# Why React?

In the current lecture, the speaker, Katie, a Software Engineer at Meta, discusses the significance of React in the tech industry. Key points include:

React's Popularity: It's widely used for building user interfaces due to its ease of learning and flexibility.

Composition vs. Inheritance: Unlike traditional object-oriented programming, React uses composition, which can be initially challenging but is supported by extensive documentation and community resources.

Custom UIs: React allows for the creation of highly customizable user interfaces, making it suitable for complex applications.

Code Re-usability: React promotes reusing components, which enhances efficiency and consistency across applications.

Comparison with Angular: React is a client-side library that requires integration with other libraries, while Angular is a full-fledged framework that provides more out-of-the-box solutions.

Learning Approach: Beginners are encouraged to start with simple projects and gradually build complexity, utilizing best practices and documentation.

This lecture emphasizes the importance of React skills for a successful career in tech. Would you like to explore any specific aspect of React further?

# React.js overview

Component-based architecture is like building a house using different blocks. Each block, or component, has its own specific purpose and can be reused in different parts of the house without needing to change the other blocks. For example, think of a website as a collection of these blocks: you might have a header block with a logo and navigation, a footer block with contact information, and a content block that displays articles. Each of these blocks can work independently, making it easier for multiple developers to collaborate on the same project without stepping on each other's toes.

This approach not only makes the development process smoother but also helps keep the code organized and manageable. Just like how you can rearrange blocks to create different layouts for your house, in React, you can combine components in various ways to create a unique user interface. This flexibility is one of the reasons why React is so popular among developers!

# Introduction to functional components

Functional components in React, which are similar to traditional JavaScript functions. Here are the key points:

Functional Components: These are reusable blocks of code that take inputs, perform actions, and return outputs, much like JavaScript functions.

JSX: React uses a syntax called JavaScript XML (JSX), which resembles HTML but allows you to write JavaScript code within it. JSX is used to create dynamic content in React components.

Creating Components: To create a functional component, you define a JavaScript function with a capitalized name. Inside the function, you can return JSX, which can include variables wrapped in curly brackets to display dynamic content.

Rendering: A React component must be used as a JSX element to render it on the webpage. The process of converting JSX to HTML is called transpiling.

This lecture sets the foundation for understanding how to build user interfaces in React using functional components and JSX.

# Creating React components

* Components are reusable blocks of code in React.
* You started by using Create React App to set up a new application.
* The app.js file was modified to create a simple functional component named header that displays "Hello World" in an H1 element.
* You learned how to render this component by calling it within the app function using JSX syntax.
* The importance of isolating components into their own files for reusability was also highlighted.
* This foundational knowledge is essential for building scalable and maintainable applications in React.

function Heading() {

return <h1>hello world</h1>;

}

function App() {

return <Heading />;

}

export default App;