











-  [React JS and its advantages](#)
 -  [Table of Contents](#)
 - [Tutorial Video - Link](#)
 -  [What Is React, and Why Should You Use it?](#)
 -  [Key Features](#)
 -  [Core Concepts and Building Blocks](#)
 - [Components](#)
 - [JSX \(JavaScript XML\)](#)
 - [Virtual DOM](#)
 - [States and Props](#)
 - [Hooks](#)
 -  [Advantages of React JS](#)
 -  [React in the Real World](#)
 -  [Getting Your Hands Dirty, create your first React App](#)
 -  [Assessment Time!](#)
 -  [Resources](#)

React JS and its advantages

React, where components reign supreme, and the UI updates as if by magic. React JS stands out as a pivotal force in frontend development, captivating developers with its simplicity and power. This article is tailored for aspiring developers and anyone curious about diving into the realm of React JS. Here, this article aim to unravel the essence of React JS, its core concepts, and showcase its advantages in the world of web development. By the end, you'll not only grasp the fundamentals of React JS but also be equipped to explain its significance and application to others.

Table of Contents

- [What Is React, and Why Should You Use it?](#)
- [Key Features](#)
- [Core Concepts and Building Blocks](#)
 - [Components](#)
 - [JSX \(JavaScript XML\)](#)
 - [Virtual DOM](#)

- [States and Props](#)
- [Hooks](#)
- [Advantages of React JS](#)
- [React in the Real World](#)
- [Getting Your Hands Dirty, create your first React App](#)
- [Assessment Time!](#)
- [Resources](#)

Tutorial Video - [Link](#)



What Is React, and Why Should You Use it?

React JS is a dynamic JavaScript library used for building user interfaces, particularly known for its efficiency in rendering complex UIs with high performance. Unlike frameworks that often prescribe a complete solution, React focuses on one thing and does it exceptionally well—it aids in creating interactive, stateful & reusable UI components.

But why React? Because it's all about **simplicity** and **efficiency**. It helps you build complex UIs from small, isolated, reusable pieces of code called components. Plus, it's like having a superpower where your web app can update automatically without needing to refresh the page.

Reference: [React Official Site](#)



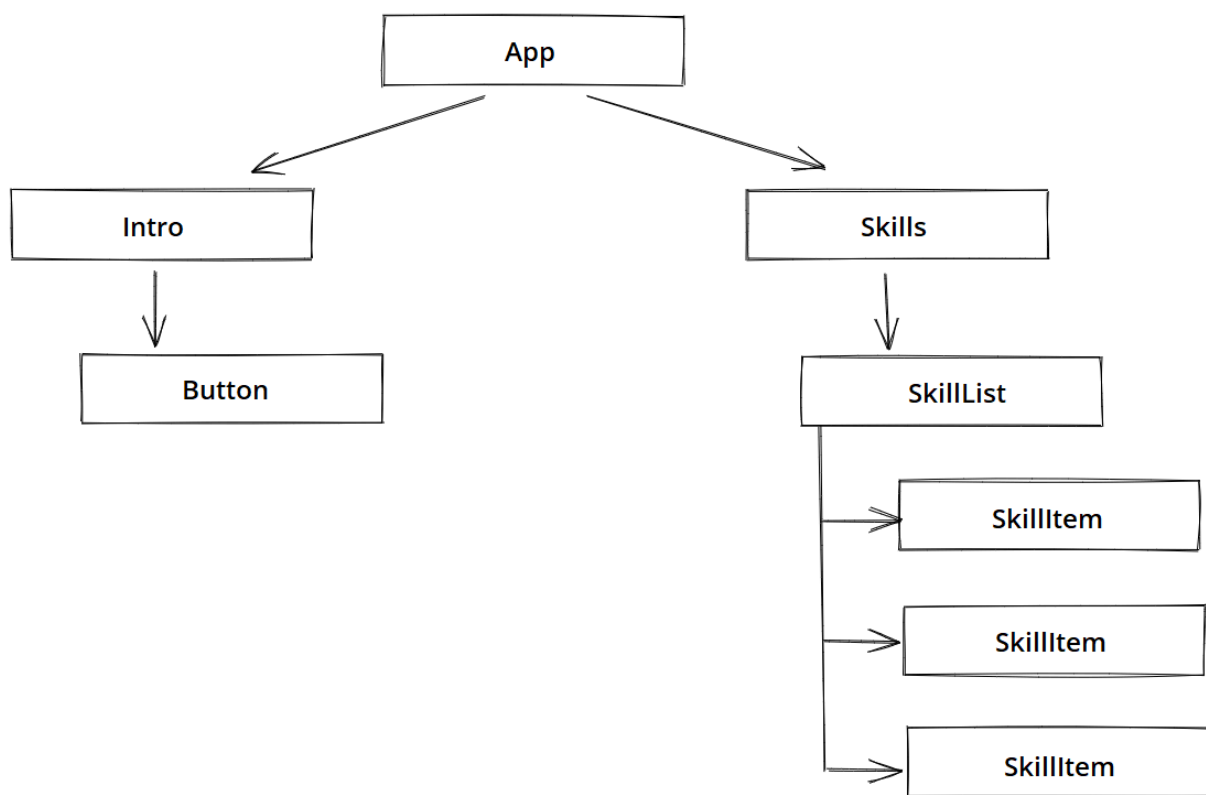
Key Features

- **Components:** Reusable building blocks of your application.
- **JSX:** Syntax extension for writing HTML-like structures within JavaScript.
- **Virtual DOM:** An in-memory representation of the UI for efficient updates.
- **Props:** Data passed down from parent to child components.
- **State:** Internal data managed within a component.
- **Hooks:** Functions that allow you to "hook into" React features without writing a full class component (introduced in React 16.8).

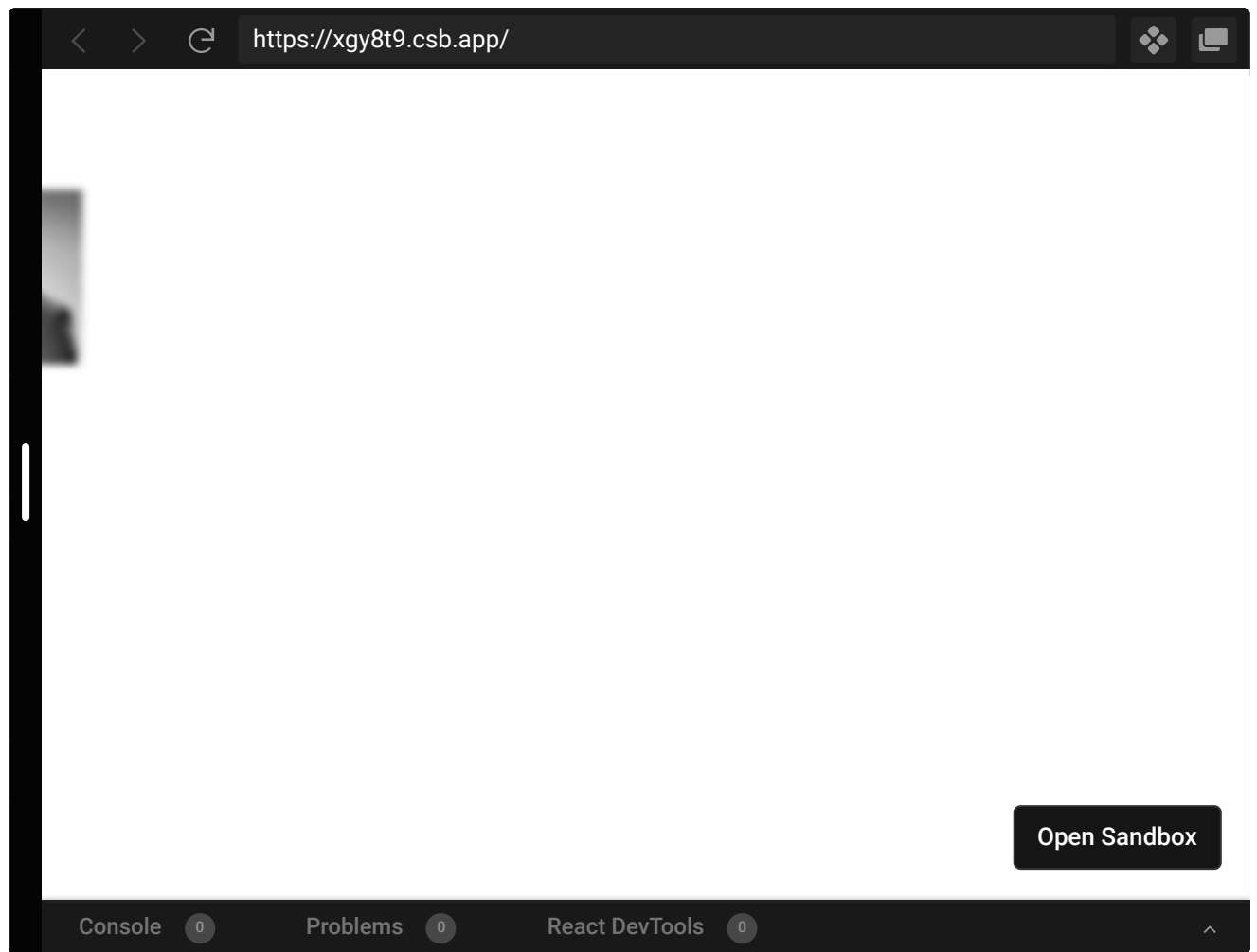
Core Concepts and Building Blocks

. Components

Think of them as custom, reusable HTML elements, each responsible for rendering a part of the UI. They can be as simple as a button or as complex as an entire form. Imagine a website header. You could create a **HeaderComponent** component that displays the logo, navigation bar, and search bar. This component would manage its own logic (e.g., handling search input) and render the relevant HTML structure.

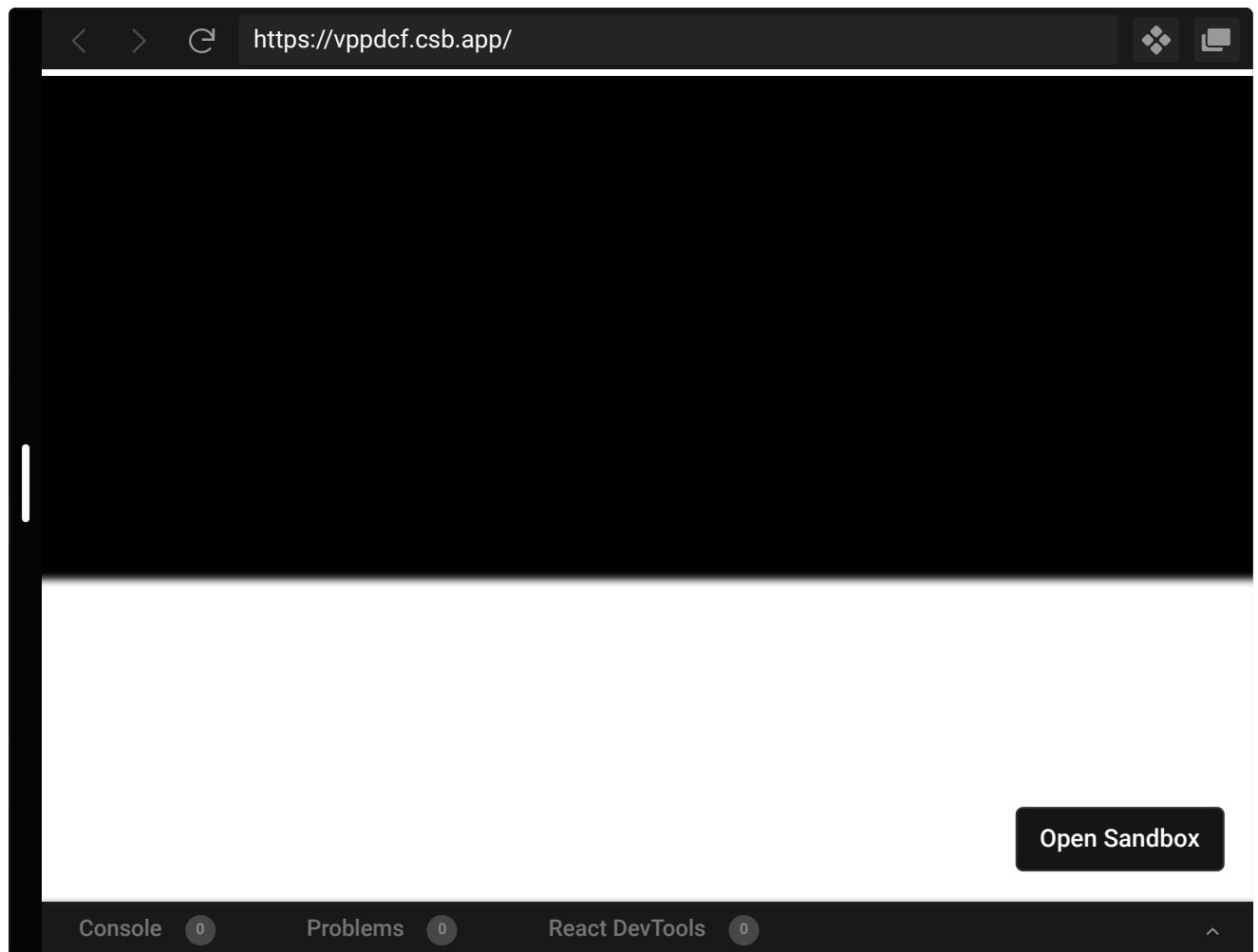


Example:



. JSX (JavaScript XML)

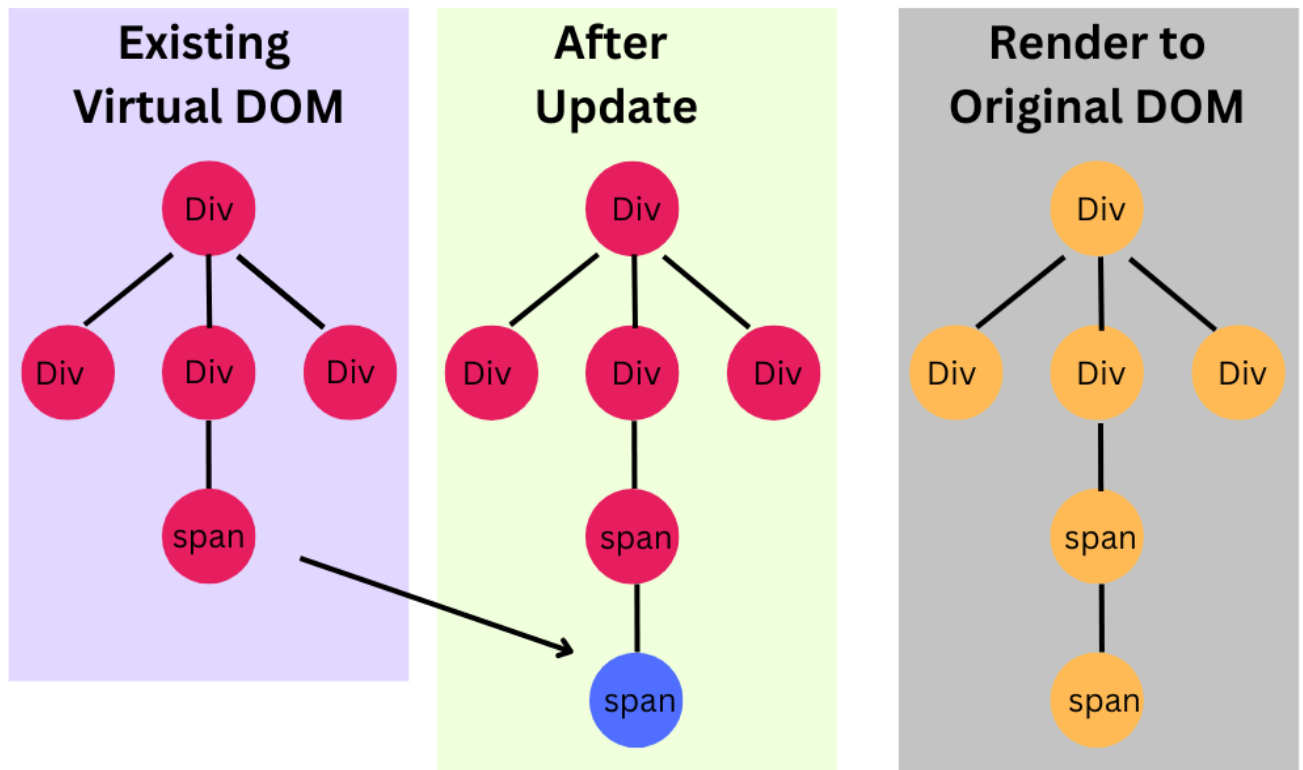
JSX is a syntax extension for JavaScript that allows you to write HTML-like structures within your JavaScript code. This improves readability and maintainability compared to writing pure JavaScript for UI elements. Example:



. Virtual DOM

The virtual DOM is a lightweight representation of the real DOM (Document Object Model) in the browser. When you make changes to your components, React compares the virtual DOM before and after the update. This diffing process identifies the minimal changes required in the real DOM, resulting in faster and more efficient updates.

Imagine the virtual DOM as a blueprint for your UI. React efficiently updates only the necessary parts of the real DOM based on changes in the blueprint.

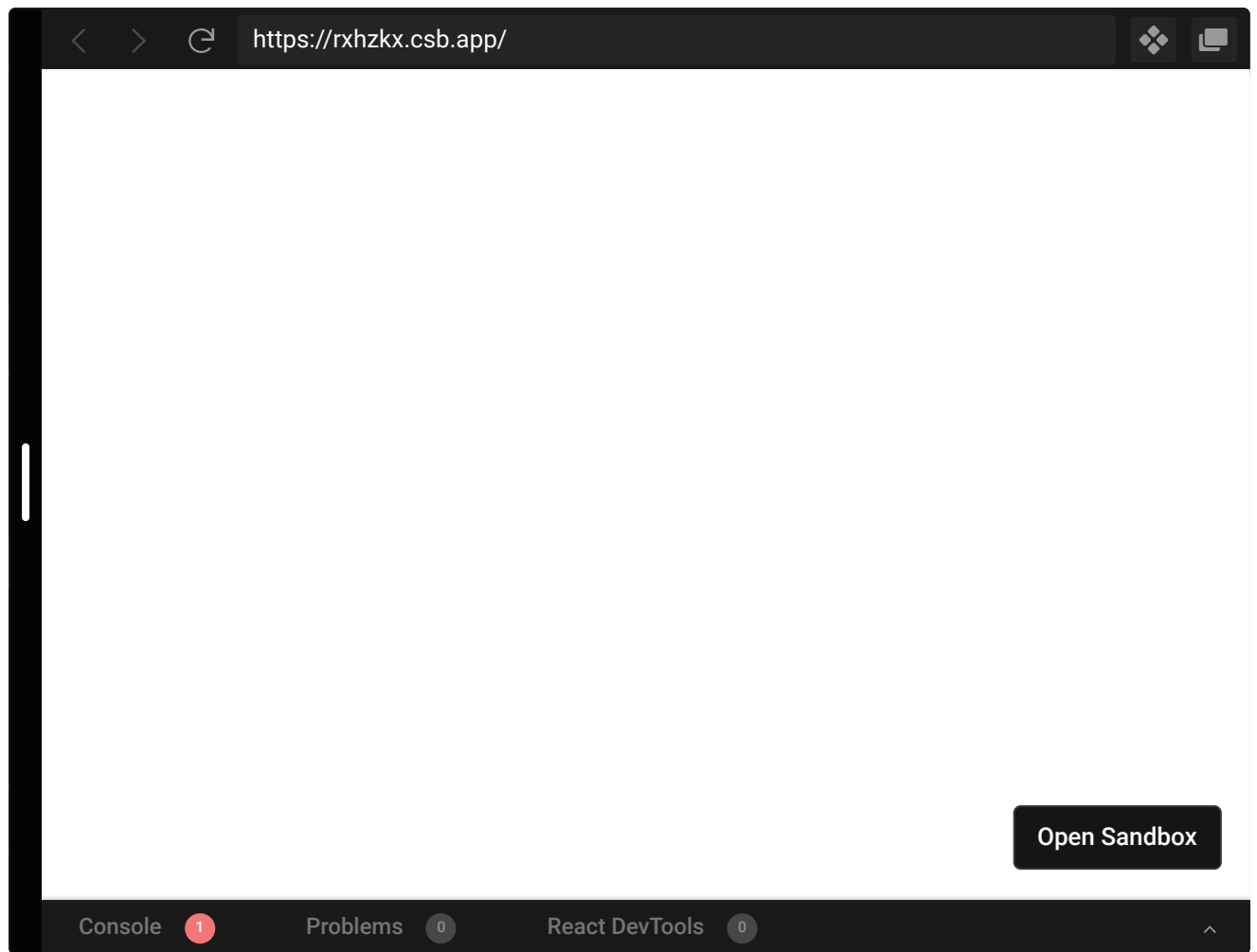


. States and Props

In React, components get to know each other and communicate via props (short for properties) and state.

- Props are like arguments to a function. They let you pass data from parent to child components, making your components reusable and dynamic.
- State is what allows your components to be interactive. It's basically a way to keep track of how the component's data changes over time, without having to reload the page. In simple words, State generally refers to application data or properties that need to be tracked.

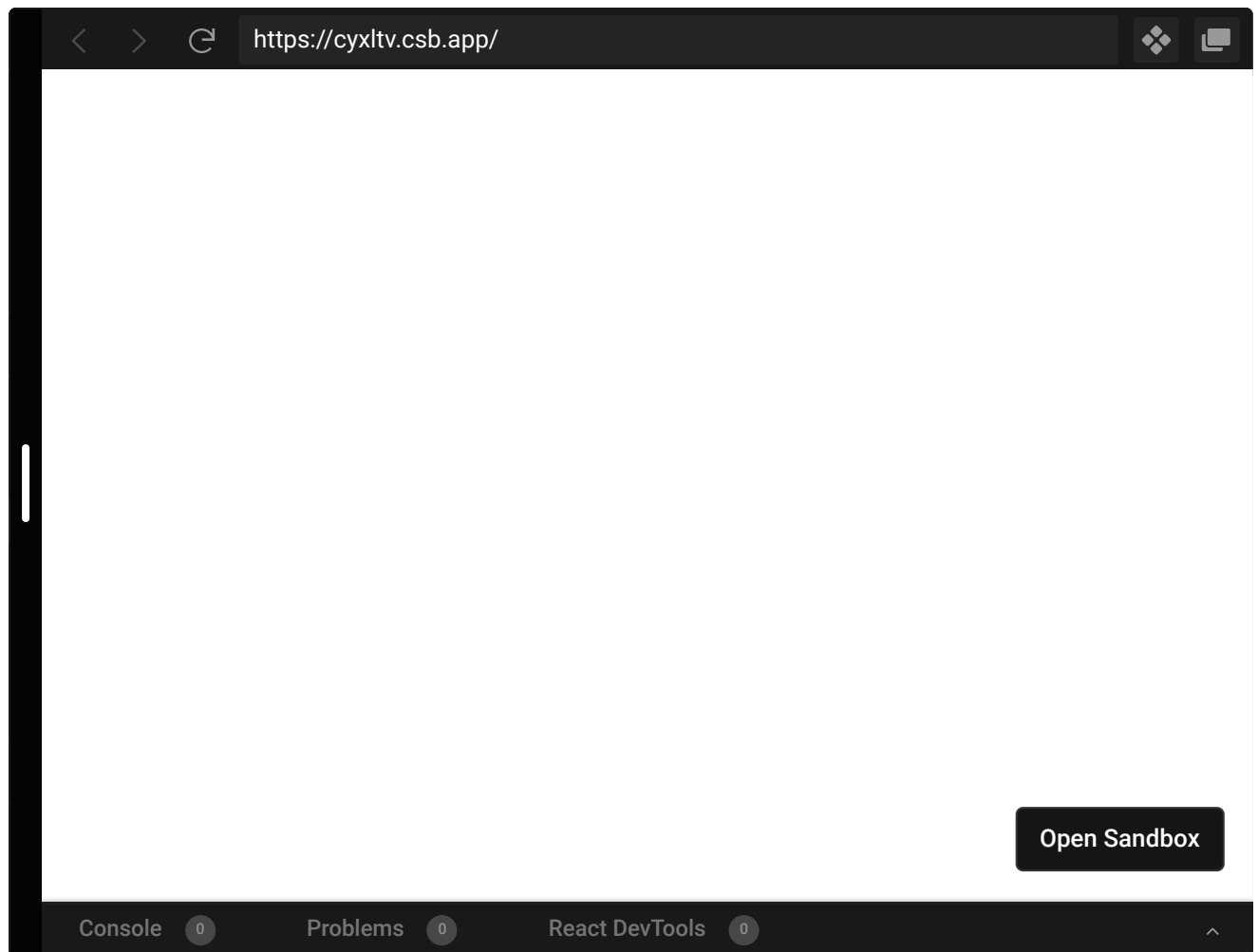
Example:



. Hooks

Introduced in React 16.8, hooks are functions that allow you to "hook into" React features without writing a full class component. They let you use state and other React features in functional components, making them more powerful and flexible.

Example:



Advantages of React JS

- **Component-Based:** Build encapsulated components that manage their state, then compose them to make complex UIs. Since components are reusable and isolated, you can focus on one piece of the puzzle at a time.
- **Performance:** React's virtual DOM and efficient diffing algorithm ensure minimal updates to the real DOM, resulting in high performance and a better user experience.
- **One-Way Data Binding:** React's unidirectional data flow (downwards) ensures that changes in child components don't affect their parents, making your code more predictable and easier to debug.
- **Ecosystem:** React has a vast and active [developer community](#) offering extensive support, libraries, and tools.
- **Reusability:** Components can be reused throughout your application, reducing code duplication and improving maintainability.
- **SEO Friendliness:** Techniques like server-side rendering can optimize React applications for search engines, making it appear in search results more effectively.



React in the Real World

React isn't just for small projects. It's used by some of the biggest names out there: Facebook (of course), Instagram, Airbnb, Netflix, and more, these applications benefit from React's component-based architecture and efficient UI updates to deliver rich and interactive user experiences. With the ecosystem around React, like Redux for state management and React Router for navigation, you can build pretty much anything you can imagine.



Getting Your Hands Dirty, create your first React App

The best way to learn React is by building stuff. Seriously, just start a project. Anything. A todo list, a blog, a portfolio... It doesn't matter. The key is to practice, make mistakes, and learn from them. Start with this [Tic-Tac-Toe game](#) from React documentation. And here's a quick guide to get you started:

Pre-requisite: Make sure you have a code IDE and Node.js installed on your machine. If not, you can download it from the [official NodeJS website](#). Ensure Node Version Manager is installed on your machine by running `nvm -v`. Consider installing nvm from [this link](#). Download the `nvm-setup.exe` file and install it by following the installation wizard. Now, after having `nvm` installed, go on updating Node version by running, `nvm install lts`. This will install latest LTS version. To use latest features of npm and react-app, we need to keep `npm` updated. Run the following command to update `npm install -g npm`.

1. **Create a New React App:** You can use [Create React App](#) to set up a new React project with a single command. Run the following in your terminal:

```
npx create-react-app my-app  
cd my-app  
npm start
```

This will create a new React project in a folder called `my-app`, start a development server, and open your app in the browser.

2. **Explore the Project Structure:** Take a look at the files and folders created by Create React App. You'll find the `src` folder, where you'll spend most of your time writing code. The `public` folder contains the `index.html` file that serves as the entry point for your app.
3. **Start Coding:** Open the project in your favorite code editor and start coding. Create new components, add styles, and experiment with React features. Remember, practice makes perfect!
4. **Learn by Doing:** As you build your app, refer to the [React documentation](#) and other resources to deepen your understanding of React concepts and best practices.



Assessment Time!

- Which of the following is NOT a core concept of React JS?
 - ☐ Components
 - ☒ Templates
 - ☐ JSX
 - ☐ Virtual DOM
- What is the main benefit of using components in React?
 - ☒ Faster load times (curious how?)
 - ☐ Improved SEO
 - ☒ Reusability
 - ☐ Better security
- What is the purpose of props in React?
 - ☐ To manage internal component state
 - ☒ To pass data from parent to child components
 - ☐ To handle user interactions
 - ☐ To style components
- What is the virtual DOM in React?
 - ☐ A real-time representation of the UI in the browser
 - ☒ A lightweight copy of the real DOM for efficient updates
 - ☐ A tool for debugging React components
 - ☐ A way to store component data



Resources

- [React Documentation](#)
- [Create React App](#)
- [React w3schools](#)
- [React Tutorial: Tic-Tac-Toe](#)
- [React Community](#)
- [React Router](#)
- [Redux](#)
- [React Patterns](#)

Happy coding! 🚀 ~ [Abhishek Jadhav](#)