LLMs in Action: Developing a COM AI Player for an Interactive Tic-Tac-Toe Game using OpenAI Developer API

Lesson 4: Frontend Development with React

Priscilla Emasoga

COM3550 Undergraduate Ambassadors Scheme
University of Sheffield

Overview

- **What are Single Page Applications?**
  - Examples of SPAs
  - Traditional Web Apps vs SPAs
  - Popular SPA Frameworks
- **Introduction to React**
  - **Key Concepts:** Components, Props, Events, Hooks
- **Tic-Tac-Toe Web Client**
  - Project Features
  - Live DEMO

What are Single Page Applications?

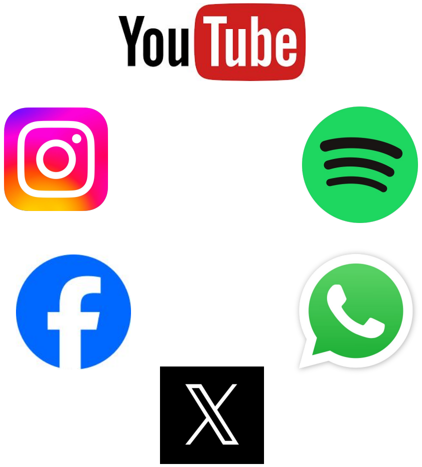
Single Page Applications (SPAs) are web applications that **load a single HTML page** and **dynamically update** the content as users interact with the app, without requiring **full page reloads**.

Question!!!!

Put on your thinking hat!

What websites or apps do you use that feel very interactive?

Examples of Single Page Applications



...and lots of others all use **SPAs** in their web applications!

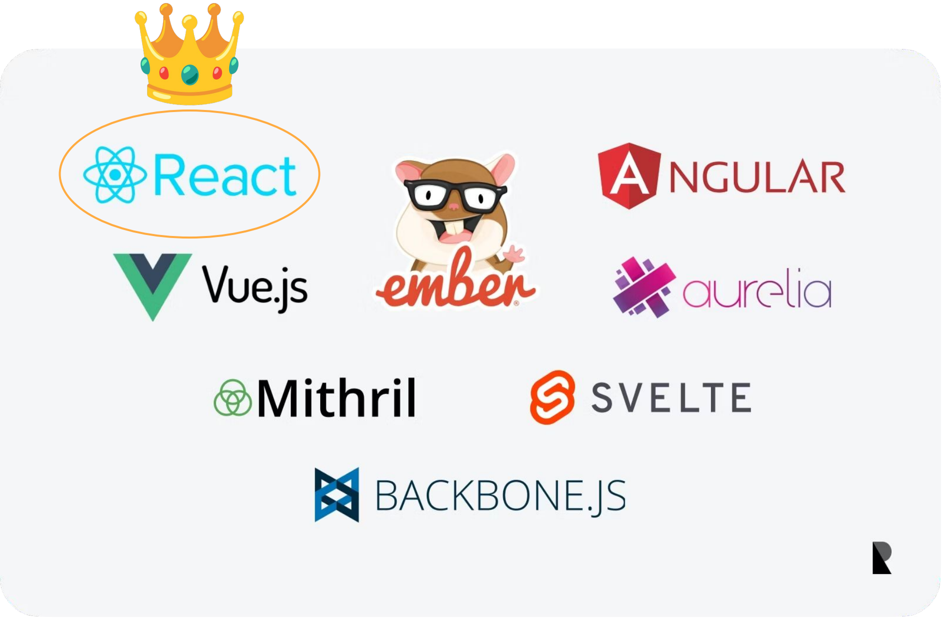
“**But wait!!!** These all have multiple web pages?”

Traditional Web Apps vs SPAs

Traditional	**SPA**
Full page reload for each new page	Content updates without page refresh
Server renders complete HTML for each request	Server primarily sends data (JSON/XML)
Higher bandwidth usage	Lower bandwidth usage after initial load
Simple architecture, SEO-friendly by default	More complex architecture, requires client-side rendering

Popular SPA Frameworks

When it comes to choosing an **SPA framework**, there are so many options!



We will focus on **React** for this course!

Introduction to React

Introduction to React

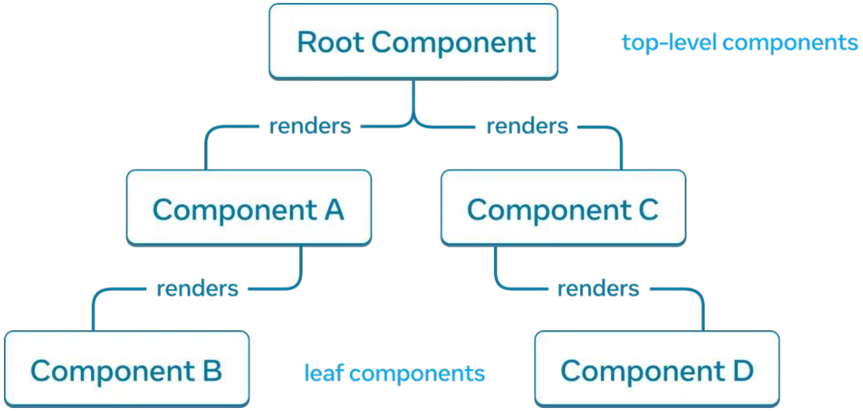
React is a JavaScript library for building user interfaces, particularly SPAs.

- **Created by:** Facebook (now Meta) in 2013
- **Purpose:** Build **reusable** UI **components** that efficiently update when data changes.

Key Concepts

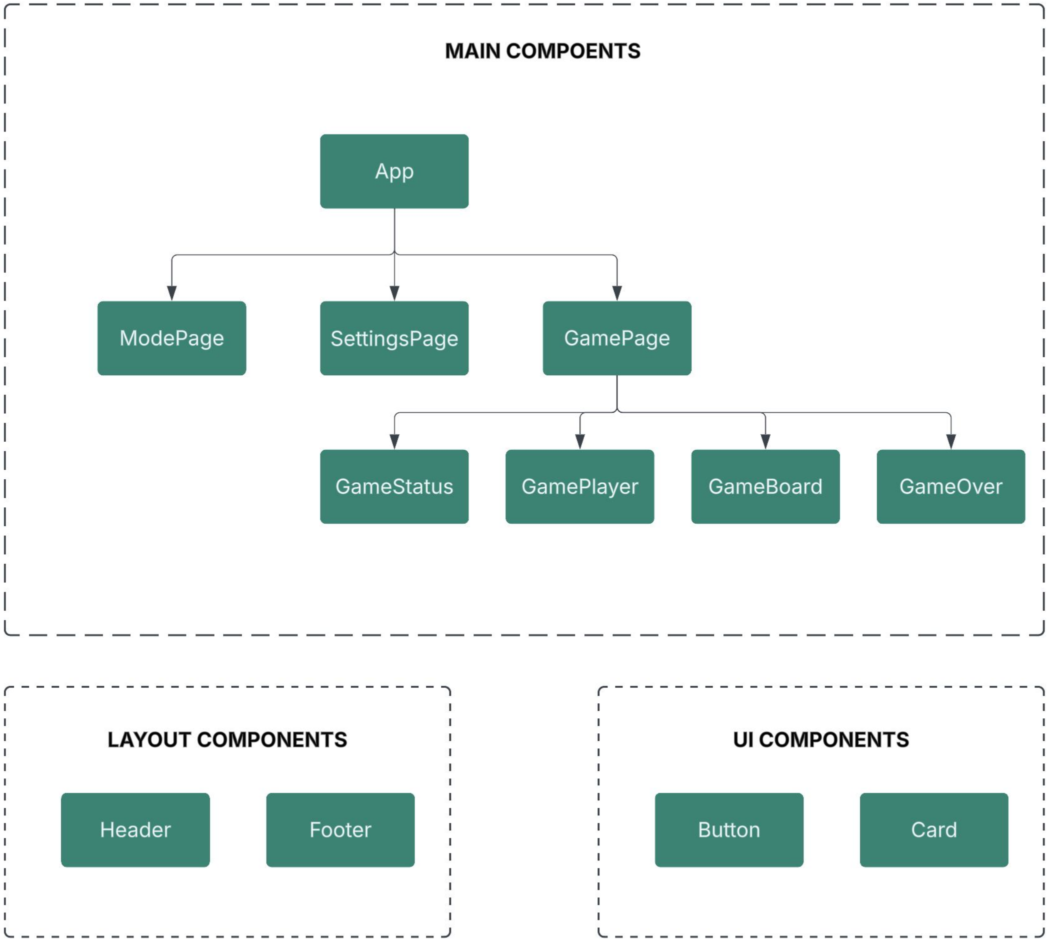
- **Components**
- Props
- Events
- Hooks e.g useState, useEffect, etc

Key Concepts: Components



- Most SPAs operate under the **component-based architecture**
- **Components are:**
  - **Building Blocks:** Each component represents a piece of the UI.
  - **Reusability** – Components can be reused across different parts of the application, reducing code duplication.
  - **Maintainability** – A well-structured component-based architecture makes debugging and scaling applications easier.
  - **Encapsulation** – Each component manages **its own state** and **logic**, ensuring separation of concerns.

Example: Our Tic-Tac-Toe Component Structure



MAIN COMPOENTS

App

ModePage

SettingsPage

GamePage

GameStatus

GamePlayer

GameBoard

GameOver

LAYOUT COMPONENTS

Header

Footer

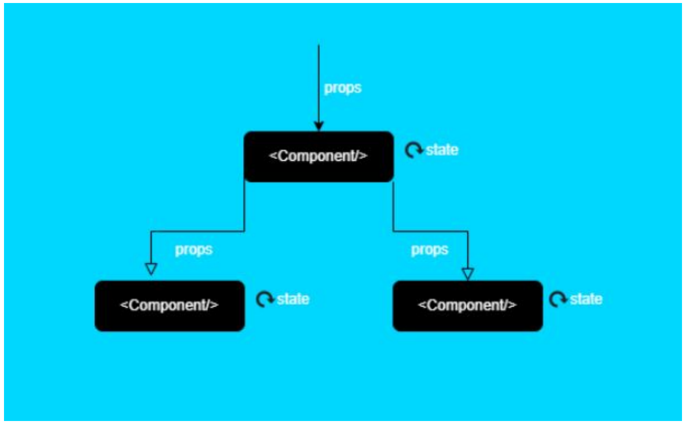
UI COMPONENTS

Button

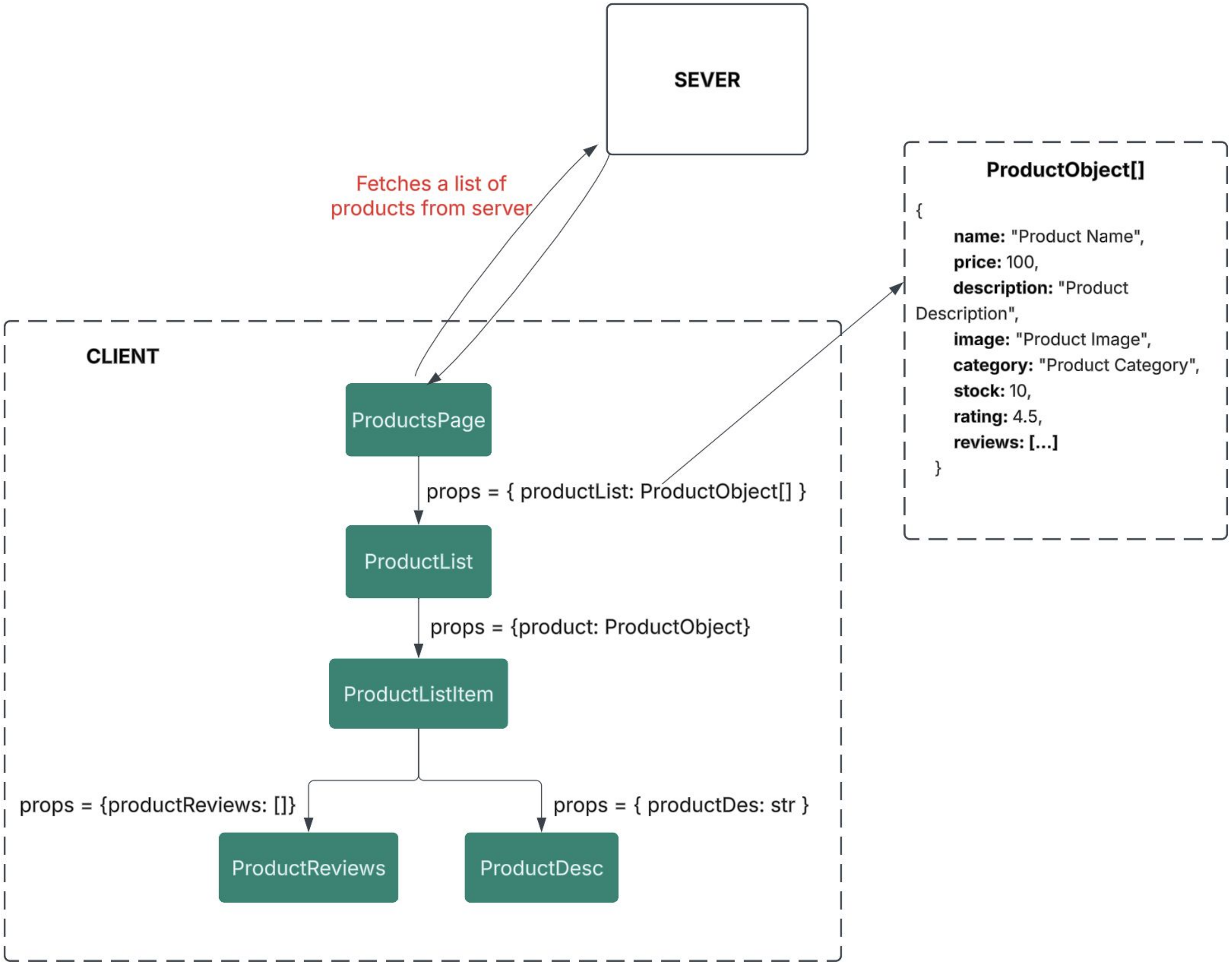
Card

Key Concepts: Props (Properties)

- Props are **read only data** passed from a **parent to child** components.
- Child components may be configurable and expect data from their parents.
- Props allow parent components to **share data** with their children.

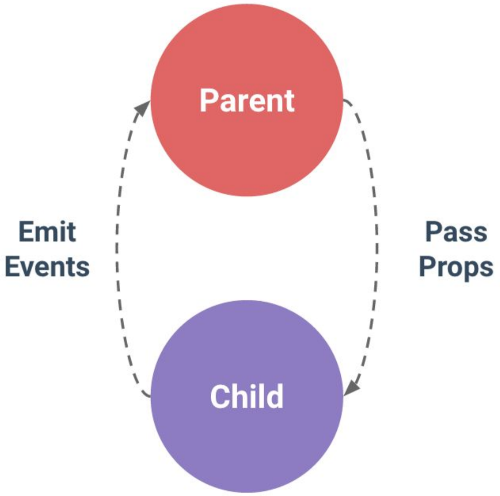


Example: Props (parent-to-child)


SEVER
Fetches a list of products from server
CLIENT
ProductsPage
props = { productList: ProductObject[] }
ProductList
props = {product: ProductObject}
ProductListItem
props = {productReviews: []}
props = { productDes: str }
ProductReviews
ProductDesc
ProductObject[]

```
{
    name: "Product Name",
    price: 100,
    description: "Product
Description",
    image: "Product Image",
    category: "Product Category",
    stock: 10,
    rating: 4.5,
    reviews: [...]
  }
```

Key Concepts: Events (child-to-parent)



- Events handle interactions like clicks, form submission, etc.
- **Child-to-parent communication:**
  - Components can communicate events (e.g., a square being clicked) back to parent components
- **Event handlers:**
  - Functions passed as props (often from the parent) that define what happens on user actions.

Key Concepts: Hooks

What are Hooks?

- JavaScript functions that let you **"hook into"** React features.
- There are a lot of built-in react hooks.
  - See more ⇒ https://react.dev/reference/react/hooks
- You can also create your own custom hooks.

Common React Hooks:

- **useState:** For managing state within a component.
- **useEffect:** To perform side effects such as data fetching from a server.
- More details on this during our practical session.

Tic-Tac-Toe Web Client

Project Features

- **Game Modes:** Single player (vs AI) or two-player
- **Customizable Players:** Set custom names for players
- **Difficulty Levels:** Easy, medium, hard AI opponents
- **Responsive Design:** Works on mobile and desktop
- **Audio Feedback:** Sound effects for game events
- **Game Statistics:** Track wins, losses, and draws
- **Persistent State:** Game state saved in **localStorage** for persistence across page reloads

Live DEMO: Tic-Tac-Toe Web Client

Summary

Summary

- SPAs provide a more dynamic, responsive user experience
- React simplifies building complex UIs through components
- Component-based architecture promotes reusability and maintainability
- State and props enable dynamic data flow
- Events handle user interactions seamlessly

Questions?