DOM and updating the DOM when changes occur in the component tree.
It provides methods like `ReactDOM.render()` and `ReactDOM.hydrate()` to render React components into a container element in the DOM.
ReactDOM takes care of efficiently updating the DOM using the virtual DOM diffing process, which helps improve the performance of React applications.
In summary, React is the core library for building user interfaces using a component-based architecture, while ReactDOM is a package that complements React specifically for rendering components and managing the DOM in web applications. If you are working with

React in the web browser environment, you will need both React and ReactDOM to create and render UI components effectively.

What is the difference between react.development.js and react.production.js?

The main difference between `react.development.js` and `react.production.js` is how they are optimized for development and production environments, respectively, when using React in a web application. These files contain the React library with different settings and behaviors tailored to each environment:

`react.development.js`:

This file is intended for development purposes. It includes additional checks, warnings, and helpful error messages that aid developers in identifying and fixing potential issues during the development process.

It provides detailed error messages and stack traces, making it easier to debug and trace the source of any errors or warnings in the code.

Since it includes extra debugging information and checks, the file size is larger compared to the production version.

`react.production.js`:

This file is optimized for production use. It is stripped of any development-related warnings, error messages, and debugging information to reduce its file size and improve performance.

It focuses on minimizing the size and maximizing the performance of the React library, making it suitable for production environments where file size and performance are critical.

Removing the development-specific checks and messages results in a smaller file size, which leads to faster loading times for the application.

When building a React application, it is crucial to use the appropriate version of the React library based on the environment. During development, you should use `react.development.js` to take advantage of the helpful debugging features, while in the production environment, you should use `react.production.js` to ensure optimal performance.

It is common practice to use build tools or bundlers like Webpack or Parcel to handle environment-specific configurations and automatically swap the appropriate React version based on whether the application is being developed or deployed to a production server. This way, developers can work with the development version during development and still benefit from the optimized production version when the application is deployed to users.